NEU Für Einsteiger und aktuelle Programmierer

November-2025

C++ & Python für Einsteiger

Entfessle die Kraft des Codes

Top-Tutorials und Guides zur Verbesserung Ihrer Programmier-kenntnisse.

Meister Python

Programmieren mit der aktuellen Python-Version.

Mit C++

Lernen Sie die leistungsstärkste Programmier-sprache der Welt kennen.

FREI
Code-Portal
50 Programme
20,000 Codezeilen
Weitere Details finden Sie im Innenteil.

Arbeiten mit Code

Unsere Tutorials führen Sie durch die komplexe Welt der Programmierung.

Der nächste Schritt mit Ihrem Code

Entdecken Sie, wie Sie Grafi ken, Abenteuerspiele, Animationen und mehr erstellen.

Jetzt auf Readly zum Lesen verfügbar

Von Ihrem vertrauenswürdigen Verlag Papercut

Klicken Sie auf unseren sicheren Link, um jetzt zu lesen!

https://bit.ly/4310Fgk

www.pclpublikationen.com

C++&Python für Einsteiger

„Jeder Narr kann Code schreiben, den ein Computer verstehen kann. Gute Programmierer schreiben Code, den Menschen verstehen können."

– *Martin Fowler (Softwareentwickler & Autor)*

Unsere moderne digitale Welt besteht aus Code. Das Internet, unsere Telefone, Fernseher, Spielekonsolen, Banken und praktisch alles, das in irgendeiner Weise mit der weiten Welt verbunden ist, basiert auf einer Form von Code, der alles antreibt und den Dingen befiehlt, wie sie sich zu verhalten haben und was sie ausführen sollen.

Dieser Code kann in verschiedenen Programmiersprachen kommen. Einige bieten bessere Stabilität, Geschwindigkeit und komplexe Algorithmen, während andere für den Einsatz mit minimalen Systemressourcen konzipiert sind. Die gesamte Weltraum- und Satellitenkommunikationsbranche beruht auf intelligenter Programmierung. Kurz gesagt, Code umgibt und hilft uns in fast allen Bereichen unseres Lebens.

Der Zeitpunkt ist daher perfekt, um das Programmieren zu lernen und zu sehen, wie ein paar einfache Codezeilen etwas Erstaunliches schaffen und mit Ihnen und Ihren Technologien interagieren können. Python und C++ sind zwei der beliebtesten und leistungsfähigsten Sprachen, die es zu erlernen gibt. Mithilfe dieser Ausgabe und etwas Kreativität werden Sie schon bald erstaunlichen Code erstellen können.

Blättern Sie weiter und legen Sie los!

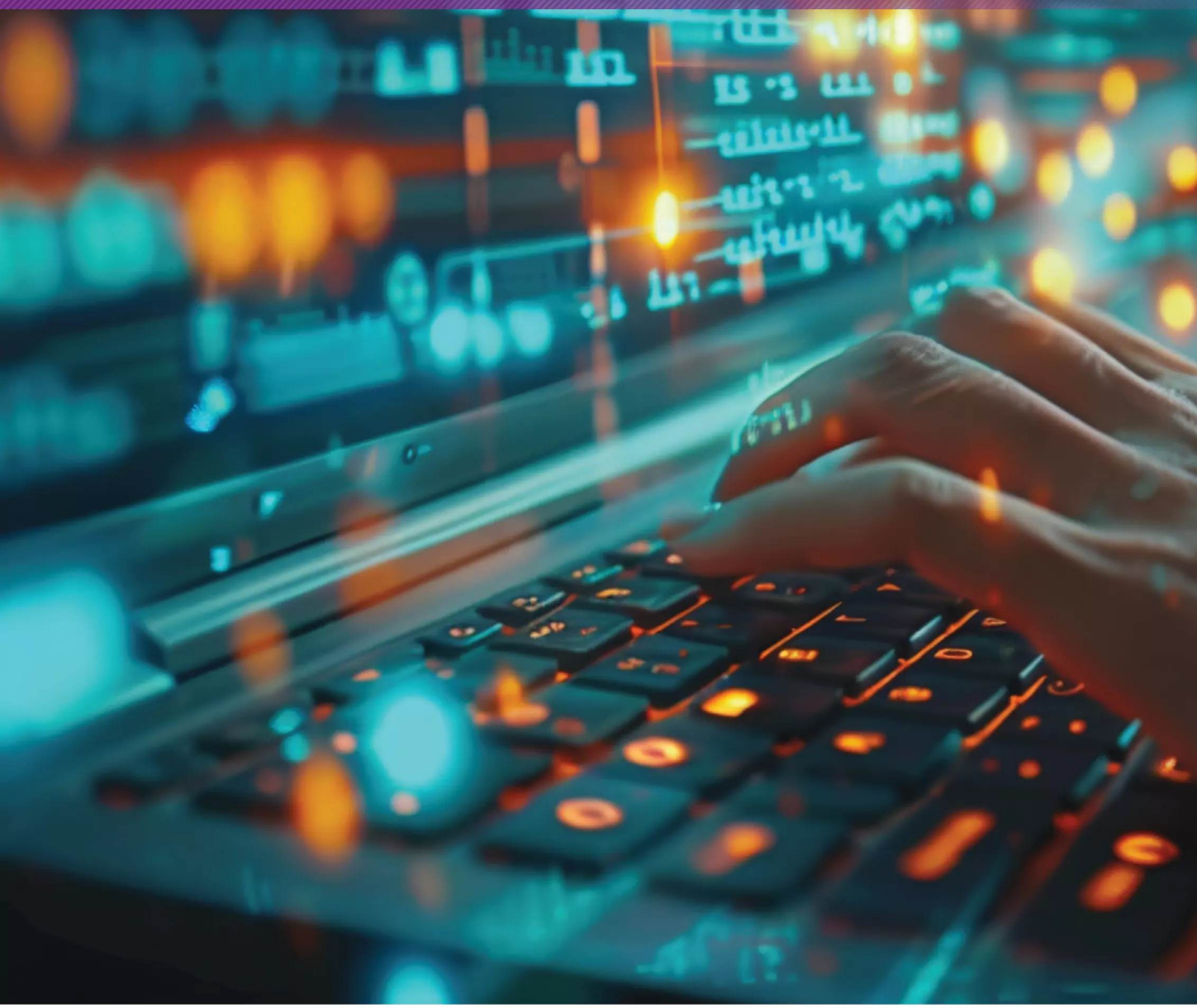
Inhalt

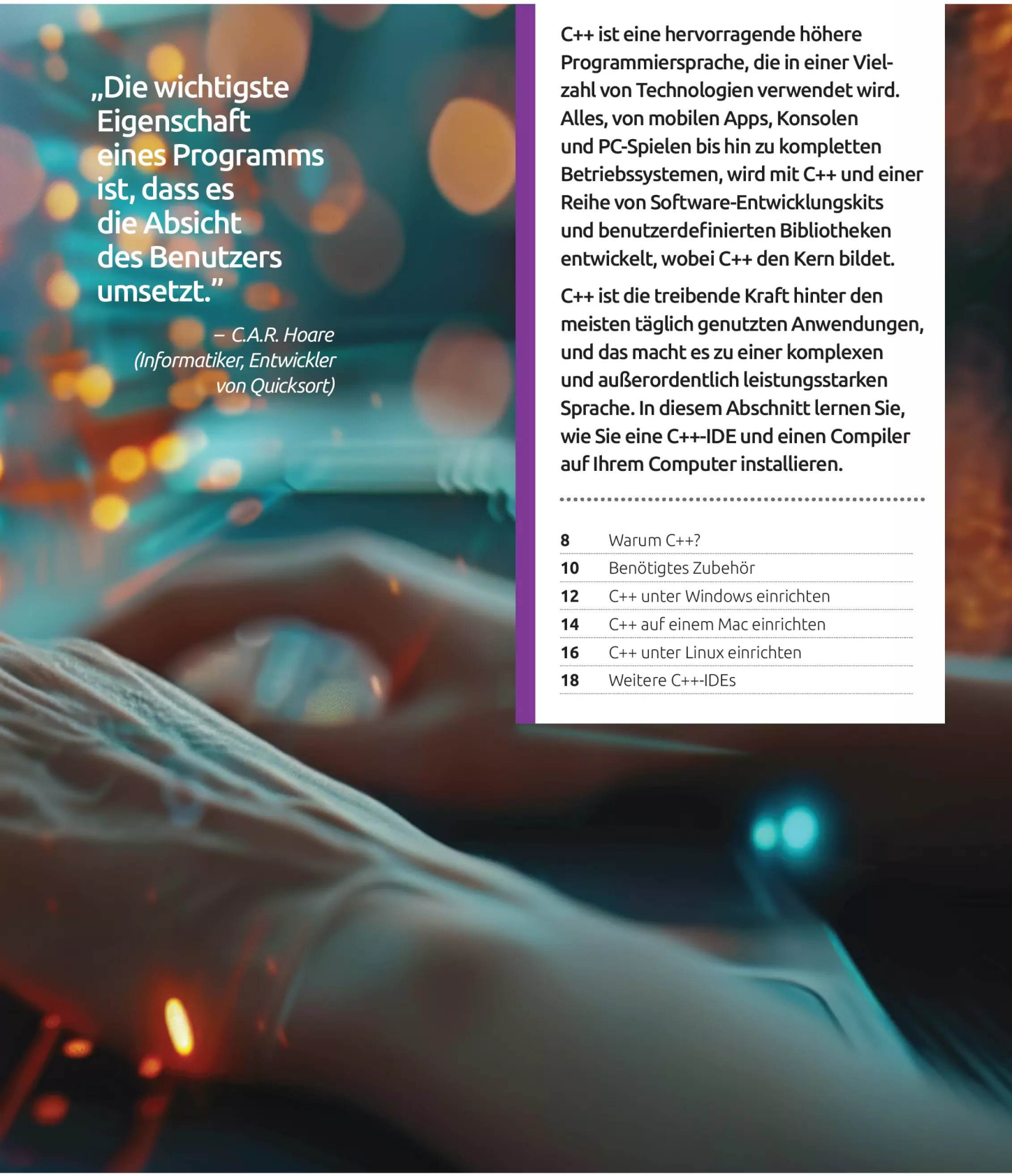
C++ & Python für Einsteiger

Code Portal
50 Python-Programme
Mehr als 20.000 Zeilen Code
Meistern Sie Python mithilfe unseres fantastischen Code-Portals, das Codes für Spiele, Tools und mehr enthält.
https://pclpublications.com/exclusives




Willkommen bei C++





„Die wichtigste Eigenschaft eines Programms ist, dass es die Absicht des Benutzers umsetzt."

– C.A.R. Hoare (Informatiker, Entwickler von Quicksort)

C++ ist eine hervorragende höhere Programmiersprache, die in einer Vielzahl von Technologien verwendet wird. Alles, von mobilen Apps, Konsolen und PC-Spielen bis hin zu kompletten Betriebssystemen, wird mit C++ und einer Reihe von Software-Entwicklungskits und benutzerdefinierten Bibliotheken entwickelt, wobei C++ den Kern bildet.

C++ ist die treibende Kraft hinter den meisten täglich genutzten Anwendungen, und das macht es zu einer komplexen und außerordentlich leistungsstarken Sprache. In diesem Abschnitt lernen Sie, wie Sie eine C++-IDE und einen Compiler auf Ihrem Computer installieren.

Warum C++?

C++ ist eine der beliebtesten derzeit verfügbaren Programmiersprachen. Ursprünglich C with Classes, also C mit Klassen, genannt, wurde die Sprache 1983 in C++ umbenannt. Sie ist eine Erweiterung der ursprünglichen C-Sprache und ist eine universell einsetzbare, objektorientierte (OOP)-Umgebung.

ÜBERALL ZU FINDEN

Aufgrund der Komplexität sowie Leistungsstärke von C++ wird die Sprache häufig für die Entwicklung von Spielen, Programmen, Gerätetreibern und sogar ganzer Betriebssysteme verwendet.

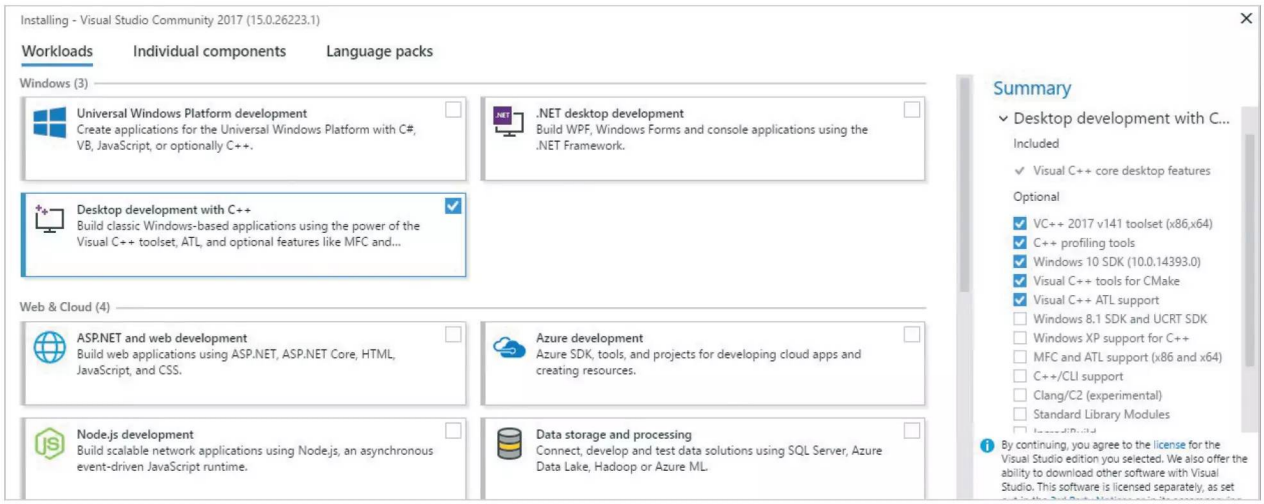
Im Jahr 1979, dem Beginn des goldenen Zeitalters des Home-Computings, hat der dänische Informatiker Bjarne Stroustrup während seiner Doktorarbeit C++ bzw. „C mit Klassen" entwickelt. Stroustrups Plan bestand darin, die ursprüngliche C-Sprache zu erweitern, die seit den frühen 70er Jahren weit verbreitet war.

C++ war in den 80er Jahren unter den Entwicklern beliebt, da diese Sprache eine verständlichere Umgebung bot und darüber hinaus mit der ursprünglichen C-Sprache zu 99 % kompatibel war. Dies bedeutete, dass sie außerhalb der gängigen Datenverarbeitungslabore und von Personen, die keinen Zugriff auf Großrechner und Rechenzentren hatten, verwendet werden konnte.

Die Auswirkungen von C++ in der digitalen Welt sind immens. Viele Programme, Apps, Spiele und sogar Betriebssysteme wurden mit C++ programmiert. Z. B. wurden alle wichtigen Anwendungen von Adobe wie Photoshop, InDesign usw. in C++ entwickelt. Auch der Browser, mit dem Sie im Internet surfen, Windows 10, Microsoft Office und das Backbone der Google-Suchmaschine sind in C++ geschrieben. Apples macOS ist weitgehend in C++ geschrieben (einige andere Sprachen

In C++ geschriebener Code ist schneller als Python-Code.

```
148     checkCuda( cudaMallocHost((void**)&callResult,     OPT_SZ) );
149     checkCuda( cudaMallocHost((void**)&putResult,      OPT_SZ) );
150     checkCuda( cudaMallocHost((void**)&stockPrice,     OPT_SZ) );
151     checkCuda( cudaMallocHost((void**)&optionStrike,   OPT_SZ) );
152     checkCuda( cudaMallocHost((void**)&optionYears,    OPT_SZ) );
153     checkCuda( cudaMalloc    ((void**)&d_callResult,   OPT_SZ) );
154     checkCuda( cudaMalloc    ((void**)&d_putResult,    OPT_SZ) );
155     checkCuda( cudaMalloc    ((void**)&d_stockPrice,   OPT_SZ) );
156     checkCuda( cudaMalloc    ((void**)&d_optionStrike, OPT_SZ) );
157     checkCuda( cudaMalloc    ((void**)&d_optionYears,  OPT_SZ) );
158 #else
159     callResult   = (float*)malloc(OPT_SZ);
160     putResult    = (float*)malloc(OPT_SZ);
161     stockPrice   = (float*)malloc(OPT_SZ);
162     optionStrike = (float*)malloc(OPT_SZ);
163     optionYears  = (float*)malloc(OPT_SZ);
164 #endif
165
166     initOptions(OPT_N, stockPrice, optionStrike, optionYears);
167
168     int blockDim = 128; // blockDim, gridDim ignored by host code
169     int gridDim  = std::min<int>(1024, (OPT_N + blockDim - 1) / blockDim);
170
171     StartTimer();
172
173 #ifdef __CUDACC__
174     printf("Running Device Version...\n");
175     checkCuda( cudaMemcpy(d_stockPrice,   stockPrice,   OPT_SZ, cudaMemcpyHostToDevice) );
176     checkCuda( cudaMemcpy(d_optionStrike, optionStrike, OPT_SZ, cudaMemcpyHostToDevice) );
177     checkCuda( cudaMemcpy(d_optionYears,  optionYears,  OPT_SZ, cudaMemcpyHostToDevice) );
178
179     BlackScholes_kernel<<<gridDim, blockDim>>>(d_callResult, d_putResult, d_stockPrice,
180                                                d_optionStrike, d_optionYears, RISKFREE,
181                                                VOLATILITY, OPT_N);
182
183     checkCudaErrors();
```



Microsoft Visual Studio ist eine tolle, kostenlose Umgebung, in der Sie C++ lernen können.

werden je nach Funktion gemischt), und die NASA, SpaceX und sogar CERN verwenden C++ für verschiedene Anwendungen, Programme, Steuerelemente und zahlreiche andere Computeraufgaben.

C++ ist äußerst effizient, bietet generell eine gute Leistung und ist eine einfachere Ergänzung der C-Kernsprache. Dieses höhere Leistungsniveau macht sie im Vergleich zu anderen Sprachen wie Python, BASIC usw. zu einer idealen Entwicklungsumgebung für die moderne Informatik, weshalb sie auch von den oben genannten Unternehmen so häufig verwendet wird.

Python ist zwar eine hervorragende Programmiersprache, allerdings bietet C++ dem Entwickler ein weitaus größeres Angebot an Programmieroptionen. Wenn Sie C++ beherrschen, können Sie selbst Code für Microsoft, Apple usw. entwickeln. Im Allgemeinen erhalten C++-Entwickler ein höheres Gehalt als Programmierer anderer Sprachen. Aufgrund ihrer Vielseitigkeit kann der C++-Programmierer zwischen Jobs und Unternehmen wechseln, ohne dass er spezielle Dinge neu lernen muss. Python ist für den Einstieg jedoch eine einfachere Sprache. Wenn Sie mit der Programmierung noch nicht vertraut sind, empfehlen wir Ihnen, mit Python zu beginnen und sich für einige Zeit mit der Programmierstruktur und den vielen Möglichkeiten auseinanderzusetzen, mit denen Sie eine Lösung für ein durch Programmierung entstandenes Problem finden. Wenn Sie mit einer Hand hinter dem Rücken gebunden ein Python-Programm erstellen können, sind Sie für C++ bereit. Natürlich können Sie auch direkt mit C++ beginnen, wenn Sie sich der Aufgabe gewachsen fühlen.

Die Anwendung von C++ ist so einfach wie Python. Sie benötigen lediglich die richtigen Tools, um mit dem Computer in C++ zu kommunizieren. Eine C++-IDE ist kostenlos, selbst das immens leistungsstarke Visual Studio von Microsoft kann kostenlos heruntergeladen und genutzt werden. Sie können C++ von jedem Betriebssystem aus nutzen, z. B. macOS, Linux, Windows oder mobile Plattformen.

Die Frage „Warum C++?“ lässt sich also damit beantworten, dass C++ schnell und effizient ist und von den meisten Anwendungen entwickelt wird, die wir regelmäßig verwenden. C++ ist innovativ und eine fantastische Programmiersprache.

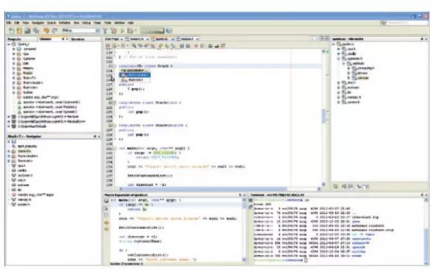
Selbst das von Ihnen verwendete Betriebssystem ist in C++ geschrieben.

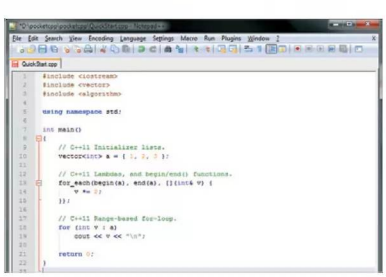
Benötigtes Zubehör

Das Lernen von C++ erfordert weder große Investitionen noch eine komplette EDV-Einrichtung. Sie benötigen lediglich einen relativ modernen Computer, alles andere ist kostenlos erhältlich.

C++ – SET-UPS

Da die meisten, wenn nicht sogar alle Betriebssysteme auf C++ basieren, liegt es auf der Hand, dass Sie unabhängig von Ihrem verwendeten Betriebssystem das Programmieren in C++ lernen können.





COMPUTER

Sofern Sie den C++-Code nicht per Hand auf ein Blatt Papier schreiben möchten (was bei älteren Programmierern üblich war), ist ein Computer ein absolutes Muss. PC-Benutzer können alle aktuellen Linux-Distributionen oder Windows-Betriebssysteme verwenden und Mac-Benutzer das neueste macOS.

IDE

C++-Code wird über eine IDE, einer Entwicklungsumgebung, eingegeben und ausgeführt. Viele IDEs werden mit Erweiterungen und Plugins geliefert, die zu einer verbesserten Funktionsweise beitragen oder zusätzliche Funktionen bieten. Eine IDE bietet oftmals Verbesserungen für das jeweilige Kernbetriebssystem.

COMPILER

Ein Compiler ist ein Programm, das die C++-Sprache in eine Binärdatei umwandelt, die der Computer verstehen kann. Einige IDEs, aber nicht alle, werden mit einem integrierten Compiler geliefert. Code::Blocks ist unsere Lieblings-IDE, die bereits einen C++-Compiler als Teil des Pakets enthält – dazu in Kürze mehr.

TEXTEDITOR

Einige Programmierer ziehen es vor, ihren C++-Code mit einem Texteditor zusammenzustellen, bevor er über einen Compiler ausgeführt wird. Sie können praktisch jeden Texteditor zum Schreiben von Code nehmen; speichern Sie diesen einfach mit einer .cpp-Erweiterung. Notepad++ ist einer der besten erhältlichen Texteditoren für Code.

INTERNETZUGANG

Es ist zwar möglich, das Programmieren auf einem Computer ohne Internetzugriff zu erlernen, es ist aber sehr schwierig, da über das Internet die entsprechende Software installiert und auf dem neuesten Stand gehalten wird, Extras oder Erweiterungen installiert werden und beim Programmieren nach Hilfe gesucht werden kann.

ZEIT UND GEDULD

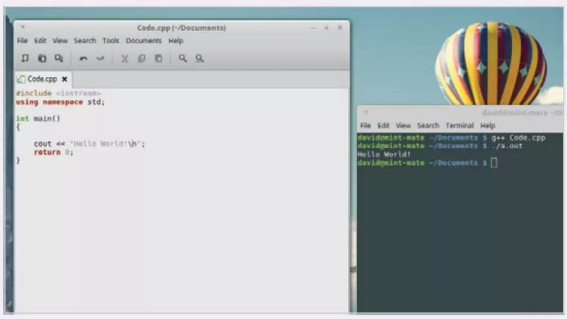
Für das Erlernen des Programmierens in C++ müssen Sie viel Zeit einplanen. Es wird leider nicht über Nacht oder in einer Woche passieren. Ein guter C++-Programmierer hat viele Jahre damit verbracht, sein Handwerk zu lernen. Seien Sie also geduldig, fangen Sie klein an und lernen Sie immer dazu.

OS-SPEZIFISCHE ANFORDERUNGEN

C++ funktioniert auf jedem Betriebssystem, es kann aber für den Beginner verwirrend sein, alle erforderlichen Elemente zusammenzubringen. Hier sind einige Besonderheiten der Betriebssysteme in Hinblick auf C++.

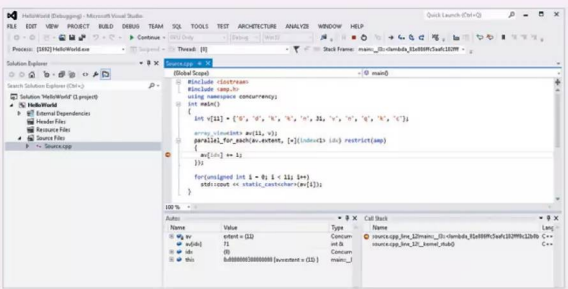
LINUX

Linux-Benutzer haben das Glück, dass in ihrem Betriebssystem bereits ein Compiler und ein Texteditor integriert sind. C++-Code kann in jedem Texteditor eingegeben werden. Wenn dieser mit der Erweiterung .cpp gespeichert wurde, kann er mit g++ kompiliert werden.



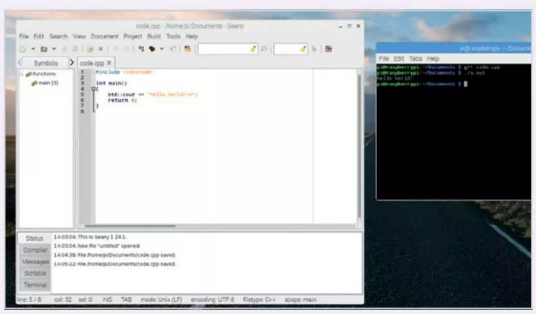
WINDOWS

Wie bereits erwähnt, ist Microsoft Visual Studio eine gute IDE. Code::Blocks bietet jedoch eine bessere IDE und einen besseren Compiler und wird zweimal jährlich mit einer neuen Version aktualisiert. Ansonsten können Windows-Benutzer ihren Code in Notepad++ eingeben und dann mit MinGW kompilieren, den auch Code::Blocks verwendet.



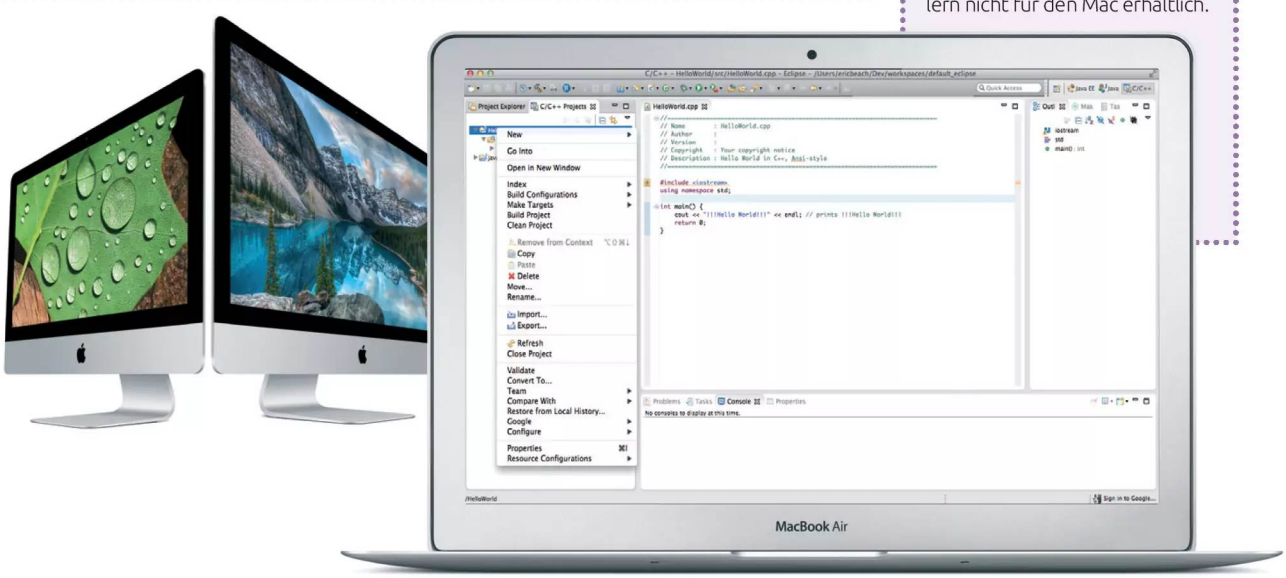
RASPBERRY PI

Raspbian ist das Betriebssystem des Raspberry Pi und basiert auf Linux. Daher kann der Code mit einem Texteditor geschrieben und dann, wie in jeder anderen Linux-Distribution, mit g++ kompiliert werden.



MAC

Mac-Besitzer müssen Xcode herunterladen und installieren, um ihren C++-Code nativ kompilieren zu können. Weitere Optionen für macOS sind Eclipse, Netbeans und Code::Blocks. Hinweis: Die neueste Version von Code::Blocks ist aufgrund eines Mangels an Mac-Entwicklern nicht für den Mac erhältlich.



C++ unter Windows einrichten

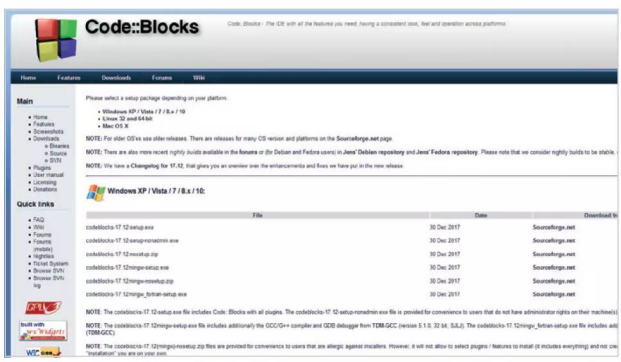
Windows-Benutzer haben bei der Programmierung in C++ eine große Auswahl. Es gibt zahlreiche IDEs und Compiler, einschließlich Visual Studio von Microsoft. Unserer Meinung nach ist Code::Blocks jedoch die beste C++-IDE für den Einstieg.

CODE::BLOCKS

Code::Blocks ist eine kostenlose C++-, C- und Fortran-IDE, die viele Funktionen hat und mit Plugins leicht erweitert werden kann. Sie ist einfach zu bedienen, wird mit einem Compiler geliefert und hat zusätzlich eine dynamische Gemeinschaft.

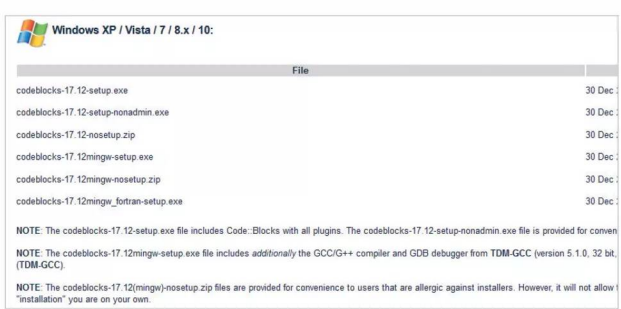
SCHRITT 1

Beginnen Sie, indem Sie die Download-Seite von Code::Blocks auf **www.codeblocks.org/downloads besuchen**. Klicken Sie dort auf den Link „Download the binary release", um zur neuesten herunterladbaren Version für Windows zu gelangen.



SCHRITT 2

Wie Sie sehen, stehen mehrere Windows-Versionen zur Verfügung. Die Version, die Sie herunterladen möchten, hat am Ende der aktuellen Versionsnummer mingw-setup.exe. Als diese Ausgabe verfasst wurde, war dies codeblocks-17.12mingw-setup.exe. Der Unterschied ist, dass die mingw-setup-Version einen C++-Compiler und -Debugger von TDM-GCC (eine Compiler-Suite) enthält.

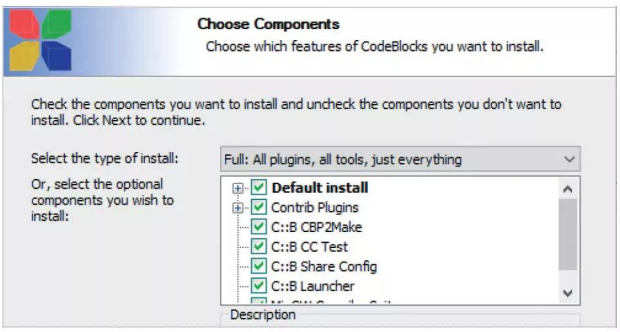


SCHRITT 3

Wenn Sie die Datei gefunden haben, klicken Sie am Zeilenende auf den Link „Sourceforge.net". Daraufhin erscheint ein Fenster zum Herunterladen von Benachrichtigungen. Klicken Sie auf „Save File", um den Download zu starten und die ausführbare Datei auf Ihrem PC zu speichern. Suchen Sie das heruntergeladene Installationsprogramm für Code::Blocks und starten Sie es per Doppelklick. Folgen Sie den Bildschirmanweisungen, um die Installation zu starten.

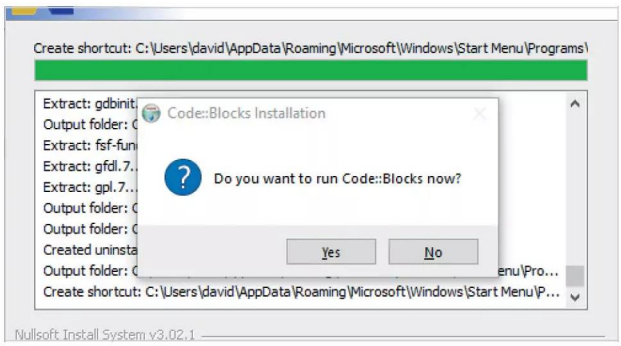
SCHRITT 4

Nachdem Sie den Lizenzbedingungen zugestimmt haben, erhalten Sie eine Auswahl an Installationsoptionen. Sie können sich für eine kleinere Installation entscheiden, wobei einige der Komponenten ausgelassen werden. Wir empfehlen jedoch, dass Sie sich für die vollständige Option, also Full, entscheiden.



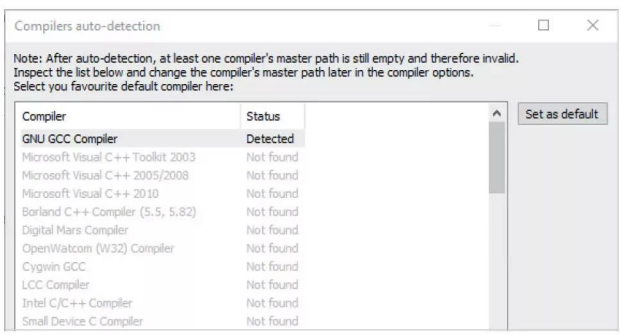
SCHRITT 5

Wählen Sie nun den Installationsort für die Code::Blocks-Dateien. Sie haben die Wahl, aber die Standardeinstellung ist in der Regel ausreichend (es sei denn, Sie haben spezielle Anforderungen). Wenn Sie auf „Next“ klicken, beginnt die Installation. Nach Beendigung werden Sie gefragt, ob Sie Code::Blocks jetzt starten möchten. Klicken Sie auf „Yes“.



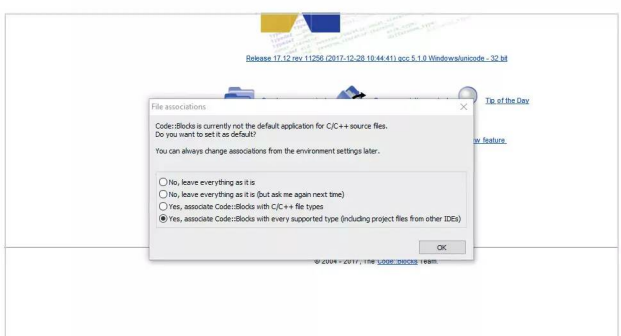
SCHRITT 6

Beim ersten Hochfahren von Code::Blocks wird eine automatische Erkennung für alle C++-Compiler ausgeführt, die Sie evtl. bereits auf Ihrem System installiert haben. Wenn Sie noch keine haben, wählen Sie die erste erkannte Option, GNU GCC Compiler, und klicken Sie auf die Default-Schaltfläche, um ihn als C++-Compiler für das System festzulegen. Klicken Sie auf OK, um fortzufahren.



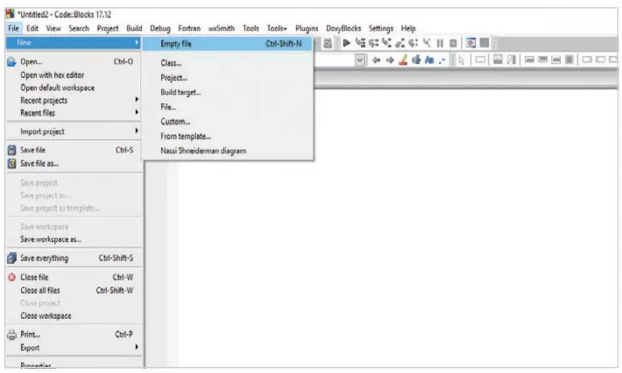
SCHRITT 7

Das Programm wird gestartet und eine weitere Meldung informiert Sie darüber, dass Code::Blocks derzeit nicht die Standardanwendung für C++-Dateien ist. Sie können entweder mit „Leave everything as it is“ alles belassen oder über die letzte Option Code::Blocks mit allen unterstützten Dateitypen verknüpfen. Wir empfehlen die letzte Option.



SCHRITT 8

Mit Code::Blocks lässt sich vieles machen, suchen Sie für eine optimale Nutzung daher nach einem guten C++-Tutorial. Klicken Sie jedoch zunächst auf File > New > Empty File. Dadurch wird ein neues leeres Fenster für die Eingabe erstellt.

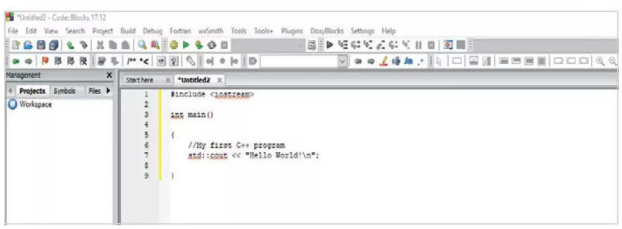


SCHRITT 9

Geben Sie in dem Fenster Folgendes ein:

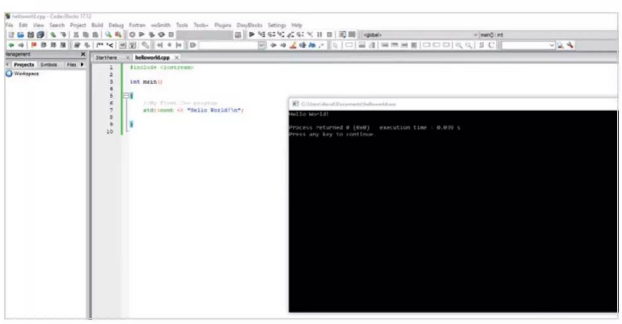
```
#include <iostream>

Int main()
{
//My first C++ program
Std::cout << “Hello World!\n”;
}
```

Beachten Sie, wie Code::Blocks die Klammern und Anführungszeichen automatisch einfügt.



SCHRITT 10

Klicken Sie auf File > Save as und speichern Sie den Code mit der Erweiterung .cpp (z. B. helloworld.cpp). Code::Blocks ändert die Ansicht in einen Farbcode gemäß den C++-Standards. Um den Code zu erstellen und auszuführen, klicken Sie oben auf dem Bildschirm auf die Schaltfläche „Build and Run“. Es ist das aus einem grünen Wiedergabepfeil vor einem gelben Zahnrad bestehende Symbol.



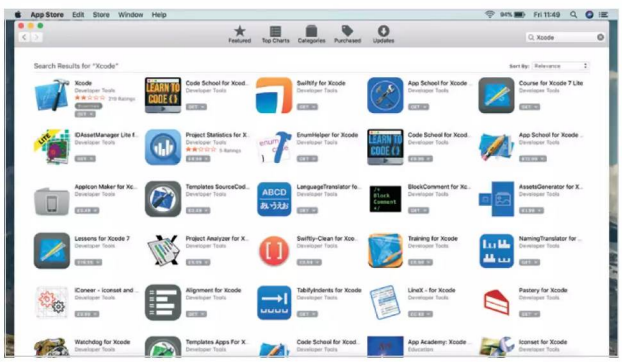
C++ auf einem Mac einrichten

Um auf einem Mac in C++ zu programmieren, müssen Sie Xcode von Apple installieren, eine kostenlose, voll funktionsfähige IDE, mit der native Apple-Apps erstellt werden. Sie kann aber auch relativ einfach zum Erstellen von C++-Code verwendet werden.

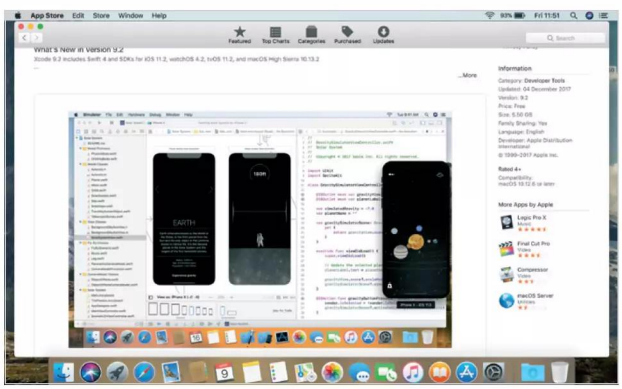
XCODE

Apples Xcode wurde zwar in erster Linie für Benutzer entworfen, die Apps für macOS, iOS, tvOS und watchOS-Apps in Swift oder Objective-C entwickeln, ist aber auch für C++ geeignet.

SCHRITT 1 Öffnen Sie den App Store auf Ihrem Mac (Apple-Menü > App Store). Geben Sie im Suchfeld Xcode ein und drücken Sie die Eingabetaste. Sie erhalten viele Vorschläge, die das App Store-Fenster füllen. Klicken Sie die erste Option, Xcode, an.

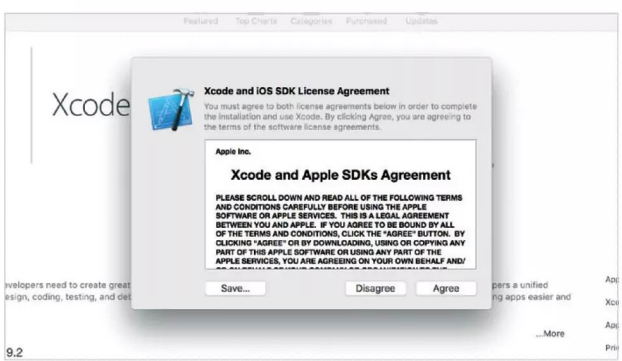


SCHRITT 2 Gehen Sie die Informationen der App durch, einschließlich der Kompatibilität, um sicherzustellen, dass Sie über die korrekte macOS-Version verfügen. Für Xcode ist macOS 10.12.6 oder höher erforderlich.



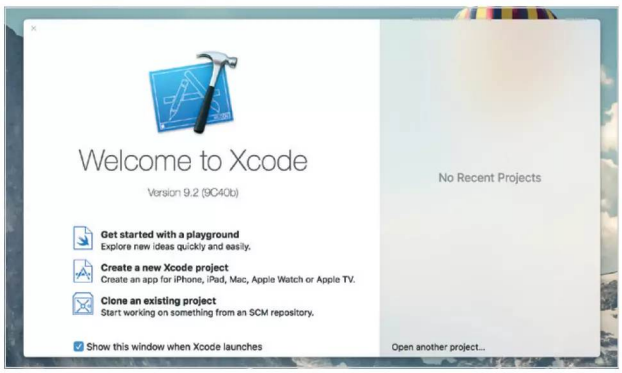
SCHRITT 3 Wenn Sie so weit sind, klicken Sie auf „Laden", woraufhin daraus „Installieren" wird. Geben Sie Ihre Apple-ID ein. Xcode wird heruntergeladen und installiert. Je nach Geschwindigkeit Ihrer Internetverbindung kann dies eine Weile dauern.

SCHRITT 4 Klicken Sie nach Abschluss der Installation auf „Öffnen", um Xcode zu starten. Akzeptieren Sie die Lizenzbedingungen und geben Sie Ihr Passwort ein, damit Xcode Änderungen am System vornehmen kann. Xcode beginnt daraufhin mit der Installation zusätzlicher Komponenten.



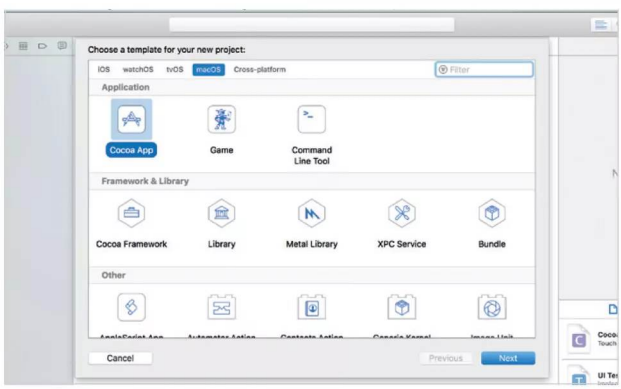
SCHRITT 5

Wenn alles installiert ist, einschließlich der zusätzlichen Komponenten, wird Xcode gestartet. Es werden die Versionsnummer sowie drei Auswahlmöglichkeiten und alle kürzlich durchgeführten Projekte angezeigt, die bei einer Neuinstallation natürlich noch nicht vorhanden sind.



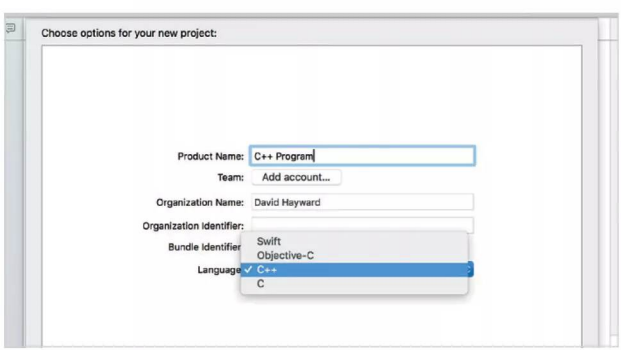
SCHRITT 6

Klicken Sie auf „Create New Xcode Projekt". Dadurch wird ein Vorlagenfenster geöffnet, aus dem Sie die Plattform auswählen können, für die Sie Code entwickeln. Wählen Sie den Tab „macOS", dann die Option „Command Line Tool" und klicken Sie anschließend auf „Next", um fortzufahren.



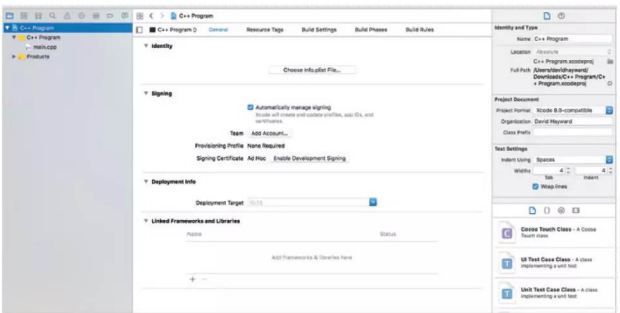
SCHRITT 7

Füllen Sie die verschiedenen Felder aus, stellen Sie jedoch sicher, dass die Language-Option unten auf C++ gesetzt ist. Wählen Sie es aus der Drop-down-Liste aus. Wenn Sie die Felder ausgefüllt haben und sichergestellt haben, dass C++ die ausgewählte Sprache ist, klicken Sie auf „Next", um fortzufahren.



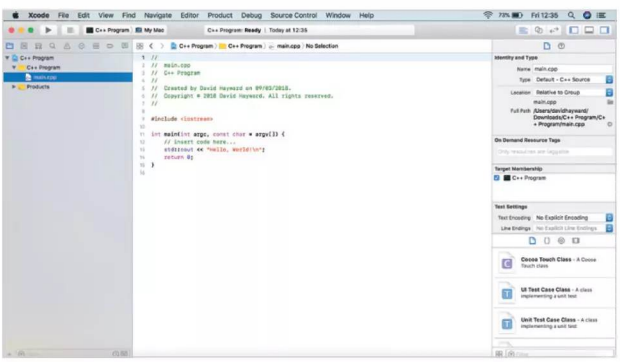
SCHRITT 8

Sie werden nun gefragt, wo Sie ein Git-Repository für Ihre zukünftigen Codes erstellen möchten. Wählen Sie einen Ort auf Ihrem Mac oder einen Netzwerkstandort aus und klicken Sie auf „Create". Sie können nun mit dem Programmieren beginnen. Links werden die Dateien aufgelistet, die in dem von Ihnen programmierten C++-Programm verwendet werden. Klicken Sie in der Liste auf die main.cpp-Datei.



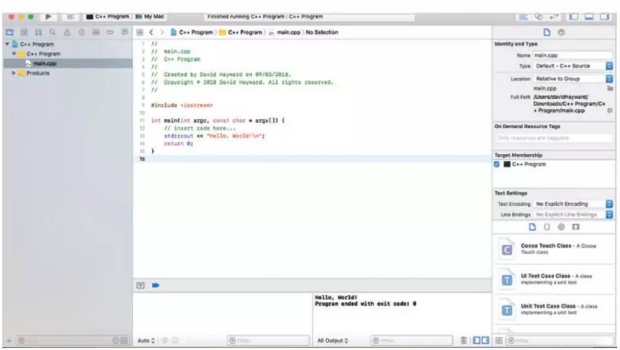
SCHRITT 9

Wie Sie sehen, hat Xcode automatisch ein einfaches Hello World-Programm für Sie abgeschlossen. Die Unterschiede hier sind, dass die Funktion int main () nun mehrere Funktionen enthält und das Layout etwas anders ist. Dies ist aber nur die Xcode-IDE, die den Content verwendet, der für Ihren Mac verfügbar ist.



SCHRITT 10

Zum Ausführen des Codes klicken Sie auf Product > Run. Evtl. werden Sie aufgefordert, den Entwicklermodus auf dem Mac zu aktivieren. Damit autorisieren Sie Xcode, Funktionen auszuführen, ohne dass bei jeder Sitzung Ihr Kennwort benötigt wird. Bei der Ausführung des Programms wird die Ausgabe am unteren Rand des Xcode-Fensters angezeigt.



C++ unter Linux einrichten

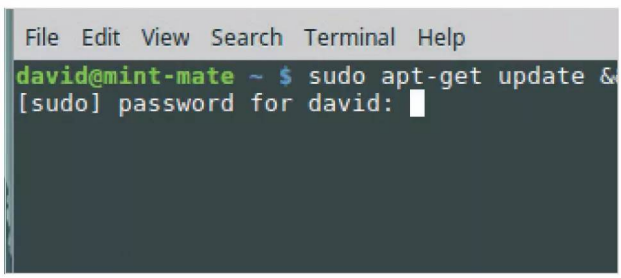
Linux bietet eine tolle Programmierumgebung für C++. In den meisten Linux-Distributionen sind die wichtigen Komponenten bereits vorinstalliert (z. B. Compiler). Die Texteditoren eignen sich hervorragend für die Code-Eingabe, einschließlich der Farbcodierung. Es gibt auch viele zusätzliche Software, die hilfreich sein kann.

LINUX++

Falls Sie mit Linux noch nicht vertraut sind, empfehlen wir, einen Blick auf unsere speziellen Linux-Ausgaben zu werfen. Sollten Sie einen Raspberry Pi haben, werden die folgenden Befehle problemlos funktionieren. Wir verwenden für dieses Beispiel Linux Mint.

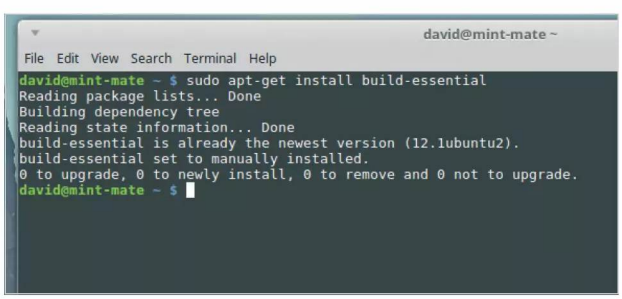
SCHRITT 1

Um sicherzustellen, dass Linux für Ihren C++-Code bereit ist, müssen Sie zunächst prüfen, ob System und Software auf dem neuesten Stand sind. Öffnen Sie ein Terminal und geben Sie `sudo apt-get update && sudo apt-get upgrade` ein. Drücken Sie die Eingabetaste und geben Sie Ihr Passwort ein. Mit diesen Befehlen wird das gesamte System und die installierte Software aktualisiert.



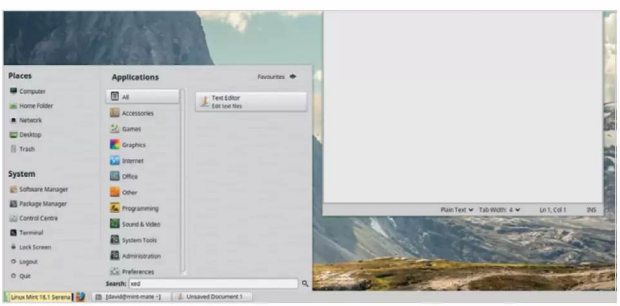
SCHRITT 2

Die meisten Linux-Distributionen haben alle erforderlichen Komponenten für das Programmieren in C++ vorinstalliert. Es lohnt sich jedoch immer zu prüfen, ob auch alles vorhanden ist. Geben Sie daher im Terminal `sudo apt-get install build-essential` ein, und drücken Sie die Eingabetaste. Sollten Komponenten fehlen, werden sie nun über diesen Befehl installiert.



SCHRITT 3

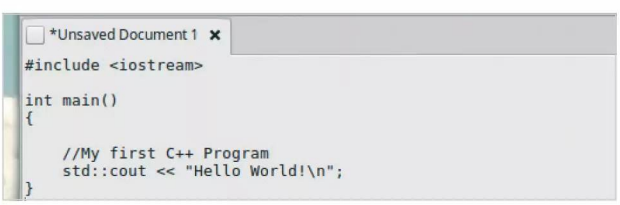
Und das war's auch schon. Es ist nun alles bereit, um mit dem Programmieren beginnen zu können. Um Ihr erstes C++-Programm auszuführen, starten Sie Xed, den Haupteditor in Linux Mint. Öffnen Sie dazu in Linux Mint das Menü und geben Sie `Xed` in die Suchleiste ein. Klicken Sie im rechten Bereich auf die Texteditor-Schaltfläche, um Xed zu öffnen.



SCHRITT 4

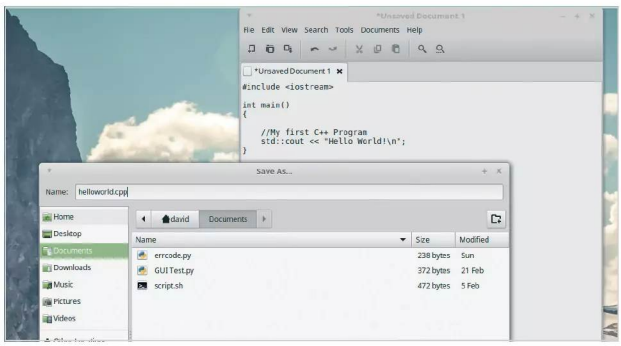
Geben Sie in Xed bzw. im Texteditor, den Sie verwenden, die Codezeilen ein, aus denen Ihr C++-Hello-World-Programm besteht. Hier noch mal zur Erinnerung:

```
#include <iostream>

int main()

{

//My first C++ program

std::cout <<  "Hello World!\n";
```



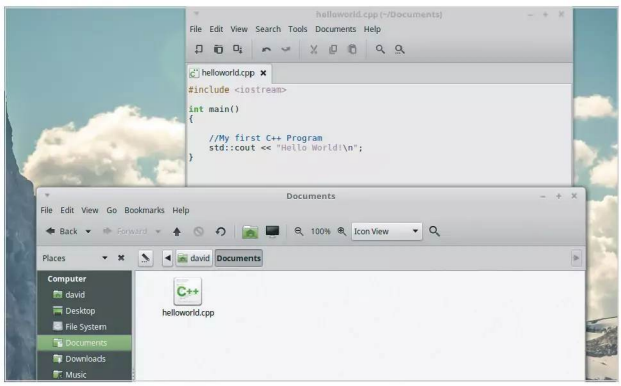
SCHRITT 5

Wenn Sie Ihren Code eingegeben haben, gehen Sie zu Datei > Speichern unter und wählen Sie den Ordner aus, in dem Sie das Programm speichern möchten. Nennen Sie die Datei helloworld.cpp oder geben Sie ihr einen beliebigen anderen Namen, sie muss aber die Erweiterung .cpp haben. Klicken Sie auf „Speichern", um fortzufahren.



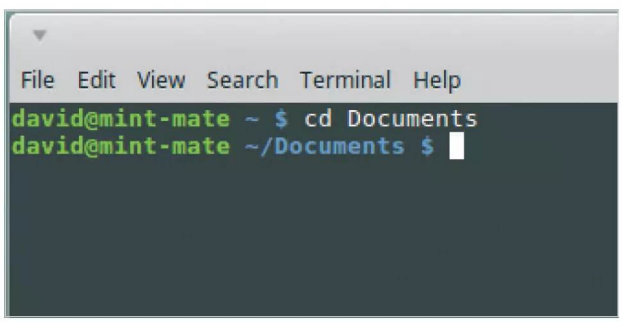
SCHRITT 6

Wie Sie sehen, hat Xed sie anhand der Dateierweiterung .cpp automatisch als C++-Datei erkannt. Die Farbcodierung ist im Code enthalten. Wenn Sie den Dateimanager öffnen, sehen Sie auch, dass das Symbol der Datei mit C++ versehen ist.



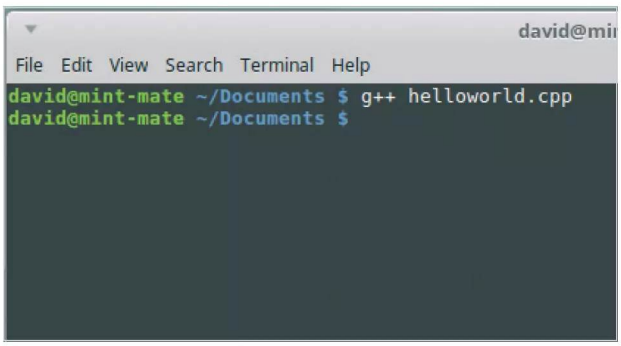
SCHRITT 7

Kehren Sie nun, nachdem Ihr Code gespeichert ist, ins Terminal zurück. Sie müssen zum Speicherort der gerade gespeicherten C++-Datei navigieren. Unser Beispiel befindet sich im Documents-Ordner, zu dem wir über den folgenden Befehl gelangen: `cd Documents`. Das Linux-Terminal achtet auf die Groß- und Kleinschreibung, daher müssen Großbuchstaben korrekt eingegeben werden.



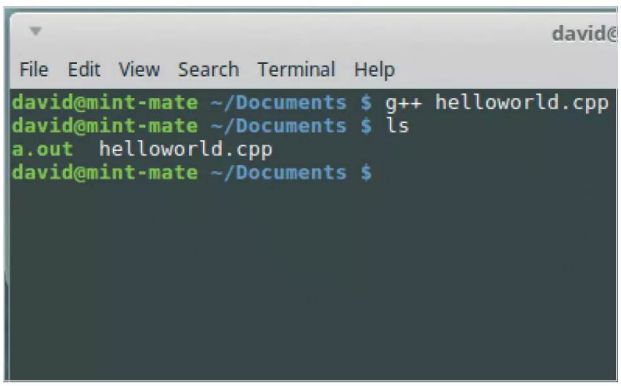
SCHRITT 8

Bevor Sie die C++-Datei ausführen können, müssen Sie sie kompilieren. In Linux ist es üblich, g++, ein Open-Source-C++-Compiler, dafür zu verwenden. Da Sie sich jetzt im selben Ordner wie die C++-Datei befinden, geben Sie im Terminal `g++ helloworld.cpp` ein, und drücken Sie die Eingabetaste.



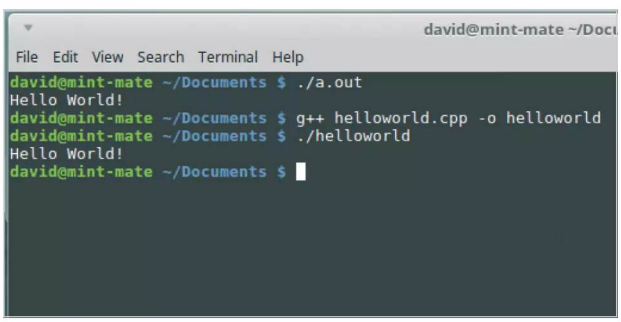
SCHRITT 9

Es wird eine kurze Pause geben, da der Code von g++ kompiliert wird. Sofern im Code keine Fehler enthalten sind, gelangen Sie zur Eingabeaufforderung zurück. Beim Kompilieren des Codes wurde eine neue Datei erstellt. Wenn Sie nun `ls` ins Terminal eingeben, sehen Sie neben Ihrer C++-Datei a.out.



SCHRITT 10

Die a.out-Datei ist der kompilierte C++-Code. Zum Ausführen des Codes geben Sie `./a.out` ein und drücken die Eingabetaste. a.out klingt jedoch nicht sehr praktisch. Um die Datei nach dem Kompilieren umzubenennen, können Sie sie mit `g++ helloworld.cpp -o helloworld` neu kompilieren. Dadurch wird eine Ausgabedatei namens helloworld erstellt, die mit `./helloworld` ausgeführt werden kann.



Weitere C++-IDEs

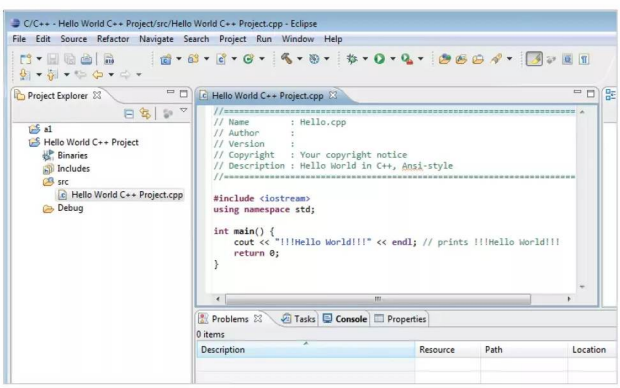
Wenn Sie die Arbeit mit Ihrem C++-Code anderweitig angehen möchten, stehen Ihnen zahlreiche Optionen zur Verfügung. Windows ist die am weitesten verbreitete Plattform für C++-IDEs, aber auch für Mac- und Linux-Benutzer stehen viele zur Auswahl.

ENTWICKLUNGSUMGEBUNGEN FÜR C++

Hier sind zehn tolle C++-IDEs, die einen Blick wert sind. Wenn Sie möchten, können Sie eine oder auch alle installieren, schauen Sie einfach, welche am besten für Sie geeignet ist.

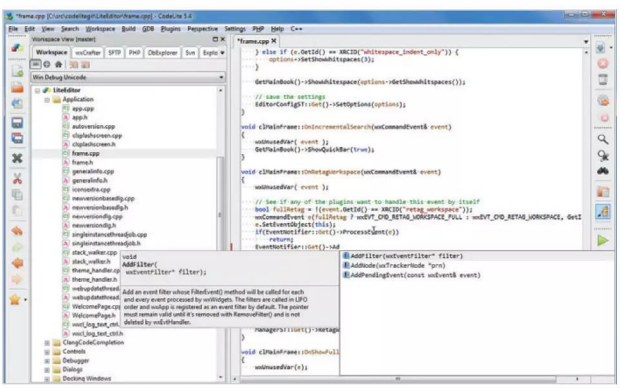
ECLIPSE

Eclipse ist eine beliebte C++-IDE, die eine Fülle von Funktionen bietet. Sie hat eine tolle, übersichtliche Oberfläche, ist einfach zu bedienen und für Windows, Linux und Mac erhältlich. Auf **www.eclipse.org/downloads/** können Sie die neueste Version herunterladen. Wenn Sie nicht weiterkommen, können Sie über den Help-Link weitere Informationen erhalten.



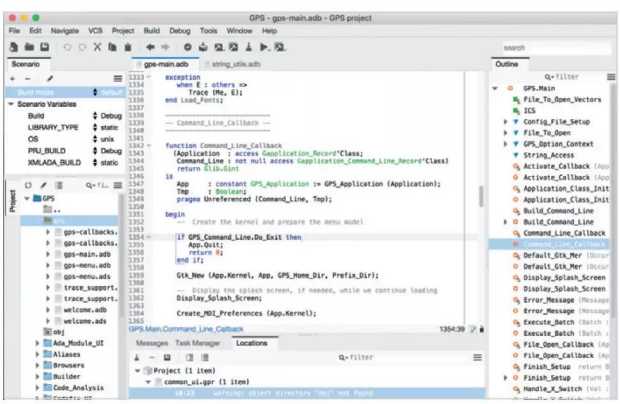
CODELITE

CodeLite ist eine kostenlose Open-Source-IDE, die regelmäßig aktualisiert wird und für Windows, Linux und macOS erhältlich ist. Sie ist leicht, unkompliziert und extrem leistungsstark. Weitere Informationen sowie Angaben zum Herunterladen und Installieren finden Sie unter **www.codelite.org/**.



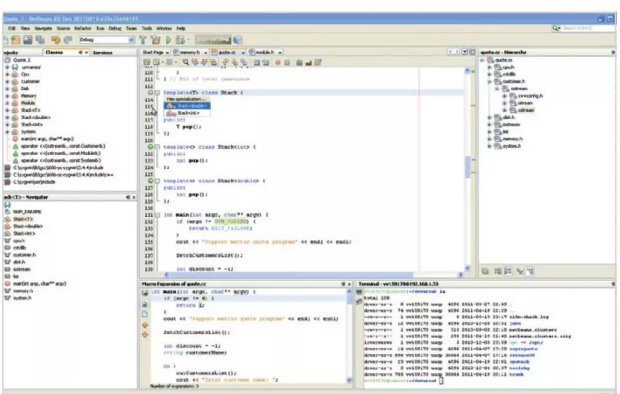
GNAT

Das GNAT Programming Studio (GPS) ist eine leistungsstarke und intuitive IDE, die das Testen, Debugging und die Code-Analyse unterstützt. Die Community Edition ist kostenlos, während die Pro-Version kostenpflichtig ist. Die Community Edition ist allerdings für Windows, Mac, Linux und sogar für den Raspberry Pi erhältlich. Sie finden GNAT unter **www.adacore.com/download**.



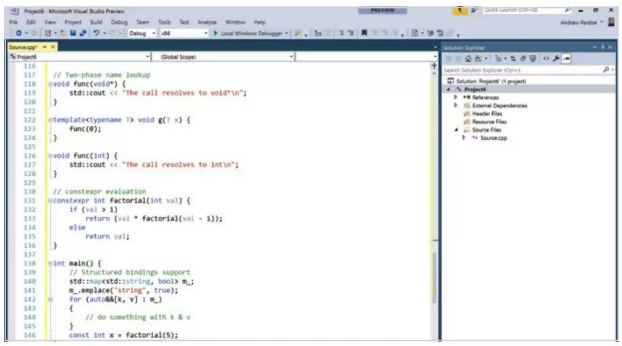
NETBEANS

Eine weitere beliebte Wahl ist NetBeans, eine exzellente IDE, die voll mit Funktionen ist und Spaß bei der Anwendung macht. Die NetBeans-IDE enthält projektbasierte Vorlagen für C++, mit denen Sie Anwendungen mit dynamischen und statischen Bibliotheken erstellen können. Weitere Infos gibt es unter **www.netbeans.org/features/cpp/index.html**.



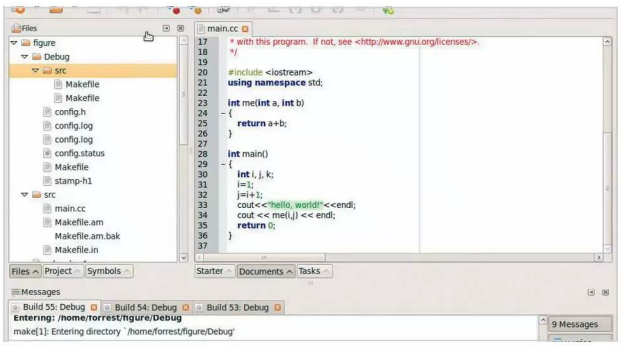
VISUAL STUDIO

Microsoft Visual Studio ist eine riesige C++-IDE, mit der Sie Anwendungen für Windows, Android, iOS und das Web erstellen können. Die Community-Version kann gratis heruntergeladen und installiert werden, die anderen Versionen bieten jedoch eine kostenlose Testphase. Besuchen Sie **www.visualstudio.com/**, um mehr darüber zu erfahren.



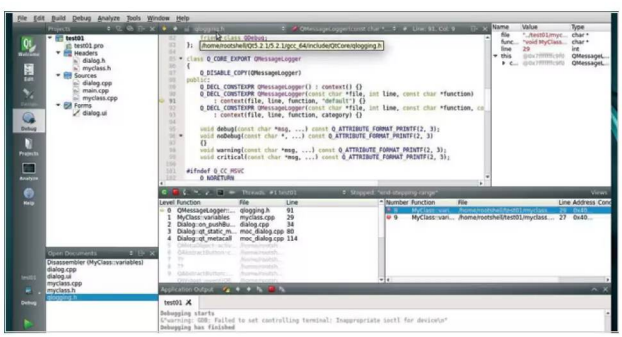
ANJUTA

Das Anjuta DevStudio ist eine reine Linux-IDE, die einige der erweiterten Funktionen bietet, die Sie normalerweise in einem kostenpflichtigen Software-Entwicklungsstudio finden würden. Es gibt einen GUI-Designer, einen Quelleditor, einen App-Assistenten, einen interaktiven Debugger und vieles mehr. Weitere Infos finden Sie auf **www.anjuta.org/**.



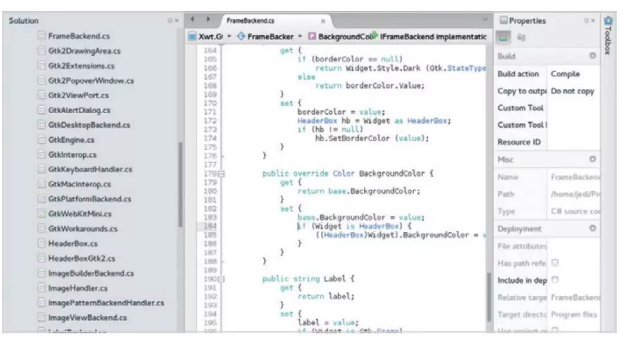
QT CREATOR

Diese plattformübergreifende IDE wurde entwickelt, um C++-Anwendungen für Desktop- und mobile Umgebungen zu erstellen. Sie enthält einen Code-Editor und integrierte Tools zum Testen und Debuggen sowie zur Bereitstellung auf Ihrer ausgewählten Plattform. Sie ist kostenpflichtig, bietet aber vor dem Kauf eine Probezeit an: **www.qt.io/qt-features-libraries-apis-tools-and-ide/**.



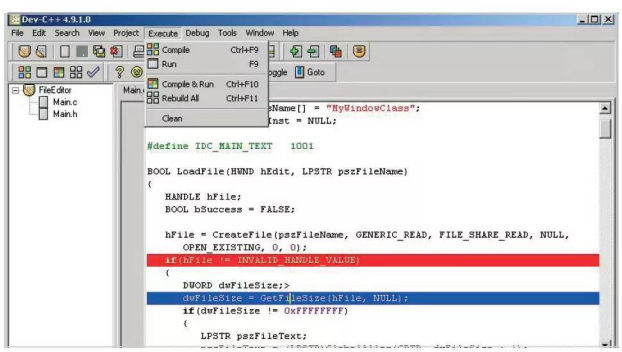
MONODEVELOP

Mit dieser hervorragenden IDE können Entwickler C++-Code für Desktop- und Webanwendungen auf allen wichtigen Plattformen schreiben. Sie hat einen erweiterten Texteditor, einen integrierten Debugger und eine konfigurierbare Workbench, die bei der Code-Erstellung hilft. Sie ist für Windows, Mac und Linux verfügbar und kann kostenlos heruntergeladen und genutzt werden: **www.monodevelop.com/**.



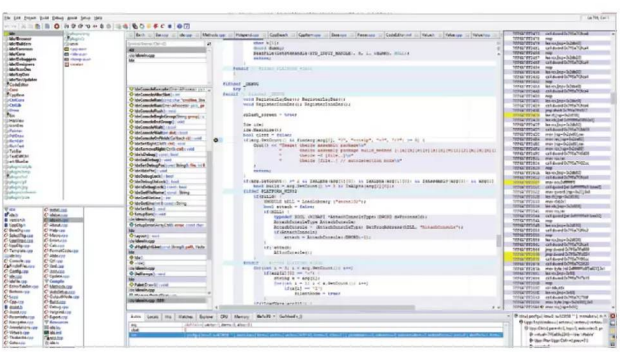
DEV C++

Bloodshed Dev C++ ist eine ältere IDE, die nur für Windows-Systeme gilt. Viele Benutzer loben jedoch ihre übersichtliche Oberfläche und wie leicht das Programmieren und Kompilieren fällt. Obwohl sie seit einiger Zeit nicht aktualisiert wurde, lohnt es sich, sie in Betracht zu ziehen, wenn Sie eine etwas andere IDE wollen: **www.bloodshed.net/devcpp.html**.

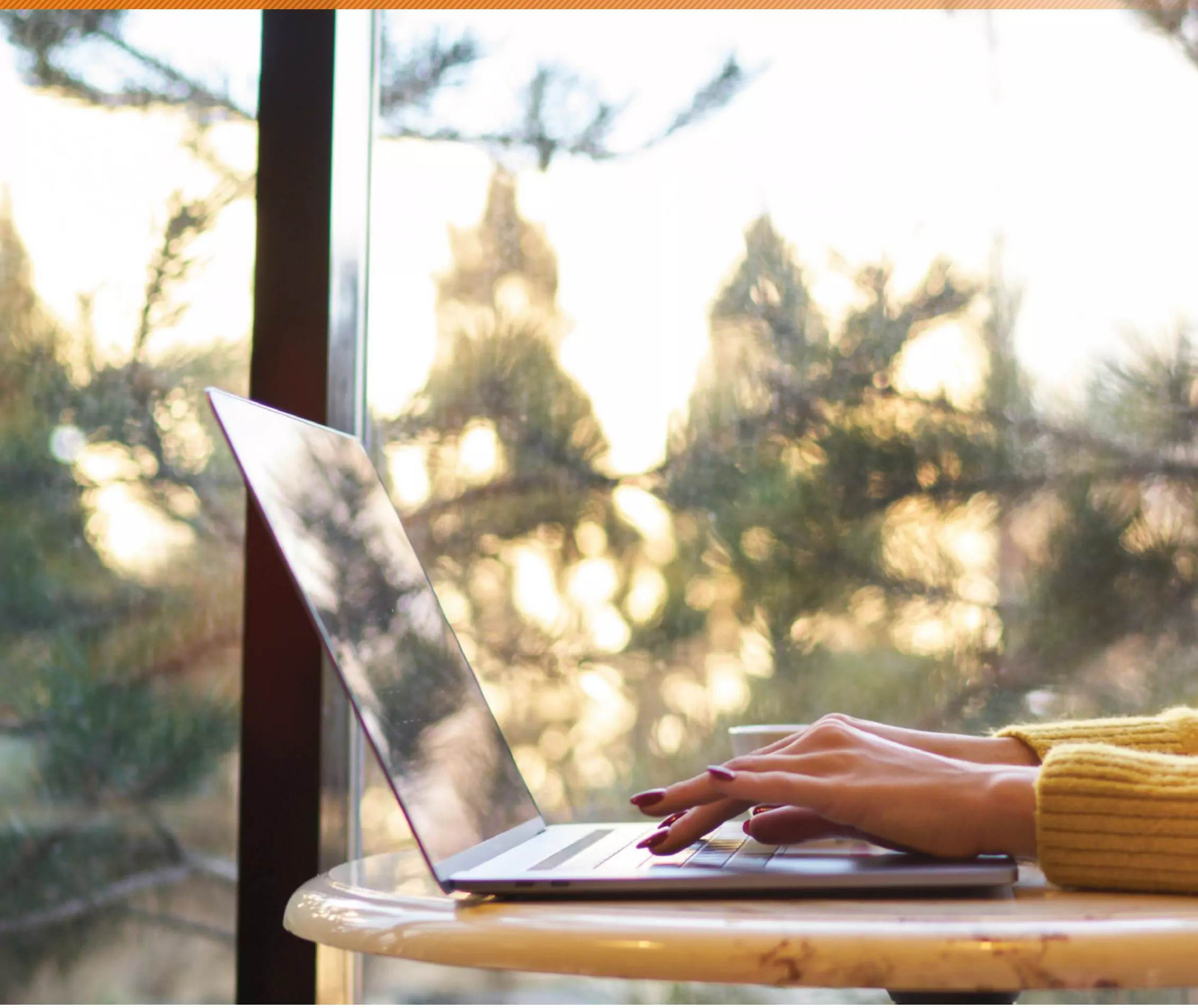


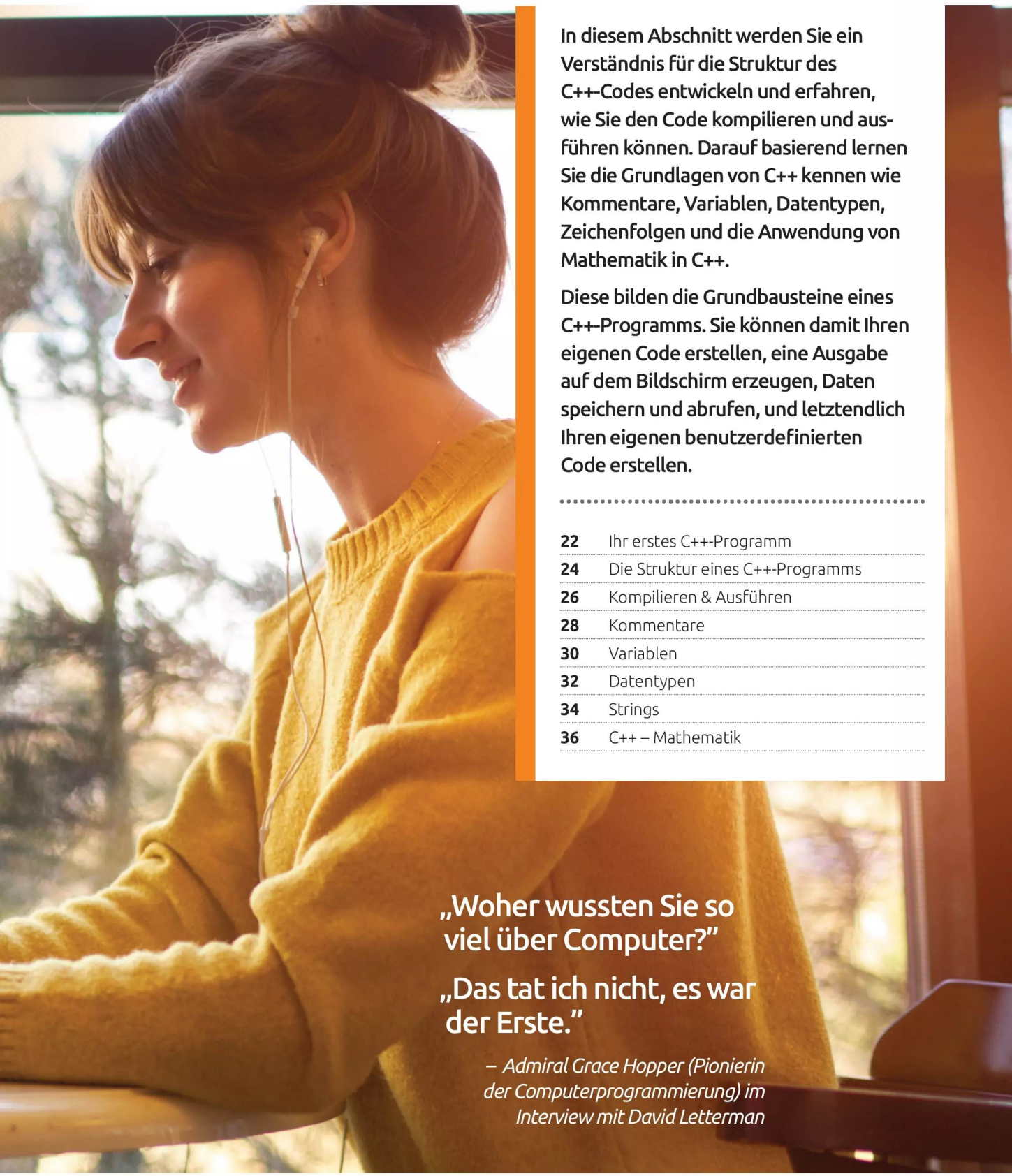
U++

Ultimate++ ist eine plattformübergreifende C++-IDE, die durch den intelligenten und aggressiven Einsatz von C++ eine schnelle Code-Erstellung ermöglicht. Für den Anfänger kann diese IDE schlichtweg überwältigend sein, erfahreneren Entwicklern lässt sie aber das Herz höher schlagen. Weitere Informationen gibt es auf **www.ultimatepp.org/index.html**.



C++ Die Grundlagen





In diesem Abschnitt werden Sie ein Verständnis für die Struktur des C++-Codes entwickeln und erfahren, wie Sie den Code kompilieren und ausführen können. Darauf basierend lernen Sie die Grundlagen von C++ kennen wie Kommentare, Variablen, Datentypen, Zeichenfolgen und die Anwendung von Mathematik in C++.

Diese bilden die Grundbausteine eines C++-Programms. Sie können damit Ihren eigenen Code erstellen, eine Ausgabe auf dem Bildschirm erzeugen, Daten speichern und abrufen, und letztendlich Ihren eigenen benutzerdefinierten Code erstellen.

„Woher wussten Sie so viel über Computer?"

„Das tat ich nicht, es war der Erste."

– Admiral Grace Hopper (Pionierin der Computerprogrammierung) im Interview mit David Letterman

Ihr erstes C++-Programm

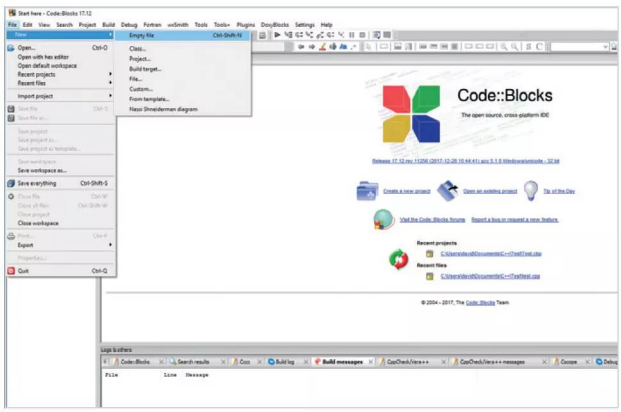
Möglicherweise haben Sie die Mac- und Linux-Beispiele zuvor bereits befolgt, wir werden ab jetzt jedoch ausschließlich in Windows und Code::Blocks arbeiten. Wir beginnen mit dem Erstellen unseres ersten C++-Programms.

HELLO, WORLD!

Beim Programmieren ist es üblich, dass mit dem ersten Code die Wörter „Hello, World!" auf dem Bildschirm ausgegeben werden. Interessanterweise wurde dies im Jahr 1968 in einer Sprache namens BCPL geschrieben.

SCHRITT 1

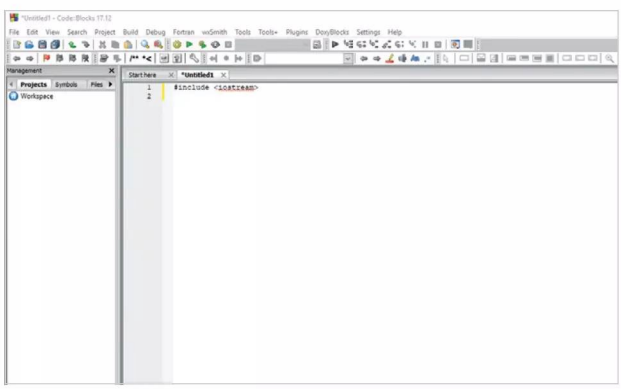
Wie bereits erwähnt, verwenden wir Windows 10 und die neueste Version von Code::Blocks für diesen Abschnitt. Starten Sie Code::Blocks und klicken Sie auf File > New > Empty File oder drücken Sie die Tastenkombination Strg + Umschalt + N.



SCHRITT 2

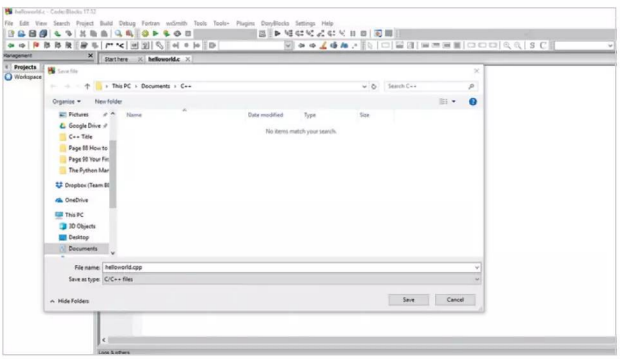
Sie sehen nun einen leeren Bildschirm mit dem Tab *Untitled1 und der Nummer 1 oben links im Hauptfenster von Code::Blocks. Machen Sie einen Klick im Hauptfenster, damit sich der Cursor neben der Nummer 1 befindet, und geben Sie Folgendes ein:

```
#include <iostream>
```



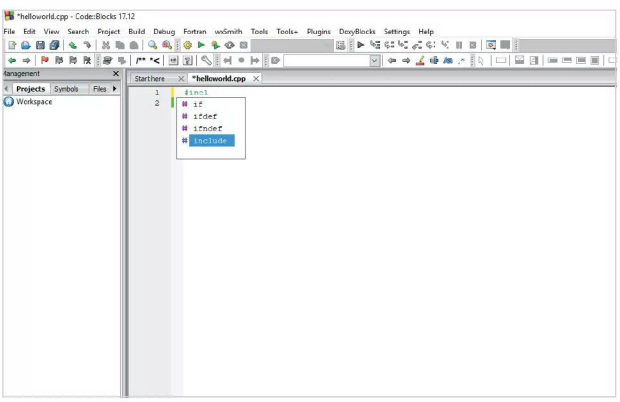
SCHRITT 3

Momentan sieht das Ganze nach nichts aus und macht auch nicht viel Sinn, aber das kommt noch. Klicken Sie nun auf File > Save File As und erstellen oder suchen Sie einen geeigneten Speicherort auf Ihrer Festplatte. Geben Sie im Feld „File Name" den Namen helloworld.cpp ein. Klicken Sie auf das Feld „Save as type" und wählen Sie C/C++ files. Klicken Sie anschließend auf „Save".



SCHRITT 4

Wie Sie sehen, hat Code::Blocks die Farbcodierung geändert und erkennt die Datei als C++-Code an. Dies bedeutet, dass Code automatisch aus dem Code::Blocks-Repository ausgewählt werden kann. Löschen Sie die Zeile #include <iostream> und geben Sie sie erneut ein. Es werden nun die Felder für die automatische Auswahl angezeigt.

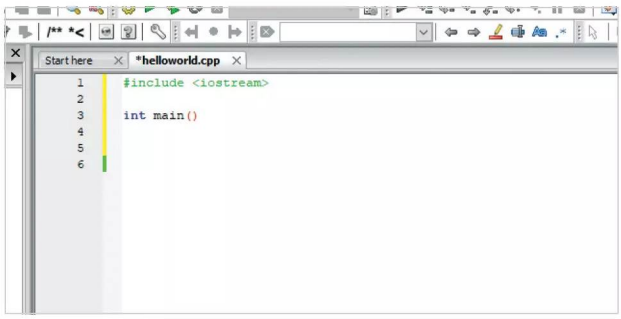


SCHRITT 5

Die automatische Auswahl von Befehlen ist äußerst praktisch und vermeidet eventuell falsche Eingaben. Drücken Sie die Eingabetaste, um zu Zeile 3 zu gelangen, und geben Sie dann Folgendes ein:

```
int main()
```

Beachten Sie, dass zwischen den Klammern kein Leerzeichen ist.

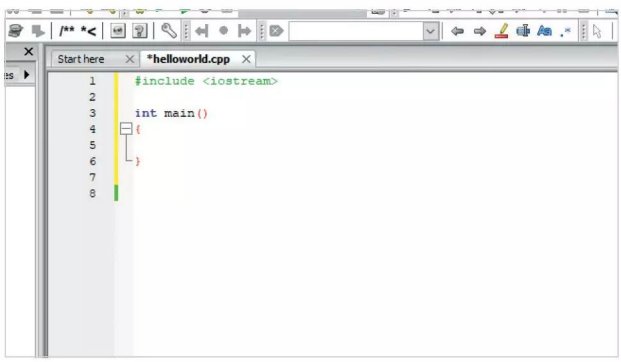


SCHRITT 6

Geben Sie in der nächsten Zeile unter int main() eine geschweifte Klammer ein:

```
{
```

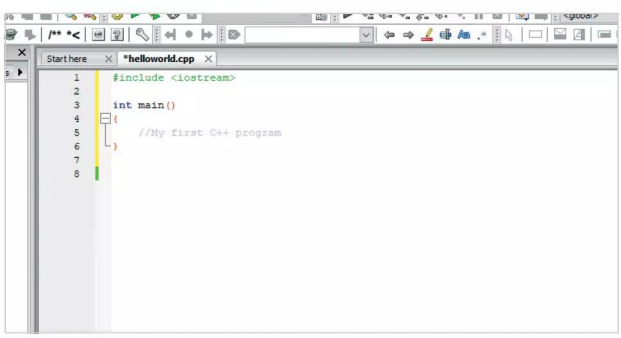
Die geschweiften Klammern erhalten Sie über die Tastenkombination Alt Gr + 7 und Alt Gr + 0.



SCHRITT 7

Wie Sie sehen, hat Code::Blocks automatisch etwas weiter unten die entsprechende geschlossene geschweifte Klammer sowie eine Einrückung eingefügt. Dies ist auf die Struktur von C++ zurückzuführen und es ist hier, wo der Code eingegeben wird. Geben Sie Folgendes ein:

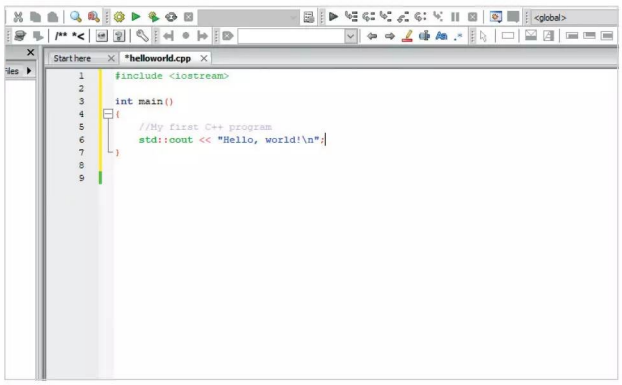
```
//My first C++ program
```



SCHRITT 8

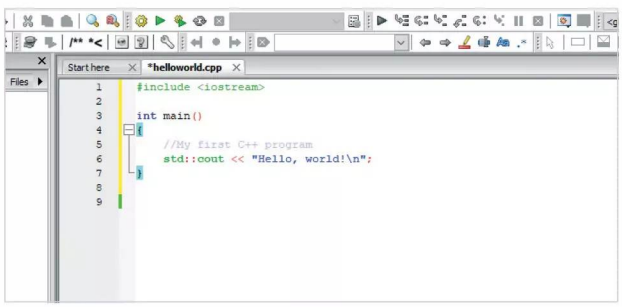
Beachten Sie erneut die veränderte Farbcodierung. Drücken Sie am Ende der Zeile des vorherigen Schritts die Eingabetaste, und geben Sie Folgendes ein:

```
std::cout << "Hello, world!\n";
```



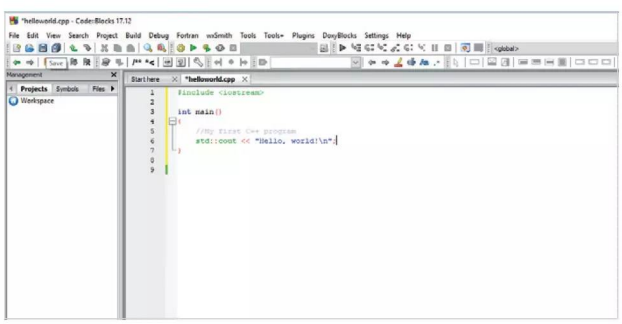
SCHRITT 9

Wie zuvor vervollständigt Code::Blocks den von Ihnen eingegebenen Code automatisch und setzt auch die hinteren Anführungszeichen, sobald Sie die vorderen eingeben. Vergessen Sie nicht das Semikolon am Ende der Zeile – dies ist eines der wichtigsten Elemente eines C++-Programms. Warum erklären wir Ihnen im nächsten Abschnitt. Setzen Sie den Cursor zunächst auf die geschlossene geschweifte Klammer und drücken Sie die Eingabetaste.



SCHRITT 10

Das ist alles, was im Moment zu tun ist. Es mag nicht besonders spannend erscheinen, aber C++ erlernt man am besten stückchenweise. Führen Sie den Code noch nicht aus, da Sie sich zuerst anschauen sollten, wie ein C++-Programm aufgebaut ist. Danach können Sie den Code erstellen und ausführen. Klicken Sie zunächst zum Speichern auf das Symbol in Form einer Diskette.



Die Struktur eines C++-Programms

C++ hat eine äußerst definierte Struktur und Vorgehensweise. Wenn etwas fehlt, sei es auch nur ein Semikolon, kann das gesamte Programm weder kompiliert noch ausgeführt werden.

#INCLUDE <C++ STRUKTUR>

Das Lernen der Grundlagen der Programmierung ermöglicht Ihnen, die Struktur eines Programms zu verstehen. Die Befehle können sich zwar von Sprache zu Sprache unterscheiden, Sie werden aber erkennen können, wie der Code funktioniert.

C++

C++ wurde 1979 vom dänischen Studenten Bjarne Stroustrup im Rahmen seiner Doktorarbeit erfunden. Ursprünglich „C with Classes" genannt, hat C++ die beliebte C-Sprache um weitere Funktionen erweitert und gleichzeitig mithilfe einer neuen Struktur eine benutzerfreundlichere Umgebung geschaffen.

Bjarne Stroustrup, der Erfinder von C++.

#INCLUDE

Die C++-Programmstruktur ist recht präzise. Jeder C++-Code beginnt mit einer Anweisung: **#include <>**. Diese weist den Präprozessor an, einen Abschnitt des Standard-C++-Codes miteinzubeziehen; z. B. enthält **#include <iostream>** den iostream-Header zur Unterstützung von Eingabe-/Ausgabeoperatoren.

Start here | *helloworld.cpp

```
1  #include <iostream>
2
3
4
5
6
```

INT MAIN()

int main() initiiert die Deklaration einer Funktion, bei der es sich um eine Gruppe von Code-Anweisungen unter dem Namen „main" handelt. Der gesamte C++-Code beginnt bei der Hauptfunktion, unabhängig davon, wo sie sich im Code befindet.

Start here | *helloworld.cpp

```
1  #include <iostream>
2
3  int main()
4
5
6
```

KLAMMERN

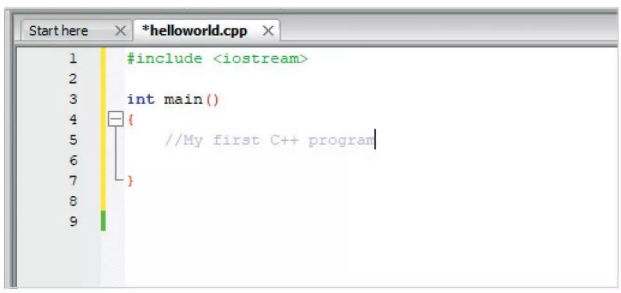
Die geöffnete geschweifte Klammer ist etwas, auf das Sie wahrscheinlich noch nicht gestoßen sind, besonders wenn Sie Python gewohnt sind. Die geöffnete Klammer markiert den Beginn der Hauptfunktion und enthält den gesamten Code, der zu dieser Funktion gehört.

Start here | *helloworld.cpp

```
1  #include <iostream>
2
3  int main()
4  {
5
6  }
```

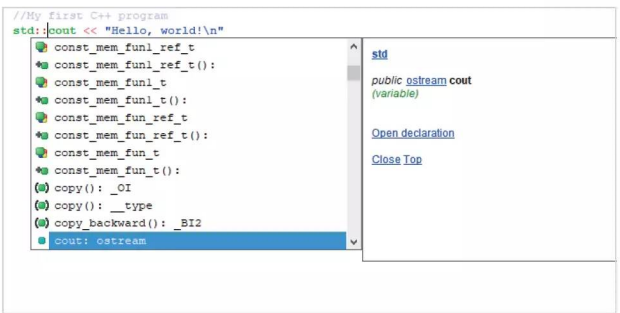
KOMMENTARE

Zeilen, die mit einem doppelten Schrägstrich beginnen, sind Kommentare. Dies bedeutet, dass sie nicht im Code ausgeführt und vom Compiler ignoriert werden. Kommentare sollen Ihnen oder einem anderen Programmierer, der Ihren Code durchgeht, helfen, zu erklären, was passiert. Es gibt zwei Arten von Kommentaren: /* betrifft mehrzeilige Kommentare, // eine einzelne Zeile.



<<

Die zwei hier verwendeten Pfeile sind Einfügeoperatoren. Dies bedeutet, dass alles, was auf die Pfeile folgt, in die Anweisung std::cout eingefügt werden muss. In diesem Fall sind es die Wörter „Hello, world", die beim Kompilieren und Ausführen des Codes auf dem Bildschirm angezeigt werden sollen.



STD

In C++ steht **std** für Standard. Es ist ein Teil des Standard-Namensraums in C++, der eine Reihe verschiedener Anweisungen und Befehle abdeckt. Sie können den **std::**-Teil eines Codes auch auslassen, er muss jedoch zu Beginn mit **using namespace std** deklariert werden::

```
#include <iostream>
using namespace std;
```

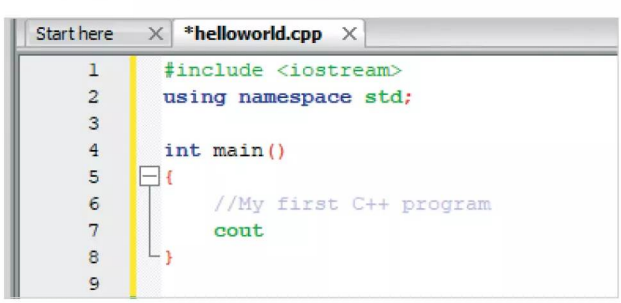
AUSGABEN

Bei Ausführung des Codes soll der Teil „Hello, world!" auf dem Bildschirm ausgegeben werden. Sie können eingeben, was immer Sie möchten, es muss sich aber in den Anführungszeichen befinden. Abhängig vom Compiler werden manchmal Klammern benötigt. Der \n-Teil zeigt an, dass eine neue Zeile eingefügt werden soll.

```
//My first C++ program
cout << "Hello, world!\n"
```

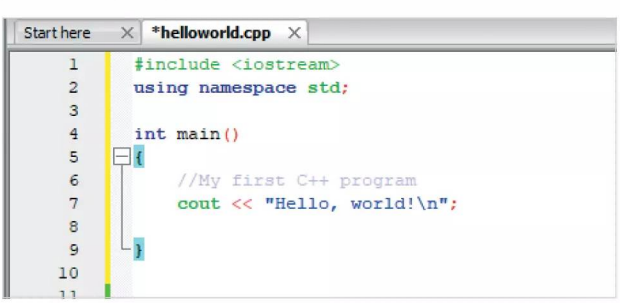
COUT

In diesem Beispiel verwenden wir cout. Cout ist Teil des Standard-Namensraums, und wir nehmen es, da wir C++ bitten, etwas aus diesem bestimmten Namensraum zu verwenden. Cout bedeutet Character OUTput und zeigt oder gibt etwas auf dem Bildschirm an bzw. aus. Wenn wir std:: weglassen, müssen wir das, wie bereits erwähnt, am Anfang des Codes deklarieren.



; UND }

Zeilen innerhalb eines Funktionscodeblocks (mit Ausnahme von Kommentaren) enden mit einem Semikolon, welches das Ende einer Anweisung markiert. Alle Anweisungen in C++ müssen ein Semikolon enthalten, andernfalls kann der Compiler den Code nicht erstellen. Die letzte Zeile enthält die schließende Klammer, die das Ende der Hauptfunktion markiert.



Kompilieren & Ausführen

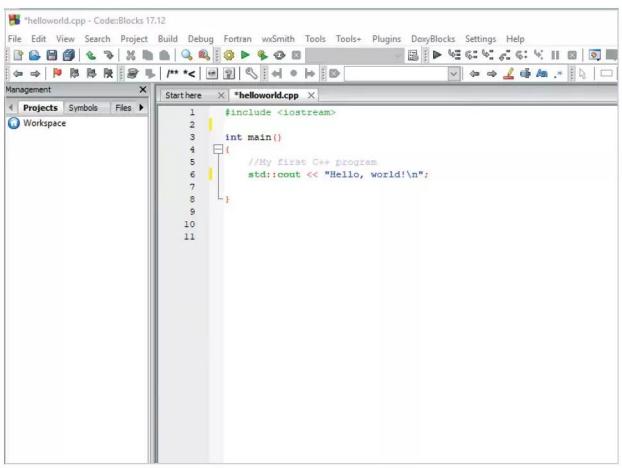
Sie haben Ihr erstes C++-Programm erstellt und verstehen nun seine strukturellen Grundlagen. Wir bringen nun die Dinge ins Rollen und werden das Programm kompilieren und ausführen und uns anschauen, was dabei herauskommt.

C++ LÄSST GRÜSSEN

Das Kompilieren und Ausführen von C++-Code in Code::Blocks ist äußerst einfach, ein Klick auf ein Symbol und schon erscheint das Ergebnis. Hier zeigen wir Ihnen, wie es funktioniert.

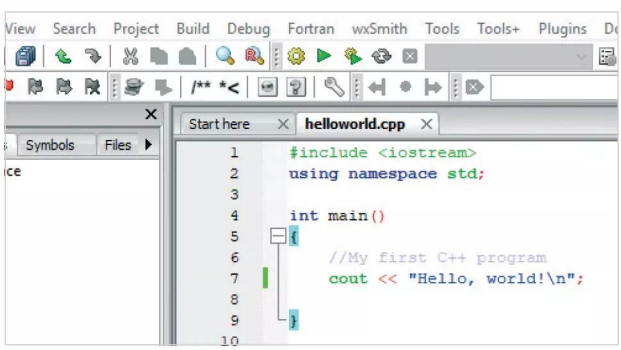
SCHRITT 1

Öffnen Sie Code::Blocks und laden Sie den zuvor gespeicherten Hello World-Code hoch. Stellen Sie sicher, dass keine sichtbaren Fehler vorhanden sind, z. B. ein fehlendes Semikolon am Ende der Zeile std::cout.



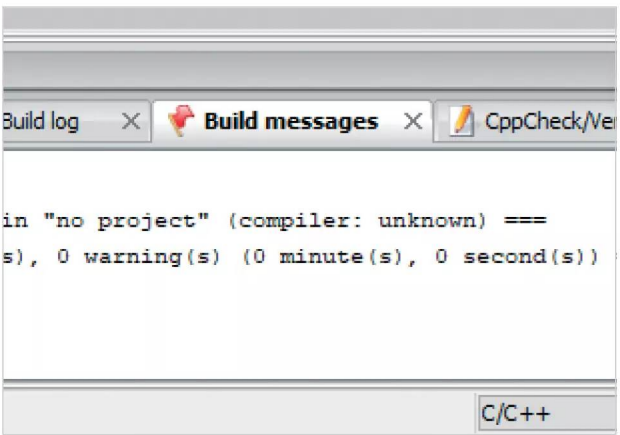
SCHRITT 2

Wenn Ihr Code dem Code in unserem Screenshot ähnelt, werfen Sie einen Blick auf die Menüleiste am oberen Bildschirmrand. In der oberen Menüleiste sehen Sie unter Fortran eine Gruppe von Symbolen: ein gelbes Zahnrad, eine grüne Wiedergabetaste und eine Kombination aus Zahnrad-/Wiedergabetaste. Dies sind die Funktionen Erstellen, Ausführen sowie Erstellen/Ausführen.



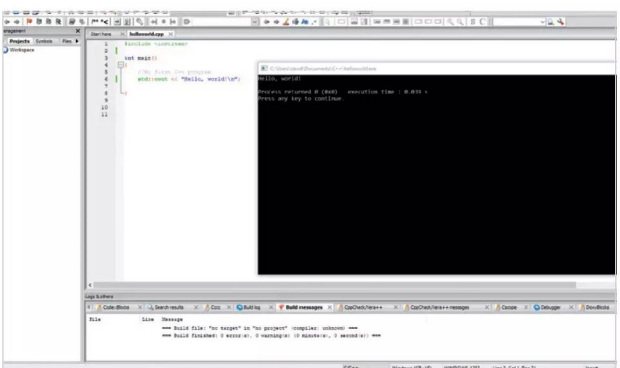
SCHRITT 3

Klicken Sie zunächst auf das gelbe Zahnrad. Ihr Code hat nun den Code::Blocks-Compiler durchlaufen und wurde auf Fehler überprüft. Sie können die Ergebnisse im unteren Fensterbereich sehen. Hier werden alle Meldungen zur Codequalität angezeigt.



SCHRITT 4

Klicken Sie nun zum Ausführen auf die grüne Wiedergabetaste. Auf dem Bildschirm erscheint ein Befehlszeilenfeld mit den Wörtern Hello, world!, gefolgt von der Zeit, die zur Ausführung des Codes benötigt wurde, und der Aufforderung, zum Fortfahren eine Taste zu drücken. Gratulation! Sie haben gerade Ihr erstes C++-Programm kompiliert und ausgeführt.



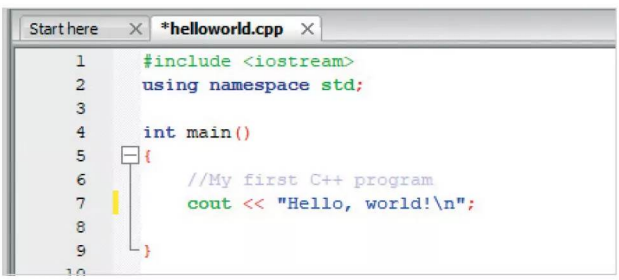
SCHRITT 5

Durch Drücken einer beliebigen Taste im Befehlszeilenfeld wird das Fenster geschlossen und Sie kehren zu Code::Blocks zurück. Wir werden den Code nun etwas ändern. Geben Sie unterhalb der Zeile #include Folgendes ein:

```
using namespace std;
```

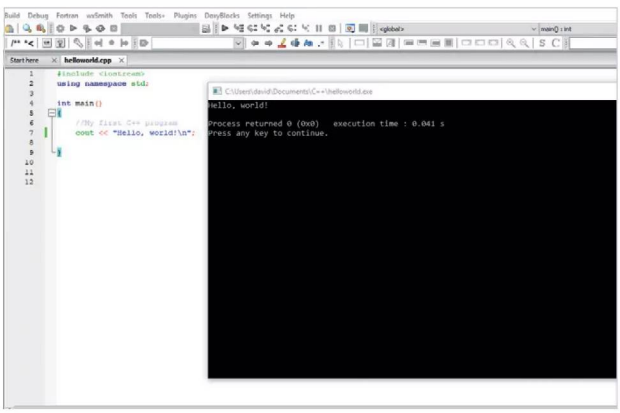
Löschen Sie anschließend den std::-Teil der cout-Zeile:

```
cout << "Hello, world\n";
```



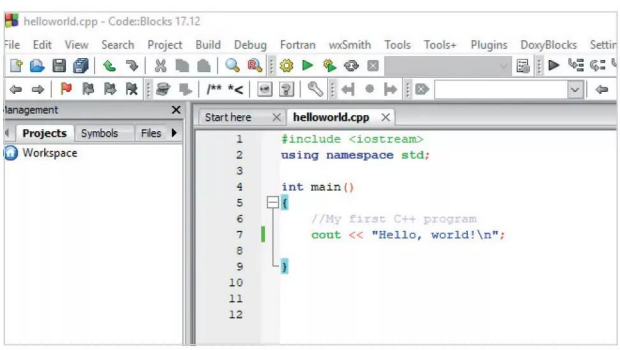
SCHRITT 6

Um die neuen Änderungen im Code zu übernehmen, müssen Sie ihn erneut kompilieren, erstellen und ausführen. Diesmal können Sie jedoch einfach auf das kombinierte Zahnrad/Wiedergabetaste-Symbol klicken.



SCHRITT 7

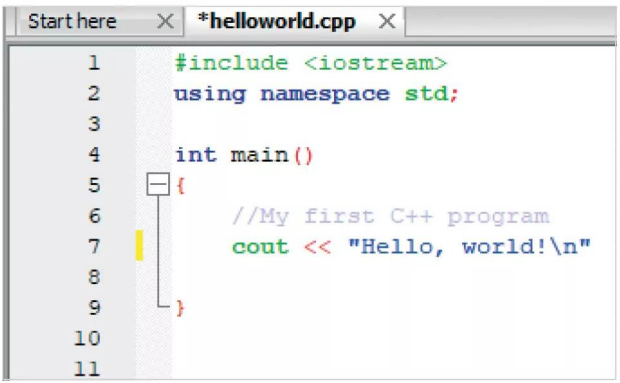
Wie wir zuvor bereits erwähnt haben, brauchen Sie std::cout nicht, wenn Sie namespace std; am Anfang des Codes stehen haben. Wir könnten einfach nur auf das Zahnrad/Wiedergabetaste-Symbol klicken, aber es lohnt sich, die verfügbaren Optionen durchzugehen. Wie Sie sehen, wurde die Datei durch Erstellen/Ausführen gespeichert.



SCHRITT 8

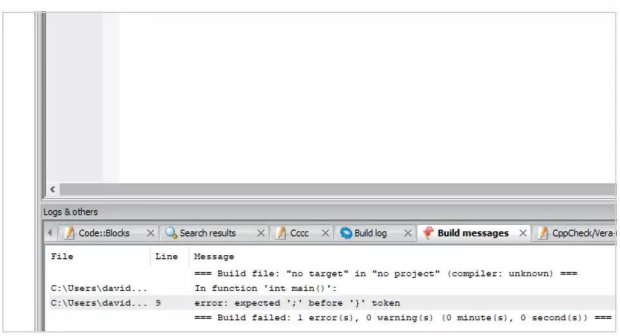
Erstellen Sie einen absichtlichen Fehler im Code, indem Sie das Semikolon aus der cout-Zeile entfernen:

```
cout << "Hello, world!\n"
```



SCHRITT 9

Klicken Sie erneut auf das Zahnrad/Wiedergabetaste-Symbol, um die Änderungen im Code zu übernehmen. Diesmal verweigert Code::Blocks die Ausführung des Codes aufgrund des von Ihnen eingegebenen Fehlers. In der Protokollleiste am unteren Bildschirmrand werden Sie über den Fehler informiert; in diesem Fall „expected ';' before '}' token", was auf das fehlende Semikolon hinweist.

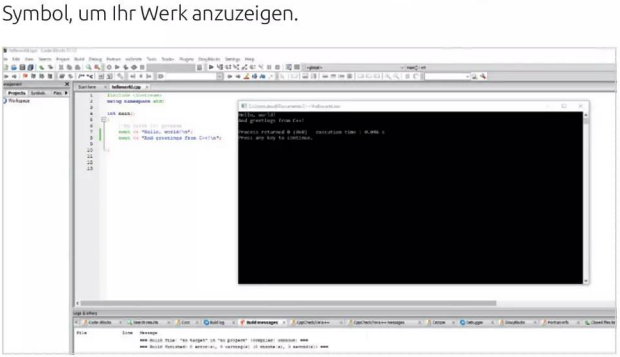


SCHRITT 10

Fügen Sie das Semikolon wieder ein und geben Sie unter der cout-Zeile eine neue Zeile ein:

```
cout << "And greetings from C++!\n";
```

Das \n fügt unter der letzten Zeile des ausgegebenen Textes eine neue Zeile hinzu. Klicken Sie auf das Zahnrad/Wiedergabetaste-Symbol, um Ihr Werk anzuzeigen.



Kommentare

Obwohl Kommentare in den kleinen Codezeilen, die zusammen ein Spiel, eine Anwendung oder sogar ein gesamtes Betriebssystem ergeben, als untergeordnetes Element erscheinen, sind sie in Wirklichkeit einer der wichtigsten Faktoren.

KOMMENTARE UND IHRE BEDEUTUNG

Kommentare im Code sind praktisch Beschreibungen für die Benutzer und erklären detailliert, was der Code an dieser bestimmten Stelle bezweckt. Sie klingen nicht besonders wichtig, aber ein Code ohne Kommentare ist einer der vielen frustrierenden Bereiche des Programmierens, sowohl für Profis als auch für Anfänger.

Kurz gesagt sollte der gesamte Code so kommentiert werden, dass der Zweck einer Zeile, eines Abschnitts oder einzelner Elemente effektiv beschrieben wird. Gewöhnen Sie sich an, so viel wie möglich zu kommentieren. Stellen Sie sich dabei vor, dass jemand, der sich mit Programmieren nicht auskennt, Ihren Code abruft und durch das Lesen Ihrer Kommentare nachvollziehen kann, was im Code passiert.

In einem professionellen Umfeld sind Kommentare für den Erfolg des Codes und letztendlich des Unternehmens von entscheidender Bedeutung. In einem Unternehmen arbeiten viele Programmierer in Teams, zusammen mit Technikern, anderen Entwicklern, Hardwareanalysten usw. Wenn Sie Teil des Teams sind, das eine maßgeschneiderte Software für das Unternehmen schreibt, können Ihre Kommentare Ihrem Team viel Zeit einsparen, falls etwas schief geht, und ein Kollege das Problem lokalisieren muss.

Versetzen Sie sich in die Lage einer Person, deren Aufgabe es ist, herauszufinden, was mit einem Programm nicht stimmt. Das Programm umfasst mehr als 800 000 Codezeilen, die auf verschiedene Module verteilt sind. Sie werden da schnell die Hilfe der ursprünglichen Programmierer in Form guter Kommentare zu schätzen wissen.

Die besten Kommentare sind immer präzise und verknüpfen den Code auf logische Weise. Dabei wird genau beschrieben, was passiert, wenn das Programm eine bestimmte Zeile oder bestimmten Abschnitt erreicht. Sie müssen nicht jede Zeile kommentieren. Zum Beispiel erfordert if x==0 keinen Kommentar, der erläutert, das x gleich 0 ist; dies ist für den Leser offensichtlich.

Wenn aber x gleich 0 das Programm für den Benutzer drastisch verändert, z. B. wenn bei einem Spiel Leben verloren gehen, muss dies sicherlich kommentiert werden.

Selbst wenn es sich um Ihren eigenen Code handelt, sollten Sie Kommentare so schreiben, als würden Sie ihn öffentlich mit anderen teilen. Auf diese Weise können Sie jederzeit zu diesem Code zurückkehren und verstehen, was Sie getan haben, wo Sie einen Fehler gemacht haben oder was hervorragend funktioniert hat.

Kommentare sind bewährte Verfahren, und wenn Sie erst einmal verstehen, wie Sie einen notwendigen Kommentar hinzufügen, werden sie schnell zur Selbstverständlichkeit.

```
        DEFB    60h,61h,32h,4Ch,4Dh,32h,4Ch,99h,32h,4Ch,4Dh,32h,4Ch,4Dh
        DEFB    32h,4Ch,99h,32h,5Bh,5Ch,32h,56h,57h,32h,33h,0CDh,32h,33h
        DEFB    34h,32h,33h,34h,32h,33h,0CDh,32h,40h,41h,32h,66h,67h,64h
        DEFB    66h,67h,32h,72h,73h,64h,4Ch,4Dh,32h,56h,57h,32h,80h,0CBh
        DEFB    19h,80h,0,19h,80h,81h,32h,80h,0CBh,0FFh
T858C:
        DEFB    80h,72h,66h,60h,56h,66h,56h,56h,51h,60h,51h,51h,56h,66h
        DEFB    56h,56h,80h,72h,66h,60h,56h,66h,56h,56h,51h,60h,51h,51h
        DEFB    56h,56h,56h,56h,80h,72h,66h,60h,56h,66h,56h,56h,51h,60h
        DEFB    51h,51h,56h,66h,56h,56h,80h,72h,66h,60h,56h,66h,56h,40h
        DEFB    56h,66h,80h,66h,56h,56h,56h,56h
;
; Game restart point
;
START:  XOR     A
        LD      (SHEET),A
        LD      (KEMP),A
        LD      (DEMO),A
        LD      (B845B),A
        LD      (B8458),A
        LD      A,2             ;Initial lives count
        LD      (NOMEN),A
        LD      HL,T845C
        SET     0,(HL)
        LD      HL,SCREEN
        LD      DE,SCREEN+1
        LD      BC,17FFh        ;Clear screen image
        LD      (HL),0
        LDIR
        LD      HL,0A000h       ;Title screen bitmap
        LD      DE,SCREEN
        LD      BC,4096
        LDIR
        LD      HL,SCREEN + 800h + 1*32 + 29
        LD      DE,MANDAT+64
        LD      C,0
        CALL    DRWFIX
        LD      HL,0FC00h       ;Attributes for the last room
        LD      DE,ATTR         ;(top third)
        LD      BC,256
        LDIR
        LD      HL,09E00h       ;Attributes for title screen
        LD      BC,512          ;(bottom two-thirds)
        LDIR
        LD      BC,31
        DI
        XOR     A
R8621:
        IN      E,(C)
        OR      E
        DJNZ    R8621   ;$-03
        AND     20h
        JR      NZ,R862F ;$+07
        LD      A,1
        LD      (KEMP),A
R862F:
        LD      IY,T846E
        CALL    C92DC
        JP      NZ,L8684
        XOR     A
        LD      (EUGHGT),A
```

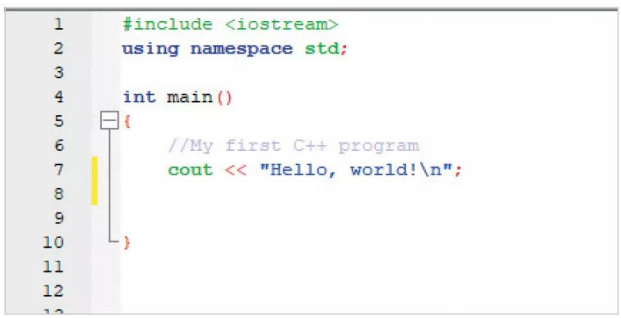
C++-KOMMENTARE

In C++ wird mithilfe eines doppelten Schrägstrichs „//" oder eines Schrägstrichs mit einem Stern „/*" kommentiert. Sie haben bereits einige kurze Beispiele gesehen, hier erfahren Sie nun, wie sie funktionieren.

SCHRITT 1

Im Beispiel des Hello World-Codes können Sie leicht verschiedene Code-Abschnitte mit dem doppelten Schrägstrich kommentieren:

```
//My first C++ program
cout << "Hello, world!\n";
```

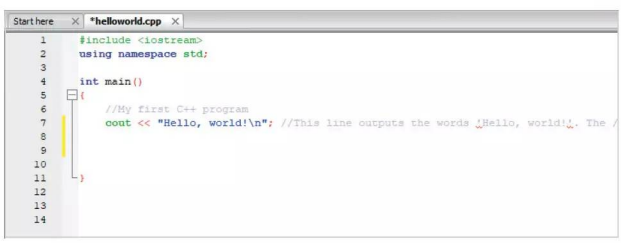


SCHRITT 2

Sie können Kommentare aber auch ans Ende einer Codezeile einfügen, um deutlicher zu erklären, was passiert:

```
cout << "Hello, world!\n"; //This line outputs the words 'Hello, world!'. The \n denotes a new line.
```

Beachten Sie, dass Sie am Ende eines Kommentars kein Semikolon einfügen müssen, da es sich um eine Zeile im Code handelt, die vom Compiler ignoriert wird.



SCHRITT 3

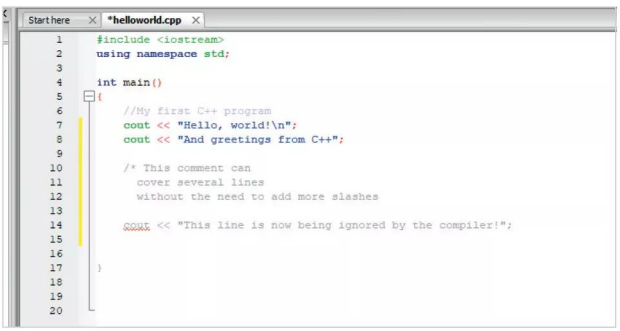
Mit dem Schrägstrich und dem Sternchen können Sie mehre Zeilen auskommentieren:

```
/* This comment can
     cover several lines
     without the need to add more slashes */
```

Vergessen Sie nur nicht, den Blockkommentar mit dem gegenüberliegenden Sternchen und dem Schrägstrich abzuschließen.

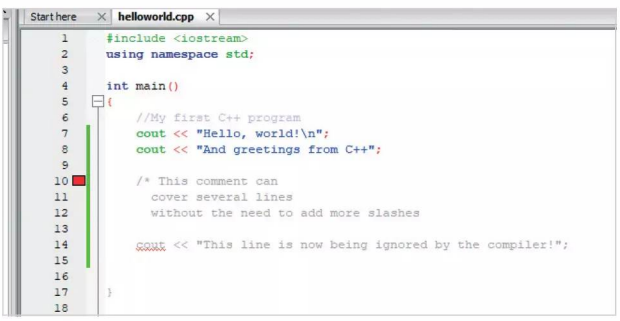
SCHRITT 4

Seien Sie sorgsam bei der Eingabe von Kommentaren, insbesondere bei Blockkommentaren. Es kann schnell passieren, dass man das schließende Sternchen und den Schrägstrich vergisst, wodurch der Code, der in den Kommentarblock fällt, ignoriert wird.



SCHRITT 5

Wenn Sie versuchen, den Code zu erstellen und auszuführen, erhalten Sie eine Fehlermeldung, z. B. eine fehlende geschweifte Klammer „}", um den Codeblock zu beenden. Wenn Sie den Fehler mehrmals gemacht haben, kann es zeitaufwendig sein, ihn zu beheben. Zum Glück hilft die Farbcodierung in Code::Blocks bei der Erkennung von Code-Kommentaren.



SCHRITT 6

Wenn Sie Blockkommentare verwenden, empfiehlt es sich in C++, jeder neuen Zeile des Kommentarblocks ein Sternchen hinzuzufügen. Dies hilft Ihnen auch dabei, sich daran zu erinnern, den Kommentarblock zu schließen, bevor Sie mit dem Code fortfahren:

```
/* This comment can
 *    cover several lines
 *    without the need to add more slashes */
```

Variablen

Zwischen den Variablen in C++ und Python gibt es einen kleinen Unterschied. In Python können Sie einfach angeben, dass „a" gleich 10 ist, in C++ muss eine Variable jedoch mit ihrem Typ deklariert werden, bevor sie verwendet werden kann.

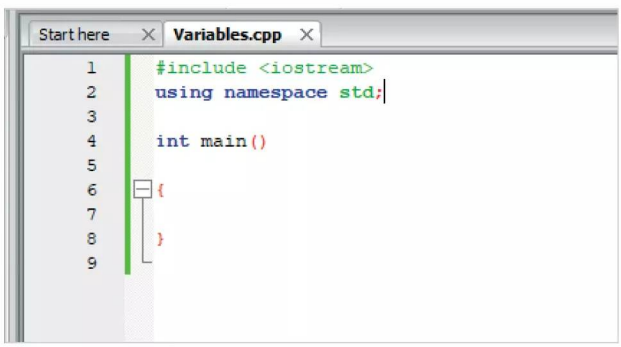
DIE DEKLARATION VON VARIABLEN

Sie können eine C++-Variable mithilfe von Anweisungen im Code deklarieren. Es gibt verschiedene Arten von Variablen, die Sie deklarieren können. Hier erfahren Sie, wie es geht.

SCHRITT 1

Öffnen Sie eine neue, leere C++-Datei und geben Sie die üblichen Code-Header ein:

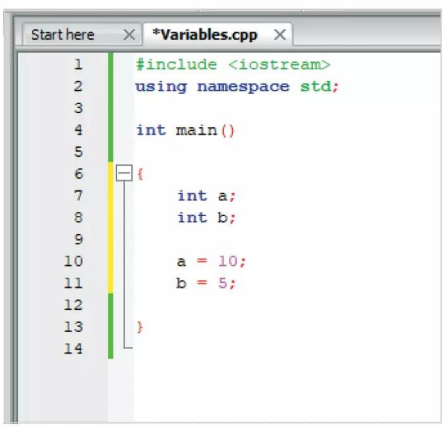
```
#include <iostream>
using namespace std;

int main()
{

}
```



SCHRITT 2

Beginnen Sie, indem Sie die beiden Variablen a und b mit den Werten 10 bzw. 5 erstellen. Sie können die Variablen mit dem Datentyp int deklarieren. Geben Sie Folgendes in die geschweiften Klammern ein:

```
int a;
int b;

a = 10;
b = 5;
```

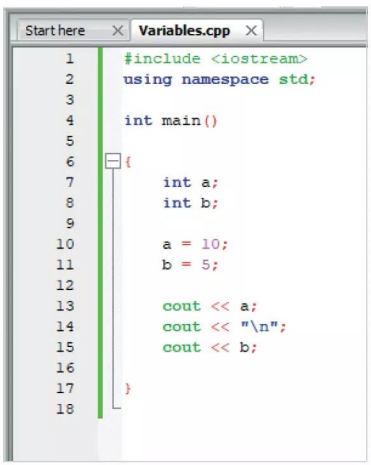


SCHRITT 3

Sie können den Code erstellen und ausführen, er wird jedoch lediglich die Werte 10 und 5 in den Integer-Variablen a und b speichern. Um den Inhalt der Variablen auszugeben, fügen Sie Folgendes hinzu:

```
cout << a;
cout << "\n";
cout << b;
```

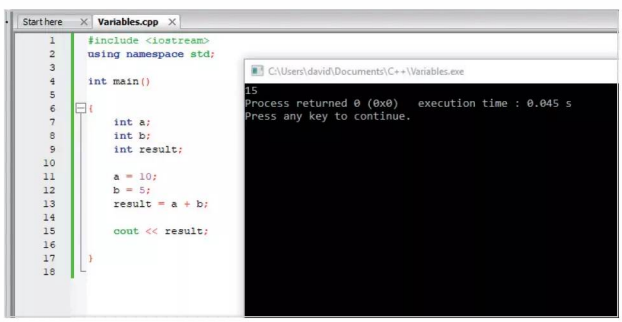
Der cout << "\ n";-Teil setzt lediglich zwischen der Ausgabe von 10 und 5 eine neue Zeile ein.



SCHRITT 4

Sie können natürlich auch eine neue Variable deklarieren, sie „result" nennen und ein paar einfache Rechenaufgaben ausgeben. Fügen Sie das Folgende in den Code ein, wie in der Abbildung zu sehen ist:

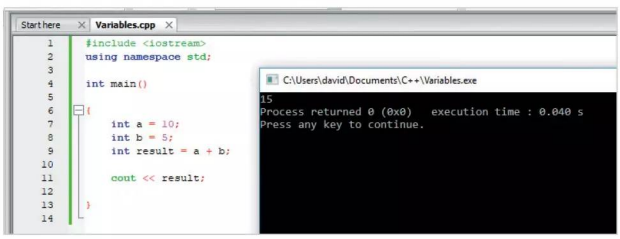
```
int result;

result = a + b;
cout << result;
```



SCHRITT 5

Sie können einer Variablen einen Wert zuweisen, sobald Sie sie deklariert haben. Der von Ihnen eingegebene Code könnte stattdessen so aussehen:

```
int a = 10;
int b = 5;
int result = a + b;

cout << result;
```



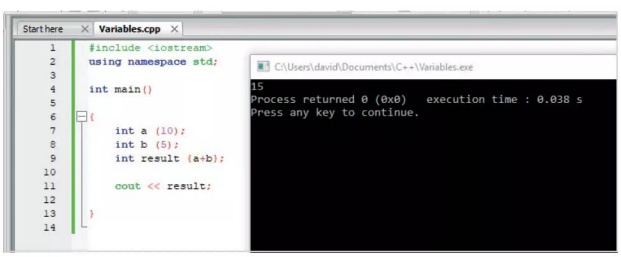
SCHRITT 6

Speziell in C++ können Sie einer deklarierten Variablen auch wie folgt Werte zuweisen:

```
int a (10);
int b (5);
```

und dann, ausgehend vom C++ 2011-Standard, mit geschweiften Klammern:

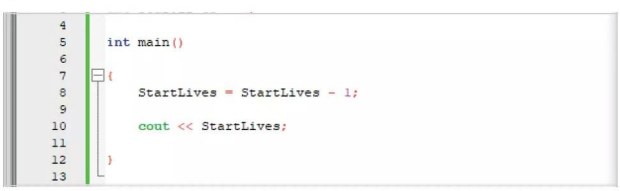
```
int result {a+b};
```



SCHRITT 7

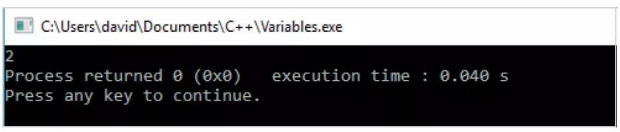
Sie können auch globale Variablen erstellen. Dies sind Variablen, die außerhalb einer Funktion deklariert und von jeder Funktion innerhalb des gesamten Codes verwendet werden können. Bisher haben Sie lokale Variablen verwendet, d. h. Variablen, die in der Funktion verwendet werden. Hier ein Beispiel:

```
#include <iostream>
using namespace std;
int StartLives = 3;

int main ()

{
      startLives = StartLives - 1;
      cout << StartLives;
}
```



SCHRITT 8

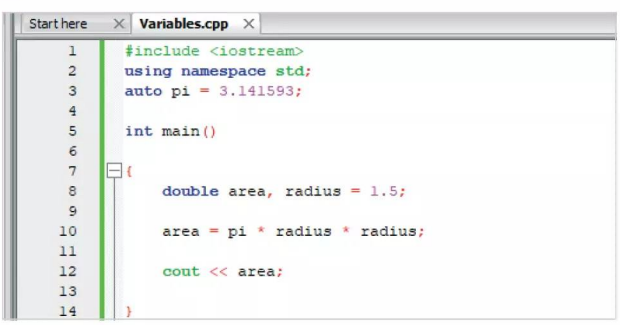
Im vorherigen Schritt wurde die globale Variable StartLives erstellt. In z. B. einem Spiel enthält oder verliert ein Spieler Leben, je nachdem, wie gut oder wie schlecht er spielt. Wenn der Spieler das Spiel neu startet, kehrt StartLives in den Standardzustand zurück: 3. Hier haben wir 3 Leben zugewiesen, dann 1 abgezogen, sodass 2 Leben übrig bleiben.



SCHRITT 9

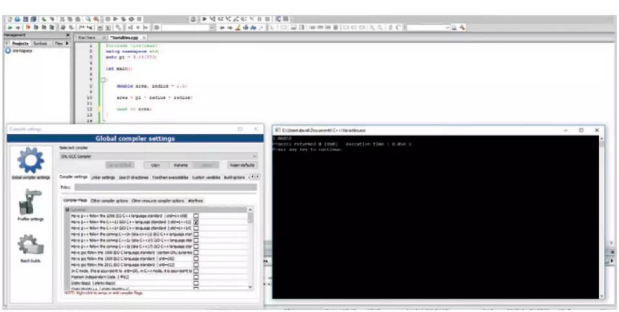
Der moderne C++-Compiler ist weitaus intelligenter als die meisten Programmierer es ihm zutrauen würden. Es gibt zahlreiche Datentypen, die Sie für Variablen deklarieren können, Sie können aber auch die auto-Funktion verwenden:

```
#include <iostream>
using namespace std;
auto pi = 3.141593;

int main()

{
    double area, radius = 1.5;

    area = pi * radius * radius;

    cout << area;

}
```



SCHRITT 10

Die auto-Funktion funktioniert aber nur, wenn Sie unter Settings > Compiler das Kontrollkästchen „Have G++ follow the C++11 ISO C++ Language Standard [-std=c++1]" aktivieren. Der neue Datentyp double bedeutet Gleitkommazahl mit doppelter Genauigkeit. Aktivieren Sie C++11 und erstellen und führen Sie den Code aus. Das Ergebnis sollte 7.06858 sein.



Datentypen

Variablen speichern Informationen, die der Programmierer später aufrufen und bei Bedarf auch bearbeiten kann. Variablen sind praktisch reservierte Speicherplätze, in denen die zugewiesenen Werte abhängig vom Datentyp gespeichert werden.

DER WERT DER DATEN

In C++ stehen dem Programmierer viele verschiedene Datentypen zur Verfügung, z. B. Integer, Gleitkommazahl, Boolean, Zeichen usw. Es ist allgemein anerkannt, dass es sieben grundlegende Datentypen gibt, die oft als integrierte primitive Typen bezeichnet werden. Sie können bei Bedarf aber auch eigene Datentypen in Ihrem Code erstellen.

Die sieben grundlegenden Datentypen sind:

TYP	BEFEHL
Integer	`Integer`
Gleitkommazahl	`float`
Zeichen	`char`
Boolean	`bool`
Gleitkommazahl mit doppelter Genauigkeit	`double`
Weite Zeichen	`wchar_t`
Ohne Wert	`void`

Diese Basistypen können mit den folgenden Modifizierern erweitert werden: Long, Short, Signed und Unsigned. Im Grunde bedeutet dies, dass die Modifizierer die Werte für den minimalen und maximalen Bereich für jeden Datentyp erweitern können. Der Datentyp int hat z. B. einen Standardwertebereich von -2147483648 bis 2147483647.

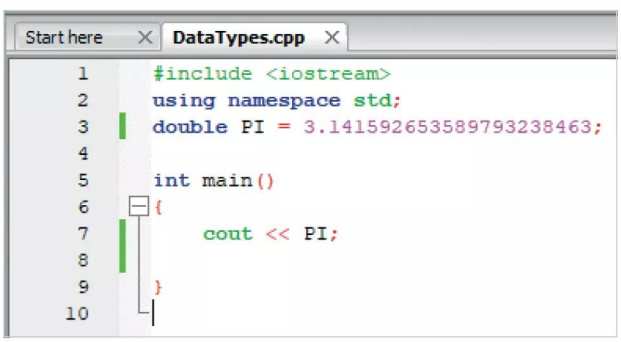
Wird nun ein Modifizierer angewendet, ändert sich der Bereich:

Unsigned int = 0 bis 4294967295
Signed int = -2147483648 bis 2147483647
Short int = -32768 bis 32767
Unsigned Short int = 0 bis 65535
Signed Short int = -32768 bis 32767
Long int = -2147483647 bis 2147483647
Signed Long int = -2147483647 bis 2147483647
Unsigned Long int = 0 bis 4294967295

Natürlich können Sie den Basistyp auch ohne Modifizierer verwenden, da für jeden Datentyp eine große Bandbreite vorhanden ist. Es gilt in C++ jedoch als gute Praxis, wenn möglich die Modifizierer zu verwenden.

Es gibt jedoch Probleme bei der Verwendung der Modifizierer. Der Befehl double stellt einen doppelten Gleitkommawert dar, den Sie für genaue Zahlen verwenden können. Diese Zahlen sind jedoch nur bis zur fünfzehnten Dezimalstelle genau. In C++ gibt es auch mit der cout-Funktion ein Problem bei der Anzeige solcher Zahlen, denn cout gibt standardmäßig nur die ersten fünf Dezimalstellen aus. Sie können dies umgehen, indem Sie eine cout.precision()-Funktion hinzufügen und einen Wert in die Klammern einfügen, aber selbst dann sind Sie weiterhin durch die Genauigkeit der doppelten Gleitkommazahl eingeschränkt. Probieren Sie als Beispiel folgenden Code aus:

```
#include <iostream>
using namespace std;
double PI = 3.141592653589793238463;

int main()
{
    cout << PI;
}
```

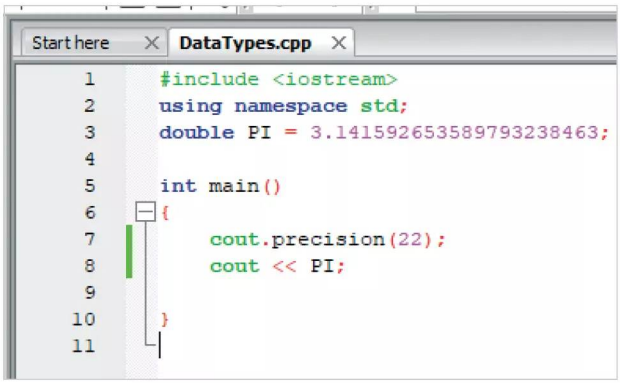


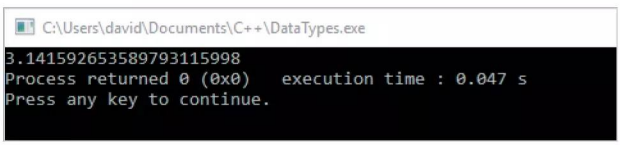
Wenn Sie den Code nun erstellen und ausführen, wird nur 3.14159 ausgegeben, was an den Einschränkungen des cout-Befehls liegt.

Sie können den Code mit der zuvor genannten cout.precision-Funktion für eine höhere Genauigkeit ändern. Mit dem folgenden Code erreichen Sie eine Genauigkeit bis zu 22 Dezimalstellen:

```
#include <iostream>
using namespace std;
double PI = 3.141592653589793238463;

int main()
{
```

```
    cout.precision(22);
    cout << PI;

}
```



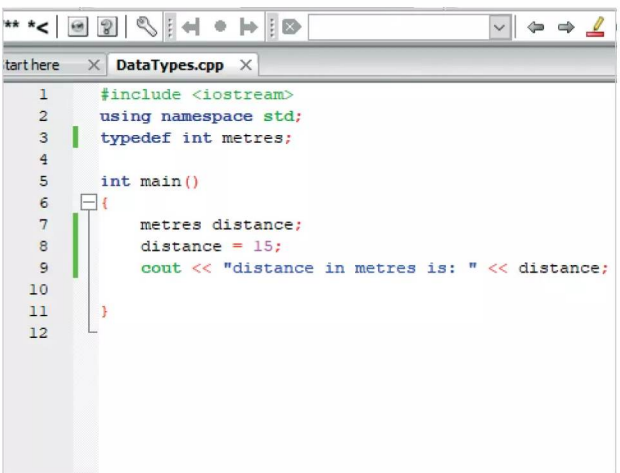


Erstellen und führen Sie den Code erneut aus. Wie Sie im Befehlszeilenfenster sehen, unterscheidet sich die durch die Pi-Variable ausgegebene Zahl von der Nummer, die Sie in C++ in der Variablen angegeben haben. Die Ausgabe gibt den Wert von Pi als 3.141592653589793115998 wieder, wobei die Zahlen ab der fünfzehnten Dezimalstelle jedoch nicht mehr korrekt sind.

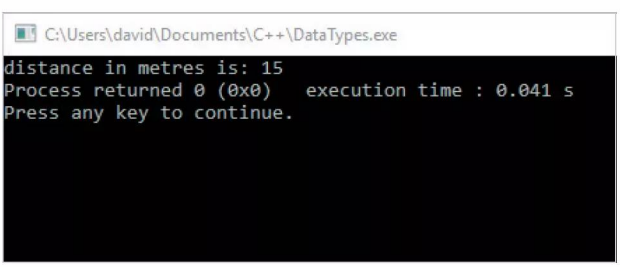
Dies liegt hauptsächlich an der Binärumrechnung im Compiler und dem IEEE 754-Standard mit doppelter Genauigkeit, der 64-Bit-Daten belegt, wovon 52 Bit für die Signifikanz (die signifikanten Stellen in einer Gleitkommazahl) vorgesehen sind, und ca. 3,5 Bits von den Werten 0 bis 9 eingenommen werden. Wenn Sie 53 durch 3,5 dividieren, erhalten Sie 15,$\overline{142857}$, was eine Genauigkeit von 15 Stellen darstellt.

Um ehrlich zu sein, für Code, der auf mehr als fünfzehn Dezimalstellen genau sein muss, würde man C++ nicht verwenden. Man würde eine spezielle wissenschaftliche Sprache nehmen, wobei C++ als Bindeglied zwischen den beiden Sprachen dient.

Mit einem Alias-ähnlichen System namens typedef können Sie Ihre eigenen Datentypen erstellen. Hier ein Beispiel:




```
#include <iostream>
using namespace std;
typedef int metres;

int main()
{
    metres distance;
    distance = 15;
    cout << “distance in metres is: “ << distance;

}
```



Dieser Code erstellt bei der Ausführung einen neuen int-Datentyp namens „metres”. Im Hauptcodeblock gibt es außerdem eine neue Variable namens „distance”, die eine Integer ist. Sie teilen somit dem Compiler im Grunde mit, dass es einen anderen Namen für int gibt. Wir haben der distance-Variablen den Wert 15 zugewiesen, wodurch „distance in metres is 15“ ausgegeben wird.

Für den Anfang mag das alles ein wenig verwirrend klingen, aber je öfter Sie C++ verwenden und Ihren eigenen Code erstellen, desto einfacher wird es.

Strings

Strings sind Objekte, die Zeichen enthalten und diese in Folge darstellen. Sie könnten beispielsweise eine universelle Begrüßung als String in Ihrem Code „Welcome" eingeben, der an beliebiger Stelle im Programm aufgerufen werden kann.

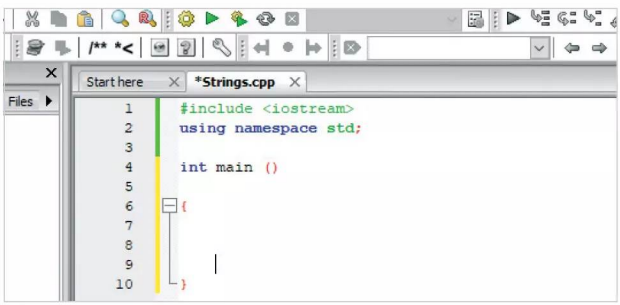
STRINGTHEORIE

Es gibt verschiedene Möglichkeiten, einen String zu erstellen, die aus der ursprünglichen C-Sprache übernommen wurden und von C++ weiterhin unterstützt werden.

SCHRITT 1

Zum Erstellen eines Strings verwenden Sie die char-Funktion. Öffnen Sie eine neue C++-Datei und beginnen Sie mit den üblichen Eingaben:

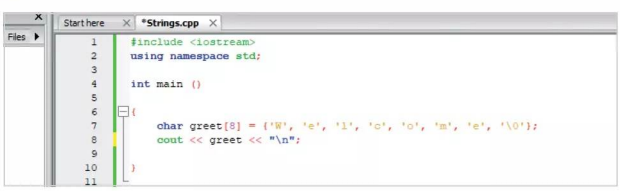
```
#include <iostream>
using namespace std;

int main ()

{

}
```



SCHRITT 2

Ein String kann leicht mit einem Array verwechselt werden. Hier ist ein Array, das mit einem Nullzeichen abgeschlossen werden kann:

```
#include <iostream>
using namespace std;

int main ()

{
    char greet[8] = {'W', 'e', 'l', 'c', 'o', 'm',
'e', '\0'};
    cout << greet << "\n";

}
```



SCHRITT 3

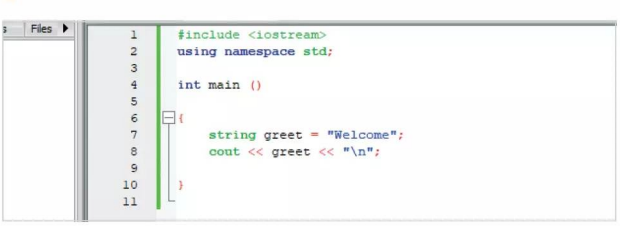
Erstellen und führen Sie den Code aus. Auf dem Bildschirm wird nun „Welcome" angezeigt. Dies ist jedoch kein String. Ein String ist eine Klasse, die Objekte definiert, die als Zeichenstrom dargestellt werden können und nicht wie ein Array abgeschlossen werden müssen. Der Code kann daher folgendermaßen dargestellt werden:

```
#include <iostream>
using namespace std;

int main ()

{
    char greet[] = "Welcome";
    cout << greet << "\n";

}
```

SCHRITT 4

In C++ gibt es eine String-Funktion, die auf ähnliche Weise funktioniert. Verwenden Sie erneut den Begrüßungscode und geben Sie Folgendes ein:

```
#include <iostream>
using namespace std;

int main ()

{
    string greet = "Welcome";
    cout << greet << "\n";

}
```



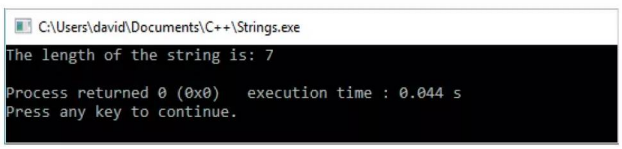
SCHRITT 5

Es gibt auch viele verschiedene Operationen, die Sie mit der String-Funktion anwenden können. Um z. B. die Länge eines Strings zu ermitteln, können Sie Folgendes eingeben:

```
#include <iostream>
using namespace std;

int main ()

{
    string greet = “Welcome”;
    cout << “The length of the string is: “;
    cout << greet.size() << “\n”;

}
```

SCHRITT 6

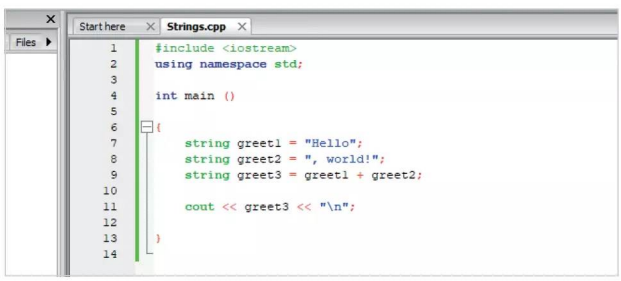
Mit greet.size() haben wir die Länge und die Anzahl der vorhandenen Zeichen des String-Inhalts ausgegeben. Wenn Sie Ihren String anders als „greet" benannt haben, müssen Sie den Befehl natürlich entsprechend ändern. Es muss immer Stringname.Operation lauten. Erstellen und führen Sie den Code aus, um die Ergebnisse anzuzeigen.



SCHRITT 7

Sie können Strings auch zusammenfügen oder kombinieren, um längere Strings zu bilden:

```
#include <iostream>
using namespace std;

int main ()

{
    string greet1 = “Hello”;
    string greet2 = “, world!”;
    string greet3 = greet1 + greet2;

    cout << greet3 << “\n”;

}
```



SCHRITT 8

Sie können auch einen Integer einfügen und etwas speichern, das mit dem String zu tun hat. In diesem Beispiel haben wir „int length" erstellt, womit wir das Ergebnis von string.size() speichern und ausgeben:

```
#include <iostream>
using namespace std;

int main ()

{
    int length;
    string greet1 = “Hello”;
    string greet2 = “, world!”;
    string greet3 = greet1 + greet2;

    length = greet3.size();

    cout << “The length of the combined strings
is: “ << length << “\n”;

}
```

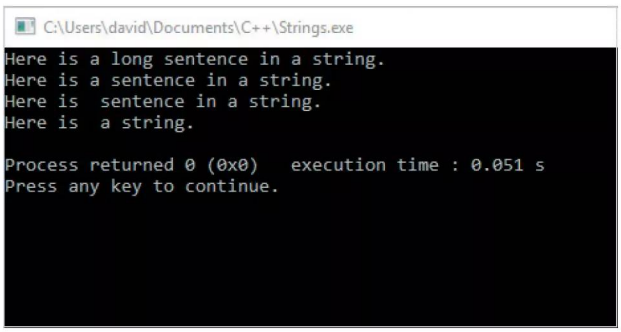
SCHRITT 9

Mit den verfügbaren Operationen, die Sie mit der String-Funktion erhalten, können Sie den Inhalt eines Strings bearbeiten. Um z. B. Zeichen aus einem String zu entfernen, könnten Sie Folgendes verwenden:

```
#include <iostream>
using namespace std;

int main ()

{
  string strg (“Here is a long sentence in a
string.”);
  cout << strg << ‘\n’;

  strg.erase (10,5);
  cout << strg << ‘\n’;

  strg.erase (strg.begin()+8);
  cout << strg << ‘\n’;

  strg.erase (strg.begin()+9, strg.end()-9);
  cout << strg << ‘\n’;

}
```

SCHRITT 10

Es lohnt sich, etwas Zeit mit den Zahlen zu verbringen, die die Zeichenpositionen im String darstellen. Hin und wieder kann es passieren, dass man daneben liegt, aber durch Übung wird man bekanntermaßen zum Meister. Der untere Screenshot zeigt das Ergebnis des Codes an.



C++ – Mathematik

Das Programmieren ist mathematischer Natur, und wie Sie vielleicht vermuten, gibt es eine Menge integrierter Möglichkeiten für die Ausführung intensiver Mathematikprogramme. C++ hat für Programmierer, die mathematische Modelle in ihre Codes implementieren möchten, viel zu bieten und kann äußerst komplex oder relativ einfach sein.

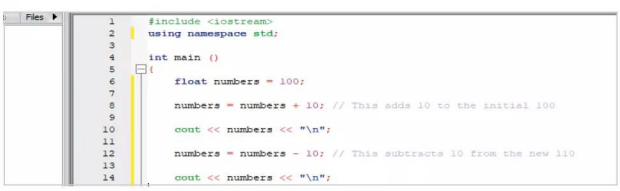
C++ = MC²

Die grundlegenden mathematischen Symbole gelten in C++ genauso wie in den meisten anderen Programmiersprachen. Mit der C++ Mathematikbibliothek können Sie jedoch auch Quadratwurzeln, Potenzen usw. berechnen.

SCHRITT 1

Die mathematischen Operationen von C++ folgen den gleichen Mustern wie denen, die in der Schule gelehrt werden, wobei Multiplikation und Division Vorrang vor Addition und Subtraktion haben. Sie können das aber ändern. Erstellen Sie zunächst eine neue Datei und geben Sie Folgendes ein:

```
#include <iostream>
using namespace std;

int main ()
{
    float numbers = 100;

    numbers = numbers + 10; // This adds 10 to the initial 100

    cout << numbers << "\n";

    numbers = numbers - 10; // This subtracts 10 from the new 110

    cout << numbers << "\n";

}
```



SCHRITT 2

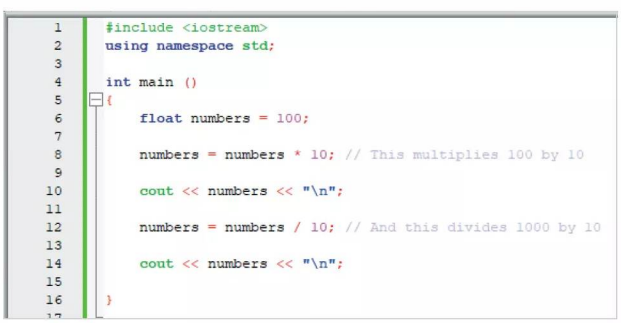
Beachten Sie, dass wir für die Zahlenvariable float verwendet haben. Sie können zwar auch Integer verwenden, müssen aber, wenn Sie plötzlich Dezimalzahlen benutzen sollten, je nach benötigter Genauigkeit zu float oder double wechseln. Führen Sie den Code aus, um die Ergebnisse anzuzeigen.

```
C:\Users\david\Documents\C++\Maths.exe
110
100

Process returned 0 (0x0)   execution time : 0.043 s
Press any key to continue.
```

SCHRITT 3

Multiplikation und Division können als solche angewendet werden:

```
#include <iostream>
          using namespace std;

int main ()
{
    float numbers = 100;

    numbers = numbers * 10; // This multiplies 100 by 10

    cout << numbers << "\n";

    numbers = numbers / 10; // And this divides 1000 by 10

    cout << numbers << "\n";

}
```



SCHRITT 4

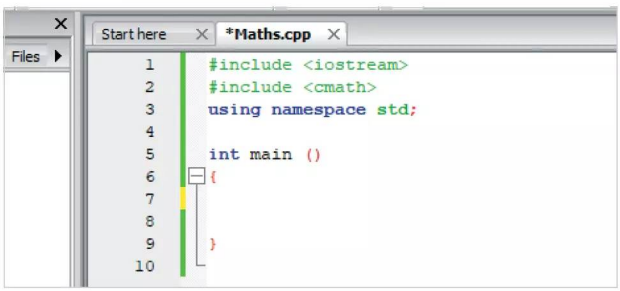
Führen Sie erneut den Code aus und schauen Sie sich die Ergebnisse an. Dies ist zwar nicht sehr spannend, ist aber ein Einstieg in die Mathematik unter C++. Da wir einen float verwenden, können Sie mit dem Code experimentieren und mit Dezimalstellen multiplizieren, dividieren, addieren und subtrahieren.

```
C:\Users\david\Documents\C++\Maths.exe
1000
100

Process returned 0 (0x0)   execution time : 0.050 s
Press any key to continue.
```

SCHRITT 5

Interessant wird es, wenn Sie die C++ Mathematikbibliothek aufrufen. In diesem Header befinden sich Dutzende mathematischer Funktionen und weitere Operationen, von der Berechnung des Cosinus und Tangenten bis hin zum Wert von Pi. Sie können den Header wie folgt aufrufen:

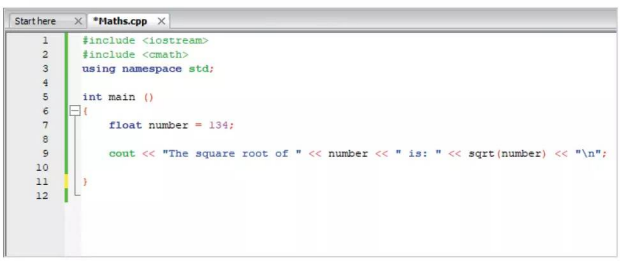
```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{

}
```



SCHRITT 6

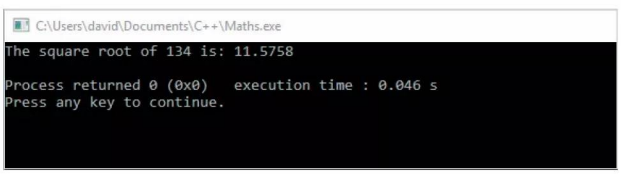
Beginnen Sie, indem Sie die Quadratwurzel einer Zahl ermitteln:

```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{
    float number = 134;

    cout << “The square root of “ << number << “ is: “ << sqrt(number) << “\n”;

}
```



SCHRITT 7

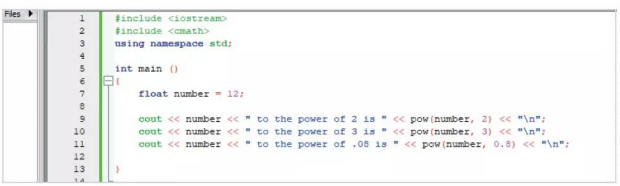
Hier haben wir einen neuen float namens number erstellt. Die Funktion sqrt(number) zeigt die Quadratwurzel von 134 an, den Wert der number-Variablen. Erstellen und führen Sie den Code aus. Die Antwort wird 11.5758 sein.



SCHRITT 8

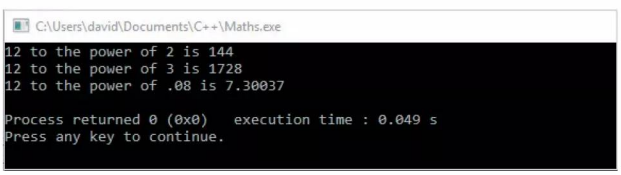
Potenzen können wie folgt berechnet werden:

```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{
    float number = 12;

    cout << number << “ to the power of 2 is “ << pow(number, 2) << “\n”;
    cout << number << “ to the power of 3 is “ << pow(number, 3) << “\n”;
    cout << number << “ to the power of .08 is “ << pow(number, 0.8) << “\n”;

}
```



SCHRITT 9

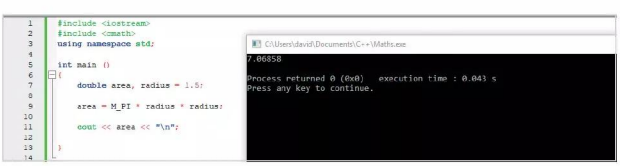
Hier haben wir einen float namens „number” mit dem Wert 12 erstellt. Die Berechnung erfolgt über pow (variable, power). Natürlich können Sie Potenzen und Quadratwurzeln auch ohne Variablen berechnen. Zum Beispiel gibt pow (12, 2) den gleichen Wert aus wie die erste cout-Zeile im Code.



SCHRITT 10

Der Wert von Pi wird auch in der cmath-Bibliothek gespeichert. Sie kann mit der Funktion M_PI aufgerufen werden. Geben Sie cout << M_PI in den Code ein und Sie erhalten 3.14159. Sie können damit auch Berechnungen ausführen:

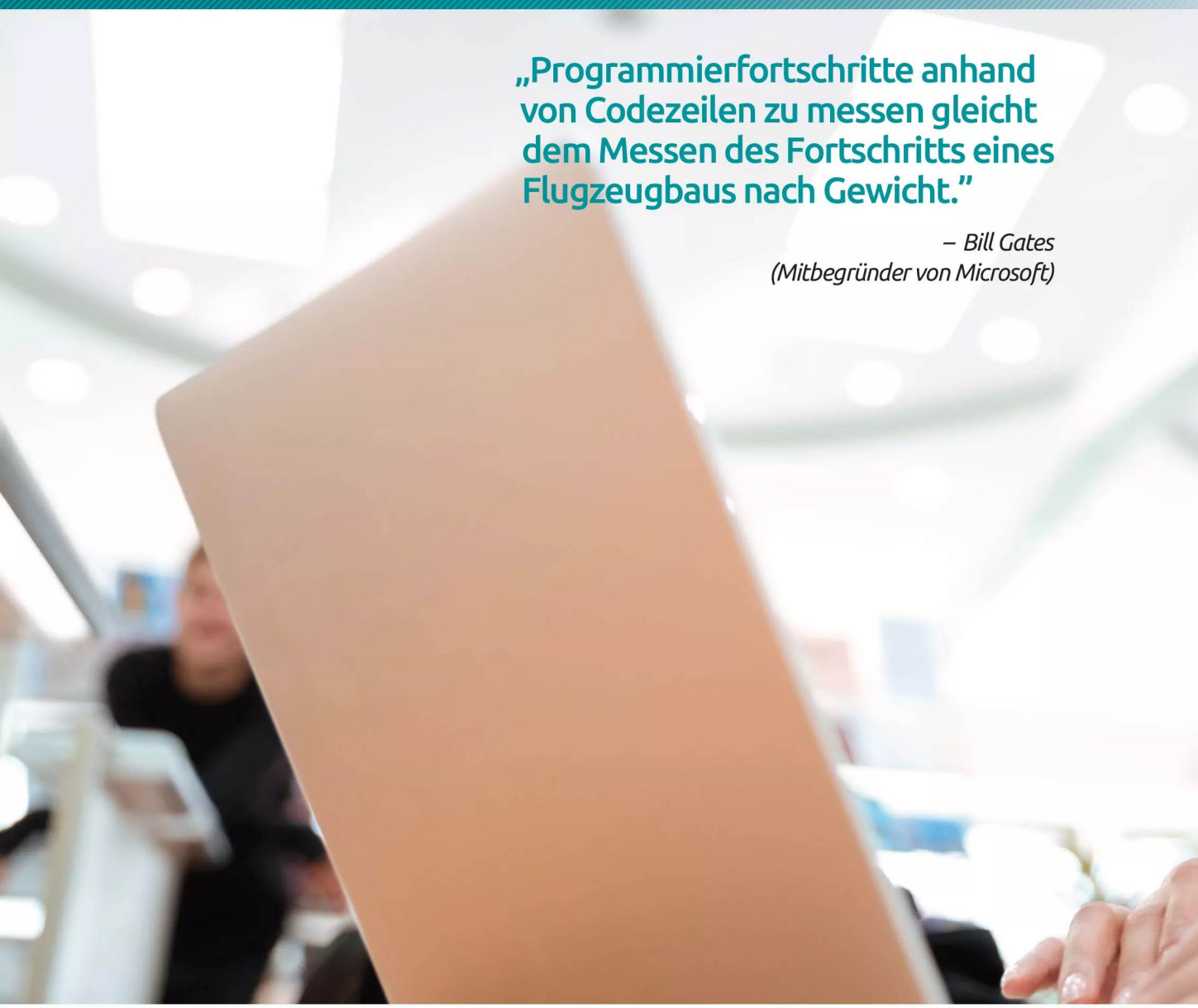
```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{
    double area, radius = 1.5;

    area = M_PI * radius * radius;

    cout << area << “\n”;

}
```

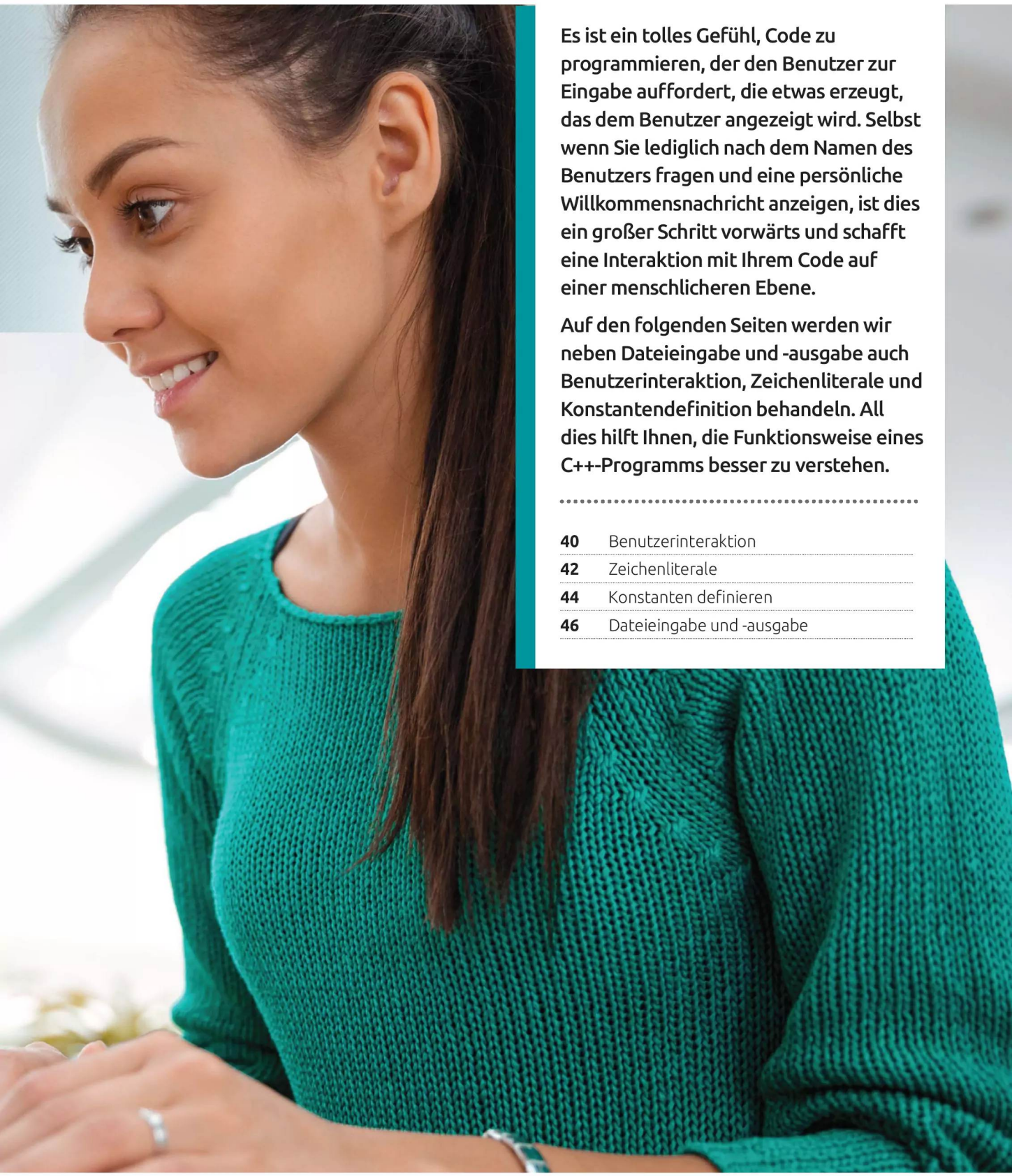


C++ – Eingabe/ Ausgabe

„Programmierfortschritte anhand von Codezeilen zu messen gleicht dem Messen des Fortschritts eines Flugzeugbaus nach Gewicht."

– Bill Gates
(Mitbegründer von Microsoft)





Es ist ein tolles Gefühl, Code zu programmieren, der den Benutzer zur Eingabe auffordert, die etwas erzeugt, das dem Benutzer angezeigt wird. Selbst wenn Sie lediglich nach dem Namen des Benutzers fragen und eine persönliche Willkommensnachricht anzeigen, ist dies ein großer Schritt vorwärts und schafft eine Interaktion mit Ihrem Code auf einer menschlicheren Ebene.

Auf den folgenden Seiten werden wir neben Dateieingabe und -ausgabe auch Benutzerinteraktion, Zeichenliterale und Konstantendefinition behandeln. All dies hilft Ihnen, die Funktionsweise eines C++-Programms besser zu verstehen.

Benutzerinteraktion

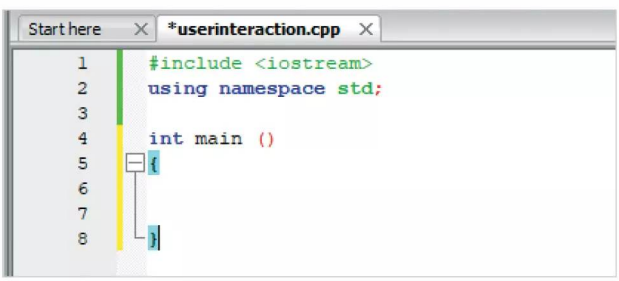
Es gibt nichts Besseres als ein Programm zu erstellen, das auf den Benutzer reagiert. Die Benutzerinteraktion ist einer der am meisten gelehrten Aspekte und man kann damit weitaus mehr machen, als nur den Benutzer mit seinem Namen zu begrüßen.

HELLO, DAVE

Sie haben in unserem Code bereits den Standardausgabestrom cout verwendet. Sie werden nun mit cin, dem Standardeingabestrom, eine Benutzerantwort anfordern.

SCHRITT 1

Alles, was der Benutzer in das Programm eingeben soll, muss irgendwo im Systemspeicher abgelegt werden, damit es abgerufen und verwendet werden kann. Daher muss jede Eingabe zuerst als Variable deklariert werden, damit sie auch vom Benutzer verwendet werden kann. Erstellen Sie zunächst eine leere C++-Datei mit den Headers.



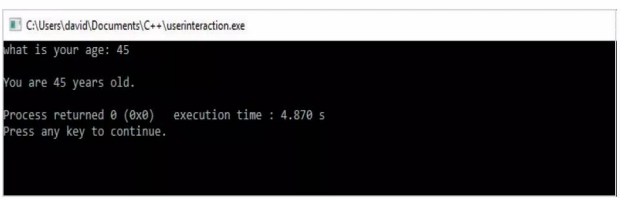
SCHRITT 2

Der Datentyp der Variablen muss der Art der Eingabe entsprechen, die Sie vom Benutzer wünschen. Um z. B. einen Benutzer nach dem Alter zu fragen, würden Sie einen Integer verwenden:

```
#include <iostream>
using namespace std;

int main ()
{
   int age;
   cout << “what is your age: “;
   cin >> age;

   cout <<”\nYou are “ << age << “ years old.\n”;

}
```

SCHRITT 3

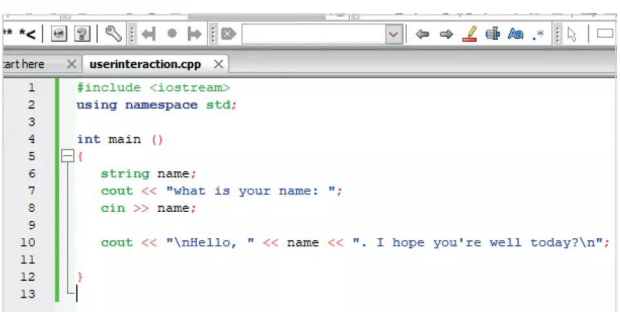
Der cin-Befehl funktioniert auf entgegengesetzte Weise zum cout-Befehl. Mit der ersten cout-Zeile geben Sie, wie durch die Pfeile angezeigt, „What is your age" an den Bildschirm aus. Der cin-Befehl verwendet gegenüberliegende Pfeile, um eine Eingabe anzuzeigen. Die Eingabe kommt in den Integer age und wird im zweiten cout-Befehl aufgerufen. Erstellen und führen Sie den Code aus.



SCHRITT 4

Wenn Sie eine Frage stellen, müssen Sie die Eingabe als String speichern. Um den Benutzer nach dem Namen zu fragen, würden Sie Folgendes eingeben:

```
#include <iostream>
using namespace std;

int main ()
{
   string name;
   cout << “what is your name: “;
   cin >> name;

   cout << “\nHello, “ << name << “. I hope you’re
well today?\n”;

}
```



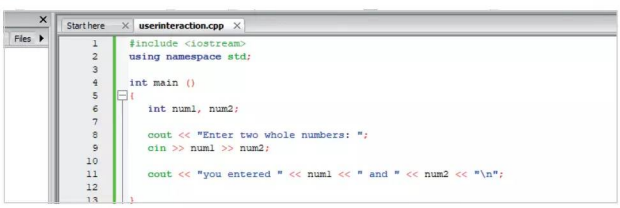
SCHRITT 5

Das Prinzip ist das gleiche wie im vorherigen Code. Die Benutzereingabe, der Name, wird in einem String gespeichert, da sie mehrere Zeichen enthält, und in der zweiten cout-Zeile abgerufen. Solange sich die name-Variable nicht ändert, können Sie sie an beliebiger Stelle in Ihrem Code abrufen.

SCHRITT 6

Sie können Eingabeaufforderungen auch verbinden. Stellen Sie aber sicher, dass Sie über eine gültige Variable verfügen, um die Eingabe zu speichern. Hier möchten wir, dass der Benutzer zwei Ganzzahlen eingibt:

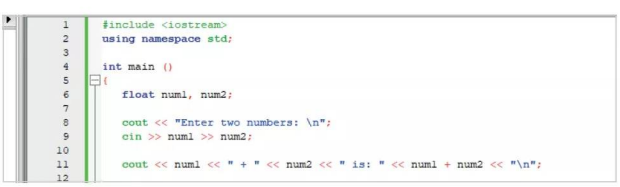
```
#include <iostream>
using namespace std;

int main ()
{
   int num1, num2;

   cout << “Enter two whole numbers: “;
   cin >> num1 >> num2;

   cout << “you entered “ << num1 << “ and “ << num2 << “\n”;

}
```



SCHRITT 7

Sobald Sie eingegebene Daten in einer Variablen gespeichert haben, können Sie sie bearbeiten. Bitten Sie zum Beispiel den Benutzer zwei Zahlen einzugeben, und führen Sie damit Berechnungen aus:

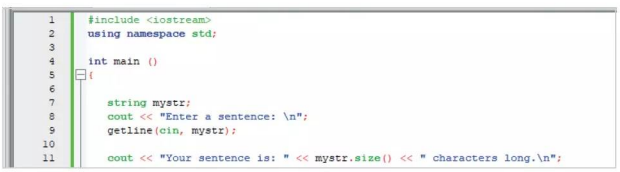
```
#include <iostream>
using namespace std;

int main ()
{
   float num1, num2;

   cout << “Enter two numbers: \n”;
   cin >> num1 >> num2;

   cout << num1 << “ + “ << num2 << “ is: “ << num1 + num2 << “\n”;

}
```



SCHRITT 8

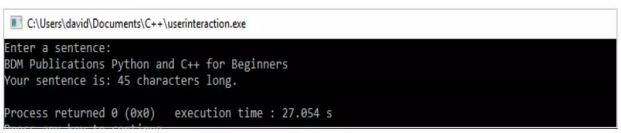
Cin eignet sich gut für die meisten Eingabeaufgaben, hat aber die Einschränkung, dass Leerzeichen immer als Abschlusszeichen gesehen werden, daher ist cin nur für einzelne und nicht mehrere Wörter vorgesehen. Die getline-Funktion greift jedoch cin als erstes Argument und die Variable als zweites auf:

```
#include <iostream>
using namespace std;

int main ()
{

   string mystr;
   cout << “Enter a sentence: \n”;
   getline(cin, mystr);

   cout << “Your sentence is: “ << mystr.size() << “ characters long.\n”;

}
```



SCHRITT 9

Erstellen und führen Sie den Code aus, und geben Sie dann einen Satz mit Leerzeichen ein. Der Code wird die Anzahl der Zeichen lesen. Wenn Sie die getline-Zeile entfernen und durch cin >> mystr ersetzen und es erneut versuchen, zeigt das Ergebnis die Anzahl der Zeichen bis zum ersten Leerzeichen an.



SCHRITT 10

Getline ist gewöhnlich ein Befehl, den neue C++-Programmierer vergessen, einzufügen. Der abschließende Leerraum ist ärgerlich, wenn Sie einfach nicht herausfinden, warum Ihr Code nicht funktioniert. Es ist daher am besten, in Zukunft getline(cin, variable) zu verwenden:

```
#include <iostream>
using namespace std;

int main ()
{

   string name;
   cout << “Enter your full name: \n”;
   getline(cin, name);

   cout << “\nHello, “ << name << “\n”;

}
```

Zeichenliterale

In C++ ist ein Literal ein Objekt oder eine Variable, die nach der Definition im gesamten Code gleich bleibt. Ein Zeichenliteral wird durch einen umgekehrten Schrägstrich definiert, z. B. das \n, das am Ende einer cout-Anweisung eine neue Zeile kennzeichnet.

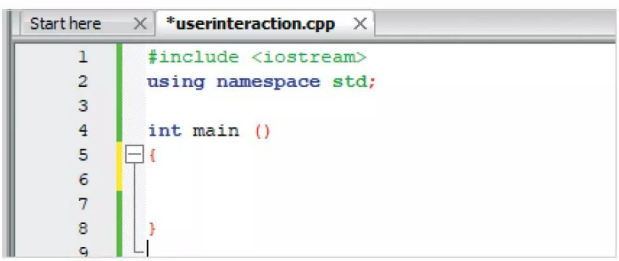
ESCAPE-SEQUENZ

Zeichenliterale werden in z. B. cout-Anweisungen auch als Escape-Sequenzen bezeichnet. Sie ermöglichen die Eingabe eines Zitats, einer Benachrichtigung, einer neuen Zeile und vieles mehr.

SCHRITT 1

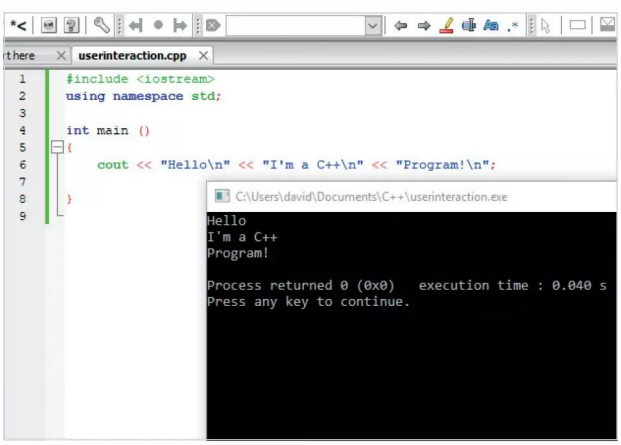
Erstellen Sie eine neue C++-Datei und geben Sie die entsprechenden Header ein:

```
#include <iostream>
using namespace std;

int main ()
{

}
```



SCHRITT 2

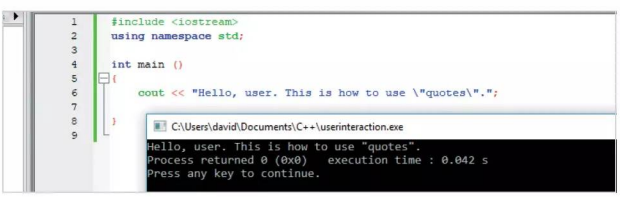
Wie Sie bereits wissen, platziert das Zeichenliteral \n eine neue Zeile, wo immer es eingesetzt wird. Die Zeile **cout << "Hello\n" << "I'm a C++\n" << "Program\n"** gibt drei Textzeilen aus, von denen jede nach dem letzten \n beginnt.



SCHRITT 3

Um in einer cout-Anweisung etwas in Anführungszeichen zu setzen, müssen Sie einen Backslash, einen umgedrehten Schrägstrich nehmen, da die Anweisung selbst bereits Anführungszeichen verwendet:

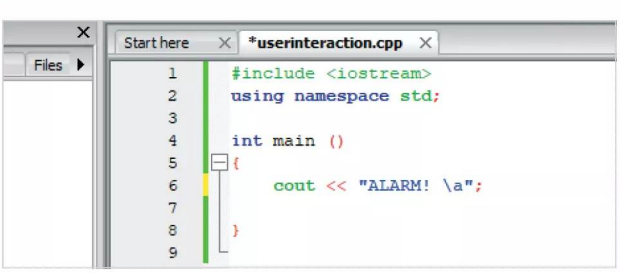
```
#include <iostream>
using namespace std;

int main ()
{
    cout << “Hello, user. This is how to use
\”quotes\”.”;

}
```



SCHRITT 4

Es gibt sogar ein Zeichenliteral, das einen Alarm auslösen kann. In Windows 10 ertönt bei der Verwendung von \a der Benachrichtigungston. Probieren Sie folgenden Code aus und drehen Sie Ihren Lautsprecher auf:

```
#include <iostream>
using namespace std;

int main ()
{
    cout << “ALARM! \a”;

}
```



EINE PRAKTISCHE LISTE

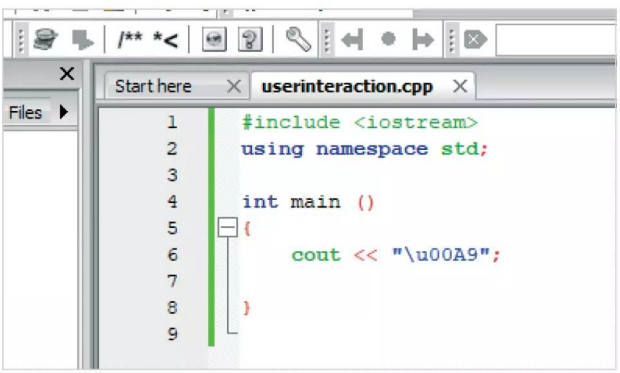
Es stehen zahlreiche Zeichenliterale bzw. Escape-Sequenzen zur Auswahl. Wir dachten daher, diese praktische Liste könnte beim Einfügen von Code sehr gelegen kommen.

ESCAPE-SEQUENZ	ZEICHEN
\\	Backslash (umgekehrter Schrägstrich)
\'	Einfaches Anführungszeichen
\"	Doppelte Anführungszeichen
\?	Fragezeichen
\a	Alarm/Signalton
\b	Rückschritt
\f	Seitenumbruch
\n	Zeilenschaltung
\r	Wagenrücklauf
\t	Horizontaler Tabulator
\v	Vertikaler Tabulator
\0	Null-Zeichen
\uxxxx	Unicode (UTF-8)
\Uxxxxxxxx	Unicode (UTF-16)

UNICODE-ZEICHEN (UTF-8)

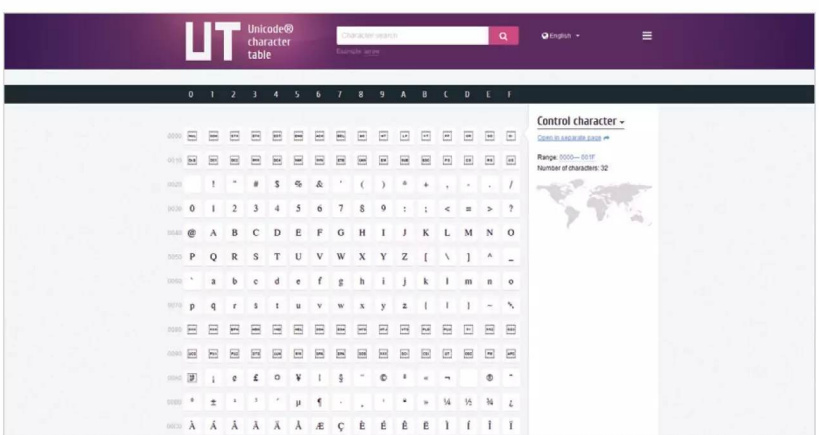
Unicode-Zeichen sind Symbole oder Zeichen, die auf allen Plattformen Standard sind. Das Copyright-Symbol kann z. B. über die Tastatur durch Eingabe des Unicode-Codes gefolgt von ALT + X eingegeben werden : 00A9 Alt + X. In C++ würden Sie Folgendes eingeben:

```
#include <iostream>
using namespace std;

int main ()
{
    cout << "\u00A9";
```



UNICODE-ZEICHENTABELLE

Eine vollständige Liste der verfügbaren Unicode-Zeichen finden Sie unter **www.unicode-table.com/de/**. Halten Sie die Maus über ein Zeichen, um dessen Code zu sehen, der in C++ eingegeben werden muss. Auch wenn die Tabelle zunächst ein wenig überwältigend zu sein scheint, sollten Sie sie mit einem Lesezeichen versehen, da Sie wahrscheinlich darauf zurückgreifen müssen, wenn Sie sich intensiver mit C++ und Zeichenliterale beschäftigen. Die Tabelle zeigt auch Zeichen aus verschiedenen Sprachen an.



Konstanten definieren

Konstanten sind feste Werte im Code. Sie können alle elementaren Datentypen haben, aber ihr Wert bleibt im gesamten Code konstant. Es gibt zwei verschiedene Möglichkeiten, eine Konstante in C++ zu definieren: mit dem Präprozessor #define und const.

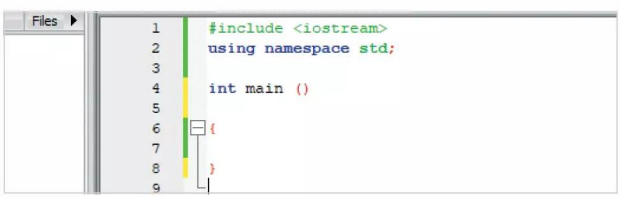
#DEFINE

Die Präprozessoren sind Anweisungen an den Compiler, um die Informationen vorzuverarbeiten und den Code zu kompilieren. Sowohl #include als auch #define sind Präprozessoren.

SCHRITT 1

Sie können mit dem Präprozessor #define die gewünschten Konstanten in Ihrem Code definieren. Erstellen Sie zunächst eine neue C++-Datei mit den üblichen Kopfzeilen:

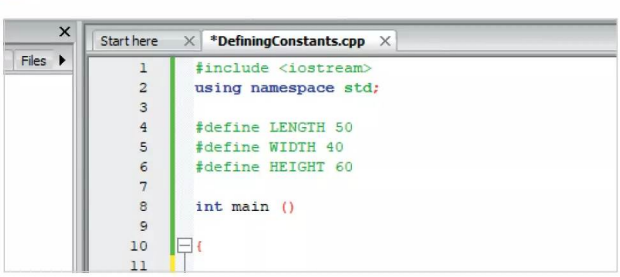
```
#include <iostream>
using namespace std;

int main ()

{

}
```



SCHRITT 2

Nehmen wir nun an, Ihr Code hat drei verschiedene Konstanten: Länge, Breite und Höhe. Sie können diese wie folgt definieren:

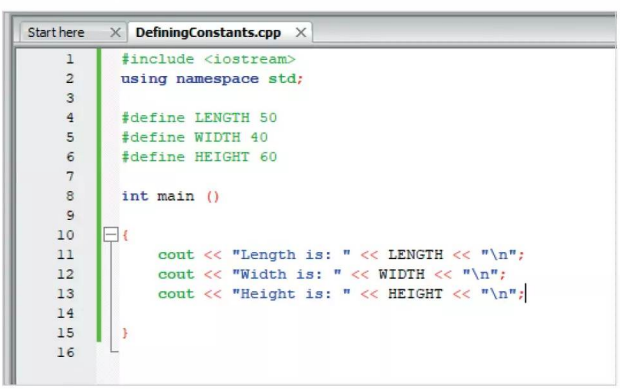
```
#include <iostream>
using namespace std;
#define LENGTH 50
#define WIDTH 40
#define HEIGHT 60

int main ()

{

}
```



SCHRITT 3

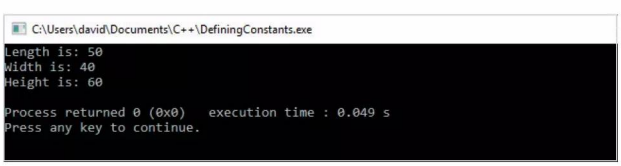
Beachten Sie die Großbuchstaben für definierte Konstanten. Es ist gute Programmierpraxis, alle Konstanten in Großbuchstaben zu definieren. Die hier zugewiesenen Werte sind 50, 40 und 60, und diese rufen wir nun ab:

```
#include <iostream>
using namespace std;

#define LENGTH 50
#define WIDTH 40
#define HEIGHT 60

int main ()

{
    cout << “Length is: “ << LENGTH << “\n”;
    cout << “Width is: “ << WIDTH << “\n”;
    cout << “Height is: “ << HEIGHT << “\n”;
}
```



SCHRITT 4

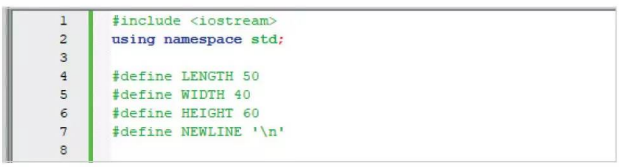
Erstellen Sie den Code und führen Sie ihn aus. Wie erwartet werden die Werte für jede der erstellten Konstanten angezeigt. Wird eine Konstante mit dem Schlüsselwort #define definiert, ist kein Semikolon erforderlich.



SCHRITT 5

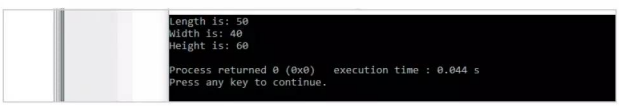
Sie können auch andere Elemente als Konstante definieren. Anstatt für eine neue Zeile \n in die cout-Anweisung einzufügen, können Sie diese z. B. am Anfang des Codes definieren:

```
#include <iostream>
using namespace std;

#define LENGTH 50
#define WIDTH 40
#define HEIGHT 60
#define NEWLINE '\n'

int main ()

{
    cout << "Length is: " << LENGTH << NEWLINE;
    cout << "Width is: " << WIDTH << NEWLINE;
    cout << "Height is: " << HEIGHT << NEWLINE;
}
```



SCHRITT 6

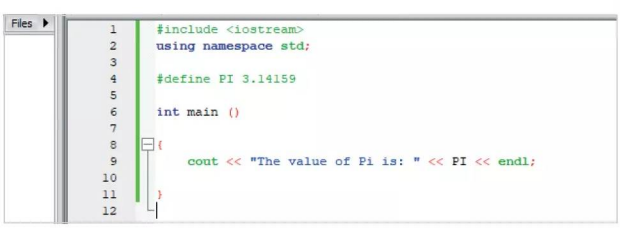
Wenn der Code erstellt und ausgeführt wird, verhält er sich genauso wie zuvor und fügt mit der neuen Konstante NEWLINE eine Zeilenschaltung in die cout-Anweisung ein. Das Erstellen einer Konstante für die Zeilenschaltung ist übrigens nur sinnvoll, wenn sie kürzer als \n oder sogar der endl-Befehl ist.



SCHRITT 7

Das Definieren einer Konstante ist eine gute Möglichkeit, die Basiswerte zu Beginn des Codes zu initialisieren. Sie können z. B. definieren, dass Ihr Spiel drei Leben hat, oder den Wert von Pi, ohne die mathematische Funktionen von C++ aufrufen zu müssen:

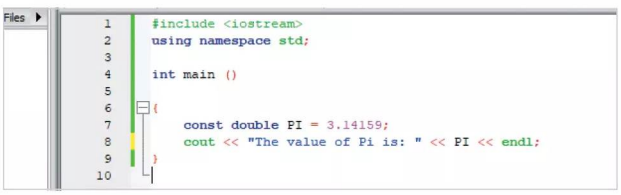
```
#include <iostream>
using namespace std;

#define PI 3.14159

int main ()

{
    cout << "The value of Pi is: " << PI << endl;

}
```



SCHRITT 8

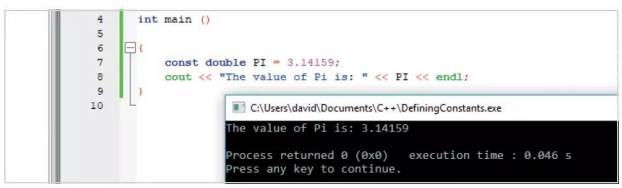
Eine weitere Methode zum Definieren einer Konstante ist das Schlüsselwort const. Verwenden Sie dieses zusammen mit einem Datentyp, einer Variablen und einem Wert: const Datentyp Variable = Wert. Hier nehmen wir Pi als Beispiel:

```
#include <iostream>
using namespace std;

int main ()

{
    const double PI = 3.14159;
    cout << "The value of Pi is: " << PI << endl;
}
```



SCHRITT 9

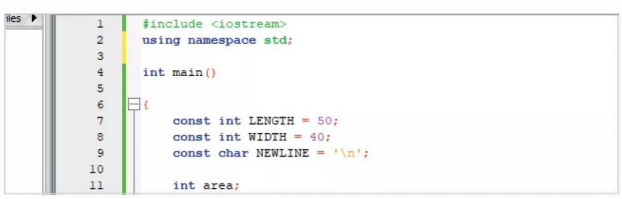
Da Sie const im Hauptcodeblock verwenden, müssen Sie die Zeile mit einem Semikolon beenden. Sie können beides verwenden, solange die Namen und Werte nicht miteinander kollidieren. Es ist erwähnenswert, dass #define keinen Speicher benötigt. Wenn Ihr Code also eine bestimmte Menge Speicher einnimmt, ist #define die beste Wahl.



SCHRITT 10

Const funktioniert ähnlich wie #define. Sie können statische Ganzzahlen und sogar Zeilenschaltungen erstellen:

```
#include <iostream>
using namespace std;

int main()

{
    const int LENGTH = 50;
    const int WIDTH = 40;
    const char NEWLINE = '\n';

    int area;
    area = LENGTH * WIDTH;

    cout << "Area is: " << area << NEWLINE;

}
```



Dateieingabe und -ausgabe

Die Standard-iostream-Bibliothek stellt C++-Programmierern die Ein- und Ausgabefunktionen cin und cout bereit. Um jedoch aus einer Datei lesen und schreiben zu können, müssen Sie eine andere C++-Bibliothek namens fstream verwenden.

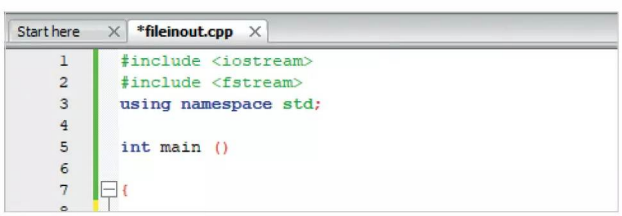
FSTREAMS

Die fstream-Bibliothek hat zwei Hauptdateitypen zum Öffnen, Lesen und Schreiben von bzw. in Dateien: ofstream und ifstream. Hier zeigen wir, wie sie funktionieren.

SCHRITT 1

Die erste Aufgabe besteht darin, eine neue C++-Datei zu erstellen und neben den üblichen Kopfzeilen den neuen fstream-Header einzufügen:

```
#include <iostream>
#include <fstream>
Using namespace std;

int main ()

{

}
```



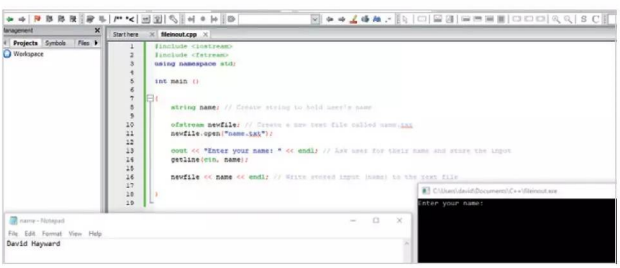
SCHRITT 2

Fragen Sie einen Benutzer nach seinem Namen und schreiben Sie diese Informationen in eine Datei. Sie benötigen den üblichen String, um den Namen zu speichern, und getline, um die Eingabe des Benutzers zu akzeptieren:

```
#include <iostream>
#include <fstream>
using namespace std;

int main ()

{

    string name;

    ofstream newfile;
    newfile.open("name.txt");

    cout << "Enter your name: " << endl;
    getline(cin, name);

    newfile << name << endl;

    newfile.close();

}
```

SCHRITT 3

Im Code des vorherigen Schritts haben Sie einen String namens name erstellt, in dem der vom Benutzer eingegebene Name gespeichert wird. Sie haben auch eine Textdatei namens name.txt erstellt (mit den Zeilen ofstream newfile und newfile.open) und den Benutzer nach seinem Namen gefragt, der gespeichert und in die Datei geschrieben wird.



SCHRITT 4

Um den Inhalt einer Datei zu lesen und auf dem Bildschirm auszugeben, müssen Sie etwas anders vorgehen. Zuerst müssen Sie eine String-Variable erstellen, um den Inhalt der Datei zu speichern (Zeile für Zeile), und dann die Datei öffnen. Verwenden Sie getline, um die Datei Zeile für Zeile zu lesen und diese Zeilen auf dem Bildschirm auszugeben. Schließen Sie anschließend die Datei.

```
    string line;
    ifstream newfile ("name.txt");

    cout << "Contents of the file: " << endl;

    getline(newfile, line);
    cout << line << endl;
    newfile.close();
```

```
#include <iostream>
#include <fstream>
using namespace std;

int main ()

{
    string line;
    ifstream newfile ("name.txt");

    cout << "Contents of the file: " << endl;

    getline(newfile, line);
    cout << line << endl;
    newfile.close();

}
```

SCHRITT 5

Der vorherige Code eignet sich hervorragend zum Öffnen einer Datei mit einer oder zwei Zeilen, aber was ist, wenn mehrere Zeilen vorhanden sind? Hier haben wir eine Textdatei des Gedichts Cimmeria von Robert E Howard geöffnet:

```
    string line;
    ifstream newfile ("c:\\users\\david\\
Documents\\Cimmeria.txt");

    cout << "Cimmeria, by Robert E Howard: \n" <<
endl;

    while (getline(newfile, line))
    cout << line << endl;

    newfile.close();
```

fileinout.cpp

```
#include <iostream>
#include <fstream>
using namespace std;

int main ()

{
    string line;
    ifstream newfile ("c:\\users\\david\\Documents\\Cimmeria.txt");

    cout << "Cimmeria, by Robert E Howard: \n" << endl;

    while (getline(newfile, line))
    cout << line << endl;

    newfile.close();

}
```

SCHRITT 6

Wie Sie sehen, haben wir eine while-Schleife eingefügt, mit der wir uns in Kürze befassen werden. Diese veranlasst, dass, solange es Zeilen gibt, die aus der Textdatei gelesen werden sollen, diese in C++ per getline abgerufen werden. Nachdem alle Zeilen gelesen wurden, wird die Ausgabe auf dem Bildschirm angezeigt und die Datei geschlossen.

```
Cimmeria, by Robert E Howard:

I remember
The dark woods, masking slopes of sombre hills;
The grey clouds' leaden everlasting arch;
The dusky streams that flowed without a sound,
And the lone winds that whispered down the passes.

Vista on vista marching, hills on hills,
Slope beyond slope, each dark with sullen trees,
Bar gaunt land lay. So when a man climbed up
A rugged peak and gazed, his shaded eye
Saw but the endless vista - hill on hill,
Slope beyond slope, each bonded like its brothers.

It was a gloomy land that seemed to hold
All winds and clouds and dreams that shun the sun,
With bare boughs rattling in the lonesome winds,
And the dark woodlands brooding over all,
Not even lightened by the rare dim sun
Which made squat shadows out of men; they called it
Cimmeria, land of Darkness and deep Night.

It was so long ago and far away
I have forgot the very name men called me.
The axe and flint tipped spear are like a dream,
And hunts and wars are shadows. I recall
Only the stillness of that sombre land;
The clouds that piled forever on the hills,
The dimness of the everlasting woods,
Cimmeria, land of Darkness and the Night.

Oh, soul of mine, born out of shadowed hills,
To clouds and winds and ghosts that shun the sun,
How many deaths shall serve to break at last
This heritage which wraps me in the grey
Apparel of ghosts? I search my heart and find
Cimmeria, land of Darkness and the Night.

Process returned 0 (0x0)   execution time : 0.045 s
Press any key to continue.
```

SCHRITT 7

Sie sehen auch, dass sich der Speicherort der Textdatei Cimmeria.txt nicht im selben Ordner wie das C++-Programm befindet. Als wir die erste name.txt-Datei erstellt haben, wurde sie in denselben Ordner geschrieben, in dem sich der Code befand; dies erfolgt standardmäßig. Um einen anderen Ordner anzugeben, müssen Sie, je nach Zeichencode/Escape-Sequenz, zwei umgekehrte Schrägstriche verwenden.

```
6
7   {
8       string line;
9       ifstream newfile ("c:\\users\\david\\Documents\\Cimmeria.txt");
10
11      cout << "Cimmeria, by Robert E Howard: \n" << endl;
12
```

SCHRITT 8

Wie erwartet können Sie fast alles in eine Datei schreiben, entweder im Editor oder über die Konsole durch den C++-Code:

```
    string name;
    int age;

    ofstream newfile;
    newfile.open("name.txt");

    cout << "Enter your name: " << endl;
    getline(cin, name);

    newfile << name << endl;

    cout << "\nHow old are you: " << endl;
    cin >> age;

    newfile << age << endl;

    newfile.close();
```

Start here | fileinout.cpp

```
1   #include <iostream>
2   #include <fstream>
3   using namespace std;
4
5   int main ()
6
7   {
8       string name;
9       int age;
10
11      ofstream newfile;
12      newfile.open("name.txt");
13
14      cout << "Enter your name: " << endl;
15      getline(cin, name);
16
17      newfile << name << endl;
18
19      cout << "\nHow old are you: " << endl;
20      cin >> age;
21
22      newfile << age << endl;
23
24      newfile.close();
25
26  }
```

SCHRITT 9

Der Code in Schritt 8 unterscheidet sich erneut, aber nur in Hinsicht auf das Hinzufügen der Ganzzahl. Beachten Sie, dass wir anstelle der vorherigen Funktion getline(cin, variable) nun cin >> age verwendet haben. Das liegt daran, dass die getline-Funktion Strings und keine ganzen Zahlen verarbeitet. Nehmen Sie daher für andere Datentypen als Strings die Standardfunktion cin.

```
Enter your name:
David Hayward

How old are you:
45

Process returned 0 (0x0)   execution time : 7.369 s
Press any key to continue.
```

SCHRITT 10

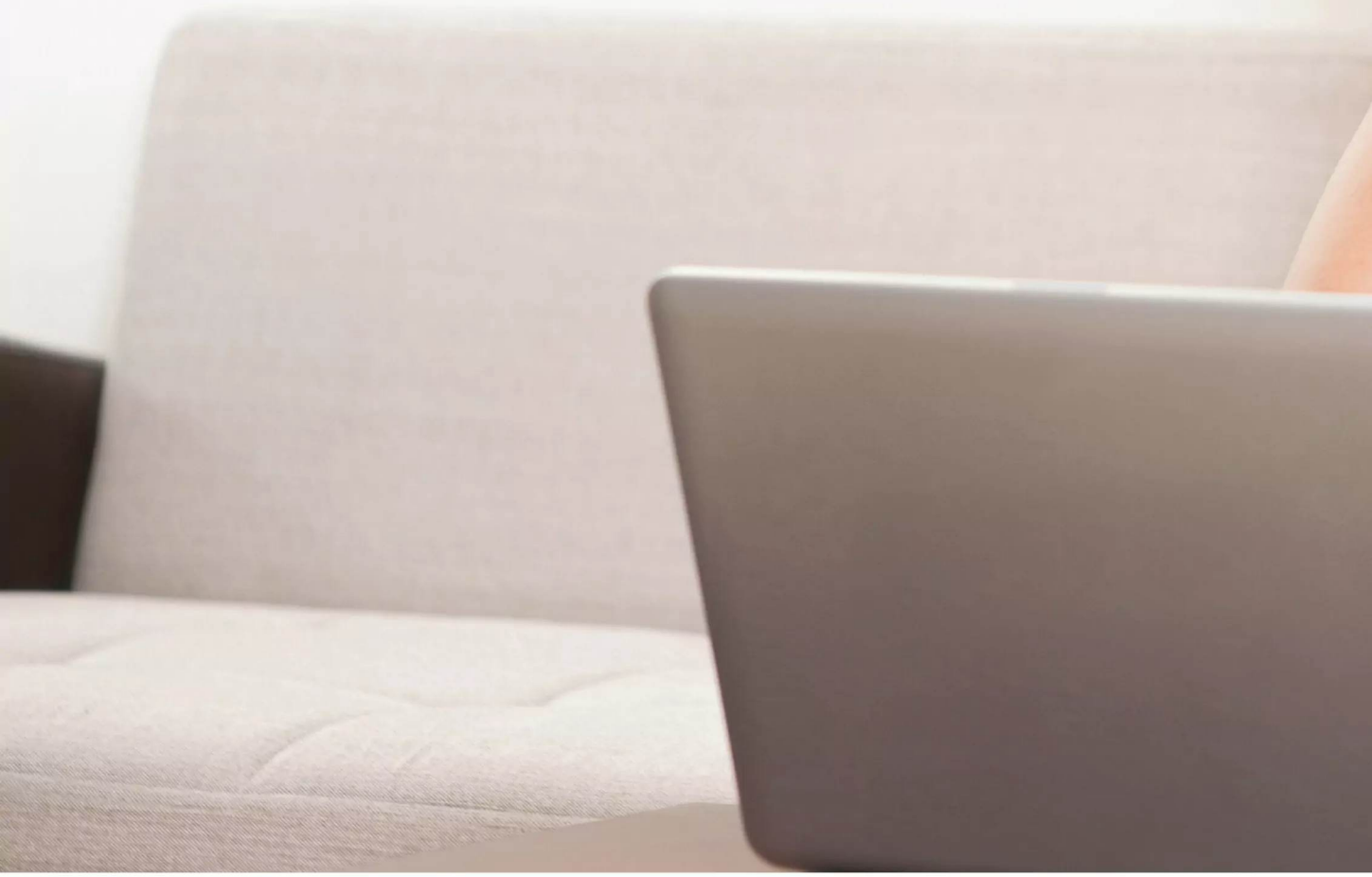
Hier ist eine Übung: Prüfen Sie, ob Sie Code erstellen können, mit dem verschiedene Elemente in eine Textdatei geschrieben werden, z. B. der Name, das Alter und die Rufnummer usw. eines Benutzers, und vielleicht auch der Wert von Pi und verschiedene mathematische Elemente.

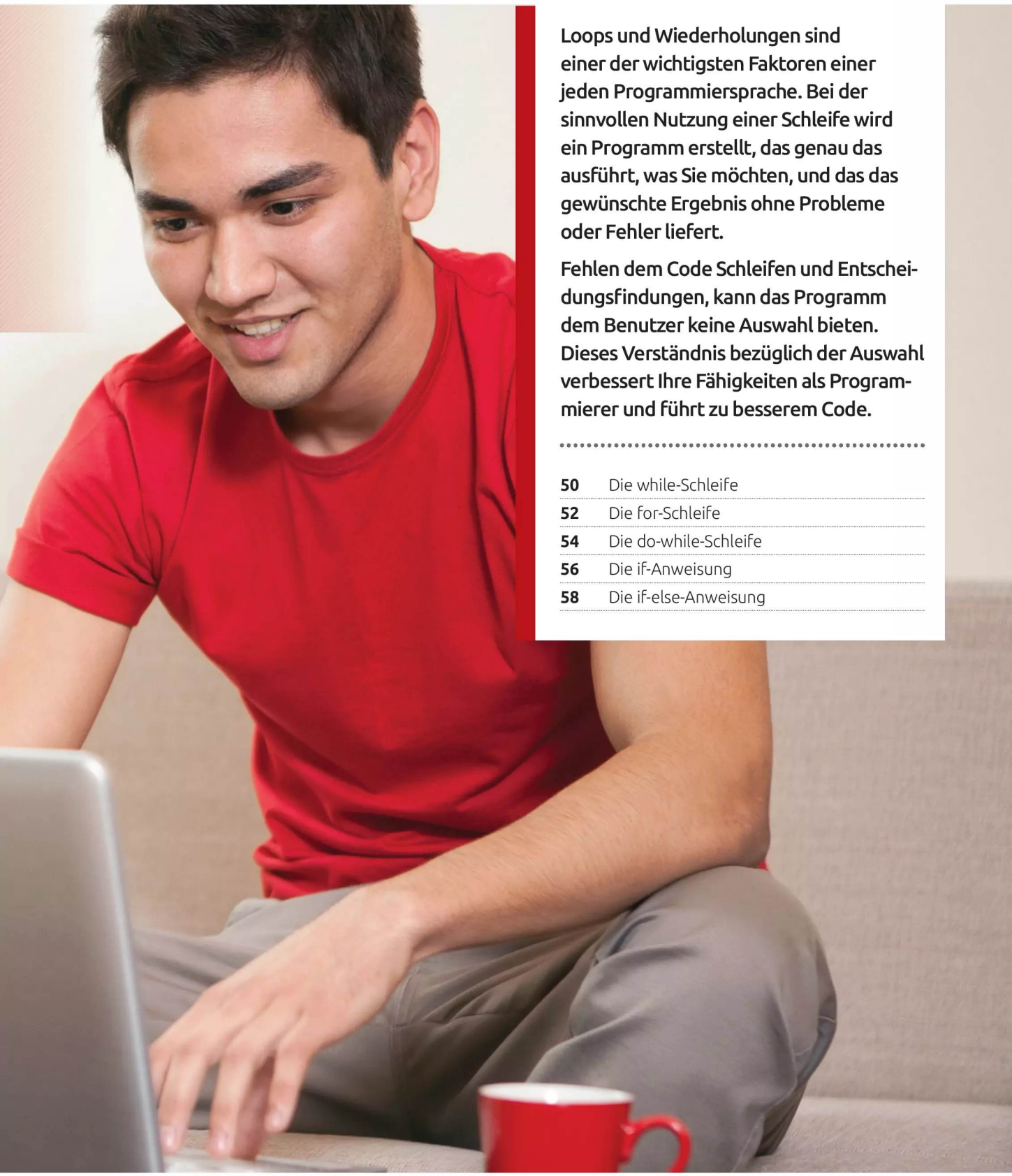
```
File  Edit  Format  View  Help
Name: David Hayward
Age: 45

Pi = 3.14159

The square root of 133 is: 11.5325

What else can you come up with...?
```

Schleifen und Entscheidungs-findungen

„Der Zweck der Softwareentwicklung besteht darin, Komplexität zu kontrollieren und nicht zu erzeugen. “

– Pamela Zave (Entwicklerin, Wissenschaftlerin und Telekommunikationsexpertin)





Loops und Wiederholungen sind einer der wichtigsten Faktoren einer jeden Programmiersprache. Bei der sinnvollen Nutzung einer Schleife wird ein Programm erstellt, das genau das ausführt, was Sie möchten, und das das gewünschte Ergebnis ohne Probleme oder Fehler liefert.

Fehlen dem Code Schleifen und Entscheidungsfindungen, kann das Programm dem Benutzer keine Auswahl bieten. Dieses Verständnis bezüglich der Auswahl verbessert Ihre Fähigkeiten als Programmierer und führt zu besserem Code.

Die while-Schleife

Eine while-Schleife wiederholt eine Anweisung oder eine Gruppe von Anweisungen, während eine bestimmte Bedingung erfüllt bleibt. Wenn die while-Schleife beginnt, initialisiert sie sich selbst, indem sie die Bedingungen der Schleife und die darin enthaltenen Anweisungen überprüft, bevor der Rest der Schleife ausgeführt wird.

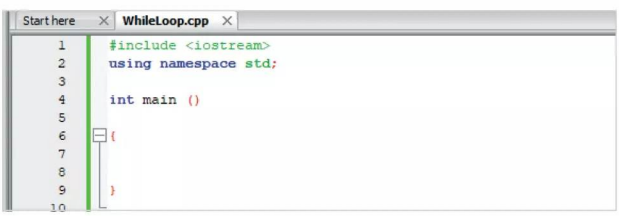
WAHR ODER FALSCH?

while-Schleifen sind eine der beliebtesten Formen von Schleifen in C++-Code. Sie wiederholen den in der Schleife enthaltenen Code solange, bis die Bedingung erfüllt ist. Sobald sich die Bedingung als falsch herausstellt, wird der Code normal fortgesetzt.

SCHRITT 1

Löschen Sie, was Sie bisher getan haben, und erstellen Sie eine neue C++-Datei. Fügen Sie die üblichen Standard-Header hinzu:

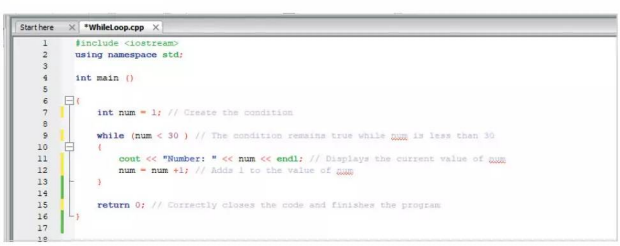
```
#include <iostream>
using namespace std;

int main ()

{

}
```



SCHRITT 2

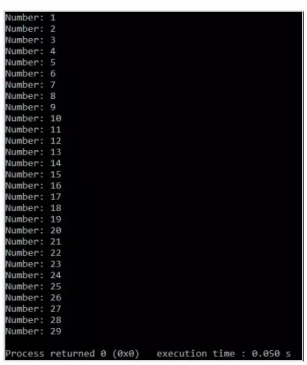
Erstellen Sie eine einfache while-Schleife. Geben Sie den unteren Code ein, erstellen und führen Sie ihn aus (wir haben dem Screenshot Kommentare hinzugefügt):

```
{
    int num = 1;

    while (num < 30 )
    {
        cout << "Number: " << num << endl;
        num = num +1;
    }

    return 0;
}
```



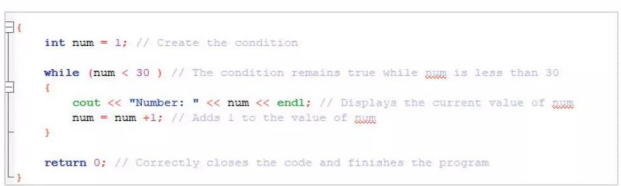
SCHRITT 3

Zuerst müssen Sie eine Bedingung erstellen. Verwenden Sie daher eine Variable namens num und geben Sie ihr den Wert 1. Erstellen Sie nun die while-Schleife, die besagt, dass die num-Variable wahr ist, solange sie kleiner als 30 ist. Innerhalb der Schleife wird der num-Wert angezeigt und um 1 addiert, bis er größer als 30 ist.



SCHRITT 4

Wir führen hier einige neue Elemente ein. Zunächst haben wir die öffnenden und schließenden Klammern für die while-Schleife, da unsere Schleife eine zusammengesetzte Anweisung ist, d. h eine Gruppe von Anweisungen. Beachten Sie auch, dass nach der while-Anweisung kein Semikolon steht. return 0 ist die bevorzugte Methode zum Beenden des Codes.



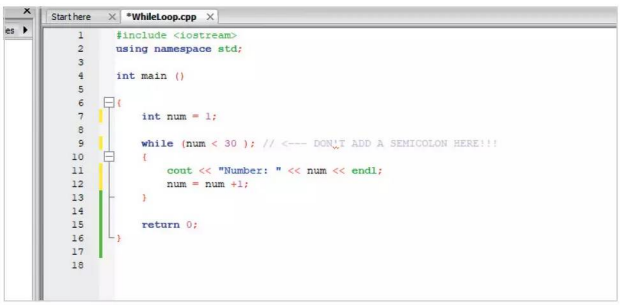
SCHRITT 5

Wenn wir den ständig steigenden num-Wert nicht sehen müssen, brauchen wir auch die gruppierte while -Anweisung nicht. Stattdessen lassen wir die num-Variable addieren, bis sie 30 erreicht hat, und lassen uns dann den Wert anzeigen:

```
{
    int num = 1;

    while (num < 30 )
        num++;
    cout << "Number: " << num << endl;

    return 0;
}
```

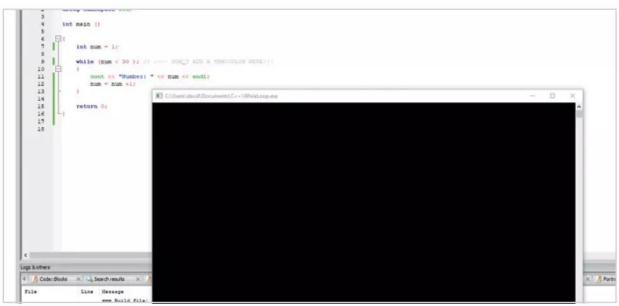
SCHRITT 6

Beachten Sie, dass am Ende einer while-Anweisung kein Semikolon hinzugefügt wird. Wie Sie wissen, stellt das Semikolon das Ende einer C++-Codezeile dar. Wenn Sie es ans Ende einer while-Anweisung setzen, bleibt Ihre Schleife so lange hängen, bis Sie das Programm schließen.



SCHRITT 7

Bei der Ausführung des Codes würde der num-Wert laut int-Anweisung 1 betragen. Trifft der Code auf die while-Anweisung, liest er die Bedingung, dass die Schleife auszuführen ist, solange 1 kleiner als 30 ist. Das Semikolon beendet die Zeile, wodurch sich die Schleife wiederholt, ohne allerdings den num-Wert um 1 zu erhöhen, da der Code nicht durch die zusammengesetzte Anweisung fortgesetzt wird.



SCHRITT 8

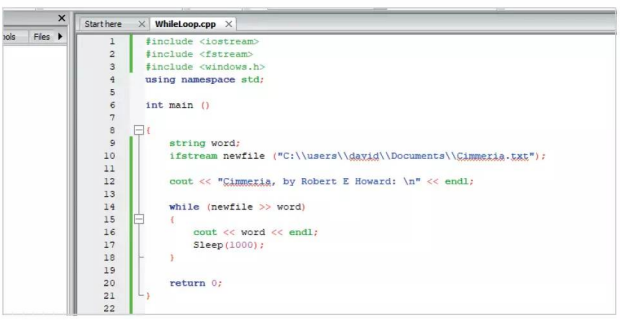
Sie können die while-Anweisung so ändern, dass, abhängig vom Code, der innerhalb der Schleife liegt, unterschiedliche Ergebnisse angezeigt werden. Um z. B. das Gedicht Cimmeria Wort für Wort zu lesen, geben Sie Folgendes ein:

```
#include <iostream>
#include <fstream>
using namespace std;

int main ()

{
    string word;
    ifstream newfile ("C:\\users\\david\\
Documents\\Cimmeria.txt");

    cout << "Cimmeria, by Robert E Howard: \n" <<
endl;

    while (newfile >> word)
    {
        cout << word << endl;
    }

    return 0;
}
```

SCHRITT 9

Sie können den Code erweitern, sodass jede Sekunde ein Wort des Gedichts erscheint. Dazu müssen Sie eine neue Bibliothek, <windows.h>, einholen. Dies ist eine reine Windows-Bibliothek die Ihnen die Sleep()-Funktion bietet:

```
#include <iostream>
#include <fstream>
#include <windows.h>
using namespace std;

int main ()

{
    string word;
    ifstream newfile ("C:\\users\\david\\
Documents\\Cimmeria.txt");

    cout << "Cimmeria, by Robert E Howard: \n" <<
endl;

    while (newfile >> word)
    {
        cout << word << endl;
        Sleep(1000);
    }

    return 0;
}
```



SCHRITT 10

Sleep() funktioniert in Millisekunden, Sleep(1000) ist daher eine Sekunde, Sleep(10000) zehn Sekunden usw. Durch die Kombination der Sleep-Funktion (zusammen mit dem benötigten Header) und einer while-Schleife können Sie z. B. Code für einen Countdown erstellen.

```
#include <iostream>
#include <windows.h>
using namespace std;

int main ()

{
    int a = 10;

    while (a != 0)
    {
        cout << a << endl;
        a = a - 1;
        Sleep(1000);
    }

    cout << "\nBlast Off!" << endl;

    return 0;
}
```

Die for-Schleife

In gewisser Hinsicht funktioniert eine for-Schleife auf ähnliche Weise wie eine while-Schleife, auch wenn sich ihre Struktur unterscheidet. Eine for-Schleife besteht aus drei Stufen: der Initialisierung, der Bedingung und einem inkrementellen Schritt. Einmal eingerichtet, wiederholt sich die Schleife, bis die Bedingung falsch ist.

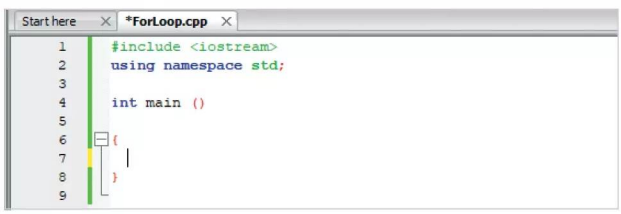
ÄHNLICH UND DOCH VERSCHIEDEN

Die Initialisierung wird nur einmal ausgeführt. Dadurch wird die Punktreferenz für die for-Schleife festgelegt. Die Schleife wertet die Bedingung aus, um zu sehen, ob sie wahr oder falsch ist, und führt dann das Inkrement aus. Danach wiederholt sie die 2. und 3. Stufe.

SCHRITT 1

Erstellen Sie mit den Standardkopfzeilen eine neue C++-Datei:

```
#include <iostream>
using namespace std;

int main ()

{

}
```



SCHRITT 2

Beginnen Sie, indem Sie eine einfache for-Schleife erstellen, die von 1 bis 30 zählt und den Wert mit jedem Inkrement auf dem Bildschirm anzeigt:

```
{
    //For Loop Begins
    for( int num = 1; num < 30; num = num +1 )
    {
        cout << "Number: " << num << endl;
    }

    return 0;

}
```

SCHRITT 3

Erstellen Sie einen Integer namens num, und weisen Sie ihm den Wert 1 zu. Legen Sie anschließend die Bedingung fest; in diesem Fall ist num kleiner als 30. In der letzten Stufe werden die Inkremente erstellt. Hier wird der num-Wert um 1 addiert.

```
{
    //For Loop Begins
    for( int num = 1; num < 30; num = num +1 )
```

SCHRITT 4

Nach der Schleife folgt eine zusammengesetzte Anweisung in geschweiften Klammern, die den aktuellen Wert der num-Integer anzeigt. Jedes Mal, wenn sich die for-Schleife wiederholt (die zweite und dritte Stufe der Schleife), addiert sie 1, bis die Bedingung < 30 falsch ist. Die Schleife endet dann und der Code wird fortgesetzt und mit return 0 beendet.

```
{
    //For Loop Begins
    for( int num = 1; num < 30; num = num +1 )
    {
        cout << "Number: " << num << endl;
    }

    return 0;

}
```

SCHRITT 5

Eine for-Schleife ist in C++ eine recht kompakte Einheit, die in ihren eigenen Klammern enthalten ist, während die anderen Elemente außerhalb der Schleife darunter angezeigt werden. Wenn Sie einen Countdown von 10 Sekunden erstellen möchten, geben Sie Folgendes ein:

```
#include <iostream>
#include <windows.h>
using namespace std;

int main ()

{
    //For Loop Begins
    for( int a = 10; a != 0; a = a -1 )
    {
        cout << a << endl;
        Sleep(1000);
    }

    cout << "\nBlast Off!" << endl;

    return 0;

}
```

SCHRITT 6

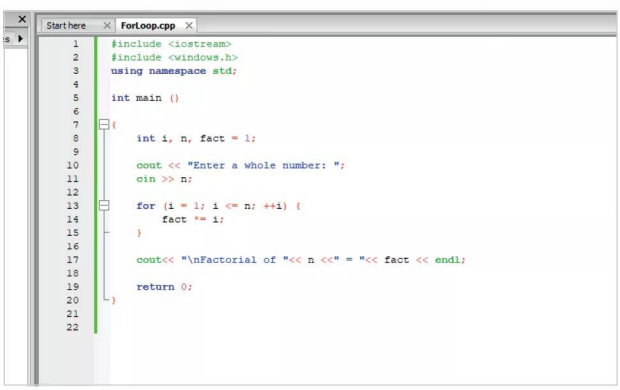
Vergessen Sie beim Countdown-Code nicht, die windows.h-Bibliothek mit einzuschließen, damit Sie den Sleep-Befehl auch nutzen können. Erstellen Sie den Code und führen Sie ihn aus. In der Befehlskonsole sehen Sie den Countdown von 10 zu 1, der in Schritten von einer Sekunde ausgeführt wird. Wenn Null erreicht wird, erscheint Blast Off!

```
C:\Users\david\Documents\C++\ForLoop.exe
10
9
8
7
6
5
4
3
2
1

Blast Off!

Process returned 0 (0x0)   execution time : 10.049 s
```

SCHRITT 7

Natürlich können Sie noch viel mehr in eine for-Schleife einfügen, z. B. Benutzereingaben:

```
    int i, n, fact = 1;

    cout << “Enter a whole number: “;
    cin >> n;

    for (i = 1; i <= n; ++i) {
        fact *= i;
    }

    cout<< “\nFactorial of “<< n <<” = “<< fact << endl;

    return 0;
```



SCHRITT 8

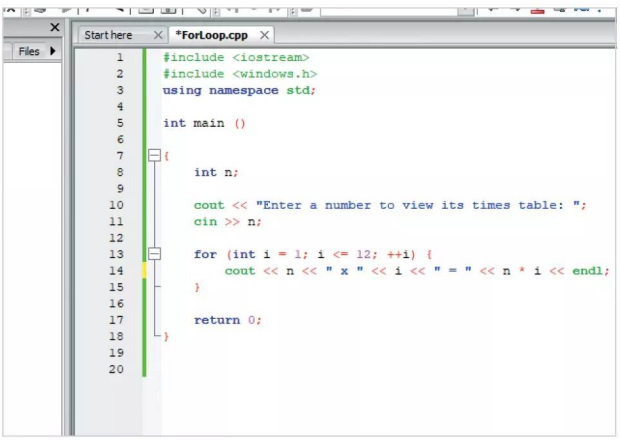
Der Code in Schritt 7 fragt nach seiner Erstellung und Ausführung nach einer Nummer und zeigt dann die Fakultät dieser Nummer durch die for-Schleife an. Die Nummer des Benutzers wird im Integer n gespeichert. Der Integer I überprüft anschließend, ob die Bedingung wahr oder falsch ist, fügt jedes Mal 1 hinzu und vergleicht die Bedingung mit der Nummer des Benutzers, n.

```
C:\Users\david\Documents\C++\ForLoop.exe
Enter a whole number: 8

Factorial of 8 = 40320

Process returned 0 (0x0)   execution time : 2.898 s
Press any key to continue.
```

SCHRITT 9

Hier ist ein Beispiel für eine for-Schleife, in der die Multiplikationstabellen einer vom Benutzer eingegebenen Nummer angezeigt werden:

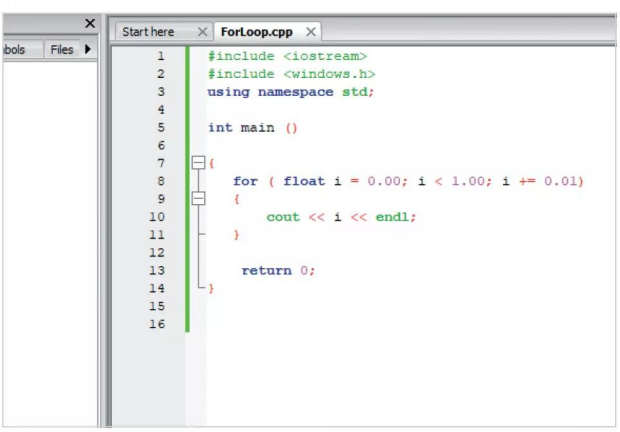
```
{
    int n;

    cout << “Enter a number to view its times table: “;
    cin >> n;

    for (int i = 1; i <= 12; ++i) {
        cout << n << “ x “ << i << “ = “ << n * i << endl;
    }

    return 0;
}
```



SCHRITT 10

Der Wert des Integer i kann von 12 bis zu einer beliebigen Zahl erweitert werden, wobei eine sehr große oder kleine Multiplikationstabelle angezeigt wird. Natürlich muss der Datentyp in einer for-Schleife keine Ganzzahl sein; solange es ein gültiger Wert ist, wird sie funktionieren.

```
for ( float i = 0.00; i < 1.00; i += 0.01)
    {
        cout << i << endl;
    }
return 0;
```



Die do-while-Schleife

Eine do-while-Schleife unterscheidet sich geringfügig von der for- und auch der while-Schleife. Sowohl die for- als auch die while-Schleife legen die Bedingung der Schleife am Anfang fest und überprüfen diese. Im Gegensatz dazu prüft eine do-while-Schleife die Bedingung am Ende der Schleife.

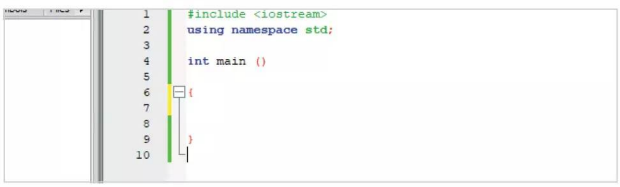
DO-SCHLEIFEN

Das Gute an einer do-while-Schleife ist, dass sie garantiert mindestens einmal durchlaufen wird. Die Struktur lautet do, gefolgt von Anweisungen, solange die Bedingung wahr ist.

SCHRITT 1

Beginnen Sie mit einer neuen, leeren C++-Datei und geben Sie die Standardkopfzeilen ein:

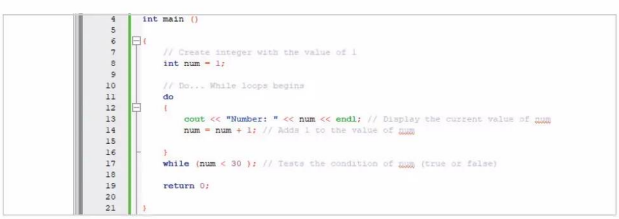
```
#include <iostream>
using namespace std;

int main ()

{


}
```



SCHRITT 2

Beginnen Sie mit einer einfachen Zählung von Nummern:

```
{
    int num = 1;

    do
    {
        cout << “Number: “ << num << endl;
        num = num + 1;

    }
    while (num < 30 );

    return 0;

}
```



SCHRITT 3

Hier ist ein Blick auf die Struktur einer do-while-Schleife. Erstellen Sie zuerst einen Integer namens num mit dem Wert 1. Die do-while-Schleife beginnt nun. Der Code im Hauptteil der Schleife wird mindestens einmal ausgeführt, dann wird die Bedingung auf wahr oder falsch überprüft.

```
// Create integer with the value of 1
int num = 1;

// Do... While loops begins
do
{
    cout << "Number: " << num << endl; // Display the current value of num
    num = num + 1; // Adds 1 to the value of num

}
while (num < 30 ); // Tests the condition of num (true or false)
```

SCHRITT 4

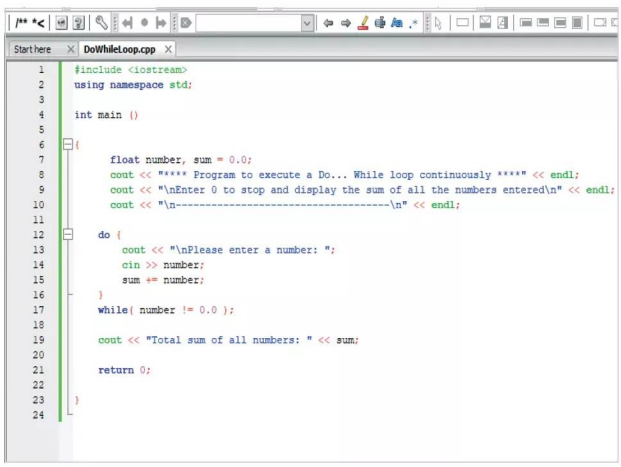
Wenn die Bedingung erfüllt ist, wird die Schleife ausgeführt. Dies setzt sich fort, bis die Bedingung falsch ist. In diesem Fall wird die Schleife beendet und der Code fortgesetzt. Sie können somit eine Schleife erstellen, in welcher der Code fortgesetzt wird, bis der Benutzer ein bestimmtes Zeichen eingibt.

```
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
Number: 6
Number: 7
Number: 8
Number: 9
Number: 10
Number: 11
Number: 12
Number: 13
Number: 14
Number: 15
Number: 16
Number: 17
Number: 18
Number: 19
Number: 20
Number: 21
Number: 22
Number: 23
Number: 24
Number: 25
Number: 26
Number: 27
Number: 28
Number: 29

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

SCHRITT 5

Der folgende Code addiert die vom Benutzer eingegebenen Zahlen, bis er eine Null eingibt:

```
{
      float number, sum = 0.0;
      cout << "**** Program to execute a Do... 
While loop continuously ****" << endl;
      cout << "\nEnter 0 to stop and display the 
sum of all the numbers entered\n" << endl;
      cout << "\n-----------------------------------
---\n" << endl;

    do {
        cout<<"\nPlease enter a number: ";
        cin>>number;
        sum += number;
    }
    while(number != 0.0);

    cout<<"Total sum of all numbers: "<<sum;

    return 0;

}
```



SCHRITT 6

Der Code in Schritt 5 funktioniert folgendermaßen: Es werden zwei Gleitkomma-Variablen erstellt, number und sum, die beide den Wert 0.0 erhalten. Es folgt eine kurze Anleitung für den Benutzer, dann beginnt die do-while-Schleife.

```
{
      float number, sum = 0.0;
      cout << "**** Program to execute a Do... While loop continuously ****" << endl;
      cout << "\nEnter 0 to stop and display the sum of all the numbers entered\n" << endl;
      cout << "\n-----------------------------------\n" << endl;
```

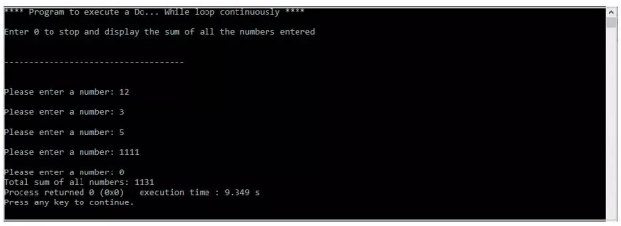
SCHRITT 7

Die do-while-Schleife fordert den Benutzer auf, eine Nummer einzugeben, die der Gleitkomma-Variablen „number" zugewiesen wurde. Der Berechnungsschritt verwendet die zweite Gleitkomma-Variable „sum", die jedes Mal, wenn der Benutzer einen neuen Wert eingibt, den Zahlenwert addiert.

```
    do {
        cout << "\nPlease enter a number: ";
        cin >> number;
        sum += number;
```

SCHRITT 8

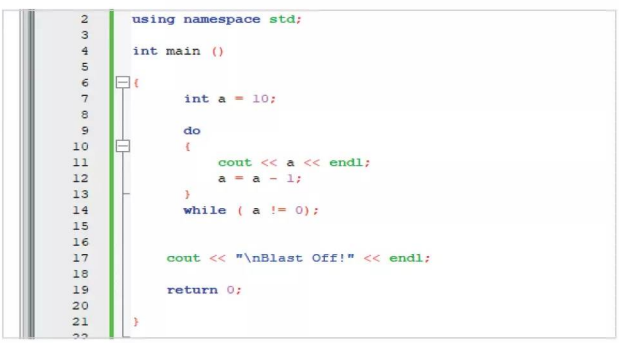
Zu guter Letzt prüft die while-Anweisung die Variablennummer. Wenn der Benutzer eine Null eingegeben hat, wird die Schleife beendet, andernfalls wird sie endlos fortgesetzt. Nach Eingabe der Null wird der Wert der Summe, der Gesamtwert aller Eingaben des Benutzers, angezeigt. Die Schleife und das Programm werden dann beendet.



SCHRITT 9

Mit dem zuvor verwendeten Countdown- und Blast Off!-Code würde eine do-while-Schleife wie folgt aussehen:

```
{
      int a = 10;

      do
      {
          cout << a << endl;
          a = a - 1;
      }
      while ( a != 0);


    cout << "\nBlast Off!" << endl;

    return 0;

}
```



SCHRITT 10

Der Hauptvorteil einer do-while-Schleife besteht darin, dass es sich um eine fußgesteuerte bzw. annehmende Schleife handelt, während die while-Schleife eine kopfgesteuerte bzw. abweisende Schleife ist. Wenn für Ihren Code eine Schleife erforderlich ist, die mindestens einmal ausgeführt werden muss (z. B. um die Anzahl der Leben in einem Spiel zu überprüfen), ist eine do-while-Schleife perfekt.

Die if-Anweisung

Die entscheidungsfindende Anweisung „if" ist wahrscheinlich eine der am häufigsten verwendeten Anweisungen in jeder Programmiersprache, ganz gleich, ob es sich um C++, Python, BASIC etc. handelt. Sie stellt einen Knotenpunkt im Code dar: ist eine Bedingung wahr, wird etwas ausgeführt, ist sie falsch, wird etwas anderes ausgeführt.

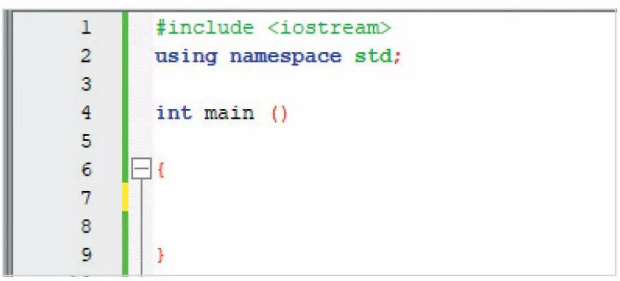
BOOLESCHER AUSDRUCK

Die if-Anweisung verwendet einen booleschen Ausdruck. Ist der boolesche Ausdruck wahr, wird der Code innerhalb der Anweisung ausgeführt; ist er falsch, wird stattdessen der Code, welcher der Anweisung folgt, ausgeführt.

SCHRITT 1

Erstellen Sie zunächst eine neue C++-Datei mit den üblichen relevanten Standardkopfzeilen:

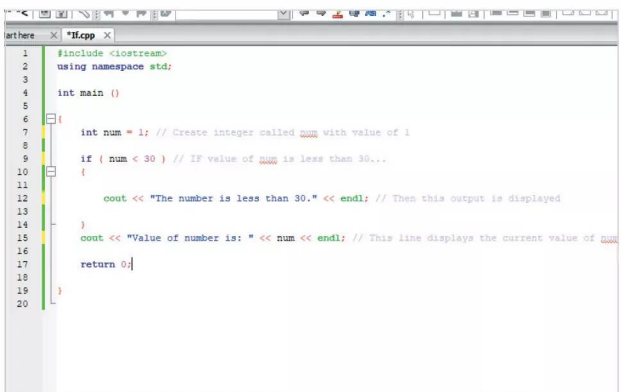
```
#include <iostream>
using namespace std;

int main ()

{


}
```



SCHRITT 2

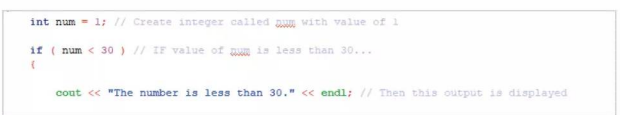
If lässt sich am besten anhand einer auf einer Zahl basierenden Bedingung erklären:

```
{
    int num = 1;

    if ( num < 30 )
    {

        cout << “The number is less than 30.” <<
endl;

    }
    cout << “Value of number is: “ << num << endl;

    return 0;

}
```



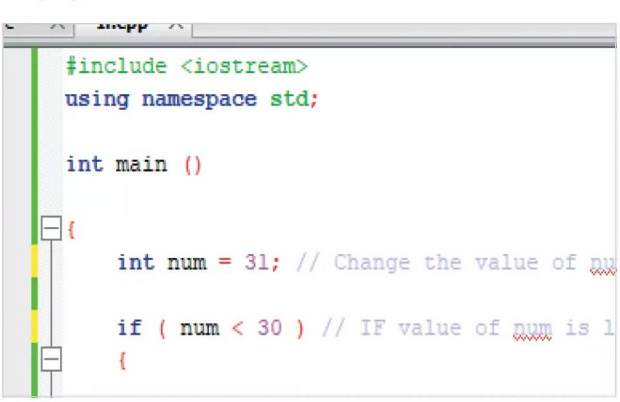
SCHRITT 3

Was ist hier passiert? Wir haben einen Integer namens num erstellt und diesem den Wert 1 zugewiesen. Darauf folgt die if-Anweisung. Dabei haben wir dem Code die Bedingung zugewiesen, dass der Code innerhalb der geschweiften Klammern ausgeführt werden soll, wenn der num-Wert kleiner als 1 ist.



SCHRITT 4

Die zweite cout-Anweisung zeigt den aktuellen num-Wert an und endet das Programm. Es ist leicht zu erkennen, wie die if-Anweisung funktioniert, wenn Sie den Ausgangswert von num von 1 in 31 ändern.



SCHRITT 5

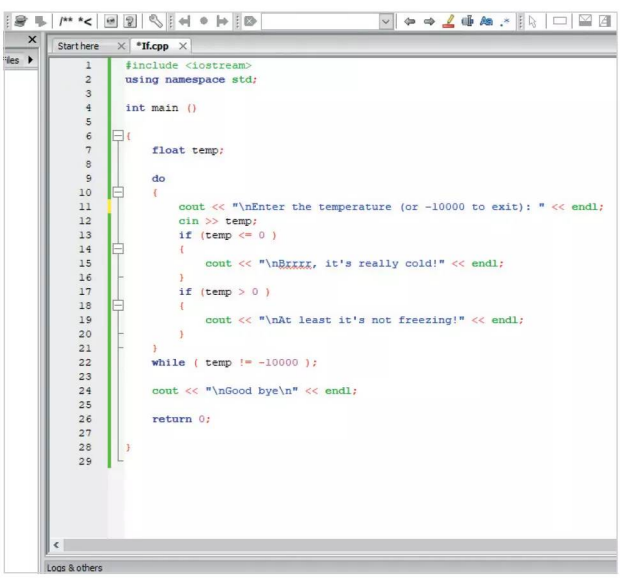
Wenn Sie den Wert auf über 30 ändern und dann den Code erstellen und ausführen, sehen Sie, dass die einzige Zeile, die auf dem Bildschirm ausgegeben wird, die zweite Cout-Anweisung ist, die den aktuellen Wert von num anzeigt. Dies liegt daran, dass die erste if-Anweisung falsch ist, sodass der Code innerhalb der geschweiften Klammern ignoriert wird.

```
C:\Users\david\Documents\C++\If.exe
Value of number is: 31

Process returned 0 (0x0)   execution time : 0.046 s
Press any key to continue.
```

SCHRITT 6

Sie können eine if-Anweisung auch in eine do-while-Schleife einfügen:

```
{
    float temp;

    do
    {
        cout << "\nEnter the temperature (or
-10000 to exit): " << endl;
        cin >> temp;
        if (temp <= 0 )
        {
            cout << "\nBrrrr, it's really cold!"
<< endl;
        }
        if (temp > 0 )
        {
            cout << "\nAt least it's not
freezing!" << endl;
        }
    }
    while ( temp != -10000 );

    cout << "\nGood bye\n" << endl;

    return 0;

}
```



SCHRITT 7

Der Code in Schritt 6 ist einfach aber effektiv. Zuerst haben wir eine Gleitkommazahl namens temp erstellt, dann eine do-while-Schleife, die den Benutzer auffordert, die aktuelle Temperatur einzugeben.

```
    float temp;

    do
    {
        cout << "\nEnter the temperature (or -10000 to exit): " << endl;
        cin >> temp;
```

SCHRITT 8

Die erste if-Anweisung prüft, ob der vom Benutzer eingegebene Wert kleiner als oder gleich 0 ist. In diesem Fall lautet die Ausgabe „Brrrr, it's really cold!". Ist die Eingabe größer als Null, gibt der Code „At least it's not freezing!" aus.

```
    do
    {
        cout << "\nEnter the temperature (or -10000 to exit): " << endl;
        cin >> temp;
        if (temp <= 0 )
        {
            cout << "\nBrrrr, it's really cold!" << endl;
        }
        if (temp > 0 )
        {
            cout << "\nAt least it's not freezing!" << endl;
        }
```

SCHRITT 9

Wenn der Benutzer schließlich den Wert -10000 eingibt, was ein unrealistischer Wert ist, wird die do-while-Schleife beendet und „Good bye" auf dem Bildschirm angezeigt.

```
{
    float temp;

    do
    {
        cout << "\nEnter the temperature (or -10000 to exit): " << endl;
        cin >> temp;
        if (temp <= 0 )
        {
            cout << "\nBrrrr, it's really cold!" << endl;
        }
        if (temp > 0 )
        {
            cout << "\nAt least it's not freezing!" << endl;
        }
    }
    while ( temp != -10000 );

    cout << "\nGood bye\n" << endl;

    return 0;

}
```

SCHRITT 10

If ist bei korrekter Anwendung ziemlich leistungsstark. Denken Sie daran, dass der Code den Inhalt der Klammern ausführt, wenn die Bedingung erfüllt ist. Wenn nicht, läuft er munter weiter. Schauen Sie, was Sie sonst noch mit if und einer Kombination aus Schleifen erstellen können.

```
Enter the temperature (or -10000 to exit):
34

At least it's not freezing!

Enter the temperature (or -10000 to exit):
2

At least it's not freezing!

Enter the temperature (or -10000 to exit):
-110

Brrrr, it's really cold!

Enter the temperature (or -10000 to exit):
0

Brrrr, it's really cold!

Enter the temperature (or -10000 to exit):
-10000

Brrrr, it's really cold!

Good bye


Process returned 0 (0x0)   execution time : 17.323 s
Press any key to continue.
```

Die if-else-Anweisung

Eine weitaus bessere Möglichkeit zur Nutzung einer if-Anweisung sind if und else, die ähnlich wie eine Standard-if-Anweisung funktionieren. Wenn der boolesche Ausdruck wahr ist, wird der Code innerhalb der geschweiften Klammern ausgeführt. Andernfalls wird stattdessen der Code in den nächsten geschweiften Klammern verwendet.

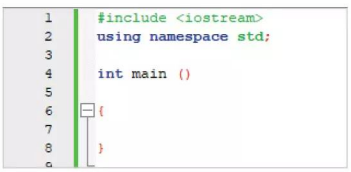
WENN-DANN

Es gibt zwei Codeabschnitte, die abhängig vom Ergebnis einer if-else-Anweisung ausgeführt werden können. Es lässt sich recht einfach visualisieren, wenn man sich an die Struktur dieser Anweisung gewöhnt hat.

SCHRITT 1

Beginnen Sie mit einer neuen C++-Datei und den Standard-Header:

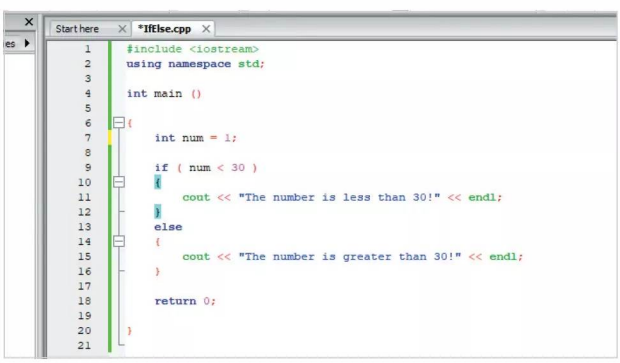
```
#include <iostream>
using namespace std;

int main ()

{

}
```



SCHRITT 2

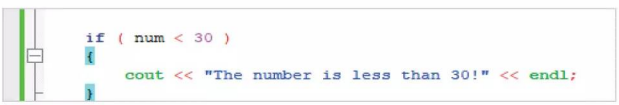
Wir erweitern den Code der if-Anweisung von der vorherigen Seite:

```
{
    int num = 1;

    if ( num < 30 )
    {
        cout << “The number is less than 30!” <<
endl;
    }
    else
    {
        cout << “The number is greater than 30!”
<< endl;
    }

    return 0;

}
```



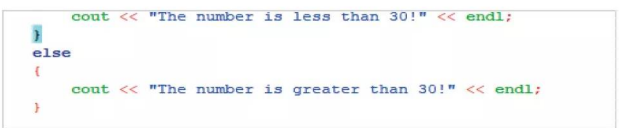
SCHRITT 3

Die erste Zeile im Code erstellt den Integer namens num und weist ihm den Wert 1 zu. Die if-Anweisung prüft, ob der num-Wert kleiner als 30 ist und gibt bei Übereinstimmung „The number is less than 30!” an die Konsole aus.



SCHRITT 4

Der else-Teil überprüft, ob die Nummer größer als 30 ist. Bei Übereinstimmung zeigt die Konsole „The number is greater than 30!” an. Anschließend wird der Code beendet.



SCHRITT 5

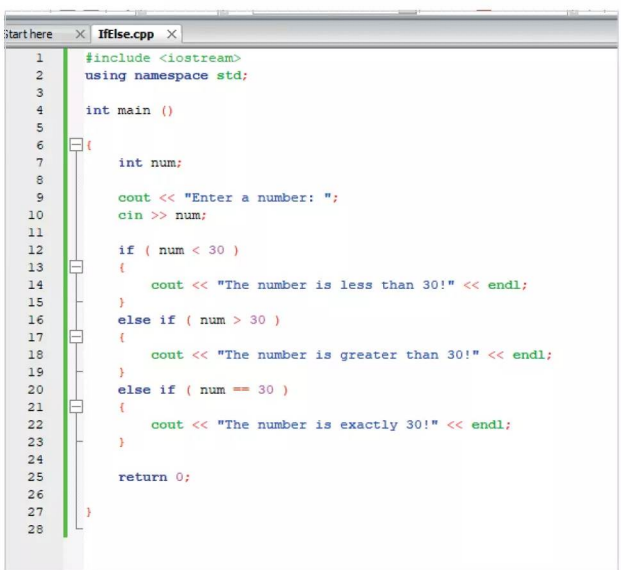
Sie können den Wert von num im Code ändern oder den Code verbessern, indem Sie den Benutzer zur Eingabe eines Werts auffordern:

```
{
    int num;

    cout << “Enter a number: “;
    cin >> num;

    if ( num < 30 )
    {
        cout << “The number is less than 30!” <<
endl;
    }
    else
    {
        cout << “The number is greater than 30!”
<< endl;
    }

    return 0;

}
```

SCHRITT 6

Der Code funktioniert wie erwartet, aber was, wenn Sie etwas anzeigen lassen möchten, wenn der Benutzer die Nummer 30 eingibt? Versuchen Sie Folgendes:

```
{
    int num;

    cout << “Enter a number: “;
    cin >> num;

    if ( num < 30 )
    {
        cout << “The number is less than 30!” <<
endl;
    }
    else if ( num > 30 )
    {
        cout << “The number is greater than 30!”
<< endl;
    }
    else if ( num == 30 )
    {
        cout << “The number is exactly 30!” <<
endl;
    }

    return 0;

}
```



SCHRITT 7

Die neue Code-Ergänzung ist die sogenannte verschachtelte if-else-Anweisung. Diese ermöglicht die Suche nach mehreren Bedingungen. Wenn der Benutzer eine Zahl eingibt, die kleiner als, größer als oder gleich 30 ist, erhält er ein anderes Ergebnis.

```
C:\Users\david\Documents\C++\IfElse.exe
Enter a number: 30
The number is exactly 30!

Process returned 0 (0x0)   execution time : 4.037 s
Press any key to continue.
```

SCHRITT 8

Erstellen Sie nun ein Zahlenspiel mit zwei Spielern. Beginnen Sie mit dem Erstellen der Variablen:

```
int num, guess, tries = 0;

    cout << “***** Two-player number guessing game
****” << endl;
    cout << “\nPlayer One, enter a number for
Player Two to guess: “ << endl;
    cin >> num;
    cout << string(50, ‘\n’);
```

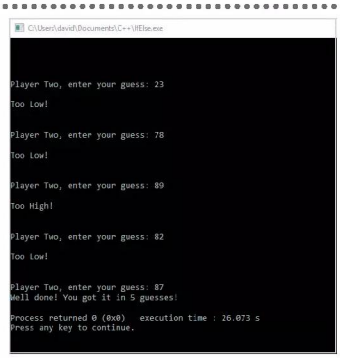
SCHRITT 9

Der cout << string(50,'\n) löscht den Bildschirm, sodass Spieler 2 die eingegebene Nummer nicht sieht. Erstellen Sie nun zusammen mit if-else eine do-while-Schleife:

```
  do
    {
        cout << “\nPlayer Two, enter your guess: “;
        cin >> guess;
        tries++;
        if (guess > num)
        {
            cout << “\nToo High!\n” << endl;
        }
        else if (guess < num)
        {
            cout << “\nToo Low!\n” << endl;
        }
        else if (guess == num)
        {
            cout << “Well done! You got it in “ <<
tries << “ guesses!” << endl;
        }
    } while (guess != num);

    return 0;
```

SCHRITT 10

Holen Sie sich einen zweiten Spieler dazu und erstellen Sie den Code. Spieler 1 gibt die zu erratene Zahl ein. Spieler 2 kann so lange raten, bis die richtige Zahl erhalten wurde. Um das Ganze zu erschweren, könnten Sie auch Dezimalzahlen nehmen.



Arbeiten mit Code

Bevor man mit dem Code schreiben beginnt, lohnt es sich, ein Auge auf die gängigen Programmierfallen zu werfen, denn schlechte Gewohnheiten schleichen sich oftmals schon zu Beginn eines Prozesses ein. Wenn Sie wissen, wie Sie Ihren Code strukturieren, bevor Sie ihn erstellen, werden Sie die meisten Fehler vermeiden, die normalerweise mit schlecht entwickeltem Code einhergehen.

In diesem Abschnitt nehmen wir die häufigsten Programmierfehler unter die Lupe. Ferner schauen wir uns an, was Sie mit Ihren zunehmenden Programmierfähigkeiten machen könnten.

„Die meisten guten Programmierer programmieren nicht des Geldes oder der Anerkennung wegen, sondern weil das Programmieren Spaß macht."

– Linus Torvalds (Entwickler des Linux-Kernels)

Häufige Programmierfehler

Wenn Sie etwas Neues beginnen, werden Sie unweigerlich Fehler machen, was an der mangelnden Erfahrung liegt, und selbst den Experten unterläuft gelegentlich ein Patzer. Doch bekanntermaßen sind Fehler da, um aus ihnen zu lernen.

X=FEHLER, PRINT Y

Beim Programmieren gibt es zahlreiche Fallgruben, viel zu viele, um hier alle aufzulisten. Wenn Sie in der Lage sind, Fehler zu erkennen und zu beheben, sind Sie einen großen Schritt weitergekommen.

STÜCK FÜR STÜCK

Es wäre schon toll, wenn wir wie Neo aus den Matrix-Filmen arbeiten könnten. Stellen Sie sich vor, Ihr Gedächtnis wird geladen und Sie wissen alles zu einem Thema. Leider geht das aber nicht. Die erste große Falle ist, zu schnell und zu viel zu lernen. Gehen Sie daher das Programmieren stückchenweise an und nehmen Sie sich Zeit.

//KOMMENTARE

Verwenden Sie Kommentare. Das Kommentieren Ihres Codes kann Ihnen bei der nächsten Anwendung viele Probleme ersparen. Durch das Einfügen von Kommentarzeilen können Sie schnell die Codeabschnitte durchgehen, die Probleme verursachen. Sie sind auch bei der Prüfung von älterem Code hilfreich.

```
54        --n;
55      }
56  #endif
57    if (n == 0)
58      return;
59
60    //
61    // Loop unrolling. Here be dragons.
62    //
63
64    // (n & (~3)) is the greatest multiple of 4 n
65    // In the while loop ahead, orig will move ov
66    // increments (4 elements of 2 bytes).
67    // end marks our barrier for not falling outs
68    char const * const end = orig + 2 * (n & (~3)
69
70    // See if we're aligned for writting in 64 or
71  #if ACE_SIZEOF_LONG == 8 && \
72      !((defined(  amd64  ) || defined (  x86_64
```

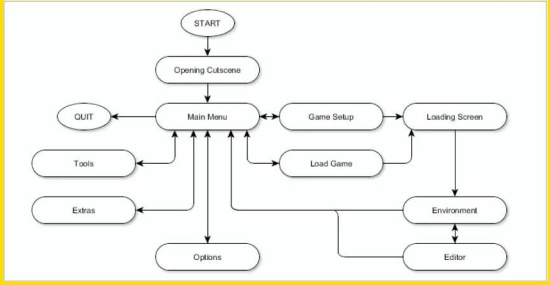
EINFACHE VARIABLEN

Variablen sinnvolle Namen zu geben ist ein Muss, um häufige Programmierfehler zu vermeiden. Buchstaben des Alphabets sind zwar in Ordnung, aber was, wenn der Code angibt, dass ein Problem mit der Variablen x vorliegt. Es ist nicht schwer, Variablen aussagekräftige Namen wie Leben, Geld, Spieler1 usw. zu geben.

```
1  var points = 1023;
2  var lives = 3;
3  var totalTime = 45;
4  write("Points: "+points);
5  write("Lives: "+lives);
6  write("Total Time: "+totalTime+" secs");
7  write("--------------------------");
8  var totalScore = 0;
```

VORAUSPLANEN

Vielleicht wachen Sie morgens mit dem Gedanken auf, Code für ein klassisches Textabenteuer zu schreiben, aber ohne einen guten Plan ist das nicht sinnvoll. Kleine Code-Abschnitte können ohne großen Aufwand geschrieben werden, ein längerer und ausführlicher Code erfordert jedoch einen guten Arbeitsplan, um Fehler zu vermeiden.



BENUTZERFEHLER

Benutzereingaben sind oft gravierende Fehler im Code, z. B. wenn der Benutzer sein Alter eingeben soll, aber anstatt Zahlen Buchstaben eingibt. Es kann passieren, dass ein Benutzer so viel eingegeben hat, dass der interne Puffer überläuft und der Code abstürzt. Achten Sie auf die Benutzereingaben und geben Sie in klaren Anweisungen an, was Sie vom Benutzer benötigen.

```
aswdfdsf
You have entered wrong input
s
You have entered wrong input
!"£"!£!"
You have entered wrong input
sdfsdf213213123
You have entered wrong input
12323423423423423234
You have entered wrong input
12
the number is: 12

Process returned 0 (0x0)   execution time : 21.495 s
Press any key to continue.
```

BACKUPS

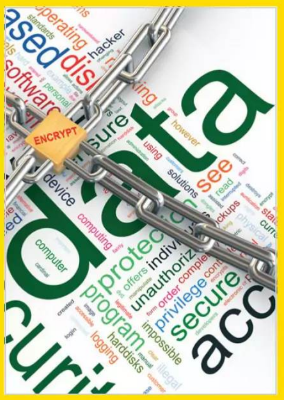
Machen Sie immer ein Backup Ihrer Arbeit, und ein sekundäres Backup für alle Änderungen, die Sie ausgeführt haben. Fehler können behoben werden, wenn ein gutes Backup vorhanden ist, auf das zurückgegriffen werden kann, wenn etwas schief geht. Es ist viel einfacher, an der Stelle zu beginnen, an der Sie aufgehört haben, als ganz von vorne zu beginnen.

DAS RAD NEU ERFINDEN

Man kann Tage damit verbringen, einen Codeabschnitt zu ermitteln, um ein bestimmtes Ergebnis zu erzielen. Das ist frustrierend und oftmals auch zeitraubend. Obwohl es sicherlich toll ist, ein Problem selbst lösen zu können, findet man denselben Code oft auch im Internet. Versuchen Sie also nicht, das Rad neu zu erfinden, sondern schauen Sie, ob jemand anderes das bereits getan hat.

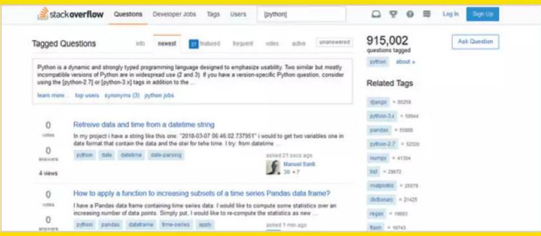
DATEN SICHERN

Wenn Sie Code schreiben, der Benutzernamen, Kennwörter oder andere sensible Daten betrifft, stellen Sie sicher, dass die Daten nicht im Klartext sind. Lernen Sie, wie Sie eine Funktion zum Verschlüsseln sensibler Daten erstellen, bevor Sie diese in eine Routine laden, in der sie übertragen oder gespeichert und möglicherweise eingesehen werden können.



HILFE!

Um Hilfe zu bitten fällt vielen schwer. Wird man ausgelacht? Verschwendet man die Zeit der Person, die man um Hilfe bittet? Es sollte sich aber niemand scheuen, Fragen zu stellen. Solange die Frage auf die richtige Art und Weise gestellt wird, sich an die Forumsregeln gehalten wird und man höflich ist, wird einem sicherlich gerne weitergeholfen.



MATHEMATIK

Wenn Ihr Code mehrere Berechnungen ausführt, müssen Sie sicherstellen, dass der dahintersteckende mathematische Teil stimmt. Es gibt unzählige Fälle, in denen Programme falsche Daten aufgrund schlechter mathematischer Codes erzeugt haben, was je nach Funktion des Codes verheerende Auswirkungen haben kann. Überprüfen Sie daher Ihre Code-Gleichungen genau.

```
set output

rmax = 5
nmax = 100

complex (x, y) = x * {1, 0} + y * {0, 1}
mandel (x, y, z, n) = (abs (z)> rmax || n>= 100)? n: mandel (x, y, z * z + complex (x, y), n + 1)

set xrange [-0.5:0.5]
set yrange [-0.5:0.5]
set logscale z
set samples 200
set isosample 200
set pm3d map
set size square
a= #A#
b= #B#
splot mandel(-a/100,-b/100,complex(x,y),0) notitle
```

Python – Anfängerfehler

Python ist eine relativ einfache Einstiegssprache für all jene, die in der Welt des Programmierens Fuß fassen wollen. Wie bei jeder anderen Programmiersprache können sich aber leicht gängige Fehler einschleichen, die eine Ausführung des Codes verhindern.

DEF BEGINNER(FEHLER=10)

Hier sind zehn häufige Python-Programmierfehler, die von den meisten Anfängern gemacht werden. Wenn Sie diese Fehler erkennen können, werden Sie sich in der Zukunft Kopfschmerzen ersparen.

VERSION

Python bietet zwei Live-Versionen seiner Sprache zum Herunterladen und Verwenden an. Es gibt die Versionen Python 2.7.x und Python 3.6.x. Die Version 3.6.x ist die aktuellste, und wir empfehlen, diese zu nehmen. Beachten Sie, dass auf Version 2.7.x geschriebener Code nicht immer mit auf 3.6.x geschriebenen Code funktioniert und umgekehrt.

EINRÜCKUNGEN, TABS & LEERZEICHEN

Python verwendet bei der Anzeige des Codes genaue Einrückungen. Die Einrückungen bedeuten, dass der Code in diesem Abschnitt Teil der vorherigen Anweisung ist und nicht mit einem anderen Teil des Codes verknüpft ist. Nehmen Sie nicht die Tabulatortaste, um eine Einrückung zu erstellen, sondern geben Sie vier Leerzeichen ein.

```
# set up counting
score = 0

# set up font
font = pygame.font.SysFont('calibri', 50)

def makeplayer():
    player = pygame.Rect(370, 635, 60, 25)
    return player

def makeinvaders(invaders):
    y = 0
    for i in invaders:
        x = 0
        for j in range(11):
            invader = pygame.Rect(75+x, 75+y, 50, 20)
            i.append(invader)
            x += 60
        y += 45
    return invaders

def makewalls(walls):
    wall1 = pygame.Rect(60, 520, 120, 30)
    wall2 = pygame.Rect(246, 520, 120, 30)
    wall3 = pygame.Rect(432, 520, 120, 30)
    wall4 = pygame.Rect(618, 520, 120, 30)
    walls = [wall1, wall2, wall3, wall4]
    return walls
```

INTERNET

Jeder Programmierer kopiert irgendwann Code aus dem Internet, um ihn in seine eigenen Routinen einzufügen. Es spricht auch nichts dagegen, den Code anderer zu verwenden, Sie müssen aber wissen, wie der Code funktioniert und was er bezweckt, bevor Sie ihn blindlings auf Ihrem eigenen Computer ausführen.

KOMMENTIEREN

Kommentare sind ein äußerst wichtiger Faktor beim Programmieren. Selbst wenn Sie der Einzige sind, der den Code jemals sehen wird, müssen Sie mithilfe von Kommentaren erklären, was passiert; verliert man bei der Funktion z. B. ein Leben? Schreiben Sie einen Kommentar, damit Sie und auch andere sehen, was die Funktion bezweckt.

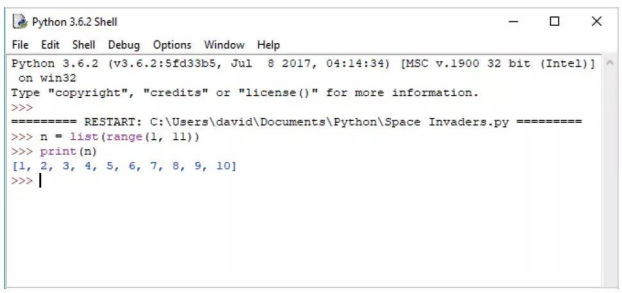
```
# set up pygame
pygame.init()
mainClock = pygame.time.Clock()

# set up the window
width = 800
height = 700
screen = pygame.display.set_mode((width, height), 0, 32)
pygame.display.set_caption('caption')

# set up movement variables
moveLeft = False
moveRight = False
moveUp = False
moveDown = False

# set up direction variables
DOWNLEFT = 1
DOWNRIGHT = 3
```

ZÄHLENDE SCHLEIFEN

Denken Sie daran, dass eine Schleife in Python die letzte Zahl, die Sie in einem Bereich angeben, nicht zählt. Wenn die Schleife also von 1 bis 10 zählen soll, müssen Sie Folgendes verwenden:

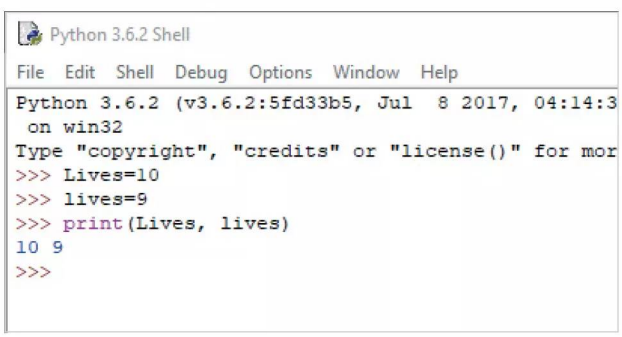
```
n = list(range(1, 11))
```

Dies gibt 1 bis 10 aus.



GROSS- UND KLEINSCHREIBUNG

Python ist eine von der Groß- und Kleinschreibung abhängige Programmiersprache, daher müssen Sie alle Variablen überprüfen, die Sie zuweisen. Zum Beispiel ist `Lives=10` eine andere Variable als lives=10. Das Aufrufen der falschen Variablen in Ihrem Code kann zu unerwarteten Ergebnissen führen.



KLAMMERN

Jeder hat schon einmal vergessen, die Extra-Klammer einzufügen, die ans Ende der Anweisung kommt. Python setzt voraus, dass die Routine genauso viele geschlossene wie geöffnete Klammern hat. Daher könnten Fehler in Ihrem Code darauf zurückzuführen sein, dass Sie vergessen haben, Ihre Klammern zu zählen; dazu gehören auch eckige Klammern.

```
def print_game_status(self):
        print (board[len(self.missed_letters)])
        print ('Word: ' + self.hide_word())
        print ('Letters Missed: ',)
        for letter in self.missed_letters:
                print (letter,)
        print ()
        print ('Letters Guessed: ',)
        for letter in self.guessed_letters:
                print (letter,)
        print ()
```

DOPPELPUNKTE

Anfänger vergessen häufig, am Ende einer strukturellen Anweisung einen Doppelpunkt hinzuzufügen:

```
class Hangman:
def guess(self, letter):
```

Der Doppelpunkt trennt den Code und erstellt die Einrückungen, zu denen der folgende Code gehört.

```
class Hangman:
        def __init__(self,word):
                self.word = word
                self.missed_letters = []
                self.guessed_letters = []

        def guess(self,letter):
                if letter in self.word and letter not in self.guessed_letters:
                        self.guessed_letters.append(letter)
                elif letter not in self.word and letter not in self.missed_letters:
                        self.missed_letters.append(letter)
                else:
                        return False
                return True

        def hangman_over(self):
                return self.hangman_won() or (len(self.missed_letters) == 6)

        def hangman_won(self):
                if '_' not in self.hide_word():
                        return True
                return False

        def hide_word(self):
                rtn = ''
                for letter in self.word:
                        if letter not in self.guessed_letters:
                                rtn += '_'
                        else:
                                rtn += letter
                return rtn
```

OPERATOREN

Ein falscher Operator ist ebenfalls ein häufiger Fehler. Wenn Sie z. B. einen Vergleich zwischen zwei Werten durchführen, müssen Sie den Gleichheitsoperator verwenden (doppeltes Gleichheitszeichen ==). Ein einzelnes Gleichheitszeichen (=) ist ein Zuweisungsoperator, der einer Variablen einen Wert zuordnet (z. B. Leben=10).

```
1 b = 5
2 c = 10
3 d = 10
4 b == c #false because 5 is not equal to 10
5 c == d #true because 10 is equal to 10
```

BETRIEBSSYSTEME

Das Schreiben von Code für mehrere Plattformen ist schwierig, insbesondere wenn Sie die externen Befehle des Betriebssystems verwenden. Wenn Ihr Code zum Beispiel den Bildschirm löschen soll, würden Sie für Windows `cls` verwenden. Für Linux müssen Sie hingegen `clear` nehmen. Um diesen Fehler zu beheben, müssen Sie den Fehler abfangen und einen alternativen Befehl erstellen.

```
# Code to detect error for using a different OS
run=1
while(run==1):
    try:
        os.system('clear')
    except OSError:
        os.system('cls')
    print('\n>>>>>>>>>>Python 3 File Manager<<<<<<<<<<\n')
```

C++ – Anfängerfehler

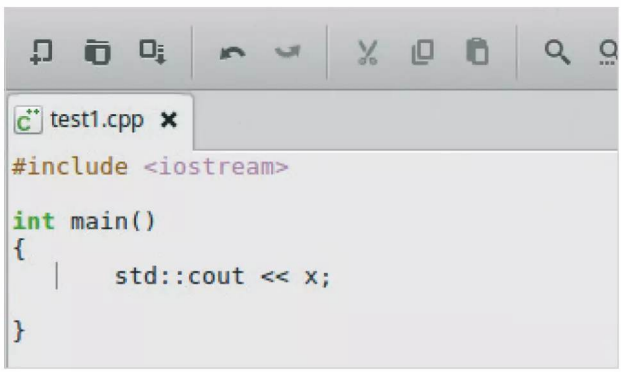
Es gibt viele Fallgruben, auf die der C++-Entwickler stoßen kann, zumal diese Sprache auch komplexer und oft unnachgiebiger ist. Anfänger sollten C++ schrittweise erlernen und erst das Gelernte verarbeiten, bevor sie mit dem nächsten Abschnitt weitermachen.

VOID(C++, FEHLER)

Zugegebenermaßen machen nicht nur C++-Anfänger die Fehler, die wir auf diesen Seiten beschreiben, auch dem gestandenen Programmierer passiert hin und wieder ein Schnitzer. Hier sind einige der häufigsten Probleme, die es zu vermeiden gilt.

NICHTDEKLARIERTE BEZEICHNER

Ein häufiger Fehler beim Programmieren in C++ und auch in den anderen Programmiersprachen ist der Versuch, eine Variable auszugeben, die nicht vorhanden ist. Das Anzeigen des Werts von x auf dem Bildschirm kann nur funktionieren, wenn Sie dem Compiler auch mitteilen, welchen Wert x haben soll.



STANDARD NAMENSRAUM

Es ist üblich, dass Anfänger in ihrem gesamten Code auf die Standardbibliothek verweisen. Wenn Sie jedoch das std::-Element einer Anweisung nicht finden, wird Ihr Code beim Kompilieren Fehler erzeugen. Sie können dies vermeiden, indem Sie unter #include Folgendes hinzufügen und dann einfach mit cout, cin usw. fortfahren:

```
using namespace std;
```

```
#include <iostream>
using namespace std;

int main()
{
    int a, b, c, d;
    a=10;
    b=20;
    c=30;
    d=40;

    cout << a, b, c, d;

}
```

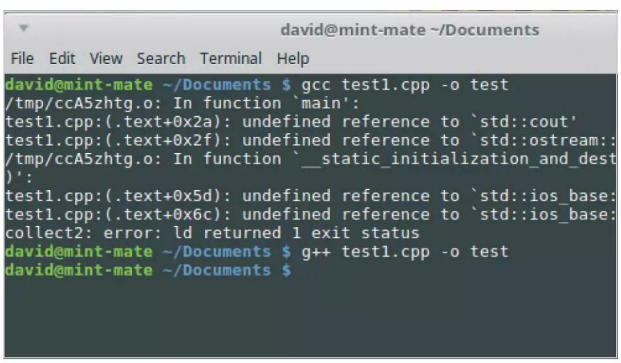
SEMIKOLONS

Denken Sie daran, dass jede Zeile eines C++-Programms mit einem Semikolon enden muss. Wenn es fehlt, betrachtet der Compiler die Zeile mit dem fehlenden Semikolon als die gleiche Zeile mit dem nächsten Semikolon. Dies verursacht alle möglichen Probleme beim Kompilieren. Vergessen Sie daher die Semikolons nicht.

```
#include <iostream>


int main()
{
    int a, b, c, d;
    a=10;
    b=20;
    c=30
    d=40;

    std::cout << a, b, c, d;

}
```

GCC OR G++

Wenn Sie unter Linux kompilieren, werden Sie zweifellos auf gcc und g++ stoßen. gcc ist die Gnu Compiler Collection (bzw. Gnu C Compiler, der frühere Name) und g++ die Gnu++ (die C++-Version) des Compilers. Wenn Sie C++ kompilieren, müssen Sie g++ verwenden, da ansonsten die falschen Compilertreiber verwendet werden.



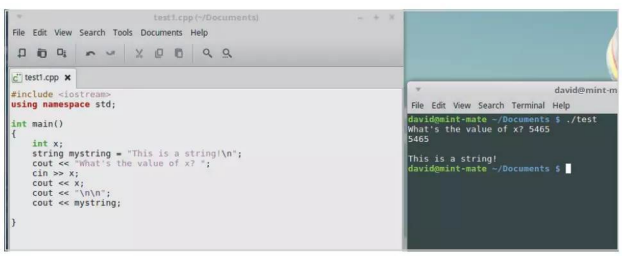
KOMMENTARE

Und wieder sind wir beim Problem der fehlenden Kommentare. Wie zuvor schon erwähnt ist es ohne lesbare Kommentare im Code sehr schwer herauszufinden, wie etwas funktioniert, sowohl für einen selbst als auch für andere. Also, das Kommentieren nicht vergessen.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{
    vector<double> v;

    double d;
    while(cin>>d) v.push_back(d);   // read elements
    if (!cin.eof()) {           // check if input failed
        cerr << "format error\n";
        return 1;               // error return
    }

    cout << "read " << v.size() << " elements\n";

    reverse(v.begin(),v.end());
    cout << "elements in reverse order:\n";
    for (int i = 0; i<v.size(); ++i) cout << v[i] << '\n';

    return 0;                   // success return
}
```

ANFÜHRUNGSZEICHEN

Auch das Fehlen von Anführungszeichen ist ein häufiger Fehler. Denken Sie daran, dass Anführungszeichen Strings und alles, was auf dem Bildschirm oder in einer Datei ausgegeben werden soll, umschließen müssen. Die meisten Compiler-Fehler sind auf fehlende Anführungszeichen im Code zurückzuführen.



ZUSÄTZLICHE SEMIKOLONS

Am Ende einer jeden C++-Zeile muss ein Semikolon eingefügt werden, allerdings gibt es auch einige Ausnahmen. Semikolons müssen am Ende einer jeden vollständigen Anweisung stehen, einige Codezeilen sind jedoch keine vollständigen Anweisungen, wie z. B.:

```
#include
if lines
switch lines
```

Wenn dies etwas verwirrend klingt, machen Sie sich keine Sorgen, der Compiler teilt Ihnen mit, wo Sie den Fehler gemacht haben.

```
// Program to print positive number entered by the user
// If user enters negative number, it is skipped

#include <iostream>
using namespace std;

int main()
{
      int number;
      cout << "Enter an integer: ";
      cin >> number;

      // checks if the number is positive
      if ( number > 0)
      {
            cout << "You entered a positive integer: " << number << endl;
      }

      cout << "This statement is always executed.";
      return 0;

}
```

ZU VIELE KLAMMERN

Die Klammern bzw. geschweiften Klammern markieren den Anfang und das Ende eines Codeblocks. Für jede { muss es also auch eine } geben. Es kann leicht passieren, beim Schreiben von Code die eine oder andere geschweifte Klammer hinzuzufügen oder wegzulassen; meist passiert dies beim Schreiben in einem Texteditor, da die IDE sie für Sie hinzufügt.

```
using namespace std;

int main()
{
     int x;
     string mystring = "This is a string!\n";
     cout << "What's the value of x? ";
     cin >> x;
     cout << x;
     {
     cout << "\n\n";
     cout << mystring;

}
```

VARIABLEN INITIALISIEREN

In C++ werden Variablen nicht standardmäßig mit 0 initialisiert. Das bedeutet, wenn Sie eine Variable mit dem Namen x erstellen, erhält sie möglicherweise eine Zufallszahl von 0 bis 18 446 744 073 709 551 616, die sich nur schwer in eine Gleichung einfügen lässt. Geben Sie einer Variablen beim Erstellen zunächst den Wert null: `x=0`.

```
#include <iostream>
using namespace std;

int main()
{
    int x;
    x=0;

    cout << x;


}
```

A.OUT

Beim Kompilieren in Linux passiert häufig der Fehler, dass vergessen wird, den C++-Code nach der Kompilierung zu benennen. Wenn Sie vom Terminal aus kompilieren, geben Sie Folgendes ein:

```
g++ code.cpp
```

Dadurch wird der Code in der Datei code.cpp kompiliert und eine a.out-Datei erstellt, die mit ./a.out ausgeführt werden kann. Wenn Sie jedoch bereits Code in a.out haben, wird er überschrieben. Benutzen Sie folgende Eingabe:

```
g++ code.cpp -o nameofprogram
```

```
File  Edit  View  Search  Terminal  Help
david@mint-mate ~/Documents $ g++ test1.cpp
david@mint-mate ~/Documents $ ./a.out
0
david@mint-mate ~/Documents $ g++ test1.cpp -o printzero
david@mint-mate ~/Documents $ ./printzero
0
david@mint-mate ~/Documents $
```

Der nächste Schritt

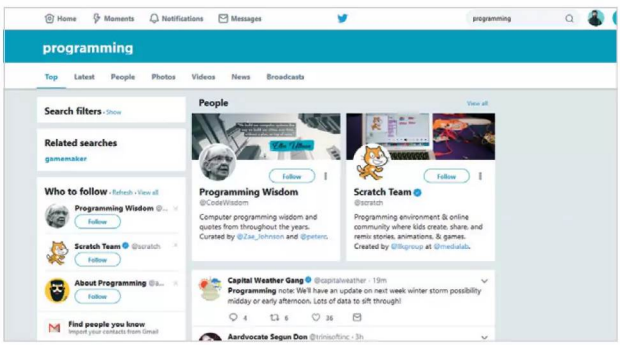
Das Programmieren ist eine kontinuierliche Lernerfahrung. Sie mögen zwar kein Anfänger mehr sein, müssen aber weiterhin Ihren Code testen, Tricks und Tipps lernen, um ihn effizienter zu gestalten, und sogar eine andere Programmiersprache lernen.

#INCLUDE<WEITERLERNEN>

Welche Möglichkeiten gibt es, um Ihre Fähigkeiten zu verbessern, neue Programmiermethoden zu erlernen, zu experimentieren, Ihren Code zu präsentieren und sogar anderen mit Ihren eigenen Erfahrungen zu helfen?

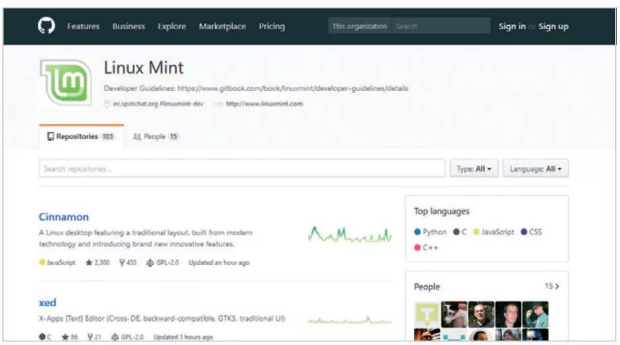
TWITTER

Twitter besteht nicht nur aus Trollen und Antagonisten. Man findet dort auch einige echte Menschen, die mehr als gewillt sind, ihre Programmierkenntnisse zu teilen. Wir empfehlen Ihnen, einige zu finden, mit denen Sie sich identifizieren und denen Sie folgen können. Sie können oftmals tolle Tipps, Tricks und Korrekturen für häufige Programmierprobleme finden.



OPEN-SOURCE-PROJEKTE

Suchen Sie nach Open-Source-Projekte, die Ihnen gefallen, und bieten Sie an, zum Code beizutragen. Es stehen Millionen von Projekten zur Auswahl. Kontaktieren Sie ein paar, und erfahren Sie, wo Hilfe benötigt wird. Auch wenn es sich dabei nur um geringfügige Code-Aktualisierungen handelt, so ist es doch eine ehrenwerte Aufgabe für Programmierer.



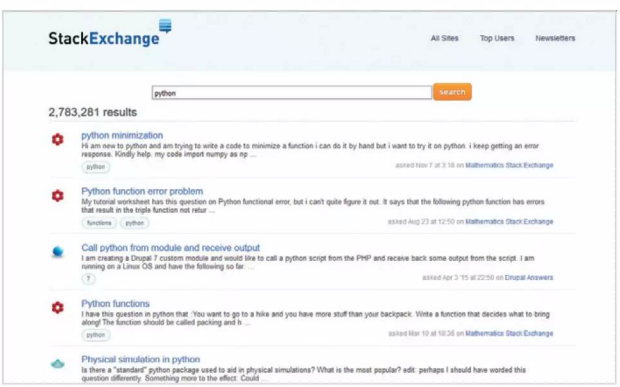
IMMER WEITER PROGRAMMIEREN

Wenn Sie Python schon recht gut beherrschen, werfen Sie einen Blick auf C++ oder sogar C#. Das Lernen einer neuen Programmiersprache hält das Gehirn in Schwung und Sie erhalten einen Einblick in eine andere Gemeinschaft und können sehen, was dort anders gemacht wird. Halten Sie Ihre Python-Kenntnisse dabei weiterhin auf dem Laufenden.

```
#pragma once

#define EXPAND_ME __FILE__ " : " __LINE__

constexpr auto pi = 314LLu;
thread_local decltype(pi) rage = 0b10;

[[deprecated("abc")]] char16_t *f() noexcept { return nullptr; }

template <typename T> struct Foo {
  static_assert(std::is_floating_point<T>::value,
                "Foo<T>: T must be floating point");
};

struct A final : Foo {
  A() = default;
  [[noreturn]] virtual void foo() override;
};
```

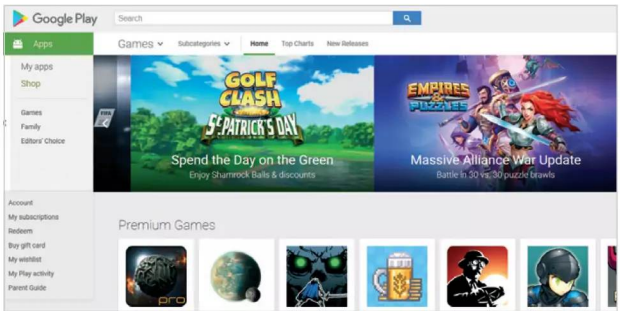
VONEINANDER LERNEN

Werden Sie auf Programmier- und Entwicklungswebsites wie StackExchange aktiv. Wenn Sie die Kenntnisse haben, können Sie anderen bei Ihren ersten Schritten helfen. Durch die Interaktion mit anderen Mitgliedern können Sie auch selbst viel lernen.



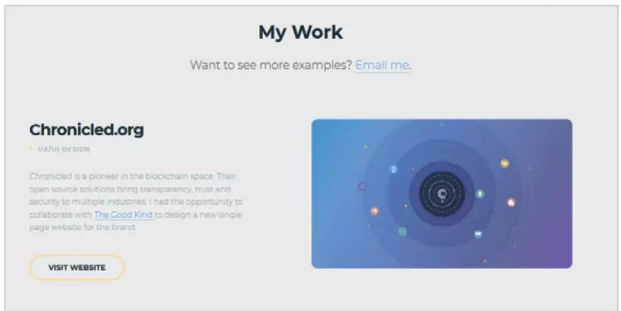
MOBILFUNKMARKT

Der Mobilfunkmarkt eignet sich hervorragend, um Ihre Programmierfähigkeiten zu testen und die von Ihnen erstellten Spiele oder Apps zu präsentieren. Wer weiß, wenn Ihre App gut ist, könnte sie der nächste große Hit sein, die in den App-Stores erscheint. Und wenn nicht, so ist es dennoch eine gute Lernerfahrung.



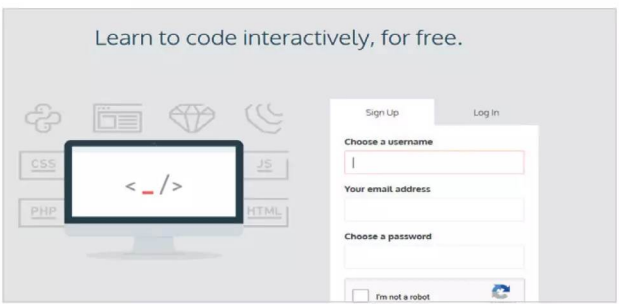
PORTFOLIOS

Wenn Sie zukünftig als Entwickler arbeiten möchten, sollten Sie ein Online-Portfolio mit Ihren Codes erwägen. Finden Sie heraus, welche Fähigkeiten erwartet werden, lernen und programmieren Sie etwas, das diese Fähigkeiten enthält, und fügen Sie sie dem Portfolio hinzu. Fügen Sie bei der Bewerbung einen Link zum Portfolio hinzu.



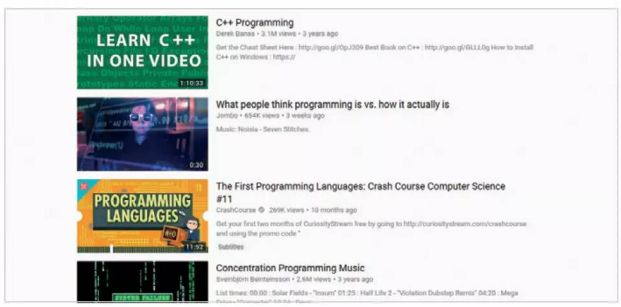
ONLINE LERNEN

Online-Kurse bieten eine gute Möglichkeit, um die nächsten Schritte im Programmieren zu machen. Häufig folgt ein Online-Kurs einem strengen Übereinkommen hinsichtlich der Programmierung.
Wenn Sie das Programmieren autodidaktisch lernen, könnte es sich lohnen, zu sehen, wie andere Entwickler ihren Code auslegen und was als akzeptabel gilt.



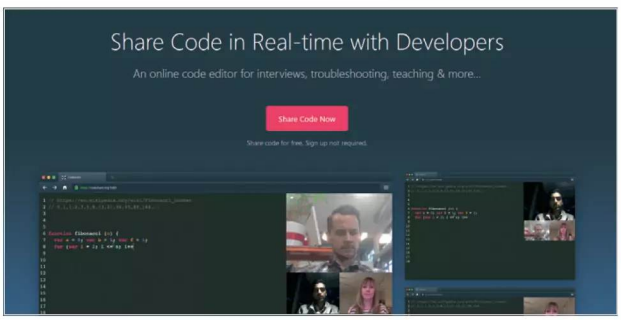
PROGRAMMIEREN LEHREN

Können Sie unterrichten? Wenn Ihre Programmierkenntnisse perfekt sind, könnten Sie sich z. B. an Volkshochschulen wenden, um zu sehen, ob Lehrer für die Programmierung benötigt werden, vielleicht für Teilzeit- oder Abendkurse. Wenn Sie nicht unterrichten, könnten Sie Ihren eigenen YouTube-Kanal erstellen, in dem Sie zeigen, wie man programmiert.



CODE TEILEN

Teilen Sie Ihren Code mit anderen, auch wenn Sie der Meinung sind, dass er nicht sehr gut ist. Die Kritik, Ratschläge und Kommentare, die Sie erhalten, helfen Ihnen, Probleme in Ihrem Code zu beheben. Vielleicht ist Ihr Code aber auch absolut fantastisch, was Sie aber nicht wissen werden, wenn Sie ihn nicht teilen.



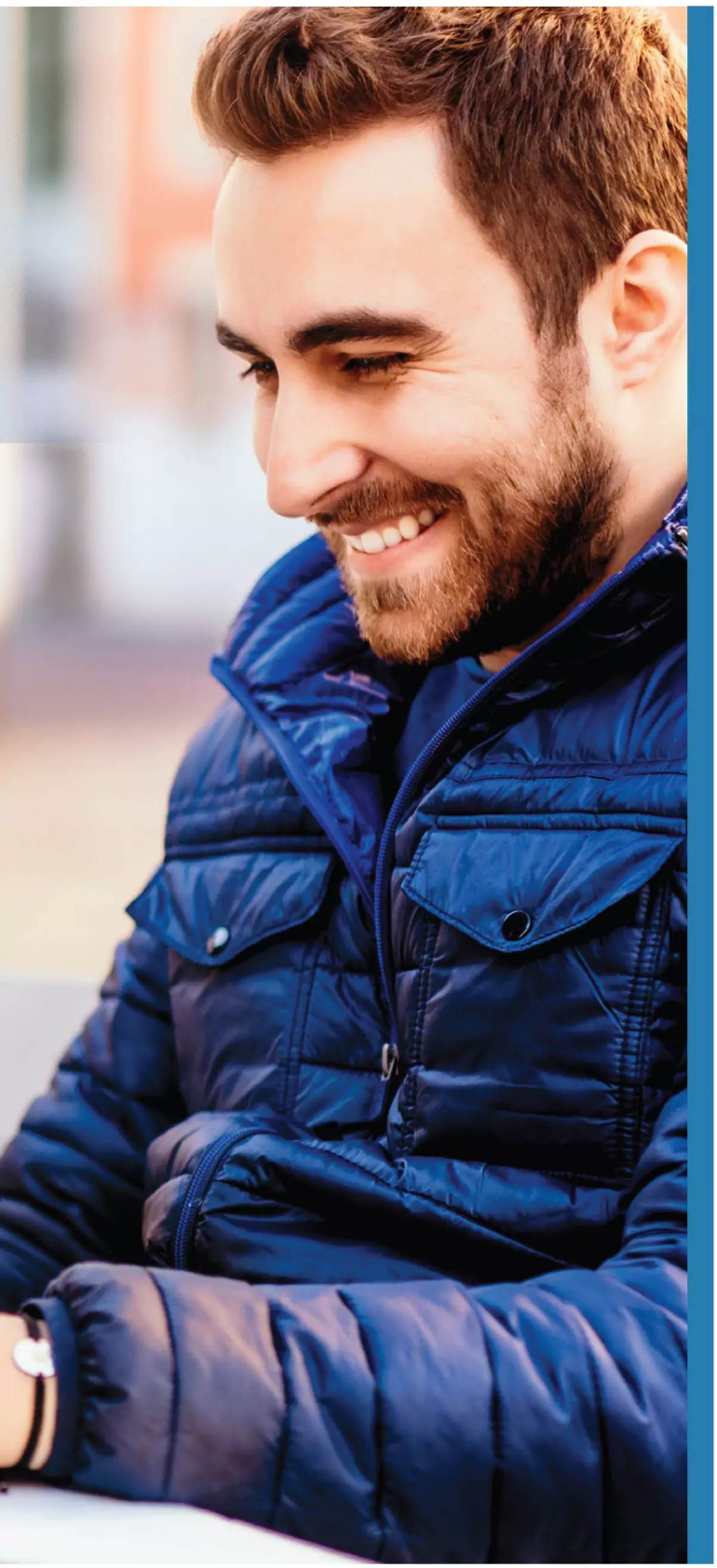
HARDWARE-PROJEKTE

Die Beteiligung an Hardware-Projekten ist eine tolle Möglichkeit, um seinen eigenen Code unter Beweis zu stellen und von anderen Mitwirkenden zu lernen. In vielen Entwicklerforen werden Projekte angeboten, für die sich Programmierer bewerben können, und in denen mit einzigartigen Codes das Beste aus der Hardware herausgeholt wird.

Willkommen bei Python

„Die besten Programme sind so geschrieben, dass sie von Rechnern schnell ausgeführt werden können und für den Menschen klar und verständlich sind."

– *Donald E. Knuth (Informatiker, Mathematiker und Autor)*



Es gibt viele verschiedene Programmiersprachen zum Lernen und Anwenden. Einige sind äußerst komplex und leistungsfähig, während andere extrem einfach sind und als Hilfsprogramme für das Betriebssystem dienen. Python befindet sich irgendwo in der Mitte und kombiniert Benutzerfreundlichkeit mit einer großen Leistungsfähigkeit, sodass der Benutzer Projekte erstellen kann, die von kleinen Hilfsprogrammen bis zu tollen Spielen und leistungsstarken Rechenaufgaben reichen.

Python ist jedoch mehr als nur eine andere Programmiersprache. Es hat eine dynamische und lebendige Community, in der Wissen, Code und Projektideen sowie Fehlerbehebungen für zukünftige Versionen ausgetauscht werden. Dieser Community ist es zu verdanken, dass die Sprache gewachsen ist und sich weiterentwickelt hat. Es liegt jetzt an Ihnen, den Sprung zu wagen und das Programmieren in Python zu erlernen.

Dieser Abschnitt hilft Ihnen bei den ersten Schritten mit Python und stellt Ihnen diese bemerkenswerte Sprache näher vor. Sie werden schon bald in der Lage sein, Ihre eigenen nützlichen Systemwerkzeuge und Textabenteuer zu programmieren und sogar eine Figur auf dem Bildschirm zu steuern.

Warum Python?

Es gibt viele verschiedene Programmiersprachen für den modernen Computer und auch für ältere 8- und 16-Bit-Computer sind einige erhältlich. Einige dieser Sprachen sind für wissenschaftliche Arbeiten konzipiert, andere für mobile Plattformen und dergleichen. Warum sollte man sich also ausgerechnet für Python entscheiden?

PYTHON-POWER

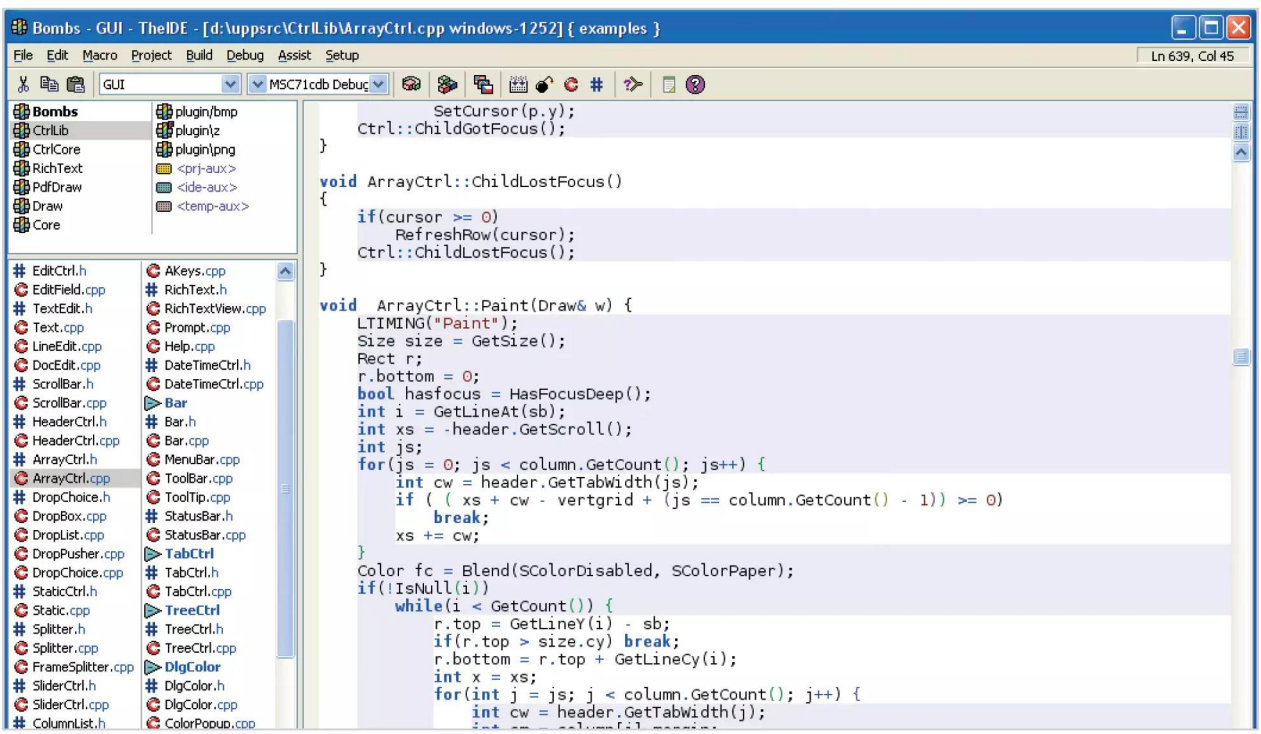
Seitdem die ersten Heimcomputer erhältlich waren, haben Enthusiasten, Benutzer und Profis bis in die frühen Morgenstunden gearbeitet und sich mit überhitzten Schaltkreisen herumgeplagt, um etwas zu erschaffen, das an Zauberei grenzt.

Diese Pioniere des Programmierens bahnten sich ihren Weg in das Neue und Unbekannte und entwickelten kleine Arbeitsabläufe, die es ermöglichten, dass der Buchstabe „A" über den Bildschirm wanderte. Dies mag für eine Generation, die Ultra-High-Definition-Grafiken und weltweites Multiplayer-Online-Gaming gewohnt ist, nicht sonderlich aufregend klingen, vor vierzig Jahren war es aber einfach brillant.

Diese Heim-Programmierer schufen die Basis für alle modernen digitalen Technologien. Einige stiegen zu Chefentwicklern für Top-Softwareunternehmen auf, während andere die verfügbare Hardware bis an ihre Grenzen brachte und das milliardenschwere Gaming-Imperium gründeten, das uns immer wieder aufs Neue überrascht.

Ganz gleich, ob Sie ein Android- oder iOS-Gerät, PC, Mac, Linux, Smart TV, eine Spielekonsole, einen MP3-Player, ein ins Auto eingebautes GPS-Gerät, eine Set-Top-Box oder tausend andere angeschlossene und „intelligente" Geräte verwenden, alle basieren auf Programmierung.

All diese erwähnten digitalen Geräte benötigen Anweisungen, die ihnen befehlen, was zu tun ist, und die eine Kommunikation mit ihnen ermöglichen. Diese Anweisungen bilden den Programmierkern des Geräts, der sich wiederum mithilfe einer Vielzahl von Programmiersprachen erstellen lässt.



C++ ist normalerweise komplexeren Programmen, Betriebssystemen, Spielen usw. vorbehalten.

Die heute verwendeten Programmiersprachen unterscheiden sich je nach Situation, Plattform, Verwendung des Geräts und nach dem, wie das Gerät mit seiner Umgebung oder seinen Benutzern kommuniziert. Betriebssysteme wie Windows, macOS usw. sind in der Regel eine Kombination aus C++, C#, Assembly und einer Form visueller Programmiersprache. Spiele verwenden im Allgemeinen C++, während für Webseiten eine Vielzahl von Sprachen wie HTML, Java, Python usw. zur Verfügung steht.

Universelle Programmiersprachen werden für die Entwicklung von z. B. Programmen, Apps und Software verwendet. Sie sind auf allen Hardwareplattformen weit verbreitet und eignen sich für nahezu jede denkbare Anwendung. Einige arbeiten schneller, während andere leichter zu erlernen und anzuwenden sind. Python ist eine solche universelle Sprache.

Python ist eine sogenannte höhere Programmiersprache, da sie über eine Vielzahl von Arrays, Variablen, Objekten, Arithmetik, Unterprogrammen, Schleifen und unzähligen weiteren Interaktionen mit der Hardware und dem Betriebssystem kommuniziert. Obwohl nicht so effizient gestaltet wie eine niedrige Programmiersprache, die mit Speicheradressen, Call-Stacks und Registern umgehen kann, hat Python den Vorteil, allgemein zugänglich und leicht erlernbar zu sein.

```
1  //file: Invoke.java
2  import java.lang.reflect.*;
3
4  class Invoke {
5    public static void main( String [] args ) {
6      try {
7        Class c = Class.forName( args[0] );
8        Method m = c.getMethod( args[1], new Class
         [] { } );
9        Object ret =  m.invoke( null, null );
10       System.out.println(
11           "Invoked static method: " + args[1]
12           + " of class: " + args[0]
13           + " with no args\nResults: " + ret );
14     } catch ( ClassNotFoundException e ) {
15       // Class.forName(  ) can't find the class
16     } catch ( NoSuchMethodException e2 ) {
17       // that method doesn't exist
18     } catch ( IllegalAccessException e3 ) {
19       // we don't have permission to invoke that
         method
20     } catch ( InvocationTargetException e4 ) {
21       // an exception ocurred while invoking that
         method
22       System.out.println(
23           "Method threw an: " + e4.
               getTargetException(  ) );
24     }
25   }
```

Java ist eine leistungsstarke Programmiersprache, die für Webseiten, Set-Top-Boxen, Fernseher und sogar Autos verwendet wird.

Python wurde vor über 26 Jahren entworfen und hat sich zu einer idealen Anfängersprache für das Programmieren eines Computers entwickelt. Python ist perfekt für Hobbyisten, Enthusiasten, Studenten, Lehrer und diejenigen, die ihre eigene einzigartige Interaktion mit dem Computer erstellen möchten.

Python kann kostenlos heruntergeladen, installiert und genutzt werden und ist für Linux, Windows, macOS, MS-DOS, OS/2, BeOS, IBM i-Serien und sogar RISC OS verfügbar. Es wurde zu einer der fünf führenden Programmiersprachen der Welt gewählt und ist bei der Hardware- und Internetentwicklung ständig einen Schritt voraus.

Die Frage „Warum Python?" lässt sich also damit beantworten, dass Python kostenlos, einfach zu erlernen, äußerst leistungsfähig, allgemein akzeptiert, effektiv und ein ausgezeichnetes Lernwerkzeug ist.

```
  40 LET py=15
  70 FOR w=1 TO 10
  71 CLS
  75 LET by=INT (RND*28)
  80 LET bx=0
  90 FOR d=1 TO 20
 100 PRINT AT px,py;" U "
 110 PRINT AT bx,by;"o"
 120 IF INKEY$="p" THEN LET py=p
y+1
 130 IF INKEY$="o" THEN LET py=p
y-1                 U
 135 FOR n=1 TO 100: NEXT n
 140 IF py<2 THEN LET py=2
 150 IF py>27 THEN LET py=27
 180 LET bx=bx+1
 185 PRINT AT bx-1,by;" "
 190 NEXT d
 200 IF (by-1)=py THEN LET s=s+1
 210 PRINT AT 10,10;"score=";s
 220 FOR v=1 TO 1000: NEXT v
 300 NEXT w

0 OK, 0:1
```

BASIC war einst die Einstiegssprache der Benutzer früher 8-Bit-Heimcomputer.

```
print(HANGMAN[0])
attempts = len(HANGMAN) - 1


while (attempts != 0 and "-" in word_guessed):
    print(("\nYou have {} attempts remaining").format(attempts))
    joined_word = "".join(word_guessed)
    print(joined_word)

    try:
        player_guess = str(input("\nPlease select a letter between A-Z" + "\n> ")).
    except: # check valid input
        print("That is not valid input. Please try again.")
        continue
    else:
        if not player_guess.isalpha(): # check the input is a letter. Also checks a
            print("That is not a letter. Please try again.")
            continue
        elif len(player_guess) > 1: # check the input is only one letter
            print("That is more than one letter. Please try again.")
            continue
        elif player_guess in guessed_letters: # check it letter hasn't been guessed
            print("You have already guessed that letter. Please try again.")
            continue
        else:
            pass

    guessed_letters.append(player_guess)

    for letter in range(len(chosen_word)):
        if player_guess == chosen_word[letter]:
            word_guessed[letter] = player_guess # replace all letters in the chosen

    if player_guess not in chosen_word:
```

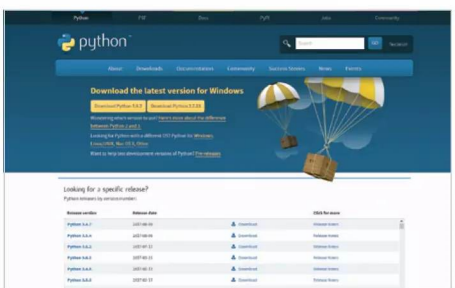
Python ist eine moderne BASIC-Version, die einfach zu lernen und eine ideale Programmiersprache für Einsteiger ist.

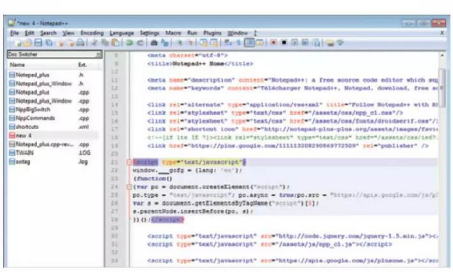
Benötigtes Zubehör

Zum Erlernen von Python brauchen Sie nicht viel Hardware und auch finanzielle Ausgaben sind für den Einstieg nicht notwendig, da ein leistungsstarker Computer nicht erforderlich und die benötigte Software kostenlos ist.

UNSER ZUBEHÖR

Zum Glück ist Python eine Multiplattform-Programmiersprache für Windows, macOS, Linux, Raspberry Pi usw. Wenn Sie eines dieser Systeme haben, können Sie mit Python loslegen.





COMPUTER

Natürlich brauchen Sie einen Computer, um zu lernen, wie man in Python programmiert und um Ihren Code zu testen. Sie können Windows (ab XP) mit einem 32- oder 64-Bit-Prozessor, einen Apple Mac oder einen Linux-PC verwenden.

IDE

Mit einer IDE (Integrierte Entwicklungsumgebung) wird Python-Code eingegeben und ausgeführt. Sie können damit Ihren Programmcode und die Werte innerhalb des Codes überprüfen und erweiterte Funktionen nutzen. Es gibt viele verschiedene IDEs, suchen Sie daher nach einer, die für Sie funktioniert und die besten Ergebnisse liefert.

PYTHON-SOFTWARE

Python ist in macOS, Linux und im Raspberry Pi bereits als Teil des Betriebssystems vorinstalliert. Sie müssen jedoch sicherstellen, dass Sie die neueste Version von Python ausführen. Windows-Benutzer müssen Python erst herunterladen und installieren, was wir uns in Kürze anschauen werden.

TEXTEDITOR

Ein Texteditor bietet zwar eine ideale Umgebung für die Eingabe von Code, ist aber nicht unbedingt erforderlich. Sie können Code direkt über die IDLE eingeben und ausführen, allerdings bietet ein Texteditor wie z. B. Sublime Text oder Notepad++ erweiterte Funktionen und Farbcodierungen bei der Codeeingabe.

INTERNETZUGANG

Python wird ständig weiterentwickelt, um eine effizientere Sprache zu schaffen. Aufgrunddessen werden mit neuen Versionen oft neue Konzepte oder veränderte Befehle und Code-Strukturen eingeführt. Über das Internet bleiben Sie auf dem Laufenden, erhalten Hilfe bei möglichen Problemen sowie Zugang zu Pythons unzähligen Modulen.

ZEIT UND GEDULD

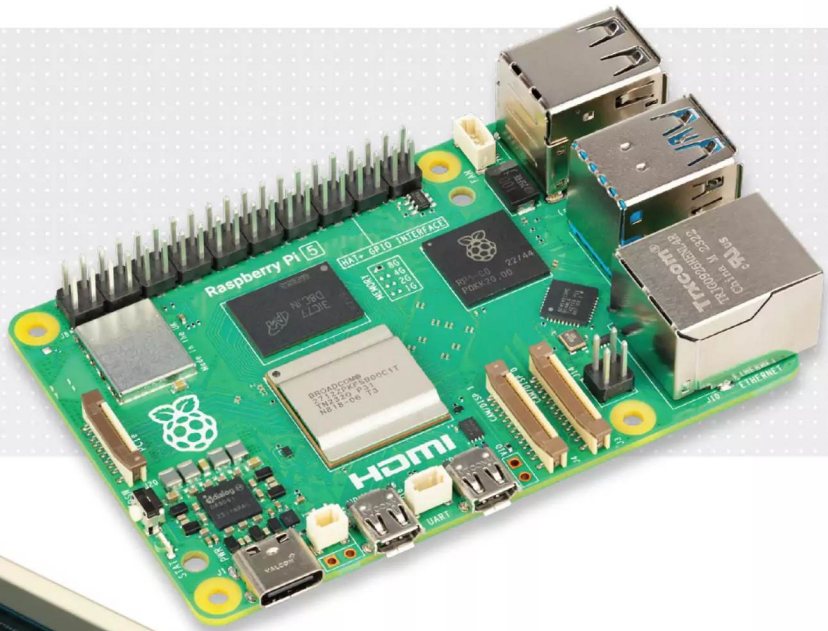
Ganz gleich, was Ihnen andere Bücher versprechen, Sie werden in 24 Stunden kein Programmierer werden. Dies nimmt Zeit und Geduld in Anspruch. Manches ist einfach verständlich, anderes dauert etwas länger. Wenn Sie sich bewusst sind, dass Sie etwas völlig Neues lernen, werden Sie Ihr Ziel erreichen.

RASPBERRY PI

Warum einen Raspberry Pi verwenden? Der Raspberry Pi ist ein kleiner und äußerst günstiger Computer, der eine fantastische Lernplattform bietet. Sein Hauptbetriebssystem Raspbian enthält bereits die neueste Python-Version sowie viele Module und Extras.

RASPBERRY PI

Das Raspberry Pi 5-Modell ist die neueste Version. mit einer leistungsstärkeren CPU, wahlweise 4 GB oder 8-GB-Speicherversionen sowie Wi-Fi- und Bluetooth-Unterstützung. Sie können einen Pi 5 ab ca. 70 € erwerben, der Preis steigt auf bis zu 95 € für die 8-GB-Speicherversion oder als Teil eines Kits Abhängig vom Pi-Modell, an dem Sie interessiert sind.

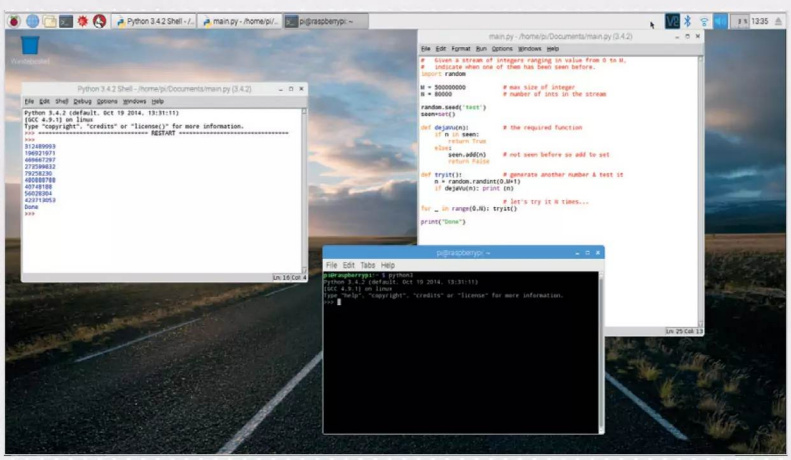


FUZE-PROJEKT

FUZE ist eine Lernumgebung, die auf dem neuesten Modell des Raspberry Pi basiert. Sie können die Workstations kaufen, die für die Elektronik-Bausätze erhältlich sind, und sogar einen Roboterarm bauen und programmieren. Weitere Informationen finden Sie unter **www.fuze.co.uk**.

RASPBIAN

Das Hauptbetriebssystem des Raspberry Pi ist eine auf Debian basierende Linux-Distribution, die alles enthält, was ein einfach zu verwendendes Paket benötigt. Raspbian ist für den Pi optimiert und bietet eine ideale Plattform für Hardware- und Software-Projekte, für die Python-Programmierung und sogar als Desktop-Computer.



Lernen Sie Python kennen

Python ist die beste Programmiersprache, die jemals erfunden wurde. Mithilfe von Pythons klarer und leicht verständlicher Sprache können Sie das Leistungspotenzial Ihres Computers vollständig ausschöpfen.

WAS BEDEUTET PROGRAMMIEREN?

Vor dem Lernen einer Programmiersprache ist es hilfreich deren Bedeutung zu verstehen, und Python bildet da keine Ausnahme. Hier werfen wir einen Blick auf die Geschichte Pythons und ihren Bezug zu anderen Programmiersprachen.

PYTHON

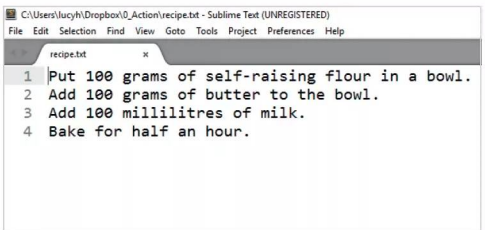
Eine Programmiersprache besteht aus einer Liste mit Anweisungen, die ein Computer befolgt. Dabei kann es sich um einfache Anweisungen handeln, wie der Anzeige Ihres Namens oder dem Abspielen einer Musik-Datei, oder komplexe, wie dem Erstellen einer virtuellen Welt. Python wurde Ende der 80er von Guido van Rossum im Centrum Wiskunde & Informatica (CWI) in den Niederlanden als Nachfolger der ABC-Sprache entwickelt.

Guido van Rossum, der Vater von Python.

PROGRAMMIERREZEPTE

Programme sind praktisch Rezepte für Computer. Hier haben wir ein Kuchenrezept:

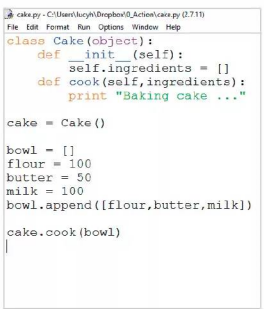
Put 100 grams of self-raising flour in a bowl.
Add 100 grams of butter to the bowl.
Add 100 millilitres of milk.
Bake for half an hour.



CODE

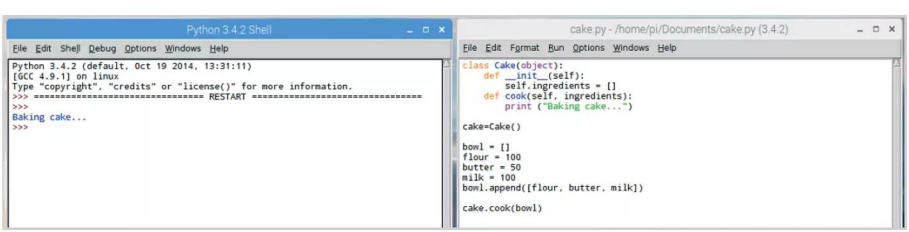
Genau wie unser Kuchenrezept besteht auch ein Programm aus Anweisungen, die der Reihe nach befolgt werden. Unser Kuchenrezept könnte als Programm wie folgt aussehen:

```
bowl = []
flour = 100
butter = 50
milk = 100
bowl.append([flour,butter,milk])
cake.cook(bowl)
```



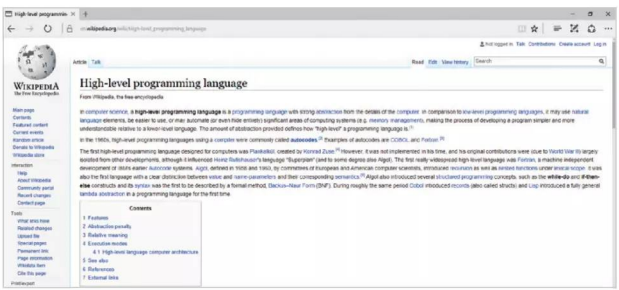
PROGRAMMIERBEFEHLE

Sie werden vielleicht einige der Python-Befehle wie bowl.append und cake.cook(bowl) nicht verstehen. Der erste Teil ist eine Liste, der zweite ein Objekt; wir werden uns aber mit beiden noch genauer befassen. Wichtig ist, dass man weiß, dass Befehle in Python einfach zu lesen sind. Wenn man erst mal weiß, wofür die Befehle stehen, ist es einfacher zu verstehen, wie ein Programm funktioniert.



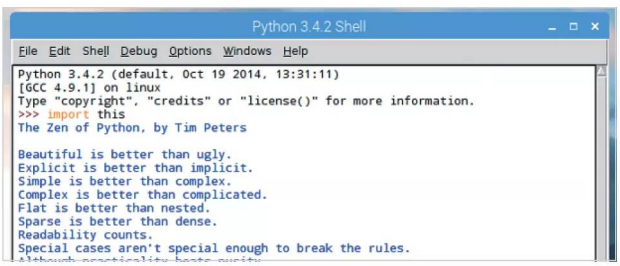
HÖHERE PROGRAMMIERSPRACHEN

Computersprachen, die leicht zu lesen sind, sind als höhere Programmiersprachen (High-Level) bekannt, da sie sozusagen über der Hardware sitzen. Sprachen, die näher an der Hardware liegen, wie z. B. Assembly, sind niedrige Programmiersprachen (Low-Level). Die Befehle der niedrigen Programmiersprachen sehen in etwa wie folgt aus: `msg db ,0xa len equ $ - msg`.



DAS ZEN DES PYTHON

Mit Python können Sie auf das Leistungspotenzial eines Computers in einer Sprache zugreifen, die von Menschen verstanden wird. Hinter all dem verbirgt sich ein Ethos, der „Zen of Python". Dies ist eine Sammlung von 20 Software-Prinzipien, die das Design der Programmiersprache beeinflussen. Zu den Prinzipien gehören „Schön ist besser als hässlich" und „Einfach ist besser als kompliziert". Geben Sie in Python `import this` ein, um die Prinzipien aufzulisten.



PYTHON 3 VS. PYTHON 2

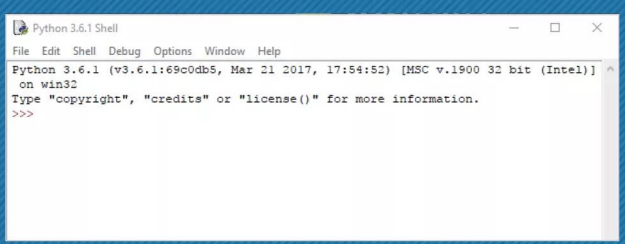
Genau wie in einem typischen Computerszenario macht die Existenz von zwei aktiven Versionen der Sprache – Python 2 und Python 3 – alles etwas komplizierter.

DIE WELT VON PYTHON

Python 3.7 ist die neueste Version der Programmiersprache. Wenn Sie jedoch online nach Python-Code suchen, werden Sie zweifellos auf Python 2 stoßen. Obwohl Sie Python 3 und Python 2 nebeneinander ausführen können, ist dies nicht zu empfehlen. Entscheiden Sie sich immer für die neueste stabile Version, die auf der Python-Website veröffentlicht wird.

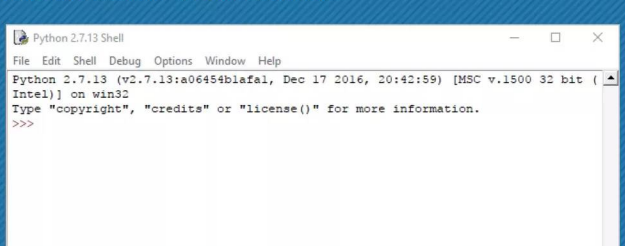
PYTHON 3.X

Im Jahr 2008 erschien Python 3 mit neuen und verbesserten Funktionen, die eine stabilere, effektivere und effizientere Programmierumgebung bieten. Leider sind die meisten (wenn auch nicht alle) der neuen Funktionen nicht mit Python 2 Scripts, Modulen und Tutorials kompatibel. Obwohl zu Beginn nicht sehr populär, hat sich Python 3 mittlerweile zu einer der führenden Python-Programmiersprachen entwickelt.



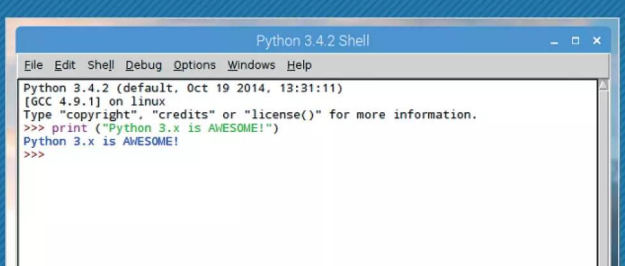
PYTHON 2.X

Warum gibt es nun aber zwei Versionen? Python 2 wurde im Jahr 2000 veröffentlicht und hat seitdem eine recht große Sammlung von Modulen, Scripts, Benutzern, Tutorials usw. angehäuft. Im Laufe der Jahre hat sich Python 2 schnell zu einer der ersten Programmiersprachen für Anfänger und Experten entwickelt, was diese Version zu einer äußerst nützlichen Ressource macht.



3.X GEWINNT

Die wachsende Popularität von Python 3 hat dazu geführt, mit der Entwicklung der neuen Funktionen zu beginnen und die vorherige Version auslaufen zu lassen. Viele Entwicklungsunternehmen wie z. B. SpaceX und die NASA verwenden Python 3 für wichtige Codeabschnitte.



Python unter Windows einrichten

Windows-Benutzer können die neueste Python-Version ganz leicht über die Python-Downloads-Seite installieren. Die meisten erfahrenen Python-Entwickler lehnen Windows zwar als bevorzugte Plattform zum Schreiben von Code ab, für Anfänger ist sie aber ideal.

PYTHON 3.X INSTALLIEREN

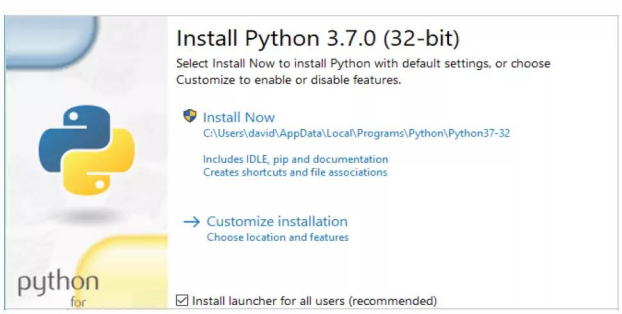
Python ist in Microsoft Windows nicht standardmäßig enthalten und muss daher manuell installiert werden. Glücklicherweise ist dies jedoch ein einfacher Vorgang.

SCHRITT 1

Öffnen Sie in Windows Ihren Webbrowser und gehen Sie zu python.org/downloads/. Halten Sie nach dem Link mit dem Download für Python 3.x. Ausschau. Bei Redaktionsschluss war die aktuelle Version 3.7.0, aber da Python regelmäßig aktualisiert wird, kann es eine spätere Version geben.

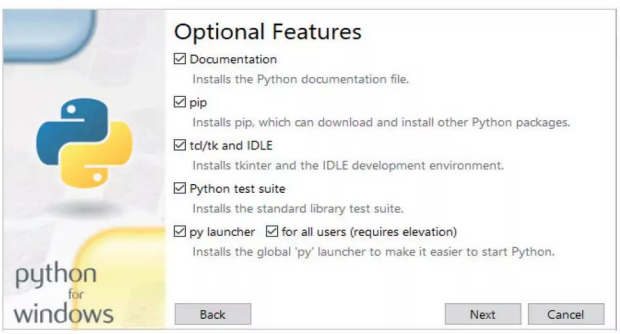
SCHRITT 2

Klicken Sie auf die Schaltfläche für den Download der Version 3.x und speichern Sie die Datei in Ihrem Downloads-Ordner. Machen Sie nach dem Herunterladen auf der exe.-Datei einen Doppelklick, um den Python-Installationsassistenten zu öffnen. Sie erhalten zwei Optionen: Install Now und Customise Installation. Wir empfehlen die Option „Customise Installation".



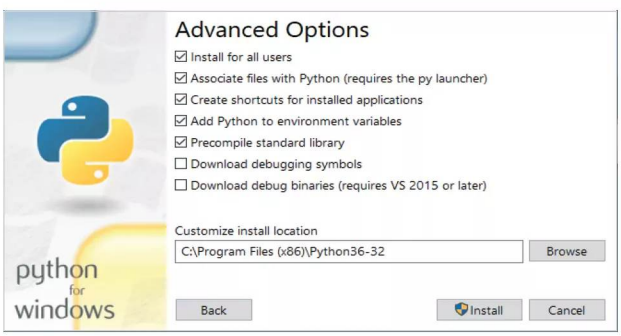
SCHRITT 3

Diese Option lässt Sie bestimmte Parameter eingeben, und auch wenn Sie die Standardeinstellungen beibehalten, ist sie eine gute Wahl, da Installierprogramme (mit Ausnahme von Python) manchmal unerwünschte zusätzliche Funktionen enthalten. Stellen Sie auf dem ersten Bildschirm sicher, dass alle Kontrollkästchen aktiviert sind, und klicken Sie auf „Next".



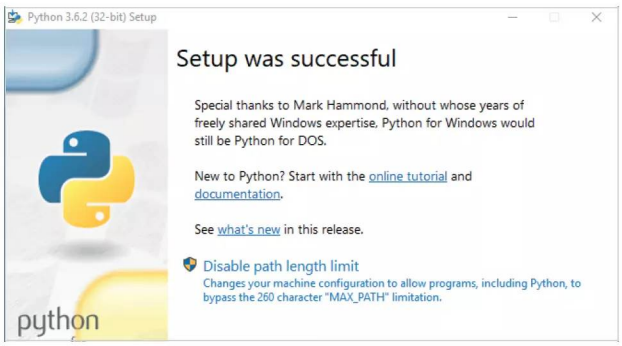
SCHRITT 4

Die Optionen auf der nächsten Seite bieten einige interessante Ergänzungen zu Python. Stellen Sie sicher, dass die Optionen Install for all users, Associate files, Create shortcuts, Add Python und Precompile standard library mit einem Häkchen versehen sind, da sie die Anwendung von Python erleichtern. Klicken Sie auf „Install", um fortzufahren.



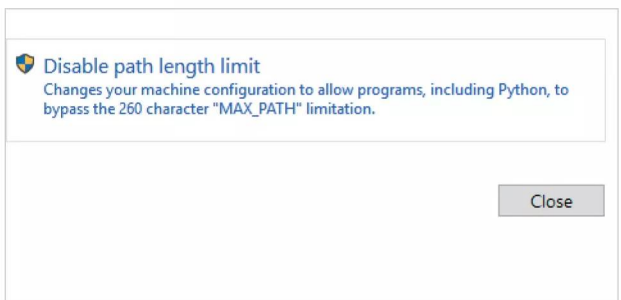
SCHRITT 5

Eventuell müssen Sie die Installation mit der Windows-Authentifizierungsbenachrichtigung bestätigen. Klicken Sie einfach auf Ja und Python wird mit der Installation beginnen. Nach der Installation finden Sie auf der letzten Seite des Python-Assistenten die neuesten Versionshinweise sowie einige Online-Tutorials.



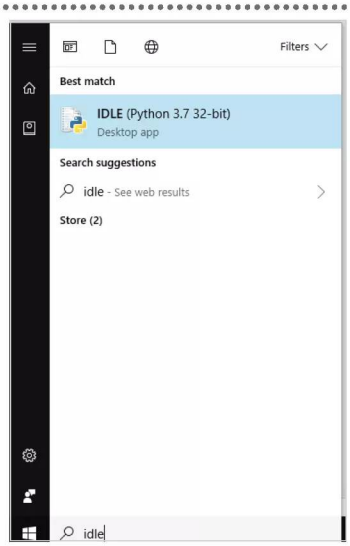
SCHRITT 6

Bevor Sie das Fenster des Installationsassistenten schließen, klicken Sie am besten auf den Link neben dem Schild, um die Pfadlängenbeschränkung zu deaktivieren. Dadurch kann Python die 260-Zeichenbeschränkung von Windows umgehen, sodass Sie Python-Programme ausführen können, die in tief verschachtelten Ordnern gespeichert sind. Klicken Sie erneut auf Ja, um den Vorgang zu bestätigen und schließen Sie das Installationsfenster.



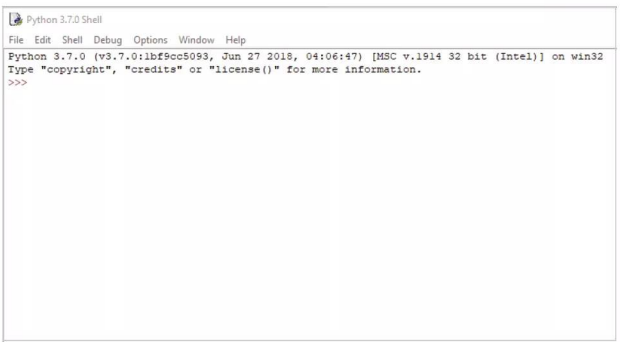
SCHRITT 7

Windows 10-Benutzer finden das installierte Python 3.x im Startmenü unter dem Abschnitt „Zuletzt hinzugefügt". Der erste Link, Python 3.7 (32-Bit), startet die Befehlszeilenversion von Python (dazu in Kürze mehr). Um die IDLE zu öffnen, geben Sie im Windows-Suchfeld IDLE ein.



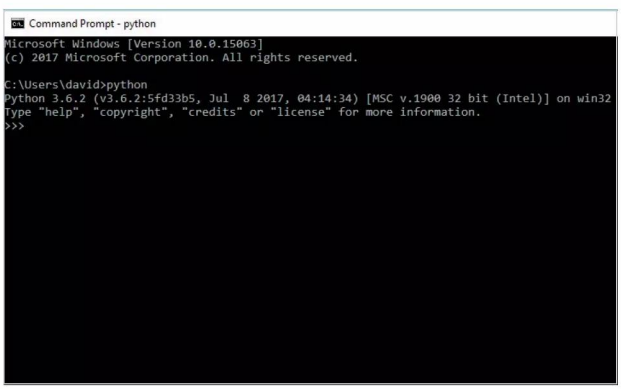
SCHRITT 8

Wenn Sie auf den IDLE-Link (Python 3.7 32-Bit) klicken, wird die Python-Shell gestartet, in der Sie Ihre Reise durch die Programmierung in Python beginnen können. Es ist kein Problem, wenn Ihre Version neuer ist, solange es sich um Python 3.x handelt, wird unser Code auch auf Ihrer Python 3-Version funktionieren.



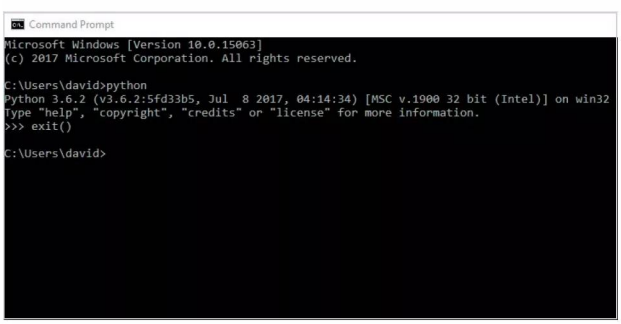
SCHRITT 9

Wenn Sie nun im Windows-Suchfeld `CMD` eingeben, wird der Link zur Eingabeaufforderung angezeigt. Klicken Sie darauf, um zur Windows-Befehlszeilenumgebung zu gelangen. Um Python über die Befehlszeile zu starten, müssen Sie `python` eingeben und die Eingabetaste drücken.



SCHRITT 10

Die Befehlszeilenversion von Python funktioniert ähnlich wie die in Schritt 8 geöffnete Shell. Beachten Sie die drei nach links zeigenden Pfeile (>>>). Obwohl sie eine perfekte Umgebung ist, ist sie nicht allzu benutzerfreundlich, von daher lassen wir fürs Erste die Finger von der Befehlszeile. Geben Sie `exit()` ein, um das Eingabeaufforderungsfenster zu verlassen und zu schließen.



Python unter Linux einrichten

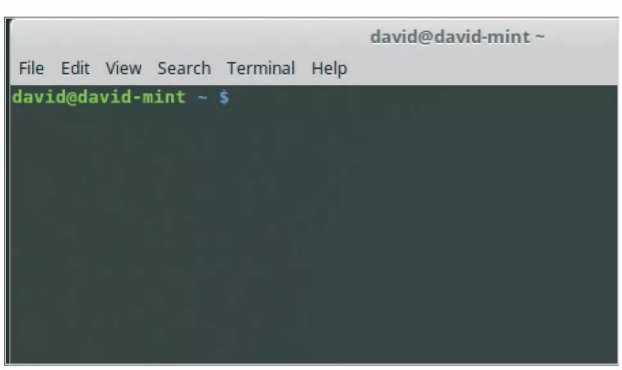
Python 2.x ist in den meisten Linux-Distributionen bereits vorinstalliert, da wir aber Python 3.x verwenden werden, müssen wir erst einige Schritte ausführen, die aber zum Glück nicht schwierig sind.

PYTHON UND DER PINGUIN

Das Linux-Betriebssystem ist äußerst vielseitig und da auf den verschiedenen Distributionen Software auf unterschiedliche Weise installiert wird, halten wir uns in diesem speziellen Tutorial an Linux Mint 18.1.

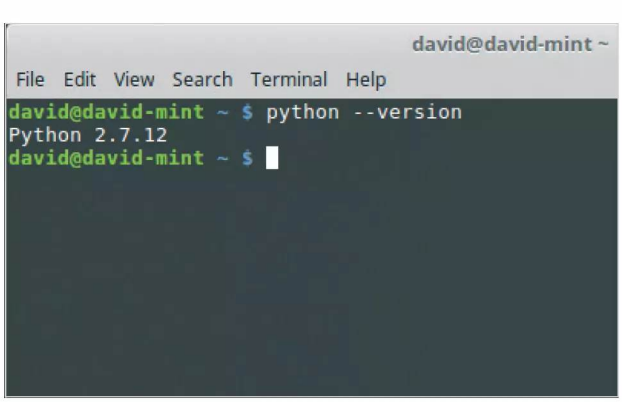
SCHRITT 1

Zuerst müssen Sie herausfinden, welche Version von Python derzeit auf Ihrem Linux-System installiert ist. Wie bereits erwähnt, werden wir Linux Mint 18.1 für diesen Abschnitt verwenden. Öffnen Sie durch Drücken von Strg + Alt + T das Terminal.



SCHRITT 2

Geben Sie als Nächstes `python --version` im Terminal ein. Die Ausgabe sollte sich auf die Version 2.x von Python beziehen, in unserem Fall ist es Python 2.7.12.

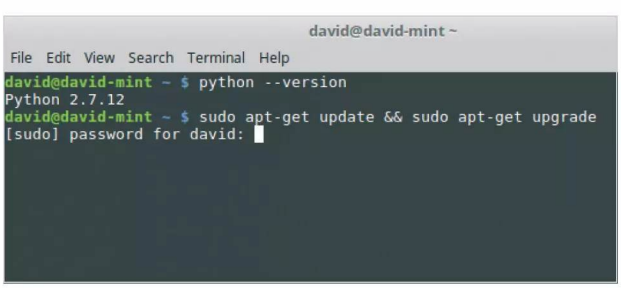


SCHRITT 3

Einige Linux-Distributionen aktualisieren beim Update des Systems Python automatisch auf die neueste Version. Um dies zu überprüfen, führen Sie zuerst mit dem folgendem Befehl ein System-Update und ein Upgrade durch:

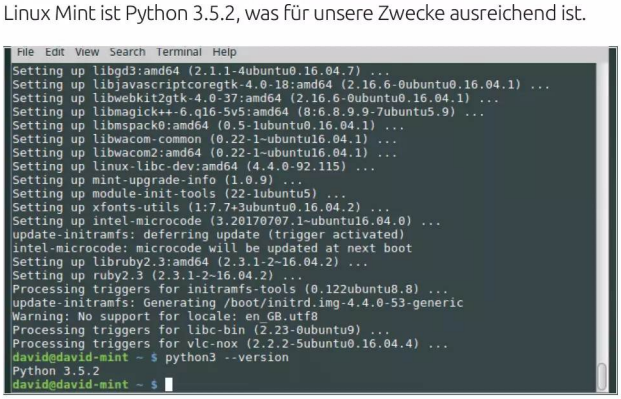
```
sudo apt-get update && sudo apt-get upgrade
```

Geben Sie Ihr Passwort ein und lassen Sie das System die Aktualisierungen ausführen.



SCHRITT 4

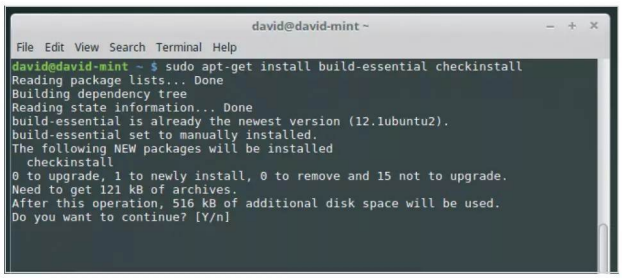
Nach Abschluss der Aktualisierungen müssen Sie evtl. mit „J" antworten, um diese zu autorisieren. Geben Sie `python3 --version` ein, um zu sehen, ob Python 3.x aktualisiert oder sogar installiert wurde. Unsere Version für Linux Mint ist Python 3.5.2, was für unsere Zwecke ausreichend ist.



SCHRITT 5

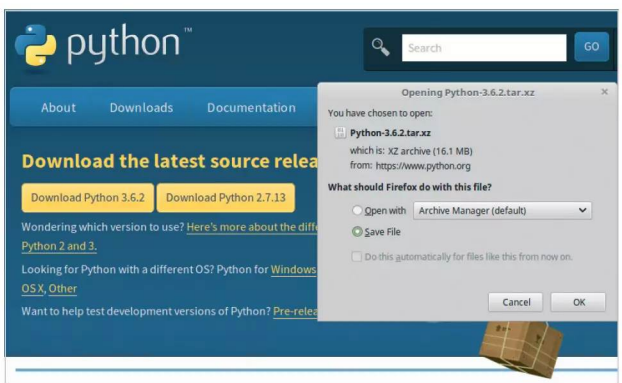
Wenn Sie jedoch die neueste Version (bei Redaktionsschluss gemäß der Python-Webseite 3.6.2) verwenden möchten, müssen Sie Python vom Quellcode erstellen. Geben Sie diese Befehle ins Terminal ein:

```
sudo apt-get install build-essential checkinstall
sudo apt-get install libreadline-gplv2-dev
libncursesw5-dev libssl-dev libsqlite3-dev tk-dev
libgdbm-dev libc6-dev libbz2-dev
```



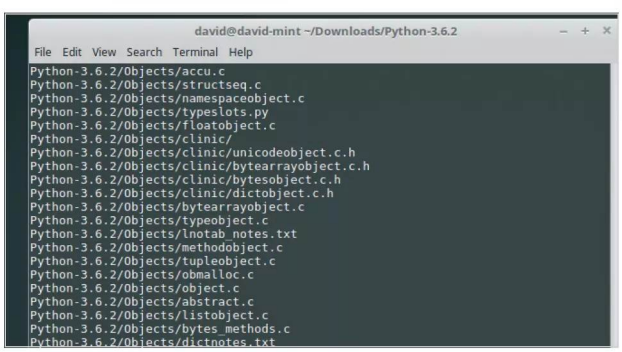
SCHRITT 6

Öffnen Sie Ihren Linux-Webbrowser und gehen Sie zur Python-Download-Seite **www.python.org/downloads**. Klicken Sie auf „Download Python 3.6.2 (bzw. die derzeit aktuelle Version), um die Quelldatei Python-3.6.2.tar.xz herunterzuladen.



SCHRITT 7

Gehen Sie im Terminal durch Eingabe des Befehls `cd Downloads/` zum Downloads-Ordner. Entpacken Sie anschließend mithilfe von `tar -xvf Python-3.6.2.tar.xz` den Inhalt des heruntergeladenen Python-Quellcodes. Öffnen Sie nun den neu entpackten Ordner mit dem Befehl `cd Python-3.6.2/`.

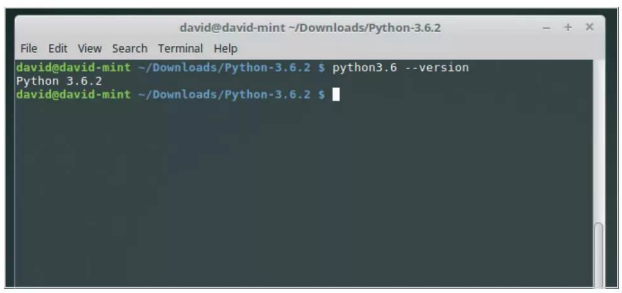


SCHRITT 8

Geben Sie innerhalb des Python-Ordners Folgendes ein:

```
./configure
sudo make altinstall
```

Dies kann abhängig von der Geschwindigkeit Ihres Computers etwas dauern. Geben Sie nach Beendigung `python3.6 --version` ein, um die installierte Version zu überprüfen.



SCHRITT 9

Für die GUI IDLE müssen Sie den folgenden Befehl ins Terminal eingeben:

```
sudo apt-get install idle3
```

Die IDLE kann dann mit dem Befehl `idle3` gestartet werden. Beachten Sie, dass IDLE eine andere Version ausführt, als die Sie vom Quellcode installiert haben.

SCHRITT 10

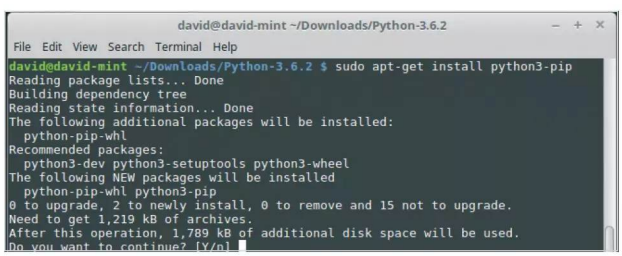
Sie benötigen zusätzlich PIP (Pip Installs Packages), ein Tool zur Installation weiterer Module und Extras. Geben Sie zur Installation von PIP Folgendes ein:

```
sudo apt-get install python3-pip
```

Suchen Sie mit dem folgenden Befehl nach den neuesten Updates:

```
pip3 install --upgrade pip
```

Schließen Sie nach Beendigung das Terminal. Python 3.x ist im Menü der Distribution über den Bereich Entwicklung erhältlich.



Erste Schritte mit Python

„Ich habe mir immer gewünscht, dass mein Computer so einfach zu bedienen ist wie mein Telefon. Mein Wunsch ist in Erfüllung gegangen, denn ich kann nicht mehr herausfinden, wie mein Telefon funktioniert."

– *Bjarne Stroustrup (Entwickler und Erfinder von C++)*

Das Programmieren zu erlernen mag auf den ersten Blick etwas entmutigend erscheinen. Zum Glück wurde die Python-Sprache in Hinblick auf Einfachheit entworfen. Wie bei den meisten Dingen müssen Sie langsam anfangen und lernen, wie Sie mit dem Code das gewünschte Ergebnis erhalten.

In diesem Abschnitt werden die wichtigsten Konzepte behandelt: Code speichern und ausführen, Variablen, Zahlen und Ausdrücke, Benutzereingaben, Bedingungen und Schleifen.

Python zum ersten Mal starten

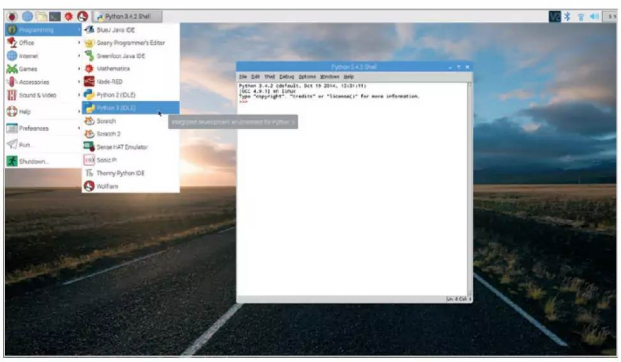
Der Raspberry Pi eignet sich hervorragend zum Programmieren lernen, insbesondere mit Python. Raspbian, das vom Pi empfohlene Betriebssystem, hat die neueste stabile Version von Python 3 bereits vorinstalliert, und ist daher eine großartige Programmierplattform.

PYTHON STARTEN

Wir werden uns nicht im Einzelnen damit befassen, wie wir den Raspberry Pi zum Laufen bringen, da zu diesem Thema bereits reichlich Material vorhanden ist. Wenn Sie so weit sind, starten Sie Ihren Pi und bereiten Sie sich auf das Programmieren vor.

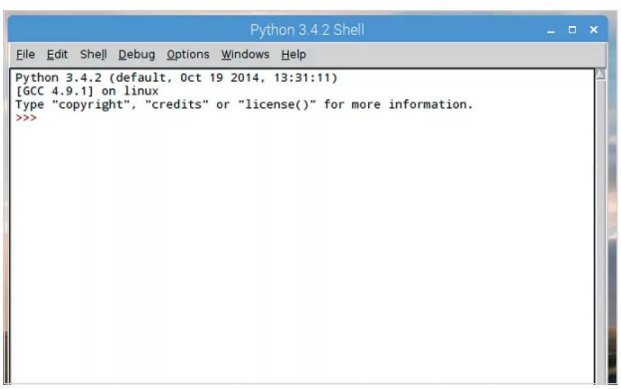
SCHRITT 1

Wenn der Raspbian-Desktop hochgefahren ist, klicken Sie auf die Menüschaltfläche, gefolgt von Entwicklung > Python 3 (IDLE). Dadurch wird die Python 3 Shell geöffnet. Windows- und Mac-Benutzer finden die Python 3 IDLE-Shell über das Windows-Startmenü bzw. den Finder.



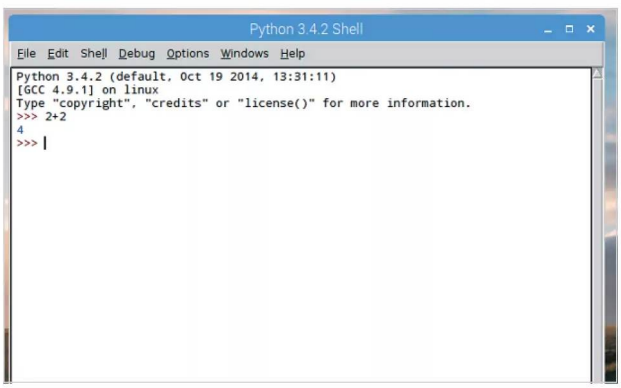
SCHRITT 2

Die Shell lässt Sie Code eingeben und zeigt die Antworten und Ausgaben des in Python programmierten Codes an. Dies ist eine Art Sandbox, in der Sie einfachen Code und Prozesse ausprobieren können.



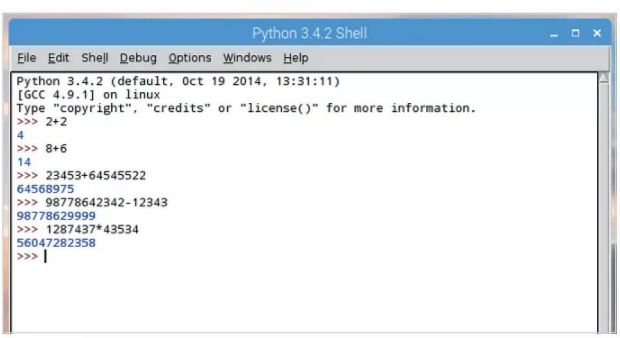
SCHRITT 3

Geben Sie als Beispiel 2+2 in die Shell ein. Nach dem Drücken der Eingabetaste zeigt die nächste Zeile die Antwort an: 4. Im Grunde hat Python den „Code“ übernommen und die entsprechende Ausgabe erzeugt.



SCHRITT 4

Die Python-Shell verhält sich ähnlich wie ein Taschenrechner, da Code im Grunde eine Reihe von mathematischen Interaktionen mit dem System darstellt. Integer, bei denen es sich um die unendliche Folge ganzer Zahlen handelt, können auf einfache Weise addiert, subtrahiert, multipliziert usw. werden.

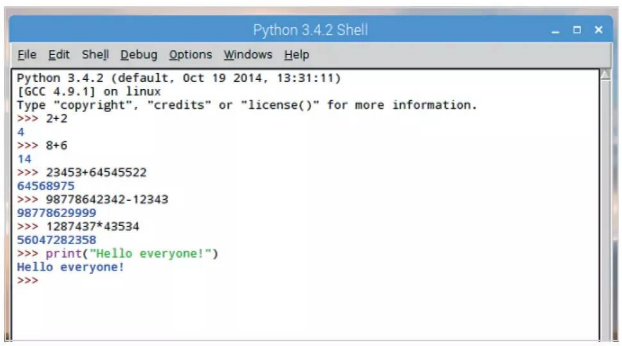


SCHRITT 5

Das ist zwar recht interessant, aber nicht besonders aufregend. Versuchen Sie stattdessen den folgenden Befehl:

```
print("Hello everyone!")
```

Geben Sie ihn wie zuvor in die IDLE ein.



SCHRITT 6

Das ist schon besser, zumal Sie soeben Ihren ersten Code geschrieben haben. Der print-Befehl gibt an den Bildschirm aus. In Python 3 werden die Klammern sowie die Anführungszeichen benötigt, um Inhalte auf dem Bildschirm auszugeben, in diesem Fall „Hello everyone!".

SCHRITT 7

Wahrscheinlich haben Sie auch die Farbcodierung in der Python IDLE bemerkt. Die Farben repräsentieren die verschiedenen Elemente des Python-Codes:

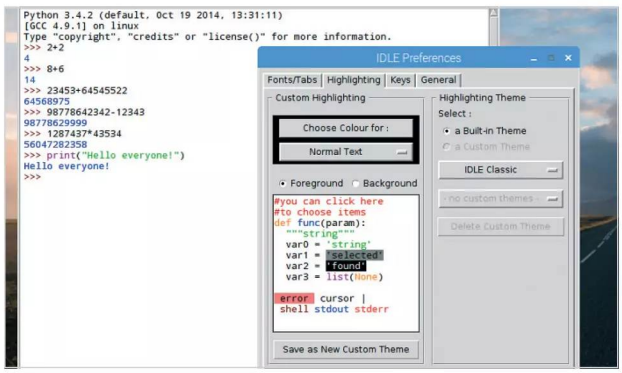
Schwarz – Daten und Variablen
Grün – Strings
Lila – Funktionen
Orange – Befehle
Blau – Benutzerfunktionen
Dunkelrot – Kommentare
Hellrot – Fehlermeldungen

IDLE Colour Coding

Colour	Use for	Examples
Black	Data & variables	`23.6 area`
Green	Strings	`"Hello World"`
Purple	Functions	`len() print()`
Orange	Commands	`if for else`
Blue	User functions	`get_area()`
Dark red	Comments	`#Remember VAT`
Light red	Error messages	`SyntaxError:`

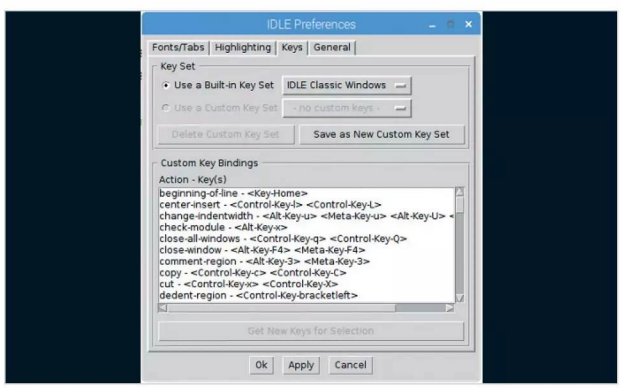
SCHRITT 8

Die Python IDLE ist eine konfigurierbare Umgebung. Wenn Ihnen die Darstellung der Farben nicht gefällt, können Sie sie jederzeit im Tab Highlights unter Options > Configure IDLE ändern. Wir raten jedoch davon ab, da Sie nicht dasselbe sehen werden wie in unseren Screenshots.



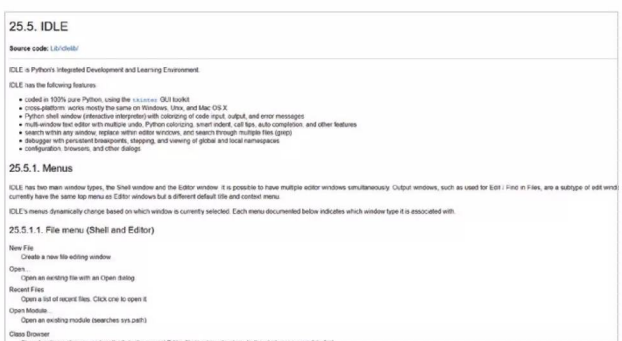
SCHRITT 9

Wie bei den meisten verfügbaren Programmen sind unabhängig vom Betriebssystem zahlreiche Tastenkombinationen erhältlich. Wir haben hier keinen Platz, um sie alle aufzulisten, aber im Tab Keys unter Options > Configure IDLE finden Sie eine Liste der aktuellen Kombinationen.



SCHRITT 10

Die Python IDLE ist eine Power-Interface, die in Python mit einem der vorhandenen GUI-Toolkits geschrieben wurde. Wenn Sie mehr über die Shell erfahren möchten, empfehlen wir Ihnen auf **www.docs.python.org/3/library/idle.html** zu gehen, wo viele der IDLE-Funktionen beschrieben werden.



Ihr erster Code

Im Grunde haben Sie bereits Ihren ersten Code mit der Funktion print("Hello everyone!") geschrieben. Wir werden das Ganze jedoch erweitern und uns genauer anschauen, wie Code eingegeben wird und auch andere Python-Beispiele ausprobieren.

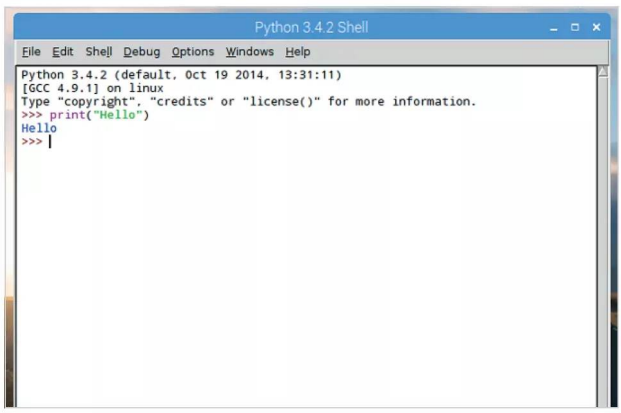
PYTHON AUSPROBIEREN

Bei den meisten Sprachen, ob Computer oder Mensch, geht es darum, in der richtigen Situation die richtigen Wörter anzuwenden. Diese Wörter müssen jedoch erst gelernt werden.

SCHRITT 1

Wenn Sie Python 3 IDLE geschlossen haben, öffnen Sie sie erneut in der von Ihnen bevorzugten Betriebssystemversion. Geben Sie in der Shell Folgendes ein:

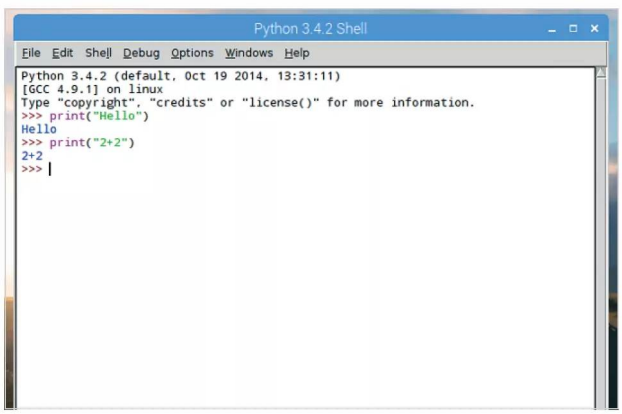
```
print("Hello")
```



SCHRITT 2

Genau wie vorhergesagt erscheint das Wort Hello in der Shell als blauer Text, der die Ausgabe eines Strings anzeigt. Dies ist ziemlich einfach und muss nicht großartig erklärt werden. Probieren Sie nun Folgendes aus:

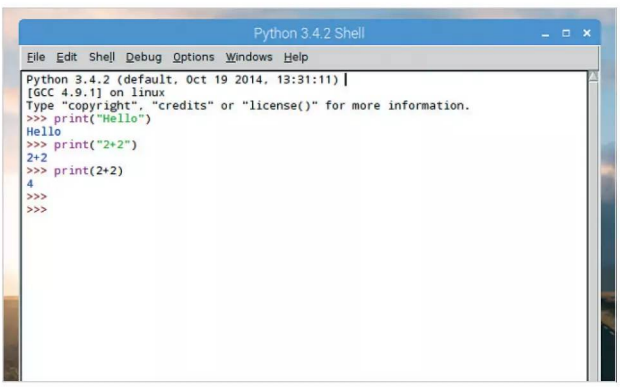
```
print("2+2")
```



SCHRITT 3

Wie Sie sehen wird anstelle der Zahl 4 2+2 auf dem Bildschirm ausgegeben. Die Anführungs-zeichen definieren, was in der IDLE Shell ausgegeben wird; um die Summe von 2+2 zu drucken, müssen Sie die Anführungszeichen entfernen:

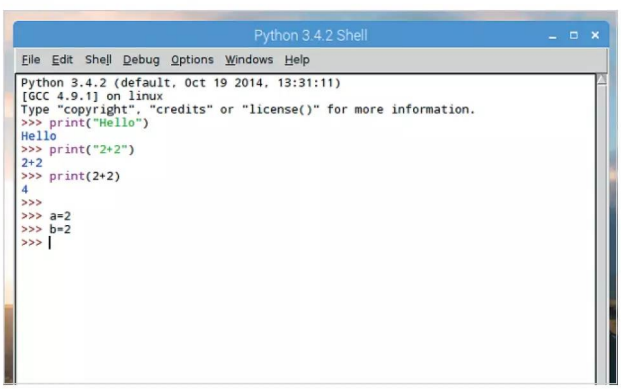
```
print(2+2)
```



SCHRITT 4

Sie können auf diese Weise fortfahren und 2+2, 464+2343 usw. in der Shell ausgeben. Ein einfacherer Weg ist die Verwendung einer Variablen, auf die wir später noch genauer eingehen werden. Geben Sie Folgendes ein:

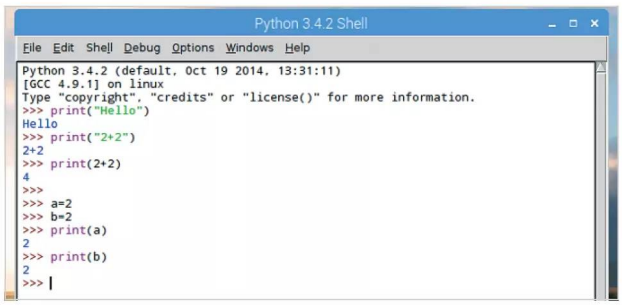
```
a=2
b=2
```



SCHRITT 5

Hier haben Sie den Buchstaben a und b zwei Werte zugeteilt: 2 und 2. Dies sind nun Variablen, die von Python so lange ausgegeben, addiert, subtrahiert, geteilt usw. werden können, wie die Zahlen gleich bleiben gleich. Probieren Sie Folgendes aus:

```
print(a)
print(b)
```

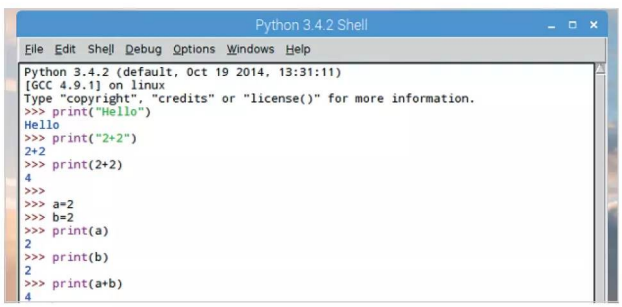


SCHRITT 6

Die Ausgabe des letzten Schritts zeigt die aktuellen Werte von a und b einzeln an, da Sie aufgrund Ihrer Eingabe getrennt ausgegeben werden. Wenn Sie sie addieren möchten, geben Sie Folgendes ein:

```
print(a+b)
```

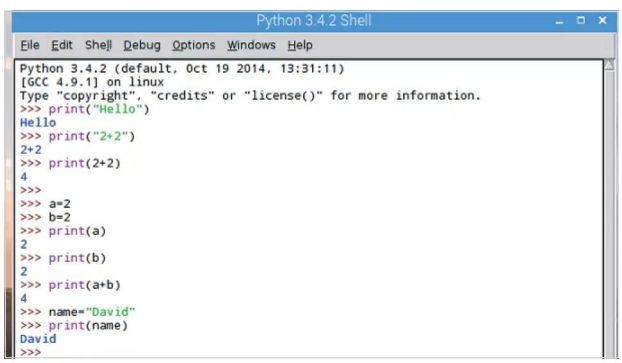
Dieser Code addiert die Werte von a und b und gibt das Ergebnis wieder.



SCHRITT 7

Sie können verschiedene Arten von Variablen mit dem print-Befehl ausprobieren. Sie könnten Variablen z. B. den Namen einer Person zuweisen:

```
name=”David”
print(name)
```

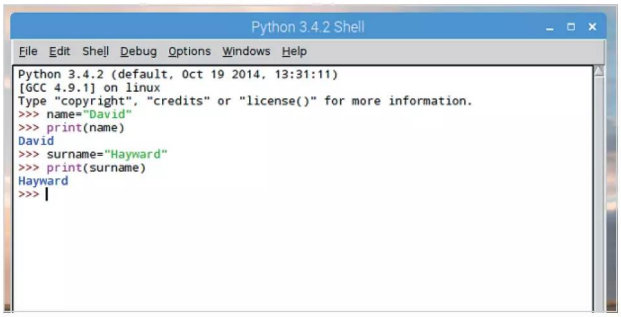


SCHRITT 8

Wir fügen nun einen Nachnamen ein:

```
surname=”Hayward”
print(surname)
```

Sie haben nun zwei Variablen, eine für den Vornamen und eine für den Nachnamen. Beide können unabhängig voneinander ausgegeben werden.

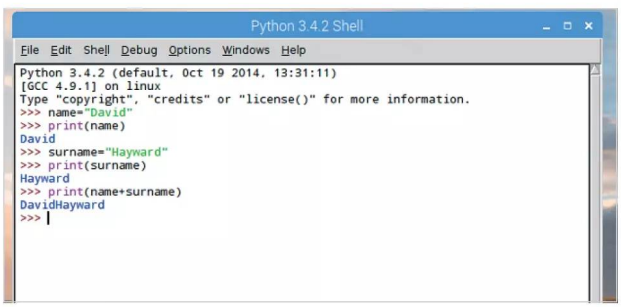


SCHRITT 9

Würden wir die gleiche Eingabe mit dem Pluszeichen anwenden, würde der Name in der Ausgabe nicht korrekt angezeigt werden. Probieren Sie es aus:

```
print(name+surname)
```

Sie müssen zwischen beiden einen Leerschritt einsetzen und sie dadurch anstelle von zwei mathematischen als zwei separate Werte definieren.



SCHRITT 10

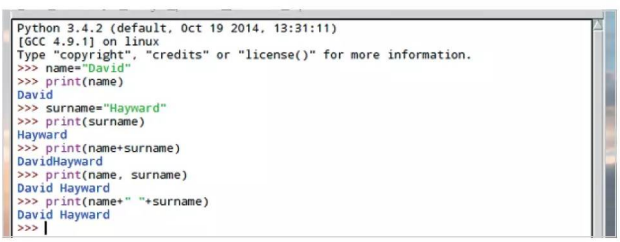
In Python 3 können Sie mithilfe eines Kommas die Variablen trennen:

```
print(name, surname)
```

Alternativ können Sie den Leerschritt selbst hinzufügen:

```
print(name+” “+surname)
```

Wie Sie sehen sieht es mit dem Komma jedoch übersichtlicher aus. Herzlichen Glückwunsch, Sie haben soeben Ihre ersten Schritte in die große Welt von Python gemacht.



Code speichern und ausführen

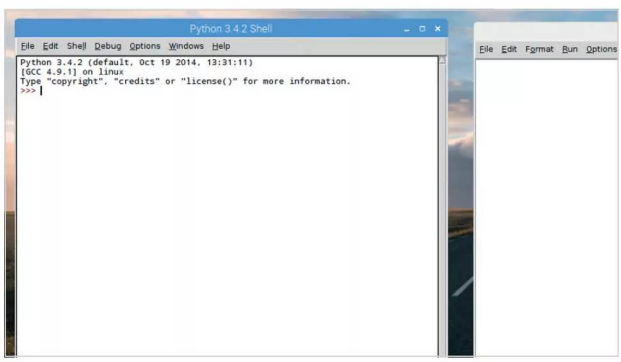
Während die IDLE-Shell für kurze Code-Abschnitte gut geeignet ist, ist sie für längere Programmlisten nicht gedacht. In diesem Abschnitt stellen wir Ihnen den IDLE Editor vor, mit dem Sie ab sofort arbeiten werden.

CODE BEARBEITEN

Sie werden letztendlich einen Punkt erreichen, an dem die Eingabe einzelner Codezeilen in die Shell nicht mehr ausreicht. Mit dem IDLE-Editor können Sie Ihren Python-Code speichern und ausführen.

SCHRITT 1

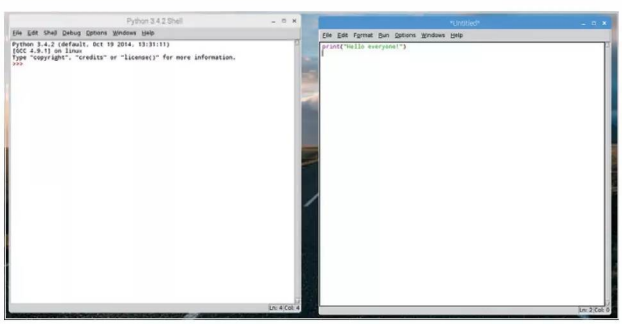
Öffnen Sie zuerst die Python IDLE Shell und klicken Sie auf File > New File. Dadurch wird ein neues Fenster namens Untitled geöffnet. Dies ist der Python IDLE Editor, in dem Sie den Code eingeben können, der für die Erstellung Ihrer zukünftigen Programme benötigt wird.



SCHRITT 2

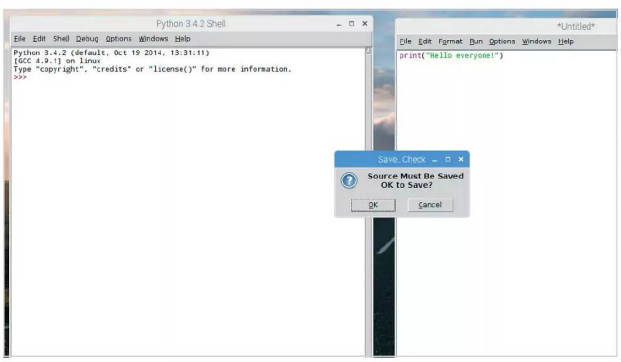
Der IDLE-Editor ist im Grunde ein einfacher Texteditor mit Python-Funktionen, Farbcodierung usw., ähnlich wie Sublime. Code wird auf die gleiche Weise wie in der Shell eingegeben. Geben Sie das folgende Beispiel aus dem vorherigen Tutorial ein:

```
print(“Hello everyone!”)
```



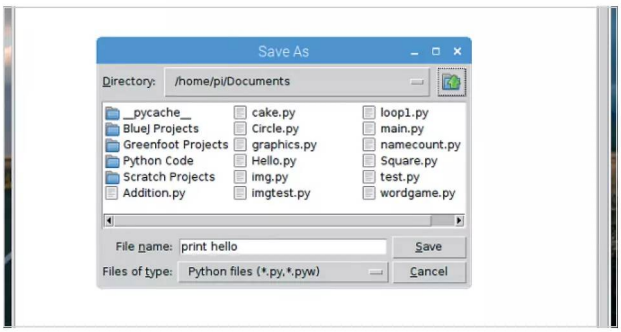
SCHRITT 3

Wie Sie sehen wird im IDLE-Editor die gleiche Farbcodierung wie in der Shell verwendet, wodurch besser zu verstehen ist, was mit dem Code passiert. Zur Ausführung des Codes müssen Sie ihn jedoch zuerst speichern. Drücken Sie F5, um das kleine Speichern-Fenster aufzurufen.



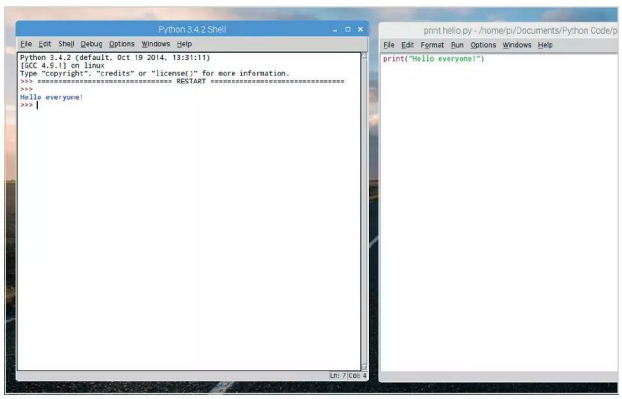
SCHRITT 4

Klicken Sie im Speichern-Fenster auf OK und wählen Sie einen Speicherort, in dem Sie alle Ihre Python-Codes speichern. Der Speicherort kann ein spezieller Ordner namens Python sein, Sie können Ihre Codes aber auch an einem beliebigen anderen Ort speichern. Sie sollten Ihr Laufwerk jedoch übersichtlich halten, um die Arbeit damit zu erleichtern.



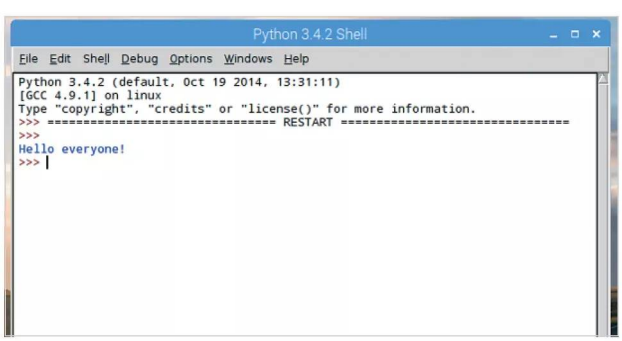
SCHRITT 5

Wir speichern den Code unter print hello und klicken zum Speichern auf „Save“. Sobald der Python-Code gespeichert ist, wird er ausgeführt und in der IDLE-Shell ausgegeben; in diesem Fall sind es die Worte „Hello everyone!”.



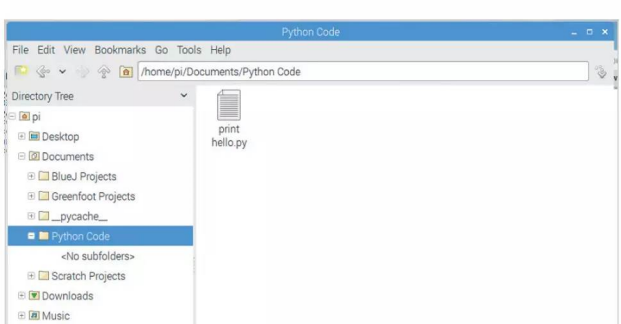
SCHRITT 6

Auf diese Weise wird der Großteil Ihres Python-Codes ausgeführt. Sie geben ihn in den Editor ein, drücken F5, speichern den Code und schauen sich die Ausgabe in der Shell an. Manchmal unterscheiden sich die Vorgänge, je nachdem, ob Sie ein separates Fenster angefordert haben, aber im Wesentlichen läuft der Prozess auf diese Weise ab. Sofern nicht anders angegeben, werden wir in dieser Ausgabe diesen Prozess befolgen.



SCHRITT 7

Wenn Sie den Speicherort des gespeicherten Python-Codes öffnen, sehen Sie, dass er mit der Erweiterung .py endet. Dies ist der Standard-Python-Dateiname. Jeder erstellte Code wird xyz.py lauten und auch jeder Code, den Sie von den vielen Python-Ressource-Webseiten herunterladen, endet in .py. Stellen Sie lediglich sicher, dass der Code für Python 3 geschrieben wurde.

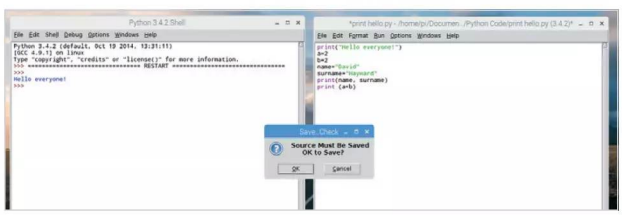


SCHRITT 8

Wir werden den Code erweitern und einige Beispiele aus dem vorherigen Tutorial eingeben:

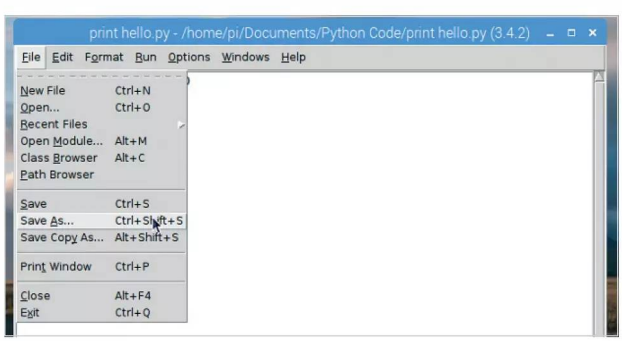
```
a=2
b=2
name=”David”
surname=”Hayward”
print(name, surname)
print (a+b)
```

Wenn Sie nun F5 drücken, werden Sie aufgefordert, die Datei erneut zu speichern, da sie geändert wurde.



SCHRITT 9

Wenn Sie auf OK klicken, wird die Datei mit den neuen Code-Einträgen überschrieben und ausgeführt, wobei die Ausgabe wieder in der Shell erscheint. Bei wenigen Zeilen ist das Überschreiben kein Problem, bei der Bearbeitung größerer Dateien kann es jedoch schwieriger werden. Gehen Sie in diesem Fall im Editor zu File > Save As, um eine Sicherungskopie zu erstellen.

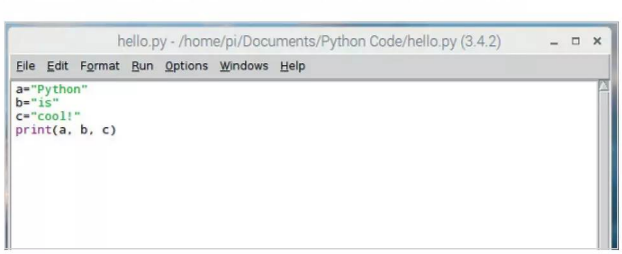


SCHRITT 10

Erstellen Sie nun eine neue Datei. Schließen Sie den Editor und öffnen Sie eine neue Datei (in der Shell über File > New File). Geben Sie Folgendes ein und speichern Sie es als hello.py:

```
a=”Python”
b=”is”
c=”cool!”
print(a, b, c)
```

Sie werden diesen Code im nächsten Tutorial verwenden.



Code über die Befehlszeile ausführen

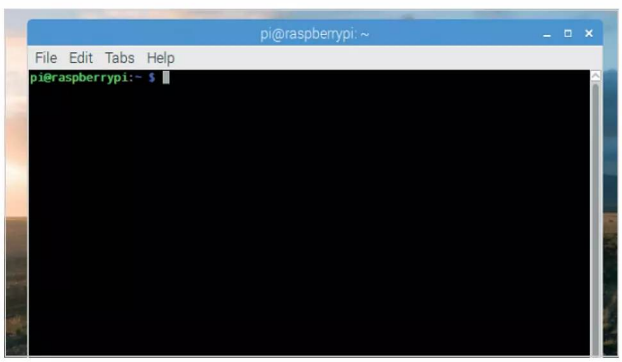
Obwohl wir in diesem Heft in der GUI IDLE arbeiten, lohnt es sich, sich auch die Anwendung von Pythons Befehlszeile anzusehen. Wie wir bereits wissen, gibt es eine Python-Version der Befehlszeile, die auch zum Ausführen von Code verwendet wird.

DER CODE IN DER BEFEHLSZEILE

Mithilfe des in der vorherigen Anleitung erstellten Codes, den wir hello.py genannt haben, schauen wir uns an, wie Code ausgeführt wird, der in der GUI in der Befehlszeile erstellt wurde.

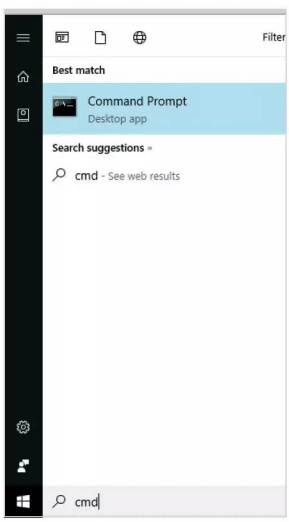
SCHRITT 1

Python bietet unter Linux zwei Möglichkeiten, Code über die Befehlszeile auszuführen. Eine ist über Python 2, während die andere die Python 3-Bibliotheken usw. verwendet. Zuerst müssen Sie jedoch die Befehlszeile bzw. das Terminal auf Ihrem Betriebssystem öffnen.



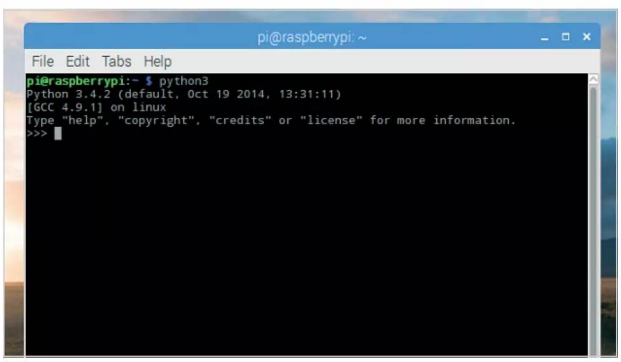
SCHRITT 2

Wie zuvor verwenden wir auch hier einen Raspberry Pi. Windows-Benutzer müssen im Suchfeld neben dem Startmenü nach CMD bzw. der Eingabeaufforderung suchen und macOS-Benutzer finden die Befehlszeile unter Gehe zu > Dienstprogramme > Terminal.



SCHRITT 3

Sie sind nun in der Befehlszeile und können Python starten. Für Python 3 müssen Sie **python3** eingeben und die Eingabetaste drücken. Dadurch gelangen Sie in die Befehlszeilenversion der Shell mit den drei nach rechts weisenden Pfeilen, die als Cursor dienen (>>>).

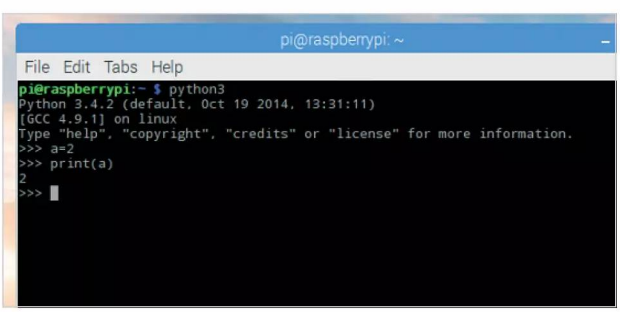


SCHRITT 4

Von hier aus können Sie den Code eingeben, den wir uns zuvor bereits angeschaut haben, wie zum Beispiel:

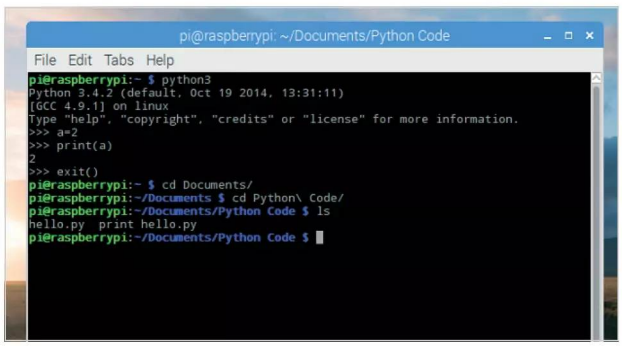
```
a=2
print(a)
```

Wie Sie sehen, funktioniert er auf die gleiche Weise.



SCHRITT 5

Geben Sie nun **exit()** ein, um die Python-Befehlszeile zu verlassen und zur Eingabeaufforderung zurückzukehren. Geben Sie den Ordner ein, in dem Sie den Code aus dem vorherigen Lernprogramm gespeichert haben, und listen Sie die verfügbaren Dateien auf. Die hello.py-Datei sollte hoffentlich zu sehen sein.



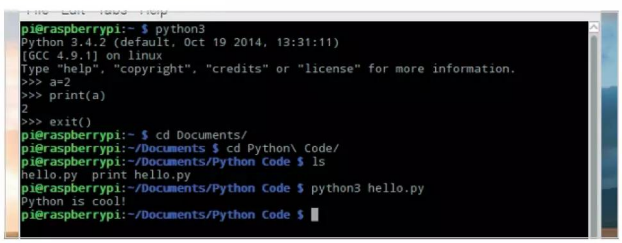
SCHRITT 6

Geben Sie innerhalb des Ordners, der den Code enthält, den Sie ausführen möchten, Folgendes in die Befehlszeile ein:

```
python3 hello.py
```

Dadurch wird der Code ausgeführt, den wir zuvor erstellt haben:

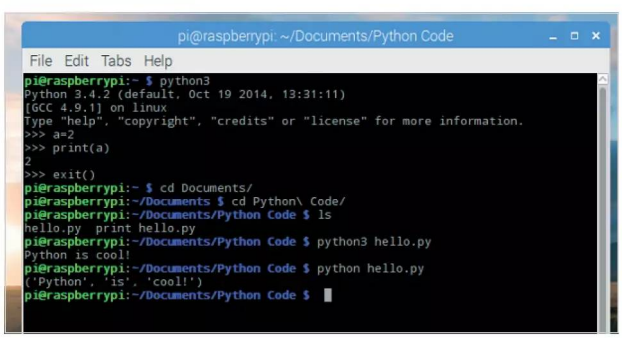
```
a=”Python”
b=”is”
c=”cool!”
print(a, b, c)
```



SCHRITT 7

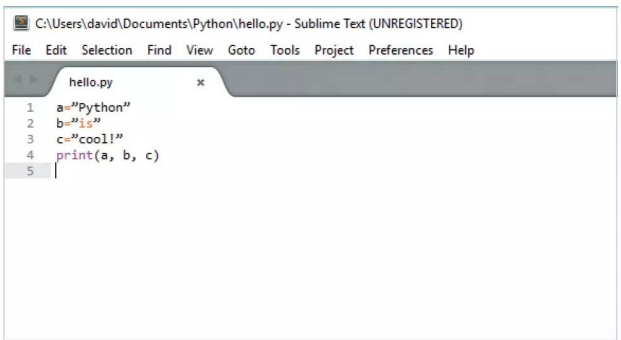
Da es sich um Python 3-Code handelt, können Sie die Syntax und das Layout von Python 3 natürlich nur verwenden, wenn Sie den python3-Befehl verwenden. Wenn Sie möchten, versuchen Sie dasselbe mit Python 2, indem Sie Folgendes eingeben:

```
python hello.py
```



SCHRITT 8

Das Ergebnis der Ausführung des Python 3-Codes über die Python 2-Befehlszeile ist recht offensichtlich. Obwohl es aufgrund der Unterschiede zwischen der Art und Weise, wie Python 3 den print-Befehl über Python 2 handhabt, nicht zu einem Fehler kommt, ist das Ergebnis nicht, wie wir es erwartet haben. Öffnen Sie die hello.py-Datei in Sublime.



SCHRITT 9

Da Sublime Text für den Raspberry Pi nicht verfügbar ist, verlassen Sie den Pi vorübergehend und verwenden Sie Sublime; dies zeigt, dass Sie nicht unbedingt Python IDLE verwenden müssen. Ändern Sie die geöffnete Datei hello.py wie folgt:

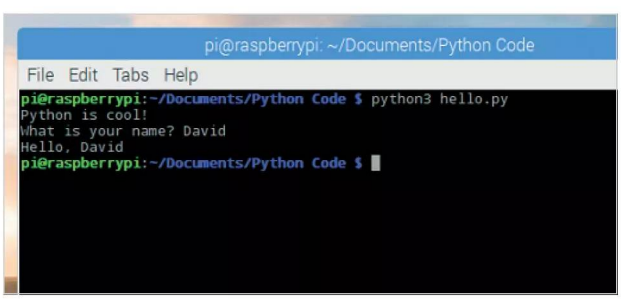
```
name=input(“What is your name? “)
print(“Hello,”, name)
```

SCHRITT 10

Speichern Sie die Datei hello.py und kehren Sie zur Befehlszeile zurück. Führen Sie nun den neu gespeicherten Code mit dem folgenden Befehl aus:

```
python3 hello.py
```

Das Ergebnis ist die Originalaussage „Python is cool!“ zusammen mit dem hinzugefügten Eingabebefehl, der Sie nach Ihrem Namen fragt, der im Befehlsfenster angezeigt wird.



Zahlen und Ausdrücke

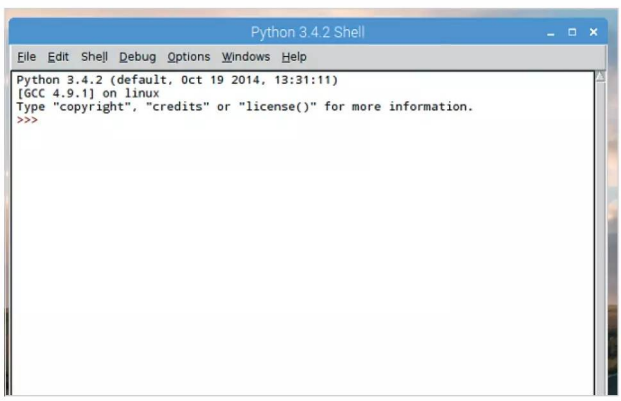
Wir haben bereits einige einfache mathematische Ausdrücke in Python gesehen wie z. B. einfache Additionen. Wir gehen nun etwas näher darauf ein, was für ein leistungsstarker Rechner Python ist. Sie können innerhalb der IDLE-Shell oder im Editor arbeiten.

MATHE, MATHE, MATHE

Mit den mathematischen Fähigkeiten von Python lassen sich wirklich beeindruckende Ergebnisse erzielen. Wie bei den meisten, wenn nicht sogar allen Programmiersprachen, ist Mathematik die treibende Kraft hinter dem Code.

SCHRITT 1

Öffnen Sie die GUI-Version von Python 3. Wie erwähnt, können Sie entweder die Shell oder den Editor verwenden. Momentan werden wir die Shell nur dazu verwenden, um unseren Mathe-Muskel aufzuwärmen, von dem wir glauben, dass er eine kleine Drüse ist, die sich im hinteren Teil des Gehirns befindet.

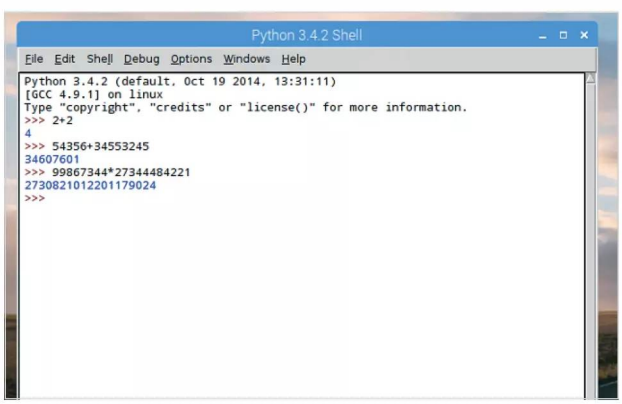


SCHRITT 2

Geben Sie Folgendes in die Shell ein:

```
2+2
54356+34553245
99867344*27344484221
```

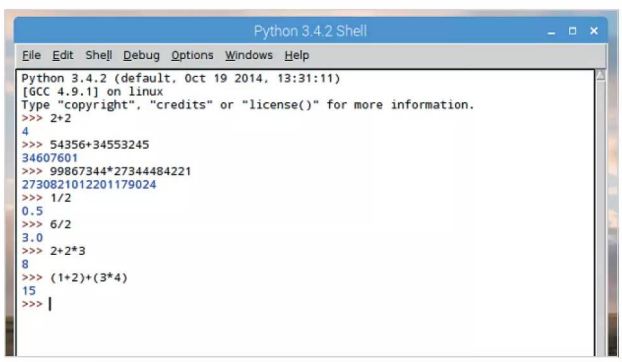
Wie Sie sehen, kann Python recht hohe Werte bearbeiten.



SCHRITT 3

Sie können alle üblichen mathematischen Rechenarten verwenden: Teilen, Multiplizieren, Klammern usw. Probieren Sie ein paar Übungen aus, z. B.:

```
1/2
6/2
2+2*3
(1+2)+(3*4)
```

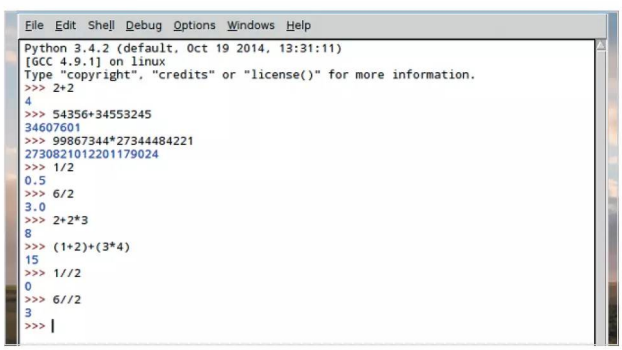


SCHRITT 4

Beim Dividieren werden Dezimalzahlen erzeugt, die in Python Floats bzw. Gleitkommaarithmetik heißen. Wenn Sie jedoch eine Ganzzahl benötigen, können Sie einen doppelten Schrägstrich verwenden:

```
1//2
6//2
```

und so weiter.



SCHRITT 5

Sie können auch mithilfe einer Operation den Rest der Division sehen. Die Eingabe

```
10/3
```

ergibt 3,333333333, also 3,3 periodisch. Wenn Sie nun

```
10%3
```

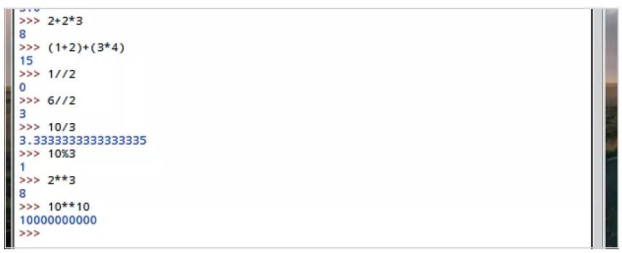
eingeben, erhalten Sie 1, was der Restmenge entspricht, die durch das Teilen von 10 durch 3 übrig geblieben ist.

SCHRITT 6

Als Nächstes haben wir den Potenz-Operator. Um die Potenz zu ermitteln, können Sie das doppelte Multiplikationssymbol oder den doppelten Stern auf Ihrer Tastatur verwenden:

```
2**3
10**10
```

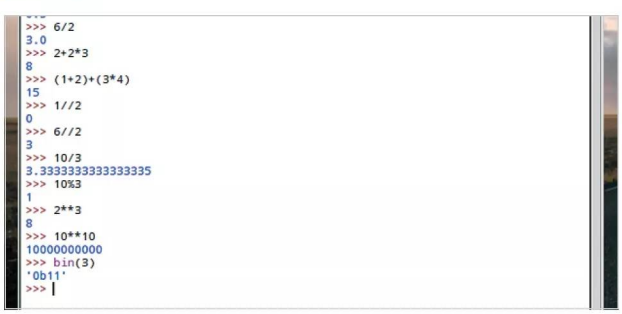
Im Grunde bedeutet dies 2 x 2 x 2, aber die Grundlagen der mathematischen Operatoren sind Ihnen sicherlich bereits bekannt. Auf diese Weise würden Sie die Aufgabe in Python ausarbeiten.



SCHRITT 7

Python hört bei Zahlen und Ausdrücken jedoch nicht auf und hat zahlreiche eingebaute Funktionen, um Zahlenreihen, absolute Werte, komplexe Zahlen und eine Vielzahl von mathematischen Ausdrücken und pythagoräischen Zungenbrechern auszuarbeiten. Um z. B. eine Zahl in eine binäre Zahl zu konvertieren, geben Sie Folgendes ein:

```
bin(3)
```

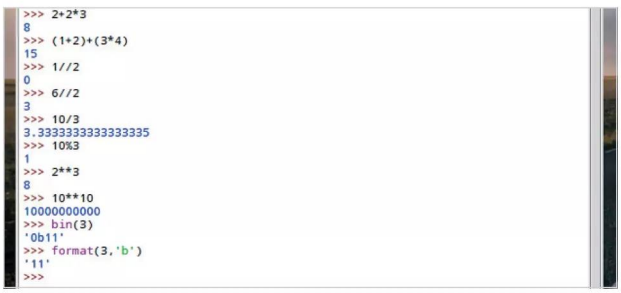


SCHRITT 8

Dies wird als 0b11 angezeigt, die Ganzzahl wird in eine Binärzahl umgewandelt und erhält das Präfix 0b. Mit der folgenden Eingabe können Sie das Präfix entfernen:

```
format(3, 'b')
```

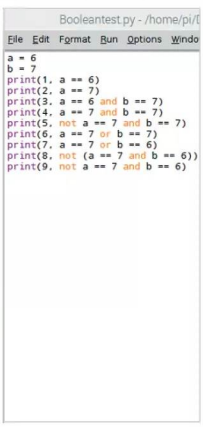
Der format-Befehl konvertiert einen Wert, in diesem Fall die Zahl 3, in eine formatierte Darstellung, die von der Formatspezifikation - dem „b" - gesteuert wird.



SCHRITT 9

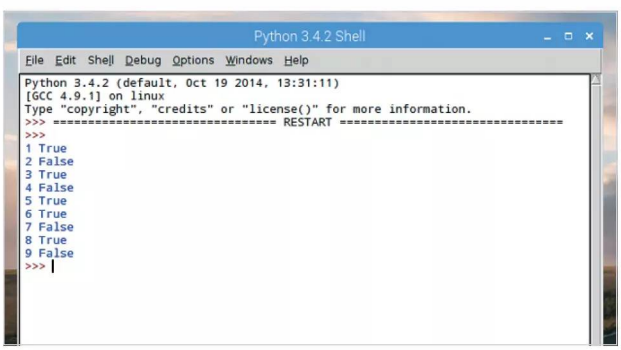
Ein boolescher Ausdruck ist eine logische Anweisung, die entweder wahr oder falsch ist. Wir können damit Daten vergleichen und testen, ob sie gleich, kleiner oder größer sind. Probieren Sie es in einer neuen Datei (New File) aus:

```
a = 6
b = 7
print(1, a == 6)
print(2, a == 7)
print(3, a == 6 and b == 7)
print(4, a == 7 and b == 7)
print(5, not a == 7 and b == 7)
print(6, a == 7 or b == 7)
print(7, a == 7 or b == 6)
print(8, not (a == 7 and b == 6))
print(9, not a == 7 and b == 6)
```



SCHRITT 10

Führen Sie den Code in Schritt 9 aus, um abhängig vom Ergebnis der beiden definierenden Werte – 6 und 7 – eine Reihe von Wahr- oder Falsch-Anweisungen zu sehen. Dies ist eine Erweiterung dessen, mit dem Sie sich zuvor befasst haben und ein wichtiger Teil der Programmierung.



Kommentare

Beim Schreiben eines Codes haben Sie die Funktion einer jeden Variablen sowie des gesamten Programms usw. in Ihrem Kopf. Ein anderer Programmierer könnte dem Code Zeile für Zeile folgen, aber mit der Zeit kann es schwierig werden, ihn nachzuvollziehen.

#KOMMENTARE!

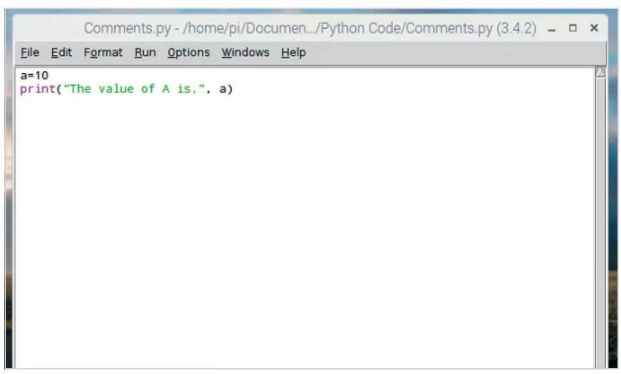
Um ihren Code lesbar zu halten, kommentieren Programmierer bestimmte Abschnitte. Wenn eine Variable verwendet wird, fügt der Programmierer in einem Kommentar z. B. hinzu, was diese bezwecken soll. Es ist ein bewährtes Verfahren.

SCHRITT 1

Beginnen Sie mit dem Erstellen einer neuen Datei im IDLE-Editor (File > New > New File) und erstellen Sie eine einfache Variable und einen print-Befehl:

```
a=10
print("The value of A is,", a)
```

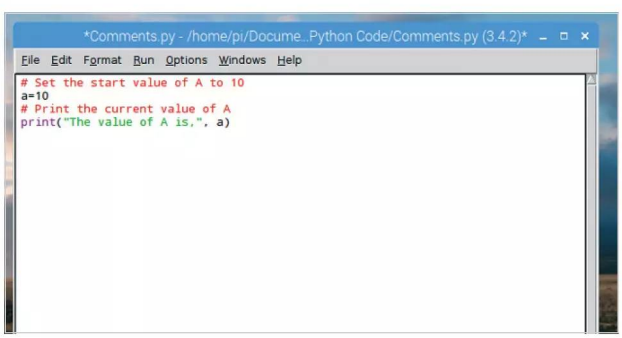
Speichern Sie die Datei und führen Sie den Code aus.



SCHRITT 2

Wenn Sie den Code ausführen, wird wie erwartet die Zeile „The value of A is 10“ im IDLE-Shell-Fenster ausgegeben. Fügen Sie nun einige der Kommentare hinzu, die Sie normalerweise im Code sehen würden:

```
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
```



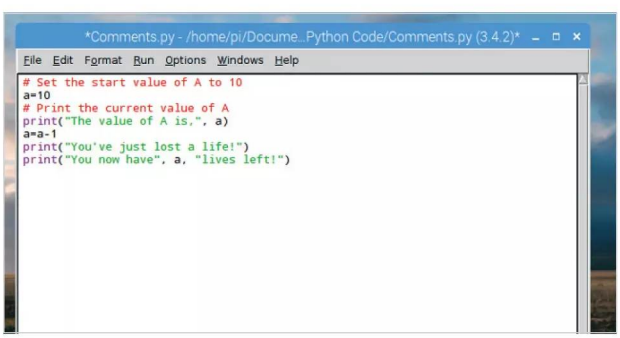
SCHRITT 3

Speichern Sie den Code erneut und führen Sie ihn aus. Sie werden sehen, dass die Ausgabe in der IDLE-Shell trotz der hinzugefügten zusätzlichen Zeilen immer noch dieselbe ist. Einfach ausgedrückt steht das Rautenzeichen (#) für eine Textzeile, die der Programmierer einfügen kann, um andere darüber zu informieren, was passiert, ohne dass der Benutzer dies bemerkt.

SCHRITT 4

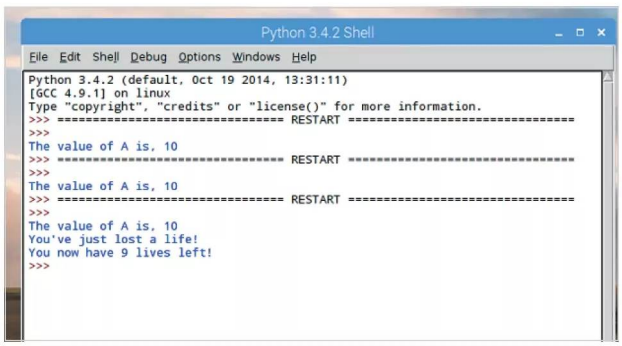
Nehmen wir an, dass die von uns erstellte Variable A die Anzahl der Leben in einem Spiel ist. Jedes Mal, wenn der Spieler stirbt, wird der Wert um 1 verringert. Der Programmierer könnte ein Programm wie das Folgende eingeben:

```
a=a-1
print("You've just lost a life!")
print("You now have", a, "lives left!")
```



SCHRITT 5

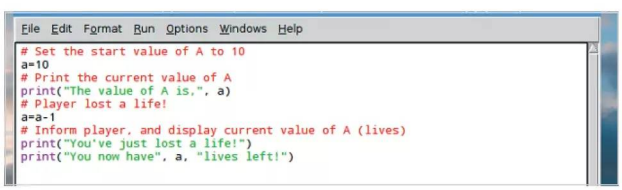
Während wir wissen, dass die Variable A für Leben steht und der Spieler gerade ein Leben verloren hat, weiß ein zufälliger Betrachter oder jemand, der den Code überprüft, das eventuell nicht. Um zu verdeutlichen, wie praktisch Kommentare sind, stellen Sie sich einfach vor, der Code wäre zwanzigtausend Zeilen lang anstatt nur sieben.



SCHRITT 6

Im Wesentlichen könnte der neue Code mit den Kommentaren wie folgt aussehen:

```
# Set the start value of A to 10
a=10
# Print the current value of A
print(“The value of A is,”, a)
# Player lost a life!
a=a-1
# Inform player, and display current value of A
(lives)
print(“You’ve just lost a life!”)
print(“You now have”, a, “lives left!”)
```



SCHRITT 7

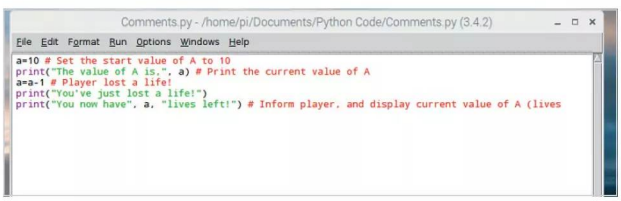
Kommentare können auf verschiedene Weise verwendet werden. Zum Beispiel sind Blockkommentare ein großer Textabschnitt, der detailliert beschreibt, was im Code vor sich geht, um z. B. dem Codeleser mitzuteilen, welche Variablen verwendet werden:

```
# This is the best game ever, and has been
developed by a crack squad of Python experts
# who haven’t slept or washed in weeks. Despite
being very smelly, the code at least
# works really well.
```

SCHRITT 8

Inline-Kommentare sind Kommentare, die einem Codeabschnitt folgen. Wir nehmen unsere vorherigen Beispiele und anstatt den Code in eine separate Zeile einzufügen, könnten wir Folgendes eingeben:

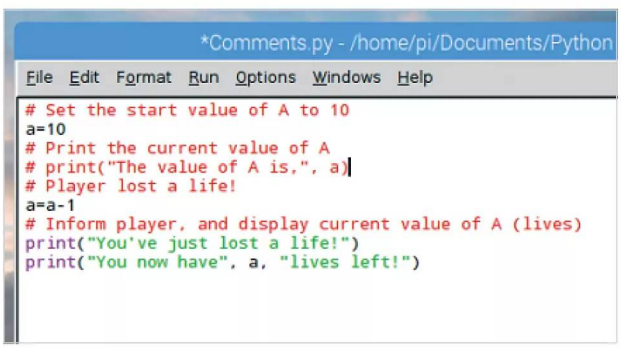
```
a=10 # Set the start value of A to 10
print(“The value of A is,”, a) # Print the current
value of A
a=a-1 # Player lost a life!
print(“You’ve just lost a life!”)
print(“You now have”, a, “lives left!”) # Inform
player, and display current value of A (lives)
```



SCHRITT 9

Mit dem Kommentar, dem Rautensymbol, können auch Codeabschnitte kommentiert werden, die im Programm nicht ausgeführt werden sollen. Wenn Sie z. B. den ersten print-Befehl entfernen möchten, geben Sie Folgendes ein:

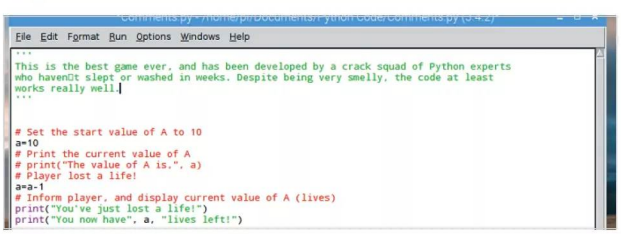
```
# print(“The value of A is,”, a)
```



SCHRITT 10

Sie können mit drei einfachen Anführungszeichen einen Blockkommentar oder einen mehrzeiligen Abschnitt auskommentieren. Platzieren Sie sie vor und hinter den auszukommentierenden Bereich, damit sie funktionieren:

```
‘’’
This is the best game ever, and has been developed
by a crack squad of Python experts who haven’t
slept or washed in weeks. Despite being very
smelly, the code at least works really well.
’’’
```



Arbeiten mit Variablen

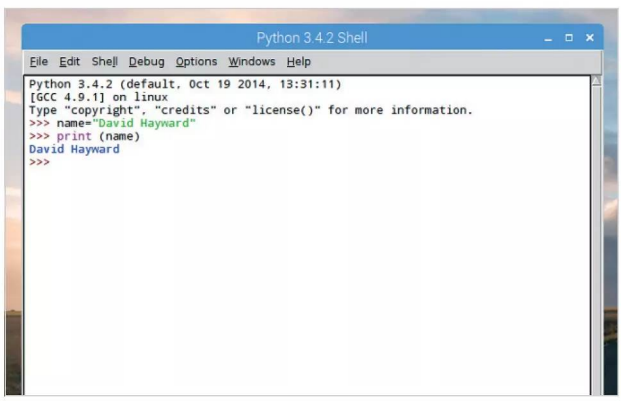
Wir haben bereits einige Beispiele für Variablen in unserem Python-Code gesehen, aber es lohnt sich immer, sich genauer anzuschauen, wie sie funktionieren und wie Python bestimmte Werte erstellt und diese einer Variablen zuweist.

VERSCHIEDENE VARIABLEN

Sie werden in diesem Tutorial mit der Python 3 IDLE Shell arbeiten. Wenn Sie dies noch nicht getan haben, öffnen Sie Python 3 oder schließen Sie die vorherige IDLE-Shell, um alten Code zu löschen.

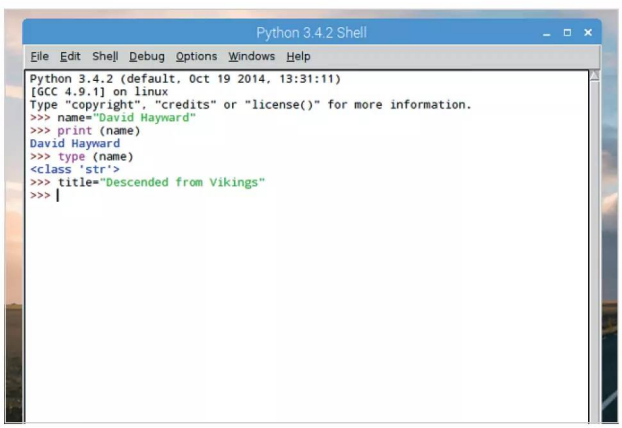
SCHRITT 1

In einigen Programmiersprachen müssen Sie zur Markierung eines Strings, d. h. einer aus mehreren Zeichen bestehenden Variablen (z. B. dem Namen einer Person), Dollarzeichen verwenden. In Python ist dies nicht notwendig. Geben Sie als Beispiel in der Shell `name=”David Hayward”` ein (bzw. Ihren eigenen Namen).



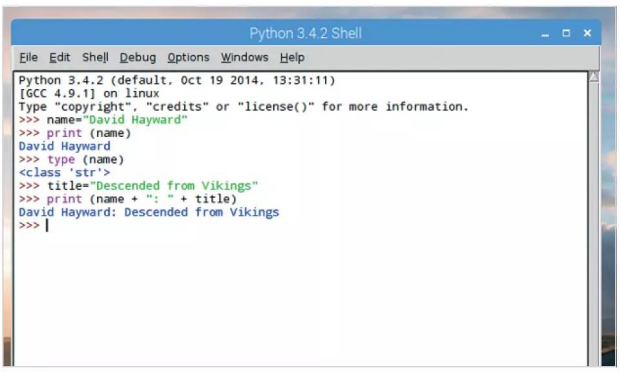
SCHRITT 2

Sie können den Typ der verwendeten Variablen mithilfe des **type ()**-Befehls überprüfen, wobei der Name der Variablen in die Klammern gesetzt wird. In unserem Beispiel wäre das: `type (name)`. Fügen Sie nun eine neue String-Variable ein: `title=”Descended from Vikings”`.



SCHRITT 3

Variablen können mit dem Pluszeichen zwischen den Variablennamen verkettet werden. In unserem Beispiel würde dies wie folgt aussehen: `print (name + “: “ + title)`. Im mittleren Teil zwischen den Anführungszeichen fügen wir einen Doppelpunkt und ein Leerzeichen hinzu, da Variablen ohne Leerzeichen miteinander verbunden werden und wir diese somit manuell einfügen müssen.



SCHRITT 4

Sie können Variablen auch innerhalb einer anderen Variablen kombinieren. Um z. B. die Namen- und Titelvariablen zu einer neuen Variablen zu kombinieren, geben wir Folgendes ein:

```
character=name + “: “ +
title
```

und die Ausgabe der neuen Variable:

```
print (character)
```

Zahlen werden als unterschiedliche Variablen gespeichert:

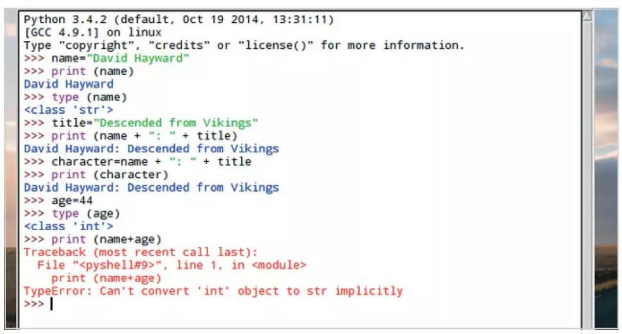
```
age=44
type (age)
```

Wie wir wissen, sind dies Ganzzahlen bzw. Integer.

SCHRITT 5

Sie können jedoch nicht Strings und Integer-Variablen im gleichen Befehl kombinieren. Sie müssen entweder den einen in den anderen umwandeln oder umgekehrt. Wenn Sie versuchen, beide zu kombinieren, erhalten Sie eine Fehlermeldung:

```
print (name + age)
```



```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name="David Hayward"
>>> print (name)
David Hayward
>>> type (name)
<class 'str'>
>>> title="Descended from Vikings"
>>> print (name + ": " + title)
David Hayward: Descended from Vikings
>>> character=name + ": " + title
>>> print (character)
David Hayward: Descended from Vikings
>>> age=44
>>> type (age)
<class 'int'>
>>> print (name+age)
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    print (name+age)
TypeError: Can't convert 'int' object to str implicitly
>>>
```

SCHRITT 6

Dieser Vorgang nennt sich Typumwandlung. Der Python-Code lautet:

```
print (character + " is " + str(age) + " years old.")
```

oder:

```
print (character, "is", age, "years old.")
```

Beachten Sie auch hier, dass im letzten Beispiel die Leerzeichen zwischen den Wörtern in den Anführungszeichen nicht benötigt werden, da die Kommas jedes Argument einzeln behandeln.

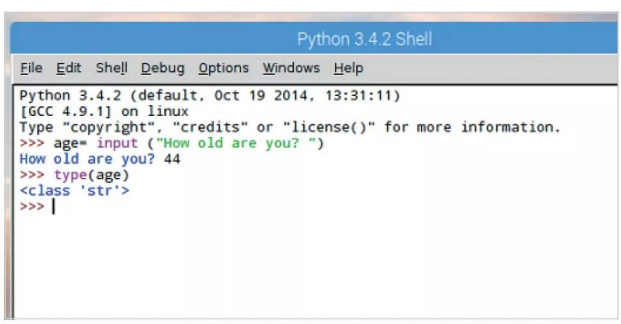
```
>>> print (name + age)
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    print (name + age)
TypeError: Can't convert 'int' object to str implicitly
>>> print (character + " is " + str(age) + " years old.")
David Hayward: Descended from Vikings is 44 years old.
>>> print (character, "is", age, "years old.")
David Hayward: Descended from Vikings is 44 years old.
>>>
>>>
```

SCHRITT 7

Ein weiteres Beispiel für die Typumwandlung ist die Anfrage nach Eingaben vom Benutzer (z. B. Eingabe des Namens). Geben Sie z. B. Folgendes ein:

```
age= input ("How old are you? ")
```

Alle mit dem input-Befehl gespeicherten Daten werden als String-Variable gespeichert.



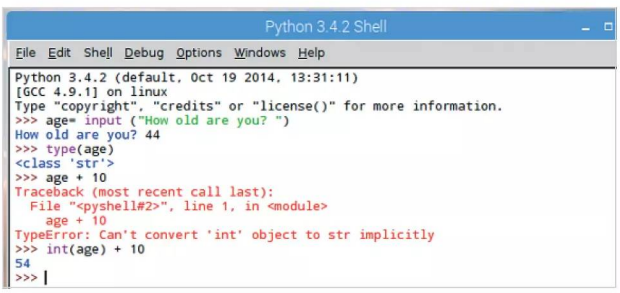
```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> age= input ("How old are you? ")
How old are you? 44
>>> type(age)
<class 'str'>
>>>
```

SCHRITT 8

Dies stellt ein kleines Problem dar, wenn Sie mit einer Zahl arbeiten möchten, die vom Benutzer eingegeben wurde, da „age + 10" aufgrund einer String-Variablen und einer Ganzzahl nicht funktioniert. Stattdessen müssen Sie Folgendes eingeben:

```
int(age) + 10
```

Dadurch wird der age-String in eine Ganzzahl umgewandelt.



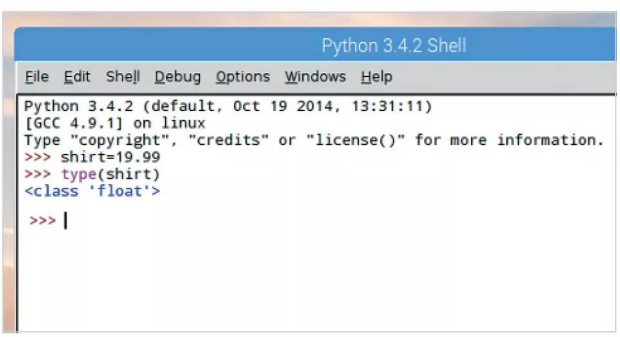
```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> age= input ("How old are you? ")
How old are you? 44
>>> type(age)
<class 'str'>
>>> age + 10
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    age + 10
TypeError: Can't convert 'int' object to str implicitly
>>> int(age) + 10
54
>>>
```

SCHRITT 9

Die Typumwandlung ist auch bei der Gleitkommaarithmetik wichtig: Zahlen, die eine Dezimalstelle haben. Geben Sie als Beispiel Folgendes ein:

```
shirt=19.99
```

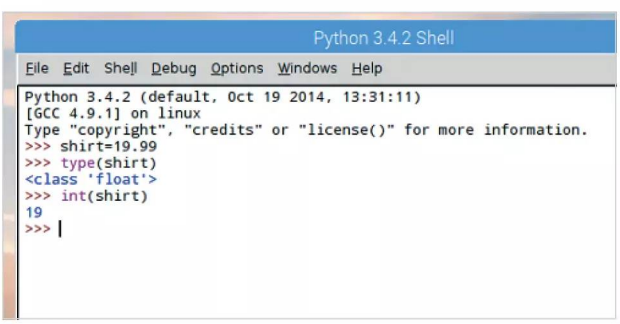
Geben Sie nun `type(shirt)` ein. Wie Sie sehen, hat Python die Zahl als „float" klassifiziert, da der Wert eine Dezimalzahl ist.



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> shirt=19.99
>>> type(shirt)
<class 'float'>
>>>
```

SCHRITT 10

Beim Kombinieren von Ganzzahlen und Gleitkommazahlen konvertiert Python normalerweise die ganze Zahl in einen Gleitkommawert; wird das Ganze jedoch umgekehrt ist zu beachten, dass Python nicht den exakten Wert zurückgibt. Bei der Konvertierung von Gleitkommazahlen in Ganzzahlen wird Python immer auf die nächste Ganzzahl abrunden; in unserem Fall erhalten wir 19 anstelle von 19.99.



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> shirt=19.99
>>> type(shirt)
<class 'float'>
>>> int(shirt)
19
>>>
```

Benutzereingaben

Wir haben in den vorherigen Beispielen einige einfache Benutzerinteraktionen mit dem Code gesehen. Wir werden uns daher nun ausschließlich darauf konzentrieren, wie Informationen vom Benutzer erhalten, gespeichert und präsentiert werden.

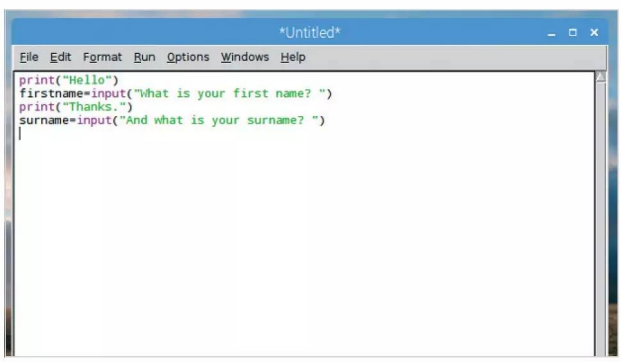
BENUTZERFREUNDLICH

Die Art der Eingabe, die Sie vom Benutzer wünschen, hängt stark von der Art des von Ihnen verfassten Programms ab. Zum Beispiel kann ein Spiel nach dem Namen eines Charakters fragen, während eine Datenbank nach persönlichen Details fragt.

SCHRITT 1

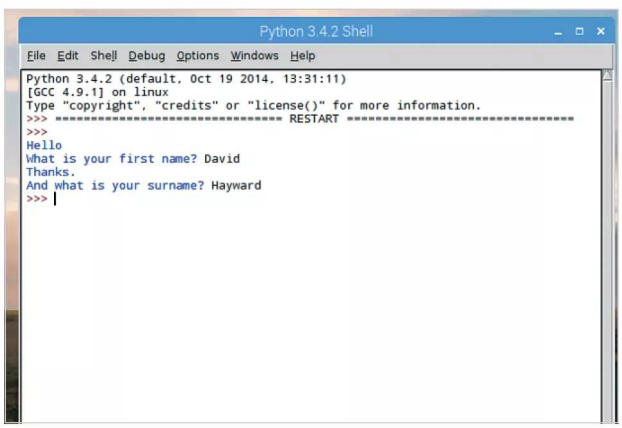
Öffnen Sie die Python 3 IDLE Shell und öffnen Sie eine neue Datei im Editor. Wir beginnen mit etwas sehr Einfachem:

```
print("Hello")
firstname=input("What is your first name? ")
print("Thanks.")
surname=input("And what is your surname? ")
```



SCHRITT 2

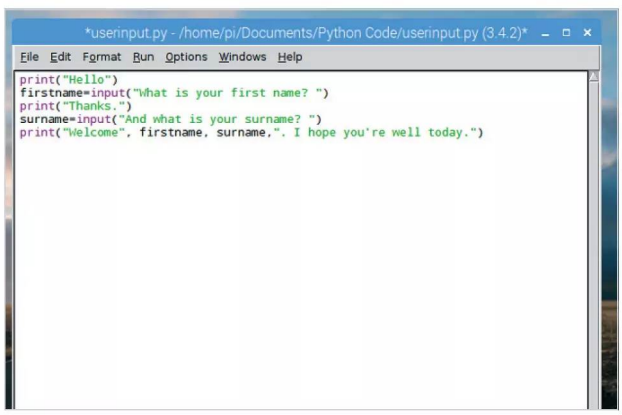
Speichern Sie den Code und führen ihn aus. Das Programm wird in der IDLE-Shell nach dem Vornamen fragen und diesen als Variablennamen speichern, gefolgt vom Nachnamen, der ebenfalls als eigene Variable (Nachname) gespeichert wird.



SCHRITT 3

Nachdem wir die Namen des Benutzers in Variablen gespeichert haben, können wir sie aufrufen, wann immer wir wollen:

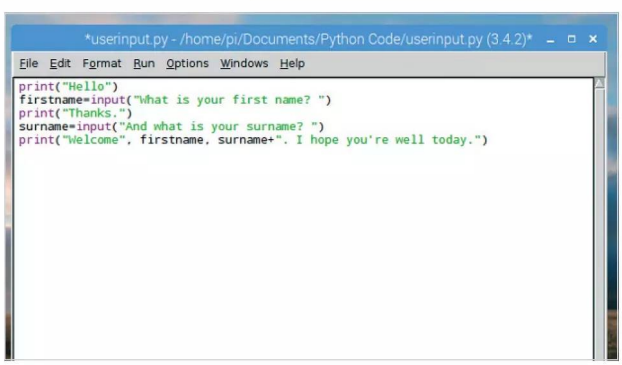
```
print("Welcome", firstname, surname, ". I hope you're well today.")
```



SCHRITT 4

Wenn Sie den Code ausführen, werden Sie sehen, dass es ein kleines Problem gibt, und zwar folgt dem Punkt hinter dem Nachnamen ein Leerzeichen. Um das zu eliminieren, können wir im Code anstelle des Kommas ein Pluszeichen hinzufügen:

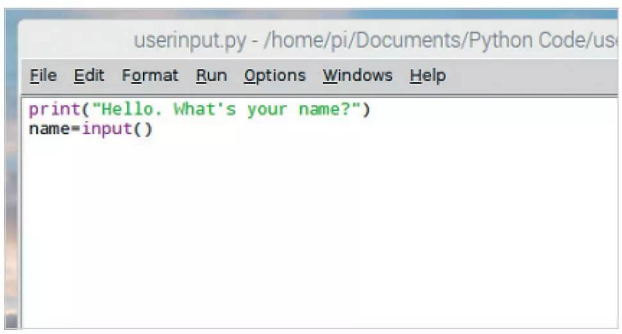
```
print("Welcome", firstname, surname+". I hope you're well today.")
```



SCHRITT 5

Sie müssen nicht immer zitierten Text in den Eingabebefehl einfügen. Sie können zum Beispiel den Benutzer nach seinem Namen fragen und die Eingabe in der folgende Zeile erscheinen lassen:

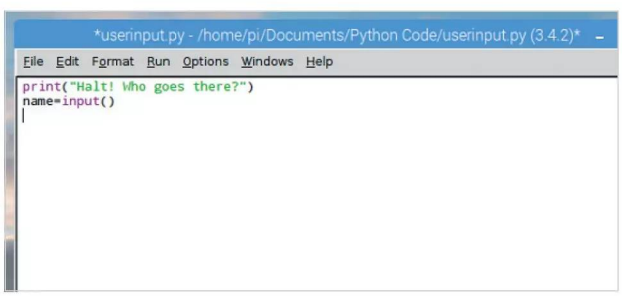
```
print(“Hello. What’s your name?”)
name=input()
```



SCHRITT 6

Der Code aus dem vorherigen Schritt wird oft als übersichtlicher angesehen, anstatt einen langen Text im Eingabebefehl zu haben, aber es ist keine in Stein gemeißelte Regel, gehen Sie daher in diesen Situationen so vor, wie Sie möchten. Erweitern Sie den Code, indem Sie Folgendes ausprobieren:

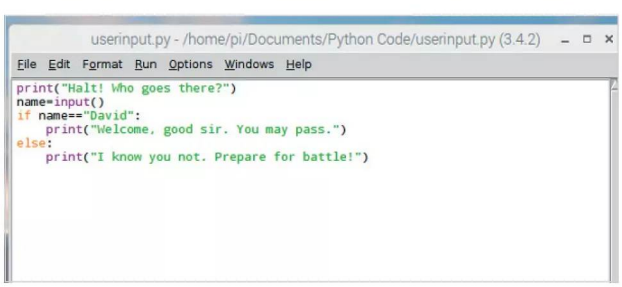
```
print(“Halt! Who goes there?”)
name=input()
```



SCHRITT 7

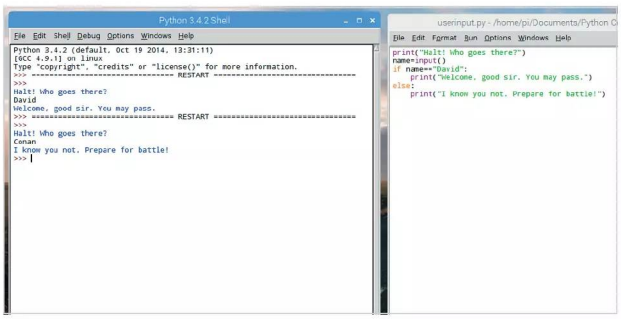
Dies ist nun ein guter Moment, um ein Textabenteuerspiel zu beginnen. Sie können den Code nun erweitern und mit der Eingabe des Benutzers das Spiel ein wenig konkretisieren:

```
if name==”David”:
  print(“Welcome, good sir. You may pass.”)
else:
  print(“I know you not. Prepare for battle!”)
```



SCHRITT 8

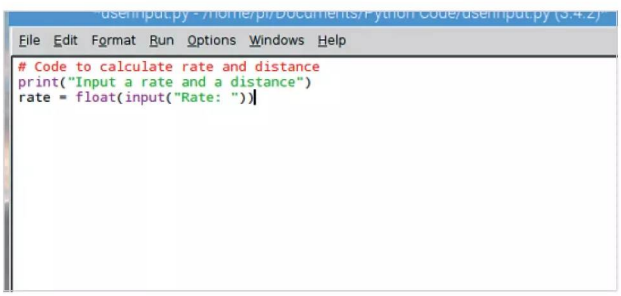
Was Sie hier erstellt haben, ist eine Bedingung, womit wir uns in Kürze näher befassen werden. Kurz gesagt, wir vergleichen die Eingaben des Benutzers gegen eine Bedingung. Wenn der Benutzer also David als seinen Namen eingibt, wird der Wächter ihm erlauben, ungehindert passieren zu können. Wird ein anderer Name als David eingegeben, fordert der Wächter ihn zum Kampf heraus.



SCHRITT 9

Wie Sie zuvor gelernt haben, ist jede Eingabe eines Benutzers automatisch ein String, also eine Zeichenkette. Sie müssen also per Typumwandlung den String in etwas anderes umwandeln. Dadurch werden im Input-Befehl einige interessante Ergänzungen erzeugt. Zum Beispiel:

```
# Code to calculate rate and distance
print(“Input a rate and a distance”)
rate = float(input(“Rate: “))
```

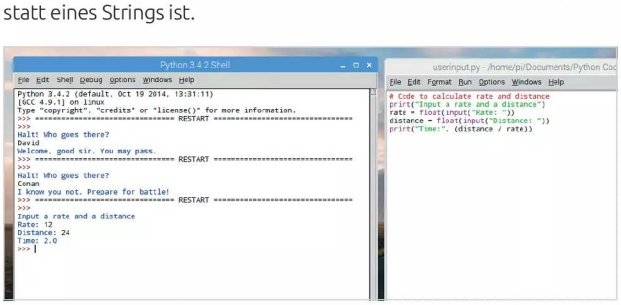


SCHRITT 10

Um den Rate- und Distance-Code zu finalisieren, können wir Folgendes hinzufügen:

```
distance = float(input(“Distance: “))
print(“Time:”, (distance / rate))
```

Speichern Sie den Code, führen Sie ihn aus und geben Sie einige Zahlen ein. Mit dem float(input-Element haben wir Python mitgeteilt, dass alles, was eingegeben wird, eine Gleitkommazahl statt eines Strings ist.



Funktionen erstellen

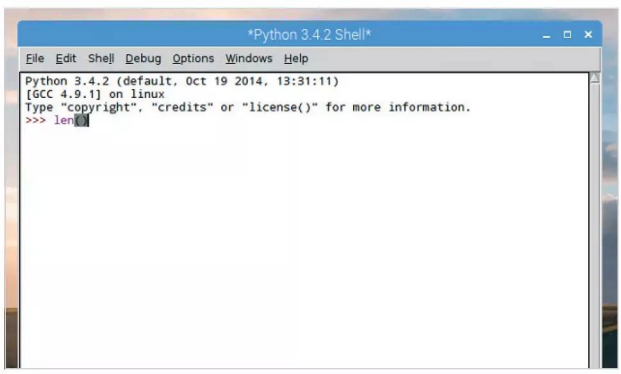
Da Sie nun die Anwendung von Variablen und Benutzereingaben beherrschen, werden wir als Nächstes Funktionen angehen. Sie haben bereits einige Funktionen wie den print-Befehl verwendet, allerdings können Sie in Python auch eigene Funktionen bestimmen.

WAS SIND FUNKTIONEN?

Eine Funktion ist ein Befehl, der in Python eingegeben wir, um etwas auszuführen. Sie ist ein kleiner in sich abgeschlossener Code, der Daten aufnimmt, bearbeitet und dann das Ergebnis zurückgibt.

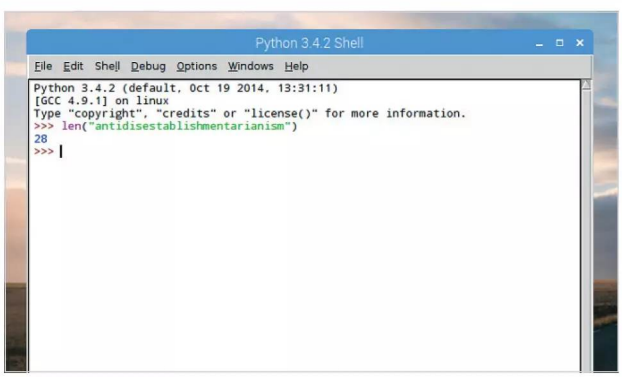
SCHRITT 1

Es sind nicht nur Daten, mit denen eine Funktion arbeitet, sie kann in Python alle möglichen nützlichen Dinge ausführen, zum Beispiel Daten sortieren, Elemente von einem Format in ein anderes Format ändern und die Länge oder den Typ von Elementen überprüfen. Grundsätzlich ist eine Funktion ein kurzes Wort, dem Klammern folgen. Zum Beispiel **len()**, **list()** oder **type()**.



SCHRITT 2

Eine Funktion nimmt Daten auf, normalerweise eine Variable, verarbeitet diese und gibt den Endwert zurück. Die zu bearbeitenden Daten werden in die Klammern eingefügt. Wenn Sie also wissen wollen, wie viele Buchstaben das Wort antidisestablishmentarianism enthält, würden Sie `len("antidisestablishmentarianism")` eingeben und als Antwort die Zahl 28 erhalten.

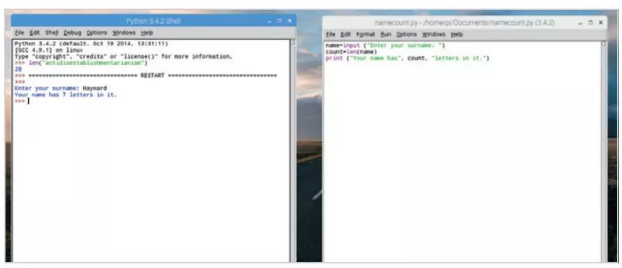


SCHRITT 3

Sie können Variablen auf ähnliche Weise durch Funktionen laufen lassen. Wenn Sie z. B. die Anzahl der Buchstaben im Nachnamen einer Person wissen möchten, könnten Sie den folgenden Code verwenden (gehen Sie für dieses Beispiel in den Texteditor):

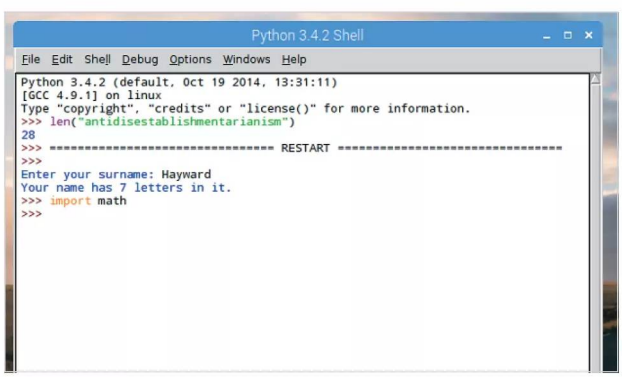
```
name=input ("Enter your surname: ")
count=len(name)
print ("Your surname has", count, "letters in
it.")
```

Drücken Sie F5 und speichern Sie den Code, um ihn auszuführen.



SCHRITT 4

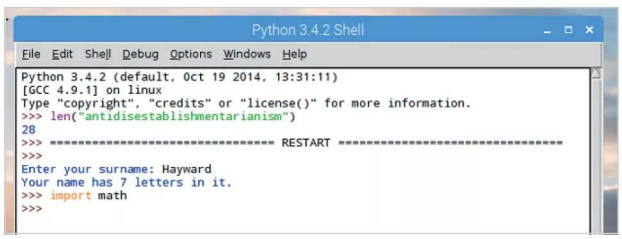
Python hat Dutzende von Funktionen eingebaut, viel zu viele, um sie hier alle erwähnen zu können. Eine Liste der in Python 3 verfügbaren integrierten Funktionen finden Sie jedoch unter **www.docs.python.org/3/library/functions.html**. Dies sind die vordefinierten Funktionen, aber da Benutzer viele weitere erstellt haben, sind dies nicht die einzigen, die verfügbar sind.



SCHRITT 5 Zusätzliche Funktionen können in Python über Module hinzugefügt werden. Python verfügt über eine große Auswahl an Modulen, die zahlreiche Programmieraufgaben abdecken. Sie fügen Funktionen hinzu und können bei Bedarf importiert werden. Um z. B. erweiterte Mathematikfunktionen zu verwenden, geben Sie Folgendes ein:

```
import math
```

Nach der Eingabe haben Sie Zugriff auf alle Funktionen des Mathematikmoduls.



SCHRITT 6 Um eine Funktion eines Moduls zu verwenden, geben Sie den Namen des Moduls ein, gefolgt von einem Punkt und dem Namen der Funktion. Zum Beispiel können Sie mit dem Mathematikmodul, das Sie gerade in Python importiert haben, die Quadratwurzelfunktion verwenden. Geben Sie dazu Folgendes ein:

```
math.sqrt(16)
```

Wie Sie sehen, wird der Code als modul.funktion(daten) dargestellt.

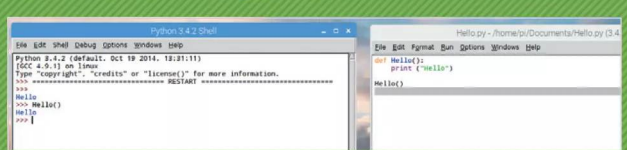
FUNKTIONEN SELBST ERSTELLEN

Es gibt viele verschiedene Funktionen, die von anderen Python-Programmierern erstellt wurden und die Sie importieren können. Sie werden sicherlich auf einige tolle Beispiele stoßen. Sie können aber auch mit dem def-Befehl Ihre eigenen erstellen.

SCHRITT 1 Gehen Sie zu File > New File, um den Editor aufzurufen. Erstellen Sie eine Funktion namens Hello, die einen Benutzer begrüßt:

```
def Hello():
    print ("Hello")

Hello()
```

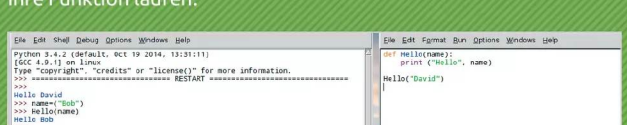
Drücken Sie F5, um das Script zu speichern und auszuführen. Sie können Hello in der Shell sehen. Wenn Sie Hello() eingeben, wird die neue Funktion zurückgegeben.



SCHRITT 2 Wir erweitern nun die Funktion, um eine Variable zu akzeptieren, z. B. den Namen des Benutzers. Ändern Sie Ihr Script wie folgt:

```
def Hello(name):
    print ("Hello", name)

Hello("David")
```

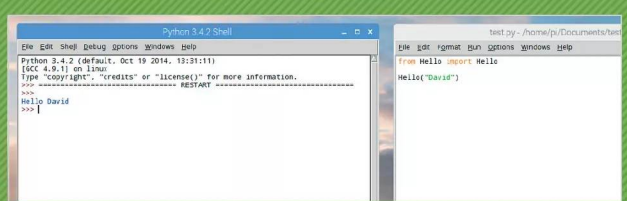
Dadurch wird nun der Variablenname akzeptiert, ansonsten wird Hello David ausgegeben. Geben Sie in der Shell Folgendes ein: name=("Bob"), then: Hello(name). Es können nun Variablen durch Ihre Funktion laufen.



SCHRITT 3 Löschen Sie nun die Zeile Hello("David"), die letzte Zeile im Script, und drücken Sie Strg + S, um das neue Script zu speichern. Schließen Sie den Editor und erstellen Sie eine neue Datei (File > New File). Geben Sie Folgendes ein:

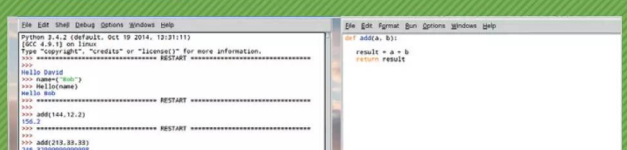
```
from Hello import Hello

Hello("David")
```

Drücken Sie F5, um den Code zu speichern und auszuführen.



SCHRITT 4 Sie haben soeben die Hello-Funktion aus dem gespeicherten Hello.py-Programm importiert und damit David begrüßt. Auf diese Weise funktionieren Module und Funktionen: Sie importieren das Modul und verwenden dann die Funktion. Probieren Sie Folgendes aus und experimentieren Sie damit:

```
def add(a, b):
    result = a + b
    return result
```



Bedingungen & Schleifen

Bedingungen und Schleifen machen ein Programm erst interessant. Sie können einfach oder ziemlich komplex sein. Wie Sie sie verwenden, hängt stark davon ab, was Sie mit dem Programm zu erreichen versuchen.

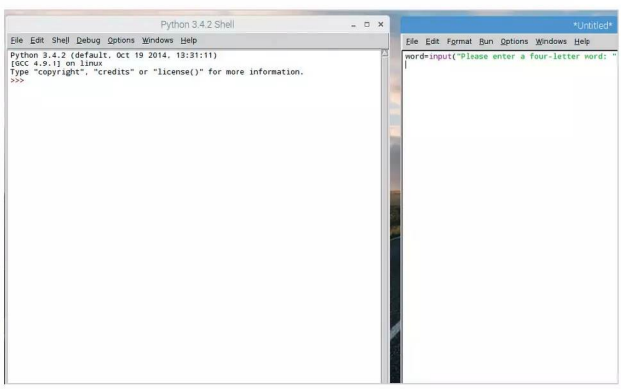
WAHRE BEDINGUNGEN

Wenn zu Beginn die Bedingungen einfach gehalten werden, macht das Lernen des Programmierens mehr Spaß. Wir beginnen daher damit, zu prüfen, ob etwas wahr ist; ist es das nicht, wird etwas anderes ausgeführt.

SCHRITT 1

Wir erstellen ein neues Python-Programm, das den Benutzer auffordert, ein Wort einzugeben und dann prüft, ob es ein aus vier Buchstaben bestehendes Wort ist oder nicht. Beginnen Sie mit File > New File und geben Sie die Eingabevariablen ein:

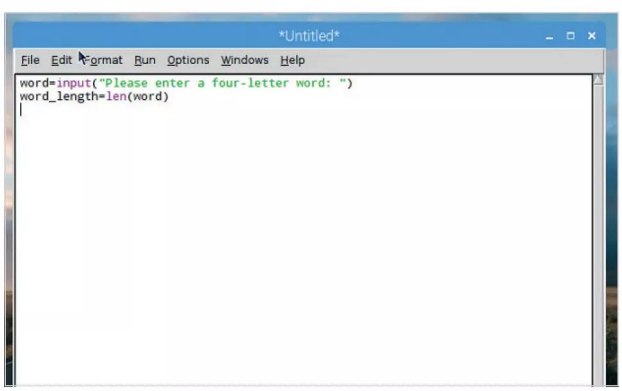
```
word=input("Please enter a four-letter word: ")
```



SCHRITT 2

Wir erstellen nun eine neue Variable und lassen die Wortvariable durch die len-Funktion laufen, um die Gesamtzahl der Buchstaben zu erhalten, die der Benutzer gerade eingegeben hat:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
```

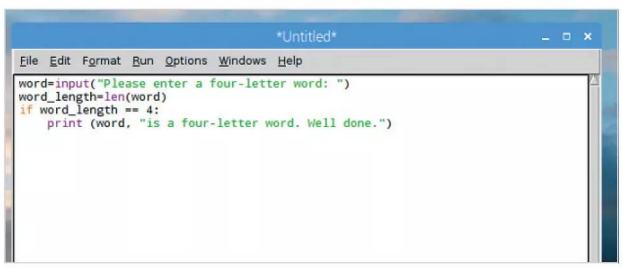


SCHRITT 3

Sie können nun mithilfe einer if-Anweisung überprüfen, ob die Variable word_length vier entspricht und eine freundliche Bestätigung ausgeben lassen, wenn dies der Fall ist:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length == 4:
   print (word, "is a four-letter word. Well done.")
```

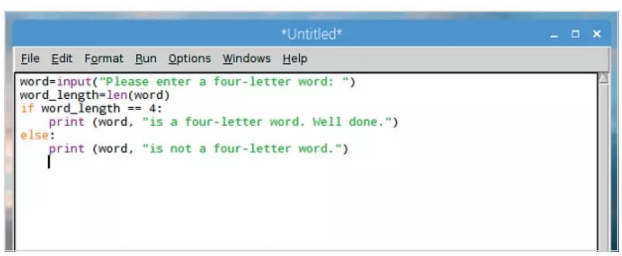
Das doppelte Gleichheitszeichen (==) überprüft, ob eine Übereinstimmung herrscht.



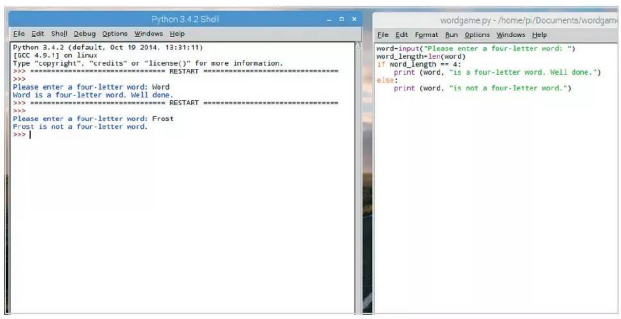
SCHRITT 4

Der Doppelpunkt am Ende von if teilt Python mit, dass alles nach dem eingerückten Doppelpunkt ausgeführt wird, wenn die Anweisung wahr ist. Bewegen Sie nun den Cursor an den Anfang des Editors:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length == 4:
   print (word, "is a four-letter word. Well
done.")
else:
   print (word, "is not a four-letter word.")
```

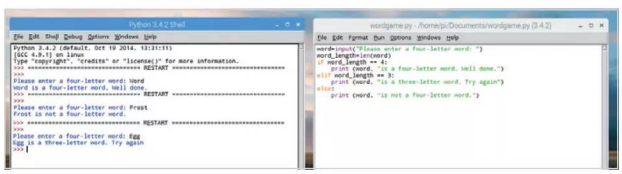


SCHRITT 5 Drücken Sie F5 und speichern Sie den Code, um ihn auszuführen. Geben Sie ein aus vier Buchstaben bestehendes Wort in die Shell ein. Sie sollten die Nachricht erhalten, dass es sich um vier Buchstaben handelt. Drücken Sie erneut F5, aber geben Sie diesmal ein aus fünf Buchstaben bestehendes Wort ein. Die Shell zeigt an, dass es sich nicht um ein aus vier Buchstaben bestehendes Wort handelt.



SCHRITT 6 Erweitern Sie den Code und fügen Sie eine weitere Bedingung hinzu. Wir haben eine Bedingung für aus drei Buchstaben bestehende Wörter hinzugefügt:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length == 4:
   print (word, "is a four-letter word. Well
done.")
elif word_length == 3:
   print (word, "is a three-letter word. Try again.")
else:
   print (word, "is not a four-letter word.")
```

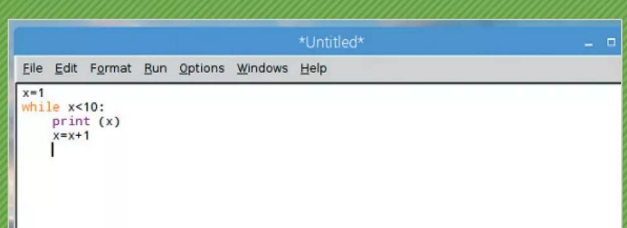


SCHLEIFEN

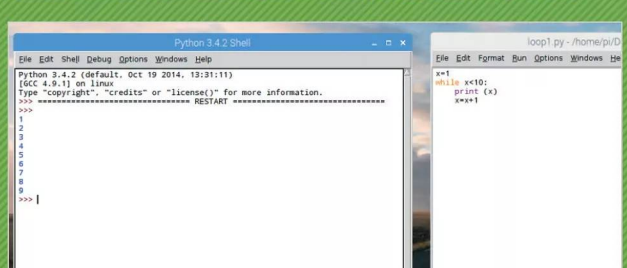
Eine Schleife ähnelt einer Bedingung, unterscheidet sich jedoch in ihrer Funktionsweise. Eine Schleife läuft mehrere Male den gleichen Code durch, normalerweise mit der Unterstützung einer Bedingung.

SCHRITT 1 Wir beginnen mit einer einfachen while-Anweisung. Wie if prüft diese, ob etwas wahr ist, und führt dann den eingerückten Code aus:

```
x = 1
while x < 10:
   print (x)
   x = x + 1
```

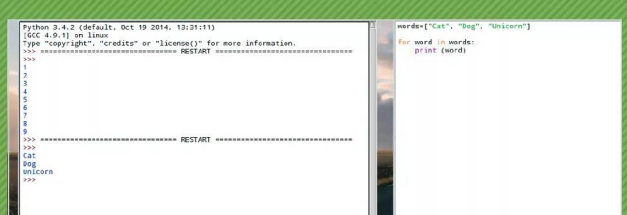


SCHRITT 2 Der Unterschied zwischen if und while ist, dass while zurückgeht und prüft, ob die Aussage noch wahr ist, wenn das Ende des eingerückten Codes erreicht wird. In unserem Beispiel ist x kleiner als 10. Bei jeder Schleife wird der aktuelle Wert von x ausgegeben und um eins addiert. Wenn x schließlich gleich 10 ist, endet das Programm.



SCHRITT 3 Die for-Schleife ist ein weiteres Beispiel. Sie durchläuft eine Reihe von Daten, normalerweise eine Liste, die als Variablen in eckigen Klammern gespeichert werden. Zum Beispiel:

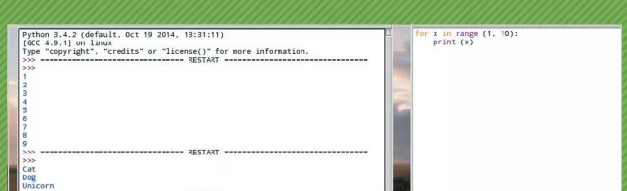
```
words=["Cat", "Dog", "Unicorn"]

for word in words:
   print (word)
```



SCHRITT 4 Die for-Schleife kann auch mithilfe der in range-Funktion im Countdown-Beispiel verwendet werden:

```
for x in range (1, 10):
  print (x)
```

Der x=x+1-Teil wird hier nicht benötigt, da die in range-Funktion eine Liste zwischen der ersten und der zuletzt verwendeten Nummer erstellt.



Python-Module

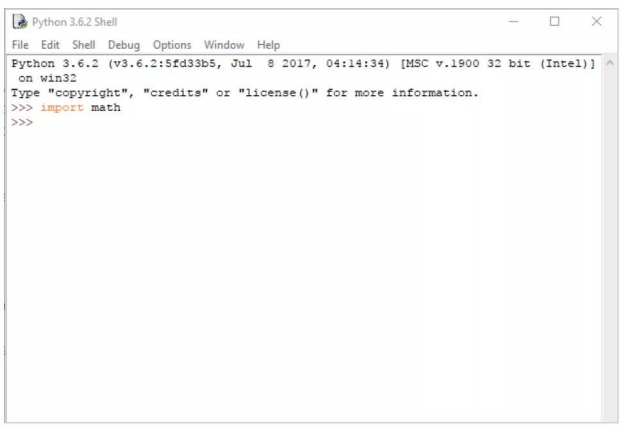
Wir haben Module bereits erwähnt (das Mathematikmodul), aber da Module für die optimale Nutzung von Python wichtig sind, ist es sinnvoll, ihnen etwas mehr Zeit zu widmen. In diesem Tutorial verwenden wir die Windows-Version von Python 3.

MEISTERN SIE MODULE

Stellen Sie sich Module als eine Art Erweiterung vor, die in Ihren Python-Code importiert wird, um seine Funktionen zu verbessern und zu erweitern. Es gibt unzählige Module und wie wir wissen, können wir sogar unsere eigenen erstellen.

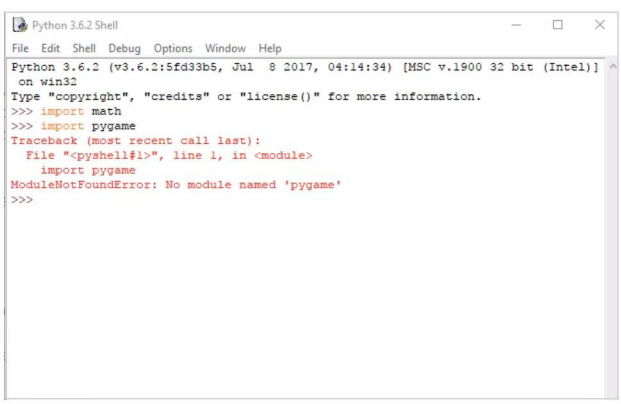
SCHRITT 1

Die in Python integrierten Funktionen sind zwar gut, allerdings auch begrenzt. Die Anwendung von Modulen erlaubt uns jedoch, anspruchsvollere Programme zu erstellen. Wie Sie wissen, handelt es sich bei Modulen um importierte Python-Scripts, z. B. import math.



SCHRITT 2

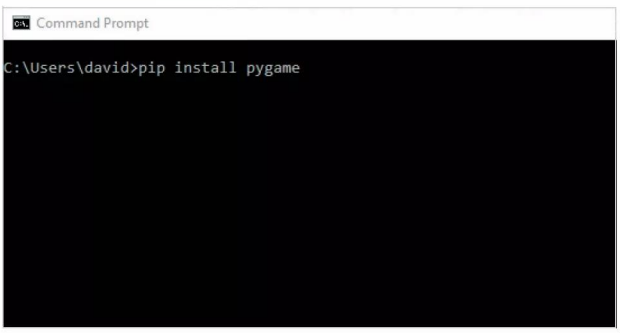
Einige Module sind vor allem auf dem Raspberry Pi standardmäßig enthalten, das Mathemodul ist z. B. ein Paradebeispiel dafür. Leider sind andere Module nicht immer verfügbar. Ein gutes Beispiel für Nicht-Pi-Plattformen ist das Pygame-Modul, das viele Funktionen zur Erstellung von Spielen enthält. Probieren Sie Folgendes aus: **import pygame**.



SCHRITT 3

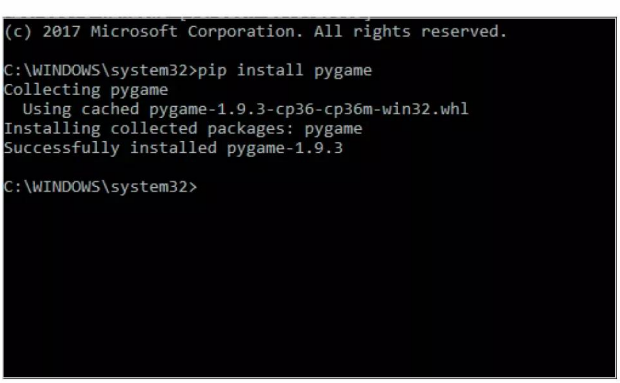
Als Ergebnis erhalten Sie eine Fehlermeldung in der IDLE-Shell, da das Pygame-Modul in Python nicht erkannt wird oder nicht installiert ist. Um ein Modul zu installieren, verwenden wir PIP (Pip Installs Packages). Schließen Sie die IDLE-Shell und öffnen Sie die Eingabeaufforderung oder das Terminal. Geben Sie in der Eingabeaufforderung mit erhöhten Administratorrechten Folgendes ein:

```
pip install pygame
```



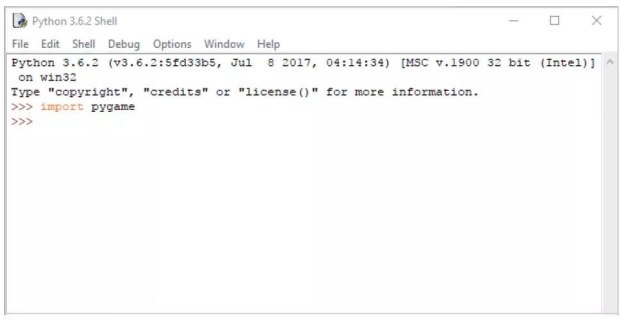
SCHRITT 4

Die PIP-Installation erfordert aufgrund der Installation von Komponenten an verschiedenen Speicherorten einen erhöhten Status. Windows-Benutzer können dazu im Suchfeld neben dem Startmenü nach „Eingabeaufforderung", suchen, mit der rechten Maustaste auf das Ergebnis klicken und dann „Als Administrator ausführen" wählen. Linux- und Mac-Benutzer können den sudo-Befehl mit sudo pip install package verwenden.



SCHRITT 5

Schließen Sie die Eingabeaufforderung bzw. das Terminal und öffnen Sie erneut die IDLE-Shell. Wenn Sie nun `import pygame` eingeben, wird das Modul problemlos in den Code importiert. Sie werden feststellen, dass der meiste Code, der aus dem Internet heruntergeladen oder kopiert wird, ein Modul enthält; diese stellen gewöhnlich die Fehlerquelle in der Ausführung dar, wenn sie fehlen.



SCHRITT 6

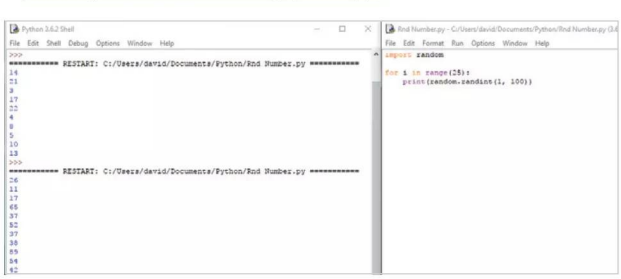
Die Module enthalten den zusätzlichen Code, der benötigt wird, um ein bestimmtes Ergebnis im eigenen Code zu erzielen, wie wir es bereits zuvor getestet haben. Der Code `import random` importiert z. B. den Code aus dem Zufallsgeneratormodul. Mit diesem Modul können Sie dann Folgendes erstellen:

```
for i in range(10):
  print(random.randint(1, 25))
```

SCHRITT 7

Wenn dieser Code gespeichert und ausgeführt wird, werden zehn Zufallszahlen von 1 bis 25 angezeigt. Sie können mit dem Code experimentieren und mehr oder weniger aus einem großen oder kleineren Bereich anzeigen, zum Beispiel:

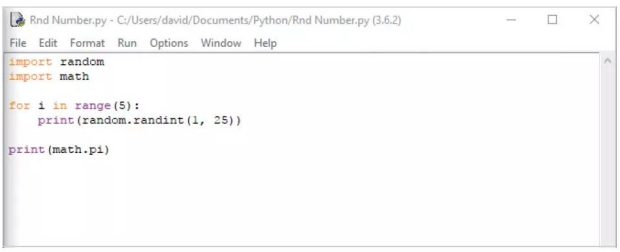
```
import random

for i in range(25):
  print(random.randint(1, 100))
```



SCHRITT 8

Es können mehrere Module innerhalb Ihres Codes importiert werden. Um unser Beispiel zu erweitern, geben Sie Folgendes ein:

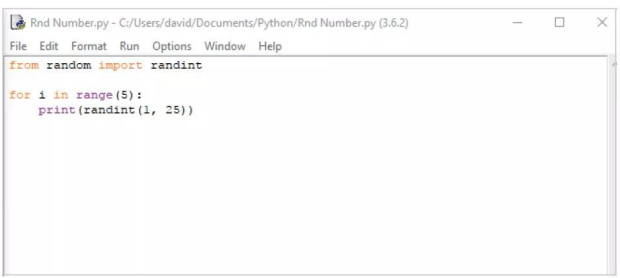
```
import random
import math

for I in range(5):
  print(random.randint(1, 25))

print(math.pi)
```



SCHRITT 9

Das Ergebnis ist eine Folge von Zufallszahlen gefolgt von dem Wert für Pi, so wie er mit der Funktion print(math.pi) durch das Mathematikmodul bezogen wird. Bestimmte Funktionen können auch mithilfe der Befehle from und import aus einem Modul abgerufen werden, z. B.:

```
from random import randint

for i in range(5):
  print(randint(1, 25))
```

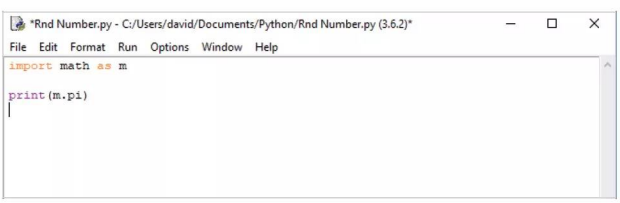


SCHRITT 10

Dies hilft einen optimalen Ansatz für die Programmierung zu erstellen. Sie können auch das Importmodul * verwenden, das alles importiert, was im erwähnten Modul definiert ist, auch wenn das oft als Verschwendung von Ressourcen betrachtet wird. Zu guter Letzt können Module als Aliasse importiert werden:

```
import math as m

print(m.pi)
```

Das Hinzufügen von Kommentaren hilft natürlich, andere wissen zu lassen, was passiert.



Python-Fehler

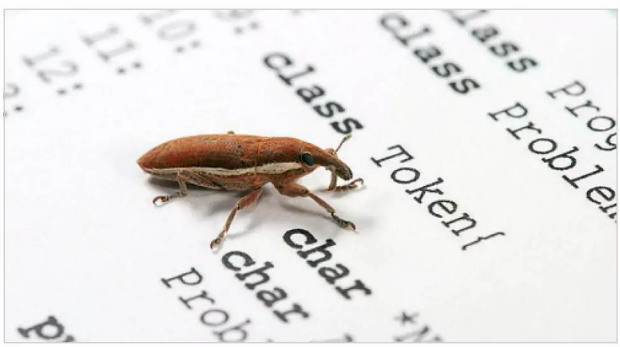
Es ist selbstverständlich, dass Sie irgendwann in Ihrem Code auf einen Fehler stoßen werden, bei dem Python nicht fortfahren kann, weil etwas fehlt, falsch oder einfach unbekannt ist. Die Fähigkeit, diese Fehler zu erkennen, macht einen guten Programmierer aus.

FEHLER BEHEBEN

Fehler im Code sind völlig normal. Sie können oft mit etwas Geduld leicht behoben werden. Wichtig ist, weiter zu schauen, zu experimentieren und zu testen, um letztendlich einen fehlerfreien Code zu haben.

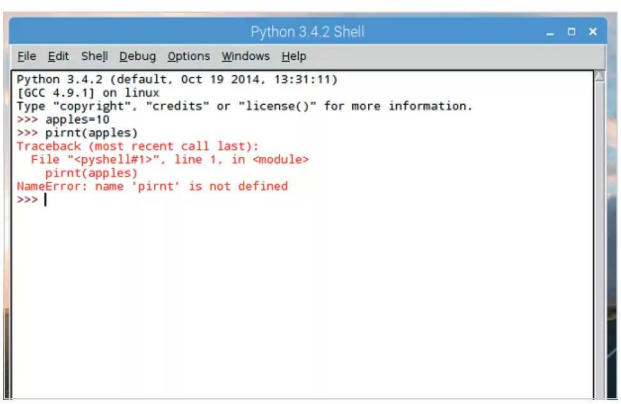
SCHRITT 1

Code ist nicht so flüssig wie das geschriebene Wort, ganz gleich, wie gut die Programmiersprache ist. Python ist definitiv einfacher als die meisten Sprachen, ist aber vor einigen lästigen Fehlern auch nicht sicher. Die häufigsten sind Tippfehler des Benutzers, die in einem einfachen Code, der aus einem Dutzend Zeilen besteht, leicht zu finden sind, aber stellen Sie sich vor, Sie müssen Fehler in einem aus mehreren tausend Zeilen bestehenden Code beheben!



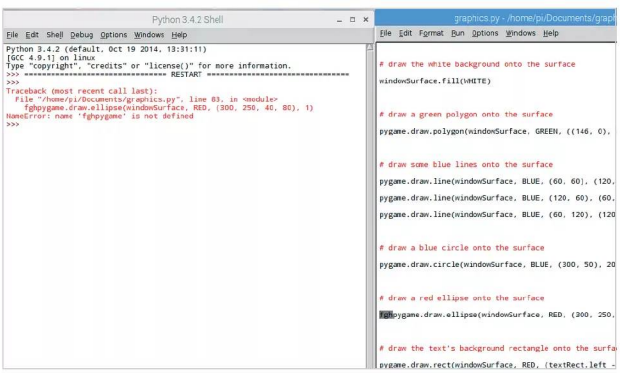
SCHRITT 2

Wie soeben erwähnt, ist der häufigste Fehler der Tippfehler. Tippfehler befinden sich häufig auf der Befehlsebene, z. B. der falschen Eingabe des print-Befehls. Sie treten jedoch auch auf, wenn Sie zahlreiche Variablen mit langen Namen haben. Der beste Rat ist, den Code einfach durchzugehen und Ihre Rechtschreibung zu überprüfen.



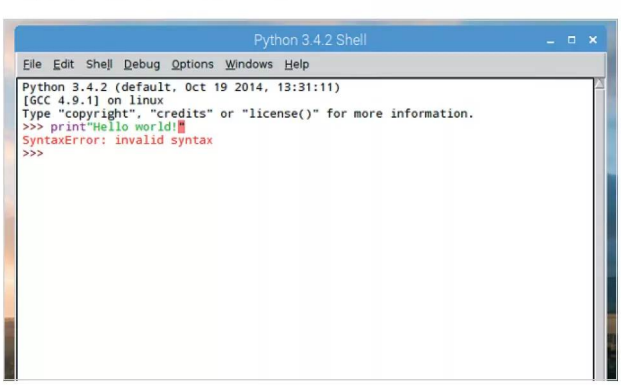
SCHRITT 3

Wenn es darum geht, Fehlermeldungen anzuzeigen, kommt Python uns zum Glück entgegen. Wenn Sie eine in rot geschriebene Fehlermeldung von der IDLE-Shell erhalten, wird der Fehler zusammen mit der Zeilennummer angegeben, in der der Fehler aufgetreten ist. Im IDLE-Editor ist dies bei langem Code ein wenig frustrierend, Texteditoren hingegen haben jedoch eine hilfreiche Zeilennummerierung.



SCHRITT 4

Syntaxfehler sind wahrscheinlich die zweithäufigsten Fehler, auf die Sie als Programmierer stoßen werden. Auch wenn die Schreibweise korrekt ist, ist der eigentliche Befehl falsch. In Python 3 tritt dies häufig auf, wenn Python 2-Syntaxen angewendet werden. Am ärgerlichsten ist der print-Befehl. In Python 3 lautet es z. B. **print("words")**, während es in Python2 **print "words"** heißt.



SCHRITT 5

Klammern können ebenfalls zu ärgerlichen Programmierfehlern führen, insbesondere wenn Sie Eingaben wie die Folgende haben:

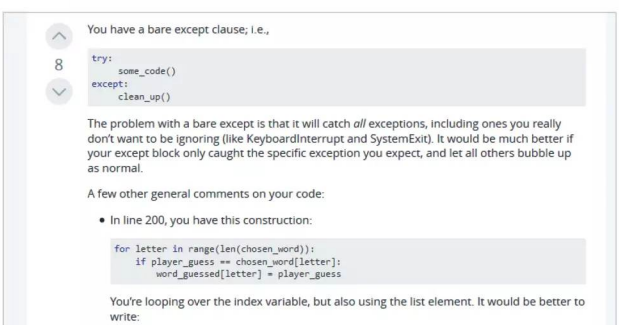
```
print(balanced_check(input()))
```

Denken Sie daran, dass es für jede geöffnete Klammer auch eine geschlossene Klammer geben muss.

```
import sys

def balanced_check(data):
    stack = []
    characters = list(data)

    for character in characters:
        reference = {
            '(': ')',
            '{': '}',
            '[': ']'
        }
        if character in reference.keys():
            stack.append(character)

        elif character in reference.values() and len(stack) > 0:
            char = stack.pop()
```

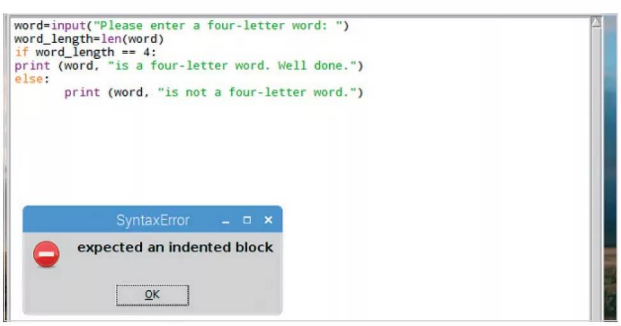
SCHRITT 6

Im Internet gibt es tausende von Python-Ressourcen, Code-Abschnitten und langwierigen Diskussionen in Foren, wie am besten etwas erreicht wird. Obwohl 99 % davon fehlerfreier Code ist, sollten Sie sich nicht dazu verlocken lassen, ständig einen beliebigen Code in Ihren Editor zu kopieren und einzufügen. In den meisten Fällen wird es nicht funktionieren und das Schlimmste ist, dass Sie nichts gelernt haben.



SCHRITT 7

Einrückungen sind ein unangenehmer Teil der Python-Programmierung, mit denen viele Anfänger zu kämpfen haben. Erinnern Sie sich an die if-Schleife aus dem Abschnitt Bedingungen & Schleifen, wo der Doppelpunkt bedeutet, dass alles, was nach der Anweisung eingerückt ist, ausgeführt werden soll, solange es wahr ist? Wird die Einrückung vergessen oder zu stark eingerückt, erscheint eine Fehlermeldung.

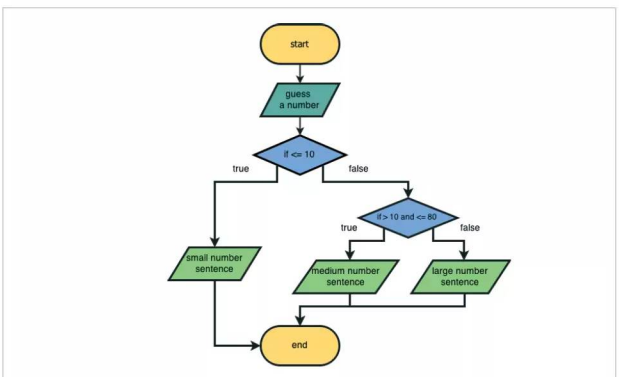


SCHRITT 8

Eine tolle Möglichkeit, Ihren Code Schritt-für-Schritt zu überprüfen ist die Visualisierungsseite des Python Tutors, die Sie unter **www.pythontutor.com/visualize.html#mode=edit** finden. Fügen Sie Ihren Code einfach in den Editor ein und klicken Sie auf die Schaltfläche „Visualization Execution", um den Code Zeile für Zeile auszuführen. Dies hilft, Fehler und Missverständnisse zu beseitigen.

SCHRITT 9

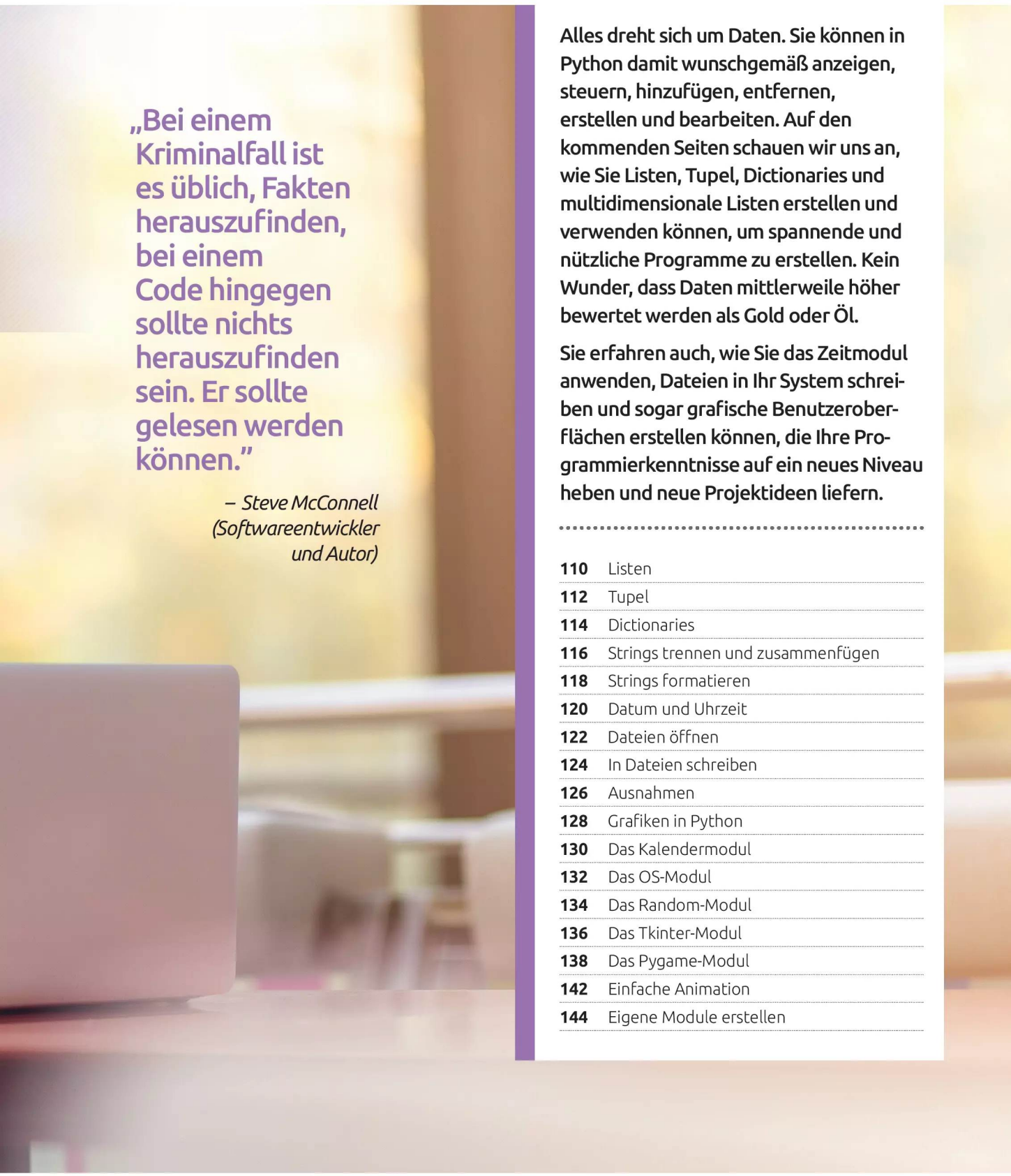
Planung sorgt für guten Code. Während es zwar etwas alte Schule ist, ist es vor der Eingabe des Codes sinnvoll zu planen, was er bezwecken soll. Listen Sie die zu verwendenden Variablen und Module auf und schreiben Sie dann ein Script für jede Benutzerinteraktion und -ausgabe.



SCHRITT 10

Rein interessehalber stammt in der Computersprache das englische Wort für Fehlerbehebung – Debugging – von Admiral Grace Hopper, die in den 40ern an einem monolithischen elektromechanischen Harvard Mark II-Computer arbeitete. Der Legende nach fand Hopper eine Motte, die in einem Relais steckte, wodurch das System nicht funktionierte. Die Entfernung der Motte wurde daher als Debugging bezeichnet.

Arbeiten mit Daten



„Bei einem Kriminalfall ist es üblich, Fakten herauszufinden, bei einem Code hingegen sollte nichts herauszufinden sein. Er sollte gelesen werden können."

– Steve McConnell (Softwareentwickler und Autor)

Alles dreht sich um Daten. Sie können in Python damit wunschgemäß anzeigen, steuern, hinzufügen, entfernen, erstellen und bearbeiten. Auf den kommenden Seiten schauen wir uns an, wie Sie Listen, Tupel, Dictionaries und multidimensionale Listen erstellen und verwenden können, um spannende und nützliche Programme zu erstellen. Kein Wunder, dass Daten mittlerweile höher bewertet werden als Gold oder Öl.

Sie erfahren auch, wie Sie das Zeitmodul anwenden, Dateien in Ihr System schreiben und sogar grafische Benutzeroberflächen erstellen können, die Ihre Programmierkenntnisse auf ein neues Niveau heben und neue Projektideen liefern.

Listen

Listen sind eine der häufigsten Arten von Datenstrukturen, auf die Sie in Python stoßen werden. Eine Liste ist einfach eine Sammlung von Elementen bzw. Daten, auf die Sie als Ganzes oder einzeln zugreifen können.

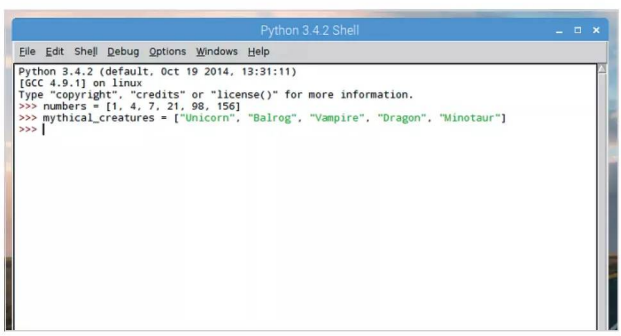
MIT LISTEN ARBEITEN

Listen sind in Python äußerst praktisch. Eine Liste kann aus Strings, Integer und auch Variablen bestehen. Sie können sogar Funktionen in Listen und Listen innerhalb von Listen einfügen.

SCHRITT 1

Eine Liste ist eine Folge von Datenwerten, die als Elemente bezeichnet werden. Sie erstellen den Namen Ihrer Liste, gefolgt vom Gleichheitszeichen, dann eckigen Klammern und den durch Kommata getrennten Elementen. Beachten Sie, dass Strings Anführungszeichen verwenden:

```
numbers = [1, 4, 7, 21, 98, 156]
mythical_creatures – [“Unicorn”, “Balrog”,
“Vampire”, “Dragon”, “Minotaur”]
```



SCHRITT 2

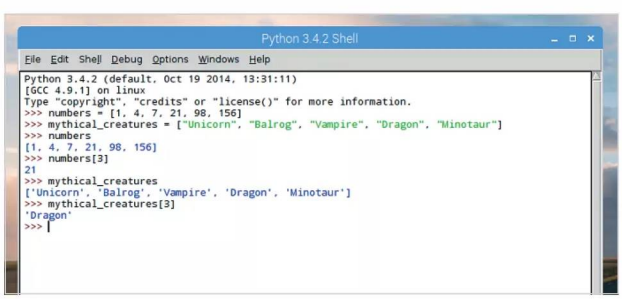
Wenn Sie Ihre Liste definiert haben, können Sie sie bzw. einen Listeneintrag über deren Namen gefolgt von einer Nummer aufrufen. Listen starten den ersten Eintrag als 0, gefolgt von 1, 2, 3 und so weiter. Hier ein Beispiel:

```
numbers
```

Ruft den gesamten Inhalt einer Liste auf.

```
numbers[3]
```

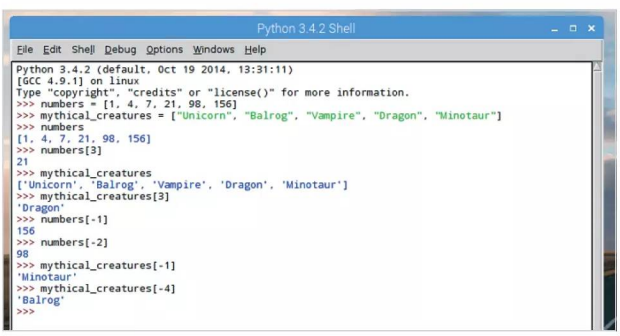
Ruft in der Liste das dritte Element von Null auf (in diesem Fall 21).



SCHRITT 3

Sie können auch auf das letzte Element in einer Liste zugreifen oder es indizieren, indem Sie ein Minuszeichen vor der Elementnummer [-1] setzen, oder auf das vorletzte Element mit [-2] usw. Wenn Sie versuchen, auf ein Element zu verweisen, das nicht in der Liste enthalten ist, z. B. [10], erhalten Sie eine Fehlermeldung:

```
numbers[-1]
mythical_creatures[-4]
```



SCHRITT 4

Slicing (in Scheiben schneiden) ähnelt dem Indexieren, Sie können jedoch mehrere Elemente in einer Liste abrufen, indem Sie die Positionsnummern durch einen Doppelpunkt trennen, z. B.:

```
numbers[1:3]
```

Dies gibt die Zahlen 4 und 7 aus, die die Positionsnummern 1 und 2 sind. Beachten Sie, dass die zurückgegebenen Werte nicht die dritte Positionsnummer enthalten, wie man es bei numbers[1:3] erwarten würde.

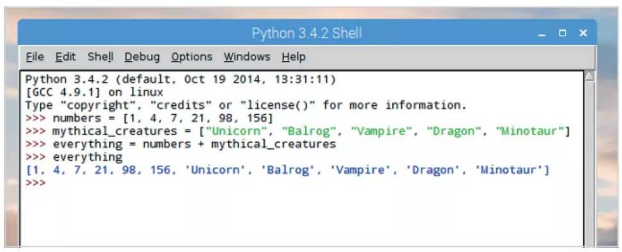
SCHRITT 5

Sie können Elemente in einer vorhandenen Liste aktualisieren und entfernen und sogar Listen zusammenführen. Um z. B. zwei Listen zu verbinden, können Sie Folgendes eingeben:

```
everything = numbers + mythical_creatures
```

Die kombinierte Liste können Sie sich dann mit folgendem Befehl ansehen:

```
everything
```



SCHRITT 6

Mit der folgenden Eingabe können Elemente zu einer Liste hinzugefügt werden:

```
numbers=numbers+[201]
```

Für Strings:

```
mythical_creatres=mythical_creatures+[“Griffin”]
```

Alternativ geht auch die append-Funktion:

```
mythical_creatures.append(“Nessie”)
numbers.append(278)
```

SCHRITT 7

Das Entfernen von Elementen kann auf zwei Arten erfolgen. Durch die Positionsnummer:

```
del numbers[7]
```

oder alternativ nach Elementname:

```
mythical_creatures.remove(“Nessie”)
```

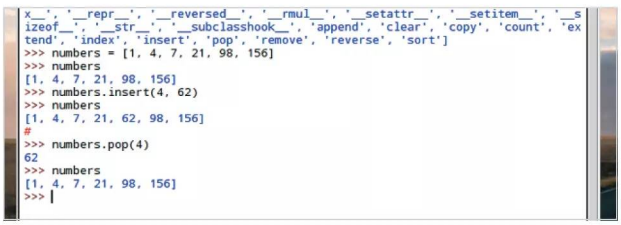
SCHRITT 8

Sie können sich anzeigen lassen, was mit Listen getan werden kann, indem Sie dir(list) in die Shell eingeben. Die Ausgabe besteht aus den verfügbaren Funktionen, z. B. werden insert und pop zum Hinzufügen und Entfernen von Elementen an bestimmten Positionen verwendet. Um die Nummer 62 an der Indexposition 4 einzufügen, geben Sie Folgendes ein:

```
numbers.insert(4, 62)
```

Hiermit entfernen Sie sie wieder:

```
numbers.pop(4)
```



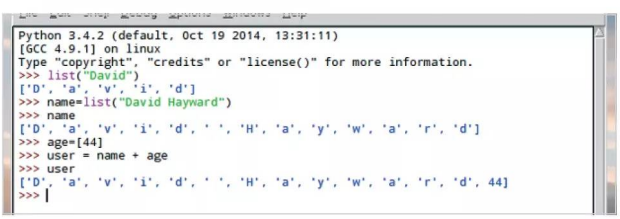
SCHRITT 9

Mit der list-Funktion werden auch Strings in ihre Komponenten zerlegt, z. B.:

```
list(“David”)
```

zerlegt den Namen David in ‘D’, ‘a’, ‘v’, ‘i’, ‘d’. Dies kann dann in eine neue Liste übernommen werden:

```
name=list(“David Hayward”)
name
age=[44]
user = name + age
user
```

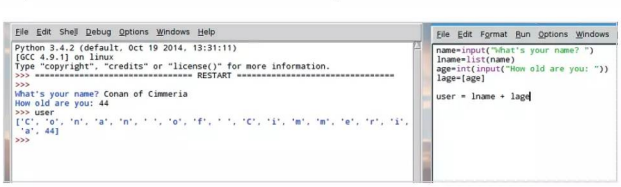


SCHRITT 10

Auf dieser Grundlage können Sie ein Programm erstellen, um den Namen und das Alter einer Person als Liste zu speichern:

```
name=input(“What’s your name? “)
lname=list(name)
age=int(input(“How old are you: “))
lage=[age]

user = lname + lage
```

Die kombinierte Namens- und Altersliste haben wir user genannt. Durch die Eingabe von user in die Shell kann sie aufgerufen werden. Experimentieren Sie und schauen Sie, was Sie alles machen können.



Tupel

Tupel sind Listen sehr ähnlich. Während Listen jedoch aktualisiert, gelöscht oder auf andere Weise geändert werden können, bleibt ein Tupel eine Konstante. Ein Tupel ist unveränderbar und eignet sich perfekt zum Speichern fester Datenelemente.

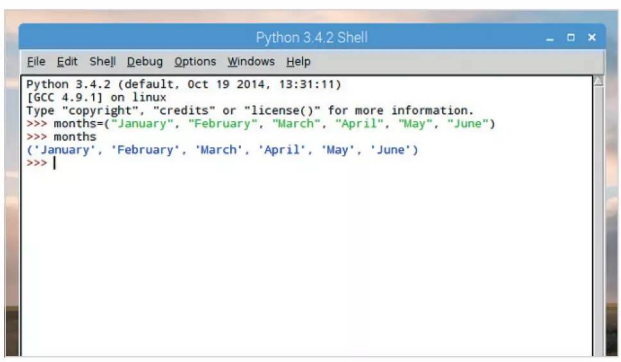
DAS UNVERÄNDERBARE TUPEL

Die Gründe für Tupel variieren je nachdem, was das Programm ausführen soll. In der Regel sind Tupel für etwas Besonderes vorbehalten, aber sie werden auch z. B. in Abenteuerspielen verwendet, in denen nicht spielende Charakternamen gespeichert sind.

SCHRITT 1

Ein Tupel wird auf die gleiche Weise wie eine Liste erstellt. In diesem Fall werden jedoch anstelle von eckigen Klammern gebogene Klammern verwendet. Hier ein Beispiel:

```
months=(“January”, “February”, “March”, “April”, “May”, “June”)
months
```

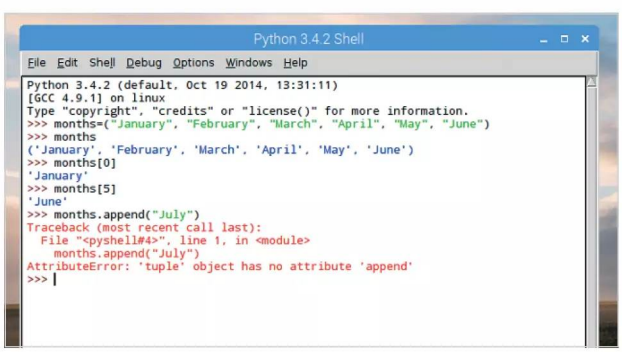


SCHRITT 2

Genau wie bei Listen können die Elemente innerhalb eines benannten Tupels entsprechend ihrer Position im Datenbereich indiziert werden, d. h.:

```
months[0]
months[5]
```

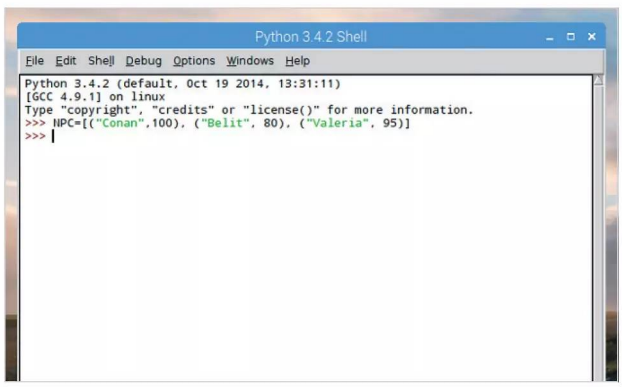
Jeder Versuch, das Tupel zu löschen oder etwas hinzuzufügen, führt jedoch zu einer Fehlermeldung in der Shell.



SCHRITT 3

Sie können gruppierte Tupel in Listen mit mehreren Datensätzen erstellen. Hier ist z. B. ein Tupel namens NPC (Non-Playable Characters), das den Charakternamen und die Kampfwertung für ein Abenteuerspiel enthält:

```
NPC=[(“Conan”, 100), (“Belit”, 80), (“Valeria”, 95)]
```

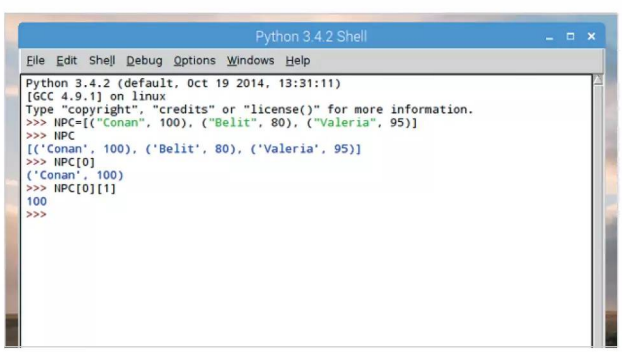


SCHRITT 4

Auf jedes dieser Datenelemente kann als Ganzes zugegriffen werden, indem NPC in die Shell eingegeben wird; sie können aber auch gemäß ihrer Position NPC[0] indiziert werden. Sie können die einzelnen Tupel auch in der NPC-Liste indizieren:

```
NPC[0][1]
```

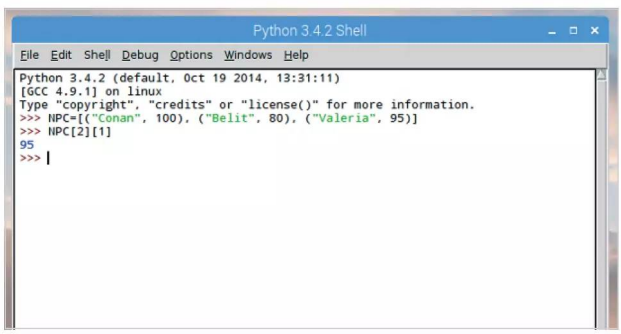
Diese Eingabe wird 100 ausgeben.



SCHRITT 5

Es ist erwähnenswert, dass sich die Indexierung beim Verweisen auf mehrere Tupel innerhalb einer Liste geringfügig von der Norm unterscheidet. Man würde meinen, dass die Kampfwertung von 95 des Charakters Valeria NPC[4][5] entspricht, aber das tut es nicht. Tatsächlich ist es:

```
NPC[2][1]
```

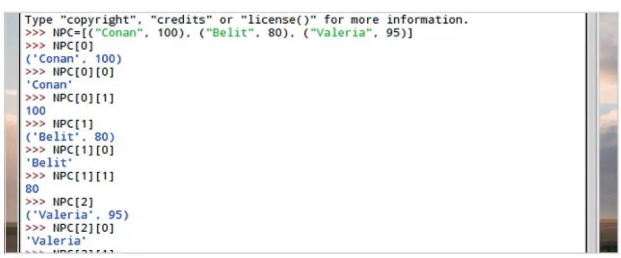


SCHRITT 6

Das bedeutet natürlich, dass die Indizierung folgendermaßen aussieht:

```
0                1, 1
0, 0             2
0, 1             2, 0
1                2,1
1, 0
```

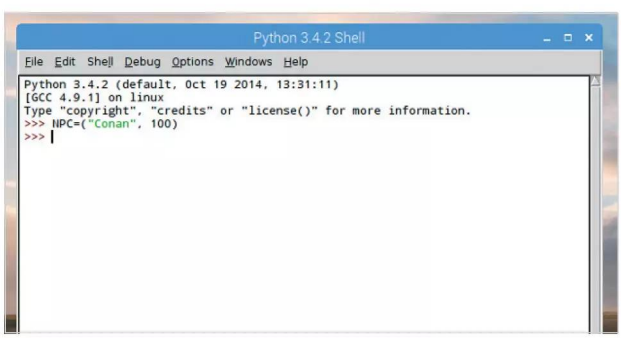
Wie Sie sich vorstellen können, kann dies bei der Bearbeitung vieler Tupel-Daten etwas verwirrend sein.



SCHRITT 7

Tupel verwenden jedoch eine Funktion namens unpacking, bei der den im Tupel gespeicherten Datenelementen Variablen zugewiesen werden. Erstellen Sie zuerst das Tupel mit den zwei Elementen name und combat rating:

```
NPC=("Conan", 100)
```

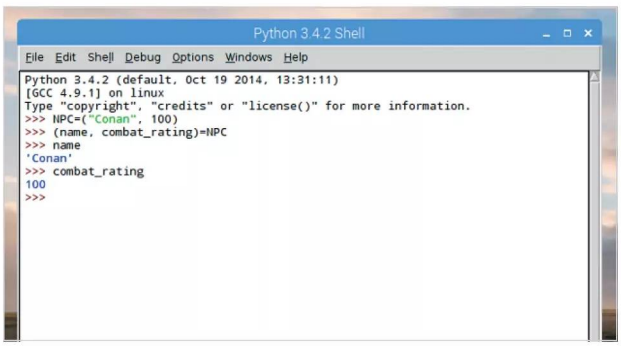


SCHRITT 8

Entpacken Sie nun das Tupel in zwei übereinstimmende Variablen:

```
(name, combat_rating)=NPC
```

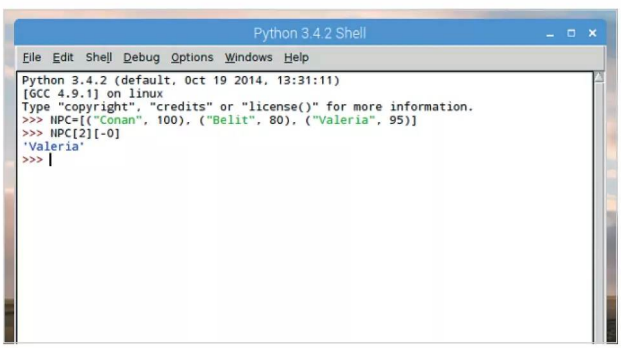
Sie können nun die Werte überprüfen, indem Sie name und combat_rating eingeben.



SCHRITT 9

Denken Sie daran, dass Sie Tupel genau wie Listen mit negativen Zahlen indizieren können, die vom Ende der Datenliste rückwärts zählen. In unserem Beispiel würden Sie auf den Charakter Valeria mit dem Tupel, der mehrere Datenelemente enthält, wie folgt verweisen:

```
NPC[2][-0]
```



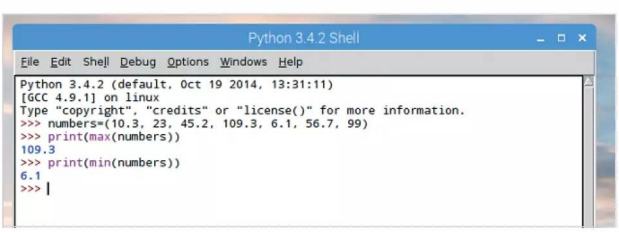
SCHRITT 10

Mit den max- und min-Funktionen können Sie den höchsten bzw. den niedrigsten Wert eines aus Zahlen bestehenden Tupels finden. Hier ein Beispiel:

```
numbers=(10.3, 23, 45.2, 109.3, 6.1, 56.7, 99)
```

Bei den Zahlen kann es sich um Integer sowie um Floats handeln. Um den höchsten und den niedrigsten Wert auszugeben, geben Sie Folgendes ein:

```
print(max(numbers))
print(min(numbers))
```



Dictionaries

Listen sind sehr nützlich, aber in Python sind Dictionaries bei weitem die technischere Art des Umgangs mit Datenelementen. Sie können zu Beginn etwas schwierig sein, aber es wird nicht lange dauern, bis Sie sie bald in Ihrem eigenen Code anwenden können.

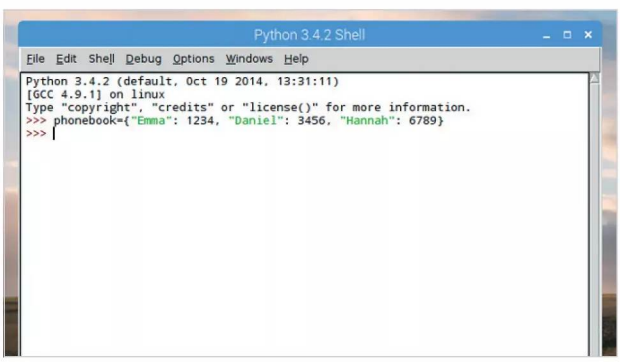
IM DOPPELPACK

Ein Dictionary ist wie eine Liste, allerdings kommt jedes Datenelement als Paar, als Schlüssel und Wert. Der Schlüsselteil muss einzigartig sein und kann entweder eine Zahl oder ein String sein, während der Wert ein beliebiges Datenelement sein kann.

SCHRITT 1

Angenommen, Sie möchten ein Telefonbuch in Python erstellen. Sie würden den Namen im Dictionary erstellen und die Daten in geschweiften Klammern eingeben, wobei Schlüssel und Wert durch einen Doppelpunkt getrennt werden: `Schlüssel: Wert`. Hier ein Beispiel:

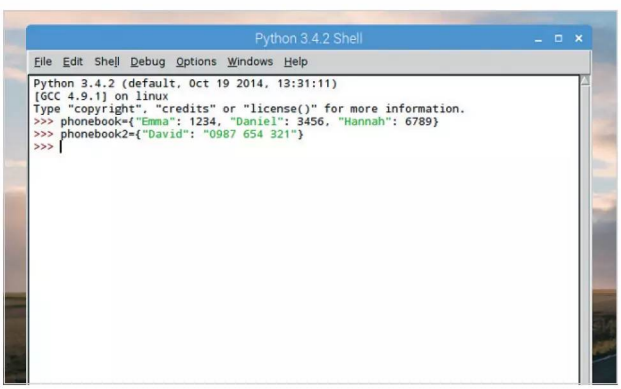
```
phonebook={“Emma”: 1234, “Daniel”: 3456, “Hannah”: 6789}
```



SCHRITT 2

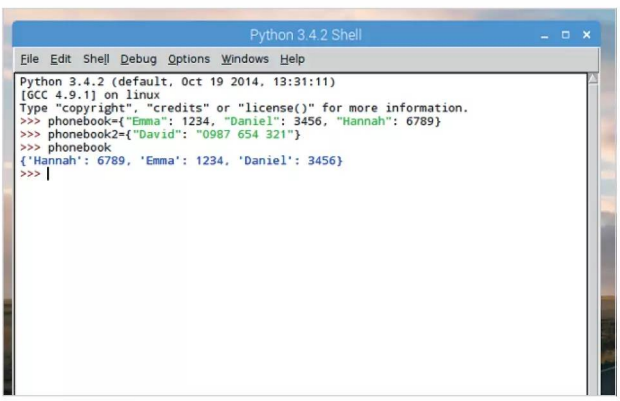
Wie bei den meisten Listen, Tupeln usw. müssen Strings in Anführungszeichen (einfach oder doppelt) eingeschlossen werden, während Ganzzahlen offen bleiben können. Der Wert kann entweder ein String oder eine ganze Zahl sein:

```
phonebook2={“David”: “0987 654 321”}
```



SCHRITT 3

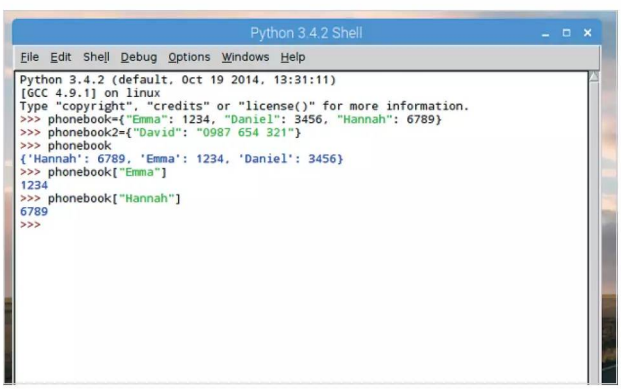
Wie auch bei Listen und Tupeln können Sie den Inhalt eines Dictionaries überprüfen, indem Sie ihm einen Namen geben, in diesem Fall nennen wir es phonebook. Dadurch werden die eingegebenen Datenelemente auf ähnliche Weise wie in einer Liste angezeigt, mit der Sie nun zweifellos vertraut sind.



SCHRITT 4

Der Vorteil eines Dictionaries besteht darin, dass Sie den Schlüssel eingeben können, um den Wert zu indizieren. Nehmen Sie das Telefonbuchbeispiel aus den vorherigen Schritten und geben Sie Folgendes ein:

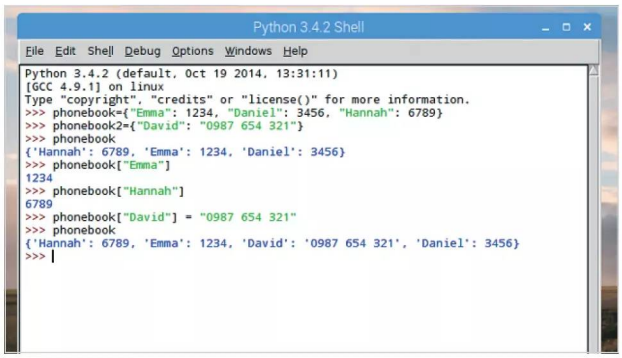
```
phonebook[“Emma”]
phonebook[“Hannah”]
```



SCHRITT 5

Das Hinzufügen eines neuen Datenelements zu einem Dictionary ist ebenfalls einfach. Geben Sie dazu die neuen Schlüssel- und Wertelemente ein:

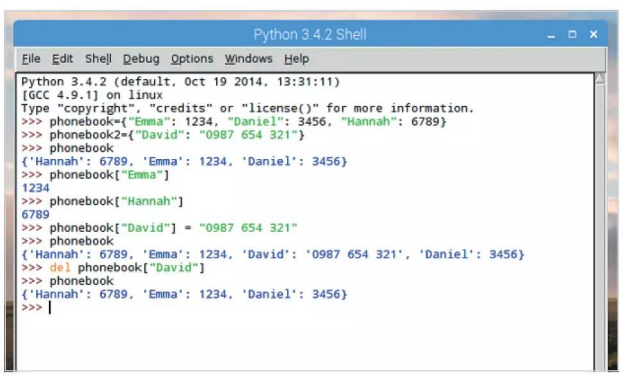
```
phonebook[“David”] = “0987 654 321”
phonebook
```



SCHRITT 6

Sie können Elemente auch aus einem Dictionary entfernen, indem Sie den Befehl del, gefolgt vom Schlüssel des Elements, eingeben. Der Wert wird ebenfalls entfernt, da beide Datenelemente als Paar agieren:

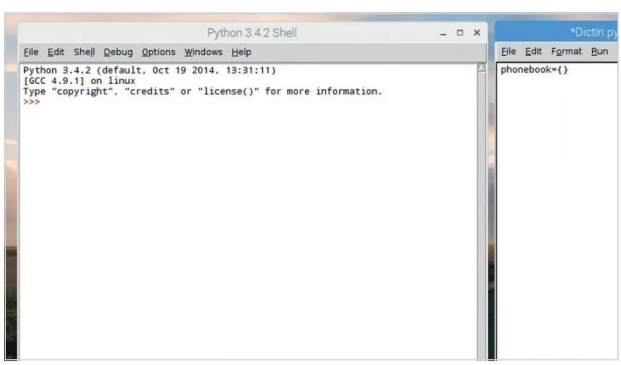
```
del phonebook[“David”]
```



SCHRITT 7

Wie wäre es mit einem Code, der den Benutzer nach dem Schlüssel und dem Wert des Dictionaries fragt? Öffnen Sie eine neue Editor-Datei und beginnen Sie mit der Programmierung in einem neuen, leeren Dictionary:

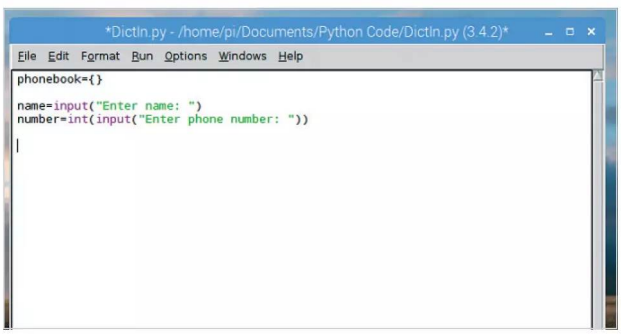
```
phonebook={}
```



SCHRITT 8

Sie müssen nun die Benutzereingaben und -variablen definieren: eine für den Namen der Person, die andere für deren Telefonnummer (wir halten es einfach, um langwierigen Python-Code zu vermeiden):

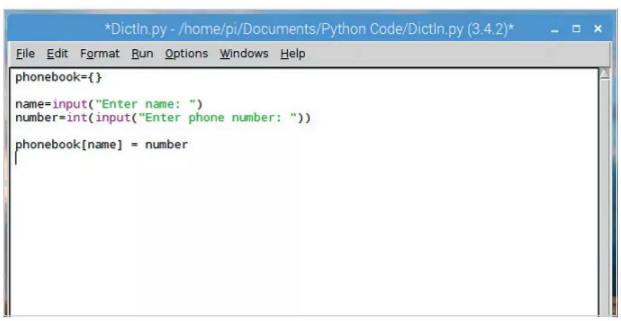
```
name=input(“Enter name: “)
number=int(input(“Enter phone number: “))
```



SCHRITT 9

Beachten Sie, dass wir die Zahl als Ganzzahl anstelle eines Strings beibehalten haben, obwohl der Wert sowohl eine Ganzzahl als auch ein String sein kann. Sie müssen nun die vom Benutzer eingegebenen Variablen zum neu erstellten leeren Dictionary hinzufügen. Wir benutzen dasselbe Verfahren wie in Schritt 5 und geben Folgendes ein:

```
phonebook[name] = number
```

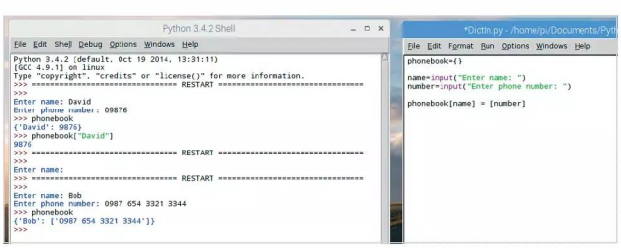


SCHRITT 10

Wenn Sie nun den Code speichern und ausführen, wird Python nach einem Namen und einer Nummer fragen. Es wird diese Einträge dann in das Telefonbuchverzeichnis eintragen. Sie können dies testen, indem Sie Folgendes in die Shell eingeben:

```
phonebook
phonebook[“David”]
```

Wenn die Zahl Leerzeichen enthalten muss, müssen Sie einen String erstellen. Entfernen Sie daher den int-Teil der Eingabe.



Strings trennen und zusammenfügen

Beim Umgang mit Daten in Python, insbesondere bei Eingaben von Benutzern, werden Sie zweifellos auf lange Strings stoßen. Eine nützliche Fähigkeit in der Python-Programmierung ist das Trennen dieser langen Strings für eine bessere Lesbarkeit.

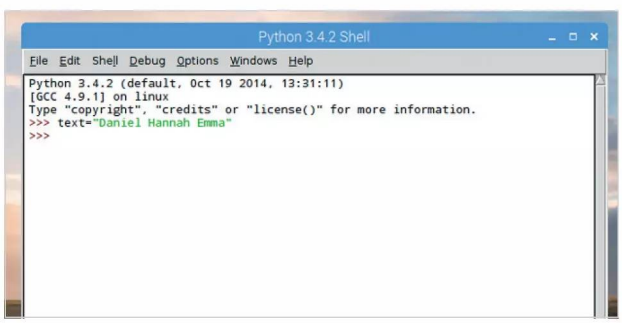
STRINGTHEORIEN

Sie haben bereits einige Listenfunktionen mit .insert, .remove und .pop kennengelernt, aber es gibt auch Funktionen, die mit Strings angewendet werden können.

SCHRITT 1

Das Hauptwerkzeug in den String-Funktionen ist .split(). Damit können Sie eine Reihe von Daten basierend auf dem Argument innerhalb der Klammern trennen. Hier haben wir z. B. einen String mit drei Elementen, die jeweils durch ein Leerzeichen getrennt sind:

```
text=”Daniel Hannah Emma”
```

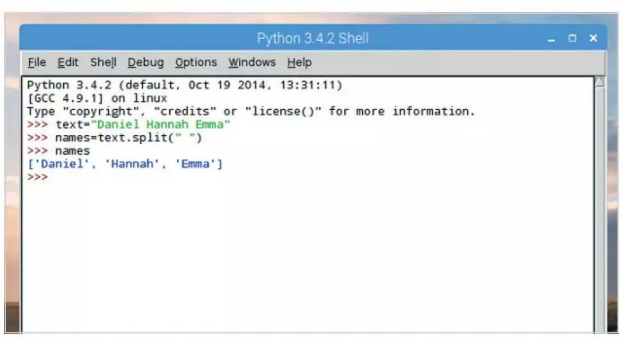


SCHRITT 2

Wir wandeln nun den String in eine Liste um und trennen den Inhalt entsprechend:

```
names=text.split(“ “)
```

Geben Sie dann den Namen der neuen Liste ein, names, um die drei Elemente zu sehen.



SCHRITT 3

Beachten Sie, dass der text.split-Teil die Klammern, Anführungszeichen und ein Leerzeichen gefolgt von schließenden Anführungszeichen und Klammern hat. Das Leerzeichen ist das Trennzeichen, es bedeutet, dass jeder Listeneintrag durch ein Leerzeichen getrennt ist. Ebenso hat CSV (Comma Separated Value) ein Komma, verwenden Sie daher:

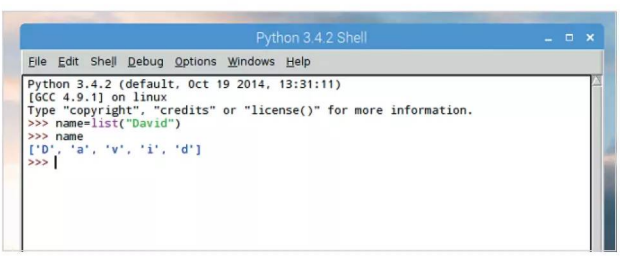
```
text=”January,February,March,April,May,June”
months=text.split(“,”)
months
```

SCHRITT 4

Wie Sie bereits gesehen haben, können Sie mithilfe eines Namens einen String in eine aus einzelnen Buchstaben bestehende Liste aufteilen:

```
name=list(“David”)
name
```

Der zurückgegebene Wert ist ‘D’, ‘a’, ‘v’, ‘i’, ‘d’. Während es unter normalen Umständen etwas nutzlos erscheinen mag, könnte es zum Beispiel für die Erstellung eines Buchstabierspiels nützlich sein.

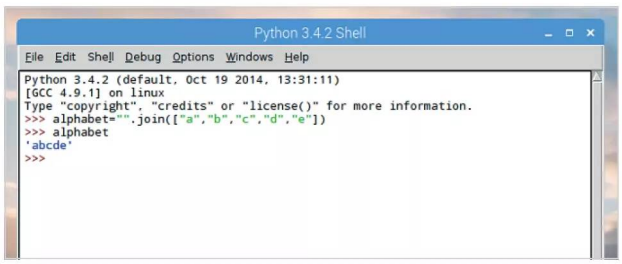


SCHRITT 5

Das Gegenteil der .split-Funktion ist .join, wo Sie separate Elemente in einem String zusammenfügen können, um ein Wort oder eine Kombination von Elementen zu bilden, abhängig von dem Programm, das Sie schreiben. Zum Beispiel:

```
alphabet="".join(["a","b","c","d","e"])
alphabet
```

Dies zeigt 'abcde' in der Shell an.

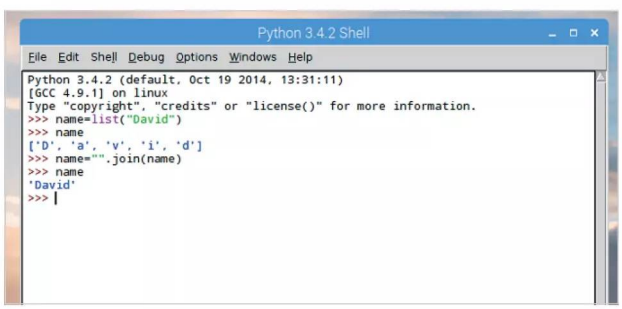


SCHRITT 6

Sie können daher .join für den in Schritt 4 erstellten getrennten Namen anwenden, indem Sie die Buchstaben erneut kombinieren, um den Namen zu bilden:

```
name="".join(name)
name
```

Wir haben den String wieder zusammengefügt und die Liste namens name mithilfe der .join-Funktion beibehalten.

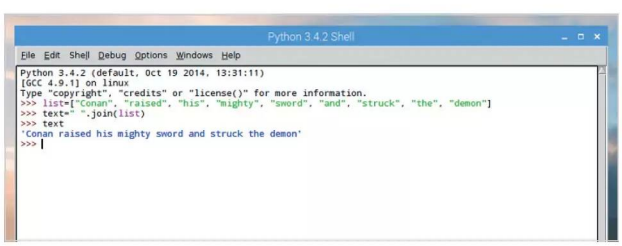


SCHRITT 7

Ein gutes Beispiel für die Verwendung der .join-Funktion ist das Kombinieren einer Liste mit Wörtern zu einem Satz:

```
list=["Conan", "raised", "his", "mighty", "sword",
"and", "struck", "the", "demon"]
text=" ".join(list)
text
```

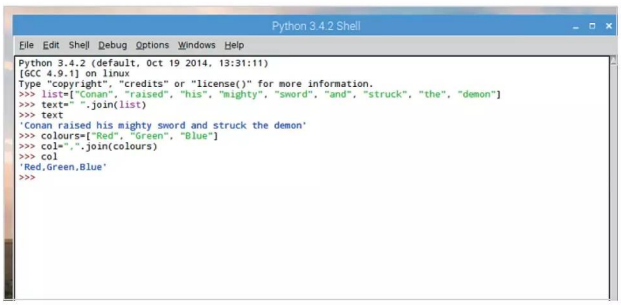
Beachten Sie den Leerschritt zwischen den Anführungszeichen vor der .join-Funktion (in Schritt 6 enthielten diese keinen Leerschritt).



SCHRITT 8

Wie bei der .split-Funktion muss das Trennzeichen kein Leerzeichen sein, es kann auch ein Komma sein, ein Punkt, ein Bindestrich oder was immer Sie bevorzugen:

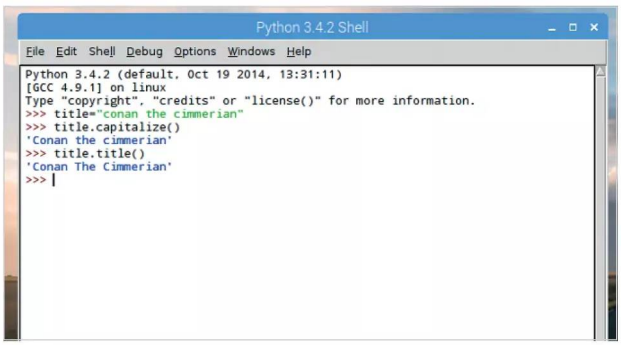
```
colours=["Red", "Green", "Blue"]
col=",".join(colours)
col
```



SCHRITT 9

Es gibt einige interessante Funktionen für Strings wie .capitalize und .title. Hier ein Beispiel:

```
title="conan the cimmerian"
title.capitalize()
title.title()
```



SCHRITT 10

Sie können auch logische Operatoren für Strings mit den Funktionen „in" und „not in" verwenden. Damit können Sie prüfen, ob ein String eine Zeichenfolge enthält (oder nicht):

```
message="Have a nice day"
"nice" in message

"bad" not in message
"day" not in message
"night" in message
```

Strings formatieren

In Python 3 hat sich die Formatierung von Strings durch die Kombination der .format-Funktion mit geschweiften Klammern stark verbessert. Dieser Ansatz ist sinnvoller und besser als in früheren Versionen.

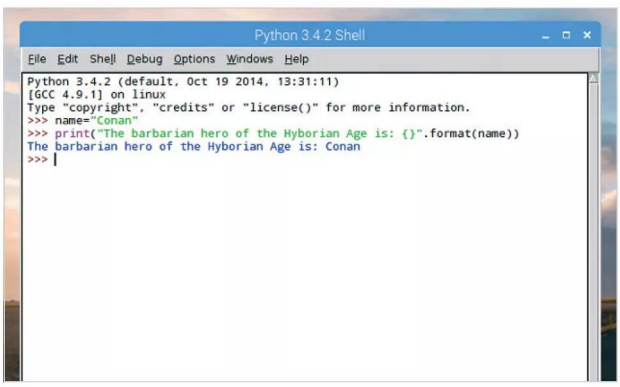
STRING-FORMATIERUNG

Seit Python 3 hat sich die Formatierung von Strings durch die Kombination der .format-Funktion mit geschweiften Klammern stark verbessert. Dies ist ein logischer und besserer Ansatz als in früheren Versionen.

SCHRITT 1

Die Grundformatierung in Python besteht darin, jede Variable mithilfe der geschweiften Klammern in den String zu integrieren:

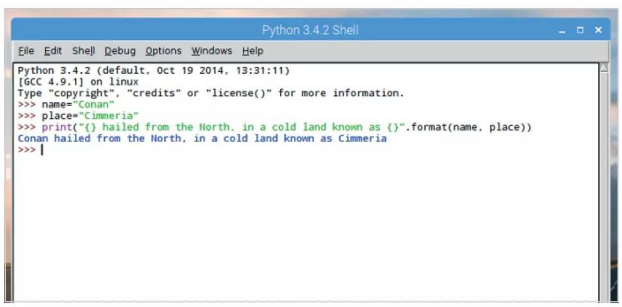
```
name=”Conan”
print(“The barbarian hero of the Hyborian Age is: {}”.format(name))
```



SCHRITT 2

Denken Sie daran, die print-Funktion mit zwei Klammern zu schließen, da Sie die Variable und die print-Funktion in ihre eigenen Klammern eingeschlossen haben. Sie können mehrere Fälle der String-Formatierung in eine einzelne print-Funktion einschließen:

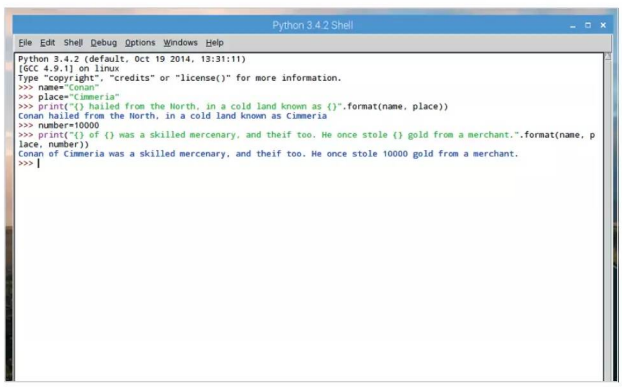
```
name=”Conan”
place=”Cimmeria”
print(“{} hailed from the North, in a cold land known as {}”.format(name, place))
```



SCHRITT 3

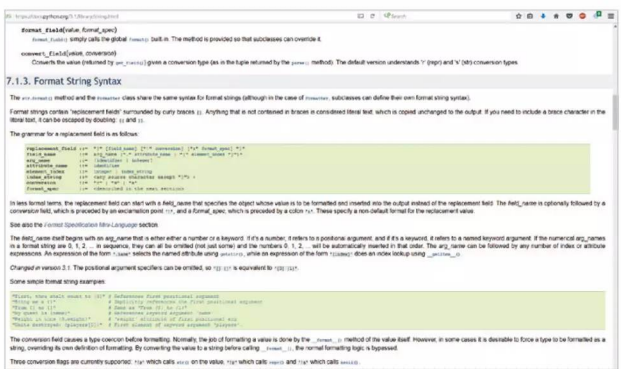
Sie können natürlich auch ganze Zahlen in den Mix aufnehmen:

```
number=10000
print(“{} of {} was a skilled mercenary, and thief too. He once stole {} gold from a merchant.”.format(name, place, number))
```



SCHRITT 4

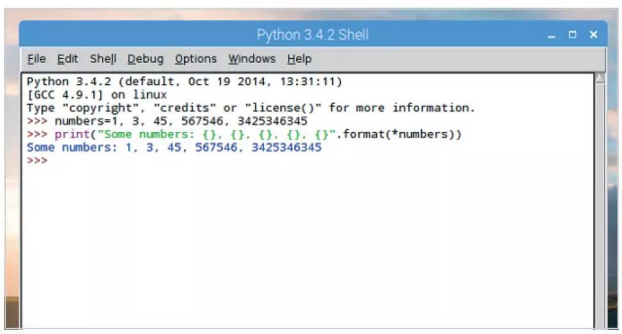
Es gibt viele verschiedene Möglichkeiten zur Formatierung von Strings. Einige sind ziemlich einfach, wie wir Ihnen hier gezeigt haben, andere hingegen können recht komplex sein, es hängt halt davon ab, was Sie von Ihrem Programm erwarten. Auf der Python Docs-Webseite **www.docs.python.org/3.1/library/string.html** finden Sie reichlich Hilfe zur String-Formatierung.



SCHRITT 5

Interessanterweise können Sie eine Liste mit der String-Formatierungsfunktion referenzieren. Setzen Sie dazu ein Sternchen vor den Listennamen:

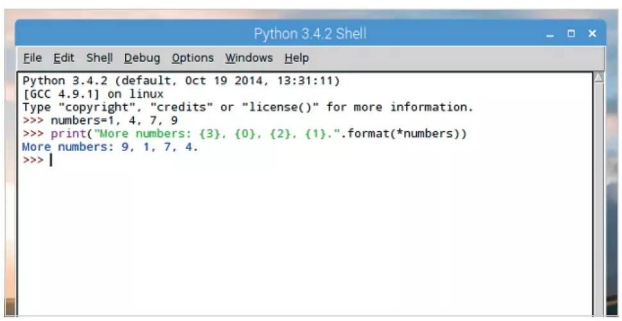
```
numbers=1, 3, 45, 567546, 3425346345
print("Some numbers: {}, {}, {}, {}, {}".
format(*numbers))
```



SCHRITT 6

Bei der Indexierung in Listen gilt für den Aufruf einer Liste mit String-Formatierung das Gleiche. Sie können jedes Element nach seiner Position indizieren (von 0 bis zur letzten Zeile):

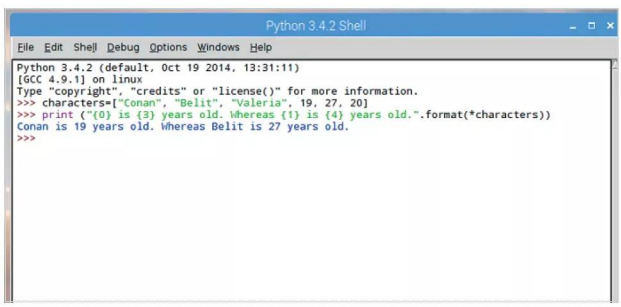
```
numbers=1, 4, 7, 9
print("More numbers: {3}, {0}, {2},
{1}.".format(*numbers))
```



SCHRITT 7

Und wie Sie wahrscheinlich vermuten, können Sie Strings und Integer in einer einzigen Liste mischen, die mit der .format-Funktion aufgerufen wird:

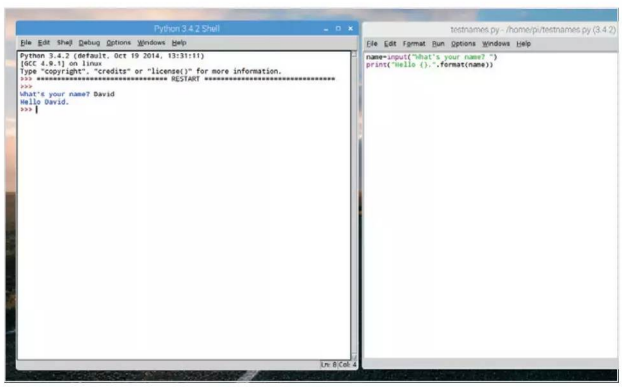
```
characters=["Conan", "Belit", "Valeria", 19, 27,
20]
print ("{0} is {3} years old. Whereas {1} is {4}
years old.".format(*characters))
```



SCHRITT 8

Sie können den Inhalt einer Benutzereingabe auf dieselbe Weise ausgeben:

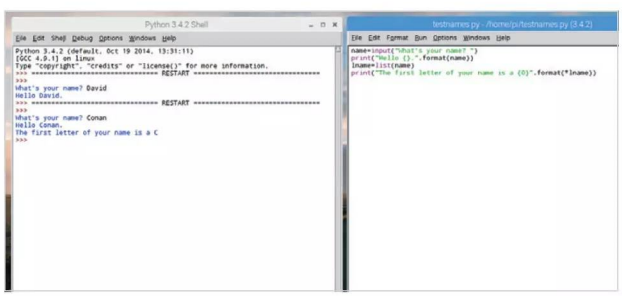
```
name=input("What's your name? ")
print("Hello {}.".format(name)
```



SCHRITT 9

Sie können dieses einfache Codebeispiel erweitern, sodass der erste Buchstabe des eingegebenen Namens angezeigt wird:

```
name=input("What's your name? ")
print("Hello {}.".format(name))
lname=list(name)
print("The first letter of your name is a {0}".
format(*lname))
```



SCHRITT 10

Sie können auch ein Listenpaar aufrufen und diese innerhalb derselben print-Funktion einzeln referenzieren. Ändern Sie den in Schritt 7 erstellten Code und erstellen Sie zwei Listen:

```
names=["Conan", "Belit", "Valeria"]
ages=[25, 21, 22]
```

Sie können nun jede Liste sowie einzelne Elemente aufrufen:

```
print("{0[0]} is {1[0]} years old. Whereas {0[1]}
is {1[1]} years old.".format(names, ages))
```

Datum und Uhrzeit

Bei der Arbeit mit Daten ist es oft nützlich, auf die Uhrzeit zugreifen zu können, zum Beispiel um einen Eintrag mit einem Zeitstempel zu versehen. Zum Glück ist es dank des Zeitmoduls einfach, Datum und Uhrzeit zu erfassen.

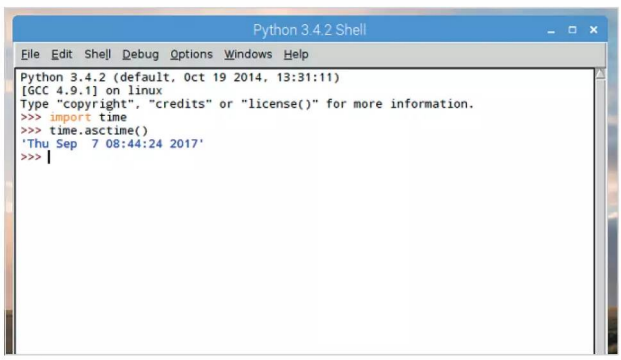
HERRSCHER DER ZEIT

Das Zeitmodul enthält Funktionen, die Ihnen helfen, die aktuelle Systemzeit abzurufen, das Datum von Strings einzulesen, Uhrzeit und Datum zu formatieren und vieles mehr.

SCHRITT 1

Zunächst müssen Sie das Zeitmodul importieren. Es ist in Python 3 integriert, sodass Sie es nicht erst über die Eingabeaufforderung und dem Pip-Befehl installieren müssen. Nach dem Import können Sie die aktuelle Uhrzeit und das Datum mit einem einfachen Befehl aufrufen:

```
import time
time.asctime()
```



SCHRITT 2

Die Zeitfunktion ist in neun Tupel unterteilt, die wie bei jedem anderen Tupel in indizierte Elemente aufgeteilt und im folgenden Screenshot zu sehen sind.

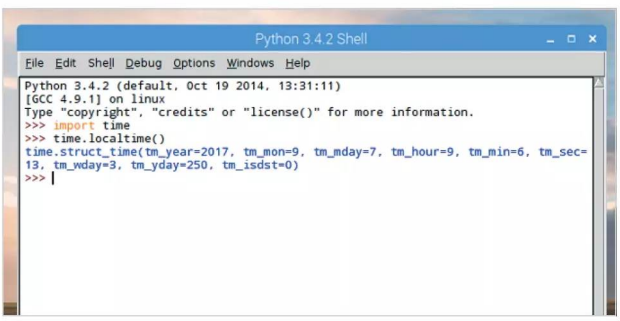
Index	Field	Values
0	4-digit year	2016
1	Month	1 to 12
2	Day	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 61 (60 or 61 are leap-seconds)
6	Day of Week	0 to 6 (0 is Monday)
7	Day of year	1 to 366 (Julian day)
8	Daylight savings	-1, 0, 1, -1 means library determines DST

SCHRITT 3

Sie können sich über die folgende Eingabe die Struktur der Zeitdarstellung anzeigen lassen:

```
time.local.time()
```

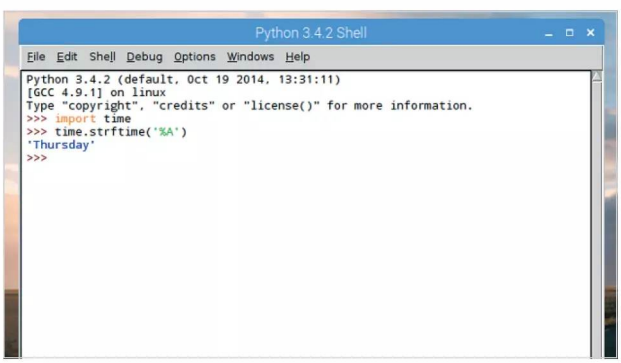
Die Ausgabe wird folgendermaßen wiedergegeben: `'time.struct_time(tm_year=2017, tm_mon=9, tm_mday=7, tm_hour=9, tm_min=6, tm_sec=13, tm_wday=3, tm_yday=250, tm_isdst=0)'` (die Ausgabewerte hängen natürlich von Ihrer aktuellen Uhrzeit ab).



SCHRITT 4

Im Zeitmodul sind zahlreiche Funktionen integriert. Eine der gebräuchlichsten ist .strftime(). Damit können Sie eine Vielzahl von Argumenten darstellen, während das Zeittupel in einen String umgewandelt wird. Um z. B. den aktuellen Wochentag anzuzeigen, geben Sie Folgendes ein:

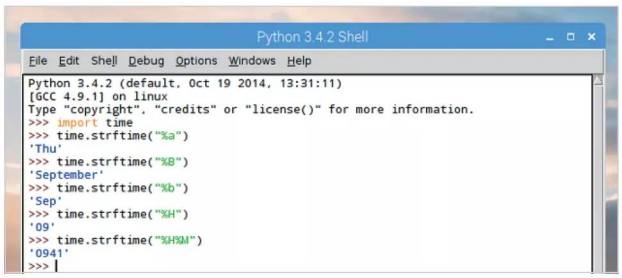
```
time.strftime('%A')
```



SCHRITT 5

Dies bedeutet natürlich, dass Sie verschiedene Funktionen in Ihren eigenen Code integrieren können, wie zum Beispiel:

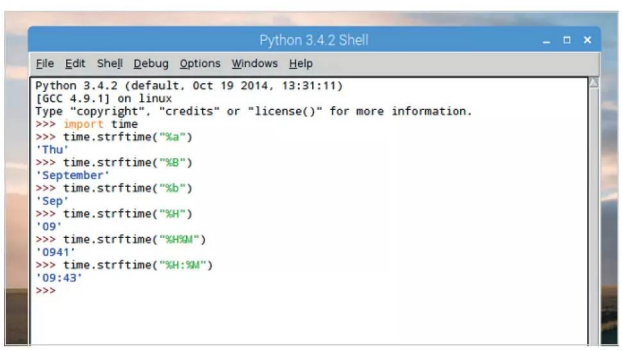
```
time.strftime("%a")
time.strftime("%B")
time.strftime("%b")
time.strftime("%H")
time.strftime("%H%M")
```



SCHRITT 6

Beachten Sie die letzten beiden Einträge mit %H und %H%M, dies sind die Stunden und Minuten. Wie wir dem letzten Eintrag entnehmen können, zeigt die Eingabe %H%M die Uhrzeit in der Shell nicht korrekt an. Sie können dies mit der folgenden Eingabe leicht korrigieren:

```
time.strftime("%H:%M")
```

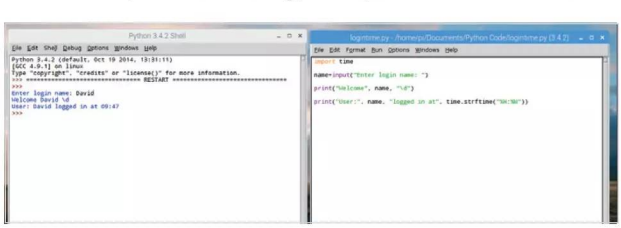


SCHRITT 7

Dies bedeutet, dass Sie entweder die aktuelle Uhrzeit oder die Uhrzeit anzeigen können, zu der etwas aufgetreten ist, z. B. wenn ein Benutzer seinen Namen eingegeben hat. Probieren Sie Folgendes im Editor aus:

```
import time
name=input("Enter login name: ")
print("Welcome", name, "\d")
print("User:", name, "logged in at", time.
strftime("%H:%M"))
```

Versuchen Sie, den Code um Tag, Monat, Jahr usw. zu erweitern.



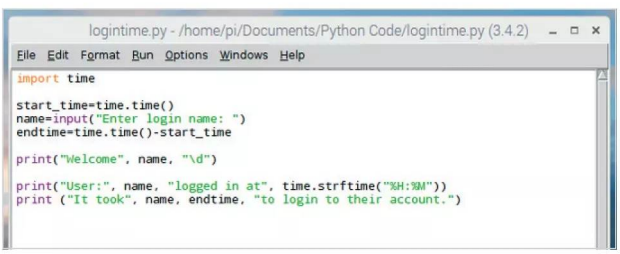
SCHRITT 8

Sie haben am Ende des vorherigen Abschnitts gesehen, dass Sie im Code zur Berechnung von Pi mit der vom Benutzer gewünschten Anzahl der Dezimalstellen, ein bestimmtes Ereignis in Python ablaufen lassen können. Nehmen Sie den oberen Code und ändern Sie ihn wie folgt:

```
start_time=time.time()
```

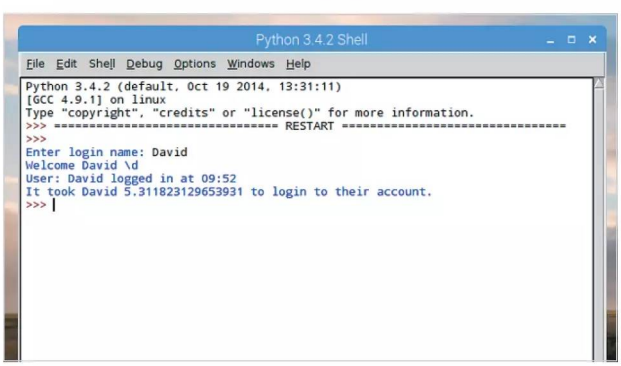
Dann haben wir auch noch folgende Eingabe:

```
endtime=time.time()-start_time
```



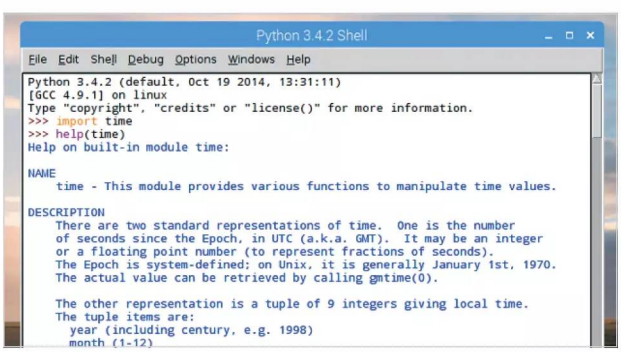
SCHRITT 9

Die Ausgabe ähnelt derer im unteren Screenshot. Die Timer-Funktion muss auf beiden Seiten der Eingabeanweisung stehen, da hier der Name der Variablen erstellt wird, abhängig von der Dauer, die der Benutzer zum Anmelden benötigt. Die Dauer wird dann in der letzten Zeile des Codes als die Variable `endtime` ausgegeben.



SCHRITT 10

Es gibt vieles, das mit dem Zeitmodul getan werden kann; einiges davon ist auch recht komplex (z. B. die Anzahl der Sekunden seit dem 1. Januar 1970 zu ermitteln). Wenn Sie tiefer in das Zeitmodul eintauchen möchten, geben Sie in der Shell `help(time)` ein, um die Hilfedatei der aktuellen Python-Version für das Zeitmodul anzuzeigen.



Dateien öffnen

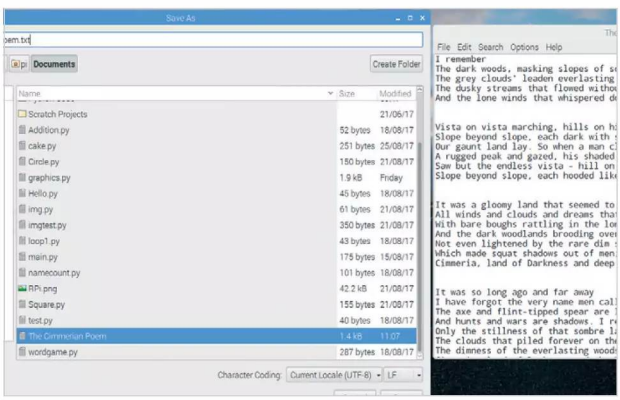
In Python können Sie Text- und Binärdateien in Ihre Programmen einlesen. Sie können Dateien auch schreiben, womit wir uns als Nächstes befassen werden. Das Lesen und Schreiben von Dateien ermöglicht die Ausgabe von Daten aus Ihren Programmen.

ÖFFNEN, LESEN UND SCHREIBEN

In Python erstellen Sie ein Dateiobjekt auf ähnliche Weise wie eine Variable, allerdings wird die Datei mit der open()-Funktion als Variable eingegeben. Dateien werden normalerweise als Text oder Binär kategorisiert.

SCHRITT 1

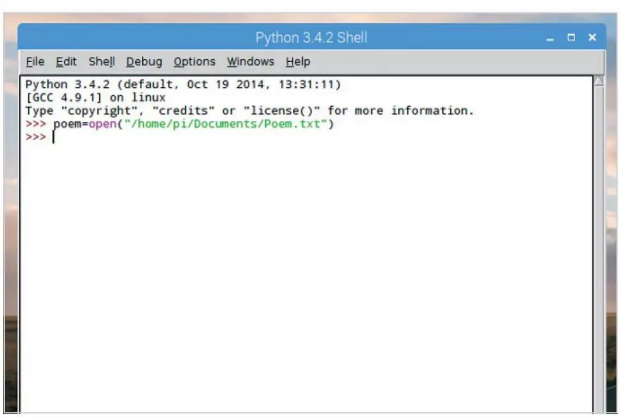
Beginnen Sie mit der Eingabe eines Textes in den Texteditor Ihres Systems. Der Texteditor ist besser geeignet als Textverarbeitungsprogramme, da diese Hintergrundformatierungen und andere Elemente enthalten. Für unser Beispiel nehmen wir das Gedicht The Cimmerian von Robert E. Howard. Sie müssen die Datei als poem.txt speichern.



SCHRITT 2

Mit der open()-Funktion wird die Datei als Objekt in eine Variable eingegeben. Sie können das Dateiobjekt beliebig benennen, müssen Python aber den Namen und den Speicherort der Textdatei mitteilen, die Sie öffnen:

```
poem=open("/home/pi/Documents/Poem.txt")
```

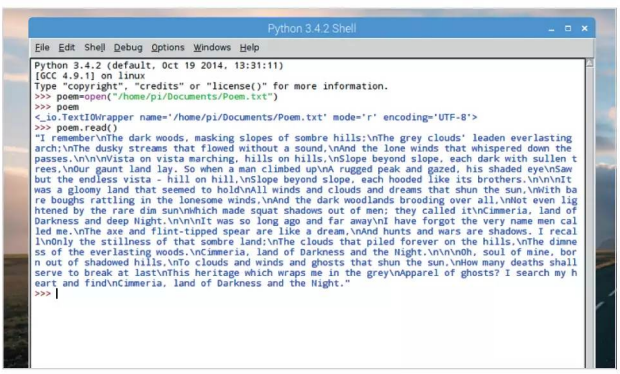


SCHRITT 3

Wenn Sie nun poem in die Shell eingeben, erhalten Sie einige Informationen zur Textdatei, die Sie geöffnet haben möchten. Sie können nun mit der Variablen „poem" den Inhalt der Datei einlesen:

```
poem.read()
```

Beachten Sie, dass ein /n-Eintrag im Text eine neue Zeile darstellt.

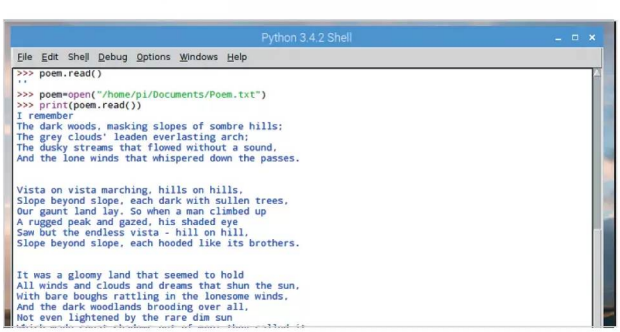


SCHRITT 4

Wenn Sie ein zweites Mal poem.read() eingeben, werden Sie feststellen, dass der Text aus der Datei entfernt wurde. Sie müssen erneut `poem=open("/home/pi/Documents/Poem.txt")` eingeben, um die Datei neu zu erstellen. Geben Sie nun jedoch Folgendes ein:

```
print(poem.read())
```

Diesmal werden die /n-Einträge zugunsten neuer Zeilen und lesbarerem Text entfernt.



SCHRITT 5

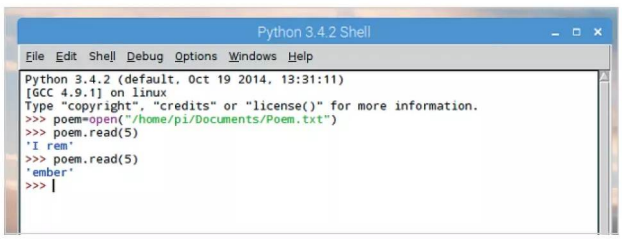
Genau wie bei Listen, Tupeln, Dictionaries usw. können Sie einzelne Zeichen des Textes indizieren, zum Beispiel:

```
poem.read(5)
```

Dies zeigt die ersten fünf Zeichen an, während Folgendes:

```
poem.read(5)
```

die nächsten fünf anzeigt. Durch die Eingabe von (1) wird jeweils ein Zeichen angezeigt.



SCHRITT 6

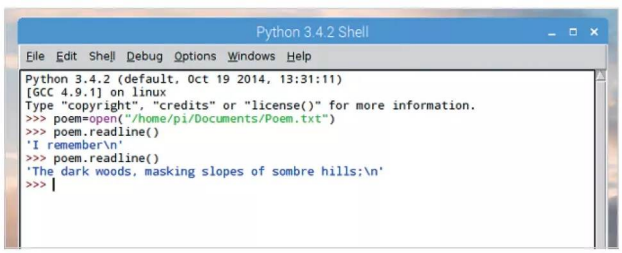
Auf ähnliche Weise können Sie mit der readline()-Funktion jeweils eine Textzeile anzeigen. Zum Beispiel:

```
poem=open("/home/pi/Documents/Poem.txt")
poem.readline()
```

Dies zeigt die erste Zeile des Textes an, während:

```
poem.readline()
```

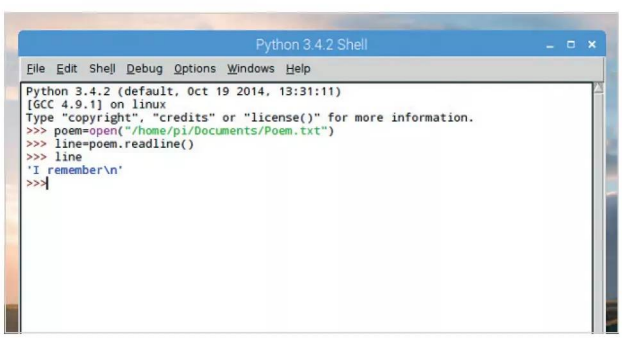
die nächste Zeile anzeigt.



SCHRITT 7

Sie haben vielleicht geahnt, dass Sie die Funktion readline() in eine Variable übernehmen können, sodass Sie sie bei Bedarf erneut aufrufen können:

```
poem=open("/home/pi/Documents/Poem.txt")
line=poem.readline()
line
```



SCHRITT 8

Sie können mit readlines() alle Zeilen des Textes erfassen und als mehrere Listen speichern. Diese können dann als Variable gespeichert werden:

```
poem=open("/home/pi/Documents/Poem.txt")
lines=poem.readlines()
lines[0]
lines[1]
lines[2]
```

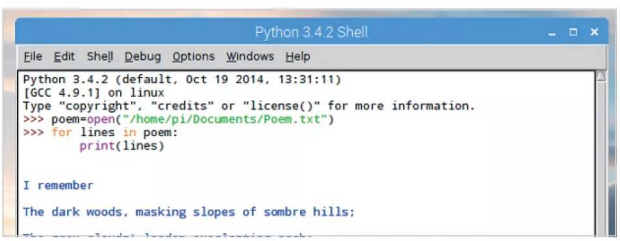
SCHRITT 9

Sie können die Textzeilen auch mit der for-Anweisung ausgeben:

```
for lines in lines:
   print(lines)
```

Da es sich hier um Python handelt, gibt es auch andere Möglichkeiten, mit denen wir die gleiche Ausgabe erhalten:

```
poem=open("/home/pi/Documents/Poem.txt")
for lines in poem:
   print(lines)
```

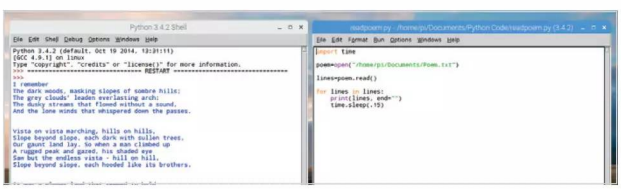


SCHRITT 10

Nehmen wir an, dass wir den Text Zeichen für Zeichen ausgeben möchten. Sie können das Zeitmodul mit dem kombinieren, was Sie sich hier bereits angeschaut haben. Probieren Sie Folgendes aus:

```
import time
poem=open("/home/pi/Documents/Poem.txt")
lines=poem.read()
for lines in lines:
  print(lines, end="")
  time.sleep(.15)
```

Die Ausgabe kann leicht in Ihren eigenen Code integriert werden.



In Dateien schreiben

Externe Dateien in Python einlesen zu können, ist sicherlich praktisch, aber in eine Datei schreiben zu können ist besser. Mit write() können Sie die Ergebnisse eines Programms in einer Datei ausgeben und diese anschließend mit der read()-Funktion in Python einlesen.

WRITE() UND CLOSE()

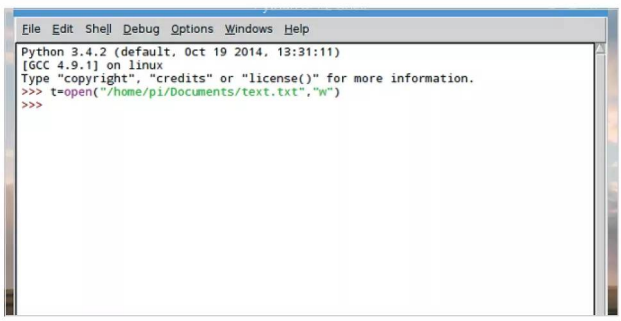
Die write()-Funktion ist etwas komplexer als read(). Neben dem Dateinamen müssen Sie auch einen Zugriffsmodus angeben, der festlegt, ob sich die betreffende Datei im Schreib- oder Lesemodus, also im Write- oder Read-Modus, befindet.

SCHRITT 1

Öffnen Sie die IDLE und geben Sie Folgendes ein:

```
t=open("/home/pi/Documents/text.txt","w")
```

Ändern Sie den Pfad /home/pi/Documents in Ihren eigenen Systempfad. Dieser Code erstellt im Schreibmodus eine Textdatei namens text.txt mit der Variablen „t". Sofern sich im Speicherort keine Datei mit diesem Namen befindet, wird eine Datei erstellt. Sollte bereits eine Datei existieren, wird sie überschrieben; seien Sie daher vorsichtig.

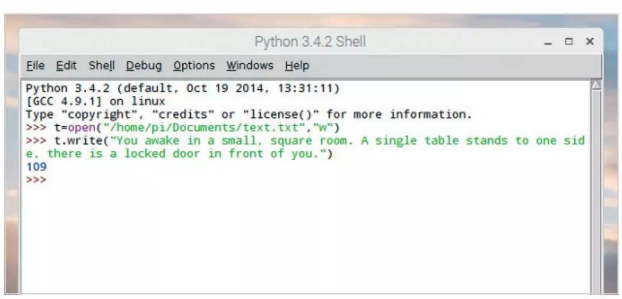


SCHRITT 2

Sie können nun mit der write()-Funktion in die Textdatei schreiben. Dies funktioniert entgegengesetzt zu read(), d. h. es werden Zeilen geschrieben anstatt eingelesen. Probieren Sie Folgendes aus:

```
t.write("You awake in a small, square room. A single table stands to one side, there is a locked door in front of you.")
```

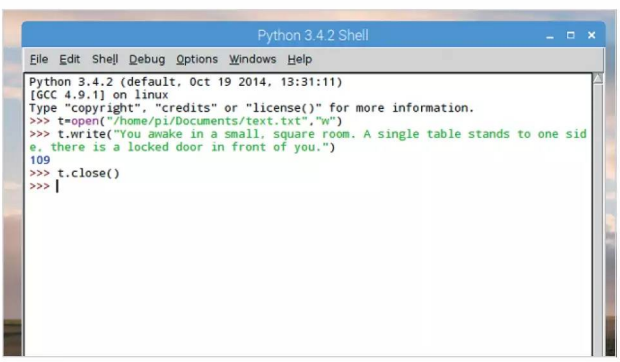
Beachten Sie die Ausgabe „109". Sie ist die Anzahl der eingegebenen Zeichen.



SCHRITT 3

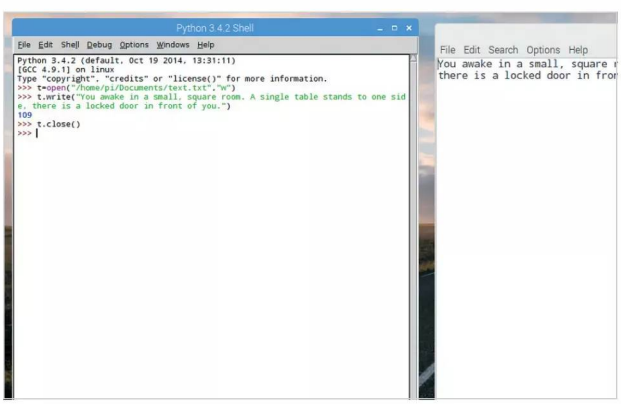
Die eigentliche Textdatei ist jedoch immer noch leer (Sie können dies überprüfen, indem Sie sie öffnen). Das liegt daran, dass Sie die Textzeile in das Dateiobjekt, aber nicht in die Datei selbst geschrieben haben. Ein Teil der write()-Funktion besteht darin, dass Sie die Änderungen in die Datei übertragen müssen. Geben Sie dazu Folgendes ein:

```
t.close()
```



SCHRITT 4

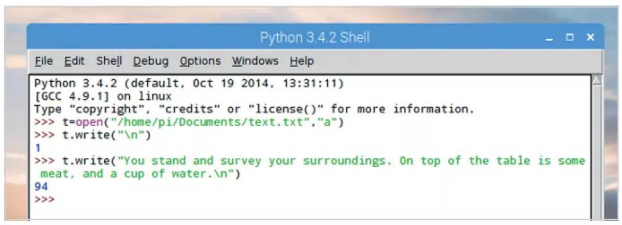
Wenn Sie die Textdatei nun mit einem Texteditor öffnen, sehen Sie, dass die von Ihnen erstellte Zeile in die Datei geschrieben wurde. Dies bietet die Grundlage für einige interessante Möglichkeiten, z. B. der Erstellung Ihrer eigenen Log-Datei oder sogar die ersten Ansätze eines Abenteuerspiels.



SCHRITT 5

Um diesen Code zu erweitern, öffnen Sie mit „a“ für die access- bzw. append-Funktion erneut die Datei. Anstatt die Datei zu löschen und eine neue zu erstellen, wird dadurch jeglicher Text ans Ende der ursprünglichen Zeile hinzugefügt, z. B.:

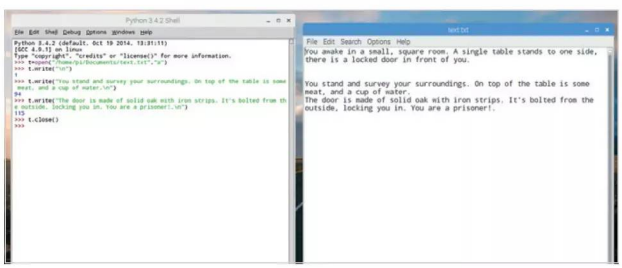
```
t=open(“/home/pi/Documents/text.txt”,”a”)
t.write(“\n”)
t.write(“ You stand and survey your surroundings. On top of the table is some meat, and a cup of water.\n”)
```



SCHRITT 6

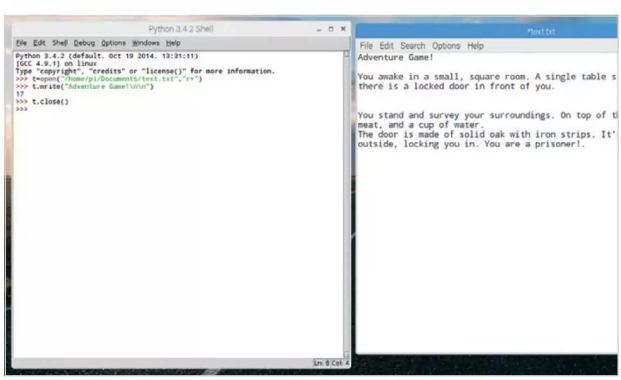
Sie können den Text zeilenweise erweitern und mit einer neuen Zeile (\n) abschließen. Wenn Sie fertig sind, beenden Sie den Code mit t.close() und öffnen Sie die Datei in einem Texteditor, um die Ergebnisse anzuzeigen:

```
t.write(“The door is made of solid oak with iron strips. It’s bolted from the outside, locking you in. You are a prisoner!.\n”)
t.close()
```



SCHRITT 7

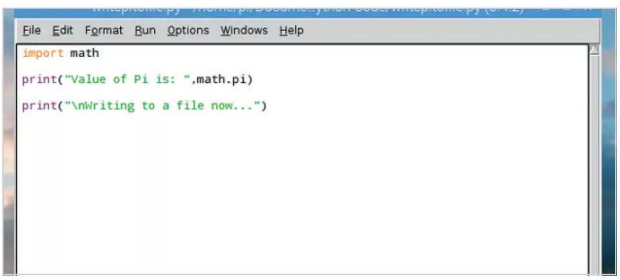
Es gibt verschiedene Arten für den Dateizugriff mit der open()-Funktion. Jede hängt davon ab, wie auf die Datei zugegriffen wird sowie von der Position des Cursors. Mit z. B. r+ wird eine Datei im Lese- und Schreibmodus geöffnet und der Cursor an den Anfang der Datei platziert.



SCHRITT 8

Sie können auch Variablen an in Python erstellte Dateien übertragen. Nehmen wir an, wir möchten den Wert von Pi in eine Datei schreiben. Wir rufen dazu Pi über das math-Modul auf, erstellen eine neue Datei und übertragen die Ausgabe von Pi in die neue Datei:

```
import math
print(“Value of Pi is: “,math.pi)
print(“\nWriting to a file now…”)
```



SCHRITT 9

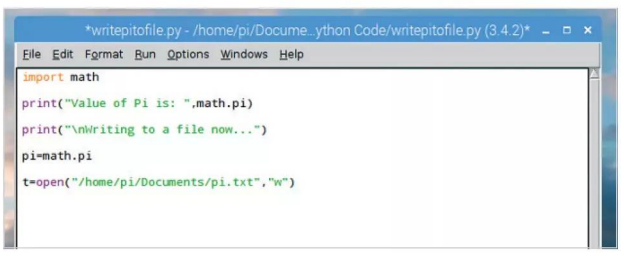
Wir erstellen nun eine Variable namens pi und weisen ihr den Wert von Pi zu:

```
pi=math.pi
```

Wir müssen auch eine neue Datei erstellen, in die Pi geschrieben werden soll:

```
t=open(“/home/pi/Documents/pi.txt”,”w”)
```

Denken Sie daran, den Dateipfad gegen Ihren eigenen auszutauschen.

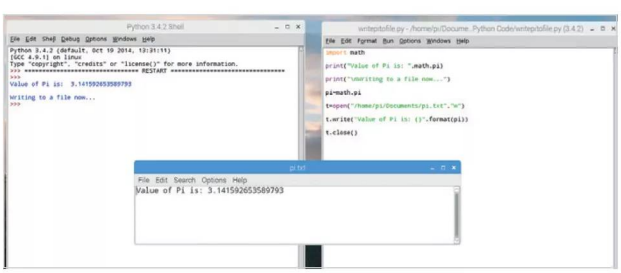


SCHRITT 10

Abschließend können wir die Variable mit einer String-Formatierung aufrufen und in die Datei schreiben, dann die Änderungen übernehmen und die Datei schließen:

```
t.write(“Value of Pi is: {}”.format(pi))
t.close()
```

Anhand der Ergebnisse lässt sich sehen, dass jede Variable an eine Datei übergeben werden kann.



Ausnahmen

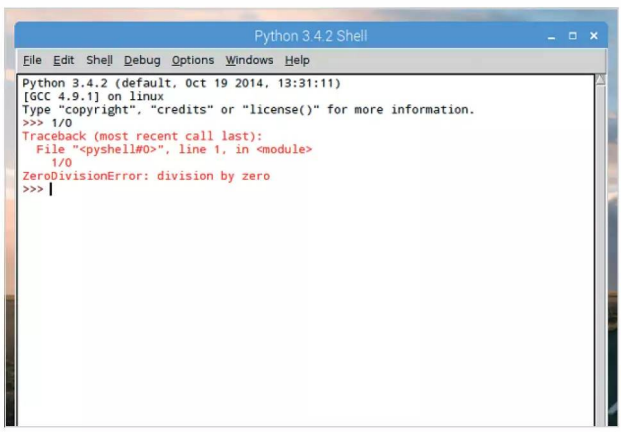
Beim Programmieren werden Sie auf Probleme stoßen, die außerhalb Ihrer Kontrolle liegen. Nehmen wir an, Sie bitten einen Benutzer, zwei Zahlen zu teilen und er versucht, durch Null zu teilen. Dies wird einen Fehler verursachen und Ihren Code unterbrechen.

AUSNAHMEOBJEKTE

Anstatt den Fluss Ihres Codes zu stoppen, enthält Python Ausnahmeobjekte, die unerwartete Fehler im Code behandeln. Sie können Fehler beheben, indem Sie Bedingungen erstellen, bei denen Ausnahmen auftreten können.

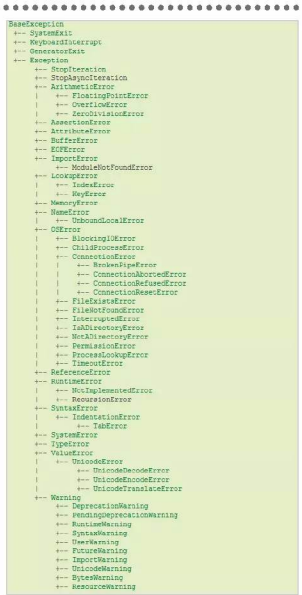
SCHRITT 1

Sie können einen Ausnahmefehler erstellen, indem Sie einfach versuchen, eine Zahl durch Null zu teilen. Sie erhalten die ZeroDivisionError-Fehlermeldung, wie im Screenshot zu sehen ist. Der ZeroDivisionError-Teil der Meldung ist die Ausnahmeklasse, von der es viele gibt.



SCHRITT 2

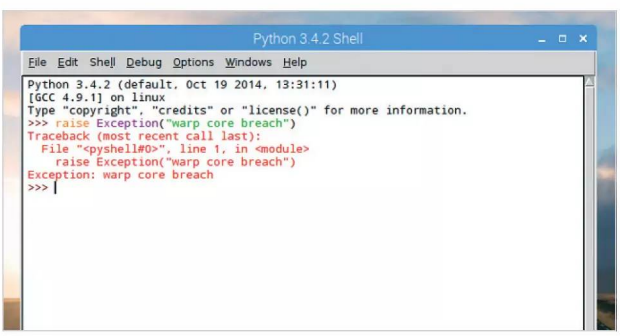
Die meisten Ausnahmen werden automatisch ausgelöst, wenn Python auf ein Problem mit dem Code stößt. Sie können jedoch eigene Ausnahmen erstellen, die darauf ausgelegt sind, den potenziellen Fehler zu enthalten und darauf zu reagieren, anstatt den Code fehlschlagen zu lassen.



SCHRITT 3

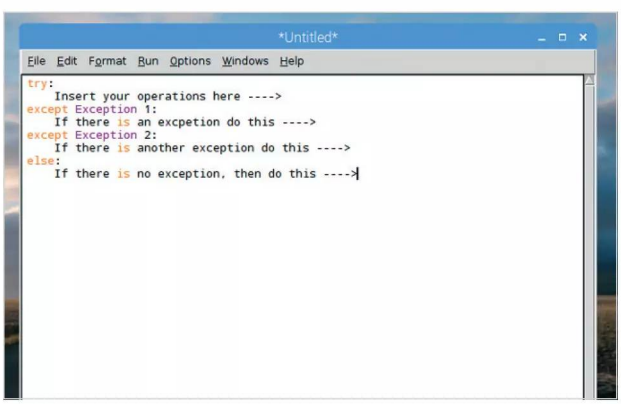
Sie können mit der Funktion raise exception in Python einen eigenen Fehlerbehandlungscode erstellen. Nehmen wir an, Ihr Code hat Sie ins All versetzt, um den Weltraum zu erkunden. Zu viele Erkundungen führen jedoch zum Ausfall des Warp-Antriebs. Um zu verhindern, dass das Spiel aufgrund der Supernova des Warp-Antriebs beendet wird, können Sie eine benutzerdefinierte Ausnahme erstellen:

```
raise Exception("warp core breach")
```



SCHRITT 4

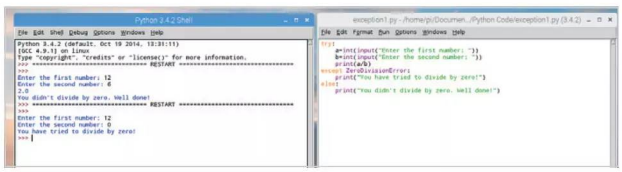
Um Fehler im Code zu erfassen, können Sie den potenziellen Fehler innerhalb eines Versuchblocks (try) umschließen. Dieser Block besteht aus try, except und else. Der Code ist in try enthalten. Bei einer Abweichung wird etwas ausgeführt, gibt es keine Abweichung wird etwas anderes ausgeführt.



SCHRITT 5

Nehmen Sie z. B. den bei der Division durch Null aufgetretenen Fehler. Sie können eine Ausnahme erstellen, bei der der Code den Fehler verarbeiten kann, ohne dass Python aufgrund des Problems beendet werden muss:

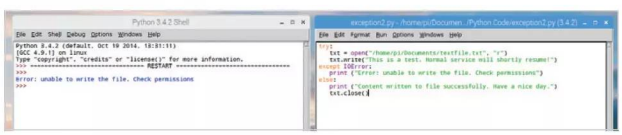
```
try:
  a=int(input("Enter the first number: "))
  b=int(input("Enter the second number: "))
  print(a/b)
except ZeroDivisionError:
  print("You have tried to divide by zero!")
else:
  print("You didn't divide by zero. Well done!")
```



SCHRITT 6

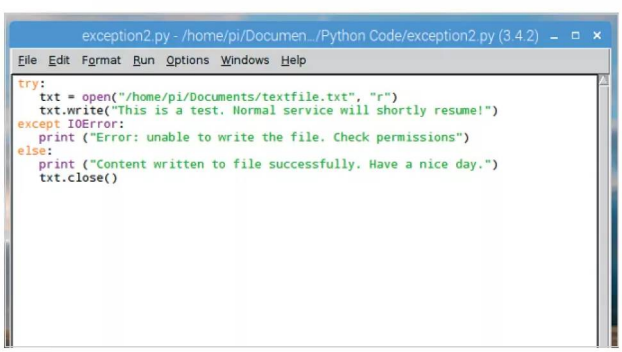
Mit Ausnahmen lassen sich viele hilfreiche Aufgaben bearbeiten. Angenommen, Sie möchten die Datei eines vorherigen Beispiels öffnen und in diese schreiben:

```
try:
  txt = open("/home/pi/Documents/textfile.txt", "r")
  txt.write("This is a test. Normal service will
shortly resume!")
except IOError:
  print ("Error: unable to write the file. Check
permissions")
else:
  print ("Content written to file successfully. Have
a nice day.")
  txt.close()
```



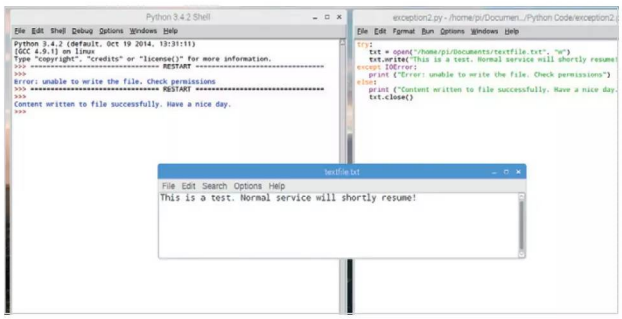
SCHRITT 7

Dies wird natürlich nicht funktionieren, da die Datei textfile.txt durch das "r" schreibgeschützt geöffnet wird. Anstatt von Python die Fehlermeldung zu bekommen, dass Sie etwas falsch gemacht haben, werden Sie mit der durch IOError erstellten Ausnahmeklasse darüber informiert, dass die Berechtigungen falsch sind.



SCHRITT 8

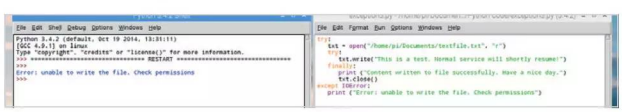
Natürlich lässt sich das Problem schnell beheben, indem Sie die schreibgeschützte Instanz "r" durch ein "w" für den Schreibvorgang austauschen. Dadurch können Sie die Datei erstellen, den Inhalt schreiben und dann die Änderungen in die Datei übernehmen. Als Endergebnis erhalten wir in diesem Fall eine erfolgreiche Ausführung des Codes.



SCHRITT 9

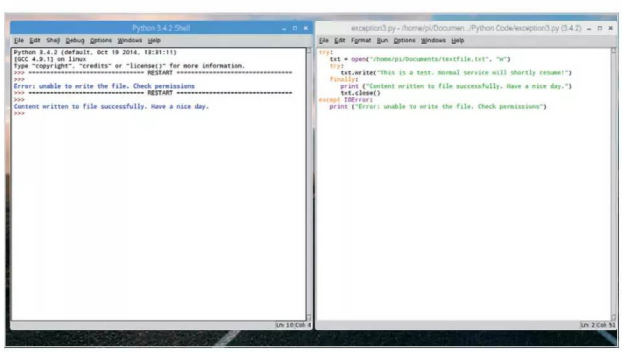
Sie können auch einen finally-Block nehmen. Er arbeitet auf ähnliche Weise, nur die else-Option funktioniert damit nicht. Wir nehmen erneut das Beispiel von Schritt 6:

```
try:
  txt = open("/home/pi/Documents/textfile.txt", "r")
  try:
    txt.write("This is a test. Normal service will
shortly resume!")
  finally:
    print ("Content written to file successfully. Have
a nice day.")
    txt.close()
except IOError:
  print ("Error: unable to write the file. Check
permissions")
```



SCHRITT 10

Wie zuvor wird ein Fehler auftreten, da Sie die schreibgeschützte Berechtigung "r" verwendet haben. Wenn Sie sie in ein "w" ändern, wird der Code ausgeführt, ohne dass der Fehler in der IDLE-Shell angezeigt wird. Es kann zu Beginn etwas schwierig sein, den Ausnahmecode richtig hinzubekommen. Mit etwas Übung werden Sie den Dreh aber bald raus haben.



Grafiken in Python

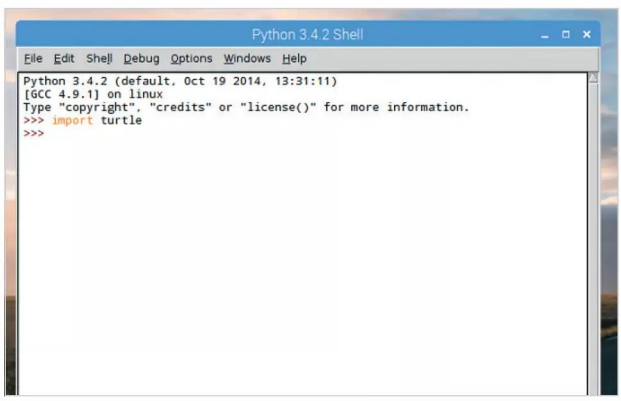
Obwohl die Arbeit mit Text auf dem Bildschirm toll ist, wird es Zeiten geben, in denen grafische Darstellungen wünschenswert sind. Python 3 verfügt über zahlreiche Möglichkeiten zur Einbindung von Grafiken, die erstaunlich leistungsstark sind.

GRAFISCHE DARSTELLUNGEN

Sie können einfache Grafiken, Linien, Quadrate usw. zeichnen, oder mithilfe eines der vielen verfügbaren Python-Module einige spektakuläre Effekte kreieren.

SCHRITT 1

Eines der besten grafischen Module zum Erlernen der Python-Grafiken ist Turtle. Das Turtle-Modul basiert auf den in vielen Schulen verwendeten Schildkröten-Robotern, die programmiert werden können, um etwas auf einem großen Stück Papier auf dem Boden zu zeichnen. Das Turtle-Modul kann mit `import turtle` importiert werden.

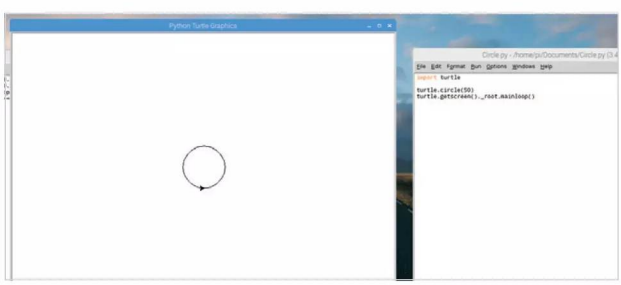


SCHRITT 2

Wir beginnen mit dem Zeichnen eines einfachen Kreises. Öffnen Sie eine neue Datei und geben Sie den folgenden Code ein:

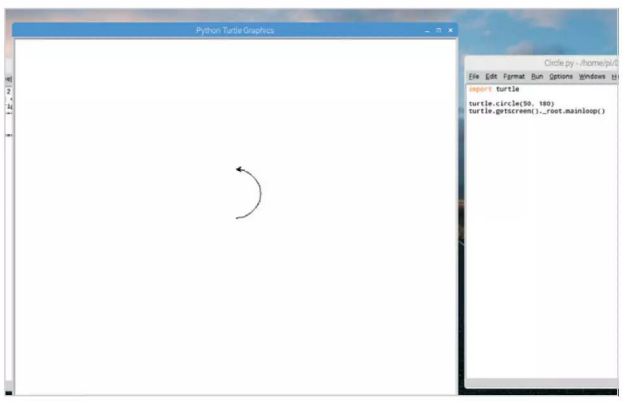
```
import turtle

turtle.circle(50)
turtle.getscreen()._root.mainloop()
```

Drücken Sie wie gewohnt F5, um den Code zu speichern und auszuführen. Ein neues Fenster öffnet sich und die Schildkröte des Turtle-Moduls zeichnet einen Kreis.



SCHRITT 3

Es ist der Befehl turtle.circle(50), der den Kreis auf dem Bildschirm zeichnet, wobei 50 die Größe ist. Sie können mit den Größen bis zu 100, 150 und sogar noch höher gehen. Mit der Eingabe `turtle.circle(50, 180)` zeichnen Sie einen Bogen, wobei 50 wieder die Größe ist, aber diesmal teilen Sie Python mit, nur 180° des Kreises zu zeichnen.

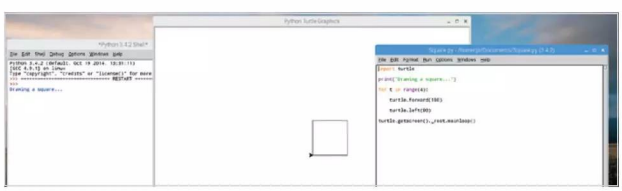


SCHRITT 4

Der letzte Teil des Kreiscodes weist Python an, das Fenster, in dem die Zeichnung gerade ausgeführt wird, offen zu halten, damit der Benutzer es durch Klicken schließen kann. Wir machen nun ein Quadrat:

```
import turtle

print("Drawing a square…")

for t in range(4):
    turtle.forward(100)
    turtle.left(90)
turtle.getscreen()._root.mainloop()
```

Wie Sie sehen, haben wir eine Schleife eingefügt, um die Seiten des Quadrats zu zeichnen.



SCHRITT 5

Indem Sie dem Quadrat-Code eine neue Zeile hinzufügen, können Sie dem Quadrat etwas Farbe verleihen:

```
turtle.color("Red")
```

Sie können das Ganze sogar in eine richtige Schildkröte umwandeln, indem Sie Folgendes eingeben:

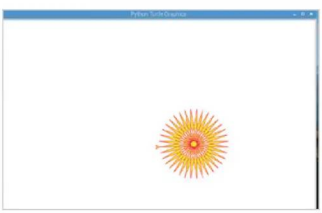
```
turtle.shape("turtle")
```

Sie können mit den Befehlen `turtle.begin_fill()` und `turtle.end_fill()` das Quadrat mit den ausgewählten Farben füllen; in unserem Fall erhält es einen roten Umriss und eine gelbe Füllung.

SCHRITT 6

Das Turtle-Modul kann einige ziemlich gute Formen hervorbringen und wenn Sie seine Funktionsweise erst mal gemeistert haben, können Sie auch komplexere Formen probieren. Geben Sie das folgende Beispiel ein:

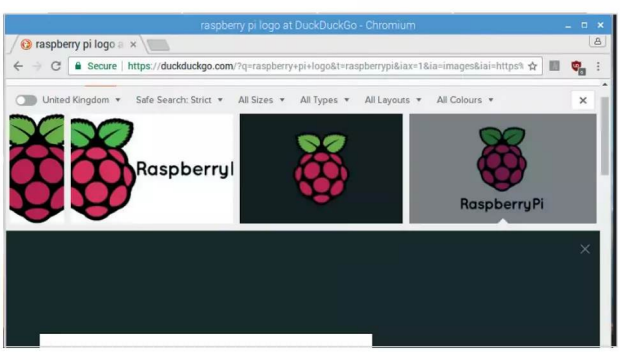
```
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
  forward(200)
  left(170)
  if abs(pos()) < 1:
    break
end_fill()
done()
```



Dies ist eine andere, aber äußerst effektiv Methode.

SCHRITT 7

Sie können Grafiken auch mit dem Pygame-Modul anzeigen. Es gibt zahlreiche Möglichkeiten, mit denen Sie mithilfe von Pygame Grafiken auf dem Bildschirm ausgeben können. Wir zeigen zunächst ein vordefiniertes Bild an. Öffnen Sie einen Browser, suchen Sie nach einem Bild und legen Sie es in dem Ordner ab, in dem Sie Ihren Python-Code speichern.



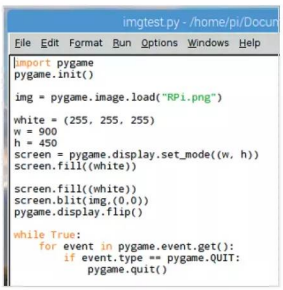
SCHRITT 8

Wir holen uns nun den Code, indem wir das Pygame-Modul importieren:

```
import pygame
pygame.init()
```

```
img = pygame.image.load("RPi.png")
```

```
white = (255, 255, 255)
w = 900
h = 450
screen = pygame.display.
set_mode((w, h))
screen.fill((white))
```




```
screen.fill((white))
screen.blit(img,(0,0))
pygame.display.flip()

while True:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      pygame.quit()
```

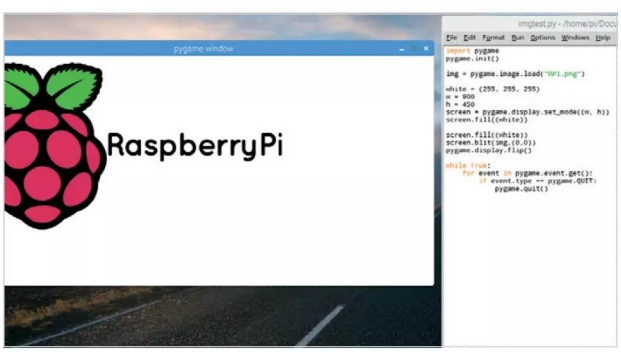
SCHRITT 9

Im vorherigen Schritt haben wir Pygame importiert, die Pygame-Engine gestartet und sie aufgefordert, unser unter RPi.png gespeichertes Raspberry Pi-Logo zu importieren. Anschließend haben wir die Hintergrundfarbe des Fensters definiert, um das Bild und die Fenstergröße gemäß den tatsächlichen Bildabmessungen anzuzeigen. Zu guter Letzt haben wir eine Schleife, mit der wir das Fenster schließen.

```
img = pygame.image.load("RPi.png")

white = (255, 255, 255)
w = 900
h = 450
screen = pygame.display.set_mode((w, h))
screen.fill((white))

screen.fill((white))
screen.blit(img,(0,0))
pygame.display.flip()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
```

SCHRITT 10

Drücken Sie F5, um den Code zu speichern und auszuführen. Ihr Bild wird in einem neuen Fenster angezeigt. Experimentieren Sie mit den Farben, Größen usw. und nehmen Sie sich Zeit, um die vielen Funktionen im Pygame-Modul kennenzulernen.



Das Kalendermodul

Neben dem Zeitmodul kann auch das Kalendermodul einige interessante Ergebnisse erzeugen. Es kann weitaus mehr als nur das Datum im ähnlichen Format des Zeitmoduls wiederzugeben; Sie können sich z. B. sogar einen Wandkalender anzeigen lassen.

ARBEITEN MIT DATEN

Das Kalendermodul ist in Python 3 bereits integriert. Sollte es jedoch aus irgendeinem Grund fehlen, können Sie es mit `pip install calendar` für Windows oder `sudo pip install calendar` für Linux und macOS hinzufügen.

SCHRITT 1

Starten Sie Python 3 und geben Sie `import calendar` ein, um das Modul und seine enthaltenen Funktionen aufzurufen. Nachdem es in den Speicher geladen wurde, können Sie mit der folgenden Eingabe beginnen:

```
sep=calendar.TextCalendar(calendar.SUNDAY)
sep.prmonth(2019, 9)
```

```
*Python 3.5.3 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>> import calendar
>>> sep=calendar.TextCalendar(calendar.SUNDAY)
>>> sep.prmonth(2019, 9)
   September 2019
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
>>>
>>>
```

SCHRITT 2

Wie Sie sehen werden die Tage des Septembers 2019 in einer Art Wandkalender angezeigt. Natürlich können Sie 2019 und 9 in jedes beliebige Jahr und jeden beliebigen Monat ändern, z. B. einen Geburtstag (1973, 6). In der ersten Zeile wird TextCalendar so konfiguriert, dass die Wochen an einem Sonntag beginnen. Wenn Sie möchten, können Sie auch Montag als Wochenanfang wählen.

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>> import calendar
>>> sep=calendar.TextCalendar(calendar.SUNDAY)
>>> sep.prmonth(2019, 9)
   September 2019
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
>>>
>>> birthday=calendar.TextCalendar(calendar.MONDAY)
>>> birthday.prmonth(1973, 6)
     June 1973
Mo Tu We Th Fr Sa Su
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
>>>
```

SCHRITT 3

Das Kalendermodul enthält zahlreiche Funktionen, die für die Erstellung Ihres eigenen Codes von Interesse sein könnten. Sie können z. B. die Anzahl der Schaltjahre zwischen zwei bestimmten Jahren ermitteln:

```
leaps=calendar.leapdays(1900, 2019)
print(leaps)
```

Beginnend bei 1904 ist das Ergebnis 29.

```
Python 3.5.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>> import calendar
>>> leaps=calendar.leapdays(1900, 2019)
>>> print(leaps)
29
>>>
```

SCHRITT 4

Sie können dieses spezielle Beispiel sogar in einen interaktiven Python-Code umwandeln:

```
import calendar
print(“>>>>>>>>>>Leap Year
Calculator<<<<<<<<<<\n”)
y1=int(input(“Enter the first year: “))
y2=int(input(“Enter the second year: “))
leaps=calendar.leapdays(y1, y2)
print(“Number of leap years between”, y1, “and”,
y2, “is:”, leaps)
```

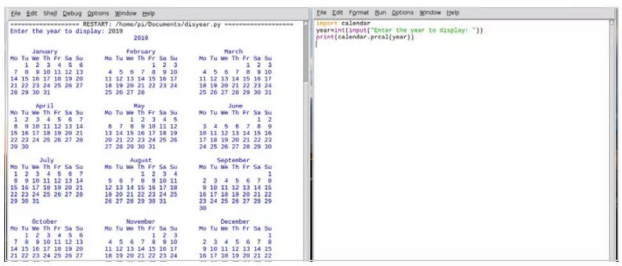
```
Python 3.5.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /home/pi/Documents/leaps.py ==========
>>>>>>>>>>Leap Year Calculator<<<<<<<<<<

Enter the first year: 1756
Enter the second year: 2022
Number of leap years between 1756 and 2022 is: 65
>>>

leaps.py - /home/pi/Documents/leaps.py (3.5.3)
File  Edit  Format  Run  Options  Window  Help
import calendar
print(">>>>>>>>>>Leap Year Calculator<<<<<<<<<<\n")
y1=int(input("Enter the first year: "))
y2=int(input("Enter the second year: "))
leaps=calendar.leapdays(y1, y2)
print("Number of leap years between", y1, "and", y2, "is:", leaps)
```

SCHRITT 5

Sie können auch ein Programm erstellen, das alle Tage, Wochen und Monate innerhalb eines bestimmten Jahres anzeigt:

```
import calendar
year=int(input("Enter the year to display: "))
print(calendar.prcal(year))
```

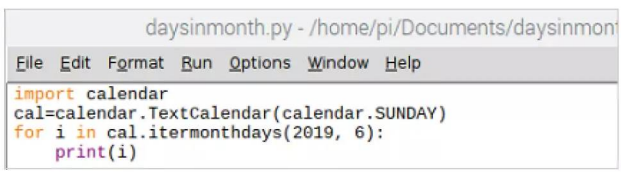
Sie werden sicherlich zustimmen, dass dies ein äußerst praktischer Code ist.



SCHRITT 6

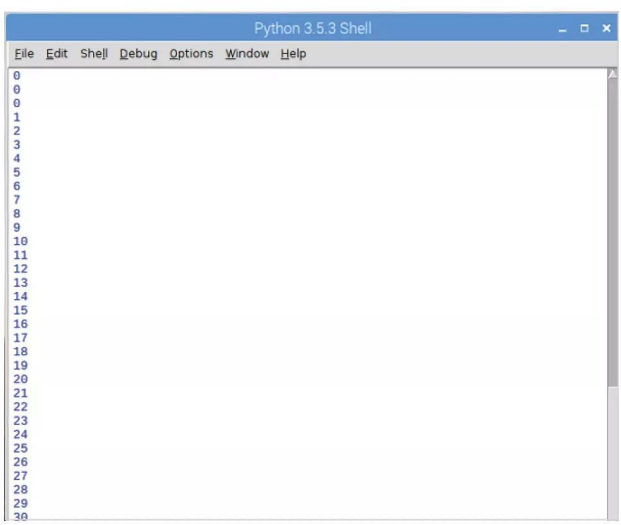
Interessanterweise können wir auch die Anzahl der Tage in einem Monat mit einer einfachen for-Schleife auflisten:

```
import calendar
cal=calendar.TextCalendar(calendar.SUNDAY)
for i in cal.itermonthdays(2019, 6):
        print(i)
```



SCHRITT 7

Wie Sie sehen hat der Code am Anfang einige Nullen erzeugt. Dies liegt am Anfangstag der Woche, in diesem Fall dem Sonntag, und an den überlappenden Tagen des Vormonats. Die Zählung der Tage beginnt daher am Samstag, dem 1. Juni 2019, und wird 30 ergeben.

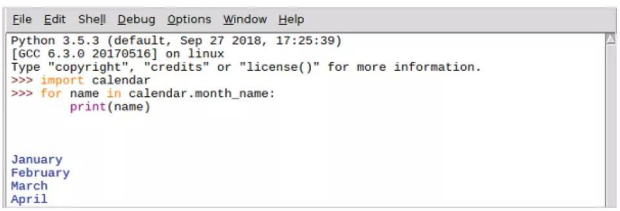


SCHRITT 8

Sie können auch die einzelnen Monate oder Wochentage ausgeben:

```
import calendar
for name in calendar.month_name:
        print(name)
```

```
import calendar
for name in calendar.day_name:
        print(name)
```

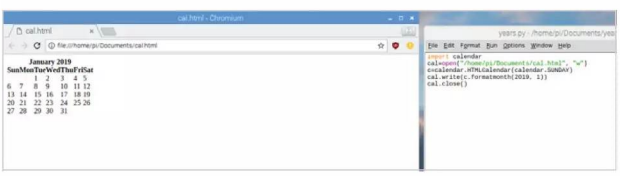


SCHRITT 9

Das Kalendermodul ermöglicht uns auch, für die Ausgabe auf Webseiten die Funktionen in HTML zu schreiben. Wir beginnen mit dem Erstellen einer neuen Datei:

```
import calendar
cal=open("/home/pi/Documents/cal.html", "w")
c=calendar.HTMLCalendar(calendar.SUNDAY)
cal.write(c.formatmonth(2019, 1))
cal.close()
```

Dieser Code erstellt eine HTML-Datei namens cal, öffnet sie mit einem Browser und zeigt den Kalender für Januar 2019 an.

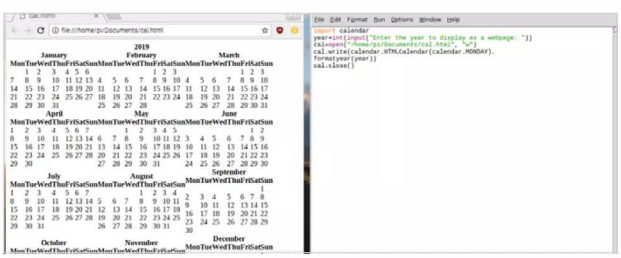


SCHRITT 10

Natürlich können Sie den Code ändern, sodass ein bestimmtes Jahr als Webseitenkalender angezeigt wird:

```
import calendar

year=int(input("Enter the year to display as a
webpage: "))
cal=open("/home/pi/Documents/cal.html", "w")
cal.write(calendar.HTMLCalendar(calendar.MONDAY).
formatyear(year))
cal.close()
```

Dieser Code fragt den Benutzer nach einem Jahr und erstellt dann die erforderliche Webseite. Denken Sie daran, den Dateipfad zu ändern.



Das OS-Modul

Das OS-Modul ermöglicht Ihnen die direkte Interaktion mit den integrierten Befehlen Ihres Betriebssystems. Die Befehle hängen von Ihrem Betriebssystem ab – einige funktionieren mit Windows, während andere für Linux und macOS entwickelt wurden.

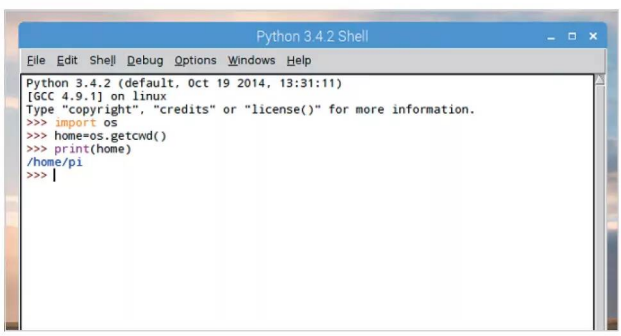
HINEIN INS SYSTEM

Eine der wichtigsten Eigenschaften des OS-Moduls ist die Fähigkeit, im System gespeicherte Dateien aufzulisten, zu verschieben, zu erstellen, zu löschen und anderweitig mit ihnen zu interagieren, was es zum perfekten Modul für Backup-Code macht.

SCHRITT 1

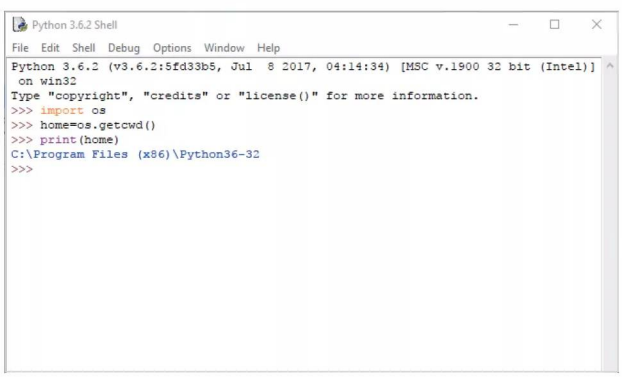
Sie können das OS-Modul mit einigen einfachen Funktionen starten, um zu sehen, wie es mit der Betriebssystemumgebung interagiert, auf der Python ausgeführt wird. Wenn Sie Linux oder den Raspberry Pi verwenden, probieren Sie Folgendes aus:

```
import os
home=os.getcwd()
print(home)
```



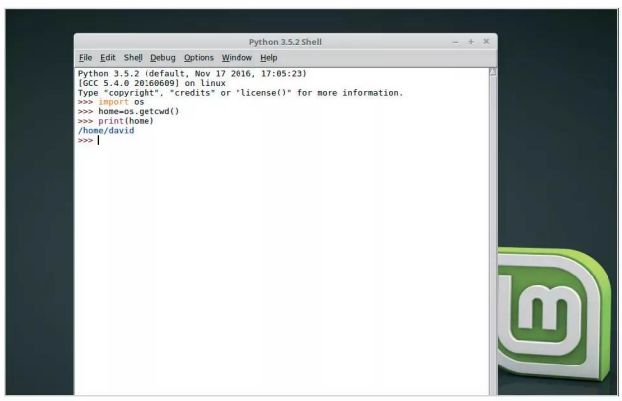
SCHRITT 2

Das Ergebnis der Home-Variablen ist der Home-Ordner des aktuellen Benutzers auf dem System. In unserem Beispiel ist das /home/pi. Je nachdem, mit welchem Benutzernamen Sie angemeldet sind und welches Betriebssystem Sie verwenden, wird dies unterschiedlich sein. Windows 10 wird z. B. Folgendes ausgeben: C:\Programme (x86)\Python36-32.



SCHRITT 3

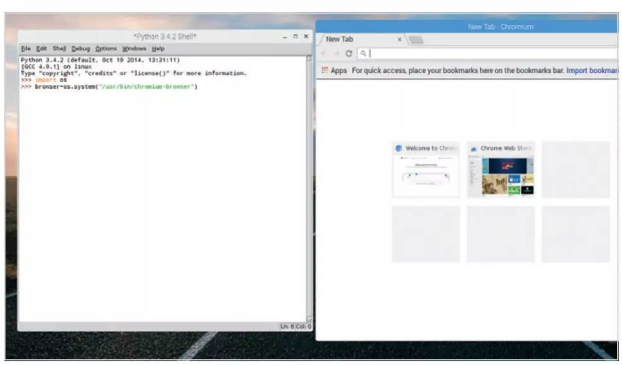
Die Windows-Ausgabe unterscheidet sich, da sie das aktuelle Arbeitsverzeichnis von Python ist, das vom System bestimmt wird. Wie Sie vielleicht vermuten, fordert die Funktion os.getcwd() Python auf, das aktuelle Arbeitsverzeichnis abzurufen. Linux-, macOS- und Raspberry Pi-Nutzer werden etwas Ähnliches sehen.



SCHRITT 4

Ein weiteres interessantes Element des OS-Moduls ist die Möglichkeit, Programme zu starten, die im Host-System installiert sind. Um beispielsweise den Chromium-Browser in einem Python-Programm zu starten, geben Sie den folgenden Befehl ein:

```
import os
browser=os.system("/usr/bin/chromium-browser")
```



SCHRITT 5

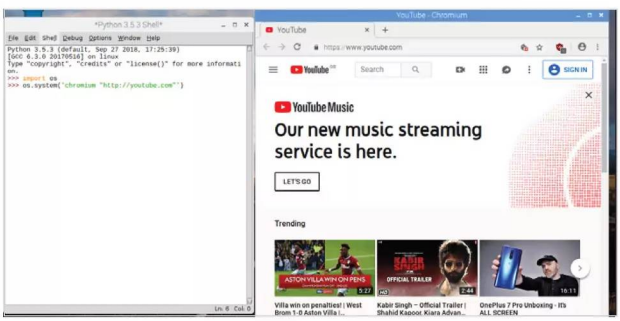
Die os.system()-Funktion ermöglicht die Interaktion mit externen Programmen. Mit dieser Methode können Sie sogar frühere Python-Programme aufrufen. Sie müssen für eine erfolgreiche Anwendung natürlich den vollständigen Pfad und den Namen der Programmdatei kennen. Sie können jedoch Folgendes eingeben:

```
import os
os.system('start chrome "https://www.youtube.com/feed/music"')
```

SCHRITT 6

Für das Beispiel in Schritt 5 haben wir Windows verwendet, um zu zeigen, dass das OS-Modul auf allen Plattformen ungefähr gleich funktioniert. In diesem Fall haben wir YouTubes Musikseite geöffnet. Es ist somit möglich, bestimmte Seiten zu öffnen:

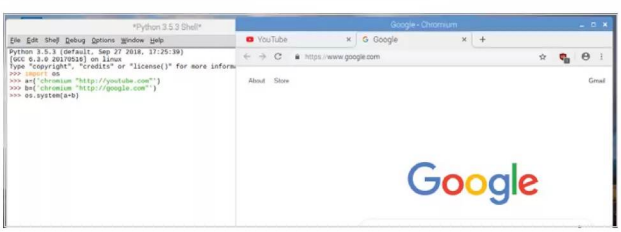
```
import os
os.system('chromium "http://youtube.com"')
```



SCHRITT 7

Beachten Sie im Beispiel des vorherigen Schritts die Anwendung von einfachen und doppelten Anführungszeichen. Die einfachen Anführungszeichen umschließen den gesamten Befehl und starten Chromium, während die doppelten Anführungszeichen die angegebene Seite öffnen. Sie können mit Variablen sogar mehrere Registerkarten im selben Browser aufrufen:

```
import os
a='chromium "http://youtube.com"')
b='chromium "http://google.com"')
os.system(a  +  b)
```

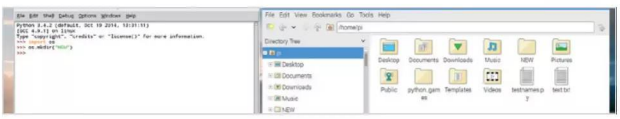


SCHRITT 8

Verzeichnisse oder Ordner bearbeiten zu können, ist eine der besten Eigenschaften des OS-Moduls. Um z. B. ein neues Verzeichnis zu erstellen, geben Sie Folgendes ein:

```
import os
os.mkdir("NEW")
```

Dadurch wird im aktuellen Arbeitsverzeichnis ein neues Verzeichnis erstellt, das nach dem Objekt in der mkdir-Funktion benannt ist.



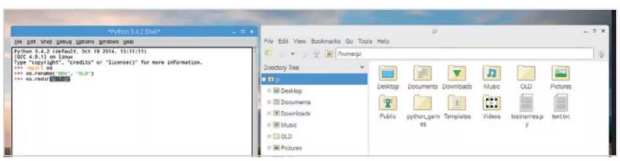
SCHRITT 9

Sie können Ihre erstellten Verzeichnisse auch durch die folgende Eingabe umbenennen:

```
import os
os.rename("NEW", "OLD")
```

Zum Löschen geben Sie Folgendes ein:

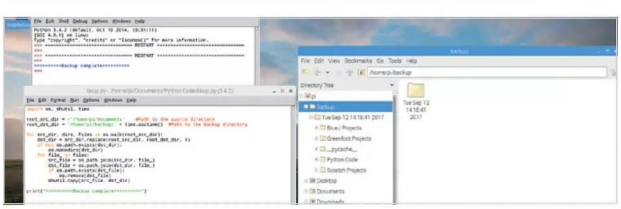
```
import os
os.rmdir("OLD")
```



SCHRITT 10

Ein weiteres Modul, das mit OS funktioniert, ist Shutil. Sie können mit dem Shutil-, dem OS- und dem Tim e-Modul ein mit einem Zeitstempel versehenes Backup-Verzeichnis erstellen und Dateien in dieses Verzeichnis kopieren:

```
import os, shutil, time

root_src_dir = r'/home/pi/Documents'
root_dst_dir = '/home/pi/backup/' + time.asctime()

for src_dir, dirs, files in os.walk(root_src_dir):
  dst_dir = src_dir.replace(root_src_dir, root_
dst_dir, 1)
  if not os.path.exists(dst_dir):
    os.makedirs(dst_dir)
  for file_ in files:
    src_file = os.path.join(src_dir, file_)
    dst_file = os.path.join(dst_dir, file_)
    if os.path.exists(dst_file):
      os.remove(dst_file)
    shutil.copy(src_file, dst_dir)

print(">>>>>>>>>>Backup complete<<<<<<<<<<")
```



Das Random-Modul

Auf das Random-Modul werden Sie in Ihrer Python-Programmierzeit wahrscheinlich öfter stoßen. Es wurde entwickelt, um zufällige Zahlen oder Buchstaben zu erstellen. Es funktioniert zwar nicht ganz zufällig, aber für die meisten Anforderungen reicht es aus.

ZUFALLSZAHLEN

Es gibt zahlreiche Funktionen innerhalb des Random-Moduls, die einige interessante und sehr nützliche Python-Programme erstellen können.

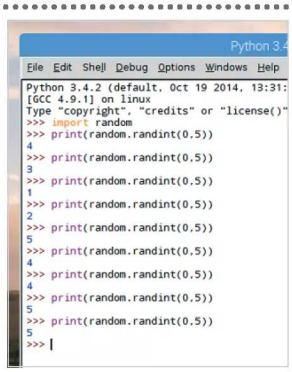
SCHRITT 1

Genau wie bei den anderen Modulen müssen Sie das Random-Modul importieren, bevor Sie eine der Funktionen verwenden können, die wir uns in diesem Tutorial anschauen werden. Wir beginnen damit, eine einfache Zufallszahl zwischen 1 und 5 auszugeben:

```
import random
print(randomint(0,5))
```

SCHRITT 2

In unserem Beispiel haben wir die Nummer vier erhalten. Geben Sie die print-Funktion noch einige Male ein, um verschiedene Integer-Werte aus der angegebenen Zahlengruppe 0 bis 5 zu erhalten. Der Gesamteffekt ist zwar nur scheinbar zufällig, für den durchschnittlichen Programmierer jedoch ausreichend.

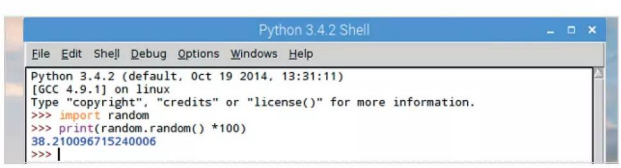


SCHRITT 3

Für eine größere Anzahl von Zahlen, einschließlich Gleitkommawerten, können Sie mithilfe des Multiplikationszeichens den Bereich erweitern:

```
import random
print(random.random() *100)
```

Dies zeigt eine Gleitkommazahl zwischen 0 und 100 mit fünfzehn Dezimalstellen an.

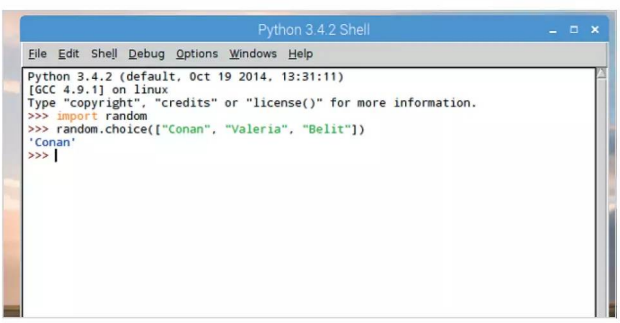


SCHRITT 4

Das Random-Modul wird jedoch nicht ausschließlich für Zahlen verwendet. Sie können damit einen zufälligen Listeneintrag auswählen, wobei die Liste alles enthalten kann:

```
import random
random.choice(["Conan", "Valeria", "Belit"])
```

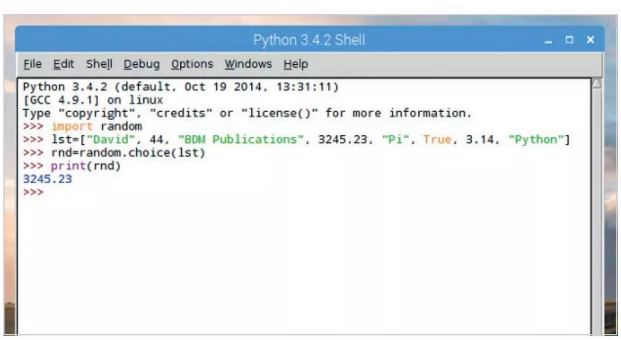
Dadurch wird zufällig einer der Namen von einem unserer Abenteurer angezeigt, was für ein Textabenteuerspiel eine tolle Ergänzung ist.



SCHRITT 5

Sie können das vorherige Beispiel erweitern, indem Sie die Funktion random.choice() aus einer Liste gemischter Variablen auswählen lassen, z. B:

```
import random
lst=["David", 44, "BDM Publications", 3245.23,
"Pi", True, 3.14, "Python"]
rnd=random.choice(lst)
print(rnd)
```

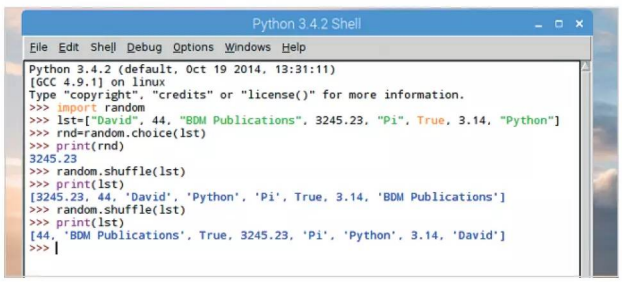


SCHRITT 6

Interessanterweise können Sie auch mit einer Funktion innerhalb des Zufallsmoduls die Elemente in der Liste mischen, wodurch die Gleichung etwas zufälliger ausfällt:

```
random.shuffle(lst)
print(lst)
```

Auf diese Weise können Sie die Liste weiter mischen, bevor ein zufälliger Eintrag angezeigt wird.

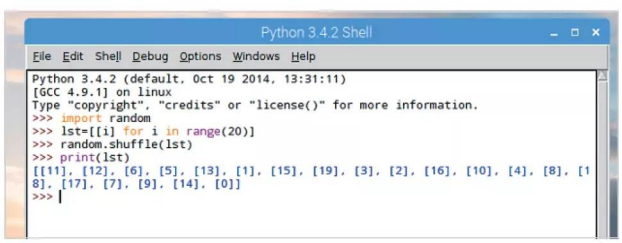


SCHRITT 7

Mit der shuffle-Funktion können Sie eine komplett zufällige Liste von Zahlen erstellen, z. B. innerhalb eines bestimmten Bereichs:

```
import random
lst=[[i] for I in range(20)]
random.shuffle(lst)
print(lst)
```

Mischen Sie die Liste weiter. Sie können jedes Mal eine andere Auswahl von Zahlen von 0 bis 20 haben.

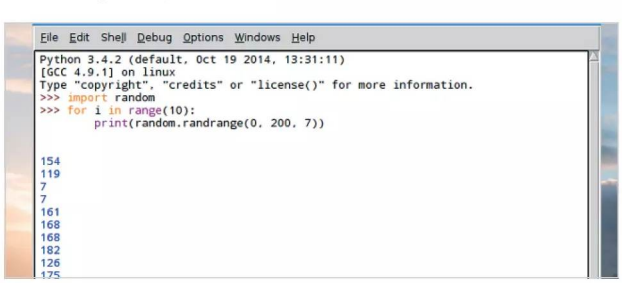


SCHRITT 8

Sie können mit der Start-, Stopp-, Schritt-Schleife auch in Schritten eine Zufallszahl aus einem bestimmten Bereich auswählen:

```
import random
for i in range(10):
  print(random.randrange(0, 200, 7))
```

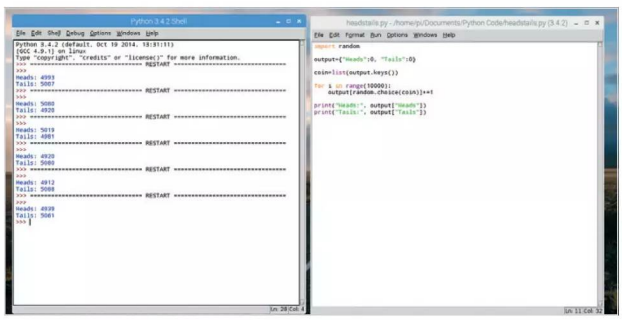
Die Ergebnisse variieren, aber Sie erhalten eine allgemeine Vorstellung davon, wie das Ganze funktioniert.



SCHRITT 9

Wir nehmen einen Beispielcode, der eine virtuelle Münze zehntausend Mal umdreht und zählt, wie oft sie Kopf oder Zahl anzeigt:

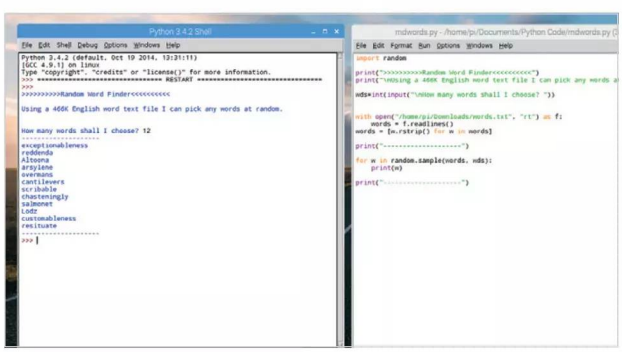
```
import random
output={“Heads”:0, “Tails”:0}
coin=list(output.keys())

for i in range(10000):
  output[random.choice(coin)]+=1

print(“Heads:”, output[“Heads”])
print(“Tails:”, output[“Tails”])
```



SCHRITT 10

Hier ist ein interessanter Code. Unter Verwendung einer Textdatei mit 466 000 Wörtern können Sie eine vom Benutzer generierte Anzahl von Wörtern aus der Datei auswählen (Textdatei unter: https://github.com/dwyl/english-words):

```
import random

print(“>>>>>>>>>>Random Word Finder<<<<<<<<<<”)
print(“\nUsing a 466K English word text file I can
pick any words at random.\n”)

wds=int(input(“\nHow many words shall I choose?
“))

with open(“/home/pi/Downloads/words.txt”, “rt”) as
f:
  words = f.readlines()
words = [w.rstrip() for w in words]

print(“----------------------”)

for w in random.sample(words, wds):
  print(w)

print(“----------------------”)
```



Das Tkinter-Modul

Die Ausführung Ihres Codes über die Befehlszeile oder die Shell ist kein Problem, aber Python kann noch viel mehr. Das Tkinter-Modul ermöglicht es dem Programmierer, eine grafische Benutzeroberfläche für die Interaktion mit dem Benutzer einzurichten.

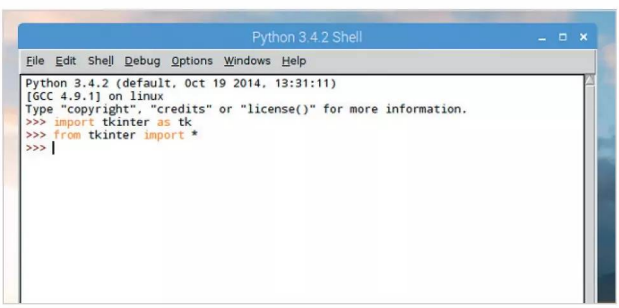
GUI - GRAPHICAL USER INTERFACE

Tkinter ist einfach zu benutzen und äußerst vielseitig. Wir schauen uns zunächst an, wie es funktioniert und Code integriert wird. Sie werden schon bald feststellen, wie leistungsstark dieses Modul ist.

SCHRITT 1

Tkinter ist normalerweise in Python 3 integriert. Wenn es jedoch nicht verfügbar ist, wenn Sie `import tkinter` eingeben, müssen Sie es in der Eingabeaufforderung per `pip install tkinter` installieren. Da wir Module nun anders als zuvor importieren, können wir sowohl bei der Eingabe als auch durch den Import der gesamten Modul-Inhalte Zeit einsparen:

```
import tkinter as tk
from tkinter import *
```

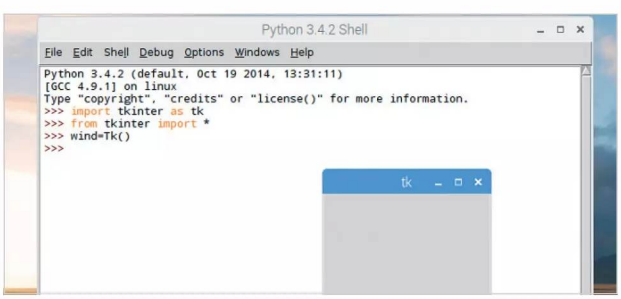


SCHRITT 2

Es wird zwar nicht empfohlen, den gesamten Inhalt eines Moduls mit dem Sternchen zu importieren, aber in der Regel richtet es auch keinen Schaden an. Wir beginnen mit der Erstellung eines einfachen GUI-Fensters. Geben Sie Folgendes ein:

```
wind=Tk()
```

Dadurch wird ein kleines, einfaches Fenster erstellt. Es gibt an dieser Stelle nichts weiter zu tun, klicken Sie daher auf das X in der Ecke, um das Fenster zu schließen.

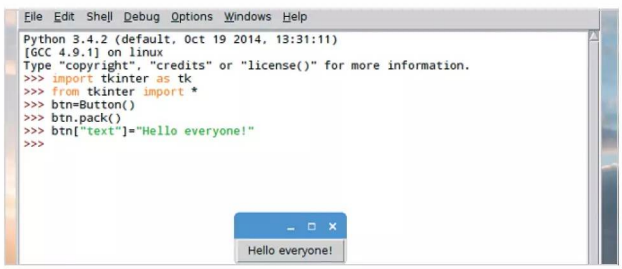


SCHRITT 3

Der ideale Ansatz besteht darin, die Funktion mainloop() in den Code einzufügen, um die Tkinter-Ereignisschleife zu steuern, doch dazu in Kürze mehr. Sie haben soeben ein Tkinter-Widget erstellt und es gibt noch weitere, mit denen wir experimentieren können:

```
btn=Button()
btn.pack()
btn["text"]="Hello everyone!"
```

Die erste Zeile konzentriert sich auf das neu erstellte Fenster. Gehen Sie in die Shell zurück und machen Sie mit den anderen Zeilen weiter.



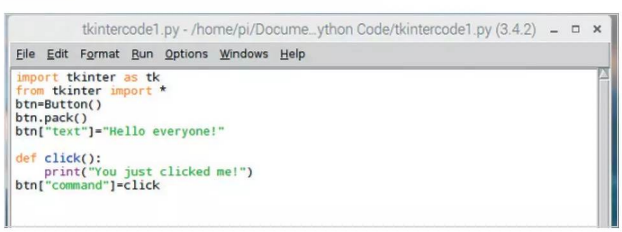
SCHRITT 4

Sie können das obige in einer neuen Datei kombinieren (File > New File):

```
import tkinter as tk
from tkinter import *
btn=Button()
btn.pack()
btn["text"]="Hello everyone!"
```

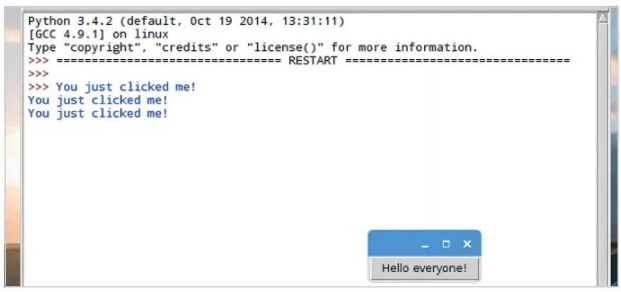
Fügen Sie nun einige interaktive Schaltflächen hinzu:

```
def click():
  print("You just clicked me!")
btn["command"]=click
```



SCHRITT 5

Speichern und führen Sie den Code in Schritt 4 aus. Ein Fenster mit „Hello everyone!" erscheint. Wenn Sie auf die Schaltfläche „Hello everyone!" klicken, gibt die Shell den Text „You just clicked me!" aus. Es ist nichts Spannendes, zeigt Ihnen aber, was mit ein paar Codezeilen erreicht werden kann.



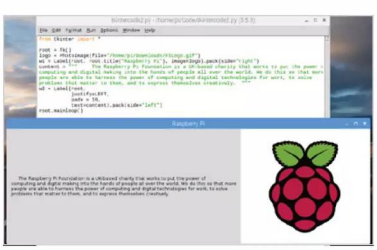
SCHRITT 6

Es können auch Text und Bilder in einem Tkinter-Fenster angezeigt werden. Es werden jedoch nur GIF-, PGM- oder PPM-Formate unterstützt, Sie müssen ein Bild daher konvertieren, bevor Sie den Code verwenden können. Für unser Beispiel nehmen wir das Raspberry Pi-Logo:

```
from tkinter import *

root = Tk()
logo = PhotoImage(file="/home/pi/Downloads/PiLogo.
gif")
w1 = Label(root, root.title("Raspberry Pi"),
image=logo).pack(side="right")
content = """  The Raspberry Pi Foundation is a
UK-based charity that works to put the power of
computing and digital making into the hands of
people all over the world. We do this so that more
people are able to harness the power of computing
and digital technologies for work, to solve problems
that matter to them, and to express themselves
creatively.  """
w2 = Label(root,
    justify=LEFT,
    padx = 10,
    text=content).pack(side="left")
root.mainloop()
```

SCHRITT 7

Der vorherige Code ist recht heftig, hauptsächlich aufgrund der content-Variablen, die einen Teil der Infoseite von der Raspberry Pi-Website enthält. Sie können den Inhalt, den root.title und das Bild natürlich persönlich anpassen.



SCHRITT 8

Sie können auch Optionsschaltflächen erstellen. Probieren Sie Folgendes aus:

```
from tkinter import *

root = Tk()

v = IntVar()

Label(root, root.title("Options"), text="""Choose
a preferred language:""",
  justify = LEFT, padx = 20).pack()
Radiobutton(root,
    text="Python",
    padx = 20,
    variable=v,
    value=1).pack(anchor=W)
Radiobutton(root,
    text="C++",
    padx = 20,
    variable=v,
    value=2).pack(anchor=W)

mainloop()
```

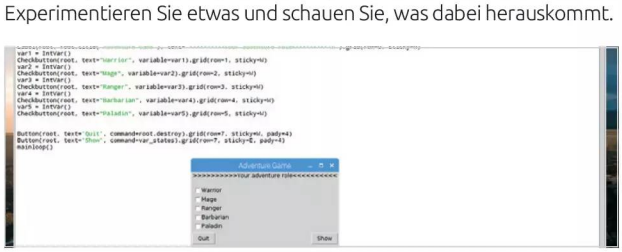
SCHRITT 9

Sie können auch Kontrollkästchen mit Schaltflächen und der Ausgabe an die Shell erstellen:

```
from tkinter import *
root = Tk()

def var_states():
  print("Warrior: %d,\nMage: %d" % (var1.get(),
var2.get()))

Label(root, root.title("Adventure Game"),
text=">>>>>>>>>>Your adventure role<<<<<<<<<<").
grid(row=0, sticky=N)
var1 = IntVar()
Checkbutton(root, text="Warrior", variable=var1).
grid(row=1, sticky=W)
var2 = IntVar()
Checkbutton(root, text="Mage", variable=var2).
grid(row=2, sticky=W)
Button(root, text='Quit', command=root.destroy).
grid(row=3, sticky=W, pady=4)
Button(root, text='Show', command=var_states).
grid(row=3, sticky=E, pady=4)

mainloop()
```

SCHRITT 10

Der Code in Schritt 9 hat einige neue Geometrieelemente in Tkinter eingeführt. Beachten Sie die Argumente sticky= N, E und W. Diese beschreiben die Positionen der Kontrollkästchen und Schaltflächen (North, East, South und West). Das Zeilenargument row platziert sie in separaten Zeilen. Experimentieren Sie etwas und schauen Sie, was dabei herauskommt.



Das Pygame-Modul

Wir haben uns das Pygame-Modul zuvor schon kurz angesehen, aber es hat noch viel mehr zu bieten. Pygame wurde entwickelt, um Python-Programmierern beim Erstellen von grafischen oder textbasierten Spielen zu helfen.

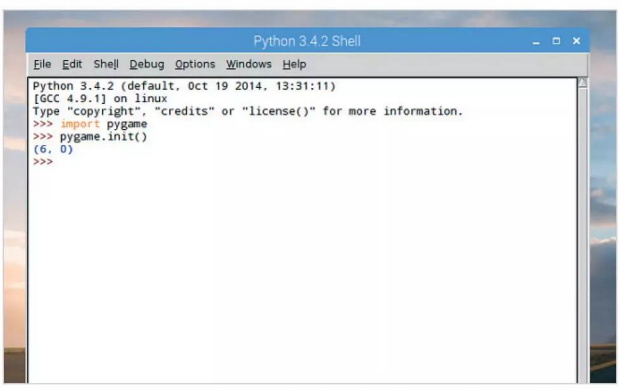
PYGAME

Pygame ist kein in Python integriertes Modul, in Raspberry Pi-Computern ist es jedoch bereits installiert. Auf anderen Betriebssystemen müssen Sie es in der Eingabeaufforderung mit pip install pygame installieren.

SCHRITT 1

Natürlich müssen Sie das Pygame-Modul in den Speicher laden, bevor Sie es verwenden können. Nachdem dies erledigt ist, muss der Benutzer es vor der Anwendung der Funktionen initialisieren:

```
import pygame
pygame.init()
```

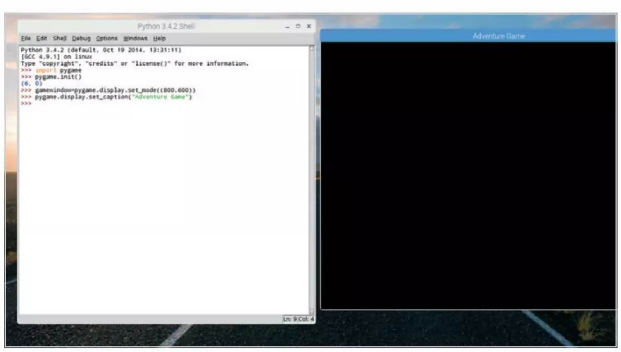


SCHRITT 2

Wir erstellen ein einfaches spielbereites Fenster, dem wir auch einen Titel geben:

```
gamewindow=pygame.display.set_mode((800,600))
pygame.display.set_caption("Adventure Game")
```

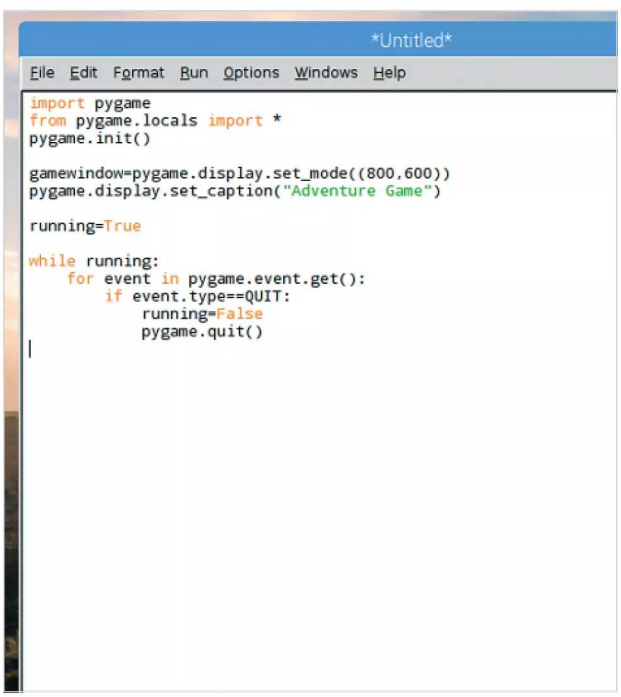
Wie Sie sehen, müssen Sie nach der Eingabe der ersten Zeile erneut in die IDLE-Shell gehen, um die Codeeingabe fortzusetzen. Den Titel des Fensters können Sie beliebig ändern.



SCHRITT 3

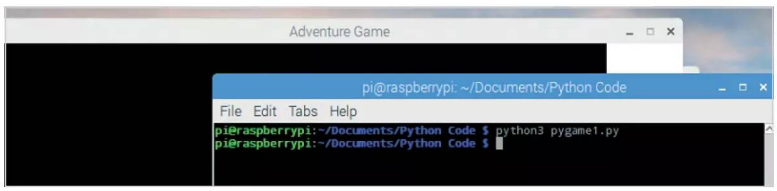
Leider können Sie das neu erstellte Pygame-Fenster nicht schließen, ohne auch die Python-IDLE-Shell zu schließen, was etwas unpraktisch ist. Aus diesem Grund müssen Sie im Editor arbeiten (New > File) und eine while-Schleife für True/False bzw. Wahr/Falsch erstellen:

```
import pygame
from pygame.locals import *
pygame.init()

gamewindow=pygame.display.set_mode((800,600))
pygame.display.set_caption("Adventure Game")

running=True

while running:
  for event in pygame.event.get():
    if event.type==QUIT:
      running=False
      pygame.quit()
```



SCHRITT 4

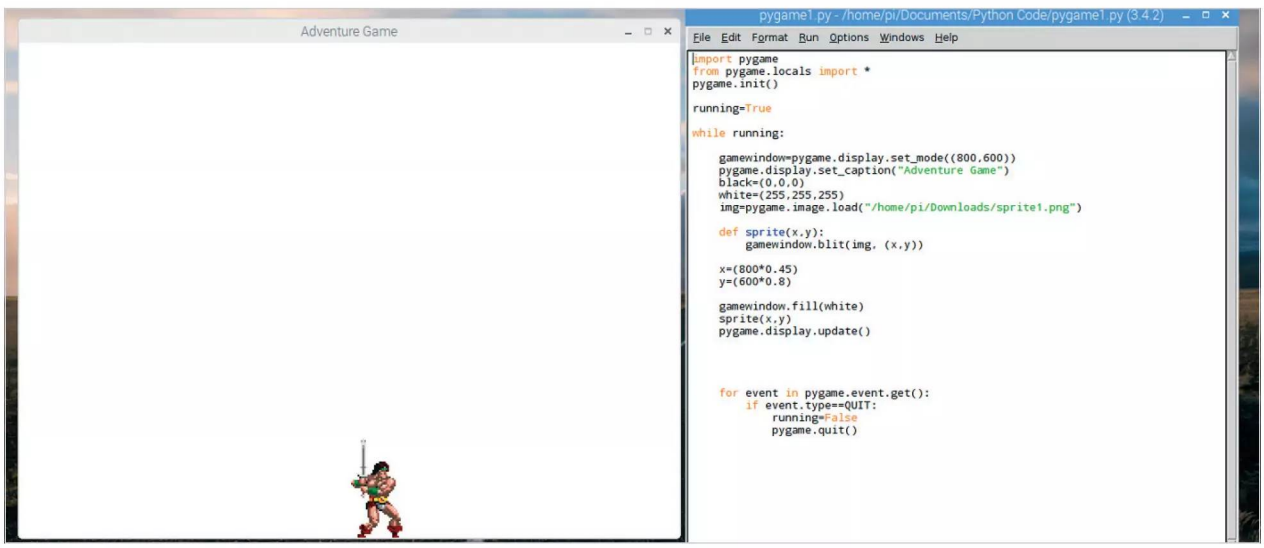
Machen Sie sich keine Sorgen, wenn sich das Pygame-Fenster immer noch nicht geschlossen hat, dies liegt lediglich an einer Diskrepanz zwischen der IDLE (die mit Tkinter geschrieben wird) und dem Pygame-Modul. Wenn Sie Ihren Code über die Befehlszeile ausführen, wird es sich problemlos schließen.



SCHRITT 5

Wir werden den Code jetzt ein wenig verschieben und den Pygame-Hauptcode innerhalb einer while-Schleife ausführen. Dadurch wird er überschaubarer und ist einfacher zu folgen. Wir haben eine Grafik heruntergeladen und müssen nun einige Parameter für Pygame bestimmen:

```
import pygame
pygame.init()

running=True

while running:

  gamewindow=pygame.display.set_mode((800,600))
  pygame.display.set_caption("Adventure Game")
  black=(0,0,0)
  white=(255,255,255)
  img=pygame.image.load("/home/pi/Downloads/
sprite1.png")

  def sprite(x,y):
    gamewindow.blit(img, (x,y))

  x=(800*0.45)
  y=(600*0.8)

  gamewindow.fill(white)
  sprite(x,y)
  pygame.display.update()

  for event in pygame.event.get():
    if event.type==pygame.QUIT:
      running=False
```



SCHRITT 6

Lassen Sie uns die Codeänderungen kurz durchgehen. Wir haben die Farben Schwarz und Weiß zusammen mit ihren jeweiligen RGB-Farbwerten definiert. Als Nächstes haben wir das heruntergeladene Bild namens sprite1.png hochgeladen und es der img-Variablen zugewiesen. Ferner haben wir eine Sprite-Funktion definiert. Die Blit-Funktion lässt uns das Bild verschieben.

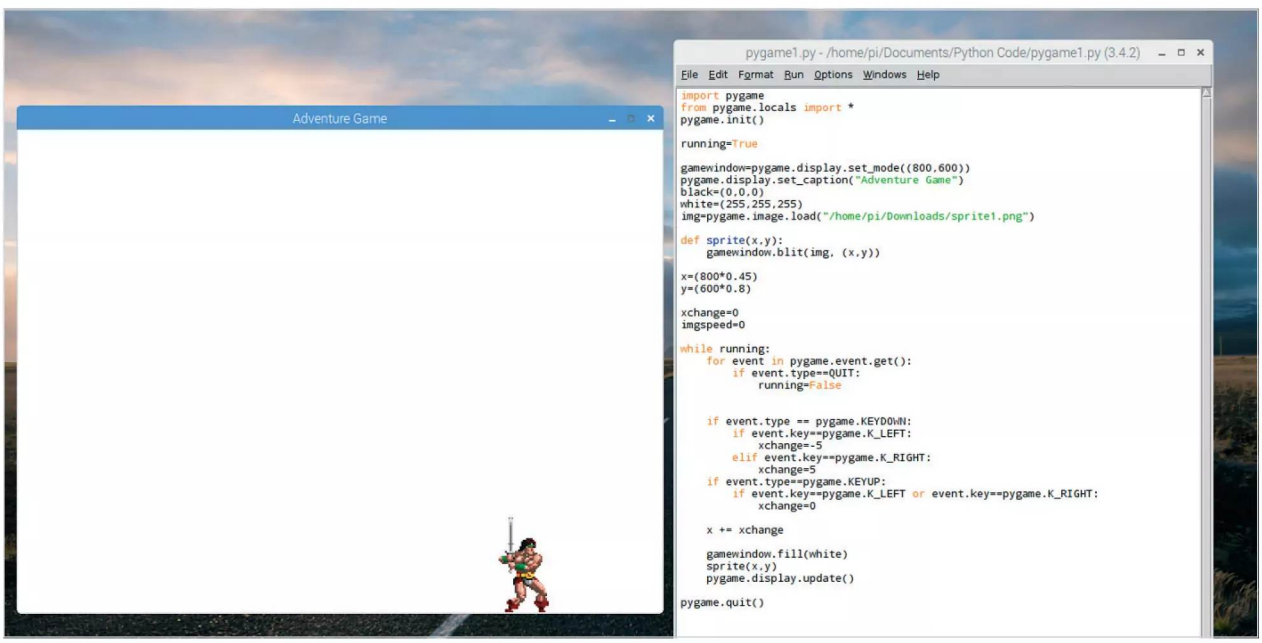
```
import pygame
from pygame.locals import *
pygame.init()

running=True

while running:

    gamewindow=pygame.display.set_mode((800,600))
    pygame.display.set_caption("Adventure Game")
    black=(0,0,0)
    white=(255,255,255)
    img=pygame.image.load("/home/pi/Downloads/sprite1.png")

    def sprite(x,y):
        gamewindow.blit(img, (x,y))
```

```
    x=(800*0.45)
    y=(600*0.8)

    gamewindow.fill(white)
    sprite(x,y)
    pygame.display.update()



    for event in pygame.event.get():
        if event.type==QUIT:
            running=False
            pygame.quit()
```

SCHRITT 7

Wir führen nun eine weitere Änderung aus, und zwar fügen wir innerhalb der while-Schleife eine Bewegungsoption hinzu sowie die Variablen, die benötigt werden, um das Sprite über den Bildschirm zu bewegen:

```
import pygame
from pygame.locals import *
pygame.init()

running=True

gamewindow=pygame.display.set_mode((800,600))
pygame.display.set_caption("Adventure Game")
black=(0,0,0)
white=(255,255,255)
img=pygame.image.load("/home/pi/Downloads/sprite1.
png")

def sprite(x,y):
  gamewindow.blit(img, (x,y))

x=(800*0.45)
y=(600*0.8)

xchange=0
imgspeed=0

while running:
  for event in pygame.event.get():
    if event.type==QUIT:
      running=False

  if event.type == pygame.KEYDOWN:
    if event.key==pygame.K_LEFT:
      xchange=-5
    elif event.key==pygame.K_RIGHT:
      xchange=5
  if event.type==pygame.KEYUP:
    if event.key==pygame.K_LEFT or event
key==pygame.K_RIGHT:
      xchange=0

  x += xchange

  gamewindow.fill(white)
  sprite(x,y)
  pygame.display.update()

pygame.quit()
```



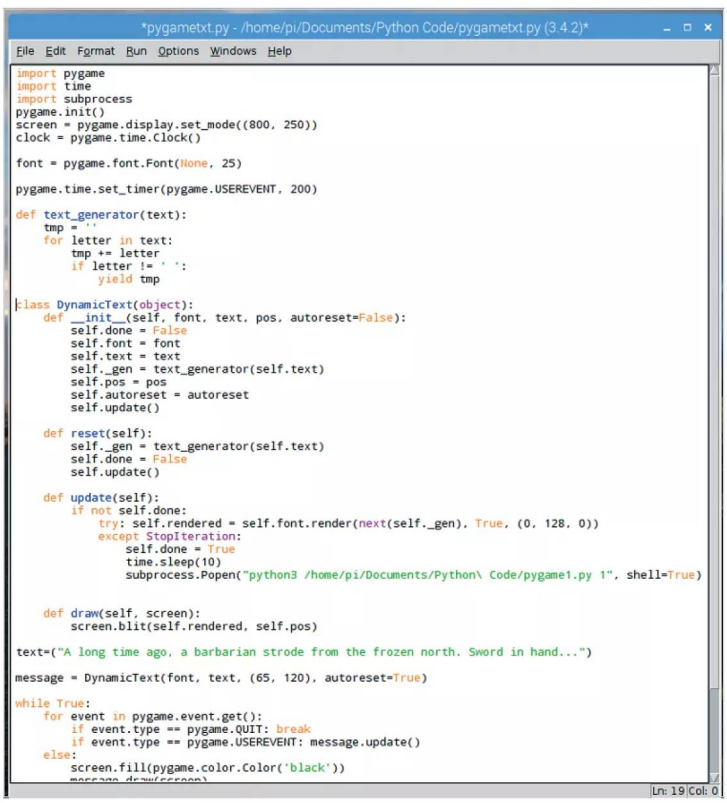
SCHRITT 8

Kopieren Sie den Code und bewegen Sie das Sprite mithilfe der linken und rechten Pfeiltasten auf der Tastatur über den unteren Bildschirmrand umher. Sie sind nun auf dem besten Weg, einen klassischen Arcade-2D-Scroller zu kreieren.

SCHRITT 9

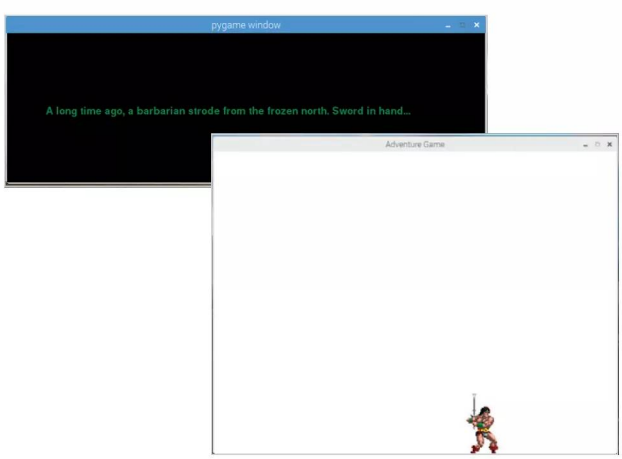
Sie können nun ein paar Ergänzungen implementieren und vom zuvor verwendetem Tutorial-Code Gebrauch machen. Die neuen Elemente sind das Unterprozessmodul, von dem eine Funktion es uns ermöglicht, ein zweites Python-Script innerhalb eines anderen zu starten. Ferner erstellen wir eine neue Datei namens pygametxt.py:

```
import pygame
import time
import subprocess
pygame.init()
screen = pygame.display.set_mode((800, 250))
clock = pygame.time.Clock()

font = pygame.font.Font(None, 25)

pygame.time.set_timer(pygame.USEREVENT, 200)

def text_generator(text):
  tmp = ''
  for letter in text:
    tmp += letter
    if letter != ' ':
      yield tmp

class DynamicText(object):
  def __init__(self, font, text, pos,
autoreset=False):
    self.done = False
    self.font = font
    self.text = text
    self._gen = text_generator(self.text)
    self.pos = pos
    self.autoreset = autoreset
    self.update()

  def reset(self):
    self._gen = text_generator(self.text)
    self.done = False
    self.update()

  def update(self):
    if not self.done:
      try: self.rendered = self.font.
render(next(self._gen), True, (0, 128, 0))
      except StopIteration:
        self.done = True
        time.sleep(10)
        subprocess.Popen("python3 /home/pi/Documents/
Python\ Code/pygame1.py 1", shell=True)

  def draw(self, screen):
    screen.blit(self.rendered, self.pos)

text=("A long time ago, a barbarian strode from the
frozen north. Sword in hand...")

message = DynamicText(font, text, (65, 120),
autoreset=True)

while True:
  for event in pygame.event.get():
    if event.type == pygame.QUIT: break
    if event.type == pygame.USEREVENT: message.
update()
  else:
    screen.fill(pygame.color.Color('black'))
    message.draw(screen)
    pygame.display.flip()
    clock.tick(60)
    continue
  break

pygame.quit()
```



SCHRITT 10

Wenn Sie diesen Code ausführen, wird ein langes, schmales Pygame-Fenster mit dem Intro-Text angezeigt, der nach rechts scrollt. Nach einer Pause von zehn Sekunden wird das Python-Script für das Spiel gestartet, in dem Sie die Kriegerfigur bewegen können. Insgesamt gesehen ist der Effekt ziemlich gut, aber es gibt immer Raum für Verbesserungen.



Einfache Animation

Mit den Python-Modulen ist es relativ einfach, Formen oder animierte Grafiken zu erstellen. Animationen können in einem Code jedoch ein kniffliges Element darstellen, da es verschiedene Möglichkeiten gibt, um das gleiche Endergebnis zu erzielen.

LICHT, KAMERA, ACTION!

Das Tkinter-Modul ist ein idealer Ausgangspunkt, um Animationen in Python zu lernen. Natürlich gibt es bessere benutzerdefinierte Module, aber Tkinter eignet sich gut, um verständlich zu machen, was für den Vorgang erforderlich ist.

SCHRITT 1

Wir beginnen, indem wir die Animation eines springenden Balls erstellen. Zuerst müssen wir ein Canvas (Fenster) und den Ball erstellen:

```
from tkinter import *
import time

gui = Tk()
gui.geometry("800x600")
gui.title("Pi Animation")
canvas = Canvas(gui, width=800,height=600,
       bg='white')
canvas.pack()

ball1 = canvas.create_oval(5,5,60,60, fill='red')

gui.mainloop()
```

SCHRITT 2

Speichern Sie den Code und führen Sie ihn aus. Es erscheint ein leeres Fenster mit einer roten Kugel in der oberen linken Ecke des Fensters. Das ist zwar toll, aber einer Animation entspricht das nicht. Wir fügen deshalb den folgenden Code hinzu:

```
a = 5
b = 5

for x in range(0,100):
        canvas.move(ball1,a,b)
        gui.update()
        time.sleep(.01)
```

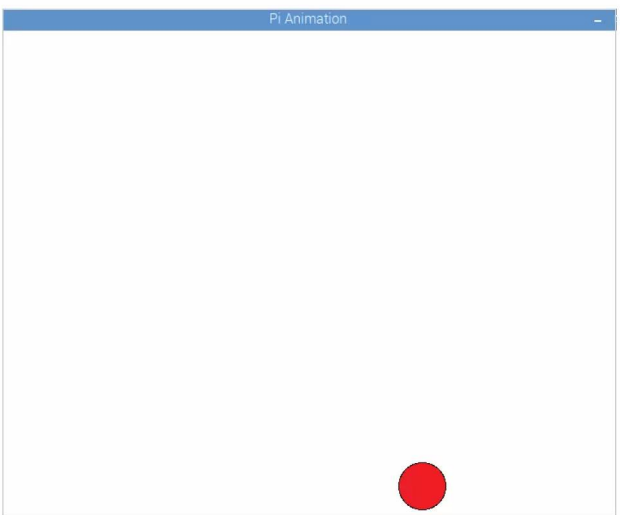
step2.py - /home/pi/Documents/step2.py (3.5.3)

File Edit Format Run Options Window Help

```
from tkinter import *
import time

gui = Tk()
gui.geometry("800x600")
gui.title("Pi Animation")
canvas = Canvas(gui, width=800, height=600, bg='white')
canvas.pack()

ball1 = canvas.create_oval(5,5,60,60, fill='red')

a = 5
b = 5

for x in range(0,100):
    canvas.move(ball1,a,b)
    gui.update()
    time.sleep(.01)

gui.mainloop()
```

SCHRITT 3

Fügen Sie den neuen Code zwischen die Zeilen `ball1 = canvas.create_oval(5,5,60,60, fill='red')` und `gui.mainloop()` ein. Speichern Sie ihn und führen Sie ihn aus. Der Ball bewegt sich nun von der oberen linken Ecke des Animationsfensters in die rechte untere Ecke. Sie können die Geschwindigkeit ändern, mit der der Ball das Fenster durchquert, indem Sie die Zeile `time.sleep(.01)` ändern. Versuchen Sie es mit (.05).



SCHRITT 4

Die Zeile `canvas.move(ball1,a,b)` ist der Teil, der den Ball von einer Ecke zur anderen bewegt, wobei sowohl a als auch b 5 entspricht. Mit der Zeile `ball1 = canvas.create_oval(5,5,60,60, fill='red')` können wir Dinge wie die Größe und Farbe des Balls ändern. Wir können auch die Werte für a und b ändern.

```
ball1 = canvas.create_oval(7,7,60,60, fill='red')

a = 8
b = 3

for x in range(0,100):
    canvas.move(ball1,a,b)
    gui.update()
    time.sleep(.05)
```

SCHRITT 5

Wir lassen nun den Ball um das Fenster herumhüpfen, bis wir das Programm schließen:

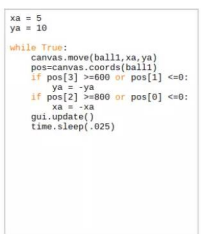
```
xa = 5
ya = 10

while True:
        canvas.move(ball1,xa,ya)
        pos=canvas.coords(ball1)
       if pos[3] >=600 or pos[1] <=0:
             ya = -ya
        if pos[2] >=800 or pos[0] <=0:
            xa = -xa
        gui.update()
       time.sleep(.025)
```

SCHRITT 6

Entfernen Sie den in Schritt 2 eingegebenen Code, und setzen Sie den Code aus Schritt 5 zwischen die Zeilen `ball1 = canvas.create_oval(5,5,60,60, fill='red')` und `gui.mainloop()` ein. Speichern Sie den Code und führen Sie ihn aus. Wurde er richtig eingegeben, wird der rote Ball von den Fensterrändern abprallen, bis Sie das Programm schließen.

SCHRITT 7

Die Animation findet innerhalb der `while True`-Schleife statt. Vor der Schleife haben wir die xa- und xy-Werte von 5 bzw. 10. Die Zeile `pos=canvas.coords(ball1)` nimmt den Wert der Ballposition im Fenster an. Werden die Grenzen des Fensters erreicht, 800 oder 600, werden die Werte negativ, und der Ball wird über den Bildschirm bewegt.



SCHRITT 8

Pygame ist allerdings ein viel besseres Modul zum Erstellen von High-End-Animationen. Erstellen Sie eine neue Datei und geben Sie Folgendes ein:

```
import pygame
from random import randrange

MAX_STARS  = 250
STAR_SPEED = 2

def init_stars(screen):
  """ Create the starfield """
  global stars
  stars = []
  for i in range(MAX_STARS):
    # A star is represented as a list with this
format: [X,Y]
    star = [randrange(0,screen.get_width() - 1),
            randrange(0,screen.get_height() - 1)]
    stars.append(star)

def move_and_draw_stars(screen):
  """ Move and draw the stars """
  global stars
  for star in stars:
    star[1] += STAR_SPEED
    if star[1] >= screen.get_height():
      star[1] = 0
      star[0] = randrange(0,639)

    screen.set_at(star,(255,255,255))
```

SCHRITT 9

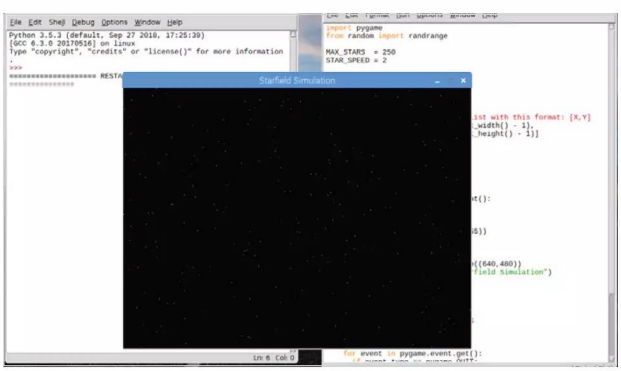
Fügen Sie nun Folgendes hinzu:

```
def main():
  pygame.init()
  screen = pygame.display.set_mode((640,480))
  pygame.display.set_caption("Starfield
Simulation")
  clock = pygame.time.Clock()

  init_stars(screen)

  while True:
    # Lock the framerate at 50 FPS
    clock.tick(50)

    # Handle events
    for event in pygame.event.get():
      if event.type == pygame.QUIT:
        return

    screen.fill((0,0,0))
    move_and_draw_stars(screen)
    pygame.display.flip()

if __name__ == "__main__":
  main()
```

```
def main():
  pygame.init()
  screen = pygame.display.set_mode((640,480))
  pygame.display.set_caption("Starfield Simulation")
  clock = pygame.time.Clock()

  init_stars(screen)

  while True:
    # Lock the framerate at 50 FPS
    clock.tick(50)

    # Handle events
    for event in pygame.event.get():
      if event.type == pygame.QUIT:
        return

    screen.fill((0,0,0))
    move_and_draw_stars(screen)
    pygame.display.flip()

if __name__ == "__main__":
  main()
```

SCHRITT 10

Speichern Sie den Code und führen Sie ihn aus. Sie werden sicherlich zustimmen, dass das simulierte Sternenfenster ziemlich beeindruckend aussieht. Stellen Sie sich vor, wie es zu Beginn eines Spielcodes oder einer Präsentation erscheint! Mit einer Kombination aus Pygame und Tkinter werden Ihre Python-Animationen fantastisch aussehen.



Eigene Module erstellen

Große Programme lassen sich viel einfacher verwalten, wenn sie in kleinere Teile zerlegt und die benötigten Teile als Module importiert werden. Indem Sie lernen, eigene Module zu erstellen, erhalten Sie auch ein besseres Verständnis über deren Funktionsweise.

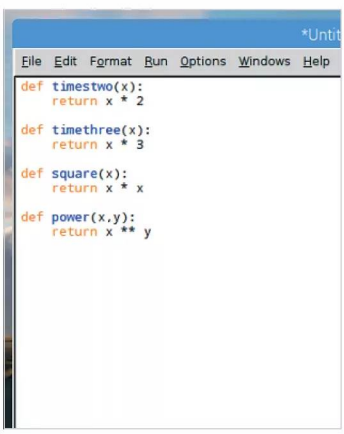
MODULE ERSTELLEN

Module sind Python-Dateien die Code enthalten und mit der .py-Endung gespeichert werden. Sie werden anschließend mit dem nunmehr bekannten import-Befehl in Python importiert.

SCHRITT 1

Wir beginnen damit, eine Reihe grundlegender mathematischer Funktionen zu erstellen. Multiplizieren Sie eine Zahl mit zwei und drei, quadrieren (square) und potenzieren Sie sie (power). Erstellen Sie eine neue Datei in der IDLE und geben Sie Folgendes ein:

```
def timestwo(x):
    return x * 2

def timesthree(x):
    return x * 3

def square(x):
    return x * x

def power(x,y):
    return x ** y
```

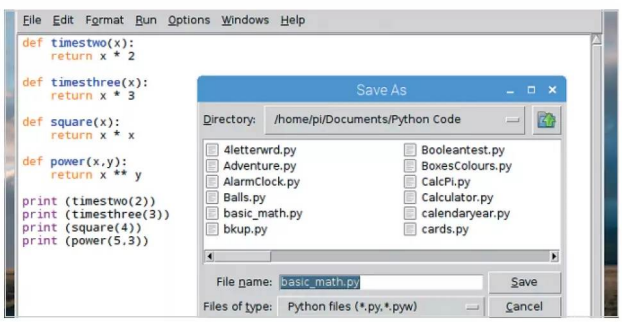


SCHRITT 2

Geben Sie unter dem obigen Code Funktionen zum Aufrufen des Codes ein:

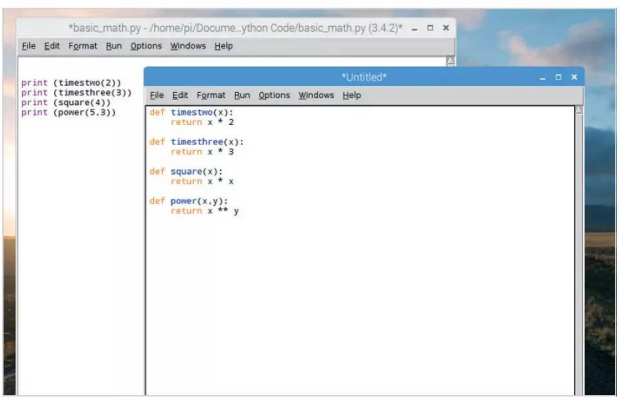
```
print (timestwo(2))
print (timesthree(3))
print (square(4))
print (power(5,3))
```

Speichern Sie das Programm unter basic_math.py und führen Sie es aus, um die Ergebnisse zu erhalten.



SCHRITT 3

Wir werden nun die Funktionsdefinitionen aus dem Programm ausschneiden und in eine separate Datei einfügen. Markieren Sie die Funktionsdefinitionen und gehen Sie zu Edit > Cut. Wählen Sie File > New File und dann im neuen Fenster Edit > Paste. Sie haben jetzt zwei separate Dateien, eine mit den Funktionsdefinitionen und die andere mit den Funktionsaufrufen.



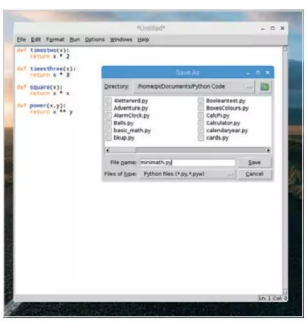
SCHRITT 4

Wenn Sie nun versuchen, den basic_math.py-Code erneut auszuführen, wird der Fehler **„NameError: name 'timestwo' is not defined"** angezeigt, da der Code keinen Zugriff mehr auf die Funktionsdefinitionen hat.

```
Traceback (most recent call last):
  File "/home/pi/Documents/Python Code/basic_math.py", line 3, in <module>
    print (timestwo(2))
NameError: name 'timestwo' is not defined
>>>
```

SCHRITT 5

Kehren Sie zum neu erstellten Fenster mit den Funktionsdefinitionen zurück und gehen Sie zu File > Save As. Speichern Sie diese Datei unter **minimath.py** und wählen Sie den gleichen Speicherort des ursprünglichen Programms **basic_math.py**. Schließen Sie nun das Fenster minimath.py; das Fenster basic_math.py bleibt geöffnet.

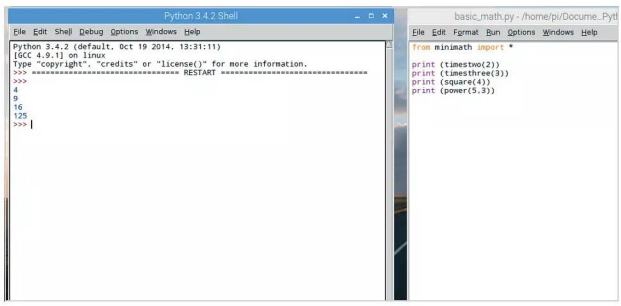


SCHRITT 6

Geben Sie im basic_math.py-Fenster ganz oben im Code Folgendes ein:

```
from minimath import *
```

Dadurch werden die Funktionsdefinitionen als Modul importiert. Drücken Sie F5, um das Programm zu speichern und auszuführen.



SCHRITT 7

Sie können mit dem Code das Programm nun etwas weiter entwickeln, indem Sie das neu erstellte Modul voll ausnutzen und einige Benutzerinteraktionen einfügen. Beginnen Sie mit der Erstellung eines Basismenüs, aus dem der Benutzer wählen kann:

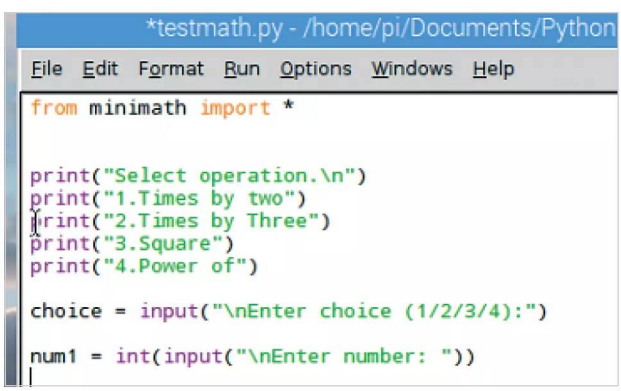
```
print("Select operation.\n")
print("1.Times by two")
print("2.Times by Three")
print("3.Square")
print("4.Power of")

choice = input("\nEnter choice (1/2/3/4):")
```

SCHRITT 8

Wir fügen nun die Benutzereingabe hinzu, um die Nummer zu erhalten, mit der der Code arbeiten wird:

```
num1 = int(input("\nEnter number: "))
```

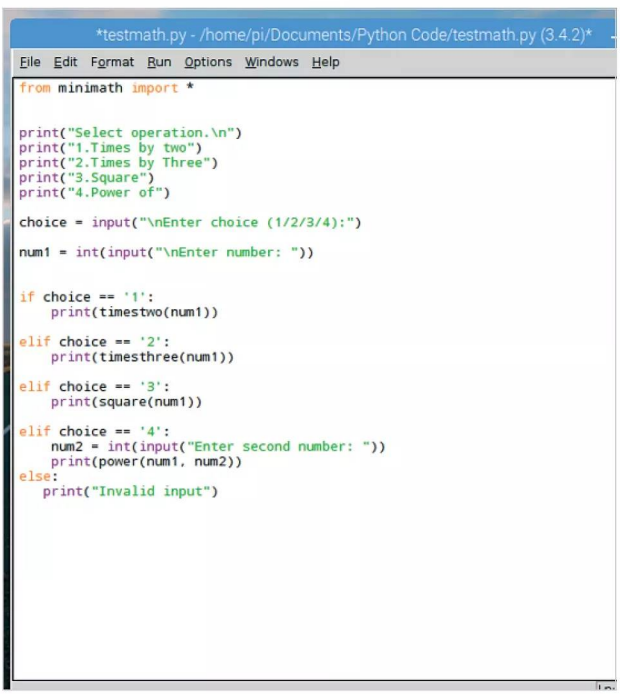
Dadurch wird die vom Benutzer eingegebene Nummer als Variable num1 gespeichert.



SCHRITT 9

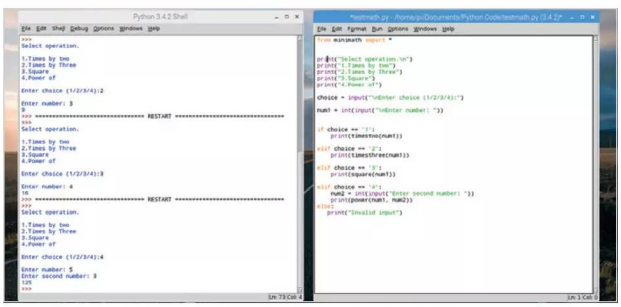
Zu guter Letzt können Sie eine Reihe von if-Anweisungen erstellen, um zu bestimmen, was mit der Zahl geschehen soll, und um von den neu erstellten Funktionsdefinitionen Gebrauch zu machen:

```
if choice == '1':
    print(timestwo(num1))

elif choice == '2':
    print(timesthree(num1))

elif choice == '3':
    print(square(num1))

elif choice == '4':
    num2 = int(input("Enter second number: "))
    print(power(num1, num2))
else:
   print("Invalid input")
```



SCHRITT 10

Beachten Sie, dass wir für die letzten verfügbaren Optionen eine zweite Variable, num2, hinzugefügt haben. Diese leitet eine zweite Zahl durch die Funktionsdefinition namens power. Speichern und führen Sie das Programm aus, um zu sehen, was passiert.

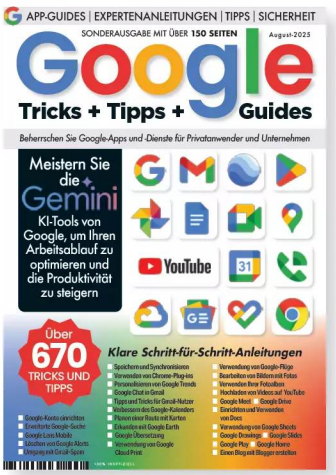


Erfahren Sie mehr über

Technologie und Software

mit unseren Expertenanleitungen auf

Readly



Papercut

www.pclpublikationen.com

Python & C++ für Einsteiger
November-2025
Herausgegeben in Großbritannien durch: Papercut Ltd
Sie können den Herausgeber dieses Magazins über die folgenden Möglichkeiten kontaktieren: C/O Papercut Ltd, Erhard-Stangl-Ring 14, 84435 Lengdorf, Bavaria.
E-Mail: enquires@pclpublications.com
Vertrieb: Readly International AB

SUPPORT ME

🙏 Hope my post useful for you, if you want support me please following one of the ways:

Buy or Renew Premium Account

👉 Rapidgator: https://rapidgator.net/account/registration/ref/49023

👉 Nitroflare: https://nitroflare.com/payment?webmaster=194862

⚠ Note: Please DON'T turn on VPN when making payment.

Donate Directly

USDT (TRC20):

TFniVipHpFsPVrUHBLsvkZJV4Mjj1MUz96

DOGE (Doge Network):

DCfVVnvNaVtxQbWyfpWsihbGnvpkuYdtJS

Every little support helps me to keep going and create more content.

THANK YOU SO MUCH!