Linux-Skripte Python C++ Windows

Januar-2024 144 Seiten

coding

tricks, guides und tipps

ÜBER 830 GUIDES & TIPPS

Der ultimative Guide zur Verbesserung Ihrer Programmierfähigkeiten und -kenntnisse

Mit vielen **Top-Tipps, Tricks und Tutorials**

- Lernen Sie die wichtigsten Grundlagen der Programmierung
- Setzen Sie Ihre Fantasie durch Programmierung frei
- Vermeiden Sie Fehler und werden Sie ein besserer Programmierer
- Erweitertes Scripting mit Windows Batch-Dateien

Holen Sie sich Ihr exklusives KOSTENLOSES Geschenk im Wert 12,00€ Hier!

Herunterladen Du bist frei Kopie von Tech Shopper Magazin

Besuchen Sie Ihren Webbrowser und befolgen Sie diese einfachen Anweisungen ...

1/ Geben Sie die folgende URL ein: **www.pclpublications.com/exclusives**
2/ Melden Sie sich an und markieren Sie in den Listen unserer exklusiven Kunden-Downloads die Option „Tech Shopper Magazine".
3/ Geben Sie Ihren individuellen Download-Code (unten aufgeführt) in die Leiste „Download-Code eingeben" ein.
4/ Klicken Sie auf „Jetzt herunterladen!". Klicken Sie auf die Schaltfläche und Ihre Datei wird automatisch heruntergeladen.
5/ Bei Ihrer Datei handelt es sich um eine hochauflösende PDF-Datei, die mit den meisten Kundengeräten/Plattformen kompatibel ist.

Exklusiver Download-Code: **PCL37862RE**

HINWEIS: Dies ist ein begrenztes Leserangebot, das vom 1. Januar 2024 bis zum 1. Juli 2024 läuft.
Es gelten die Allgemeinen Geschäftsbedingungen, die Sie unter www.pclpublications.com finden

coding

tricks, guides und tipps

Die Programmierung ist überall zu finden. Vom Anklicken eines Symbols auf dem Desktop über das Öffnen eines Webbrowsers bis hin zum Berechnen der großen Entfernungen zwischen den Sternen oder Spielen des neuesten Videospiels. Das Programmieren ist eine der wichtigsten digitalen Fähigkeiten in der modernen Welt, und sie wird noch wichtiger, wenn die neue Ära der miteinander verbundenen Geräte und Medien beginnt.

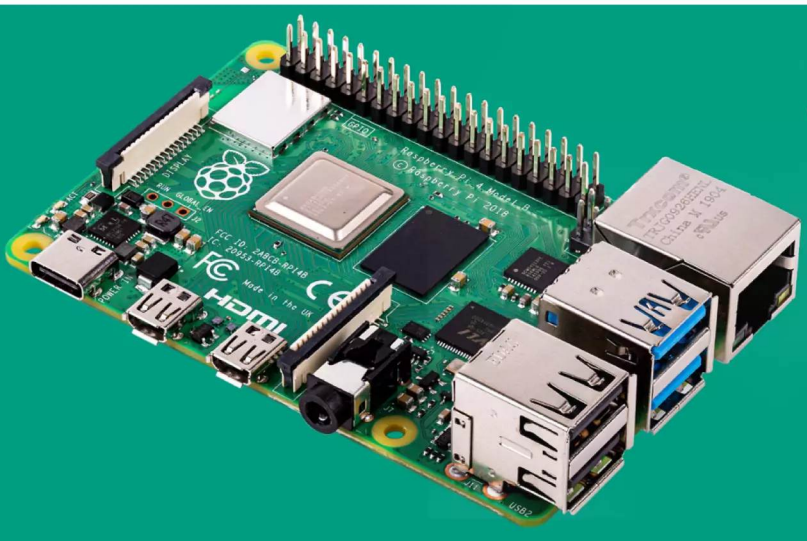
Programmieren lernen ist nicht einfach, aber wir helfen Ihnen beim Einstieg. In dieser Ausgabe lernen Sie die ersten Schritte des Programmierens mit Python, C++ und Linux-Skripten und vieles mehr. Wir gehen auch auf die häufigsten Probleme und Fehler ein, die jedem Programmierer unterlaufen können, und wie Sie diese in Zukunft vermeiden können.

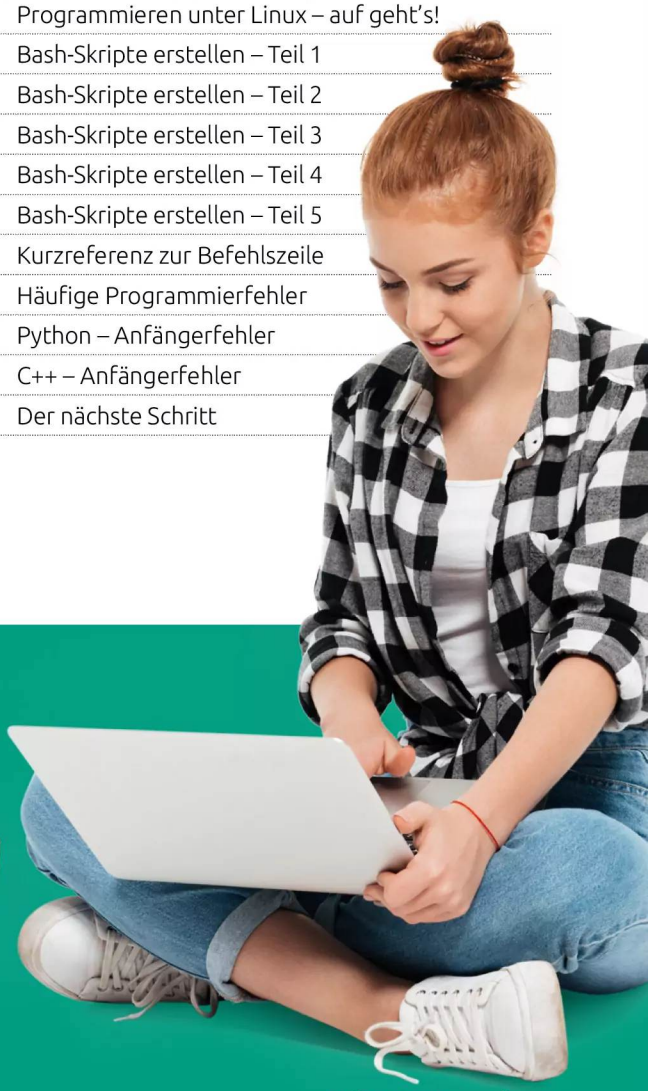
Ganz gleich ob Sie planen, am Large Hadron Collider, an Bord der Internationalen Raumstation, oder in Organisationen und Unternehmen als Cybersicherheitsspezialist zu arbeiten, oder einfach nur etwas Neues und Lustiges lernen möchten, mithilfe dieser Ausgabe werden Sie das Programmieren besser verstehen. Wir haben eine tolle Sammlung von Tutorials und Schritt-für-Schritt-Anleitungen zusammengestellt, die Ihnen bei den ersten Schritten helfen und erläutern, was Sie benötigen, um Ihre Ideen in Einsen und Nullen umzuwandeln und ein besserer Programmierer zu werden.

Das Programmieren lernen hört nie auf und Sie werden jeden Tag und jedes Mal, wenn Sie Ihren Code ausführen, etwas Neues lernen.

</>

Inhalt

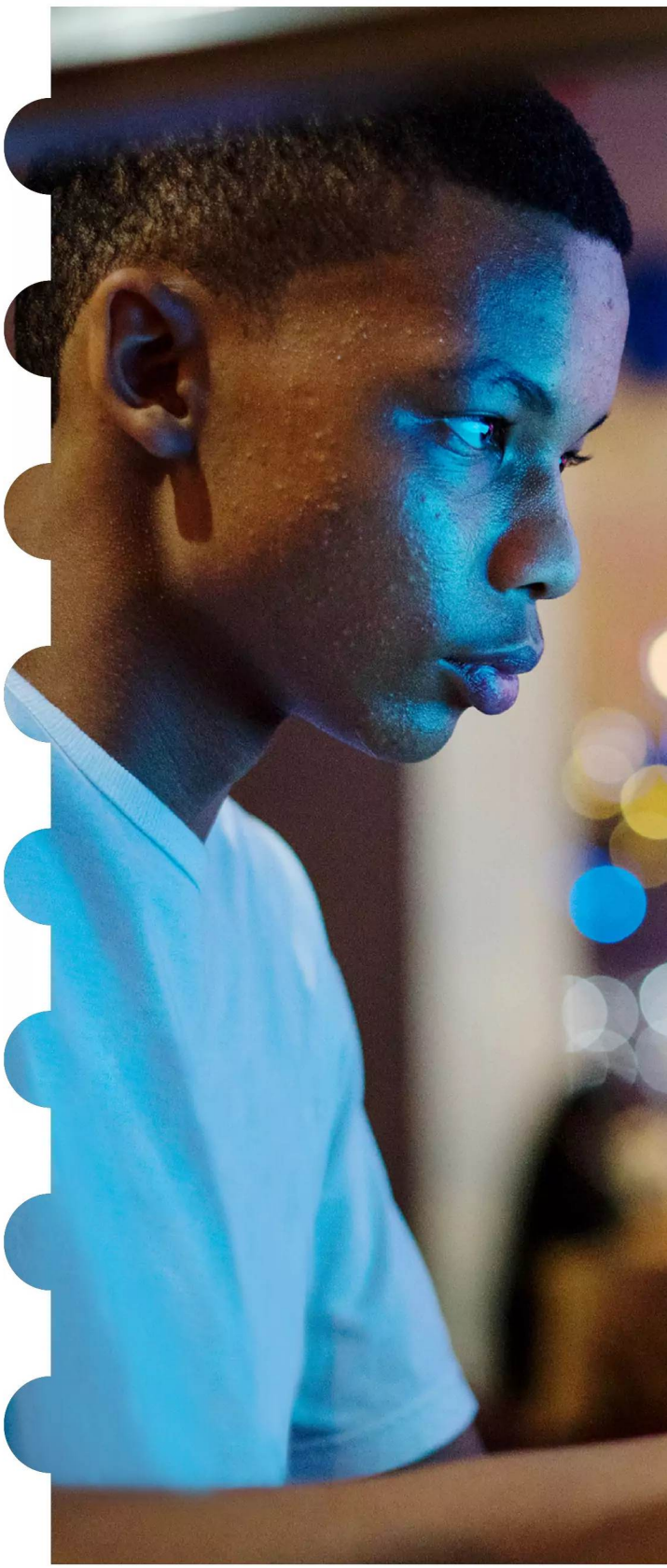




Sie möchten mit dem Programmieren beginnen? Es wird nicht einfach werden und Sie werden auch auf einige Fallgruben stoßen, aber Programmieren zu können ist eine erstaunliche Fähigkeit, die Ihnen gute Dienste leisten wird.

Der erste Schritt ist auch der schwierigste. Welche Programmiersprache sollte man wählen? Wo findet man die benötigten Tools? Und was bedeuten all diese Begriffe? Machen Sie sich keine Sorgen, wir helfen Ihnen dabei weiter.

In diesem Abschnitt erklären wir, was Sie für die ersten zaghaften Schritte in die Welt des Programmierens benötigen.



Print ("Coding für Jedermann!")

„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

Programmierer werden

Ob Programmierer, Entwickler oder Coder, all dies sind Bezeichnungen für jemanden, der Code erstellt. Dieser kann für alles mögliche erstellt werden, von einem Videospiel bis hin zu einem wichtigen Element an Bord der Internationalen Raumstation. Wie wird man aber eigentlich Programmierer?

Die Zeiten haben sich seit der Programmierung in den 80er Jahren zwar geändert, die Grundwerte sind aber erhalten geblieben.

„Es liegt an Ihnen, wie weit Sie das Abenteuer Programmieren gehen möchten."

```
#include<stdio.h>
#include<dos.h>
#include<stdlib.h>
#include<conio.h>
void getup()
{
    textcolor(BLACK);
    textbackground(15);
    clrscr();
    window(10,2,70,3);
    cprintf("Press X to Exit, Press Space to Jump");
    window(62,2,80,3);
    cprintf("SCORE : ");
    window(1,25,80,25);
    for(int x=0;x<79;x++)
    cprintf("n");
    textcolor(0);
}

int t,speed=40;
void ds(int jump=0)
{
    static int a=1;

    if(jump==0)
        t=0;
    else if(jump==2)
        t--;
    else t++;
    window(2,15-t,18,25);
    cprintf("                ");
    cprintf("          ллллллм");
    cprintf("          ллллллллл");
    cprintf("          ллллллппп");
    cprintf(" л      мллллпппп ");
    cprintf(" ллм  мллллллммм ");
    cprintf(" ллллллллллл  п ");
    cprintf("   пллллллллп     ");
    if(jump==1 || jump==2){
    cprintf("    ллп пл       ");
    cprintf("    лм   лм      ");
    }else if(a==1)
    {
    cprintf("    пллп  ппп    ");
    cprintf("      лм         ");
    a=2;
    }
    else if(a==2)
    {
    cprintf("     плм пл      ");
    cprintf("          лм     ");
    a=1;
    }
```

Einem logischen Muster folgen und das Endergebnis sehen zu können, ist eine der am meisten geschätzten Fähigkeiten eines Programmierers.

MEHR ALS CODE

Für diejenigen, die alt genug sind, um sich an die 80er Jahre zu erinnern, der goldenen Ära des Home-Computings, sah die Welt der Informatik völlig anders aus. Von 8-Bit-Computern, die man als Ganzes erwerben konnte, im Gegensatz zu Bausätzen mit Teilen, die man zusammenlöten musste, konnte man nur träumen. Einen in die Hände zu bekommen, war pure, in einem Plastikkasten verpackte Glückseligkeit. Es war aber nicht so sehr die neue Technologie der Computer die begeisterte, sondern die Tatsache, dass man zum ersten Mal kontrollieren konnte, was auf dem „Fernseher" angezeigt wurde.

Anstatt einfach eines der Tausenden verfügbaren Spiele zu spielen, beschlossen viele Benutzer, dass sie ihre eigenen Inhalte, ihre eigenen Spiele, erstellen wollten; oder einfach etwas, das ihnen bei den Hausaufgaben oder mit dem Haushaltsbudget helfen könnte. Aufgrund der Einfachheit des 8-Bit-Heimcomputers war es möglich, aus wenigen BASIC-Codezeilen etwas zu erstellen, und somit war die erste Generation von Selfmade-Programmierern geboren.

Von da an wurde die Programmierung exponentiell erweitert. Es dauerte nicht lange, bis der Heimprogrammierer der Vergangenheit angehörte und riesige Teams aus Designern, Programmierern, Künstlern und Musikern an einem einzigen Spiel beteiligt waren. Dies führte natürlich dazu, dass aus dem Programmierer mehr wurde als nur jemand, der eine Figur auf den Bildschirm brachte und diese per Tastendruck bewegen konnte.

Die Zeit ist aber nicht stehengeblieben, ebenso wenig wie die Technologien, die wir verwenden. Die Grundlagen des Programmierens ändern sich jedoch nicht. Was genau braucht man also, um Programmierer zu werden?

Eine der geläufigsten Eigenschaften eines Programmierers ist die Fähigkeit, logische Muster zu sehen. Darunter verstehen wir jemanden, der logisch etwas von Anfang bis Ende verfolgen und sich das beabsichtigte Ergebnis vorstellen kann. Während Sie sich möglicherweise nicht als solche Person fühlen, ist es möglich, seinem Gehirn beizubringen, auf diese Art zu denken. Es braucht zwar etwas Zeit, aber wenn Sie beginnen, auf diese Art und Weise nachzudenken, werden Sie in der Lage sein, Code zu erstellen und zu folgen.

Neben dem logischen Denken kommt ein Verständnis für die Mathematik. Sie müssen kein Mathe-Genie sein, müssen die Grundlagen der Mathematik jedoch verstehen. In der Mathematik geht es darum, ein Problem lösen zu können, und Code fällt meistens unter das Dach der Mathematik.

Das Gesamtbild erkennen zu können ist für den modernen Programmierer sicherlich von Vorteil. Zweifellos wird man als Programmierer Teil eines Teams von anderen Programmierern sein und höchstwahrscheinlich auch Teil eines noch größeren Teams von Designern, die alle zum Endprodukt beitragen. Während man vielleicht nur ein kleines Element dieses Endprodukts erstellt, hilft es, die Arbeiten der anderen zu verstehen, um letztendlich ein besseres Ergebnis zu erschaffen.

Zu guter Letzt benötigt ein guter Programmierer auch Kreativität. Auch hier müssen Sie kein kreatives Genie sein, sondern nur die Vorstellungskraft besitzen, um das Endprodukt sehen zu können und wie der Benutzer damit interagieren wird.

Natürlich gehört noch viel mehr dazu, ein Programmierer zu sein, u. a. das Lernen des eigentlichen Codes. Mit Zeit, Geduld und der Entschlossenheit, dazuzulernen, kann aber jeder Programmierer werden. Egal, ob Sie in einem Team für AAA-Videospiele mitmachen möchten oder nur eine automatisierte Routine erstellen wollen, die Ihre Arbeit am Computer vereinfacht, es liegt an Ihnen, wie weit Sie das Abenteuer Programmieren gehen möchten!

Eine kurze Geschichte der Programmierung

Wer glaubt, dass das Programmieren einer Maschine zur Automatisierung eines Prozesses oder zum Berechnen eines Werts ein modernes Konzept ist, das erst in den letzten fünfzig Jahren entstanden ist, irrt sich. Die Programmierung gibt es bereits seit geraumer Zeit.

01000011 01101111 01100100 01100101

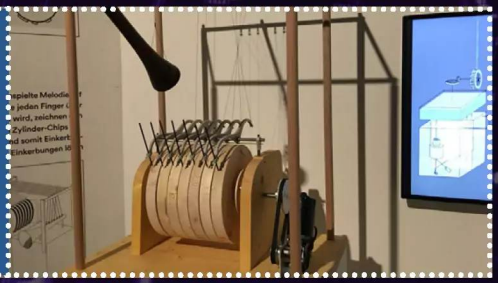
Im Wesentlichen bestehen alle Formen der Programmierung aus Einsen und Nullen, Ein oder Aus. Dies trifft auf einen modernen Computer genauso zu wie auf das älteste bekannte Rechengerät.

~87 V. CHR.

Es ist schwierig, genau zu bestimmen, wann Menschen begannen, ein Gerät zu „programmieren“. Es ist jedoch allgemein anerkannt, dass der Antikythera-Mechanismus möglicherweise das erste „programmierte” Artefakt ist. Es stammt aus dem Jahr 87 v. Chr. und ist ein altgriechischer analoger Computer und ein Orrery, mit dem astronomische Positionen vorhergesagt werden.

~850 N. CHR.

Die Banū Mūsā-Brüder, drei persische Gelehrte, die im Haus der Weisheit in Bagdad arbeiteten, veröffentlichten um 850 n. Chr. das Buch der genialen Geräte. Unter den aufgeführten Erfindungen befand sich ein mechanisches Musikinstrument, eine hydroangetriebene Orgel, die automatisch austauschbare Zylinder spielte.



1800

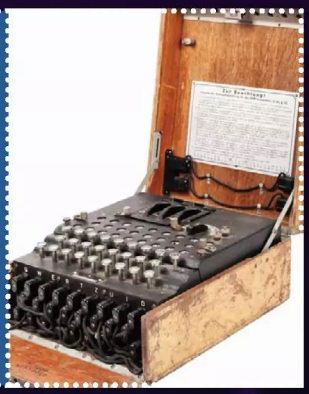
Joseph Marie Jacquard erfand einen programmierbaren Webstuhl, bei dem Karten mit gestanzten Löchern für das Textildesign verwendet wurden. Es wird jedoch angenommen, dass sein Entwurf auf einem früheren automatisierten Webprozess von Basile Bouchon aus dem Jahr 1725 basierte.

1842-1843

Ada Lovelace übersetzte die Memoiren des italienischen Mathematikers Francis Maneclang über Charles Babbages „Analytical Engine“ (engl. für Analytische Maschine). Sie machte sich beim Schreiben reichlich Notizen und beschrieb eine Methode zur Berechnung der Bernoulli-Zahlen mit der Maschine. Dies gilt als erstes Computerprogramm. Nicht schlecht, wenn man bedenkt, dass es zu dem Zeitpunkt keine Computer gab.

1930-1950

Während des 2. Weltkriegs wurden erhebliche Fortschritte bei programmierbaren Maschinen erzielt. Vor allem die kryptografischen Maschinen, mit denen Militärcodes entschlüsselt wurden. Die Enigma wurde vom deutschen Ingenieur Arthur Scherbius erfunden und von Alan Turing in Bletchley Park berühmt gemacht.



Ab den 70ern rückte die Entwicklung von C, SQL, C with Classes (C++), MATLAB, Common Lisp usw. in den Vordergrund. Die 80er waren zweifellos das goldene Zeitalter des Heim-Computers. Damals konnten Siliziumprozessoren auch von gewöhnlichen Leuten erworben werden. Das Aufkommen der 8-Bit-Maschinen führte zu einem Boom von Heim-Programmierern.

1951-1958

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2

C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 7.   START   LDS     #STACK

                        ****************************************
                        * FUNCTION: INITA - Initialize ACIA
                        * INPUT: none
                        * OUTPUT: none
                        * CALLS: none
                        * DESTROYS: acc A

0013                    RESETA  EQU     %00010011
0011                    CTLREG  EQU     %00010001

C003 86 13              INITA   LDA A   #RESETA   RESET ACIA
C005 B7 80 04                   STA A   ACIA
C008 86 11                      LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04                   STA A   ACIA

C00D 7E C0 F1           SIGNON  JMP     SIGNON    GO TO START OF MONITOR

                        ****************************************
                        * FUNCTION: INCH - Input character
                        * INPUT: none
                        * OUTPUT: char in acc A
                        * DESTROYS: acc A
                        * CALLS: none
                        * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04   INCH    LDA A   ACIA      GET STATUS
C013 47                 ASR A             SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC     INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1    GET CHAR
C019 84 7F              AND A   #$7F      MASK PARITY
C01B 7E C0 79           JMP     OUTCH     ECHO & RTS

                        ****************************************
                        * FUNCTION: INHEX - INPUT HEX DIGIT
                        * INPUT: none
                        * OUTPUT: Digit in acc A
                        * CALLS: INCH
                        * DESTROYS: acc A
                        * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR     INCH      GET A CHAR
C020 81 30              CMP A   #'0       ZERO
C022 2B 11              BMI     HEXERR    NOT HEX
C024 81 39              CMP A   #'9       NINE
C026 2F 0A              BLE     HEXRTS    GOOD HEX
C028 81 41              CMP A   #'A
C02A 2B 09              BMI     HEXERR    NOT HEX
C02C 81 46              CMP A   #'F
C02E 2E 05              BGT     HEXERR
C030 80 07              SUB A   #7        FIX A-F
C032 84 0F      HEXRTS  AND A   #$0F      CONVERT ASCII TO DIGIT
C034 39                 RTS
```

Der erste echte Computercode war die Assemblersprache (ASM). ASM war spezifisch für die Architektur der Maschine, auf der sie verwendet wurde. 1951 fielen Programmiersprachen unter den Oberbegriff Autocode. Es folgte die Entwicklung von Sprachen wie IPL, FORTRAN und ALGOL 58.

1959

Admiral Grace Hopper war Teil des Teams, das den UNIVAC I-Computer entwickelte, und sie entwickelte einen Compiler dafür. Im Laufe der Zeit wurde aus dem von ihr entwickelten Compiler COBOL (Common Business-oriented Language) eine Computersprache, die bis heute verwendet wird.

1960-1970

In den 60er und 70er Jahren wurde die Computerprogrammierung hauptsächlich von Universitäten, Militärs und großen Unternehmen eingesetzt. Ein großer Schritt in Richtung einer benutzerfreundlicheren Sprache war die Entwicklung von BASIC (Beginners All-Purpose Symbolic Instruction Code) Mitte der 60er.

```
10 INPUT "What is your name: "; U$
20 PRINT "Hello "; U$
25 REM
30 INPUT "How many stars do you want: "; N
35 S$ = ""
40 FOR I = 1 TO N
50 S$ = S$ + "*"
55 NEXT I
60 PRINT S$
65 REM
70 INPUT "Do you want more stars? "; A$
80 IF LEN(A$) = 0 THEN GOTO 70
90 A$ = LEFT$(A$, 1)
100 IF (A$ = "Y") OR (A$ = "y") THEN GOTO 30
110 PRINT "Goodbye ";
120 FOR I = 1 TO 200
130 PRINT U$; " ";
140 NEXT I
```

1970-1985

Seit den1990ern

Die Internet-Ära brachte viele neue Programmiersprachen mit sich und ermöglichte den Zugang zu den Tools und Kenntnissen, die für ein besseres Lernen der Programmierung erforderlich sind. Der Benutzer lernte, wie er Code erstellt, freigibt und nach anderem Code sucht, um seinen eigenen zu verbessern.

Die Wahl der Programmier-sprache

Es ist unmöglich, jede Programmiersprache in einem einzigen Magazin dieser Größe ausführlich zu erklären. Es werden fast täglich neue Sprachen und Möglichkeiten entwickelt, um mit einem Computer oder Gerät zu „sprechen“ und ihm Anweisungen zu geben, und mit dem Aufkommen des Quanten-Computings werden immer komplexere Methoden entwickelt.
Hier ist eine Liste der häufigsten Sprachen und deren wichtigsten Funktionen.

SQL SQL steht für Structured Query Language. SQL ist eine Standardsprache für den Zugriff auf Datenbanken und deren Bearbeitung. Obwohl SQL ein ANSI-Standard (American National Standards Institute) ist, gibt es verschiedene Versionen der SQL-Sprache. Zur Kompatibilität unterstützen sie jedoch alle auf ähnliche Weise die wichtigsten Befehle wie z. B. Auswählen, Aktualisieren und Löschen.

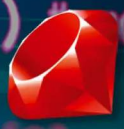
JAVASCRIPT JavaScript (oft als JS abgekürzt) ist eine leichtgewichtige, interpretierte, objektorientierte Sprache mit erstklassigen Funktionen. JavaScript wird auf der Client-Seite des Webs ausgeführt, mit der das Verhalten der Webseiten beim Auftreten eines Ereignisses programmiert werden kann. JavaScript ist eine einfach zu erlernende und auch leistungsfähige Skriptsprache, die häufig zur Steuerung des Webseitenverhaltens verwendet wird.

JAVA Java ist die Grundlage für praktisch jede Art von Netzwerkanwendung und ist der globale Standard für die Entwicklung von Unternehmenssoftware, webbasierten Inhalten, Spielen und mobilen Apps. Die zwei Hauptkomponenten der Java-Plattform sind das Java Application Programming Interface (API) und die Java Virtual Machine (JVM), die Java-Code in Maschinensprache übersetzt.

C# C# ist eine elegante objektorientierte Sprache, mit der Entwickler eine Vielzahl sicherer und robuster Anwendungen erstellen können, die auf dem .NET Framework ausgeführt werden. Mit C# können Sie Windows-Client-Anwendungen, XML-Webdienste, Client-Server-Anwendungen, Datenbankanwendungen und vieles mehr erstellen. Die geschweifte Klammersyntax von C# ist für jeden, der mit C, C++ oder Java vertraut ist, sofort erkennbar.

PYTHON Python ist eine weit verbreitete höhere Programmiersprache, die von Guido van Rossum für allgemeine Zwecke entwickelt und 1991 erstmals veröffentlicht wurde. Als interpretierte Sprache hat Python eine Designphilosophie, welche die Lesbarkeit von Code betont, und eine Syntax, die es Programmierern ermöglicht, Konzepte in weniger Codezeilen auszudrücken, was neuen Programmierern das Lernen erleichtern kann.

C++ C++ ist eine universelle Programmiersprache. Sie verfügt über zwingende, objekt-orientierte und generische Programmierfunktionen. Sie wurde mit Hinblick auf die Systemprogrammierung und auf eingebettete, ressourcenbeschränkte und große Systeme konzipiert, wobei Leistung, Effizienz und Flexibilität in der Anwendung als Höhepunkte des Designs hervorgehoben wurden.



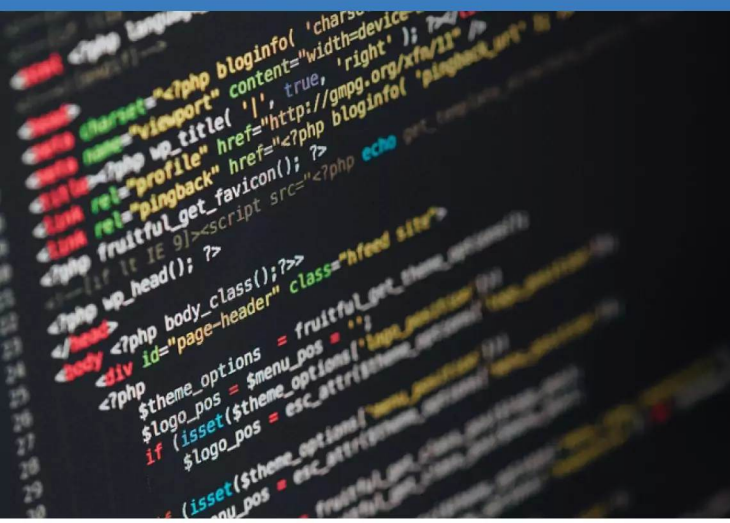
RUBY Ruby ist eine Sprache der Balance. Ihr Schöpfer, Yukihiro „Matz" Matsumoto, kombinierte Teile seiner Lieblingssprachen (Perl, Smalltalk, Eiffel, Ada und Lisp), um eine neue Programmiersprache zu formen. Seit ihrer Veröffentlichung 1995 hat Ruby weltweit engagierte Programmierer angezogen. Ruby gilt als flexible Sprache. Zentrale Teile von Ruby können nach Belieben entfernt oder neu definiert werden, während vorhandene Teile erweitert werden können.

PERL Perl ist eine universelle Sprache, die für eine Vielzahl von Aufgaben verwendet wird, u. a. die Systemverwaltung, Webentwicklung, Netzwerkprogrammierung und GUI-Entwicklung. Zu den Hauptmerkmalen zählt, dass sie einfach ist, sowohl prozedurale als auch objektorientierte Programmierung unterstützt, eine leistungsfähige integrierte Unterstützung für die Textverarbeitung bietet und über eine der beeindruckendsten Sammlungen von Drittanbieter-Modulen verfügt.

SWIFT Swift ist eine leistungsstarke und intuitive Sprache für macOS, iOS, watchOS und tvOS. Swift-Code zu schreiben ist interaktiv und macht Spaß. Die Syntax ist kurz und zugleich ausdrucksstark. Swift enthält moderne Funktionen, die unter Entwicklern beliebt sind. Swift-Code ist von Grund auf sicher und erzeugt blitzschnelle Software. Auf dem iPad ist Swift Playgrounds verfügbar, eine App, die als Anleitung zum Programmieren dient.

Erstellen einer Programmierplattform

Eine Programmierplattform kann Folgendes sein: eine Art von Hardware, auf der man programmieren kann, ein bestimmtes Betriebssystem oder sogar eine benutzerdefinierte Umgebung, die für die einfache Erstellung von Spielen vorab erstellt und entwickelt wurde. Es ist ein recht loser Begriff, da sie eine Mischung aus all dem sein kann und es hängt halt von Ihren Zielen und Ihrer bevorzugten Programmiersprache ab.



Das Programmieren kann eine dieser Erfahrungen sein, die sich fantastisch anhört, aber oft verwirrend ist, da eine Reihe von Sprachen zur Auswahl stehen sowie unzählige Apps, mit denen Sie in einer bestimmten oder in einer Reihe von Sprachen programmieren können, und genauso viel Software von Drittanbietern. Wenn Sie auf das Internet zugreifen, werden Sie feststellen, dass für die Sprache, in der Sie programmieren möchten, unzählige Programmieranleitungen zur Verfügung stehen, zusammen mit vielen weiteren Codebeispielen. Zu Beginn ist das alles etwas überwältigend.

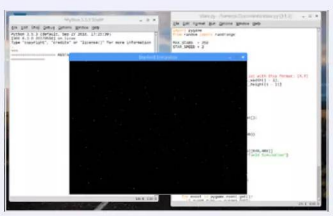
Der Trick ist, einen Gang runterzuschalten und zunächst nicht zu tief ins Programmieren einzutauchen. Sie benötigen eine solide Basis, auf der Sie Ihre Fähigkeiten aufbauen können, sowie die erforder-lichen Tools, mit denen Sie die grundlegenden Schritte ausführen können. Hier kommt das Erstellen einer Programmierplattform ins Spiel. Sie wird Ihre Lernbasis sein, wenn Sie Ihre ersten Schritte in die Welt des Programmierens wagen.

HARDWARE

Zum Lernen der Programmierung wird keine spezielle Ausrüstung oder ein Spitzencomputer benötigt. Ein Computer, und sei er ein noch so einfaches Modell, reicht aus. Sollte es sich bei dem fraglichen Computer natürlich um einen Commodore 64 handeln, könnten Sie Schwierigkeiten haben, einem modernen Sprachtutorial zu folgen; andererseits haben einige der besten Programmierer der heutigen Zeit auf 8-Bit-Computern begonnen.

Sie benötigen Zugang zum Internet, um die Entwicklungsumgebung für die Programmierung herunterzuladen, zu installieren und zu aktualisieren, sowie einen Computer mit Windows 10, macOS oder Linux. Andere Betriebssysteme gehen zwar auch, aber die meisten Coderessourcen wurden für eines oder alle dieser drei Betriebssysteme geschrieben.

SOFTWARE



In Bezug auf die Software verfügen die meisten Entwicklungsumgebungen über Tools, mit denen Sie Code schreiben, kompilieren und ausführen können, und die kostenlos heruntergeladen und installiert werden können. Einige spezielle Tools sind kostenpflichtig, für den Einstieg aber nicht erforderlich. Sie müssen also keine zusätzliche Software kaufen, um das Programmieren zu erlernen.

Mit der Zeit werden Sie vielleicht die gängige Entwicklungsumgebung hinter sich lassen und eine Sammlung Ihrer selbst entdeckten Tools zum Code schreiben verwenden. Letztendlich ist es eine Frage der persönlichen Präferenz und mit zunehmender Erfahrung werden Sie auch verschiedene Tools für die jeweiligen Aufgaben verwenden.

BETRIEBSSYSTEME

Windows 10 ist das am häufigsten verwendete Betriebssystem der Welt. Daher ist es selbstverständlich, dass die meisten Tools zum Programmieren für das führende Betriebssystem von Microsoft geschrieben wurden. Sie sollten macOS und insbesondere Linux aber nicht abschreiben.

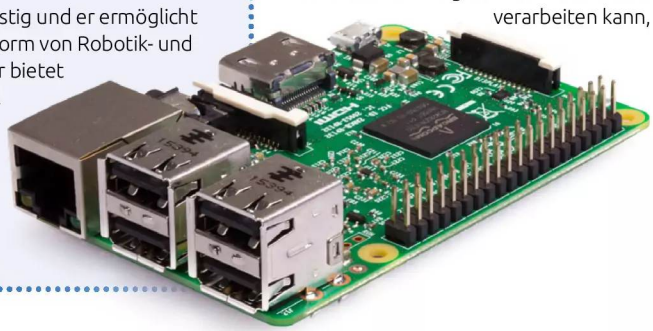
macOS-Benutzer verfügen über die gleiche Anzahl von Programmier-Tools wie Windows-Benutzer. Viele professionelle Programmierer verwenden sogar einen Mac anstelle eines PCs, weil das Mac-Betriebssystem auf Unix basiert (dem Betriebssystem auf Befehlszeilenbasis, das weltweit einen Großteil der Dateisysteme und Server antreibt). Mit dieser Unix-Ebene können Programme auch ohne spezielle IDE in nahezu jeder Sprache getestet werden.

Linux ist jedoch mit Abstand eines der beliebtesten und wichtigsten Betriebssysteme zum Programmieren. Es verfügt über ein Unix-ähnliches Backbone, kann kostenlos heruntergeladen, installiert und verwendet werden und enthält die meisten Tools, die zum Programmieren lernen erforderlich sind. Linux ist die treibende Kraft hinter den meisten Servern im Internet. Es wird auf fast allen Supercomputern sowie speziell in Organisationen wie der NASA, dem CERN und dem Militär eingesetzt und bildet auch die Basis für Android-Geräte, Smart-TVs und Kfz-Systeme. Linux ist eine hervorragende Programmierplattform und kann in einer virtuellen Maschine ohne Auswirkung auf die Windows- oder macOS-Installation installiert werden.

RASPBERRY PI

Wenn Sie noch nie vom Raspberry Pi gehört haben, werfen Sie einen Blick auf die Website **www.raspberrypi.org**. Der Raspberry Pi ist ein kleiner, voll funktionsfähiger Computer. Es wird mit einem eigenen angepassten Linux-basierten Betriebssystem geliefert, auf dem alles vorinstalliert ist, was Sie zum Programmieren lernen in Python, C++, Scratch usw. benötigen.

Für nur ca. 40 € ist er unglaublich günstig und er ermöglicht es Ihnen, verschiedene Hardware in Form von Robotik- und Elektronikprojekten zu nutzen. Ferner bietet er ein umfassendes Desktop-Erlebnis. Obwohl der Raspberry Pi nicht der leistungsstärkste Computer der Welt ist, hat er eine Menge zu bieten und ist eine der besten verfügbaren Programmierplattformen.



VIRTUELLE MASCHINEN

Eine virtuelle Maschine ist eine Software, mit der Sie ein voll funktionsfähiges Betriebssystem innerhalb der Software selbst installieren können. Das installierte Betriebssystem weist dem Host-Computer benutzerdefinierte Ressourcen zu, stellt Speicher, Festplattenspeicher usw. bereit und teilt die Internetverbindung des Host-Computers.

Der Vorteil einer virtuellen Maschine ist, dass Sie mit Linux arbeiten können, ohne dass sich dies auf Ihr aktuell installiertes Host-Betriebssystem auswirkt. Dies bedeutet, dass neben der Ausführung und Nutzung von Windows 10 die virtuelle Maschine gestartet werden kann, auf der Linux geladen und alle Linux-Funktionen verwendet werden können.

Dies macht sie natürlich zu einer fantastischen Programmierplattform, da Sie verschiedene Betriebssysteme auf dem Host-Computer ausführen können, während Sie verschiedene Programmiersprachen verwenden. Sie können Ihren Code testen, ohne befürchten zu müssen, Ihr Host-Betriebssystem zu beschädigen, und problemlos zu einer vorherigen Konfiguration zurückkehren, ohne alles neu installieren zu müssen.

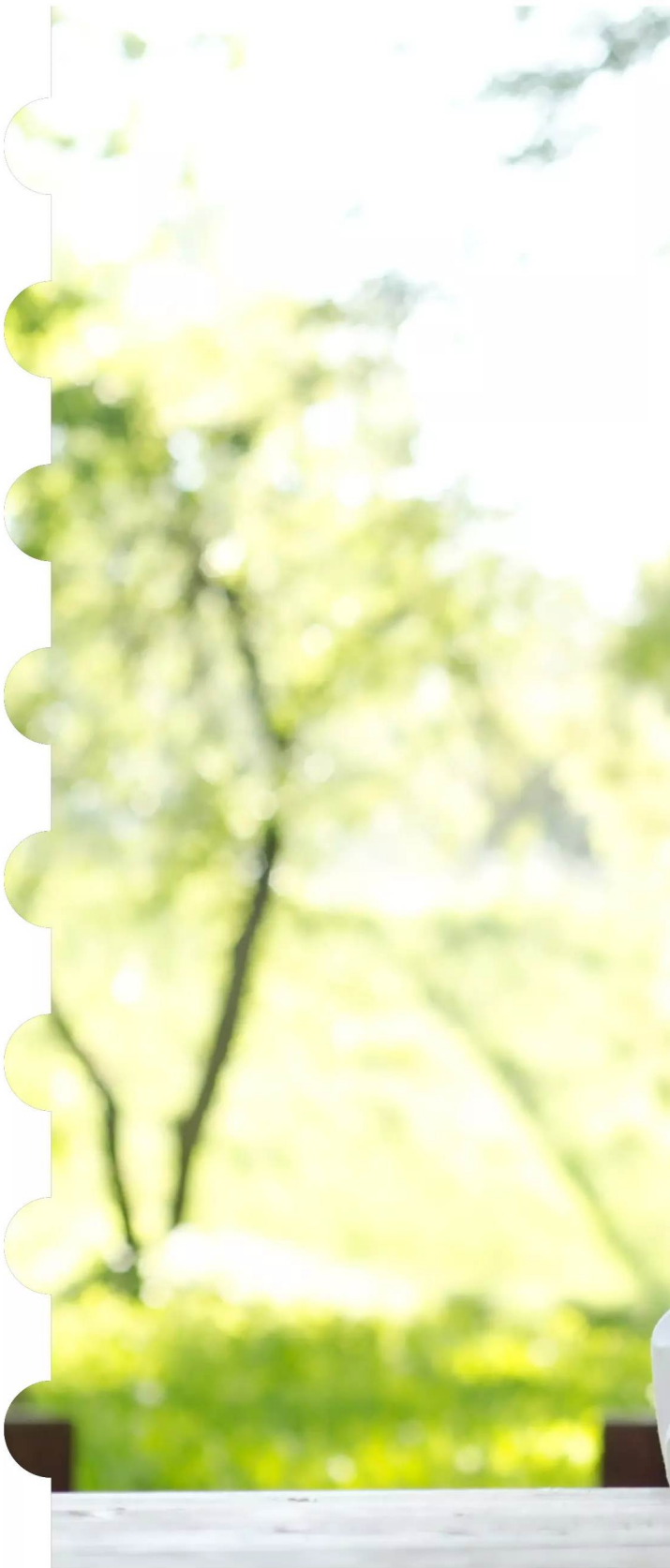
Virtualisierung ist für viele große Unternehmen ein Schlüsselfaktor. Sie nehmen statt eines einzelnen Servers für Windows eine virtuelle Umgebung, bei der jede Windows Server-Instanz eine virtuelle Maschine ist, die auf mehreren leistungsstarken Rechnern ausgeführt wird. Dies reduziert die Anzahl der physischen Rechner und ermöglicht dem IT-Team eine bessere Verwaltung der Ressourcen und die Nutzung eines gesamten Servers für eine bestimmte Aufgabe, was letztendlich Zeit einspart.

IHRE EIGENE PROGRAMMIERPLATTFORM

Ganz gleich welche Methode Sie wählen, denken Sie daran, dass sich Ihre Programmierplattform wahrscheinlich ändern wird, wenn Sie an Erfahrung sammeln und eine Sprache einer anderen vorziehen. Scheuen Sie sich auch nicht zu experimentieren. Sie werden letztendlich eine eigene Plattform erstellen, die den gesamten Code verarbeiten kann, den Sie eingeben.

Python wurde erstmals vor dreißig Jahren eingeführt und hat seitdem viele Änderungen und Verbesserungen erfahren. Die Python-Version 3.x wurde 2008 veröffentlicht und ist seitdem die bevorzugte Programmiersprache für KI, Big Data und die wissenschaftliche Gemeinschaft. Python ist eine einfach zu erlernende, aber dennoch leistungsstarke Sprache, mit der Sie Ihrem Ziel, Entwickler zu werden, einen Schritt näherkommen.

In diesem Abschnitt erfahren Sie, was benötigt wird und wie Sie Python auf Ihrem System installieren können.



Einführung in Python

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

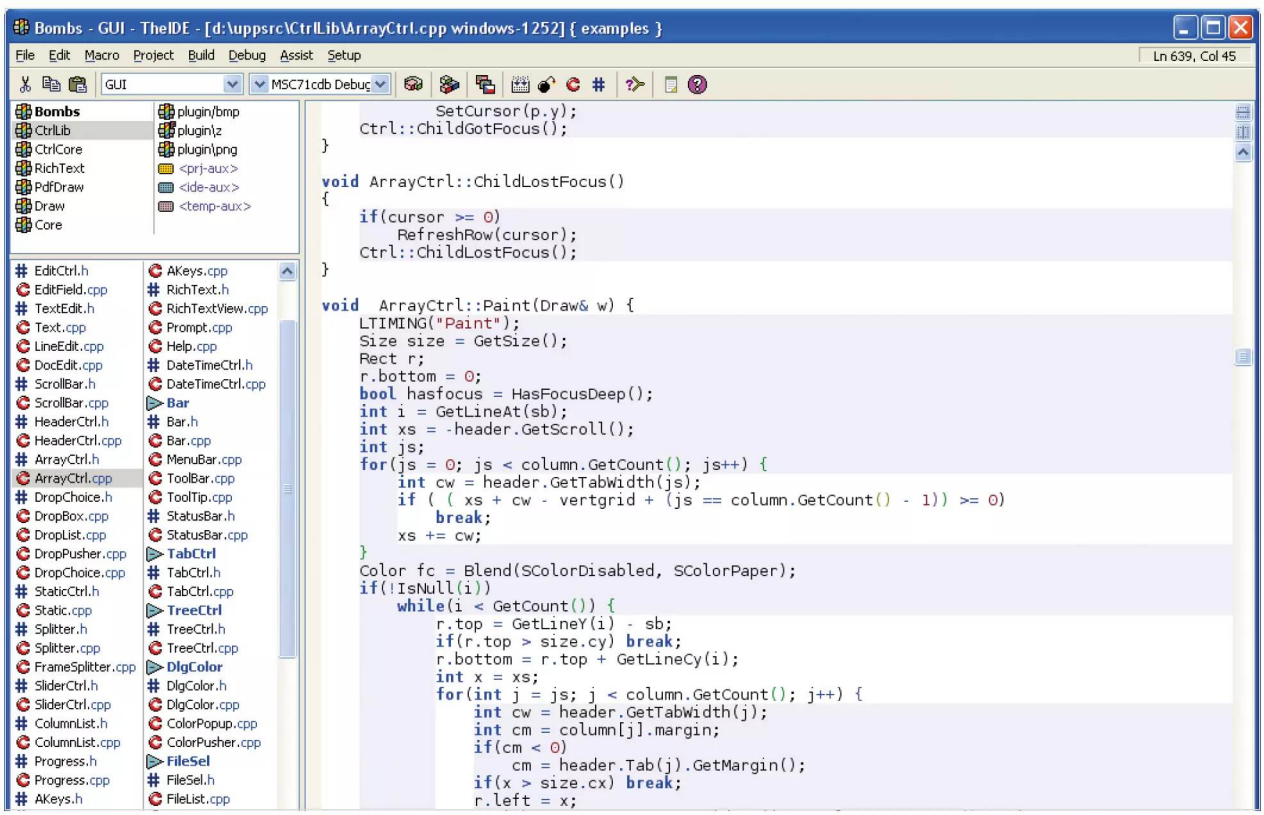
Seitdem die ersten Heimcomputer erhältlich waren, haben Enthusiasten, Benutzer und Profis bis in die frühen Morgenstunden gearbeitet und sich mit überhitzten Schaltkreisen herumgeplagt, um etwas zu erschaffen, das an Zauberei grenzt.

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 genial.

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, sie basieren alle 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.



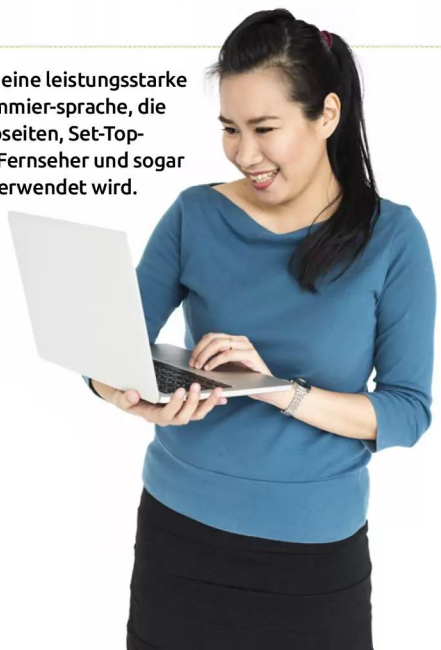
Die heute verwendeten Programmiersprachen unterscheiden sich je nach Situation, Plattform, Verwendung des Geräts und nach dem, wie das Gerät mit seiner Umgebung oder seinen Benutzern kommuniziert. Betriebssysteme wie Windows, macOS usw. sind in der Regel eine Kombination aus C++, C#, Assembly und einer 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   }
26 }
```

Java ist eine leistungsstarke Programmier-sprache, 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, Studierenden, 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
```

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.

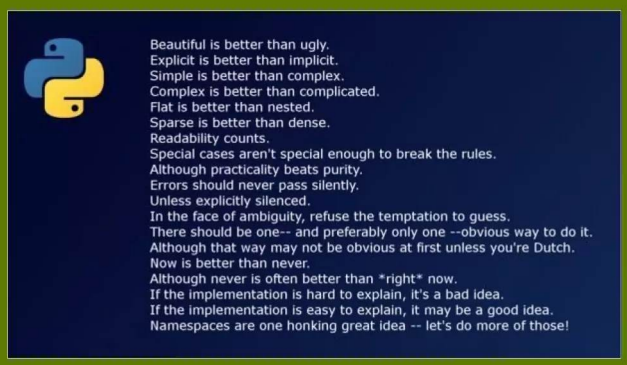
Was kann man mit Python alles machen?

Python ist eine objektorientierte Open-Source-Programmiersprache, die einfach zu verstehen und zu schreiben ist, aber auch leistungsstark und äußerst formbar ist. Diese Eigenschaften machen Python zu einer wichtigen Programmiersprache.

Die Fähigkeit von Python, mit wenigen Anweisungen gut lesbaren Code zu erstellen, hat erhebliche Auswirkungen auf unsere moderne digitale Welt. Von der idealen ersten Wahl für Programmierer bis hin zur Möglichkeit, interaktive Geschichten und Spiele zu erstellen; von wissenschaftlichen Anwendungen über künstliche Intelligenz bis hin zu webbasierten Anwendungen, die einzige Grenze für Python liegt in der Vorstellungskraft des Programmierers.

Pythons formbares Design macht es zu einer idealen Sprache für viele verschiedene Situationen und Aufgaben. Auch für bestimmte Aspekte beim Programmieren, für die effizienterer Code erforderlich ist, wird Python eingesetzt. Z. B. nutzt die NASA Python sowohl als eigenständige Sprache als auch als Brücke zwischen anderen Programmiersprachen. Auf diese Weise können Wissenschaftler und Ingenieure der NASA auf Daten zugreifen, die sie benötigen, ohne mehrere Sprachbarrieren überwinden zu müssen. Python füllt diese Barrieren und bietet die Mittel, eine Aufgabe erledigt zu bekommen.

Es gibt viele Beispiele, in denen Python hinter den Kulissen agiert, weshalb es eine so wichtige Sprache ist, die es zu erlernen gilt.

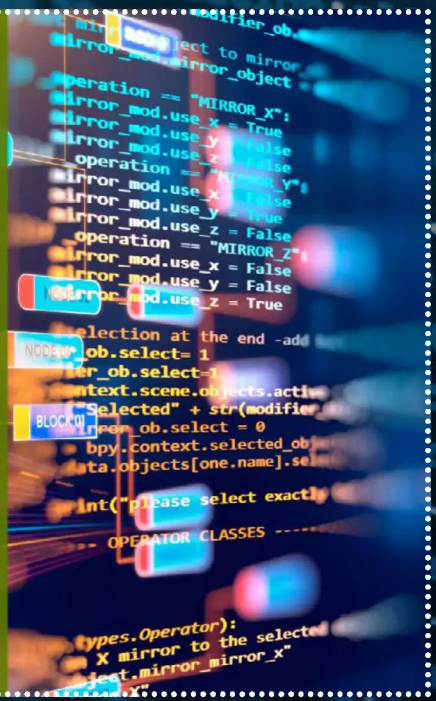


BIG DATA

Big Data steht für extrem große Datenmengen, die zur Analyse zur Verfügung stehen, um Muster, Trends und Interaktionen zwischen Mensch, Gesellschaft und Technologie aufzudecken. Natürlich ist dies nicht nur auf diese Bereiche beschränkt. Big Data wird derzeit in einer Vielzahl von Branchen verwendet, von sozialen Medien, Gesundheit und Wohlfahrt über Ingenieurwesen bis hin zur Weltraumforschung und darüber hinaus.

Python spielt eine wichtige Rolle in der Welt von Big Data. Mit Python werden große Teile dieser verfügbaren Datenmengen analysiert und spezifische Informationen basierend auf den Anforderungen des Benutzers/Unternehmens aus der Fülle der vorhandenen Zahlen extrahiert. Dank einer beeindruckenden Sammlung von Datenverarbeitungsbibliotheken kann Python die Daten in einer für den Menschen les- und nutzbaren Form darstellen.

Es gibt unzählige Bibliotheken und kostenlos erhältliche Module, die eine schnelle, sichere und vor allem genaue Verarbeitung von Daten-Cluster aus Supercomputern ermöglichen. Das CERN verwendet z. B. ein benutzerdefiniertes Python-Modul, um die 600 Millionen Kollisionen pro Sekunde zu analysieren, die der Large Hadron Collider (LHC) erzeugt. Eine andere Sprache verarbeitet zwar die Rohdaten, aber Python hilft beim Durchsuchen der Daten, damit Wissenschaftler zu den gewünschten Inhalten gelangen, ohne eine wesentlich komplexere Programmiersprache erlernen zu müssen.



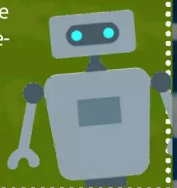
KÜNSTLICHE INTELLIGENZ (KI)

KI und maschinelles Lernen sind zwei der bahnbrechendsten Aspekte der modernen Informatik. KI ist der Überbegriff für jeden Computerprozess, bei dem die Maschine etwas Intelligentes ausführt und ähnlich wie der Mensch arbeitet und reagiert. Maschinelles Lernen ist eine Untergruppe der KI und bietet dem gesamten KI-System die Möglichkeit, aus seinen Erfahrungen zu lernen.

Bei KI geht es jedoch nicht einfach um die Erstellung autonomer Roboter, die die menschliche Zivilisation ausrotten wollen. KI kann in einer Vielzahl von alltäglichen Computeranwendungen verwendet werden, bei denen die „Maschine" bzw. der Code aus den Aktionen einer bestimmten Form von Eingabe lernen und vorhersagen muss, was die Eingabe wahrscheinlich erfordert oder als Nächstes ausführt.

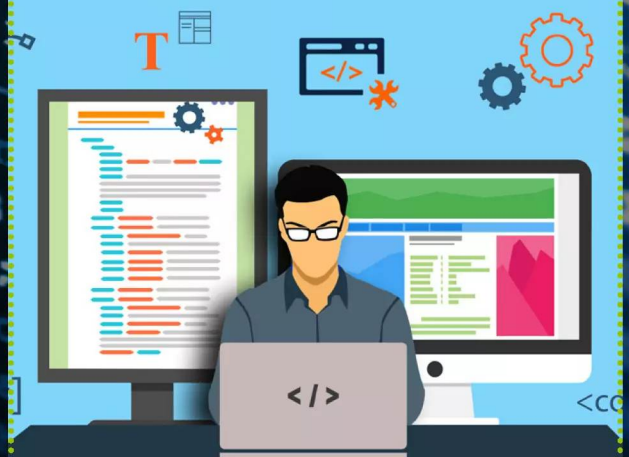
Dieses Modell kann auch auf soziale Medien übetragen werden. Haben Sie schon einmal auf Instagram nach einem Promi gesucht und bemerkt, dass Ihre Suchanfragen in anderen Social Media-Plattformen jetzt auf ähnliche Promis ausgerichtet sind? Dies ist ein erstklassiges Beispiel für die Verwendung von KI in gezielter Werbung. Hinter dem Code und den Algorithmen, die vorhersagen, wonach Sie suchen, verbirgt sich Python.

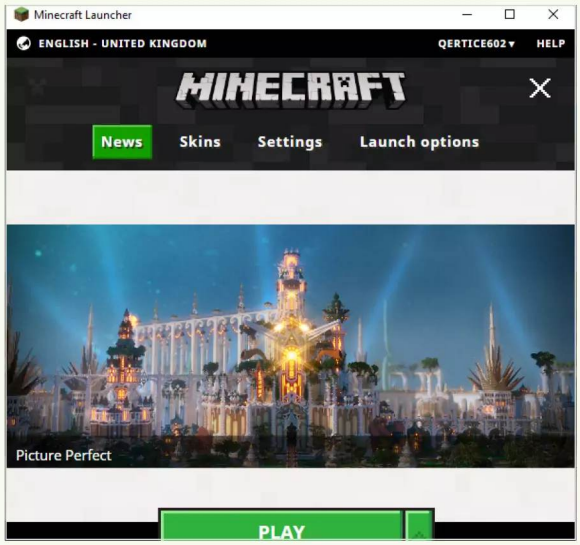
Spotify z. B. analysiert mit Python-basierten Code Ihre Musikgewohnheiten und bietet Wiedergabelisten an, die auf dem basieren, was Sie in der Vergangenheit gehört haben. All dies macht Python zum Vorreiter bei der zukünftigen Funktionsweise des Internets.



WEBENTWICKLUNG

Die Webentwicklung hat sich seit den Anfängen der HTML-Skripterstellung im eingeschränkten Texteditor erheblich weiterentwickelt. Die vielen Frameworks und Web-Verwaltungsdienste, die nunmehr verfügbar sind, haben das Erstellen einer Seite komplexer gemacht. Mit Python kann der Webentwickler dynamische und äußerst sichere Web-Apps erstellen, die die Interaktion mit anderen Webdiensten und Apps wie Instagram und Pinterest ermöglichen. Python ermöglicht auch das Sammeln von Daten von anderen Websites und auch von Apps, die auf anderen Websites erstellt wurden.





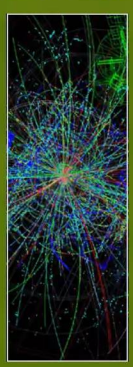
GAMING

Obwohl Sie nicht viele AAA-Spiele finden werden, die mit Python programmiert wurden, werden Sie überrascht sein, dass Python bei vielen hochrangigen modernen Spielen als Extra eingesetzt wird.

Die Hauptanwendung von Python beim Spielen liegt in der Skripterstellung, bei der ein Python-Skript Anpassungen am Kern der Spiel-Engine vornehmen kann. Viele Karteneditoren sind Python-kompatibel, und Sie werden auch darauf stoßen, wenn Sie Modifizierungen für Spiele wie Die Sims erstellen.

Viele der Online-MMORPG-Spiele (Massively Multiplayer Online Role-Playing Game) verwenden Python als Begleitsprache für die Server-Elemente. Dazu gehören Code zur Suche nach potenziellem Betrug, zum Lastausgleich auf den Servern des Spiels, zum Matchmaking der Spielerfähigkeiten und zum Überprüfen, ob das clientseitige Spiel des Spielers mit den Serverversionen übereinstimmt. Es gibt auch ein Python-Modul, das in einen Minecraft-Server integriert werden kann, mit dem der Serveradministrator Blöcke hinzufügen, Nachrichten senden und viele der Hintergrundkomplexitäten des Spiels automatisieren kann.

PYTHON ÜBERALL



Wie Sie sehen, ist Python eine recht vielseitige Programmiersprache. Indem Sie Python lernen, erhalten Sie umfassende Kenntnisse, mit denen Sie beruflich oder einfach als Hobby in die nächste Generation von Informatikern einsteigen können.

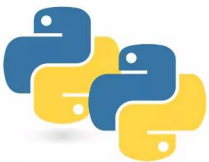
Ganz gleich, welchen Pfad Sie auf Ihrer Reise durch das Programmieren einschlagen, Python sollte auf jeden Fall dabei sein.

PYTHON in ZAHLEN

Python hat jede Menge zu bieten. Hier sind einige erstaunliche Fakten und Zahlen zu einer der beliebtesten Programmiersprachen der letzten Jahre.

Guido Van Rossum, der Entwickler von Python, kam auf den Namen Python, nachdem er Skripte von **Monty Python's Flying Circus** gelesen hatte.

Alexa, Amazons virtuelle persönliche Assistentin, verwendet Python zur Unterstützung der Spracherkennung.



Die Datenanalyse und maschinelles Lernen sind die beiden am häufigsten verwendeten Beispiele für Python.

Ende 2022 war Python die am meisten diskutierte Sprache im Internet.

PYTHON- UND LINUX-KENNTNISSE SIND DIE DRITT-BELIEBTESTEN IT-FÄHIGKEITEN IN GROSSBRITANNIEN.

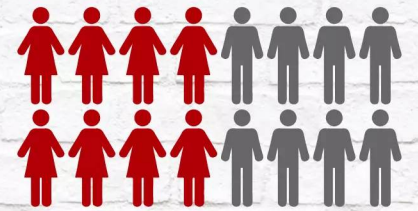
Disney Pixar verwendet Python in seiner RenderMan-Software, um zwischen anderen Grafikpaketen zu arbeiten.

ÜBER 75 % DER EMPFOHLENEN INHALTE VON NETFLIX WERDEN DURCH MIT PYTHON PROGRAMMIERTES MASCHINELLES LERNEN GENERIERT.

90 % ALLER FACEBOOK-BEITRÄGE WERDEN DURCH MIT PYTHON PROGRAMMIERTES MASCHINELLES LERNEN GEFILTERT.

SCHÄTZUNGEN ZUFOLGE BENUTZEN ÜBER 75 % DER WORKFLOW-AUTOMATISIERUNGS-SYSTEME DER NASA AN BORD DER ISS PYTHON.

16 000



In Großbritannien werden alle sechs Monate über 16 000 Python-Jobs ausgeschrieben.

AUF PYTHON BASIERENDE JOBS LIEGEN AN

16.

STELLE DER BEGEHRTESTEN BERUFE IN GROSSBRITANNIEN.

Python Data Science dürfte in den kommenden Jahren zum gefragtesten Job werden.

Google ist das Top-Unternehmen für die Einstellung von Python-Entwicklern, dicht gefolgt von Microsoft.

Data Science, Blockchain und maschinelles Lernen sind die am schnellsten zunehmenden Programmierfähigkeiten in Python.

New York und San Francisco führen die Spitze der Python-Entwicklerstädte an.

Python-Entwickler erhalten ein durchschnittliches Gehalt von

68.000 €

95 % ALLER ANFÄNGER IM PROGRAMMIEREN BEGINNEN MIT PYTHON ALS PRIMÄR- ODER SEKUNDÄRSPRACHE UND WENDEN SIE AUCH WEITERHIN AN.

75 % ALLER PYTHON-ENTWICKLER ARBEITEN MIT PYTHON 3, WÄHREND 25 % DIE VERALTETE PYTHON 2-VERSION VERWENDEN.

79 % ALLER PROGRAMMIERER VERWENDEN PYTHON TÄGLICH, 21% VERWENDEN PYTHON ALS SEKUNDÄRSPRACHE.

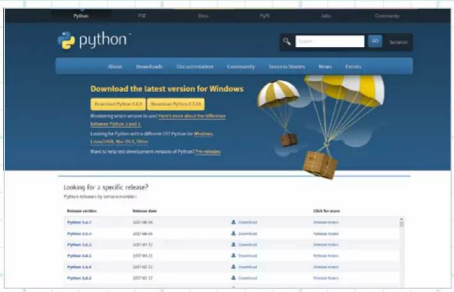
49 % DER WINDOWS 10-ENTWICKLER VERWENDEN PYTHON 3 ALS HAUPTPROGRAM-MIERSPRACHE.

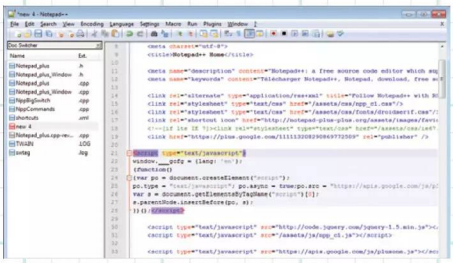
Benötigtes Zubehör

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 Codeeingabe, ist aber nicht unbedingt erforderlich. Sie können Code direkt über die IDLE eingeben und ausführen, allerdings bietet ein Texteditor wie Sublime Text oder Notepad++ erweiterte Funktionen und Farbcodierungen bei der Codeeingabe.

INTERNETZUGANG

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

ZEIT UND GEDULD

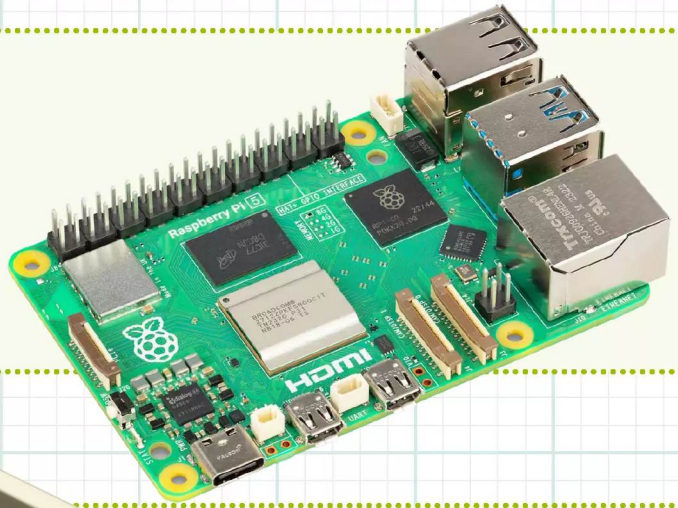
Ganz gleich, was Ihnen andere Bücher versprechen, Sie werden in 24 Stunden kein Programmierer werden. Dies nimmt Zeit und Geduld in Anspruch. Manches ist einfach verständlich, anderes dauert etwas länger. Seien Sie sich bewusst, dass Sie etwas völlig Neues lernen, und Sie werden Ihr Ziel erreichen.

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

DerRaspberry Pi 5 Model ist die neueste Version, die mit einer leistungsstärkeren CPU, einer Speicherauswahl von 4 GB oder 8 GB, WLAN und Bluetooth-Unterstüt-zung kommt. Der Preis beginnt bei ca. 70 € und geht bis zu ca. 90 € für die 8-GB-Version. Als Teil eines Bausatzes ist er ab ca. 55 € erhältlich, abhängig vom Bausatz, an dem Sie interessiert sind.

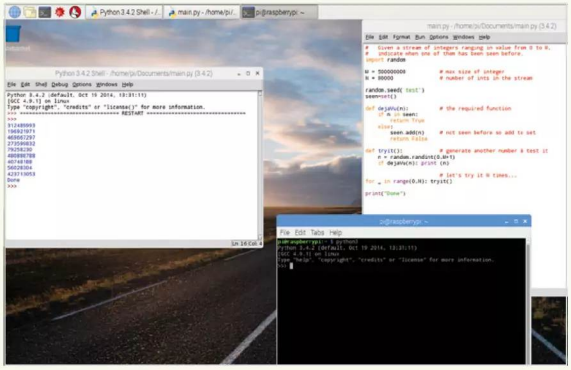


FUZE PROJECT

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.



MAGAZINE

Wir haben mehrere tolle Raspberry Pi-Titel, die über **www.pclpublications.com** erhältlich sind. In unseren Pi-Titeln erfahren Sie mehr zum Kauf, zur Einrichtung und Verwendung Ihres ersten Raspberry Pi. Es gibt einige tolle Projektbeispiele und Schritt-für-Schritt-Anleitungen, mit denen Sie das Beste aus dem Raspberry Pi herauszuholen.

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

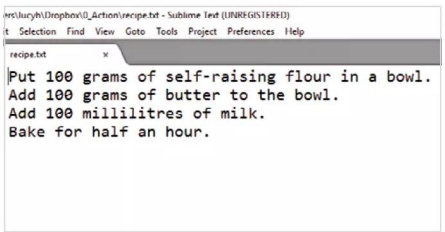
Eine Programmiersprache besteht aus einer Liste mit Anweisungen, die ein Computer befolgt. Dabei kann es sich um einfache Anweisungen handeln, wie der Anzeige Ihres Namens oder dem Abspielen einer Musikdatei, oder komplexe, wie dem Erstellen einer virtuellen Welt. Python wurde Ende der 80er von Guido 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 wie Rezepte für Computer. Ein Kuchenrezept könnte wie folgt aussehen:

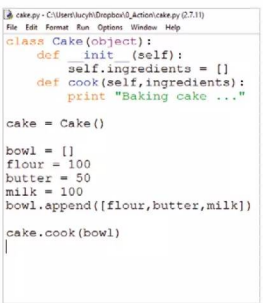
Put 100 grams of self-raising flour in a bowl.
Add 100 grams of butter to the bowl.
Add 100 millilitres of milk.
Bake for half an hour.



CODE

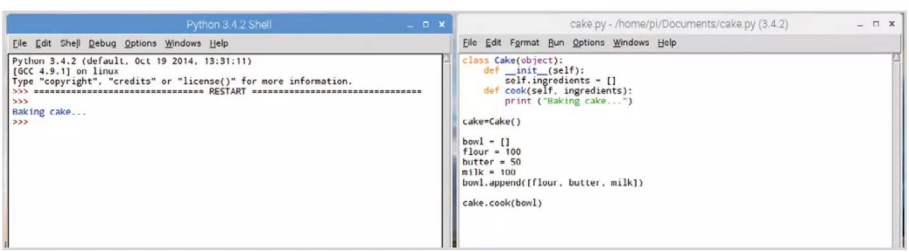
Genau wie unser Kuchenrezept besteht auch ein Programm aus Anweisungen, die der Reihe nach befolgt werden. Unser Kuchenrezept könnte als Programm wie folgt aussehen:

```
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.

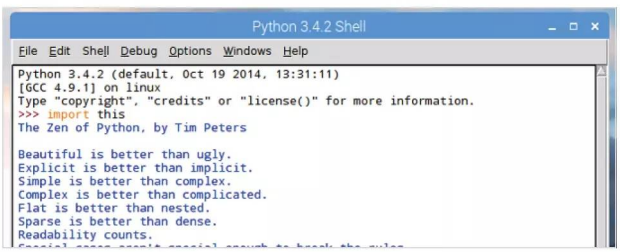


HÖHERE PROGRAMMIERSPRACHEN

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

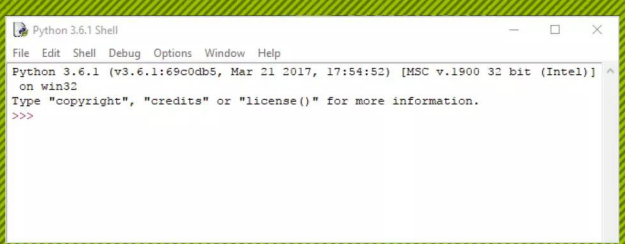
Genau wie in einem typischen Computerszenario macht die Existenz von zwei aktiven Versionen der Sprache – Python 2 und Python 3 – alles etwas komplizierter.

DIE WELT VON PYTHON

Python 3 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.

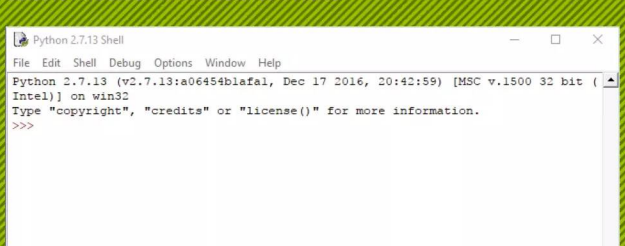
PYTHON 3.X

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) dieser neuen Funktionen nicht mit Python 2 Skripten, Modulen und Tutorials kompatibel. Obwohl zu Beginn nicht sehr populär, hat sich Python 3 mittlerweile zu einer der führenden Python-Programmiersprachen entwickelt.



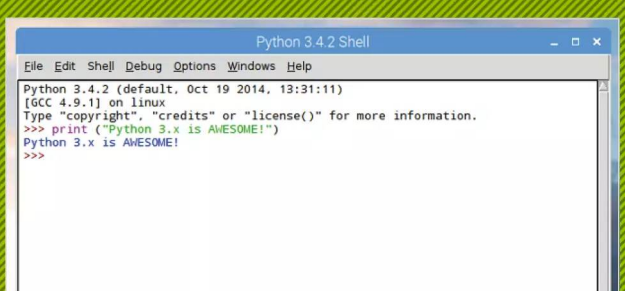
PYTHON 2.X

Warum gibt es nun aber zwei Versionen? Python 2 wurde im Jahr 2000 veröffentlicht und hat seitdem eine recht große Sammlung von Modulen, Skripten, 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 WINS

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 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

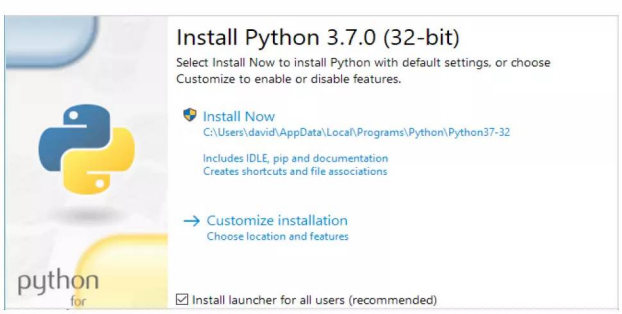
Python ist in Microsoft Windows nicht standardmäßig enthalten und muss daher manuell installiert werden. Glücklicherweise ist dies jedoch ein einfacher Vorgang.

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.

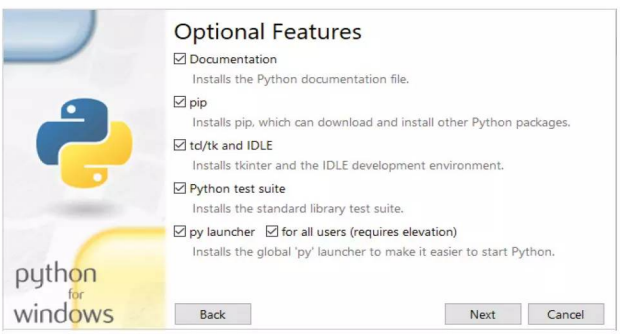
SCHRITT 2

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“.



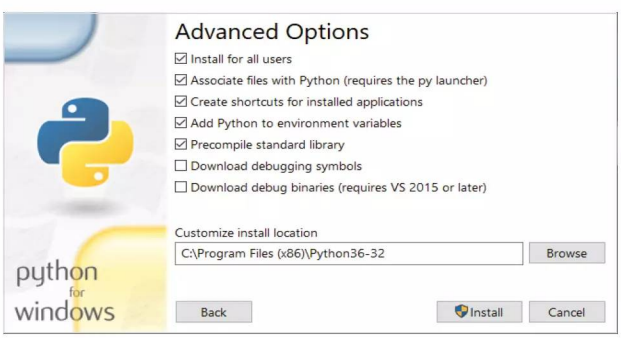
SCHRITT 3

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“.



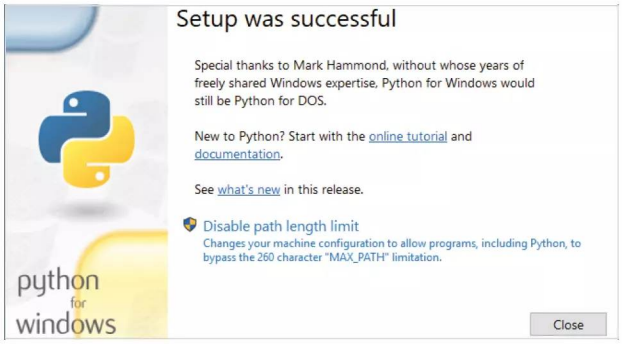
SCHRITT 4

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.



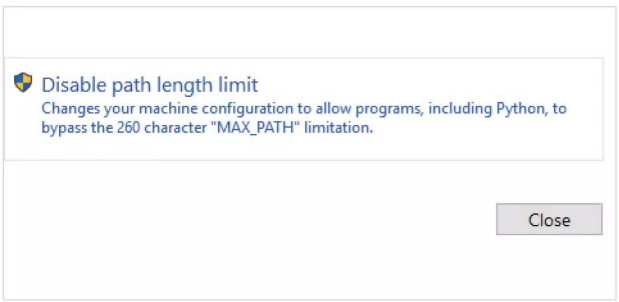
SCHRITT 5

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.



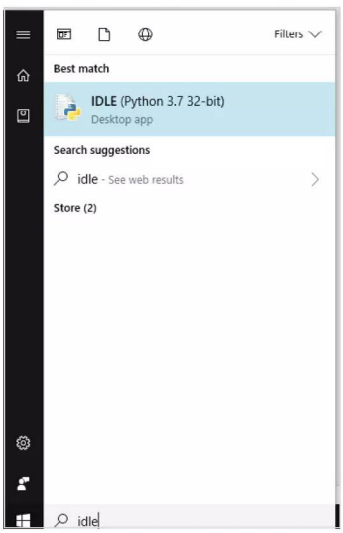
SCHRITT 6

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.

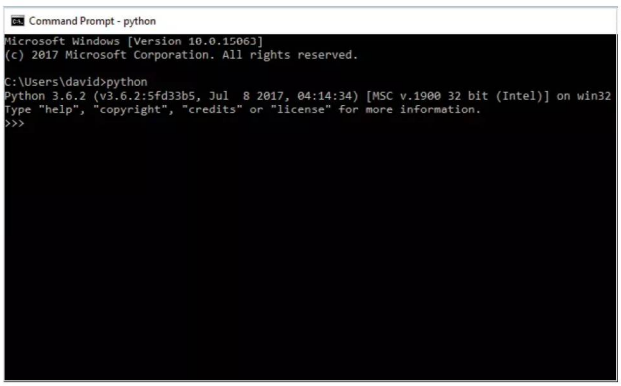


SCHRITT 8

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.

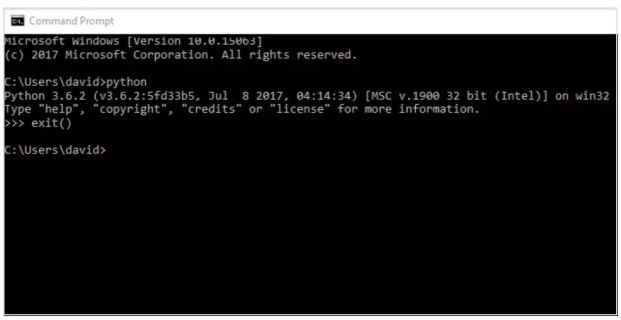
SCHRITT 9

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

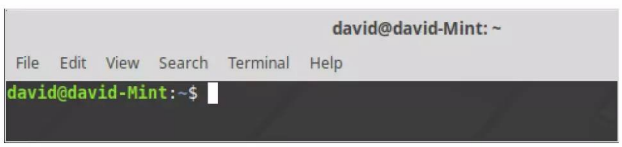
Python 3 ist auf dem Raspberry Pi Python 3 bereits vorinstalliert, fehlt jedoch auf anderen Linux-Distributionen. Sollten Sie keinen Raspberry Pi verwenden, finden Sie hier Informationen zum Überprüfen und Installieren von Python unter Linux.

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.

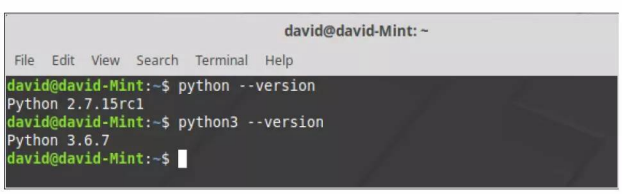
SCHRITT 1

Zuerst müssen Sie herausfinden, welche Version von Python derzeit auf Ihrem Linux-System installiert ist. Öffnen Sie durch Drücken von Strg + Alt + T das Terminal.



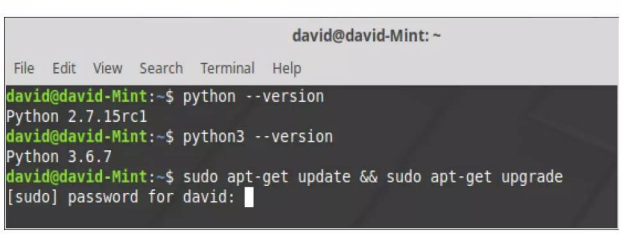
SCHRITT 2

Geben Sie nun `python --version` im Terminal ein. Die Ausgabe sollte sich auf die Version 2.x von Python beziehen. Die meisten Linux-Distributionen kommen standardmäßig mit Python 2 und 3, da für Python 2 noch reichlich Code vorhanden ist. Geben Sie nun Folgendes ein: `python3 --version`.



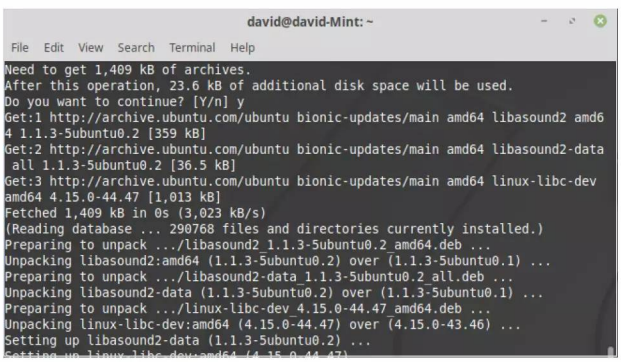
SCHRITT 3

In unserem Fall haben wir Python 2 und 3 installiert. Solange Python 3.x.x installiert ist, wird der Code in unseren Tutorials funktionieren. Es lohnt sich immer zu überprüfen, ob die Distribution mit den neuesten Versionen aktualisiert wurde. Geben Sie `sudo apt-get update && sudo apt-get upgrade` ein, um das System zu aktualisieren.



SCHRITT 4

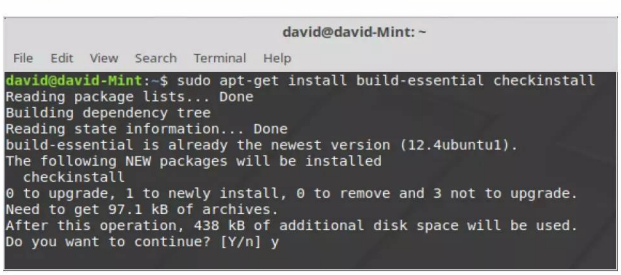
Geben Sie nach Abschluss der Aktualisierungen erneut `python3 --version` ein, um zu sehen, ob Python 3.x aktualisiert oder sogar installiert ist. Solange Sie Python 3.x verwenden, verwenden Sie die neueste Hauptversion. Die Zahlen nach 3. geben Patches und weitere Updates an. Sie sind oftmals unnötig, können aber wichtige neue Elemente enthalten.



SCHRITT 5

Wenn Sie die neueste Version wollen, müssen Sie Python aus dem Quellcode erstellen. Beginnen Sie, indem Sie die folgenden Befehle ins Terminal eingeben:

```
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 die Downloads-Schaltfläche, gefolgt von der Schaltfläche unterhalb des Python-Source-Fensters. Wählen Sie im erscheinenden Dialogfenster den Speicherort und beginnen Sie den Download.

SCHRITT 7

Gehen Sie im Terminal durch Eingabe von `cd Downloads/` zum Downloads-Ordner. Entpacken Sie nun mithilfe von `tar -xvf Python-3.Y.Y.tar.xz` den Inhalt des heruntergeladenen Python-Quellcodes (ersetzen Sie Y.Y. durch die Nummern der heruntergeladenen Version). Öffnen Sie nun mit `cd Python-3.Y.Y/` den neu entpackten Ordner.

```
david@david-Mint: ~/Downloads/Python-3.7.2
File  Edit  View  Search  Terminal  Help
Python-3.7.2/Objects/clinic/complexobject.c.h
Python-3.7.2/Objects/clinic/bytearrayobject.c.h
Python-3.7.2/Objects/clinic/enumobject.c.h
Python-3.7.2/Objects/clinic/bytesobject.c.h
Python-3.7.2/Objects/clinic/floatobject.c.h
Python-3.7.2/Objects/clinic/funcobject.c.h
Python-3.7.2/Objects/clinic/longobject.c.h
Python-3.7.2/Objects/clinic/dictobject.c.h
Python-3.7.2/Objects/clinic/structseq.c.h
Python-3.7.2/Objects/clinic/tupleobject.c.h
Python-3.7.2/Objects/clinic/moduleobject.c.h
Python-3.7.2/Objects/clinic/odictobject.c.h
Python-3.7.2/Objects/bytearrayobject.c
Python-3.7.2/Objects/typeobject.c
Python-3.7.2/Objects/lnotab_notes.txt
Python-3.7.2/Objects/methodobject.c
Python-3.7.2/Objects/tupleobject.c
Python-3.7.2/Objects/obmalloc.c
Python-3.7.2/Objects/object.c
Python-3.7.2/Objects/abstract.c
Python-3.7.2/Objects/listobject.c
Python-3.7.2/Objects/bytes_methods.c
Python-3.7.2/Objects/dictnotes.txt
Python-3.7.2/Objects/typeslots.inc
david@david-Mint:~/Downloads$ cd Python-3.7.2/
david@david-Mint:~/Downloads/Python-3.7.2$
```

SCHRITT 8

Geben Sie innerhalb des Python-Ordners Folgendes ein:

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

Dies kann je nach Geschwindigkeit Ihres Computers etwas dauern. Geben Sie nach Beendigung `python3.7 --version` ein, um die installierte Version zu überprüfen. Sie haben nun Python 3.7 zusammen mit älteren Versionen von Python 3.x.x und Python 2 installiert.

```
config.status: creating Makefile.pre
config.status: creating Misc/python.pc
config.status: creating Misc/python-config.sh
config.status: creating Modules/ld_so_aix
config.status: creating pyconfig.h
creating Modules/Setup
creating Modules/Setup.local
creating Makefile


If you want a release build with all stable optimizations active (PGO, etc),
please run ./configure --enable-optimizations


david@david-Mint:~/Downloads/Python-3.7.2$ sudo make altinstall
```

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.

```
david@david-Mint: ~/Downloads/Python-3.7.2
File  Edit  View  Search  Terminal  Help
david@david-Mint:~/Downloads/Python-3.7.2$ sudo apt-get install idle3
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  blt idle idle-python3.6 python3-tk tk8.6-blt2.5
Suggested packages:
  blt-demo tix python3-tk-dbg
The following NEW packages will be installed
  blt idle idle-python3.6 idle3 python3-tk tk8.6-blt2.5
0 to upgrade, 6 to newly install, 0 to remove and 3 not to upgrade.
Need to get 938 kB of archives.
After this operation, 4,221 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

SCHRITT 10

Sie benötigen zusätzlich auch 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 anschließend nach den neuesten PIP-Updates:

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

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

```
david@david-Mint: ~/Downloads/Python-3.7.2
File  Edit  View  Search  Terminal  Help
david@david-Mint:~/Downloads/Python-3.7.2$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python-pip-whl python3-distutils python3-lib2to3
Recommended packages:
  python3-dev python3-setuptools python3-wheel
The following NEW packages will be installed
  python-pip-whl python3-distutils python3-lib2to3 python3-pip
0 to upgrade, 4 to newly install, 0 to remove and 3 not to upgrade.
Need to get 1,984 kB of archives.
After this operation, 4,569 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

PYTHON UNTER macOS

Die Installation von Python auf macOS erfolgt auf beinahe gleiche Weise wie die Windows-Installation. Gehen Sie zur Python-Website, halten Sie den Mauszeiger über den Downloads-Link und wählen Sie Mac OS X aus den Optionen aus. Anschließend werden Sie zu den Python-Versionen für den Mac sowie zu den erforderlichen Installationsprogrammen für MacOS 64-Bit für OS X 10.9 und höher geführt.

Python auf dem Pi

Wenn Sie noch überlegen, welche Plattform Sie zur Installation und Nutzung von Python nehmen sollten, dann sollten Sie den Raspberry Pi in Betracht ziehen. Der Pi hat viele Vorteile: Er ist billig, benutzerfreundlich und außerordentlich flexibel.

DIE POWER DES PI

Eine weitaus leistungsstärkere Programmierplattform zum Schreiben und Testen von Code ist zwar ideal, aber nicht immer realisierbar. Die meisten von uns sind nicht in der Lage, allein für den Einstieg mehrere hundert Euros zu investieren. Hier kommt uns der Raspberry Pi zu Hilfe.

Der Raspberry Pi ist ein fantastisches Stück moderner Hardware, das unsere frühere Faszination, wie Computer funktionieren, wie programmiert wird und wie Elektronik auf Gründungsebene funktioniert, wieder hervorholt. Dank seiner einzigartigen Mischung aus Hardware und kundenspezifischer Software hat er sich als hervorragende Plattform zum Programmieren lernen erwiesen, insbesondere mit Python.

Während Sie mit dem Pi mühelos lernen können, mit anderen Programmiersprachen zu programmieren, ist es Python, das in den Vordergrund gerückt ist. Der Raspberry Pi verwendet Raspbian als empfohlenes Standardbetriebssystem. Raspbian ist ein Linux-Betriebssystem oder genauer gesagt, eine Debian-basierte Distribution von Linux. Dies bedeutet, dass im Gegensatz zu einer Neuinstallation von Windows 10, die keine Python-spezifische Basis hat, ein integriertes Element der Python-Programmierung bereits vorhanden ist. Die Raspberry Pi Foundation hat sich die Mühe gemacht, eine breite Palette von Python-Modulen, -Erweiterungen und sogar Beispielen zu integrieren. Alles, was zu tun ist, ist der Kauf eines Raspberry Pi. Befolgen Sie die Anweisungen zum Einrichten und sobald der Desktop geladen ist, können Sie mit dem Programmieren mit Python beginnen.

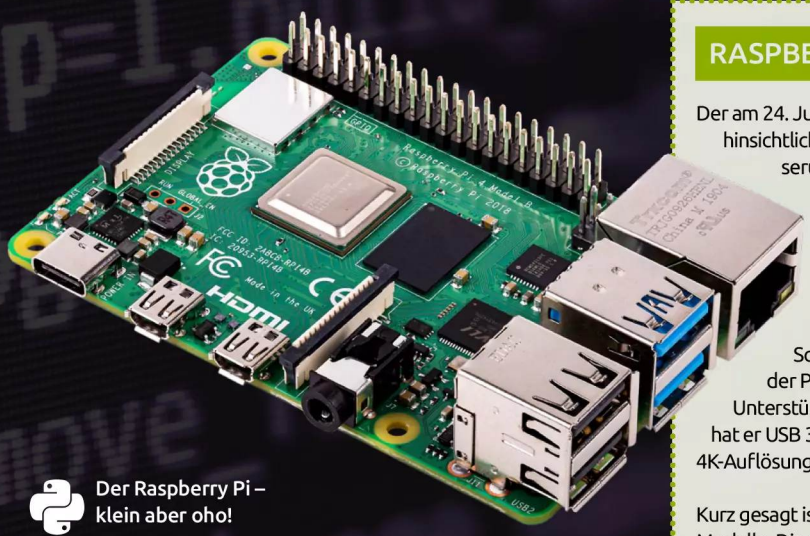
Der Raspberry Pi hat aber noch viel mehr zu bieten, was ihn zu einer hervorragenden Wahl für jene macht, die mit Python ihre ersten Schritte im Programmieren machen. Der Pi ist bemerkenswert einfach als kopfloser Computer einzurichten, d. h. dass Sie mit ein paar Änderungen hier und da von jedem anderen Computer oder Gerät in Ihrem Heimnetzwerk aus eine Fernverbindung zum Raspberry Pi herstellen können. Wenn Sie z. B. die Fernverbindungsoptionen eingerichtet haben, können Sie den Pi einfach an einer beliebigen Stelle in Ihrem Haus in Reichweite Ihres WLAN-Routers an die Stromversorgung anschließen. Solange der Pi verbunden ist, können Sie über Windows oder macOS genauso einfach auf den Desktop zugreifen, als würden Sie mit Tastatur und Maus vor dem Pi sitzen.

Diese Methode spart viel Geld, da Sie keine weitere Tastatur, Maus und keinen weiteren Monitor benötigen und auch Platz für all diese Extras einsparen. Wenn es an Platz und Geld mangelt, können Sie beim Kauf eines der zahlreichen erhältlichen Kits für etwa 70 € eine vorinstallierte SD-Karte (mit dem neuesten Raspbian-Betriebssystem), ein Gehäuse, eine Netzbuchse und Kabel bekommen. Dies lohnt sich, da Sie den Pi mit sehr geringem Aufwand unter einem Schreibtisch an die Stromversorgung anschließen können und weiterhin eine Verbindung herstellen und programmieren können. Der größte Vorteil ist natürlich der zusätzliche Content, den die Raspberry Pi Foundation zur Verfügung stellt, da deren Ziel es ist, den Benutzern beim Lernen zu helfen, ganz gleich, ob es sich um Programmierung, Elektronik oder einen anderen Aspekt der Informatik handelt. Um dieses Ziel zu erreichen, bietet die Pi Foundation verschiedene IDEs, mit denen der Benutzer Python-Code kompilieren kann. Neben Python 2 und Python 3 gibt es sogar eine Python-Bibliothek, die das Kommunizieren mit Minecraft ermöglicht.

Es gibt noch weitere Vorteile, z. B. die Möglichkeit, Python-Code mit Scratch (eine von MIT entwickelte objektorientierte Programmiersprache, mit der Kinder die Funktionsweise der Programmierung verstehen können) zu kombinieren und die GPIO-Verbindung auf dem Pi zu programmieren, um alle angeschlossenen Robotik- oder Elektronikprojekte zu steuern. Raspbian enthält auch einen Sense-HAT-Emulator (ein HAT ist eine an die Hardware angeschlossene Schaltung, die dem Pi verschiedene Elektronik-, Robotik- und Motorisierungsprojekte bietet), auf den über Python-Code zugegriffen werden kann.

Folglich dient der Raspberry Pi als ausgezeichnete Programmierbasis und Projektgrundlage. Aus diesen und vielen anderen Gründen haben wir den Pi in diesem Titel als unsere Hauptbasis für Python-Code verwendet. Auf einem Pi geschriebener und ausgeführter Code kann auch unter Windows, anderen Linux- und macOS-Versionen verwendet werden. Wenn der Code ein bestimmtes Betriebssystem erfordert, werden wir im Text darauf hinweisen.

Alles, was Sie brauchen, um mit Python zu programmieren, ist im Betriebssystem enthalten!



Der Raspberry Pi – klein aber oho!

RASPBERRY PI 4

Der am 24. Juni 2019 eingeführte Raspberry Pi 4 Model B ist hinsichtlich der Leistung und Hardware eine deutliche Verbesserung. Er war auch neben dem ursprünglichen Pi eines der am schnellsten ausverkauften Modelle.

Mit einem neuen 64-Bit-Quad-Core-ARM-Cortex-A72-Prozessor mit 1,5 GHz und einer Auswahl an 1 GB-, 2 GB- oder 4 GB-Speicherversionen ist der Pi 4 einem echten Desktop-Computer einen Schritt nähergekommen. Darüber hinaus bekam der Pi 4 überraschenderweise eine Dual-Monitor-Unterstützung in Form von Micro-HDMI-Anschlüssen. Ferner hat er USB 3.0-Anschlüsse, Bluetooth 5.0 und eine GPU, die 4K-Auflösungen und OpenGL ES 3.0-Grafiken verarbeiten kann.

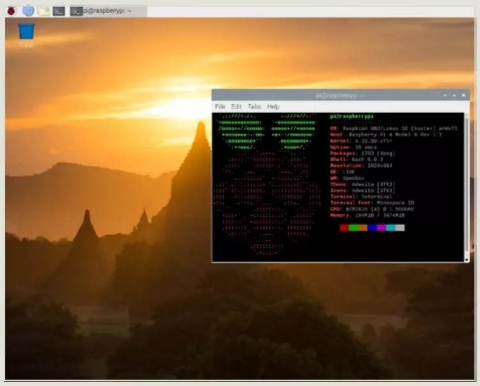
Kurz gesagt ist der Pi 4 der leistungsstärkste der aktuellen Modelle. Die verschiedenen Speicherversionen kosten jedoch entsprechend. Die 1-GB-Version kostet ca. 40 €, die 2-GB-Version ca. 50 € und die 4-GB-Version ca. 60 €. Denken Sie daran, beim Kauf auch ein oder zwei Micro-HDMI-Kabel mit einzukalkulieren.

RASPBIAN BUSTER

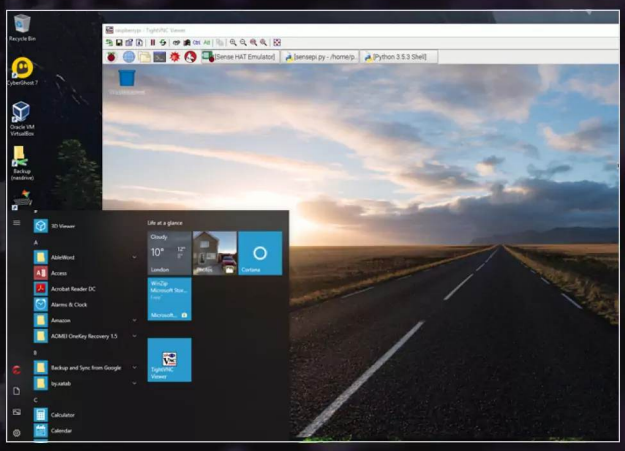
Zusätzlich zur Veröffentlichung des Pi 4 kompilierte das Raspberry Pi-Team auch eine neue Version des Raspbian-Betriebssystems mit dem Codenamen Buster.

In Verbindung mit der neuen Hardware des Pi 4 bietet Buster einige Updates an, obwohl es im Großen und Ganzen dem Aussehen und der Bedienung der Vorgängerversion von Raspbian sehr ähnlich ist. Die Aktualisierungen richten sich hauptsächlich an die 4K-Anzeige und Wiedergabe und bieten dem Pi 4 eine Reihe neuer Grafiktreiber und Leistungsverbesserungen.

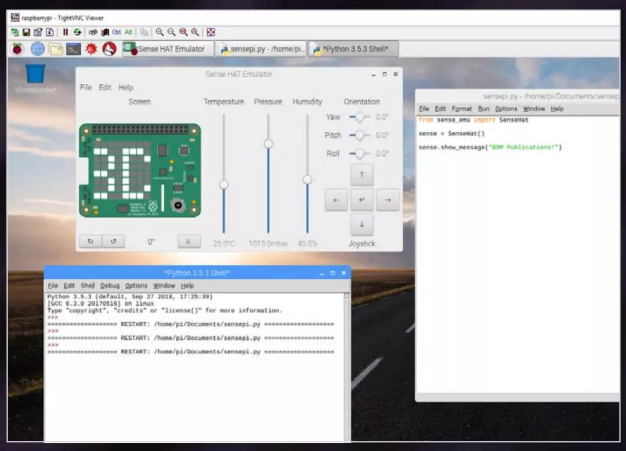
Was Sie in diesem Buch finden, wird mit dem Raspberry Pi 4 und Raspbian Buster funktionieren!



Nach der Einrichtung können Sie von jedem Gerät/PC aus eine Fernverbindung zum Pi-Desktop herstellen.



Sie können über Windows mit Python sogar angeschlossene Hardware aus der Ferne testen.



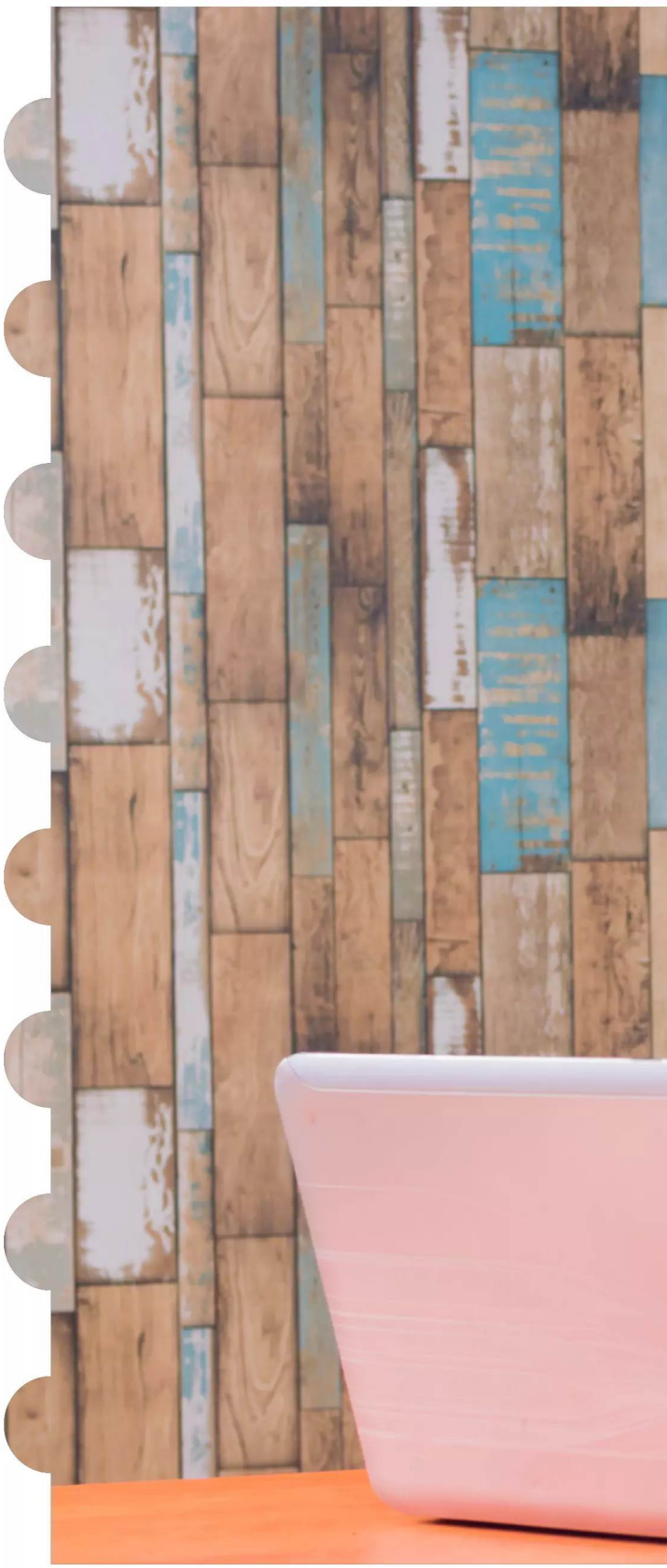
Nachdem Python nun auf Ihrem Computer installiert und betriebsbereit ist, werden Sie nun einige der wichtigsten Grundlagen der Sprache kennenlernen.

In diesem Abschnitt erhalten Sie eine Einführung in die Grundlagen von Python, von den ersten Codezeilen bis hin zur Arbeit mit Variablen und der Benutzerinteraktion. Wenn Sie diese beherrschen, werden Sie in der Lage sein, mit Code anderer Python-Entwickler zu interagieren und den Aufbau und die Funktionsweise des Codes zu verstehen.

Wie ein Python-Entwickler einmal sagte: „Ebnen Sie mit Python den Weg in Ihre Zukunft."

„Komplexität zu beherrschen macht die Computerprogrammierung aus."

– Brian Kernighan (Mitentwickler von UNIX und Autor)



Programmieren mit Python

Python zum ersten Mal starten

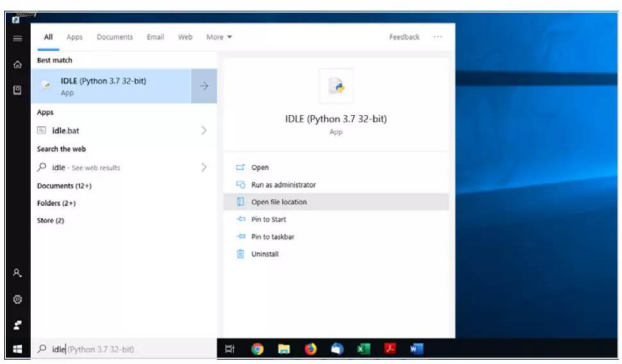
Für die folgenden Beispielen verwenden wir Python 3 unter Windows 10. Machen Sie sich keine Sorgen, wenn Ihre Python-Version 3.4.2 oder niedriger als die aktuelle Version ist. Solange Sie Python 3 verwenden, wird der Code funktionieren.

PYTHON STARTEN

Wenn man etwas Neues lernt, sollte man langsam beginnen und deshalb wollen wir fürs Erste einfach nur etwas auf dem Bildschirm anzeigen lassen. Mit zunehmender Erfahrung kann dann ein Gang höher geschaltet werden.

SCHRITT 1

Klicken Sie auf die Windows-Startschaltfläche und geben Sie „Idle" ein. Als Ergebnis erscheint die aktuell installierte Version von Python, z. B. **IDLE (Python 3.7 32-Bit)**. Sie können sie der Einfachheit halber auch an „Start" anheften. Klicken Sie auf das Symbol, um die Python-Shell zu starten.



SCHRITT 2

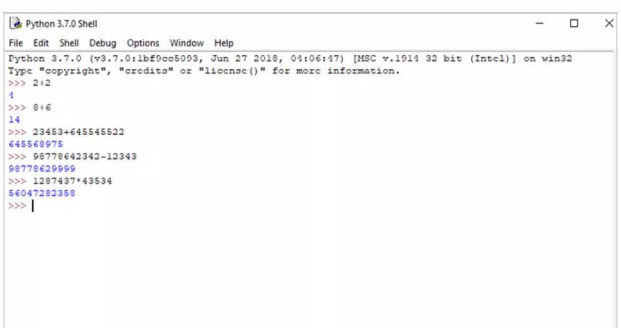
Die Shell lässt Sie Code eingeben und zeigt die Antworten und Ausgaben des in Python programmierten Codes an. Dies ist eine Art Sandbox, in der Sie einfachen Code und Prozesse ausprobieren können.

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.

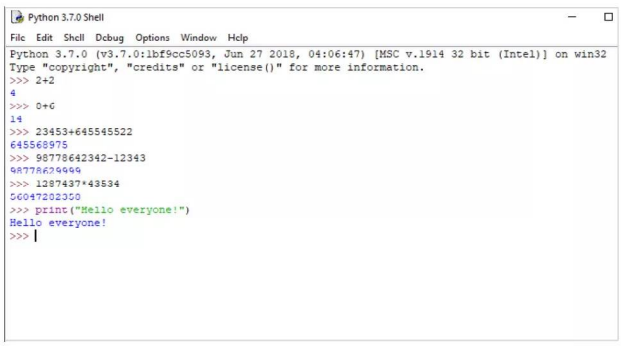


SCHRITT 5

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

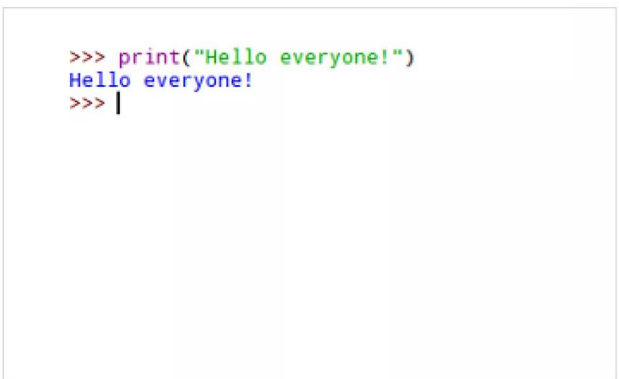
```
print("Hello everyone!")
```

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, in diesem Fall Hello everyone!, auf dem Bildschirm auszugeben.



SCHRITT 7

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

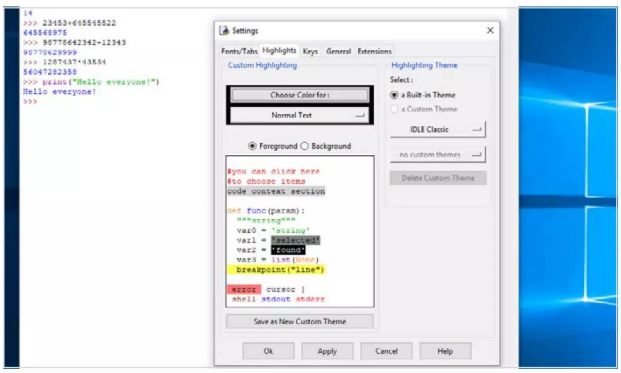
Schwarz – Daten und Variablen
Grün – Strings
Lila – Funktionen
Orange – Befehle
Blue – Benutzerfunktionen
Dunkelrot – Kommentare
Hellrot – Fehlermeldungen

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:`

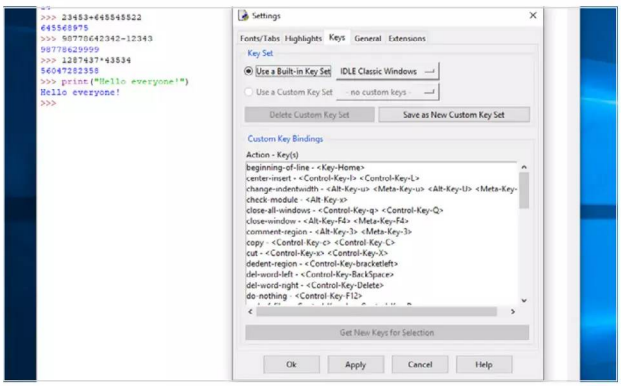
SCHRITT 8

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.



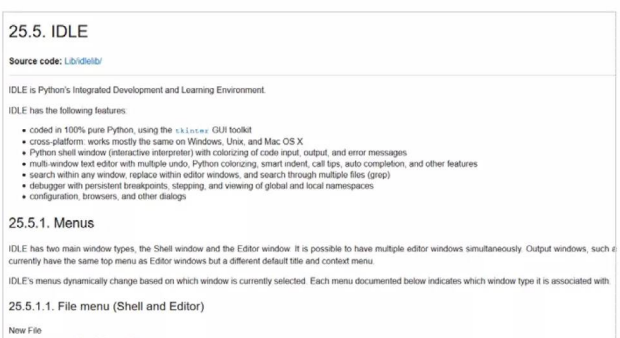
SCHRITT 9

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 **https://docs.python.org/3/library/idle.html** zu gehen, wo viele der IDLE-Funktionen beschrieben werden.


25.5. IDLE

Source code: Lib/idlelib/

IDLE is Python's Integrated Development and Learning Environment.

IDLE has the following features:

- coded in 100% pure Python, using the tkinter GUI toolkit
- cross-platform: works mostly the same on Windows, Unix, and Mac OS X
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

25.5.1. Menus

IDLE has two main window types, the Shell window and the Editor window. It is possible to have multiple editor windows simultaneously. Output windows, such as currently have the same top menu as Editor windows but a different default title and context menu.

IDLE's menus dynamically change based on which window is currently selected. Each menu documented below indicates which window type it is associated with.

25.5.1.1. File menu (Shell and Editor)

New File
Create a new file editing window.

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.

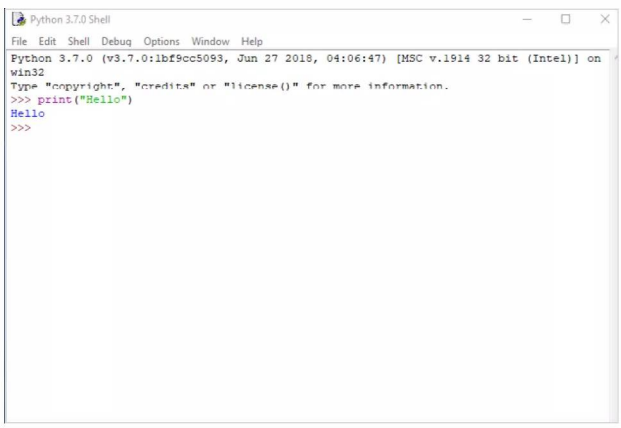
PYTHON AUSPROBIEREN

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 wie auf der vorherigen Seite beschrieben. Geben Sie in der Shell Folgendes ein:

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



SCHRITT 2

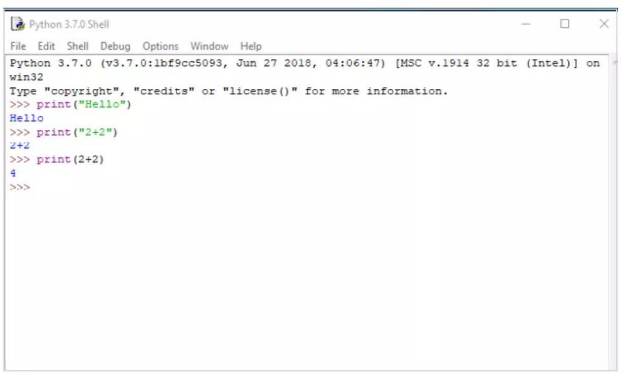
Wie vorhergesagt erscheint das Wort Hello in der Shell als blauer Text, der die Ausgabe eines Strings anzeigt. Dies ist ziemlich einfach und muss nicht großartig erklärt werden. Probieren Sie nun Folgendes aus:

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

SCHRITT 3

Wie Sie sehen wird anstelle der Zahl 4 2+2 auf dem Bildschirm ausgegeben. Die Anführungszeichen 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

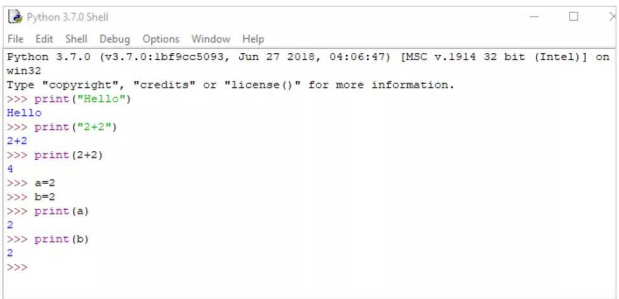
Sie können auf diese Weise fortfahren und 2+2, 464+2343 usw. in der Shell ausgeben. Ein einfacherer Weg ist die Verwendung einer Variablen, auf die wir später noch genauer eingehen werden. Geben Sie Folgendes ein:

```
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)
```

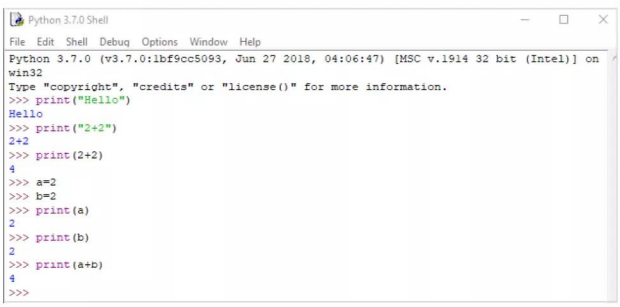


SCHRITT 6

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)
```

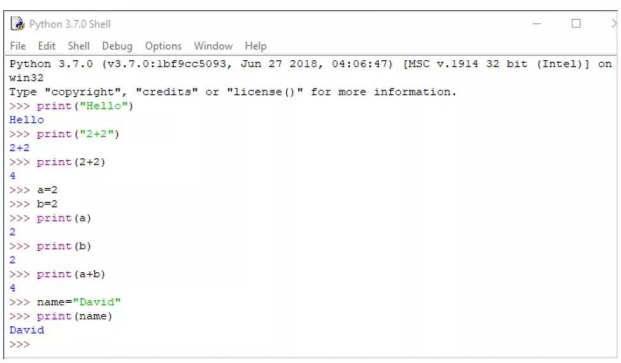
Dieser Code addiert die Werte von a und b und gibt das Ergebnis wieder.



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)
```

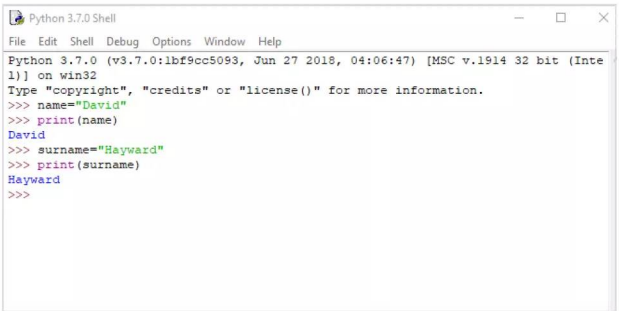


SCHRITT 8

Wir fügen nun den Nachnamen hinzu:

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

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

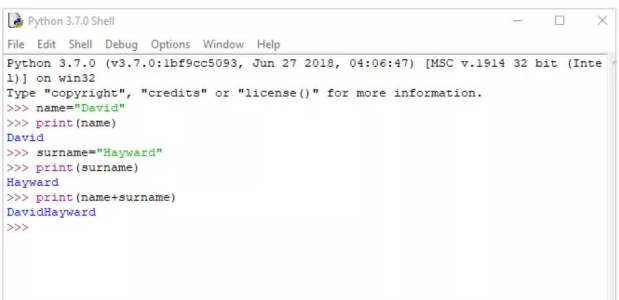


SCHRITT 9

Würden wir die gleiche Eingabe mit dem Pluszeichen anwenden, würde der Name in der Shell nicht korrekt ausgegeben 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 Codeabschnitte 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.

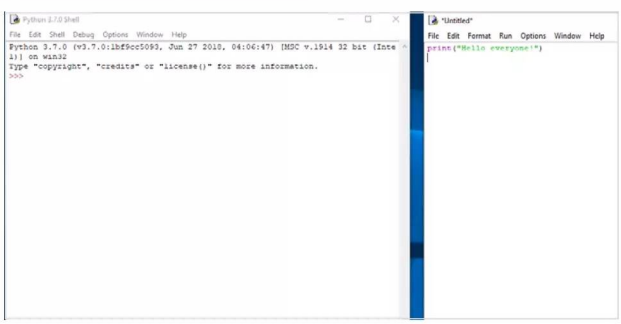
SCHRITT 1

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

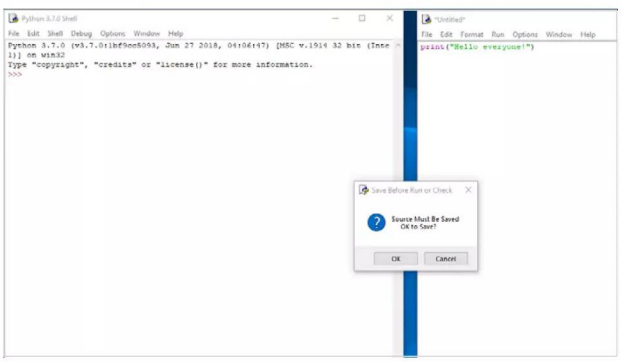
Der IDLE-Editor ist im Grunde ein einfacher Texteditor mit Python-Funktionen, Farbcodierung usw., ähnlich wie Sublime. Code wird auf die gleiche Weise wie in der Shell eingegeben. Geben Sie das folgende Beispiel aus dem vorherigen Tutorial ein:

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



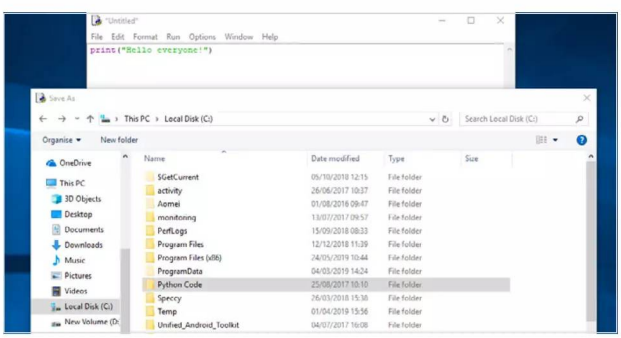
SCHRITT 3

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.



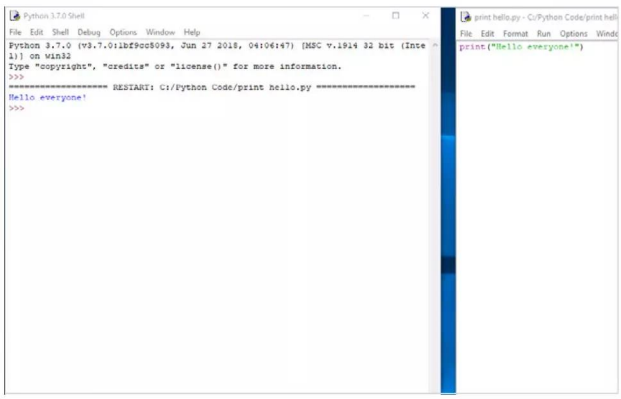
SCHRITT 4

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!".

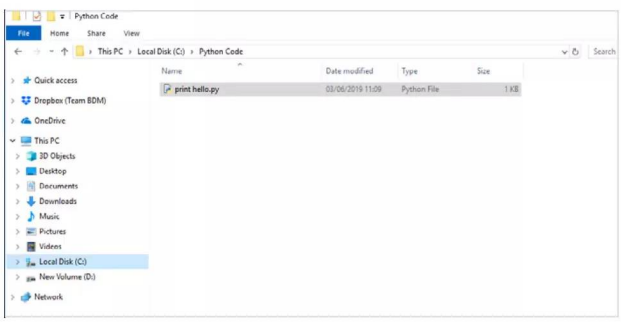


SCHRITT 6

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.

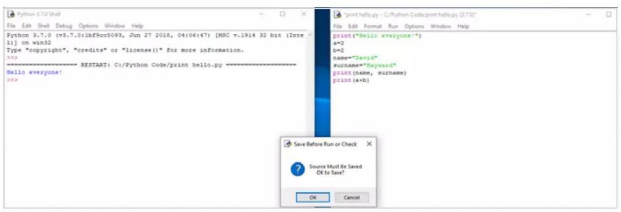


SCHRITT 8

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

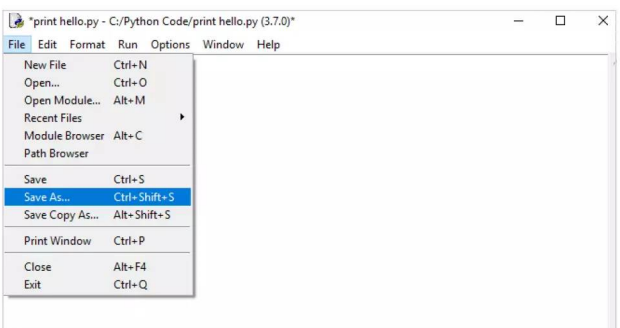
```
a=2
b=2
name="David"
surname="Hayward"
print(name, surname)
print (a+b)
```

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 Codeeinträgen 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.

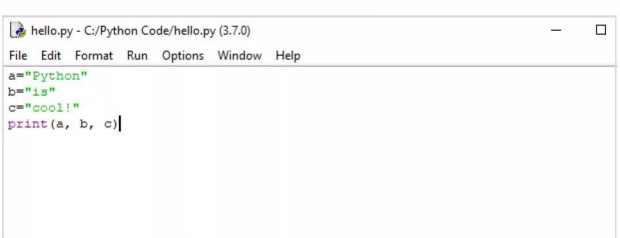


SCHRITT 10

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

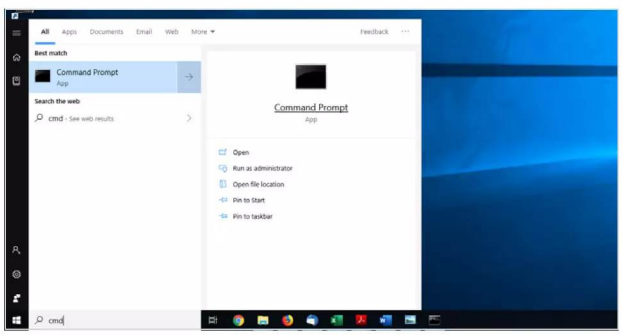
Obwohl wir in der GUI IDLE arbeiten werden, ist auch die Anwendung von Pythons Befehlszeile einen Blick wert. Je nachdem, was für einen Code sie schreiben, ist eine Ausführung über die Befehlszeile der IDLE zu bevorzugen.

DER CODE IN DER BEFEHLSZEILE

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

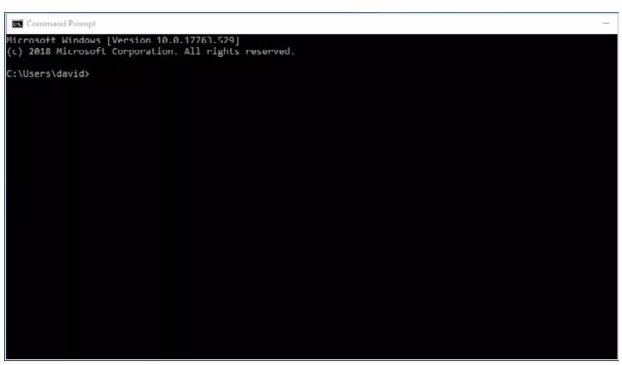
SCHRITT 1

Bei der Erstinstallation von Python wurden automatisch alle erforderlichen Komponenten installiert, damit Code auch außerhalb der GUI IDLE ausgeführt werden kann, d. h. in der Befehlszeile. Klicken Sie zunächst auf die Windows-Startschaltfläche und geben Sie **cmd** oder **Eingabeaufforderug** ein.



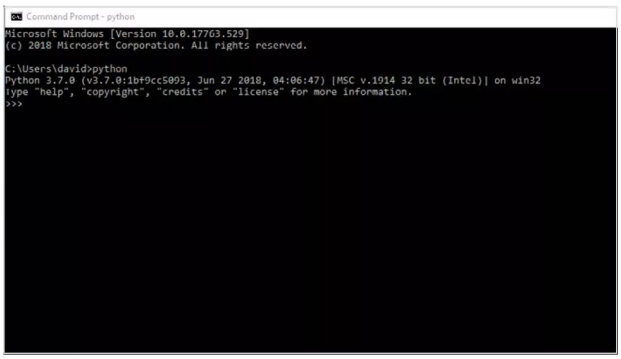
SCHRITT 2

Klicken Sie auf das Ergebnis der Suche – die Eingabeaufforderungs-App. Dadurch wird ein neues Fenster mit einem schwarzen Hintergrund und weißem Text geöffnet. Dies ist die Befehlszeile, die unter den Betriebssystemen macOS, Linux und Raspberry Pi auch als Terminal bezeichnet wird.



SCHRITT 3

Sie sind nun in der Befehlszeile und können Python über den Befehl **python** und Drücken der Eingabetaste starten. Dadurch gelangen Sie in die Befehlszeilenversion der Shell mit den drei nach rechts weisenden Pfeilen, die als Cursor dienen (>>>).

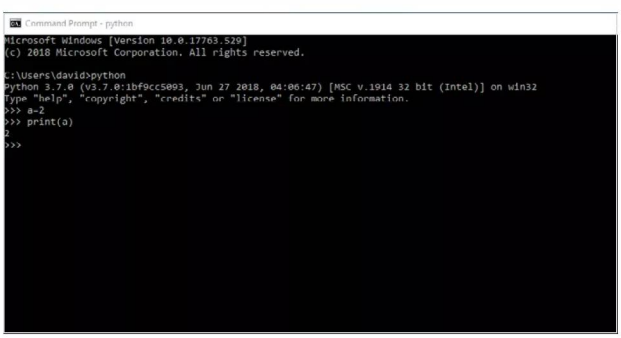


SCHRITT 4

Von hier aus können Sie Code eingeben, den wir uns zuvor bereits angeschaut haben, z. B.:

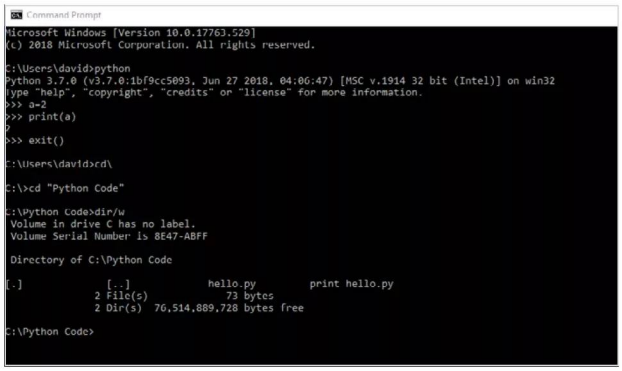
```
a=2
print(a)
```

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 Tutorial gespeichert haben, und listen Sie die verfügbaren Dateien auf. Die hello.py-Datei sollte hoffentlich dabei sein.



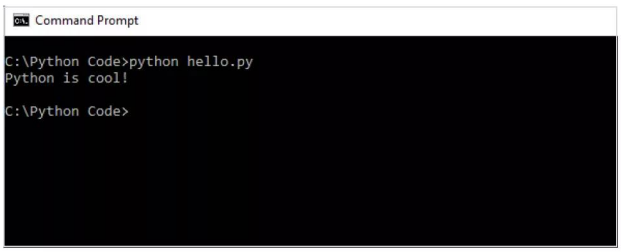
SCHRITT 6

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

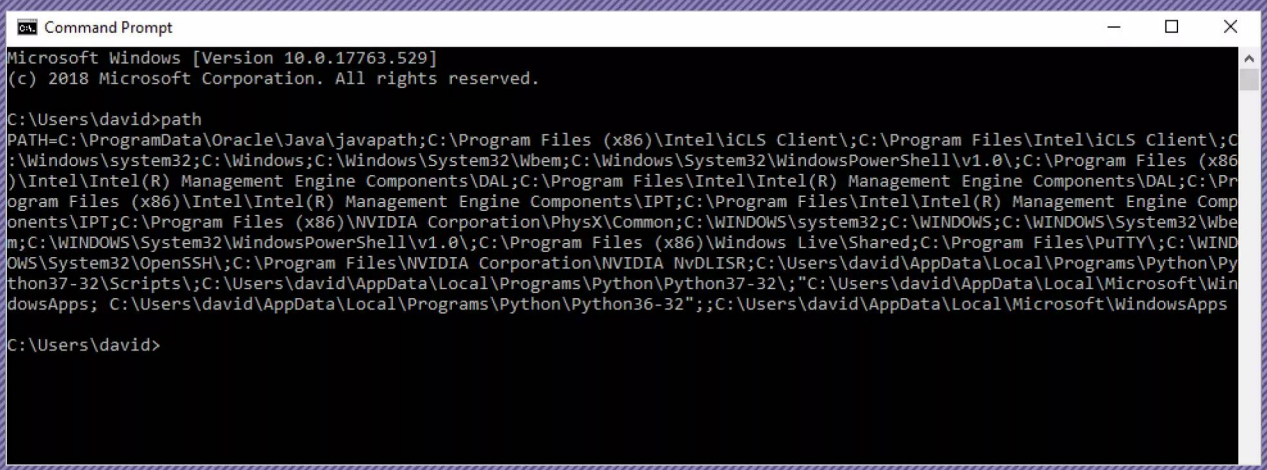
```
python3 hello.py
```

Dadurch wird der zuvor erstellte Code ausgeführt:

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



VERSCHIEDENE PYTHON-VERSIONEN



Wenn Sie Python 3 zuvor auf einem Mac, Raspberry Pi oder unter Linux verwendet haben, wundern Sie sich vielleicht, warum die Windows-Version von Python die Befehlszeile `python` anstelle von `python3` verwendet.

Der Grund dafür ist, dass auf UNIX-ähnlichen Systemen wie macOS und Linux bereits ältere Python-Bibliotheken vorinstalliert sind, da einige der macOS- und Linux-Systemdienstprogramme auf Python 2 angewiesen sind. Die Installation einer neueren Version von Python und die dadurch entstehende Änderung des ausführbaren Namens können schwerwiegende Folgen für das System haben.

Daher beschlossen die Entwickler, dass der beste Ansatz für macOS- und Linux-Systeme darin besteht, die Befehlszeile `python` ausschließliche für Python 2 zu belassen, und dass neuere, vom Benutzer installierte Python-Versionen `python3` sein würden.

Dies ist kein Problem für Windows, da es nur die vom Benutzer selbst installierten Python-Bibliotheken verwendet. Installiert ein Windows-Benutzer Python, wird die Befehlszeileninstanz automatisch der Windows-Kernvariablen PATH hinzugefügt, die Sie durch Eingabe von path in der Befehlszeile anzeigen können. Dies verweist auf die Datei python.exe, die zum Ausführen von Python-Code über die Befehlszeile erforderlich ist.

Wir raten davon ab, sowohl Python 2 als auch Python 3 unter Windows 10 zu installieren. Natürlich können Sie das machen, Python 3 ist jedoch die neueste Version, obwohl Python 2 immer noch einen festen Platz in der Programmierwelt hat. In diesem Fall müssen Sie einen der Python-Versionsnamen umbenennen, da diese in verschiedenen Ordnern installiert werden und beide python.exe als ausführbare Befehlszeile verwenden. Wenn Sie also nicht unbedingt beide Python-Versionen installieren müssen, sollten Sie sich an Python 3 halten.

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 nehmen. Wenn Sie sich für den Texteditor eines Drittanbieters entschieden haben, müssen Sie die IDLE-Shell für diesen Teil Tutorials verwenden.

SCHRITT 2

Geben Sie Folgendes in die Shell ein:

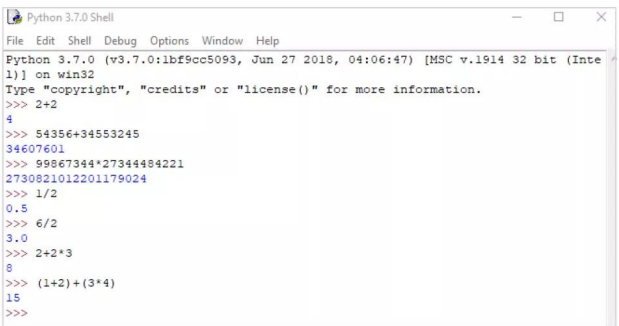
```
2+2
54356+34553245
99867344*27344484221
```

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)
```

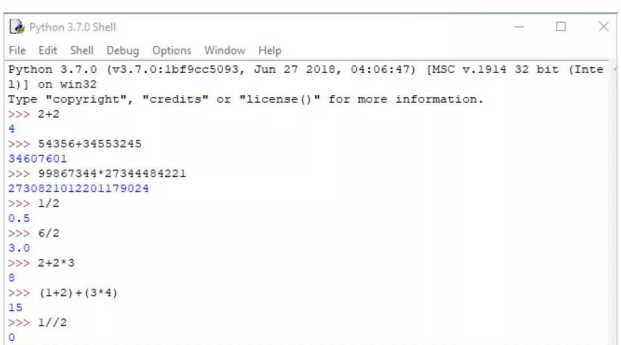


SCHRITT 4

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 Ausdrücke 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:

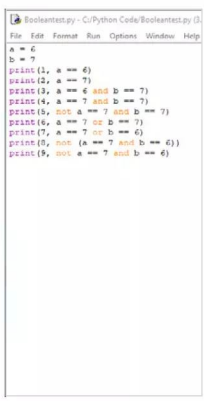
```
format(3, 'b')
```

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

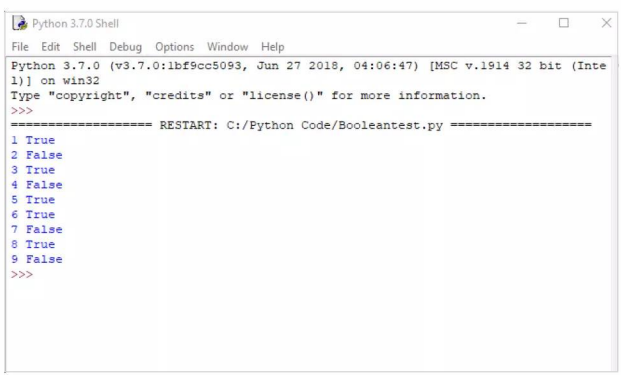
Ein boolescher Ausdruck ist eine logische Anweisung, die entweder wahr oder falsch ist. Wir können damit Daten vergleichen und testen, ob sie gleich, kleiner oder größer sind. Probieren Sie Folgendes in einer neuen Datei (New File) aus:

```
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!

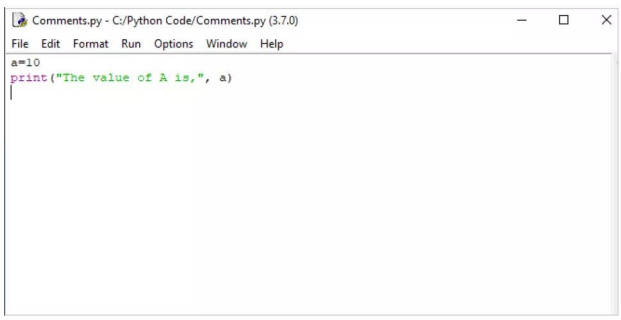
Um ihren Code lesbar zu halten, kommentieren Programmierer bestimmte Abschnitte. Wenn eine Variable verwendet wird, fügt der Programmierer in einem Kommentar z. B. hinzu, was diese bezwecken soll. Es ist ein bewährtes Verfahren.

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)
```

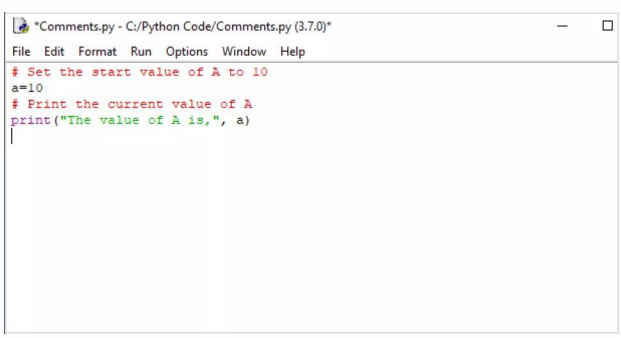
Speichern Sie die Datei und führen Sie den Code aus.



SCHRITT 2

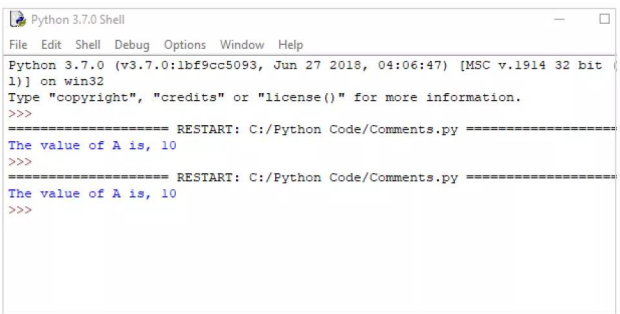
Wenn Sie den Code ausführen, wird wie erwartet die Zeile **The value of A is 10** im IDLE-Shell-Fenster ausgegeben. Fügen Sie nun einige der Kommentare hinzu, die Sie normalerweise im Code sehen würden:

```
# 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

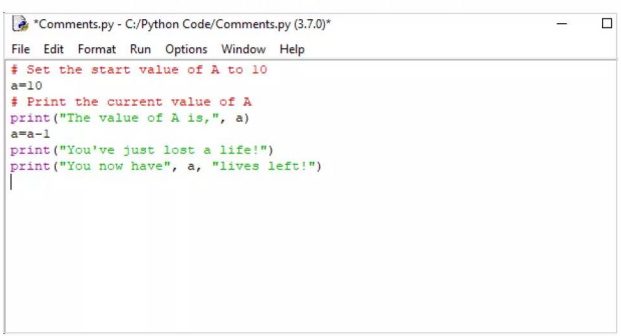
Speichern Sie den Code erneut und führen Sie ihn aus. Sie werden sehen, dass die Ausgabe in der IDLE-Shell trotz der hinzugefügten zusätzlichen Zeilen immer noch dieselbe ist. Einfach ausgedrückt steht das Rautenzeichen (#) für eine Textzeile, die der Programmierer einfügen kann, um andere darüber zu informieren, was passiert, ohne dass der Benutzer dies bemerkt.



SCHRITT 4

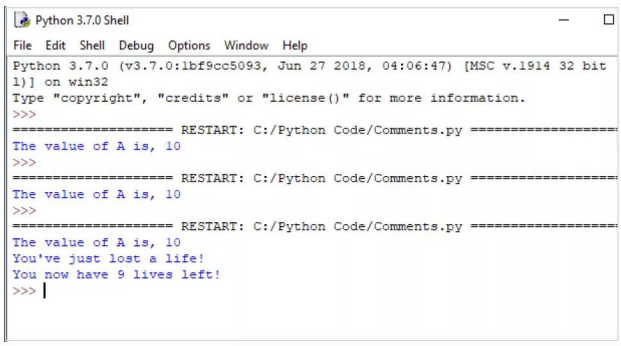
Nehmen wir an, dass die von uns erstellte Variable A die Anzahl der Leben in einem Spiel ist. Jedes Mal, wenn der Spieler stirbt, wird der Wert um 1 verringert. Der Programmierer könnte ein Programm wie das Folgende eingeben:

```
a=a-1
print(“You’ve just lost a life!”)
print(“You now have”, a, “lives left!”)
```



SCHRITT 5

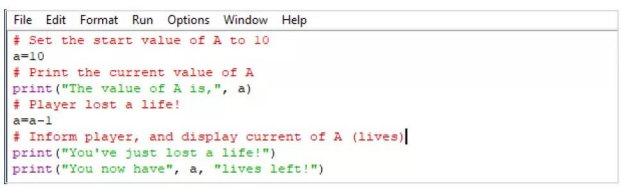
Während wir wissen, dass die Variable A für Leben steht und der Spieler gerade ein Leben verloren hat, weiß ein zufälliger Betrachter oder jemand, der den Code überprüft, das eventuell nicht. Um zu verdeutlichen, wie praktisch Kommentare sind, stellen Sie sich einfach vor, der Code wäre zwanzigtausend Zeilen lang anstatt nur sieben.



SCHRITT 6

Im Wesentlichen könnte der neue Code mit den Kommentaren wie folgt aussehen:

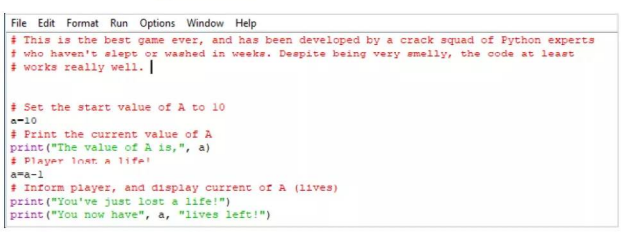
```
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
# Player lost a life!
a=a-1
# Inform player, and display current value of A (lives)
print("You've just lost a life!")
print("You now have", a, "lives left!")
```



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:

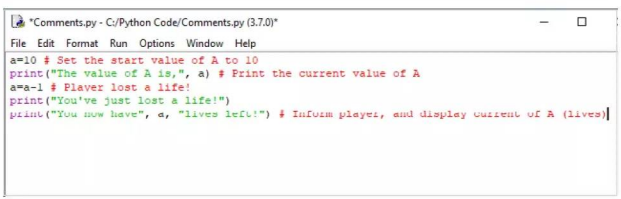
```
# This is the best game ever, and has been developed by a crack squad of Python experts
# who haven't slept or washed in weeks. Despite being very smelly, the code at least
# works really well.
```



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:

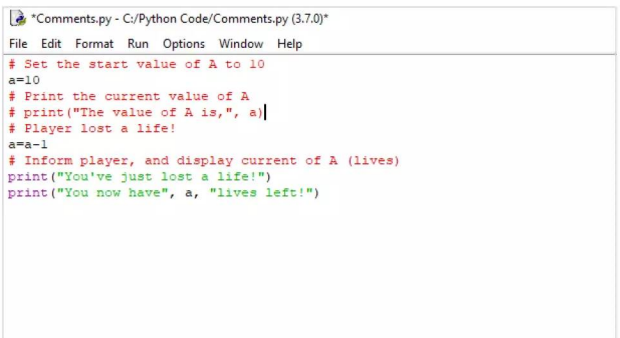
```
a=10 # Set the start value of A to 10
print("The value of A is,", a) # Print the current value of A
a=a-1 # Player lost a life!
print("You've just lost a life!")
print("You now have", a, "lives left!") # Inform player, and display current value of A (lives)
```



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:

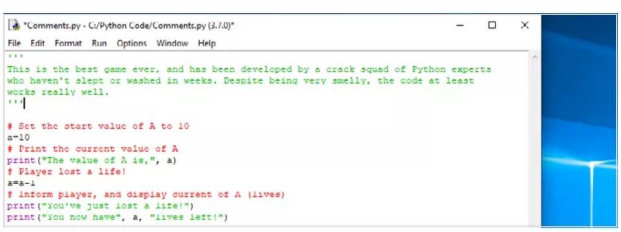
```
# print("The value of A is,", a)
```



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

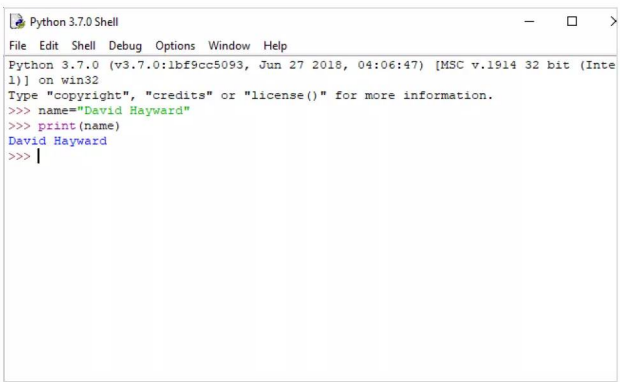
Wir haben bereits einige Beispiele für Variablen in unserem Python-Code gesehen, aber es lohnt sich immer, sich genauer anzuschauen, wie sie funktionieren und wie Python bestimmte Werte erstellt und diese einer Variablen zuweist.

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.

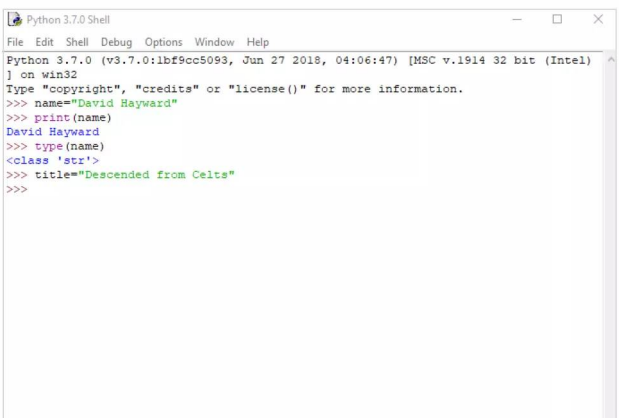
SCHRITT 1

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).



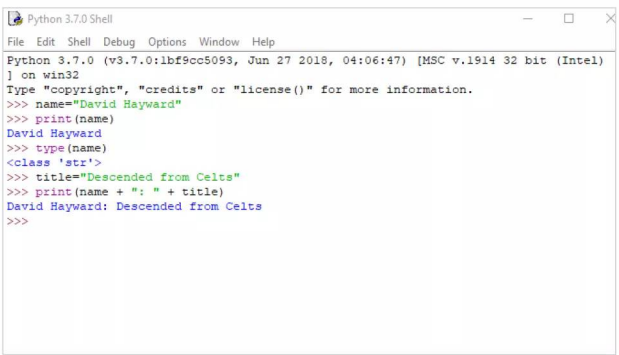
SCHRITT 2

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
```

Wir geben dann die Ausgabe der neuen Variablen wie folgt aus:

```
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)
```

SCHRITT 6

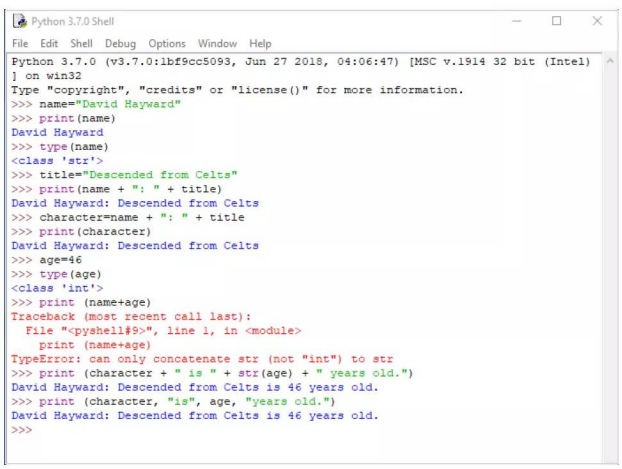
Dieser Vorgang nennt sich Typumwandlung. Der Python-Code lautet:

```
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.

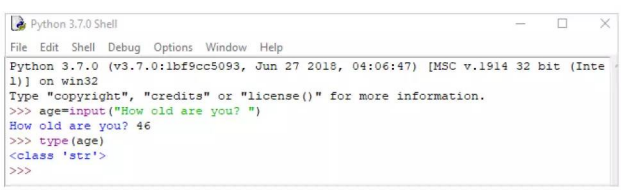


SCHRITT 7

Ein weiteres Beispiel für die Typumwandlung ist die Anfrage nach Eingaben vom Benutzer (z. B. Zahlen). Geben Sie z. B. Folgendes ein:

```
age= input ("How old are you? ")
```

Alle mit dem input-Befehl gespeicherten Daten werden als String-Variable gespeichert.

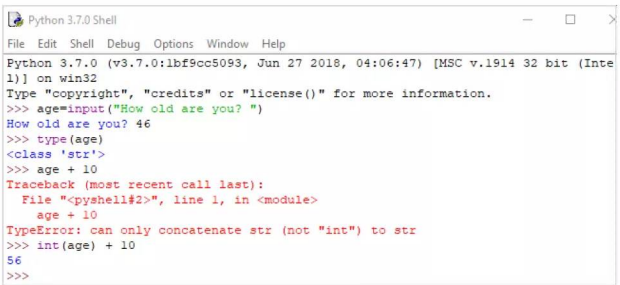


SCHRITT 8

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.

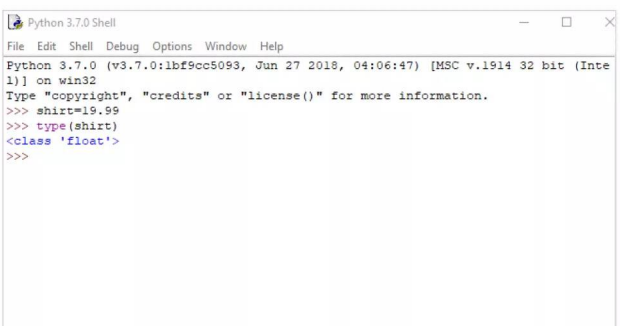


SCHRITT 9

Die Typumwandlung ist auch bei der Gleitkommaarithmetik wichtig, d. h. Zahlen mit einer Dezimalstelle. Geben Sie als Beispiel Folgendes ein:

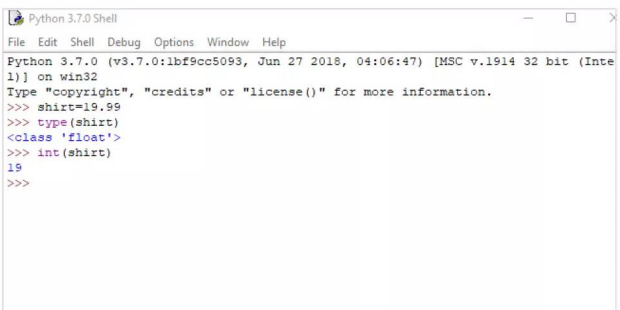
```
shirt=19.99
```

Geben Sie nun `type(shirt)` ein. Wie Sie sehen, hat Python die Zahl als „float" klassifiziert, da der Wert eine Dezimalzahl ist.



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.



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.

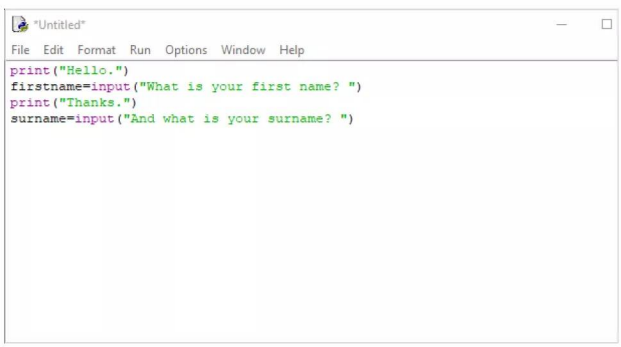
BENUTZERFREUNDLICH

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

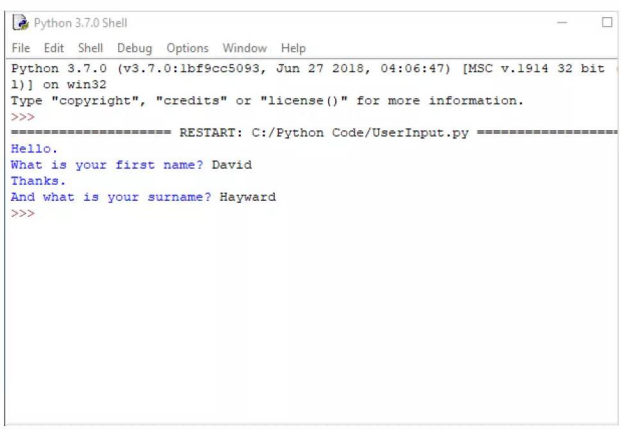
Öffnen Sie die Python 3 IDLE-Shell und öffnen Sie eine neue Datei im Editor. Wir beginnen mit etwas sehr Einfachem:

```
print("Hello")
firstname=input("What is your first name? ")
print("Thanks.")
surname=input("And what is your surname? ")
```



SCHRITT 2

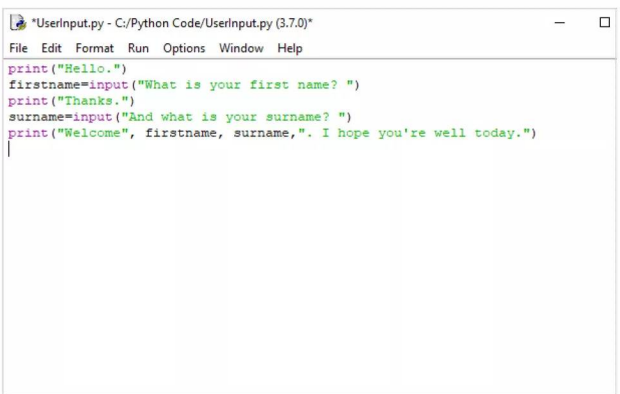
Speichern Sie den Code und führen ihn aus. Das Programm wird in der IDLE-Shell nach dem Vornamen fragen und diesen als Variablennamen speichern (**firstname**), gefolgt vom Nachnamen, der ebenfalls als eigene Variable (**surname**) gespeichert wird.



SCHRITT 3

Nachdem wir die Namen des Benutzers in Variablen gespeichert haben, können wir sie aufrufen, wann immer wir wollen:

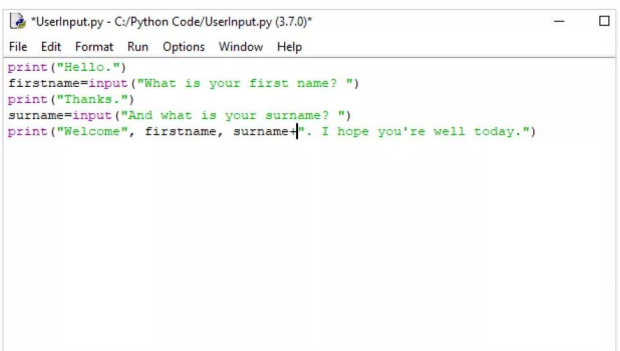
```
print("Welcome", firstname, surname, ". I hope
you're well today.")
```



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:

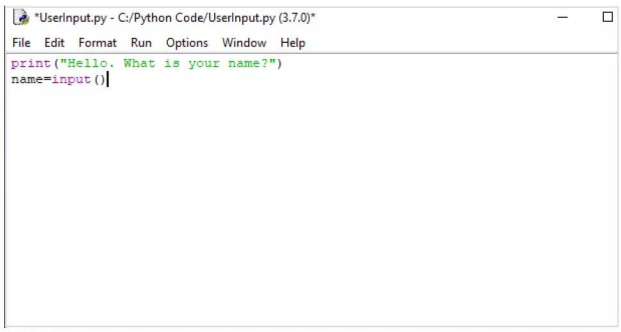
```
print("Welcome", firstname, surname+". I hope
you're well today.")
```



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

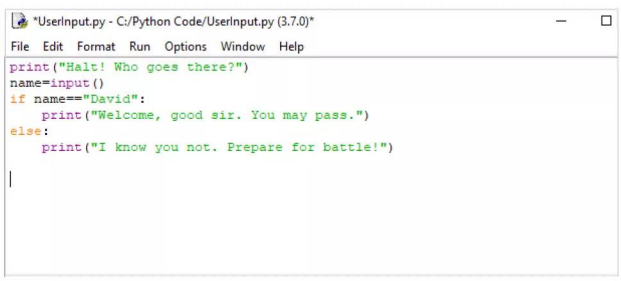
Der Code aus dem vorherigen Schritt wird oft als übersichtlicher angesehen, anstatt einen langen Text im Eingabebefehl zu haben, aber es ist keine in Stein gemeißelte Regel, gehen Sie daher in diesen Situationen so vor, wie Sie möchten. Erweitern Sie den Code, indem Sie Folgendes ausprobieren:

```
print("Halt! Who goes there?")
name=input()
```

SCHRITT 7

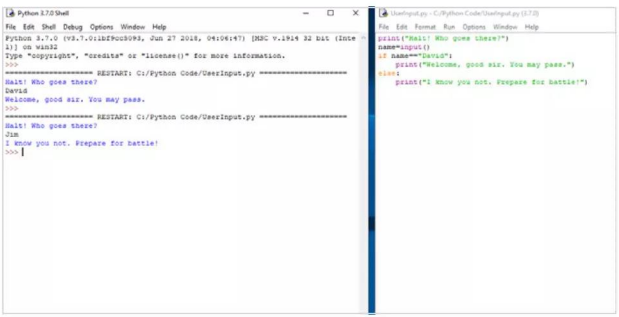
Dies ist nun ein guter Moment, um ein Textabenteuerspiel zu beginnen. Sie können den Code erweitern und mit der Eingabe des Benutzers das Spiel ein wenig konkretisieren:

```
if name=="David":
  print("Welcome, good sir. You may pass.")
else:
  print("I know you not. Prepare for battle!")
```



SCHRITT 8

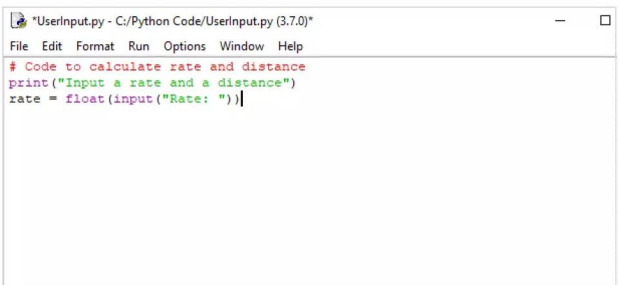
Was Sie hier erstellt haben, ist eine Bedingung, womit wir uns in Kürze näher befassen werden. Kurz gesagt, wir vergleichen die Eingaben des Benutzers gegen eine Bedingung. Wenn der Benutzer also David als seinen Namen eingibt, wird der Wächter ihm erlauben, ungehindert passieren zu können. Wird ein anderer Name als David eingegeben, fordert der Wächter ihn zum Kampf heraus.



SCHRITT 9

Wie Sie zuvor erfahren haben, ist jede Eingabe eines Benutzers automatisch ein String, also eine Zeichenkette. Sie müssen also per Typumwandlung den String in etwas anderes umzuwandeln. 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: "))
```

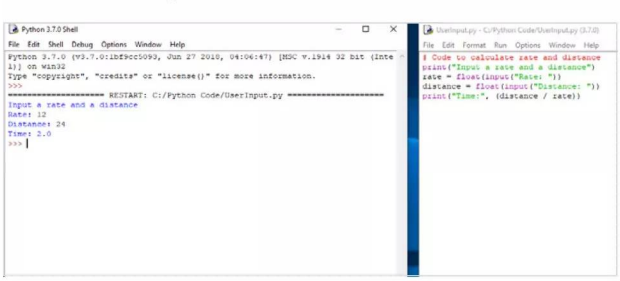


SCHRITT 10

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 anstatt 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, Python lässt sie aber auch eigene Funktionen bestimmen.

WAS SIND FUNKTIONEN?

Eine Funktion ist ein Befehl, der in Python eingegeben wird, 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 ändern und die Länge oder den Typ von Elementen überprüfen. Im Grunde ist eine Funktion ein kurzes Wort, dem Klammern folgen. Zum Beispiel **len()**, **list()** oder **type()**.

SCHRITT 2

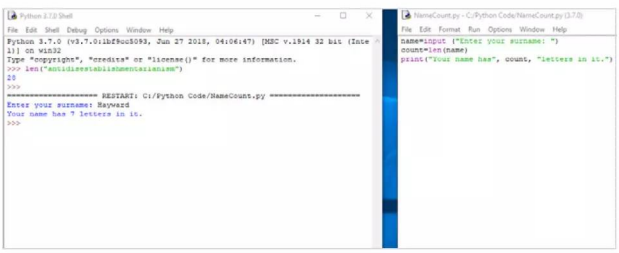
Eine Funktion nimmt Daten auf, normalerweise eine Variable, verarbeitet diese und gibt den Endwert zurück. Die zu bearbeitenden Daten werden in die Klammern eingefügt. Wenn Sie also wissen wollen, wie viele Buchstaben das Wort antidisestablishmentarianism enthält, würden Sie `len("antidisestablishmentarianism")` 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):

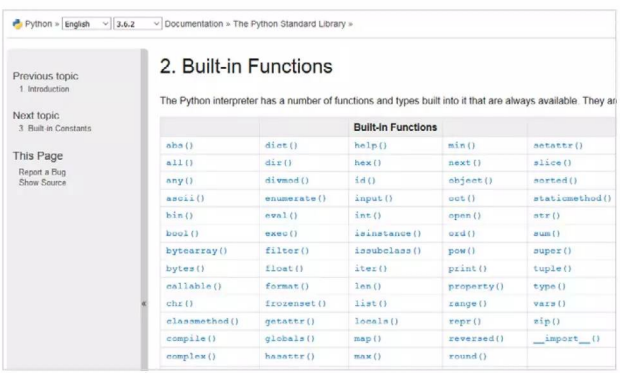
```
name=input ("Enter your surname: ")
count=len(name)
print ("Your surname has", count, "letters in it.")
```

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. Für z. B. erweiterte Mathematikfunktionen geben Sie Folgendes ein:

```
import math
```

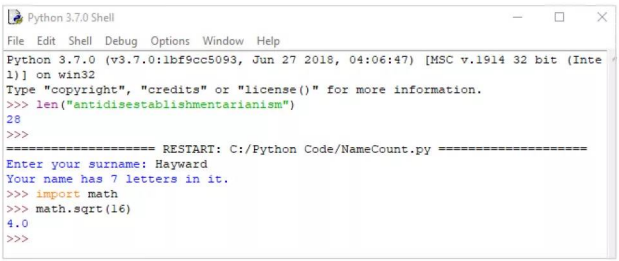
Nach der Eingabe haben Sie Zugriff auf alle Funktionen des Mathematikmoduls.

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()
```

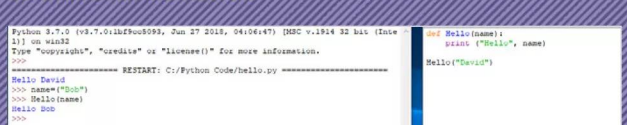
Drücken Sie **F5**, um das Skript zu speichern und auszuführen. Sie können Hello in der Shell sehen. Wenn Sie Hello() eingeben, wird die neue Funktion zurückgegeben.

SCHRITT 2

Wir erweitern nun die Funktion, um eine Variable zu akzeptieren, z. B. den Namen des Benutzers. Ändern Sie Ihr Skript 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")** und dann **Hello(name)**. Es können nun Variablen durch Ihre Funktion laufen.

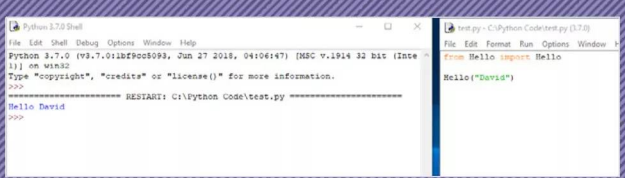


SCHRITT 3

Löschen Sie nun die Zeile **Hello("David")**, die letzte Zeile im Skript, und drücken Sie **Strg + S**, um das neue Skript 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 Programm zu erreichen versuchen.

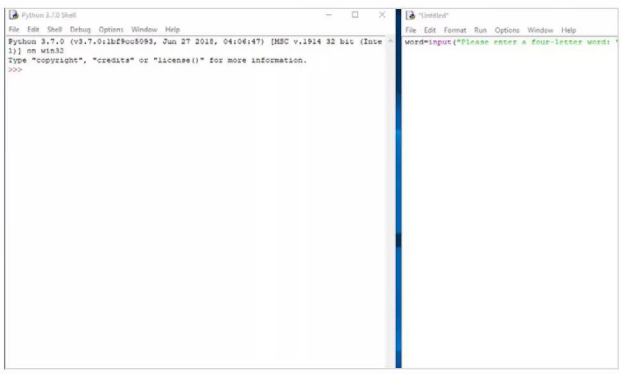
WAHRE BEDINGUNGEN

Wenn zu Beginn die Bedingungen einfach gehalten werden, macht das Programmieren lernen 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:

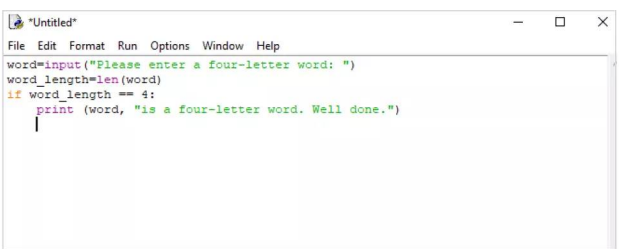
```
word=input("Please enter a four-letter word: ")
word_length=len(word)
```

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.")
```

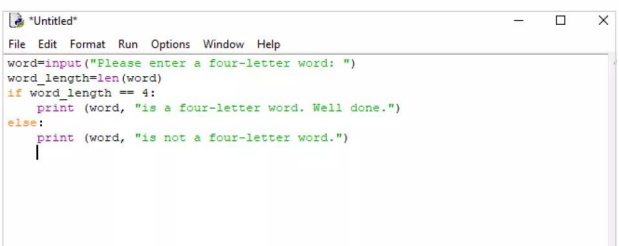
Das doppelte Gleichheitszeichen (==) überprüft, ob eine Übereinstimmung herrscht.



SCHRITT 4

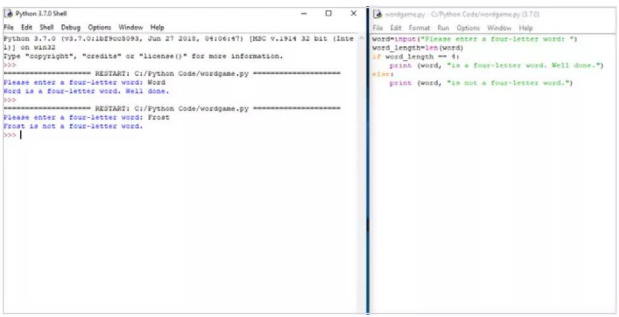
Der Doppelpunkt am Ende von if teilt Python mit, dass alles nach dem eingerückten Doppelpunkt ausgeführt wird, wenn die Anweisung wahr ist. Bewegen Sie nun den Cursor an den Anfang des Editors:

```
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

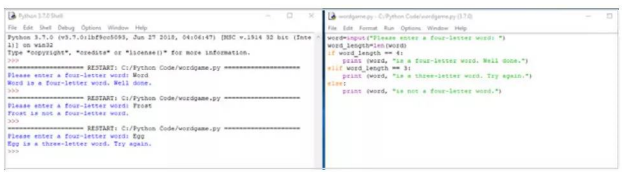
Drücken Sie **F5** und speichern Sie den Code, um ihn auszuführen. Geben Sie ein aus vier Buchstaben bestehendes Wort in die Shell ein. Sie sollten die Nachricht erhalten, dass es sich um vier Buchstaben handelt. Drücken Sie erneut **F5**, aber geben Sie diesmal ein aus fünf Buchstaben bestehendes Wort 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.")
```



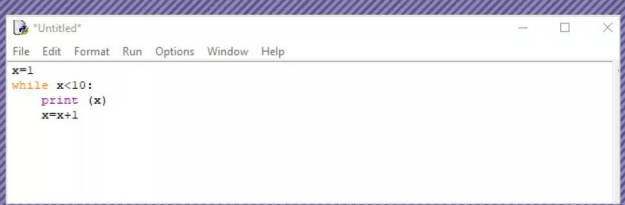
SCHLEIFEN

Eine Schleife ähnelt einer Bedingung, unterscheidet sich jedoch in ihrer Funktionsweise. Eine Schleife läuft mehrere Male den gleichen Code durch und wird dabei normalerweise von einer Bedingung unterstützt.

SCHRITT 1

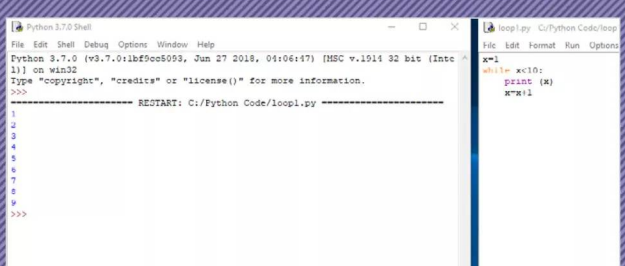
Wir beginnen mit einer einfachen while-Anweisung. Wie if prüft diese, ob etwas wahr ist, und führt dann den eingerückten Code aus:

```
x = 1
while x < 10:
   print (x)
   x = x + 1
```



SCHRITT 2

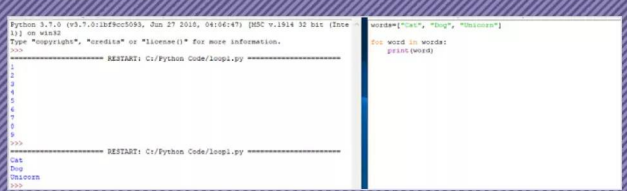
Der Unterschied zwischen **if** und **while** ist, dass while zurückgeht und prüft, ob die Aussage noch wahr ist, wenn das Ende des eingerückten Codes erreicht wird. In unserem Beispiel ist x kleiner als 10. Bei jeder Schleife wird der aktuelle Wert von x ausgegeben und um eins addiert. Wenn x schließlich 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 zuvor bereits erwähnt, z. B. das Mathematikmodul, aber da Module für die optimale Nutzung von Python wichtig sind, ist es sinnvoll, ihnen etwas mehr Zeit zu widmen.

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-Skripte, z. B. **import math**.

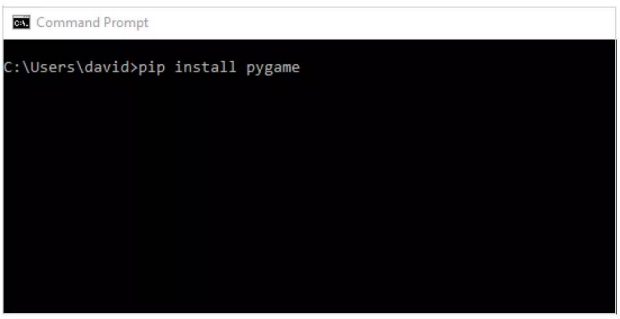
SCHRITT 2

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

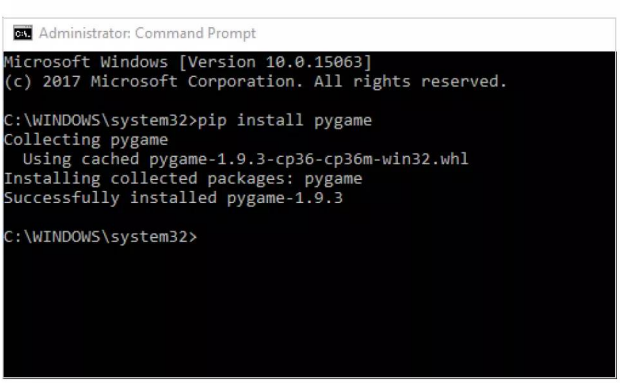
Als Ergebnis erhalten Sie eine Fehlermeldung in der IDLE-Shell, da das Pygame-Modul in Python nicht erkannt wird oder nicht installiert ist. Um ein Modul zu installieren, verwenden wir PIP (Pip Installs Packages). Schließen Sie die IDLE-Shell und öffnen Sie die Eingabeaufforderung oder das Terminal. 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

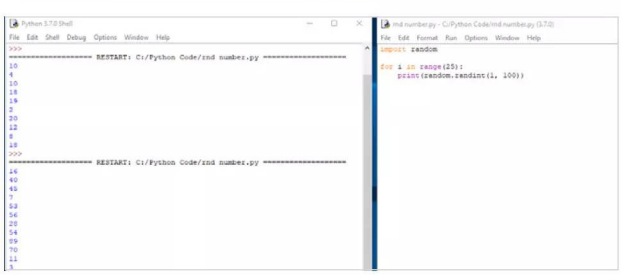
Die Module enthalten den zusätzlichen Code, der benötigt wird, um ein bestimmtes Ergebnis im eigenen Code zu erzielen, wie wir es bereits zuvor getestet haben. Der Code `import random` importiert z. B. den Code aus dem Zufallsgeneratormodul. Mit diesem Modul können Sie dann Folgendes erstellen:

```
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:

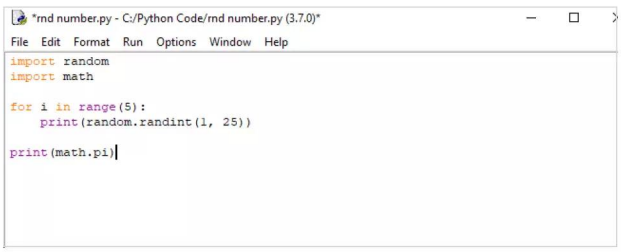
```
import random

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.:

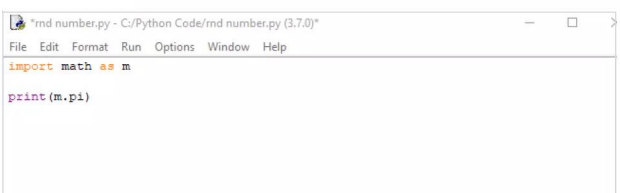
```
from random import randint

for i in range(5):
  print(randint(1, 25))
```

SCHRITT 10

Dies hilft einen optimalen Ansatz für die Programmierung zu erstellen. Sie können auch **import modul*** 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.



C++ wurde vor über 35 Jahren eingeführt und gilt als eine der am meisten genutzten und leistungsfähigsten Programmiersprachen. Es wird Ihnen vielleicht nicht bewusst sein, aber das meiste, das Sie vor sich sehen, wenn Sie an einem Computer arbeiten, wurde mit C++ programmiert.

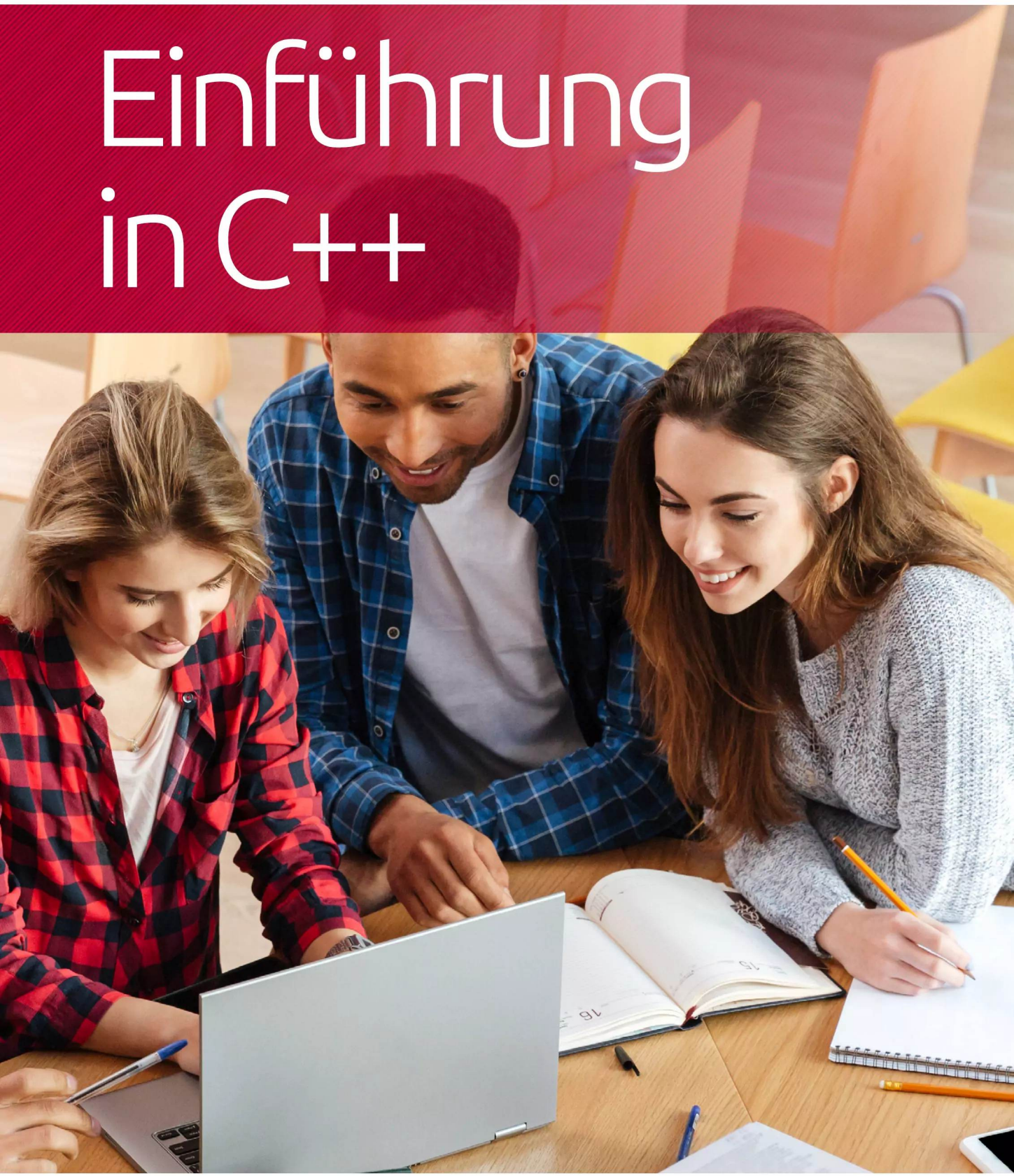
Ganze Betriebssysteme, Office-Pakete, Spiele und sogar die Benutzeroberfläche von Spielekonsolen und Smart-TVs wurden entweder ausschließlich mit C++ oder mit einer Kombination aus C++ und einer anderen Sprache entwickelt. Man kann also sagen, dass das Erlernen von C++ ein Muss für das neue Jahrzehnt ist.

In diesem Abschnitt stellen wir Ihnen C++ vor und helfen Ihnen beim Einstieg in diese bemerkenswerte Sprache.

„Der wichtigste Aspekt der Softwareentwicklung besteht darin, sich klar zu machen, was man zu erstellen versucht."

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

Einführung in C++



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.

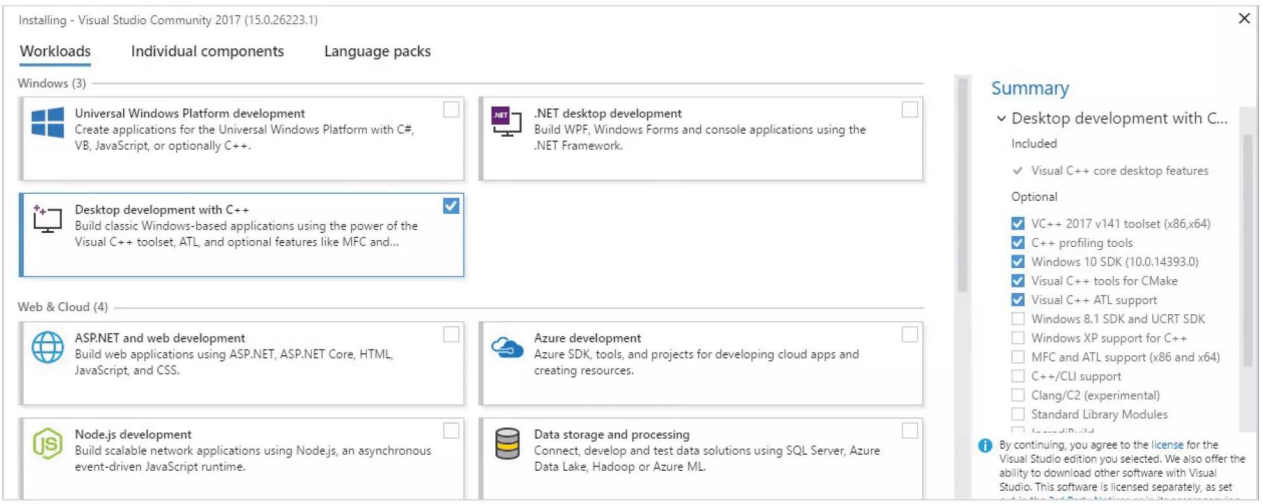
Im Jahr 1979, dem Beginn des goldenen Zeitalters des Home-Computings, hat der dänische Informatiker Bjarne Stroustrup während seiner Doktorarbeit C++ bzw. „C mit Klassen" entwickelt. Stroustrups Plan bestand darin, die ursprüngliche C-Sprache zu erweitern, die seit den frühen 70er Jahren weit verbreitet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

Ferner ist C++ 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 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 steht kostenlos zum Download und zur Nutzung zur Verfügung. Sie können C++ von jedem Betriebssystem aus nutzen, z. B. macOS, Linux, Windows oder mobile Plattformen.

Genau wie bei Python lässt sich die Frage „Warum C++?“ 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

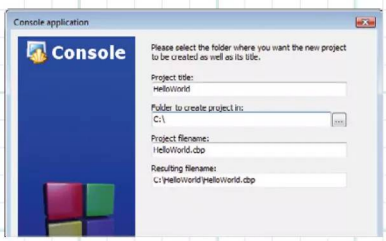
Das Lernen von C++ erfordert weder große Investitionen noch eine komplette EDV-Einrichtung. Sie benötigen lediglich einen relativ modernen Computer, alles andere ist kostenlos erhältlich.

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.

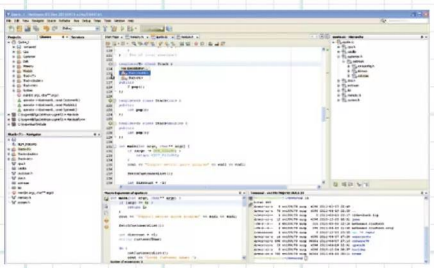


IDE

Wie auch bei Python wird eine IDE verwendet, um den C++-Code einzugeben und auszuführen. 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.

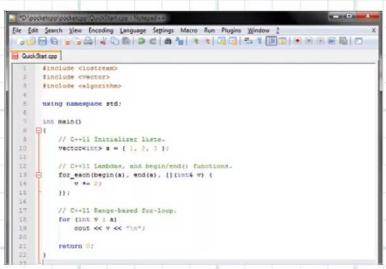
COMPILER

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 Code-Texteditoren.



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

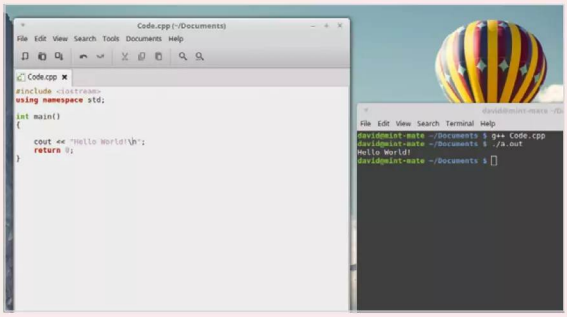
Genau wie bei Python müssen Sie viel Zeit für das Erlernen des Programmierens in C++ einplanen. Es wird leider nicht über Nacht oder in einer Woche passieren. Ein guter C++-Programmierer hat viele Jahre damit verbracht, sein Handwerk zu lernen. Seien Sie also geduldig, fangen Sie klein an und lernen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

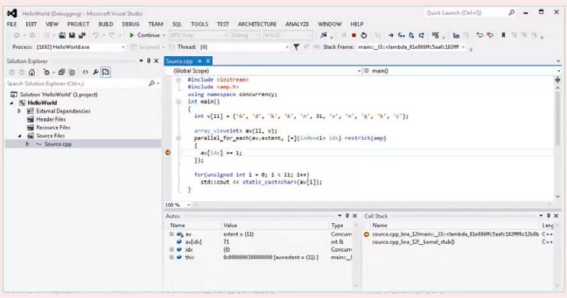
LINUX

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.



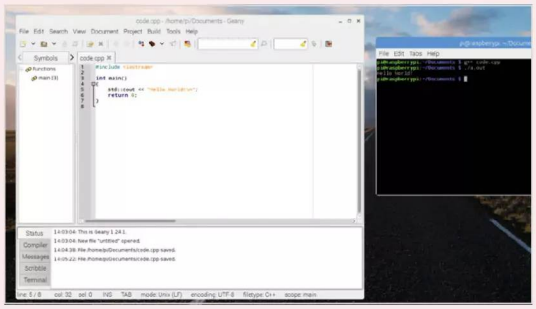
WINDOWS

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.



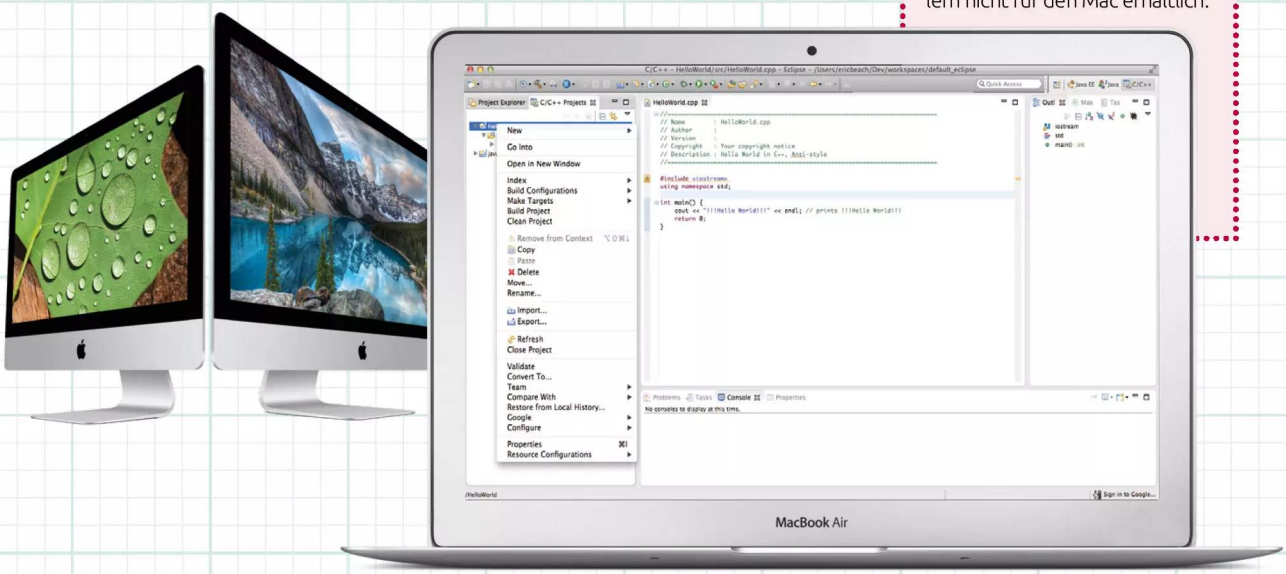
RASPBERRY PI

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.



Lernen Sie C++ kennen

Wer ein Games Designer werden oder an den neuesten Technologien in den Bereichen Wissenschaft oder Maschinenbau arbeiten möchte, muss in C++ programmieren können. Um das Programmieren zu lernen, ist man aber nie zu alt.

#INCLUDE <C++ IST KLASSE!>

Das Lernen der Grundlagen der Programmierung, z. B. durch Python, ermöglicht Ihnen, die Struktur eines Programms zu verstehen. Auch bei unterschiedlichen Befehlen werden Sie erkennen, wie der Code funktioniert.

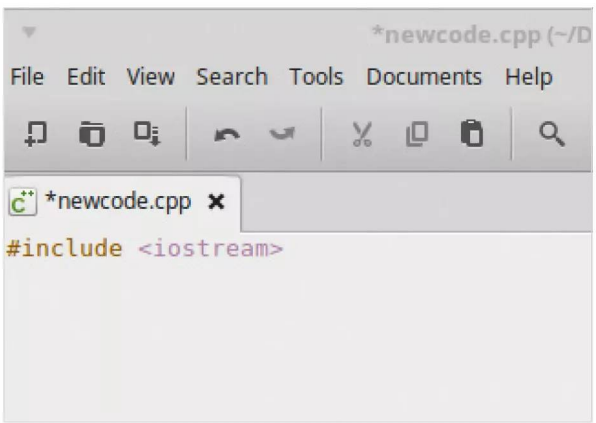
C++

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

Bjarne Stroustrup, der Erfinder von C++.

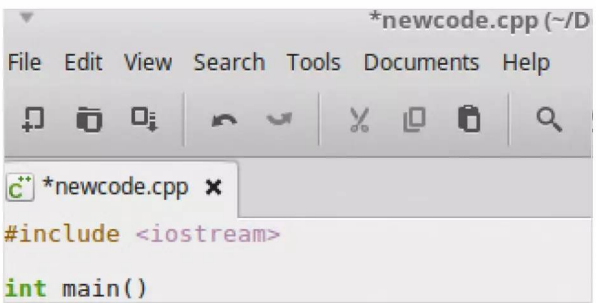
#INCLUDE

Die C++-Programmstruktur unterscheidet sich nur geringfügig von der Python-, aber grundlegend von der BASIC-Struktur. 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.



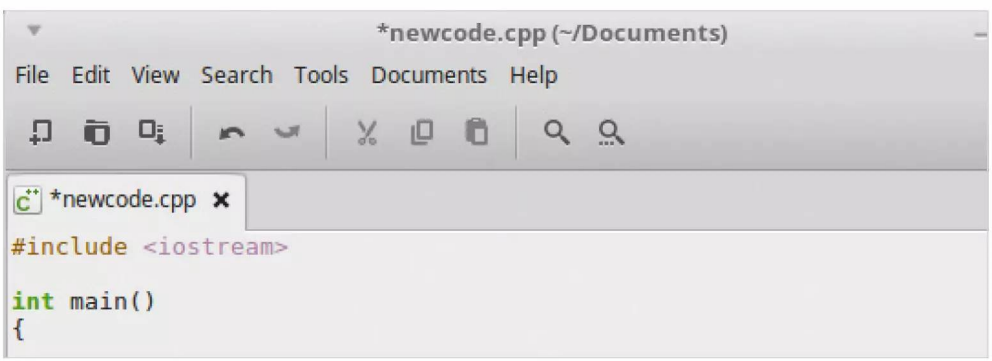
INT MAIN()

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



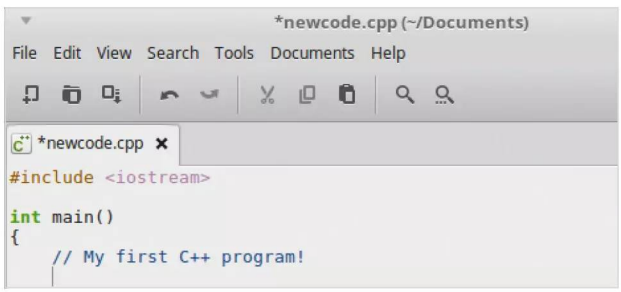
KLAMMERN

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



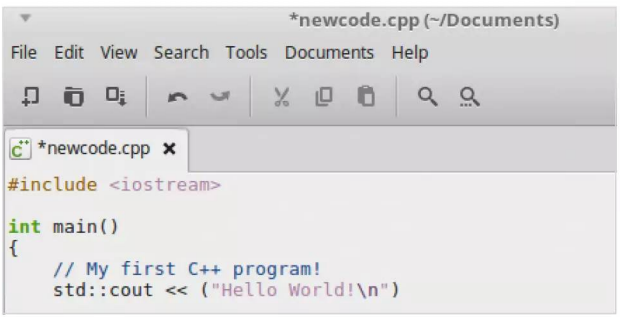
KOMMENTARE

Zeilen, die mit einem doppelten Schrägstrich beginnen, sind Kommentare. Dies bedeutet, dass sie nicht im Code ausgeführt und vom Compiler ignoriert werden. Wozu sind sie da? 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.



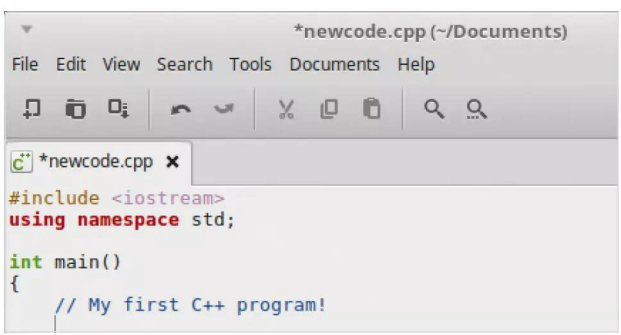
<<

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.

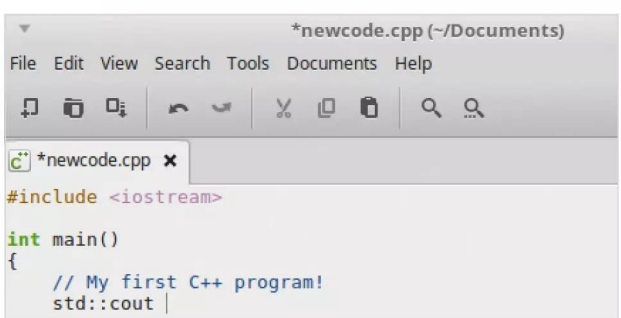


AUSGABEN

Bei Ausführung des Codes soll der Teil ("Hello World!") auf dem Bildschirm angezeigt werden. Sie können eingeben, was immer Sie möchten, solange es sich in den Anführungszeichen befindet. Die Klammern werden nicht benötigt, aber einige Compiler bestehen darauf. Der \n-Teil zeigt an, dass eine neue Zeile eingefügt werden soll.

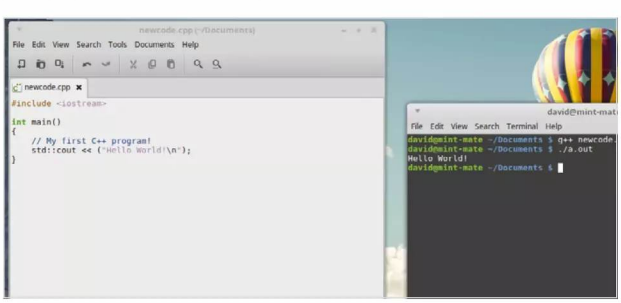
COUT

In diesem Beispiel verwenden wir „cout", das Teil des Standard-Namensraums ist, weshalb es hier auch vorhanden ist. Cout bedeutet Character OUTput und zeigt oder gibt etwas auf dem Bildschirm an bzw. aus. Wenn wir `std::` weglassen, müssen wir es, wie bereits erwähnt, am Anfang des Codes deklarieren.



; UND }

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



C++ unter Windows einrichten

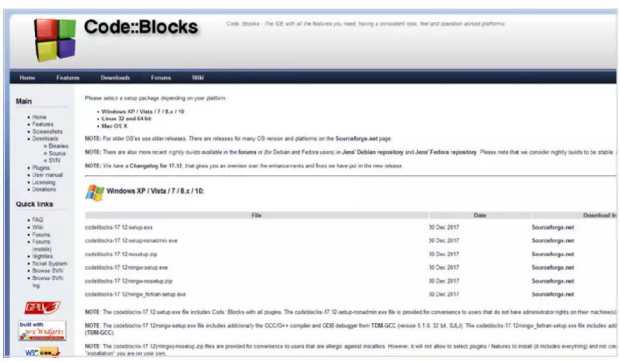
Windows-Anwender haben bei der Programmierung in C++ eine große Auswahl. Es gibt zahlreiche IDEs und Compiler, einschließlich Visual Studio von Microsoft. Unserer Meinung nach ist Code::Blocks jedoch die beste C++-IDE für den Einstieg.

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.

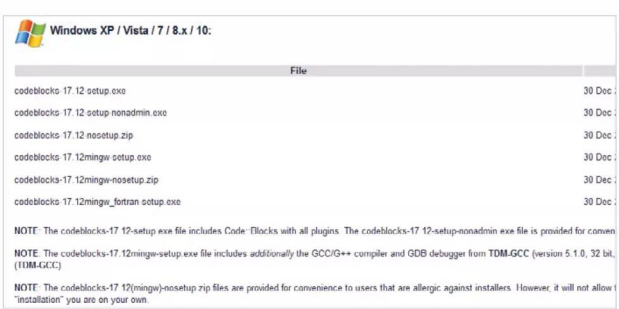
SCHRITT 1

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 dieser Text 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.

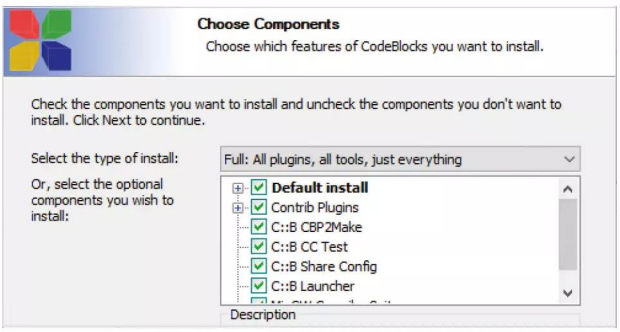


SCHRITT 3

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.

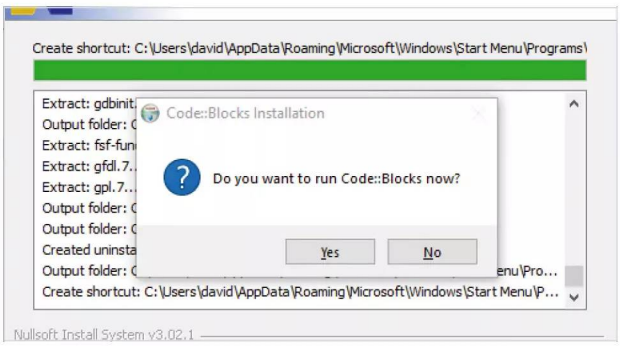
SCHRITT 4

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, sich für die vollständige Option, also Full, zu 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

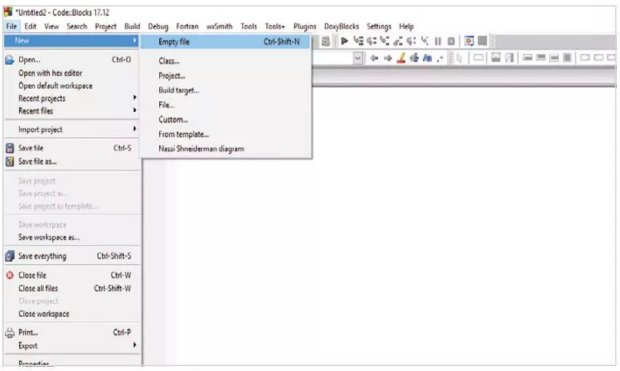
Beim ersten Hochfahren von Code::Blocks wird eine automatische Erkennung für alle C++-Compiler ausgeführt, die Sie evtl. bereits auf Ihrem System installiert haben. Wenn Sie noch keinen haben, wählen Sie die erste erkannte Option, GNU GCC Compiler, und klicken Sie auf die Default-Schaltfläche, um ihn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

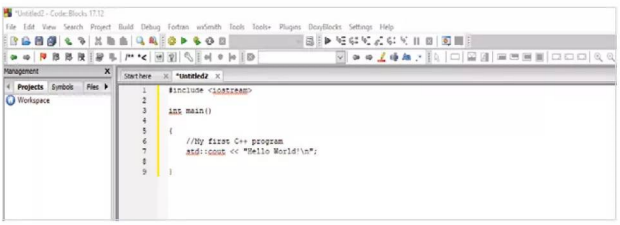


SCHRITT 9

Geben Sie in dem Fenster Folgendes ein:

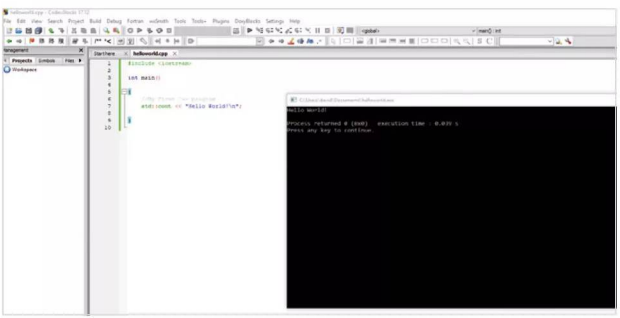
```
#include <iostream>

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

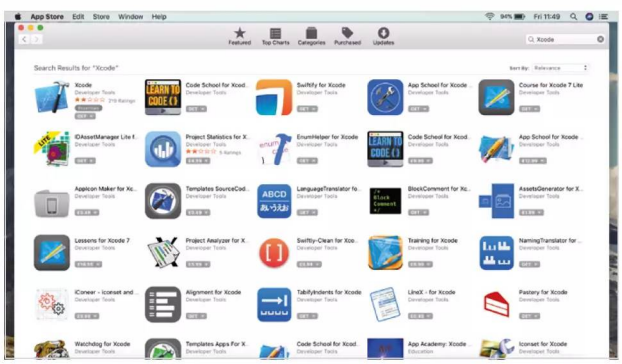
Um auf einem Mac in C++ zu programmieren, müssen Sie Xcode von Apple installieren, eine kostenlose, voll funktionsfähige IDE, mit der native Apple-Apps erstellt werden. Sie kann aber auch relativ einfach zum Erstellen von C++-Code verwendet werden.

XCODE

Apples Xcode wurde hauptsächlich für Benutzer entwickelt, die Apps für macOS, iOS, tvOS und watchOS-Apps in Swift oder Objective-C entwickeln. Sie ist aber auch für C++ geeignet.

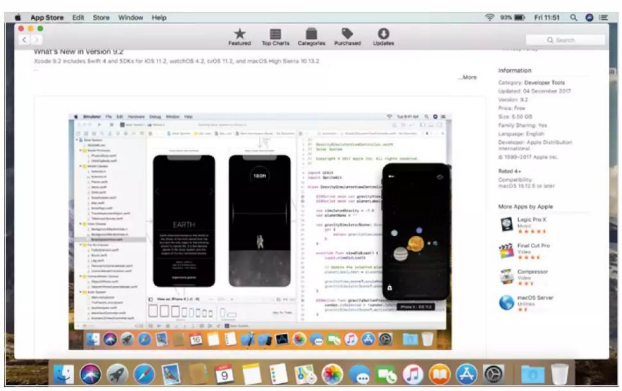
SCHRITT 1

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.

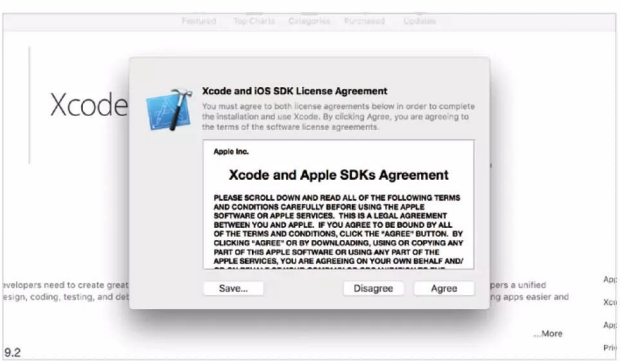


SCHRITT 3

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

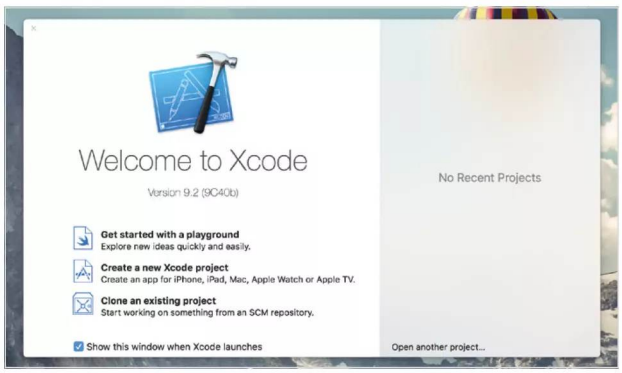
SCHRITT 4

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.



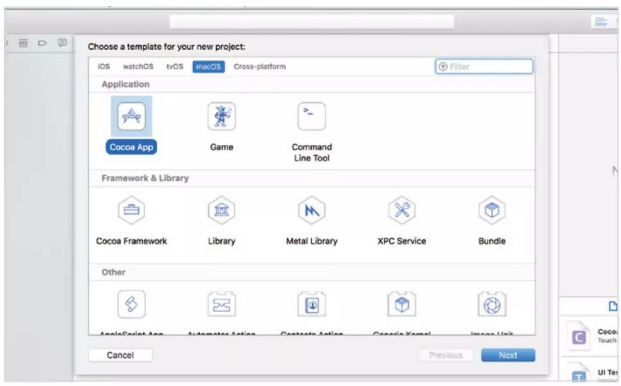
SCHRITT 5

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.



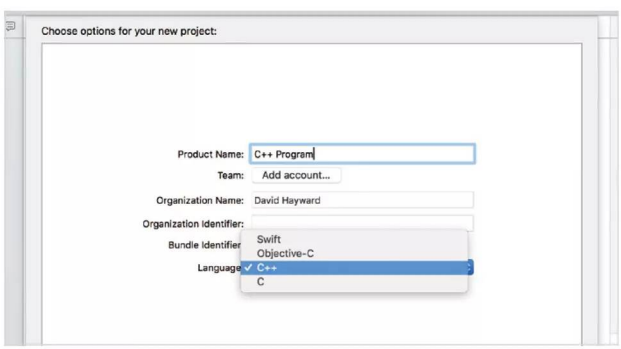
SCHRITT 6

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.



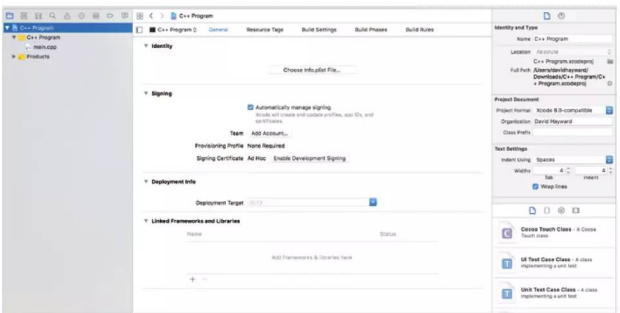
SCHRITT 7

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.



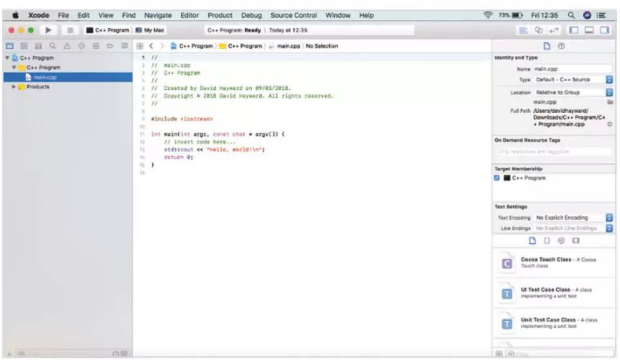
SCHRITT 8

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.



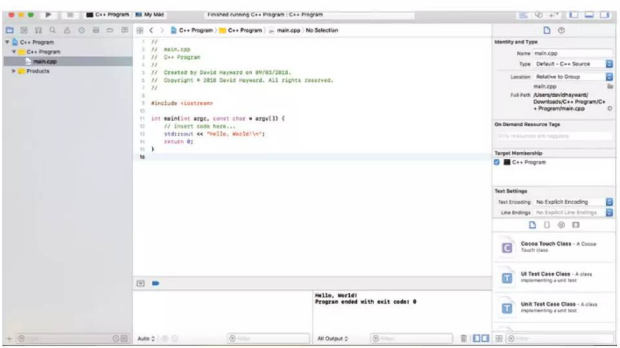
SCHRITT 9

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, welche die Inhalte verwendet, die für Ihren Mac verfügbar sind.



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. Wenn das Programm ausgeführt wird, wird die Ausgabe am unteren Rand des Xcode-Fensters angezeigt.



C++ unter Linux einrichten

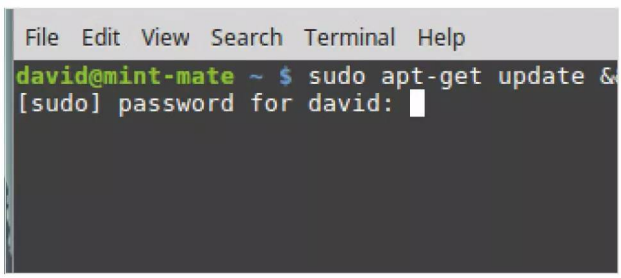
Linux bietet eine tolle Programmierumgebung für C++. In den meisten Linux-Distributionen sind die wichtigen Komponenten wie der Compiler bereits vorinstalliert. Die Texteditoren eignen sich hervorragend für die Codeeingabe, einschließlich der Farbcodierung. Es gibt auch viele zusätzliche Software, die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

LINUX++

Für dieses spezielle Tutorial verwenden wir eine neue Installation von Linux Mint. Weitere Informationen zu Linux Mint finden Sie im nächsten Abschnitt.

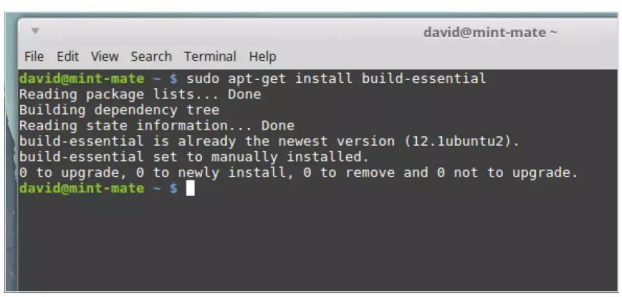
SCHRITT 1

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. Diese Befehle aktualisieren das gesamte System und die installierte Software.



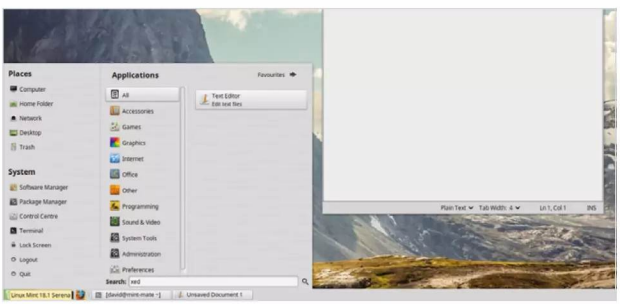
SCHRITT 2

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

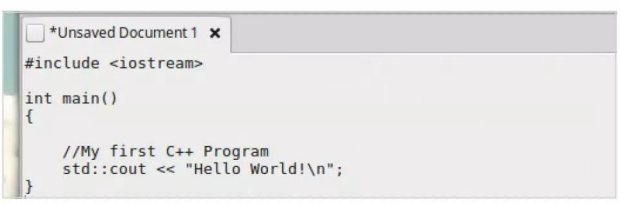
Und das war's auch schon. Es ist nun alles bereit, um mit dem Programmieren beginnen zu können. Um Ihr erstes C++-Programm auszuführen, starten Sie Xed, den Haupteditor in Linux Mint. Öffnen Sie dazu in Linux Mint das Menü und geben Sie `Xed` in die Suchleiste ein. Klicken Sie im rechten Bereich auf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



SCHRITT 4

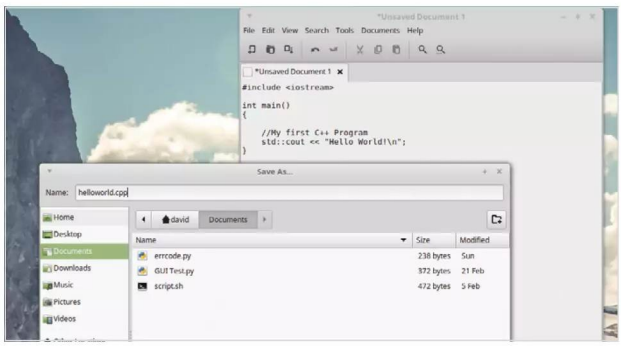
Geben Sie in Xed bzw. im Texteditor, den Sie verwenden, die Codezeilen ein, aus denen Ihr C++-Hello- World-Programm besteht. Hier noch mal zur Erinnerung:

```
#include <iostream>

int main()

{

//My first C++ program

std::cout <<  “Hello World!\n”;
```



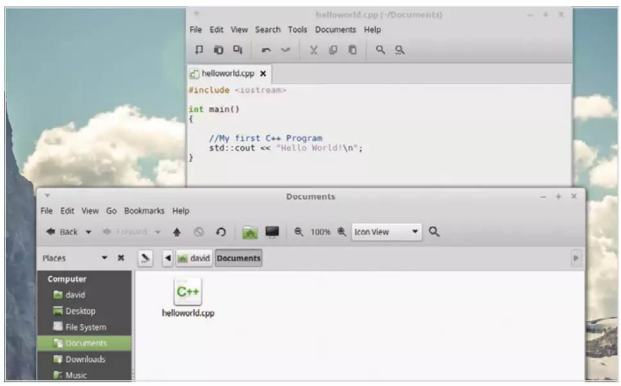
SCHRITT 5

Wenn Sie Ihren Code eingegeben haben, klicken Sie auf 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.



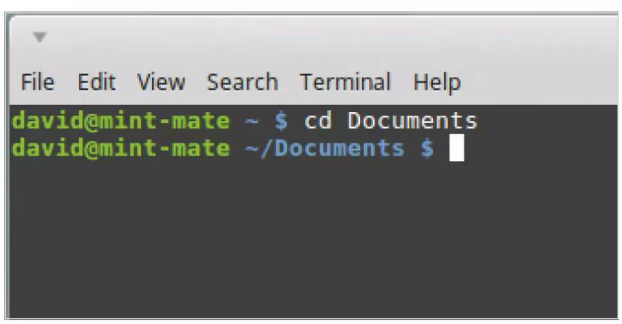
SCHRITT 6

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.



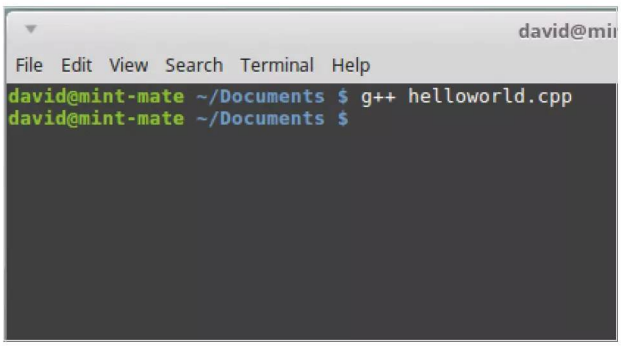
SCHRITT 7

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.



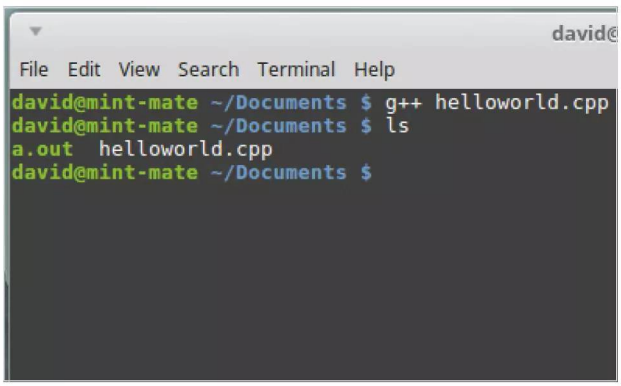
SCHRITT 8

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.



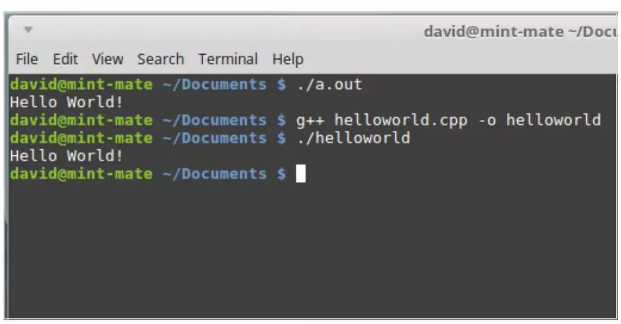
SCHRITT 9

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 IDEs für C++

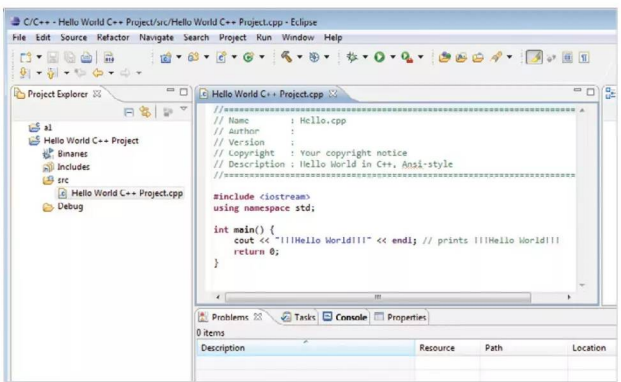
Wenn Sie die Arbeit mit Ihrem C++-Code anderweitig angehen möchten, stehen Ihnen zahlreiche Optionen zur Verfügung. Windows ist die am weitesten verbreitete Plattform für C++-IDEs, aber auch für Mac- und Linux-Benutzer stehen viele zur Auswahl.

ENTWICKLUNGSUMGEBUNGEN FÜR C++

Hier sind zehn tolle IDEs für C++, 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.

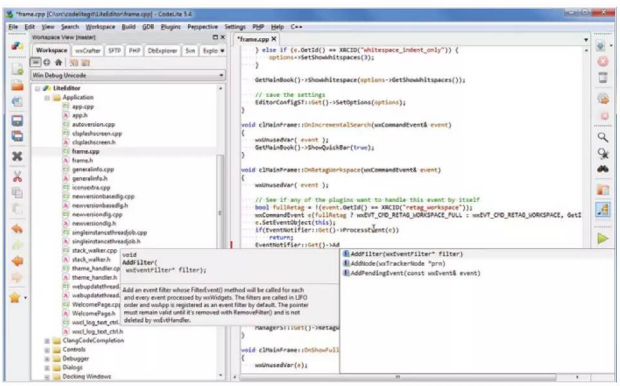
ECLIPSE

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.



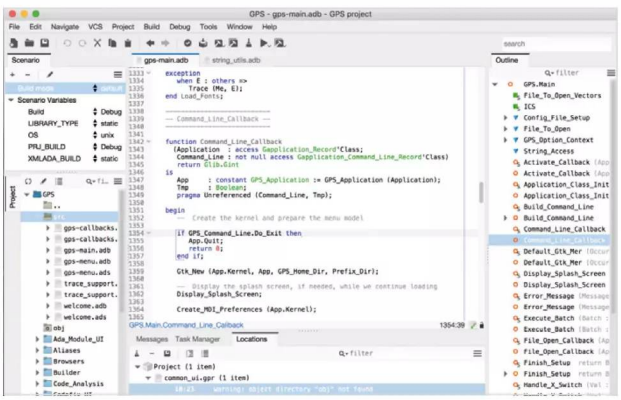
CODELITE

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/**.



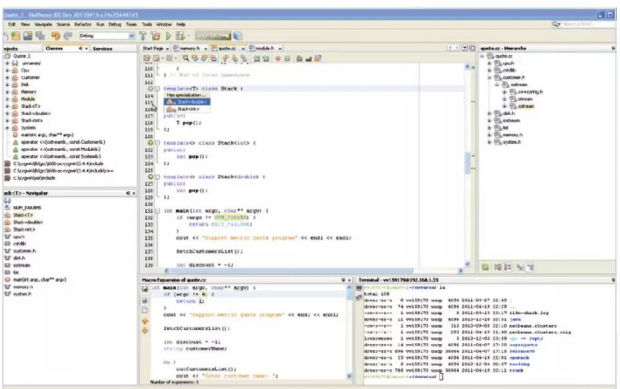
GNAT

Das GNAT Programming Studio (GPS) ist eine leistungsstarke und intuitive IDE, die das Testen, Debugging und die Codeanalyse 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**.



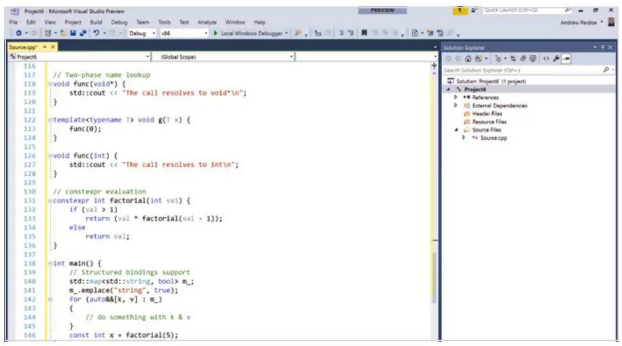
NETBEANS

Eine weitere beliebte Wahl ist NetBeans, eine exzellente IDE, die voll mit Funktionen ist und Spaß in 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**.



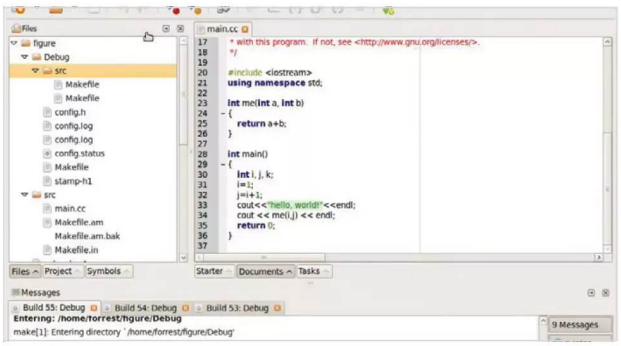
VISUAL STUDIO

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.



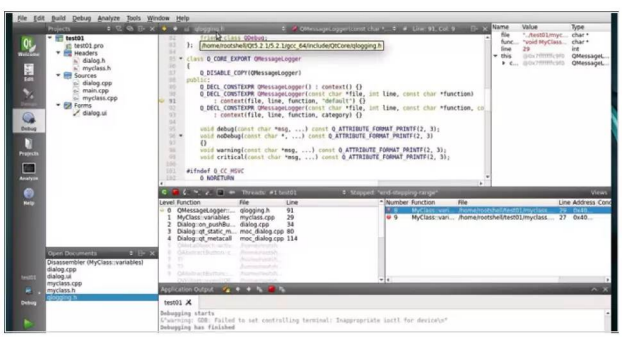
ANJUTA

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/**.



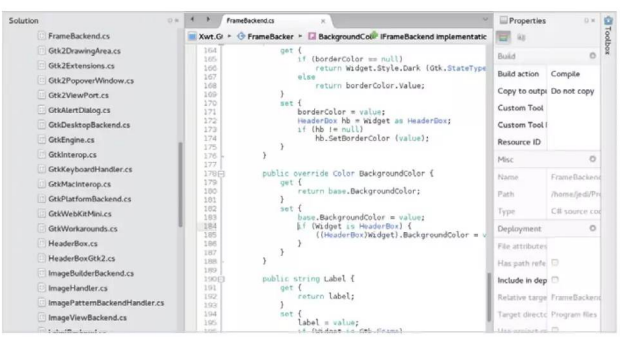
QT CREATOR

Diese plattformübergreifende IDE wurde entwickelt, um C++-Anwendungen für Desktop- und mobile Umgebungen zu erstellen. Sie enthält einen Codeeditor 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/**.



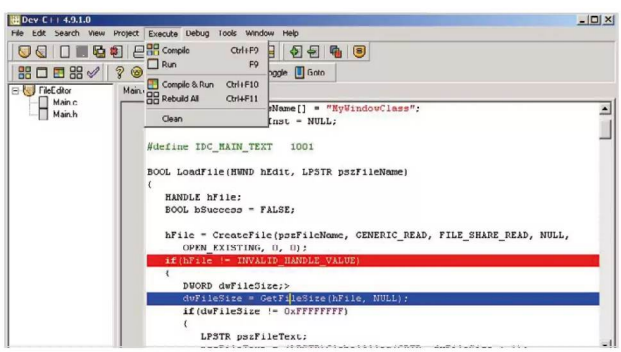
MONODEVELOP

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 Codeerstellung hilft. Sie ist für Windows, Mac und Linux verfügbar und kann kostenlos heruntergeladen und genutzt werden: **www.monodevelop.com/**.



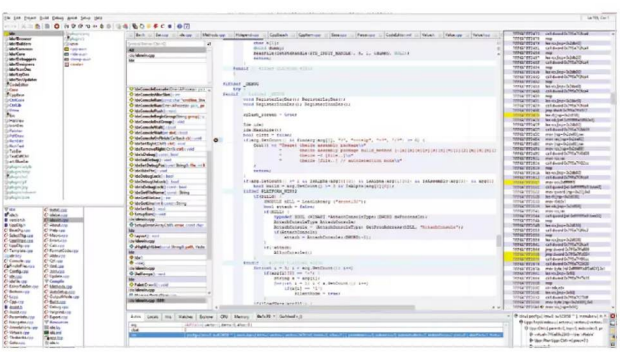
DEV C++

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 Codeerstellung 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++ kann sich beim Erlernen als komplexe Sprache erweisen, aber selbst der kompetenteste und professionellste C-Programmierer hat einmal klein angefangen.

In diesem Abschnitt befassen wir uns daher mit den Grundlagen für den Einstieg in C++. Lernen Sie, wie Sie Ihren eigenen C++-Code kompilieren und ausführen, und machen Sie anschließend mit Variablen, Datentypen, Strings, Mathematik, Benutzerinteraktion etc. weiter.

Mit C++ lässt sich Vieles erreichen, aber Sie müssen zuerst eine gute Basis schaffen, auf der Sie Ihre zukünftigen Fähigkeiten aufbauen können.

„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)

Programmieren mit C++

Ihr erstes C++- Programm

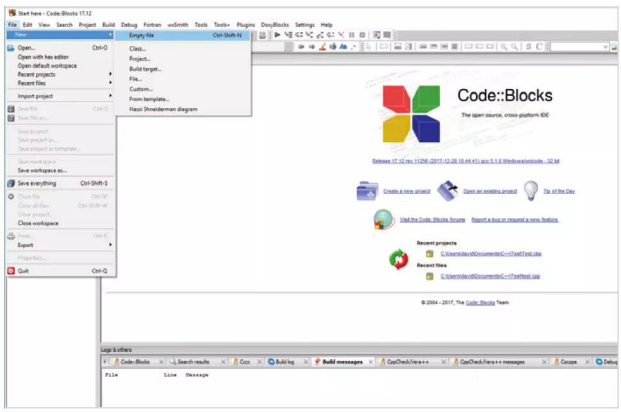
Möglicherweise haben Sie die Mac- und Linux-Beispiele zuvor bereits befolgt, wir werden ab jetzt jedoch ausschließlich in Windows und Code::Blocks arbeiten. Wir beginnen mit dem Erstellen unseres ersten C++-Programms.

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

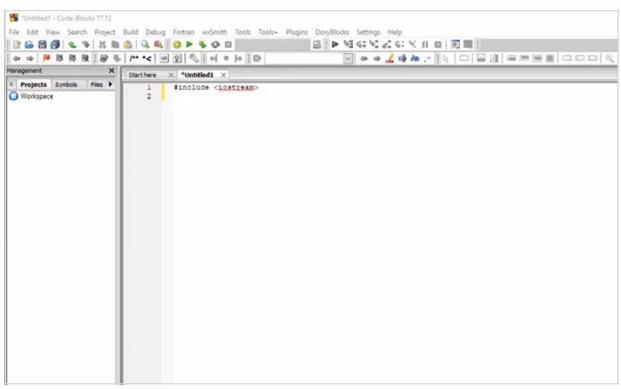
Wie bereits erwähnt, verwenden wir Windows 10 und die neueste Version von Code:: Blocks für diesen Abschnitt. Starten Sie Code::Blocks und klicken Sie auf File > New > Empty File oder drücken Sie die Tastenkombination Strg + Umschalt + N.



SCHRITT 2

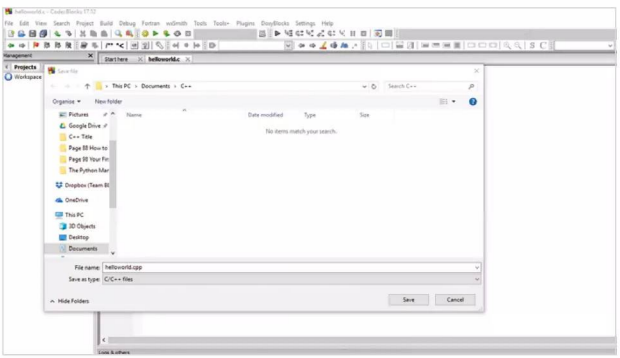
Sie sehen nun einen leeren Bildschirm mit dem Tab *Untitled1 und der Nummer 1 oben links im Hauptfenster von Code::Blocks. Machen Sie einen Klick im Hauptfenster, damit sich der Cursor neben der Nummer 1 befindet, und geben Sie Folgendes ein:

```
#include <iostream>
```



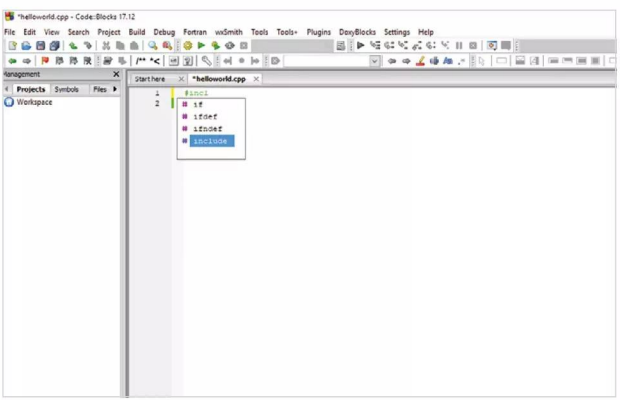
SCHRITT 3

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.

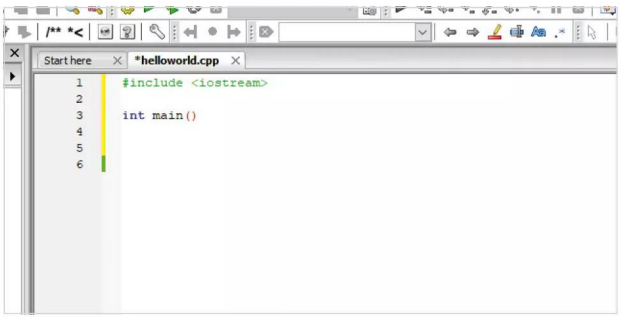


SCHRITT 5

Die automatische Auswahl von Befehlen ist äußerst praktisch und vermeidet eventuelle 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.

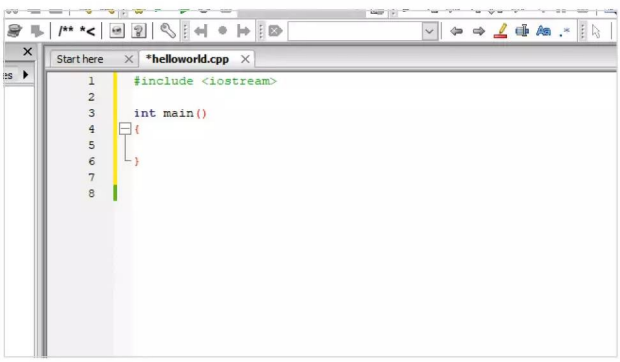


SCHRITT 6

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

```
{
```

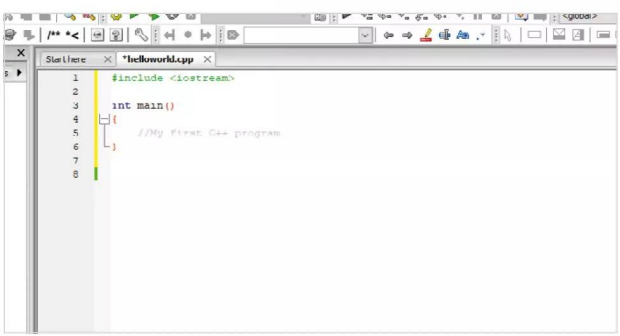
Die geschweiften Klammern erhalten Sie über die Tastenkombination Alt Gr + 7 und Alt Gr + 0.



SCHRITT 7

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

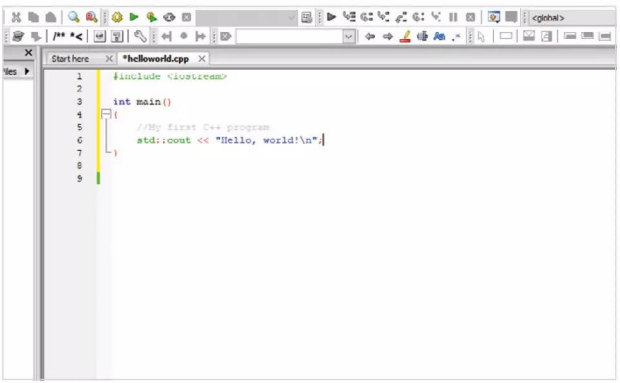
```
//My first C++ program
```



SCHRITT 8

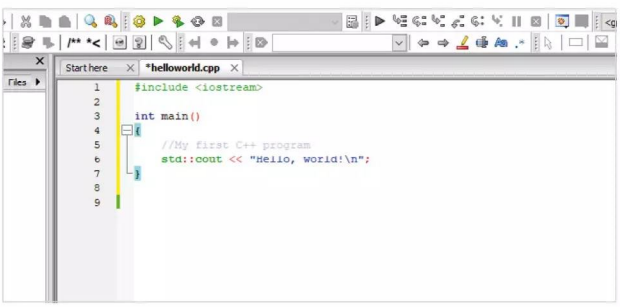
Beachten Sie erneut die veränderte Farbcodierung. Drücken Sie am Ende der Zeile des vorherigen Schritts die Eingabetaste und geben Sie Folgendes ein:

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



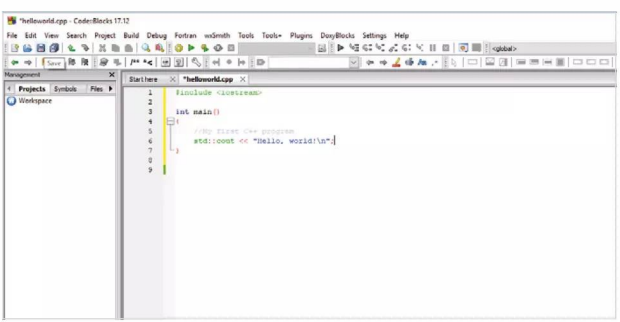
SCHRITT 9

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.



Kompilieren & Ausführen

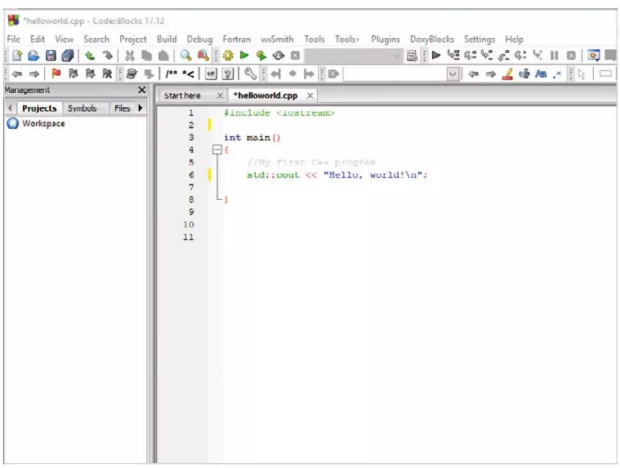
Sie haben Ihr erstes C++-Programm erstellt und verstehen nun seine strukturellen Grundlagen. Wir bringen nun die Dinge ins Rollen und werden das Programm kompilieren und ausführen und uns anschauen, was dabei herauskommt.

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.

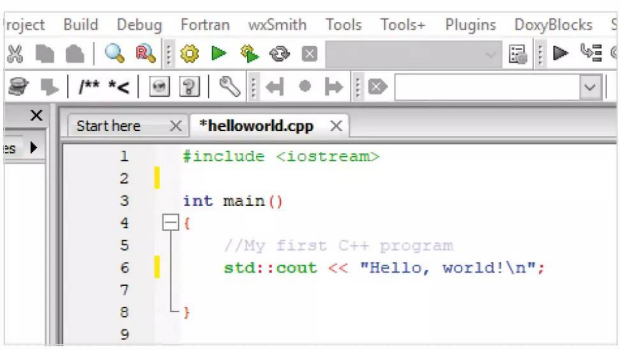
SCHRITT 1

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.



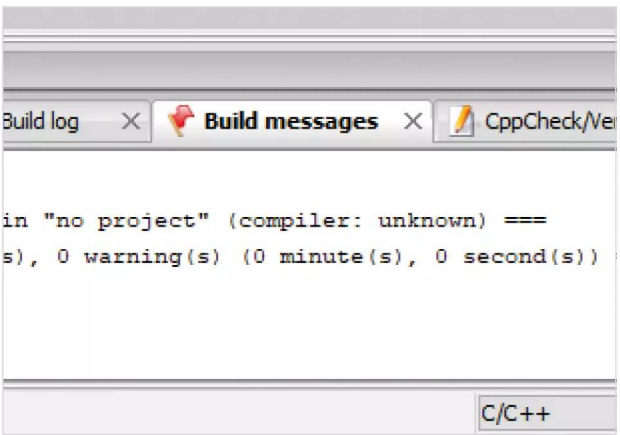
SCHRITT 2

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.



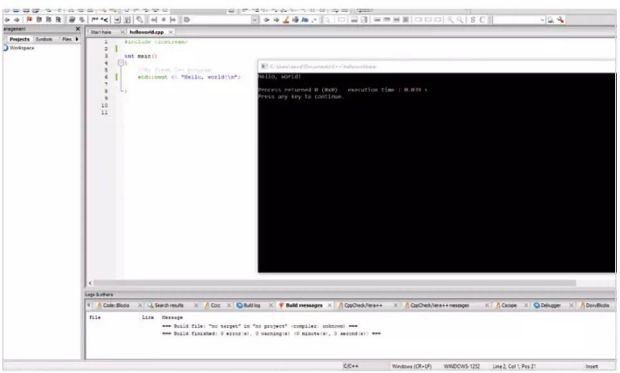
SCHRITT 3

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.



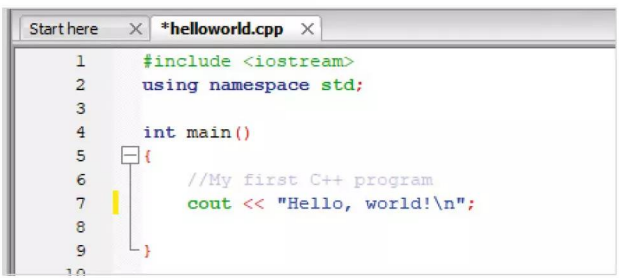
SCHRITT 5

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;
```

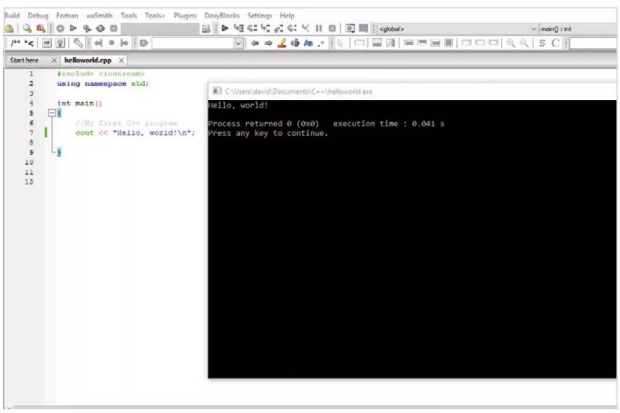
Löschen Sie anschließend den std::-Teil der cout-Zeile:

```
cout << “Hello, world\n”;
```



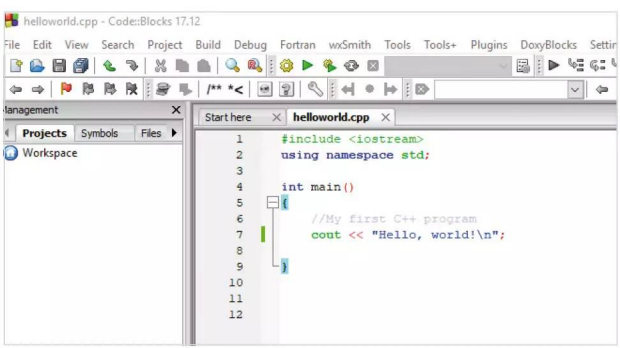
SCHRITT 6

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

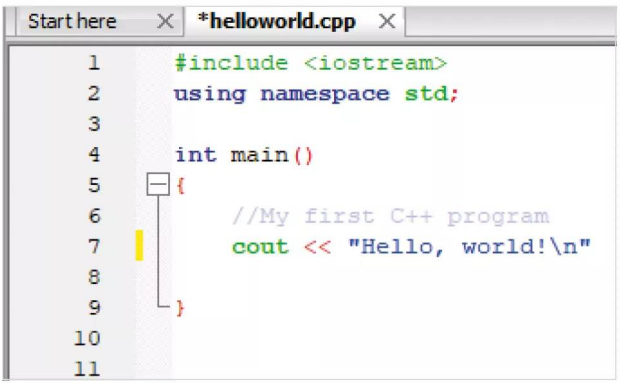
Wie wir zuvor bereits erwähnt haben, brauchen Sie std::cout nicht, wenn Sie namespace std; am Anfang des Codes stehen haben. Wir könnten einfach nur auf das Zahnrad/Wiedergabetaste-Symbol klicken, aber es lohnt sich, die verfügbaren Optionen durchzugehen. Wie Sie sehen, wurde die Datei durch Erstellen/Ausführen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



SCHRITT 8

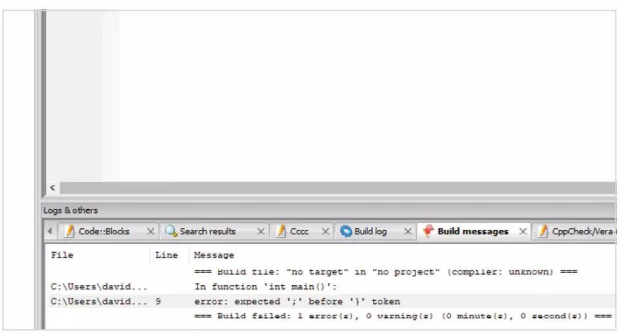
Erstellen Sie einen absichtlichen Fehler im Code, indem Sie das Semikolon aus der cout-Zeile entfernen:

```
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.

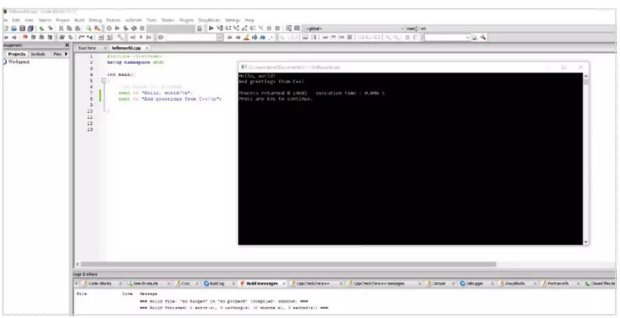


SCHRITT 10

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/Wiedergabesymbol, 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 Ingenieuren, 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
```

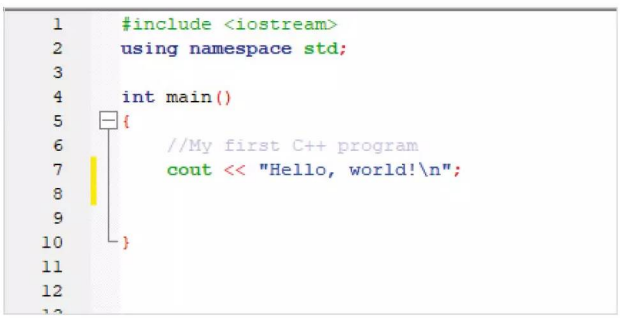
C++-KOMMENTARE

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 Codeabschnitte mit dem doppelten Schrägstrich kommentieren:

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

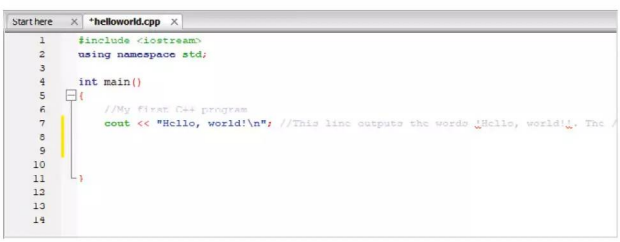


SCHRITT 2

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 */
```

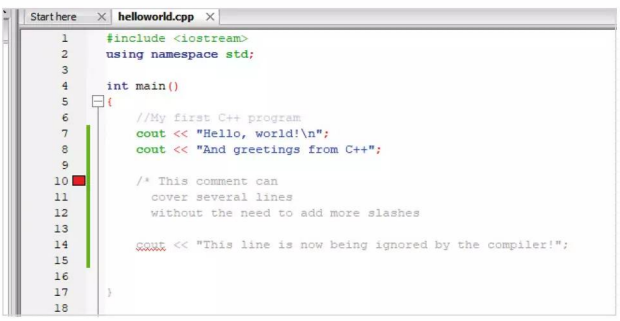
Vergessen Sie nur nicht, den Blockkommentar mit dem gegenüberliegenden Sternchen und dem Schrägstrich abzuschließen.

SCHRITT 4

Seien Sie vorsichtig beim Kommentieren, 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. Glücklicherweise hilft die Farbcodierung in Code::Blocks, Codekommentare zu identifizieren.



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.

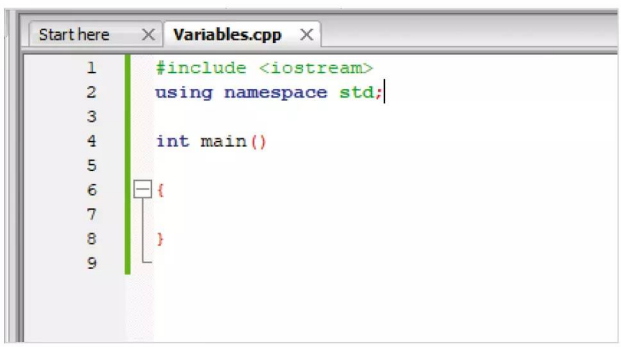
DIE DEKLARATION VON VARIABLEN

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:

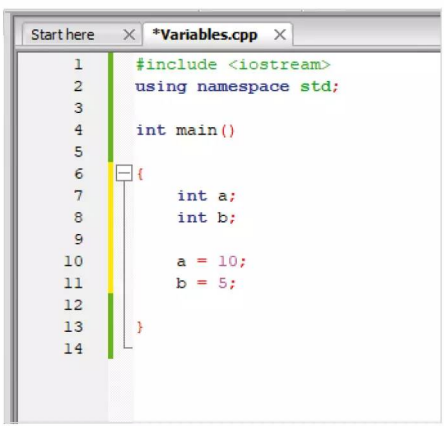
```
#include <iostream>
using namespace std;

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;
```

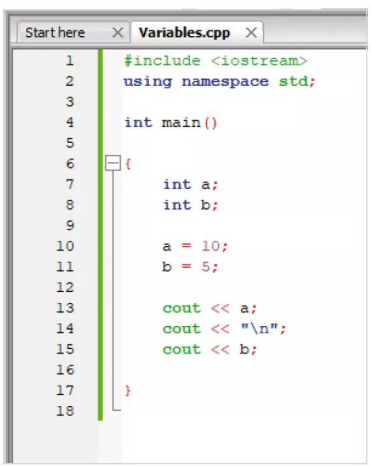


SCHRITT 3

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;
```

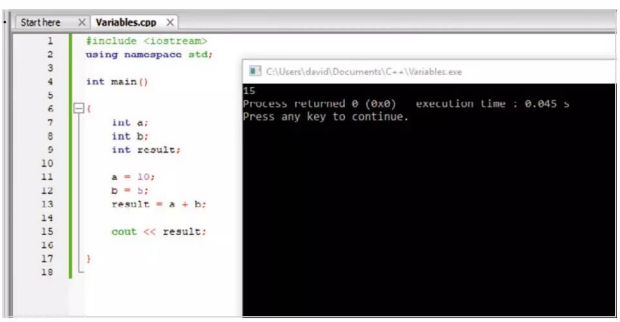
Der cout << "\ n";-Teil setzt lediglich zwischen der Ausgabe von 10 und 5 eine neue Zeile ein.



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:

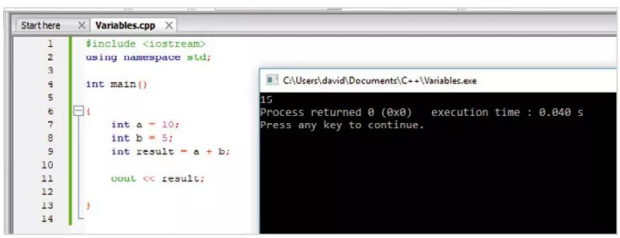
```
int result;

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;
```



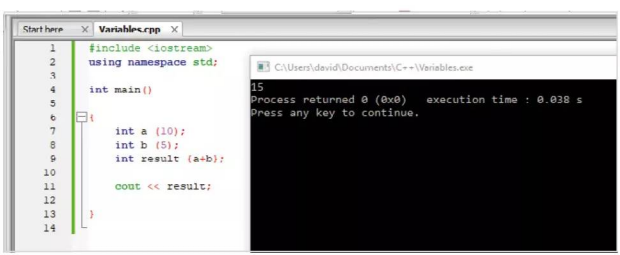
SCHRITT 6

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:

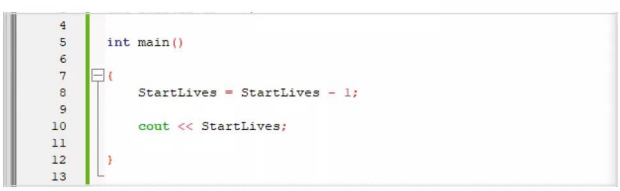
```
int result {a+b};
```



SCHRITT 7

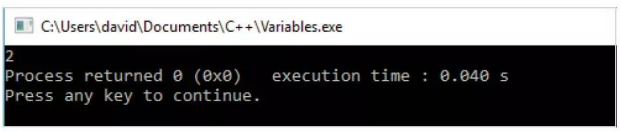
Sie können auch globale Variablen erstellen. Dies sind Variablen, die außerhalb einer Funktion deklariert und von jeder Funktion innerhalb des gesamten Codes verwendet werden können. Bisher haben Sie lokale Variablen verwendet, d. h. Variablen, die in der Funktion verwendet werden. Hier ein Beispiel:

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

int main ()

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



SCHRITT 8

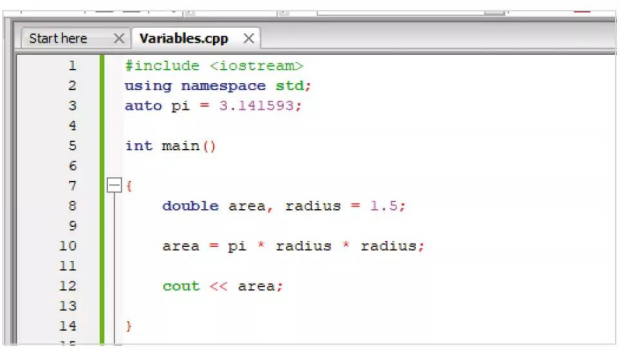
Im vorherigen Schritt wurde die globale Variable StartLives erstellt. In z. B. einem Spiel enthält oder verliert ein Spieler Leben, je nachdem, wie gut oder wie schlecht er spielt. Wenn der Spieler das Spiel neu startet, kehrt StartLives in den Standardzustand zurück: 3. Hier haben wir 3 Leben zugewiesen,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



SCHRITT 9

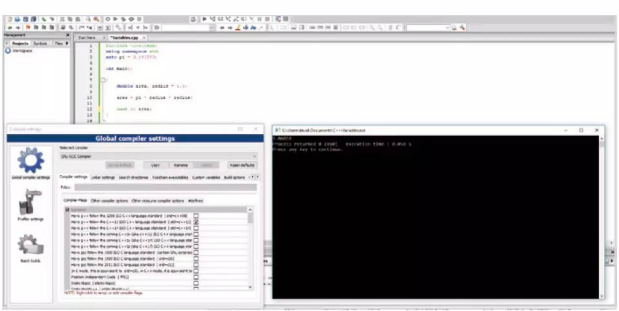
Der moderne C++-Compiler ist weitaus intelligenter als die meisten Programmierer es ihm zutrauen würden. Es gibt zahlreiche Datentypen, die Sie für Variablen deklarieren können, Sie können aber auch die auto-Funktion verwenden:

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

int main()

{
    double area, radius = 1.5;

    area = pi * radius * radius;

    cout << area;

}
```



SCHRITT 10

Auto wird jedoch nicht funktionieren, es sei denn, Sie gehen zu Settings > Compiler und aktivieren das Kontrollkästchen „Have G++ follow the C++11 ISO C++ Language Standard [-std=c++1]". 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.

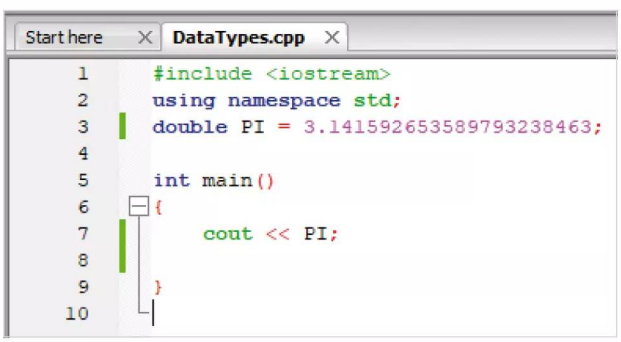
Wird nun ein Modifizierer angewendet, ändert sich der Bereich:

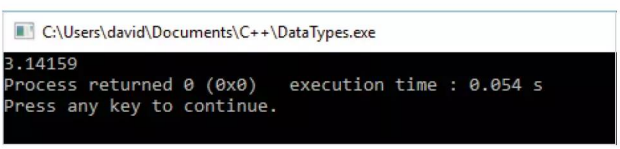
Unsigned int = 0 bis 4294967295
Signed int = -2147483648 bis 2147483647
Short int = -32768 bis 32767
Unsigned Short int = 0 bis 65,535
Signed Short int = -32768 bis 32767
Long int = -2147483647 bis 2147483647
Signed Long int = -2147483647 bis 2147483647
Unsigned Long int = 0 bis 4294967295

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;
}
```



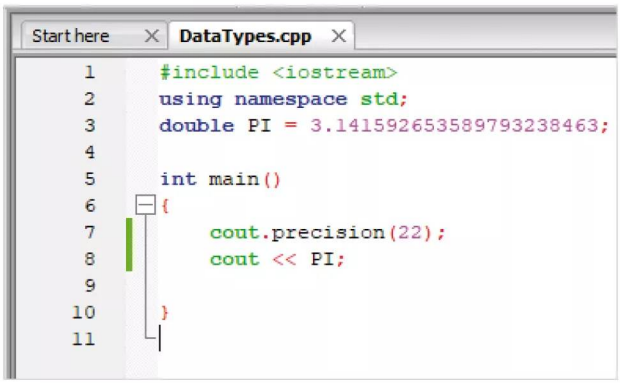


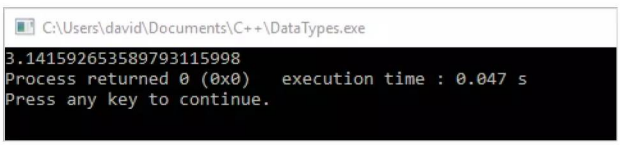
Wenn Sie den Code nun erstellen und ausführen, wird nur 3.14159 ausgegeben, was an den Einschränkungen des cout-Befehls liegt.

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;
}
```



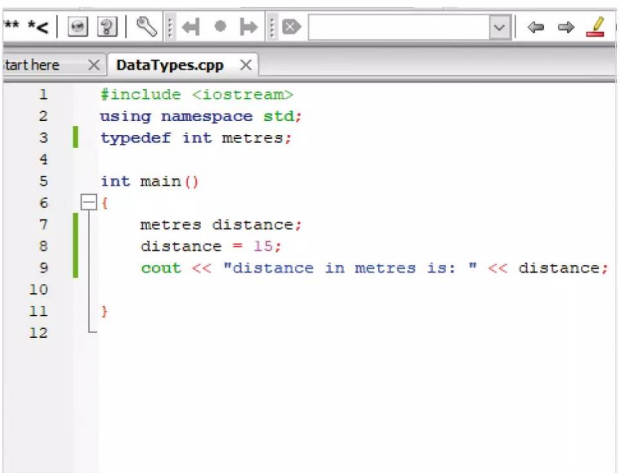


Erstellen und führen Sie den Code erneut aus. Wie Sie im Befehlszeilenfenster sehen, unterscheidet sich die durch die Pi-Variable ausgegebene Zahl von der Nummer, die Sie in C++ in der Variablen angegeben haben. Die Ausgabe gibt den Wert von Pi als 3.141592653589793115998 wieder, wobei die Zahlen ab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

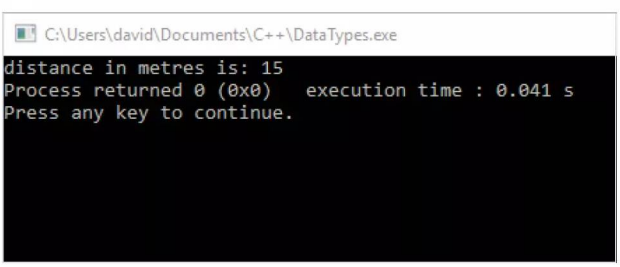
Dies liegt hauptsächlich an der Binär-Umrechnung im Compiler und dem IEEE 754-Standard mit doppelter Genauigkeit, der 64-Bit-Daten belegt, wovon 52 Bit für die Signifikanz (die signifikanten Stellen in einer Gleitkommazahl) vorgesehen sind und ca. 3,5 Bits von den Werten 0 bis 9 eingenommen werden. Wenn 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 die Ausgabe „distance in metres is 15“ anzeigt.

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.

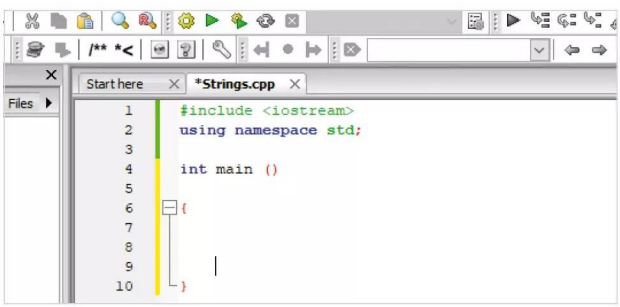
STRINGTHEORIE

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:

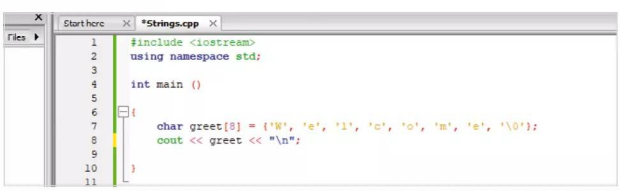
```
#include <iostream>
using namespace std;

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

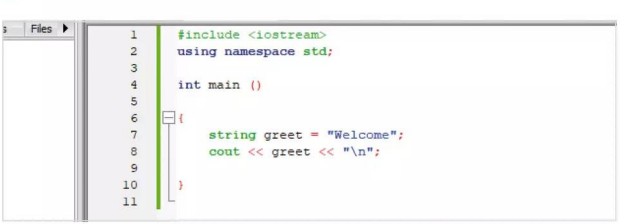
Erstellen und führen Sie den Code aus. Auf dem Bildschirm wird nun „Welcome" angezeigt. Dies ist jedoch kein String. Ein String ist eine Klasse, die Objekte definiert, die als Zeichenstrom dargestellt werden können und nicht wie ein Array abgeschlossen werden müssen. Der Code kann daher folgendermaßen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

}
```

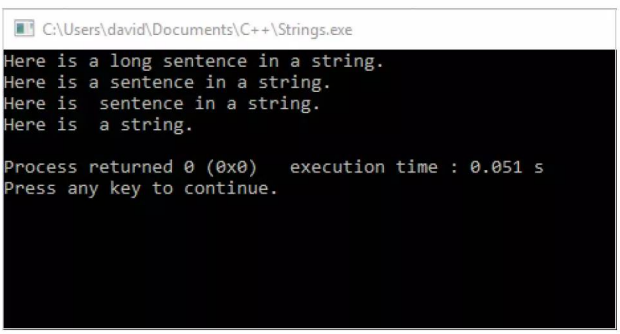
SCHRITT 9

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.

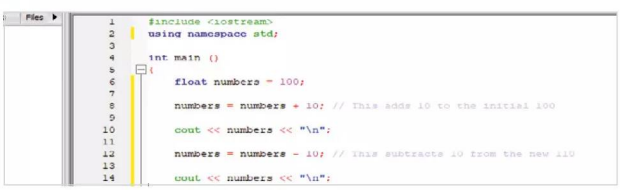
C++ = MC²

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

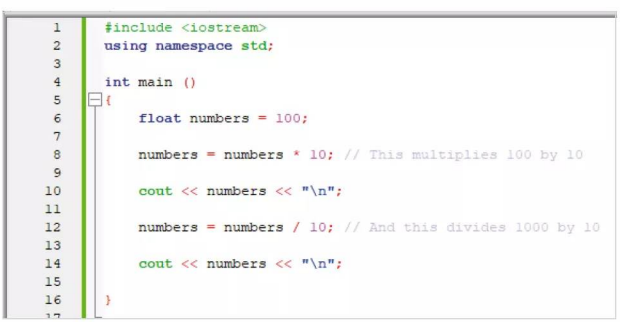
Beachten Sie, dass wir für die Zahlenvariable float verwendet haben. Sie können zwar auch Integer verwenden, müssen aber, wenn Sie plötzlich Dezimalzahlen benutzen sollten, je nach benötigter Genauigkeit zu float oder double wechseln. Führen Sie den Code aus, um die Ergebnisse anzuzeigen.

```
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

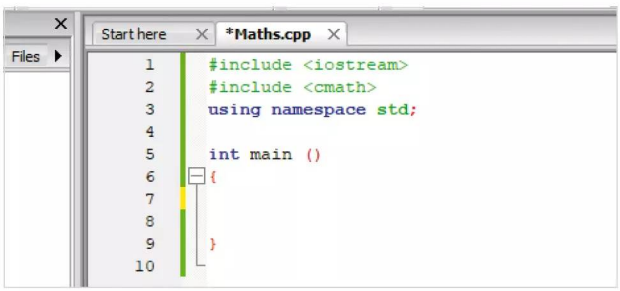
Führen Sie erneut den Code aus und schauen Sie sich die Ergebnisse an. Dies ist zwar nicht sehr spannend, ist aber ein Einstieg in die Mathematik in C++. Da wir einen float verwenden, können Sie mit dem Code experimentieren und mit Dezimalstellen multiplizieren, dividieren, addieren und subtrahieren.

```
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:

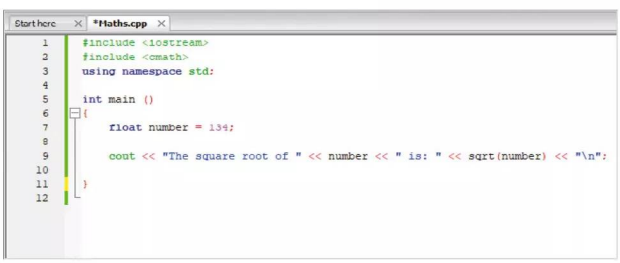
```
#include <iostream>
#include <cmath>
using namespace std;

int main ()
{

}
```



SCHRITT 6

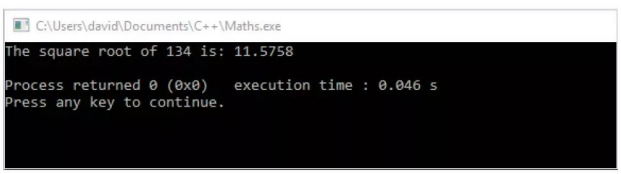
Beginnen Sie, indem Sie die Quadratwurzel einer Zahl ermitteln:

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

int main ()
{
    float number = 134;

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

}
```



SCHRITT 7

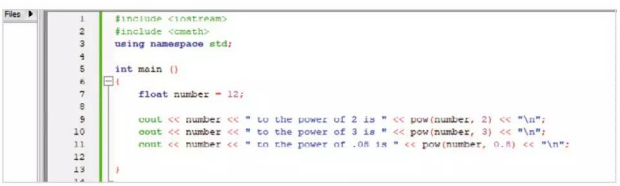
Hier haben wir einen neuen float namens number erstellt. Die Funktion sqrt(number) zeigt die Quadratwurzel von 134 an, den Wert der Variablen number. Erstellen und führen Sie den Code aus. Die Antwort wird 11.5758 sein.



SCHRITT 8

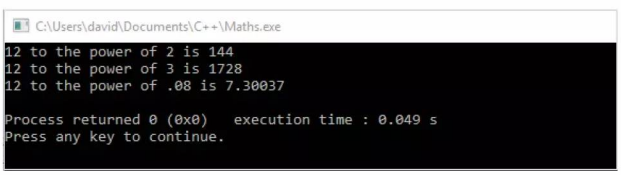
Potenzen können wie folgt berechnet werden:

```
#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

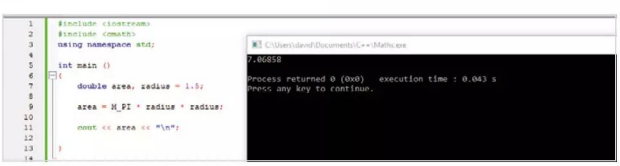
Hier haben wir einen float namens number mit dem Wert 12 erstellt. Die Berechnung erfolgt über pow (variable, power). Natürlich können Sie Potenzen und Quadratwurzeln auch ohne Variablen berechnen. Zum Beispiel gibt pow (12, 2) den gleichen Wert aus wie die erste cout-Zeile im Code.



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”;

}
```



Benutzerinteraktion

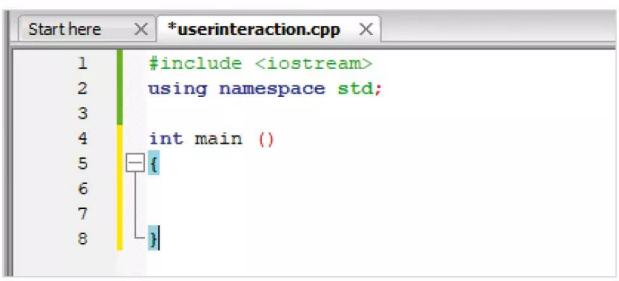
Es gibt nichts Besseres als ein Programm zu erstellen, das auf den Benutzer reagiert. Die Benutzerinteraktion ist einer der am meisten gelehrten Aspekte und man kann damit weitaus mehr machen, als nur den Benutzer mit seinem Namen zu begrüßen.

HELLO, DAVE

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

SCHRITT 1

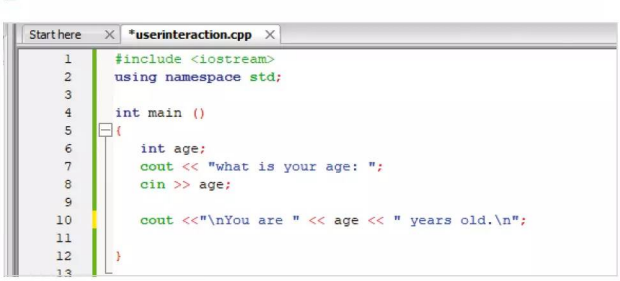
Alles, was der Benutzer in das Programm eingeben soll, muss irgendwo im Systemspeicher abgelegt werden, damit es abgerufen und verwendet werden kann. Daher muss jede Eingabe zuerst als Variable deklariert werden, damit sie auch vom Benutzer verwendet werden kann. Erstellen Sie zunächst eine leere C++-Datei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



SCHRITT 2

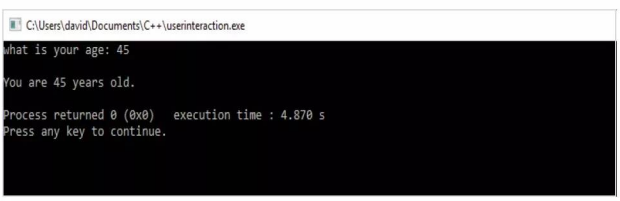
Der Datentyp der Variablen muss der Art der Eingabe entsprechen, die Sie vom Benutzer wünschen. Um z. B. einen Benutzer nach dem Alter zu fragen, würden Sie einen Integer verwenden:

```
#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

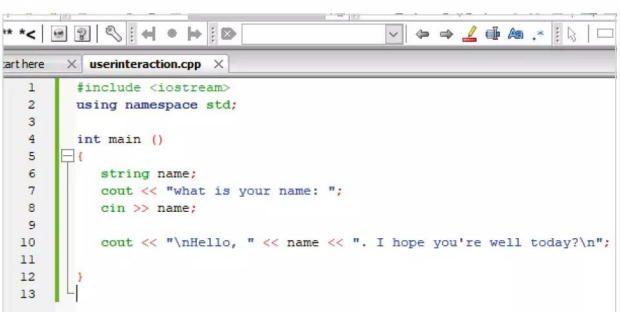
Der cin-Befehl funktioniert auf entgegengesetzte Weise zum cout-Befehl. Mit der ersten cout-Zeile geben Sie, wie durch die Pfeile angezeigt, „What is your age" an den Bildschirm aus. Der cin-Befehl verwendet gegenüberliegende Pfeile, um eine Eingabe anzuzeigen. Die Eingabe kommt in den Integer age und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

}
```



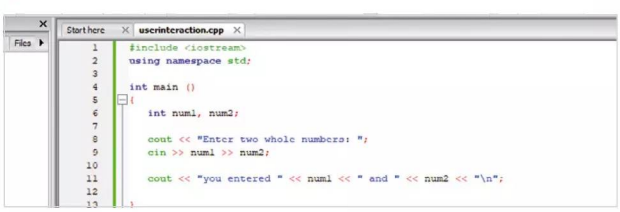
SCHRITT 5

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:

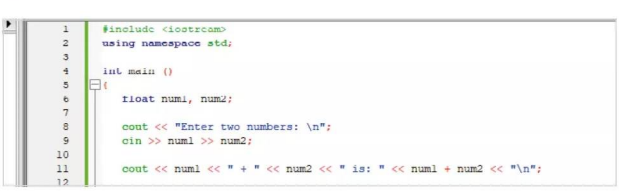
```
#include <iostream>
using namespace std;

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:

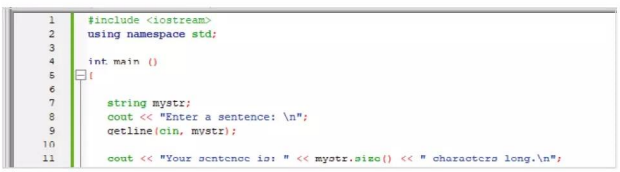
```
#include <iostream>
using namespace std;

int main ()
{
   float num1, num2;

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

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

}
```



SCHRITT 8

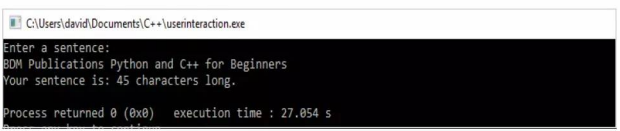
Cin eignet sich gut für die meisten Eingabeaufgaben, hat aber die Einschränkung, dass Leerzeichen immer als Abschlusszeichen gesehen werden, daher ist cin nur für einzelne und nicht mehrere Wörter vorgesehen. Die getline-Funktion greift jedoch cin als erstes Argument und die Variable als zweites auf:

```
#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

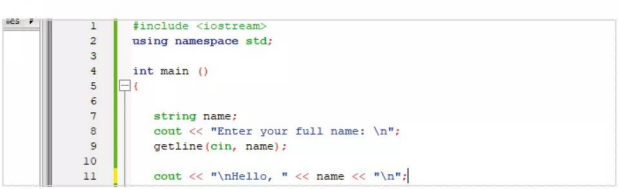
Erstellen und führen Sie den Code aus, und geben Sie dann einen Satz mit Leerzeichen ein. Der Code wird die Anzahl der Zeichen lesen. Wenn Sie die getline-Zeile entfernen und durch cin >> mystr ersetzen und es erneut versuchen, zeigt das Ergebnis die Anzahl der Zeichen bis zum ersten Leerzeichen an.



SCHRITT 10

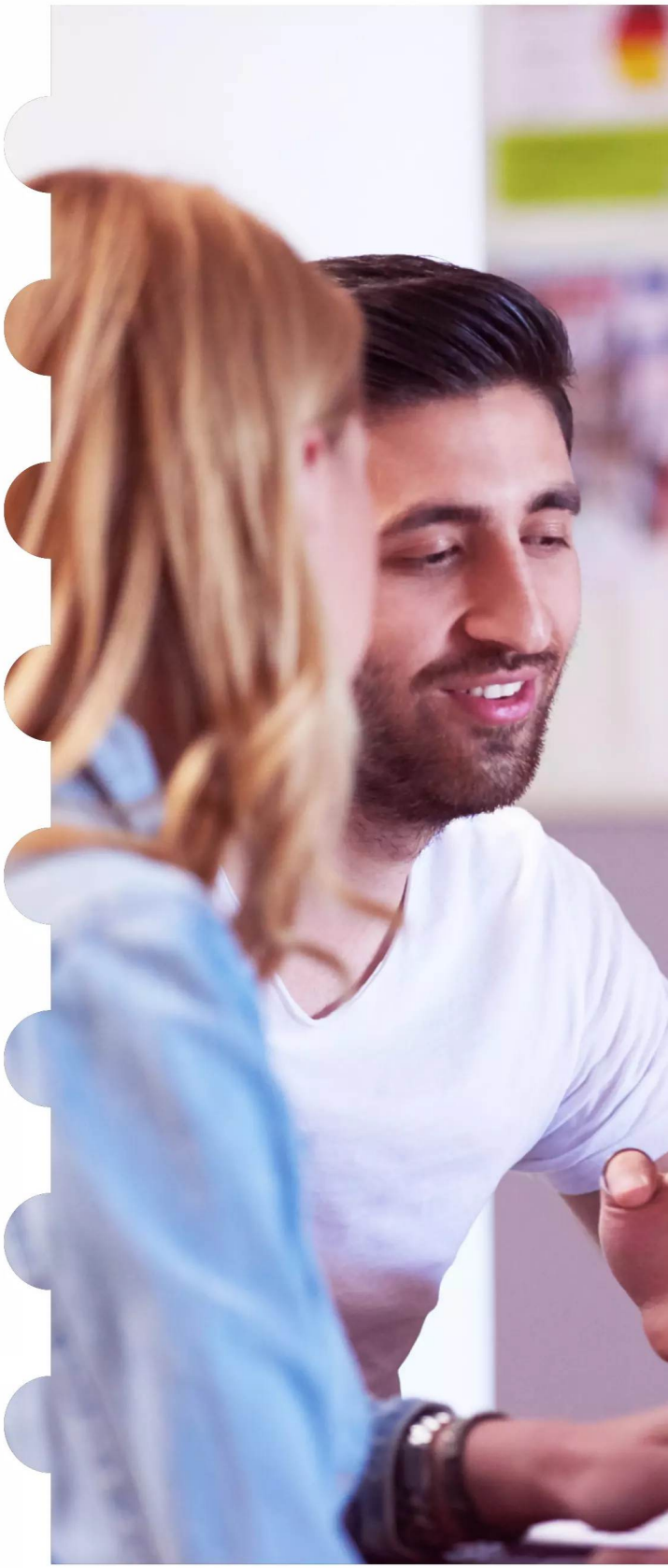
Getline ist gewöhnlich ein Befehl, den neue C++-Programmierer vergessen, einzufügen. Der abschließende Leerraum ist ärgerlich, wenn Sie einfach nicht herausfinden, warum Ihr Code nicht funktioniert. Es ist daher am besten, in Zukunft getline(cin, variable) zu verwenden:

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

int main ()
{

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

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

}
```



Wussten Sie, dass Windows über eine eigene Skriptsprache verfügt? Batch-Dateien gibt es bereits seit den Anfängen von Windows, und obwohl sie von der modernen grafischen Benutzeroberfläche von Windows überschattet werden, sind sie weiterhin vorhanden und genauso leistungsfähig wie vor dreißig Jahren.

Das Programmieren mit Batch-Dateien wird von Systemadministratoren weiterhin genutzt. Es lohnt sich daher, ein wenig Zeit damit zu verbringen und zu lernen, wie sie funktionieren und was sie können. In diesem Abschnitt stellen wir Ihnen Batch-Dateien vor und befassen uns mit Benutzerinteraktionen, Variablen und Schleifen. Damit der Spaß nicht zu kurz kommt, haben wir auch ein Batch-Datei-Quiz dabei.



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

– Bill Gates (Mitbegründer von Microsoft)

Programmieren mit Windows 10-Batch-Dateien

Was ist eine Batch-Datei?

Die Windows-Batch-Datei gibt es seit den Anfängen von DOS und war einst ein wichtiger Faktor beim Hochfahren in ein funktionierendes System. Mit einer Batch-Datei lässt sich vieles machen, aber zunächst schauen wir uns an, was sie eigentlich ist.

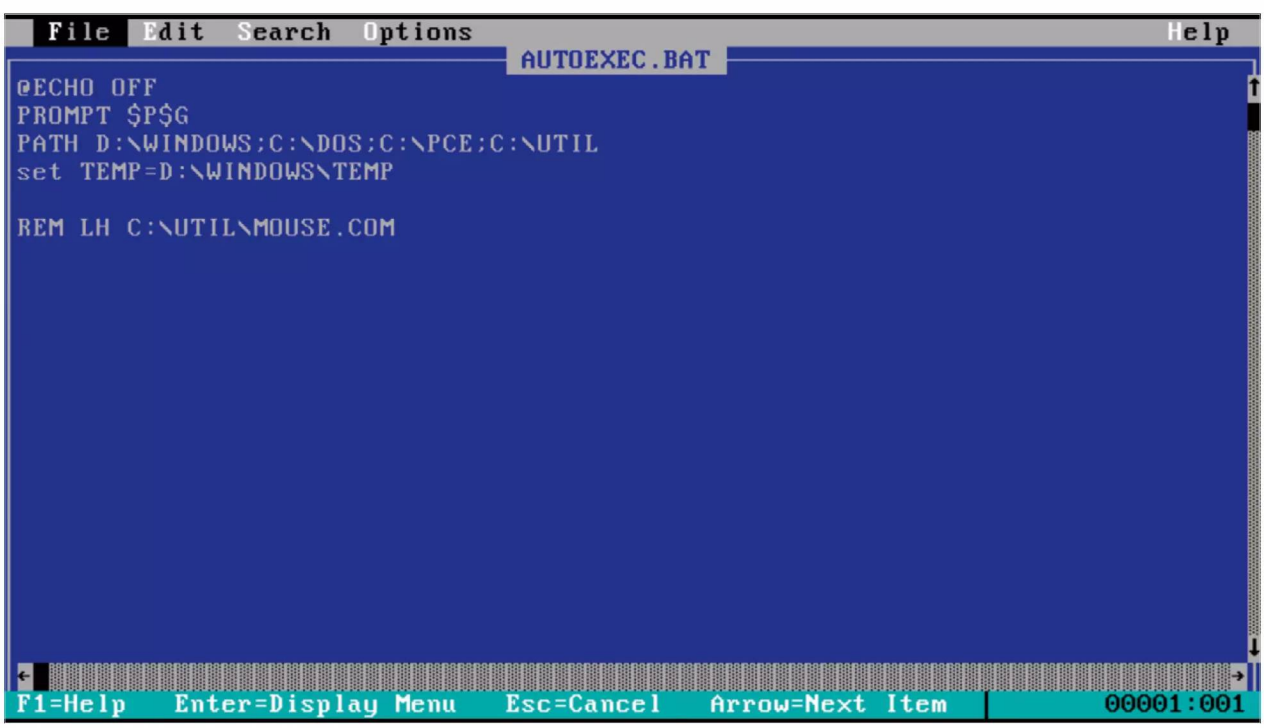
.BAT MAN

Eine Windows-Batch-Datei ist einfach eine Skriptdatei, in der eine Reihe von Befehlen Zeile für Zeile ausgeführt wird, ähnlich wie bei einem Linux-Skript. Die Befehlsfolge wird vom Befehlszeileninterpreter ausgeführt und in einer Textdatei mit der Erweiterung .BAT gespeichert. Dadurch weiß Windows, dass es sich um eine ausführbare Datei handelt, in diesem Fall ein Skript.

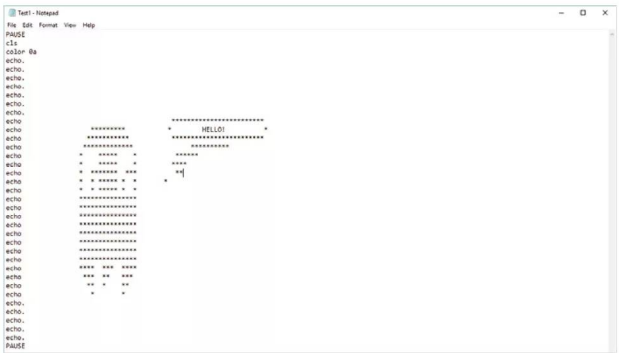
Batch-Dateien gibt es seit den ersten Microsoft DOS-Versionen. Obwohl es sich nicht ausschließlich um eine Microsoft-Skriptdatei handelt, werden Batch-Dateien hauptsächlich mit Microsofts Betriebssystemen in Zusammenhang gebracht. In den Anfangstagen, als ein PC noch mit einer DOS-Version startete (die beim Einschalten eine einfache Eingabeaufforderung erzeugte), wurde die Batch-Datei in Form einer Systemdatei namens Autoexec.bat verwendet. Autoexec.bat war ein Skript, das automatisch Befehle ausführte, sobald das Betriebssystem die Datei Config.sys ausgeführt hatte.

Wenn ein Benutzer seinen auf DOS-basierenden Computer einschaltete und das BIOS den Systemspeicher usw. überprüft hatte, suchte DOS nach der Datei Config.sys, um bestimmte Anzeigeanforderungen und Hardwaretreiber zu laden und ihnen einen Platz im Computer zuzuweisen, alle verfügbaren Speicher-Manager zuzuteilen und dem System mitzuteilen, wo sich die Command.com-Datei befindet, dem Befehlszeileninterpreter für DOS. Anschließend ging die Datei Autoexec.bat die einzelnen Zeilen durch. Dabei wurden Programme geladen, die die Maus oder das optische Laufwerk in den von der Datei Config.sys zugewiesenen Speicherbereichen aktivierten.

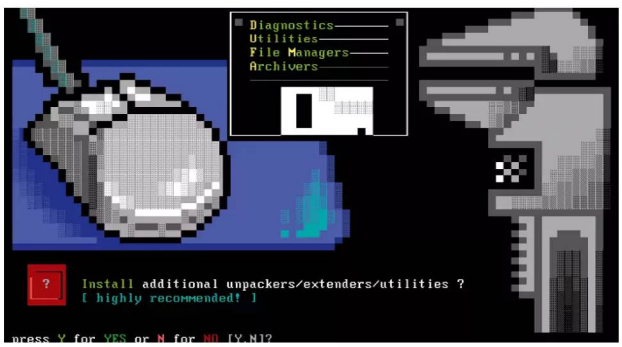
Der DOS-Benutzer von damals konnte je nach Bedarf unterschiedliche Autoexec.bat-Dateien erstellen. Wenn er z. B. für ein Spiel so viel Speicher wie möglich benötigte, erstellte er eine Gruppe von Config.sys- und Autoexec.bat-Dateien, die nur ein Minimum an Treibern usw. laden würden. Wurde Zugriff auf das Netzwerk benötigt, konnte eine



PC-Benutzer machten mit der Autoexec.bat-Datei ihre ersten Erfahrungen mit Batch-Dateien.



Batch-Dateien sind reiner Text und werden häufig im Editor erstellt.



Batch-Dateien wurden häufig als Hilfsprogramme verwendet, um Benutzern bei komplexen Aufgaben zu unterstützen.

Autoexec.bat-Datei erstellt werden, mit der der Netzwerkkartentreiber geladen wurde und automatisch auf das Netzwerk zugegriffen werden konnte. Jedes dieser einmaligen Set-ups wurde auf eine Diskette geladen und bei Bedarf vom Benutzer hochgefahren.

Die Autoexec.bat war die erste derartige Datei, auf die viele PC-Benutzer gestoßen sind, da ihre Computer 16 Bit oder sogar 8 Bit hatten; wir sprechen hier schließlich vom Ende der achtziger und Anfang der neunziger Jahre. Die Batch-Datei war das Hauptwerkzeug des Benutzers für die Automatisierung von Aufgaben, das Erstellen von Verknüpfungen und Abenteuerspielen und die Umsetzung komplexer Prozesse in einfachere Vorgänge.

Heutzutage ist eine Batch-Datei jedoch nicht mehr nur zum Laden von Treibern oder Booten des PCs gedacht. Sie können eine Batch-Datei genau wie jede andere Skriptsprachendatei nutzen, indem Sie sie so programmieren, dass sie zur Eingabe von Benutzereingaben auffordert und die Ergebnisse auf dem Bildschirm anzeigt. Sie können eine Datei auch speichern und sogar an einen lokalen oder über ein Netzwerk angeschlossenen Drucker senden. Sie können Skripte erstellen, um Ihre Dateien an verschiedenen Orten zu sichern, Datumsstempel vergleichen, um nur die zuletzt geänderten Inhalte zu sichern sowie das Skript so programmieren, dass dies alles automatisch geschieht. Batch-Dateien sind sehr leistungsfähig. Sie werden zwar nicht mehr so häufig verwendet wie zu alten DOS-Zeiten, sind aber weiterhin vorhanden und funktionieren sogar in der neuesten Windows 10-Version.

Zum Programmieren mit Batch-Dateien in Windows ist lediglich Windows (Version 10 oder älter) erforderlich. Sie müssen nur den Editor öffnen und zur Windows-Eingabeaufforderung gehen. Um zu erfahren, wie das Ganze funktioniert, lesen Sie weiter.

DIE MACHT DER BATCH-DATEI

Genau wie jede andere Programmierschnittstelle, die das System direkt abfragen und bearbeiten kann, erfordern Batch-Dateien bei der Programmierung eine gewisse Sorgfalt. Es ist schwer, das System mit einer Batch-Datei zu beschädigen, da die wichtigsten Elemente des modernen Windows-Systems über die Benutzerkontensteuerung geschützt werden. Diese erlaubt den Zugriff auf wichtige Systemdateien nur mit erhöhten Berechtigungen. Wenn Sie also eine Batch-Datei erstellen, die eine Systemdatei löscht, wird die Benutzerkontensteuerung aktiviert und der Prozess angehalten.

Wenn Sie in der Eingabeaufforderung jedoch als Administrator mit erhöhten Rechten arbeiten, wird die Benutzerkontensteuerung die Batch-Datei nicht infrage stellen und unabhängig von gelöschten Dateien weiterarbeiten.

Es ist unwahrscheinlich, dass jemand eine Batch-Datei erstellt, die sein Betriebssystem absichtlich löscht. Um dies zu verhindern, gibt es Systemkontrollen. Es sei aber erwähnt, dass im Internet Batch-Dateien mit schädlichem Code erhältlich sind. Ähnlich wie bei einem Virus kann eine bösartige Batch-Datei (wenn sie mit Administratorrechten ausgeführt wird) Systemschäden verursachen. Führen Sie daher als Administrator keine aus dem Internet heruntergeladene Batch-Datei aus, ohne vorher deren Funktionsweise zu überprüfen.

Auf den nächsten Seiten erfahren Sie mehr über Batch-Dateien, machen Sie sich daher nicht allzu viele Gedanken über die Zerstörung Ihres Systems. All dies zeigt nur, wie leistungsfähig die einfache Batch-Datei sein kann.

Sie können komplexe oder einfache Batch-Dateien erstellen, die ASCII-Bilder auf dem Bildschirm anzeigen.

Erste Schritte mit Batch-Dateien

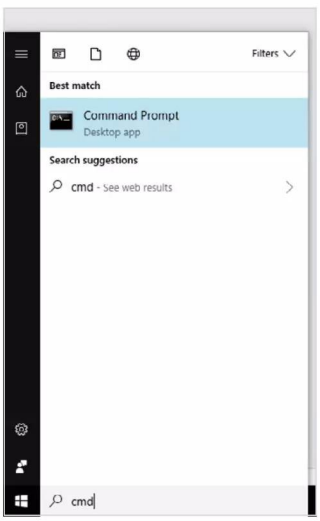
Bevor Sie mit dem Programmieren von Batch-Dateien beginnen, gibt es ein paar Dinge, die Sie wissen sollten. Eine Batch-Datei kann nur ausgeführt werden, wenn sie die .bat-Erweiterung hat. Ihre Bearbeitung im Editor ist allerdings nicht immer einfach.

NEUE BATCH-DATEI

In diesem Abschnitt über Batch-Dateien arbeiten wir mit dem Editor (engl. Originalname: Notepad), der Eingabeaufforderung und in einem Ordner namens „Batch Files". Zunächst zeigen wir Ihnen, wie Sie zur Windows-Eingabeaufforderung gelangen.

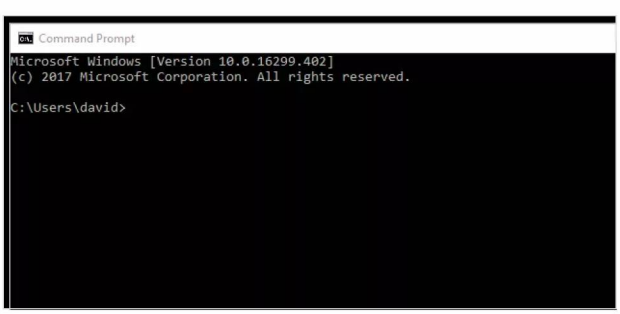
SCHRITT 1

Die Windows-Eingabeaufforderung mag auf den Anfänger etwas abschreckend wirken, aber sie ist lediglich eine andere Schnittstelle (bzw. Shell), über die auf das Dateisystem zugegriffen wird. Sie können in der Eingabeaufforderung an eine beliebige Stelle gehen, genau wie bei der grafischen Schnittstelle. Klicken Sie zunächst auf das Windows-Menüzeichen und geben Sie im Suchfeld „CMD" oder „Eingabeaufforderung" ein.



SCHRITT 2

Klicken Sie auf das Suchergebnis namens „Eingabeaufforderung" (Desktop-App). Ein neues Fenster wird geöffnet. Das Eingabeaufforderungsfenster sieht nicht sehr spektakulär aus, zeigt aber die MS-Windows-Versionsnummer und Copyright-Informationen an, gefolgt von der eigentlichen Eingabeaufforderung. Diese zeigt das aktuelle Verzeichnis bzw. den aktuellen Ordner sowie den Benutzernamen an.



SCHRITT 3

Geben Sie im Eingabeaufforderungsfenster `dir/w` ein, um alle aktuellen Dateien und Verzeichnisse aufzulisten. In diesem Fall ist es Ihr Home-Verzeichnis, das Windows jedem angemeldeten Benutzer zuweist. Sie können mit dem Befehl cd (change directory) navigieren. Probieren Sie Folgendes aus:

```
cd Documents
```

Drücken Sie anschließend die Eingabetaste.

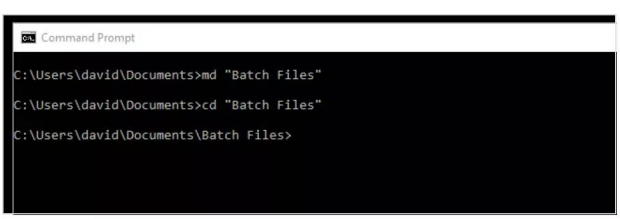
SCHRITT 4

Sie sollten nun `\Documents>` sehen; dies bedeutet, dass Sie sich im Documents-Verzeichnis befinden. Erstellen Sie nun ein neues Verzeichnis namens Batch Files:

```
md "Batch Files"
```

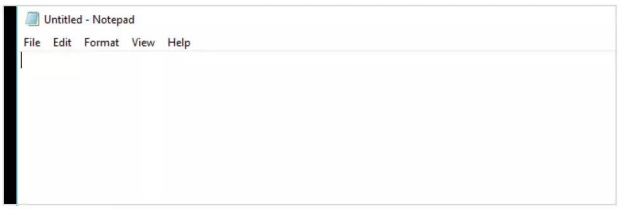
Die Anführungszeichen sind erforderlich, da Windows ansonsten zwei Verzeichnisse erstellt: Batch und Files. Wechseln Sie nun zum neuen Batch Files-Verzeichnis (zum Wechseln des Verzeichnisses werden keine Anführungszeichen benötigt):

```
cd Batch Files
```



SCHRITT 5

Nachdem Sie das Verzeichnis zum Speichern Ihrer Batch-Dateien eingerichtet haben, zeigen wir Ihnen, wie Sie diese erstellen. Lassen Sie das Eingabeaufforderungsfenster geöffnet und klicken Sie erneut auf das Windows-Menüzeichen. Geben Sie diesmal Editor oder Notepad ein und klicken Sie auf das Suchergebnis, um den Editor zu öffnen. Er ist ein einfacher Texteditor, der sich aber ideal zum Erstellen von Batch-Skripten eignet.



SCHRITT 6

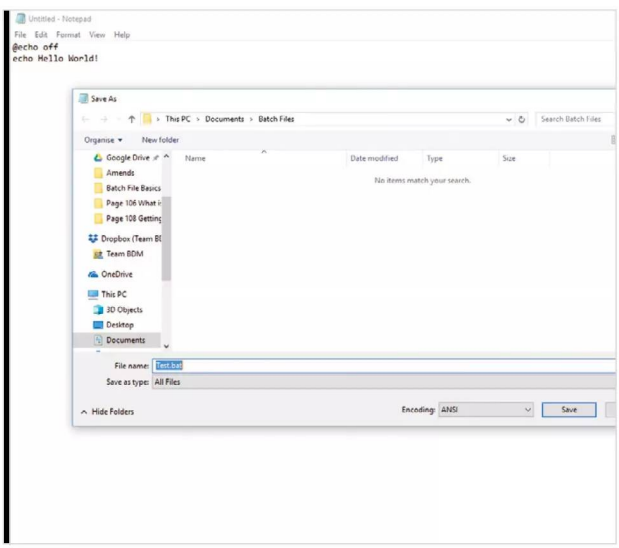
Zum Erstellen Ihrer ersten Batch-Datei geben Sie im Editor Folgendes ein:

```
@echo off
echo Hello World!
```

Eine Batch-Datei zeigt standardmäßig alle Befehle an, die Zeile für Zeile durchlaufen werden. Der Befehl @echo off bewirkt, dass diese Funktion für das gesamte Skript deaktiviert wird; das @-Zeichen bezweckt, dass dieser Befehl auf sich selbst anzuwenden ist

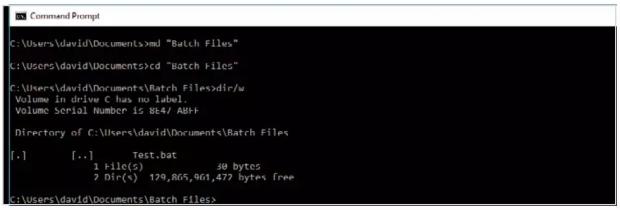
SCHRITT 7

Beim Speichern im Editor lautet die Standarderweiterung .txt, um eine Textdatei zu kennzeichnen. Die Erweiterung soll jedoch .bat sein. Klicken Sie auf Datei > Speichern unter und navigieren Sie zum neu erstellten Batch Files-Verzeichnis im Dokumente-Ordner. Wählen Sie im Drop-down-Menü neben Dateityp „Alle Dateien". Geben Sie als Dateiname **Test.bat** ein.



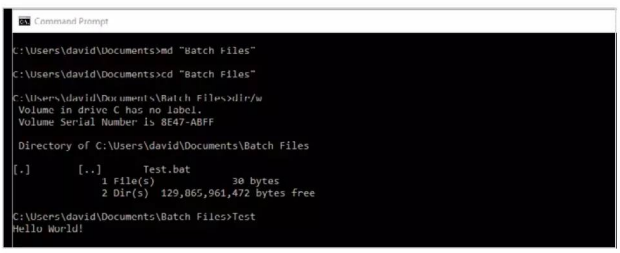
SCHRITT 8

Geben Sie in der Eingabeaufforderung erneut `dir/w` ein, um die neu erstellte Test.bat-Datei aufzulisten. Der /w-Teil von dir/w bedeutet, dass die Dateien quer über den Bildschirm und nicht nach unten angezeigt werden. Wenn Sie das bevorzugen, geben Sie nur `dir` ein (Sie brauchen jedoch mehr Dateien, um dies zu verdeutlichen). Die w-Ergänzung gilt jedoch als lesbarer.



SCHRITT 9

Um die soeben erstellte Batch-Datei auszuführen, geben Sie einfach deren Namen, also Test, in das Eingabeaufforderungsfenster ein. Sie müssen den .bat-Teil nicht hinzufügen, da Windows ihn als ausführbare Datei erkennt und dies die einzige Datei mit diesem Namen im aktuellen Verzeichnis ist. Drücken Sie die Eingabetaste und sehen Sie, wie Sie in der Eingabeaufforderung mit Hello World! begrüßt werden.

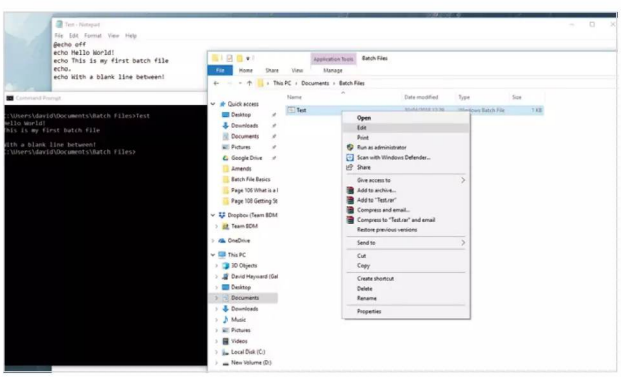


SCHRITT 10

Der echo-Befehl zeigt an, was auf dem Bildschirm erscheint. Machen Sie im Windows Explorer auf der Test.bat-Datei einen Rechtsklick und wählen Sie „Bearbeiten" aus, um weitere echo-Befehle hinzuzufügen. Probieren Sie Folgendes aus:

```
@echo off
echo Hello World!
echo This is my first batch file
echo.
echo With a blank line between!
```

Denken Sie daran, jede neue Änderung in der Batch-Datei zu speichern.



Ausgaben in Batch-Dateien

Es ist zwar toll, wenn das Eingabeaufforderungsfenster anzeigt, was nach dem echo-Befehl in die Batch-Datei geschrieben wird, wir wollen aber etwas nützlichere und interaktive Ausgaben und schalten daher einen Gang höher.

EINGABE DER AUSGABE

Batch-Dateien sind in der Lage, einen normalen Windows-Befehl auszuführen, und können zusätzliche Optionen und Flags hinzufügen.

SCHRITT 1

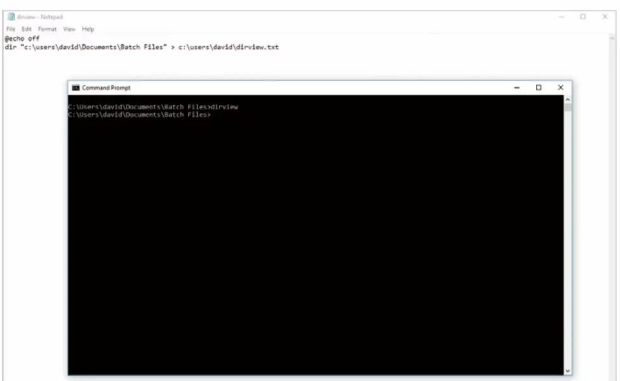
Wir gehen das Ganze einfach an und erstellen eine neue Batch-Datei namens dirview.bat, kurz für Directory View. Beginnen Sie mit dem Befehl @echo off und fügen Sie darunter Folgendes hinzu:

```
dir “c:\users\IHRNAME\Documents\Batch Files” > c:\users\IHRNAME\dirview.txt
```

Ersetzen Sie IHRNAME mit Ihrem Windows-Benutzernamen.

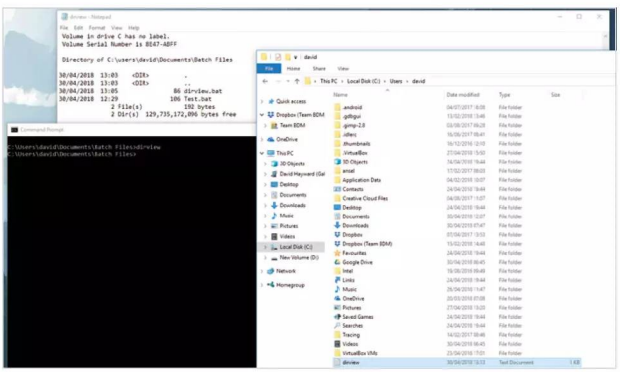
SCHRITT 2

Die neue Zeile listet mit dem dir-Befehl den Inhalt des Batch Files-Verzeichnisses in Ihrem Home-Verzeichnis auf und legt die Ausgabe in einer Textdatei namens `dirview.txt` im Stammverzeichnis Ihres Home-Verzeichnisses ab. Da sich alles in Ihrem Home-Bereich befindet, verhindert dies, dass die Windows-Benutzerkontensteuerung erhöhte Berechtigungen erfordert. Speichern Sie die Batch-Datei und führen Sie sie aus.



SCHRITT 3

Sie haben zweifellos festgestellt, dass es keinen Hinweis darauf gibt, dass die Batch-Datei funktioniert, da auf dem Bildschirm keine bedeutsame Ausgabe erscheint. Wenn Sie aber im Explorer zu `c:\USER\IHRNAME` gehen (`IHRNAME` wieder mit Ihrem Windows-Benutzernamen ersetzen) und `dir view.txt` doppelt anklicken, sehen Sie die Ausgabe der Batch-Datei.



SCHRITT 4

Wenn Sie das Öffnen der Textdatei mit der Ausgabe automatisieren möchten, fügen Sie der Batch-Datei die folgende Zeile hinzu:

```
notepad.exe c:\users\IHRNAME\dirview.txt
```

Speichern Sie die Datei, und führen Sie sie über die Eingabeaufforderung erneut aus. Diesmal wird die Ausgabe erstellt und der Editor mit dem Ausgabeinhalt automatisch geöffnet.

AUSGABE MIT VARIABLEN

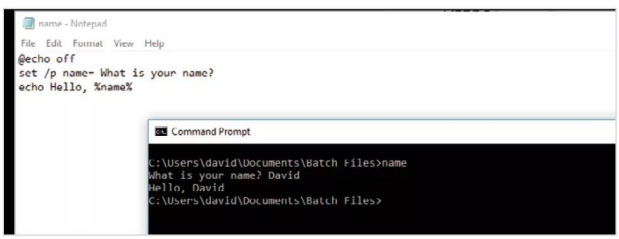
Variablen bieten eine interessantere Möglichkeit, um etwas auf dem Bildschirm auszugeben und für mehr Interaktion zwischen dem Benutzer und der Batch-Datei zu sorgen. Probieren Sie das folgende Beispiel aus.

SCHRITT 1

Erstellen Sie eine neue Batch-Datei namens name.bat. Beginnen Sie mit dem Befehl `@echo off` und fügen Sie dann die folgenden Zeilen hinzu:

```
set /p name= What is your name?
echo Hello, %name%
```

Hinter dem Fragezeichen ist ein Leerzeichen, damit es auf dem Bildschirm besser aussieht. Speichern und führen Sie die Batch-Datei aus.

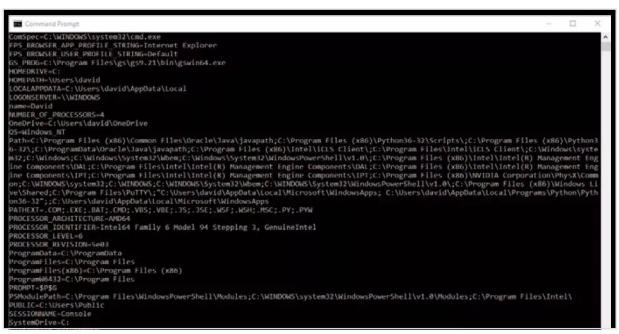


SCHRITT 2

set **/p** name erstellt eine Variable namens name, wobei **/p** angibt, dass ein **=Eingabeaufforderung-String** folgt. Mit dem set-Befehl werden System- und Umgebungsvariablen angezeigt, festgelegt oder entfernt. Geben Sie z. B. im Eingabeaufforderungsfenster Folgendes ein, um die aktuellen Systemvariablen anzuzeigen:

```
set
```

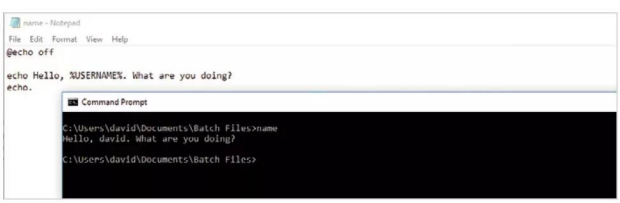
Beachten Sie die Variable name=, die wir gerade erstellt haben.



SCHRITT 3

Mit set gespeicherte Variablen können Sie mit der **%VARIABLENNAME%**-Syntax aufrufen. In der Batch-Datei haben wir den Inhalt der name-Variable mit der neu erstellten %name%-Syntax aufgerufen. Ihr Benutzername wird z. B. als Variable gespeichert. Probieren Sie dies in einer Batch-Datei aus:

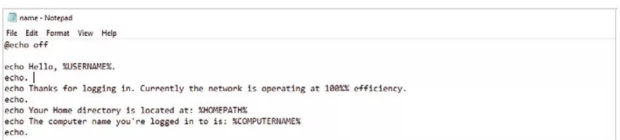
```
echo Hello, %USERNAME%. What are you doing?
```



SCHRITT 4

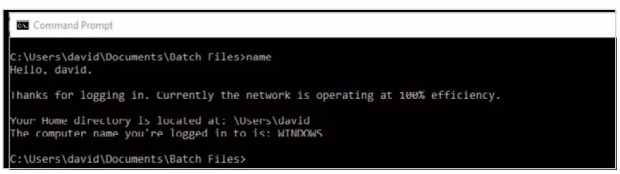
Dies ist äußerst nützlich, wenn Sie eine persönliche Batch-Datei erstellen möchten, die automatisch ausgeführt wird, wenn sich ein Benutzer bei Windows anmeldet. Mit den von Windows selbst erstellten Standardsystemvariablen können Sie eine Batch-Datei erstellen, die jeden Benutzer begrüßt:

```
@echo off

echo Hello, %USERNAME%.
echo.
echo Thanks for logging in. Currently the network is operating at 100%% efficiency.
echo.
echo Your Home directory is located at: %HOMEPATH%
echo The computer name you're logged in to is: %COMPUTERNAME%
echo.
```



SCHRITT 5

Speichern und führen Sie die Änderungen der Batch-Datei aus. Sie können name.bat überschreiben und weiterhin verwenden, wenn Sie möchten. Die Batch-Datei verwendet die aktuellen Systemvariablen und zeigt sie entsprechend dem Anmeldenamen des Benutzers und dem Computernamen an. Hinweis: Das doppelte Prozentzeichen bedeutet, dass das Prozentzeichen angezeigt wird; es ist keine Variable.



SCHRITT 6

Sie können auch die Batch-Datei ausführen und auf dem Benutzer-Desktop als Textdatei anzeigen:

```
@echo off

echo Hello, %USERNAME%. > c:%HOMEPATH%\user.txt
echo. >> c:%HOMEPATH%\user.txt
echo Thanks for logging in. Currently the network is operating at 100%% efficiency. >> c:%HOMEPATH%\user.txt
echo. >> c:%HOMEPATH%\user.txt
echo Your Home directory is located at: %HOMEPATH% >> c:%HOMEPATH%\user.txt
echo The computer name you're logged in to is: %COMPUTERNAME% >> c:%HOMEPATH%\user.txt
echo. >> c:%HOMEPATH%\user.txt

notepad c:%HOMEPATH%\user.txt
```

Das > gibt in eine neue Datei namens user.txt aus, während >> die Zeilen innerhalb der Datei hinzufügt.

Mit Variablen spielen

Mit Ihren eigenen sowie auch den System- und Umgebungsvariablen lässt sich viel erreichen. Die Kombination der Letzteren kann zu einer leistungsstarken und äußerst nützlichen Batch-Datei führen, insbesondere in Verbindung mit anderen Befehlen.

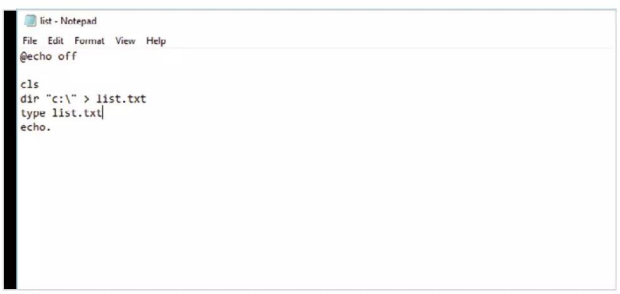
MEHRERE VARIABLEN ANWENDEN

Hier ist ein gutes Beispiel für das Kombinieren von System- und Umgebungsvariablen mit einigen Ihrer eigenen sowie einer Reihe externer Windows-Befehle.

SCHRITT 1

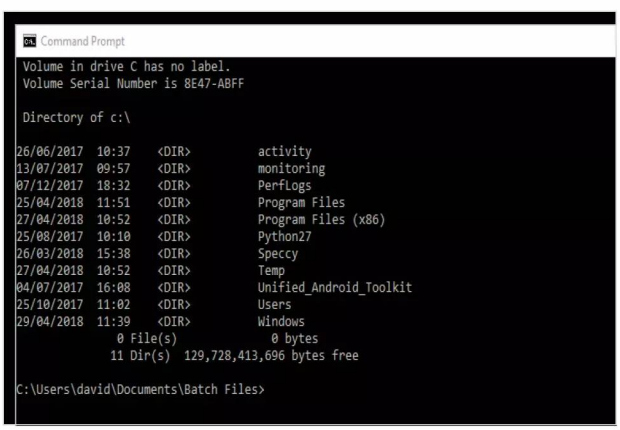
Erstellen Sie eine neue Batch-Datei namens list.bat und starten Sie sie mit dem Befehl `@echo off`. Beginnen Sie, indem Sie den Eingabeaufforderungsbildschirm löschen und eine Liste der aktuellen Verzeichnisse auf dem Computer anzeigen:

```
cls
dir “c:\” > list.txt
type list.txt
echo.
```



SCHRITT 2

Speichern Sie die Batch-Datei und führen Sie sie aus. In der Eingabeaufforderung können Sie den Inhalt aller Dateien und Verzeichnisse im Stammverzeichnis des C:\-Laufwerks sehen. Da jeder Benutzer unter Windows berechtigt ist, dies anzuzeigen, sind keine erhöhten Berechtigungen erforderlich.



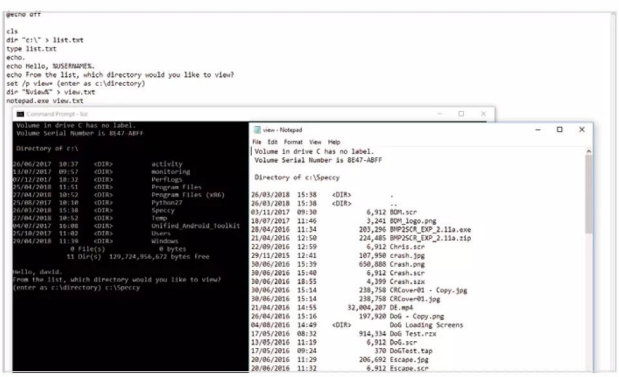
SCHRITT 3

Erstellen Sie nun eine Batch-Datei, die den Inhalt eines Verzeichnisses als Textdatei auf dem Bildschirm des Benutzers anzeigt. Fügen Sie der Batch-Datei list.bat Folgendes hinzu:

```
echo Hello, %USERNAME%.
echo From the list, which folder would you like to view?
set /p view= (enter as c:\folder)
dir “%view%” > view.txt
notepad.exe view.txt
```

SCHRITT 4

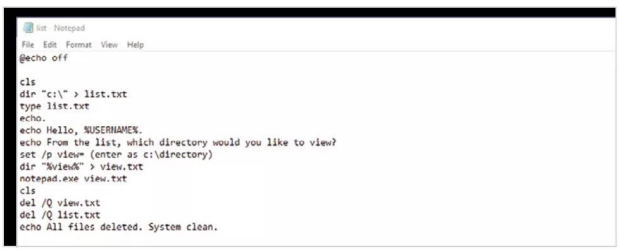
Die Batch-Datei fordert hier den Benutzer auf, eines der Verzeichnisse in der von ihr generierten Liste in Form von **c:\directory** einzugeben. Vorausgesetzt, der Benutzer gibt ein gültiges Verzeichnis ein, wird deren Inhalt als Textdatei angezeigt. Die view-Variable wurde hier zusammen mit %HOMEPATH% erstellt, um die Eingabe und die Textdatei zu speichern.



SCHRITT 5

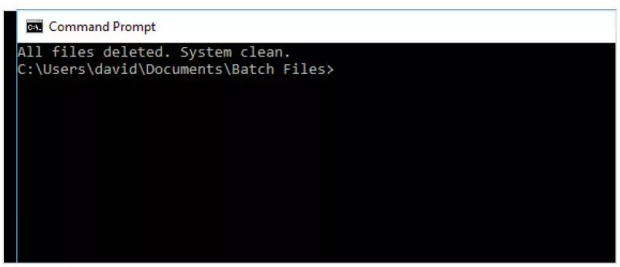
Es ist immer sinnvoll, Textdateien, die für die temporäre Ansicht des Benutzers erstellt werden, anschließend wieder zu entfernen. Es gibt nichts Schlimmeres als unzählige zufällige Textdateien, die das Dateisystem durcheinanderbringen. Sorgen Sie daher mit folgender Eingabe für Ordnung:

```
cls
del /Q view.txt
del /Q list.txt
echo All files deleted. System clean.
```



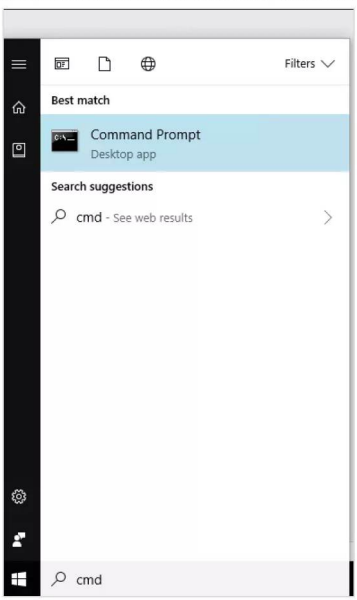
SCHRITT 6

Die Ergänzungen der Batch-Datei leeren mit dem cls-Befehl das Eingabeaufforderungsfenster und löschen sowohl die Dateien view.txt als auch list.txt, die von der Batch-Datei erstellt wurden. Der /Q-Teil im del-Befehl bedeutet, dass die Dateien ohne Benutzereingabe oder Benachrichtigung gelöscht werden. Die Nachricht am Ende teilt dem Benutzer mit, dass die Dateien entfernt wurden.



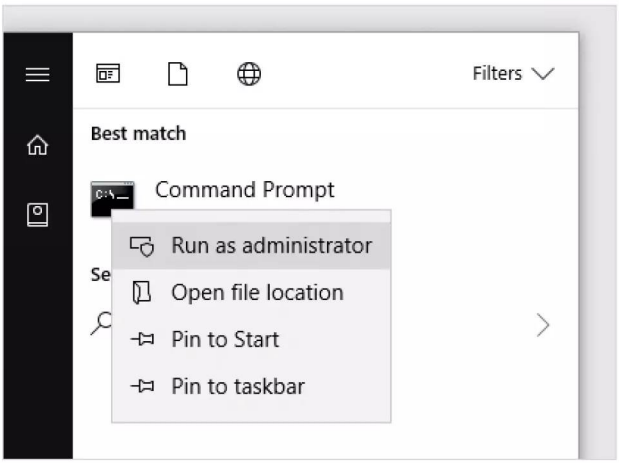
SCHRITT 7

Abhängig von Ihrer Systemkonfiguration erhalten Sie möglicherweise keine Verzeichnisinformationen oder die Meldung, dass der Zugriff verweigert wird. Dies liegt daran, dass die Benutzerkontensteuerung den Zugriff auf geschützte Bereiche des Systems wie C:\Windows oder C:\Program Files blockiert. Sie müssen die Batch-Datei daher als Administrator ausführen. Klicken Sie auf das Windows-Menüzeichen und geben Sie erneut **Eingabeaufforderung** oder **CMD** ein.



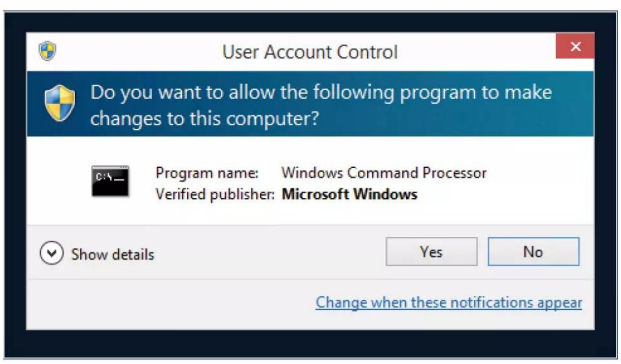
SCHRITT 8

Anstatt in den Ergebnissen per Linksklick die Eingabeaufforderung zu öffnen, machen Sie auf ihr einen Rechtsklick und wählen Sie im Menü **Als Administrator ausführen**. Als Administrator besteht das Risiko, dass Sie Systemdateien beschädigen, solange Sie aber vorsichtig sind und nichts anderes tun, als Verzeichnisse anzuzeigen, wird nichts passieren.



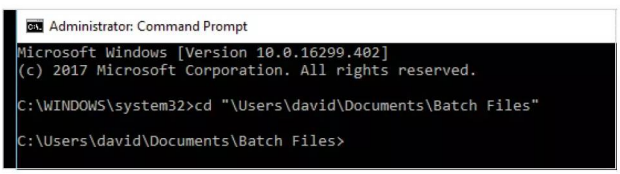
SCHRITT 9

Diese Aktion löst die Warnmeldung der Benutzerkontensteuerung aus, in der Sie gefragt werden, ob Sie sicher sind, dass Sie die Windows-Eingabeaufforderung mit den erhöhten Administratorrechten ausführen möchten. In der Regel würden wir dies nicht empfehlen, da die Benutzerkontensteuerung Ihr System schützt. Klicken Sie in diesem Fall jedoch auf Ja.



SCHRITT 10

Bei aktiver Benutzerkontensteuerung sieht die Eingabeaufforderung etwas anders aus. Zunächst wird standardmäßig im Ordner C:\WINDOWS\system32 geöffnet und der Fenstername enthält die Bezeichnung Administrator. Um die Batch-Datei auszuführen, müssen Sie mit **cd \Users\USERNAME\Documents\Batch Files** zum Batch Files-Verzeichnis navigieren. Mit der Tab-Taste lassen sich die Verzeichnisnamen automatisch vervollständigen.



Batch-Programmierung

Es sind die kleinen Ergänzungen, die wir einer Batch-Datei hinzufügen können, die sie hervorheben und letztendlich nützlicher machen. Die Befehlszeile ist der grafischen Windows-Interface ebenbürtig und bringt Batch-Dateien besonders gut zur Geltung.

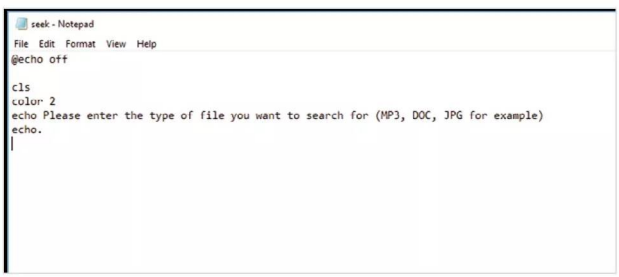
NACH DATEIEN SUCHEN

Hier ist eine interessante kleine Batch-Datei, die Sie leicht für Ihren eigenen Gebrauch erweitern können. Sie fragt den Benutzer nach einem Dateityp, nach dem sie suchen soll, und zeigt die Ergebnisse an.

SCHRITT 1

Wir führen hier einige neue Befehle ein, die wir äußerst nützlich finden. Erstellen Sie eine neue Batch-Datei namens seek.bat und geben Sie Folgendes ein:

```
@echo off
cls
color 2
echo Please enter the type of file you want to
search for (MP3, DOC, JPG for example)
echo.
```



SCHRITT 2

Der neue Befehl lautet color (amerikanische Schreibweise); er ändert die Farbe der Eingabeaufforderung. Die Farbattribute werden durch zwei Hex-Ziffern angegeben. Die erste Ziffer entspricht der Hintergrundfarbe der Befehlskonsole und die zweite dem Vordergrund. Sie können einen der folgenden Werte haben:

0 = Black	8 = Grey
1 = Blue	9 = Light Blue
2 = Green	A = Light Green
3 = Aqua	B = Light Aqua
4 = Red	C = Light Red
5 = Purple	D = Light Purple
6 = Yellow	E = Light Yellow
7 = White	F = Bright White

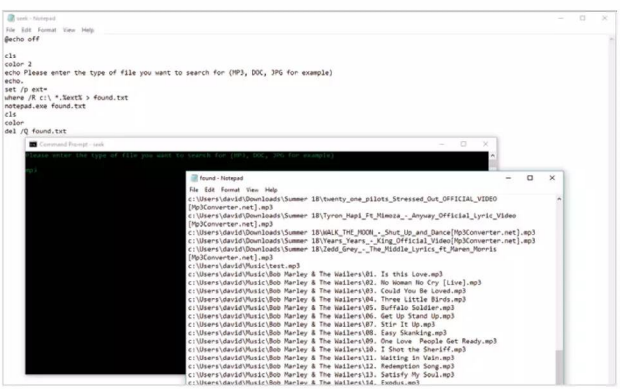
SCHRITT 3

Wir erweitern nun die Batch-Datei seek.bat:

```
@echo off
cls
color 2
echo Please enter the type of file you want to
search for (MP3, DOC, JPG for example)
echo.
set /p ext=
where /R c:\ *.%ext% > found.txt
notepad.exe found.txt
cls
color
del /Q found.txt
```

SCHRITT 4

Ein weiterer neuer Befehl, where, sucht, basierend auf der Anfrage des Benutzers, nach einer bestimmten Datei oder einem Verzeichnis. Hier haben wir eine leere Variable namens ext erstellt, in die der Benutzer den Dateityp eingeben kann. Anschließend wird mit where gesucht und die Ergebnisse in einer Textdatei mit dem Namen found.txt abgelegt. Speichern Sie die Batch-Datei und führen Sie sie aus.



AUSWAHLMENÜS

Das Erstellen von Auswahlmenüs mit dem choice-Befehl ist eine klassische Anwendung für eine Batch-Datei und eignet sich gut, um Ihre Programmierfähigkeiten für Batch-Dateien zu erweitern. Der folgende Code hilft Ihnen zu verstehen, wie das Ganze funktioniert.

SCHRITT 1

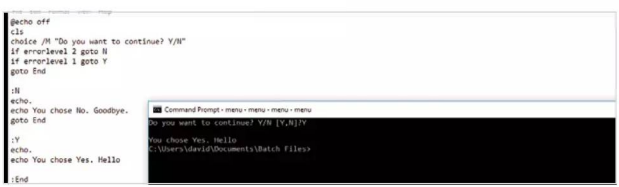
Anstatt eine Variable zur Bearbeitung der Benutzereingabe zu nehmen, können Batch-Dateien mit dem choice-Befehl und einem ErrorLevel-Parameter ein Menü erstellen. Erstellen Sie eine neue Datei namens menu.bat und geben Sie Folgendes ein:

```
@echo off
cls
choice /M “Do you want to continue? Y/N”
if errorlevel 2 goto N
if errorlevel 1 goto Y
goto End:
```

SCHRITT 2

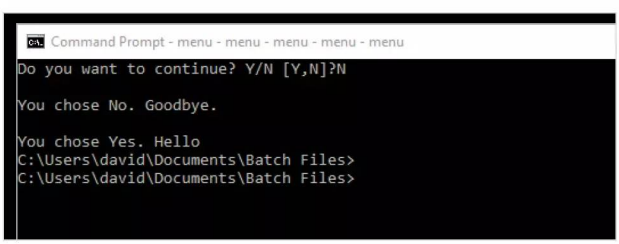
Die Ausführung des Codes erzeugt einen Fehler, da wir in der Datei einen goto-Befehl ohne einen Verweis darauf aufgerufen haben. Der goto-Befehl geht zu einer bestimmten Zeile in der Batch-Datei. Beenden Sie die Datei mit den folgenden Anweisungen und führen Sie sie erneut aus:

```
:N
echo.
echo You chose No. Goodbye.
goto End

:Y
echo.
echo You chose Yes. Hello

:End
```



SCHRITT 3

Die Ausgabe Ihrer Wahl hängt davon ab, ob Sie Y oder N auswählen. Der :End-Teil markiert lediglich das Ende der Datei (auch als EOF bekannt), ohne dem die Batch-Datei jede Zeile durchläuft und die Y-Antworten anzeigt, auch wenn Sie N eingeben; es ist daher wichtig, den goto-Befehlen zu folgen.



SCHRITT 4

ErrorLevels sind im Grunde Variablen. Die /M-Option des choice-Befehls ermöglicht die Anzeige eines beschreibenden Nachrichtenstrings. Erweitern Sie nun das Menü:

```
@echo off
cls
echo.
echo ------------------------------------------
echo.
echo Please choose a directory.
echo.
echo Press 1 for c:\Music
echo.
echo Press 2 for c:\Documents
echo.
echo Press 3 for c:\Pictures
echo.
echo Press 4 for c:\Videos
echo.
echo ------------------------------------------
choice /C 1234
if errorlevel 4 goto Videos
if errorlevel 3 goto Pictures
if errorlevel 2 goto Documents
if errorlevel 1 goto Music
```

SCHRITT 5

Fügen Sie nun die goto-Abschnitte hinzu:

```
:Videos
cls
CD %HOMEPATH%\Videos
echo You are now in the Videos directory.
goto End

:Pictures
cls
CD %HOMEPATH%\Pictures
echo You are now in the Pictures directory.
goto End

:Documents
cls
CD %HOMEPATH%\Documents
echo You are now in the Documents directory.
goto End

:Music
cls
CD %HOMEPATH%\Music
echo you are now in the Music directory.
goto End

:End
```

SCHRITT 6

Bei der Ausführung wird ein Menü angezeigt, und mit jeder Auswahl wechselt der Code ins vom Benutzer eingegebene Verzeichnis. Die Systemvariable %HOMEPATH% gibt die Verzeichnisse Videos etc. des angemeldeten Benutzers an.

Schleifen & Wiederholungen

Schleifen- und Wiederholungsbefehle bilden die Basis einer jeden Programmiersprache, einschließlich der Batch-Dateien. Sie können z. B. einen einfachen Countdown oder sogar nummerierte Dateien oder Verzeichnisse im System erstellen.

ZÄHLER

Das Erstellen von Code, der auf- bzw. abwärts zählt, eignet sich ideal für das Demonstrieren von Schleifen. Wir schauen uns dazu die if-Anweisung etwas genauer an und fügen ein paar weitere Variablen sowie die Befehle else, timeout und eof (end of file) hinzu.

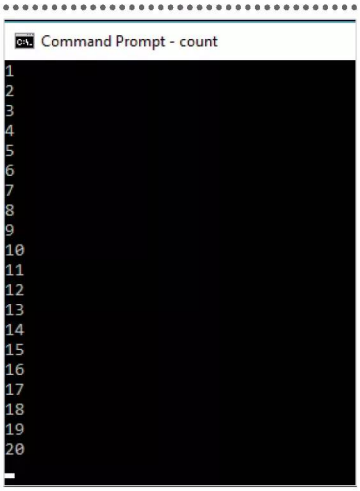
SCHRITT 1

Erstellen Sie eine neue Batch-Datei namens count.bat. Geben Sie den folgenden Code ein, speichern Sie ihn und führen Sie ihn aus:

```
@echo off
cls
set /a counter=0

:numbers
set /a counter=%counter%+1
if %counter% ==100 (goto :eof) else (echo
%counter%)
timeout /T 1 /nobreak > nul
goto :numbers
```

SCHRITT 2

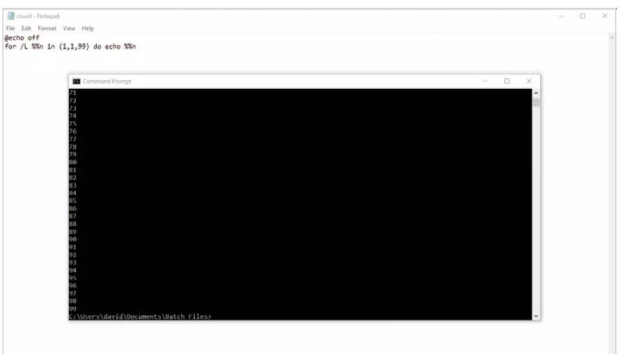
Der count.bat-Code beginnt bei Nummer eins und zählt aufwärts, bis er 100 erreicht. Der timeout-Befehl legt eine Pause von einer Sekunde zwischen den Zahlen fest und die else-Anweisung setzt das Ganze fort, bis die counter-Variable 100 entspricht; ist dies der Fall, wird über eof (end of file) die Schleife geschlossen.



SCHRITT 3

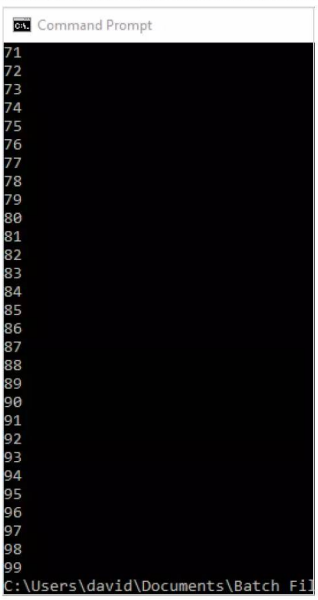
Die count.bat-Datei ist ein grobes Beispiel zur Veranschaulichung einer Schleife. Ein besserer Ansatz wäre die Nutzung einer for-Schleife. Probieren Sie dieses Beispiel aus:

```
@echo off
for /L %%n in (1,1,99) do echo %%n
```



SCHRITT 4

Wir haben hier for, gefolgt von der /L-Option, die einen Zahlenbereich umfasst. Als Nächstes haben wir den Parameter %%n, der eine Zahl kennzeichnet. Dann der in (1,1,99)-Teil, welcher der Anweisung mitteilt, wie zu zählen ist, d. h. 1 (Startnummer), 1 (zu ergreifende Schritte), 99 (Endnummer). Der do-Teil bedeutet, dass der folgende Befehl auszuführen ist.



SCHRITT 5

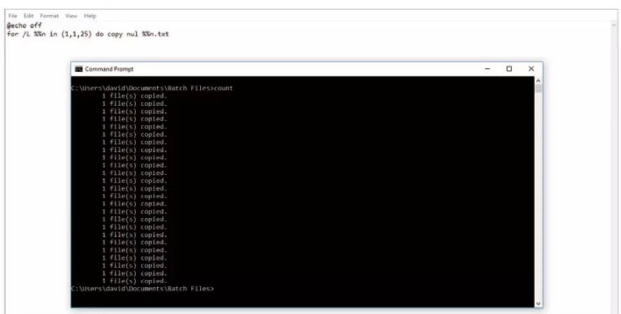
Sie können die Pause zwischen den Zahlen ganz leicht in die weitaus einfachere for-Schleife einfügen, indem Sie nach der do for-Schleife mehrere Befehle hinzufügen. Die Klammern und das Und-Zeichen (&) trennen die verschiedenen Befehle. Probieren Sie Folgendes:

```
@echo off
for /L %%n in (1,1,99) do (echo %%n & timeout /T 1
/nobreak > nul)
```

SCHRITT 6

Eine der großen Zeiteinsparungen bei Batch-Dateien ist das Erstellen mehrerer nummerierter Dateien. Angenommen, Sie möchten 25 Textdateien in einem Verzeichnis haben, die alle von 1 bis 25 nummeriert sind. Die for-Schleife machts möglich:

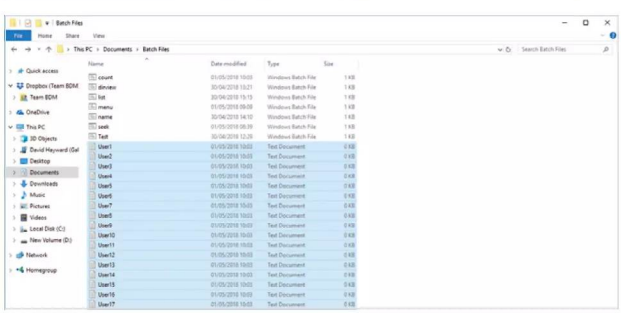
```
@echo off
for /L %%n in (1,1,25) do copy nul %%n.txt
```



SCHRITT 7

Wenn Sie im Windows Explorer zum aktuellen Batch-Dateien-Verzeichnis navigieren, werden jetzt 25 Textdateien angezeigt, die alle ordentlich nummeriert sind. Natürlich können Sie dem Dateinamen etwas wie user1.txt usw. hinzufügen. Ändern Sie dazu den Code wie folgt:

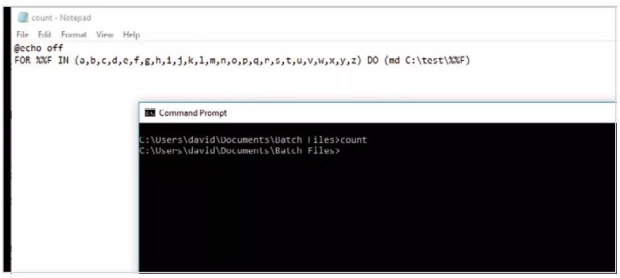
```
@echo off
for /L %%n in (1,1,25) do copy nul User%%n.txt
```



SCHRITT 8

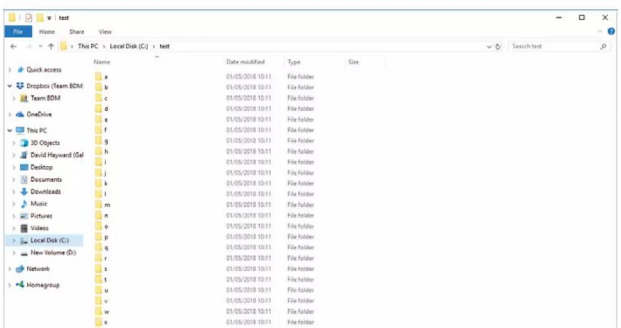
Es gibt verschiedene Anwendungsmöglichkeiten für die for-Schleife. In diesem Beispiel erstellt der Code innerhalb des Verzeichnisses c:\text, den die Batch-Datei per md-Befehl erstellt, 26 Verzeichnisse, eines für jeden Buchstaben des Alphabets:

```
@echo off
FOR %%F IN (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,
s,t,u,v,w,x,y,z) DO (md C:\test\%%F)
```



SCHRITT 9

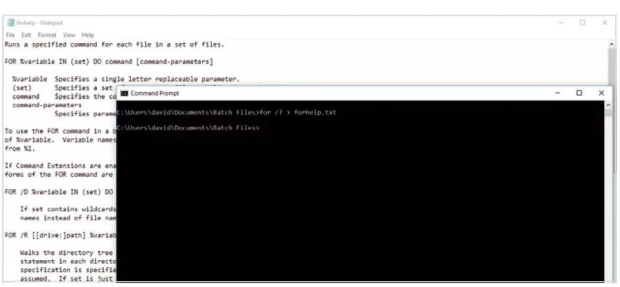
Schleifen können leistungsstarke und äußerst nützliche Elemente in einer Batch-Datei sein. Das Erstellen von 26 Verzeichnissen mag nicht allzu nützlich erscheinen, aber stellen Sie sich jedoch vor, Sie müssen 1 000 Benutzer in einem Netzwerk erstellen und jedem eine eigene Reihe eindeutiger Verzeichnisse zuweisen. Hier spart eine Batch-Datei enorm viel Zeit.



SCHRITT 10

Sollten Sie mit den verschiedenen Befehlen in einer Batch-Datei mal nicht weiterkommen, gehen Sie in die Eingabeaufforderung und geben Sie den Befehl gefolgt vom Schrägstrich und Fragezeichen ein, z. B. for/? oder if/?. Sie erhalten eine Hilfedatei auf dem Bildschirm, in der die Anwendung der Befehle beschrieben wird. Um das Lesen zu erleichtern, geben Sie sie in einer Textdatei aus:

```
For /? > forhelp.txt
```

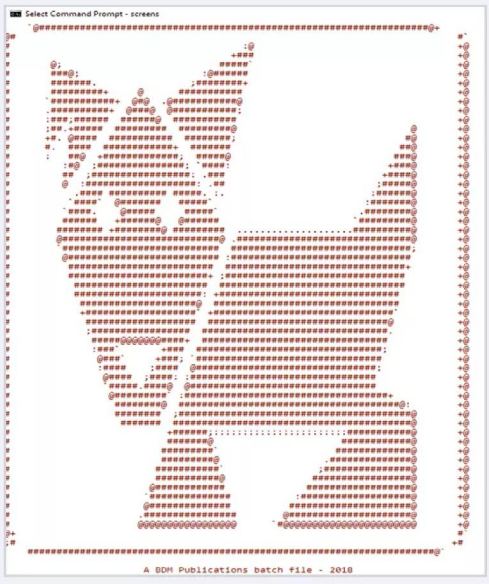


Ein Batch-Spiel erstellen

Basierend auf dem, was Sie bisher mit Batch-Dateien gelernt haben, können Sie wahrscheinlich mit Ihrem eigenen einfachen Textabenteuer oder Multiple-Choice-Spiel aufwarten. Hier ist eins, das wir erstellt haben und das Sie nach Herzenslust bearbeiten können.

Stellen Sie Ihre eigenen Fragen zusammen und fügen Sie einen Einführungs- oder Ladebildschirm hinzu. Legen Sie den Ladebildschirm als separate Batch-Datei an und speichern Sie ihn z. B. als screens.bat. Sie können ihn dann über die Batch-Datei des Hauptspiels zu Beginn mit dem call-Befehl laden, gefolgt vom color-Befehl, um die Farben des Spiels zurückzusetzen:

```
@echo off
Cls
Call screens.bat
color
:start
set /a score=0
set /a question=0
cls
set /p name= What is your name?
```




```
@echo off
Cls
:start
set /a score=0
set /a question=0
cls
set /p name= What is your name?

:menu
cls
echo.
echo ****************************************************
   **
echo.
echo Welcome %name% to the super-cool trivia game.
echo.
echo Press 1 to get started
echo.
echo Press 2 for instructions
echo.
echo Press Q to quit
echo.
echo ****************************************************
   **
choice /C 12Q
if errorlevel 3 goto :eof
if errorlevel 2 goto Instructions
if errorlevel 1 goto Game

:Instructions
cls
echo.
echo ************************************
echo.
echo The instructions are simple. Answer the
   questions correctly.
echo.
echo ************************************
pause
cls
goto menu

:Game
set /a question=%question%+1
cls
if %question% ==5 (goto end) else (echo you are on
   question %question%)
echo.
echo get ready for the question...
echo.
timeout /T 5 /nobreak > nul
if %question% ==5 (goto end) else (goto %question%)

:1
cls
echo.
echo ******************************
echo.
```

```
echo Your current score is %score%
echo.
echo ******************************
echo.
echo.
echo Question %question%.
echo.
echo Which of the following version of Windows is the
  best?
echo.
echo A. Windows 10
echo.
echo B. Windows ME
echo.
echo C. Windows Vista
echo.
choice /C abc
if errorlevel 3 goto wrong
if errorlevel 2 goto wrong
if errorlevel 1 goto correct

:2
cls
echo.
echo ******************************
echo.
echo Your current score is %score%
echo.
echo ******************************
echo.
echo.
echo Question %question%.
echo.
echo Which of the following version of Windows is the
  most stable?
echo.
echo A. Windows 10
echo.
echo B. Windows 95
echo.
echo C. Windows ME
echo.
choice /C abc
if errorlevel 3 goto wrong
if errorlevel 2 goto wrong
if errorlevel 1 goto correct

:3
cls
echo.
echo ******************************
echo.
echo Your current score is %score%
echo.
echo ******************************
echo.
echo.
echo Question %question%.
echo.
echo Which of the following Windows version is the
  latest?
echo.
echo A. Windows 10
echo.
echo B. Windows 98
echo.
echo C. Windows 7
echo.
choice /C abc
if errorlevel 3 goto wrong
if errorlevel 2 goto wrong
if errorlevel 1 goto correct

:4
cls
echo.
echo ******************************
echo.
echo Your current score is %score%
echo.
echo ******************************
echo.
echo.
echo Question %question%.
echo.
echo Which of the following Windows uses DirectX 12?
echo.
echo A. Windows 10
echo.
echo B. Windows 3.11
echo.
echo C. Windows XP
echo.
choice /C abc
if errorlevel 3 goto wrong
if errorlevel 2 goto wrong
if errorlevel 1 goto correct

:Wrong
cls
echo ******************
echo.
echo WRONG!!!!
echo.
echo ******************
set /a score=%score%-1
pause
goto :game

:correct
cls
echo ******************
echo.
echo CORRECT. YIPEE!!!
echo.
echo ******************
set /a score=%score%+1
pause
goto :game

:end
cls
echo ******************************
echo.
echo Well done, %name%, you have answered all the
  questions
echo.
echo And your final score is....
echo.
echo %score%
echo.
echo ********************************
choice /M "play again? Y/N"
if errorlevel 2 goto :eof
if errorlevel 1 goto start
```

Linux ist ein Betriebssystem, das kostenlos heruntergeladen und verwendet werden kann. Es ist die treibende Kraft hinter den Supercomputern der Welt und sogar den wissenschaftlichen Laboren des Large Hadron Collider. Linux hat eine riesige, weltweite Gemeinschaft und bedeutet Freiheit von der geschlossenen Plattformstruktur von Windows und macOS.

Das Erlernen von Scripting unter Linux kommt Ihren Aussichten als Programmierer zugute, und bietet Ihnen ein neues Verständnis über das Betriebssystem an sich sowie die Interaktion zwischen verschiedenen Sprachen.

Zu guter Letzt werfen wir einen Blick auf einige der häufigsten Programmierfehler und erklären, wie sie behoben werden können.

Programmieren unter Linux

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

– Linus Torvalds
(Entwickler des Linux-Kernels)

Warum Linux?

Viele Entwickler, ganz gleich, mit welchen der erhältlichen Sprachen sie arbeiten, verwenden Linux als Betriebssystembasis für das Programmieren und Testen, aber warum? Linux hat viele Vorteile gegenüber anderen Systemen, und trotz einiger Macken ist Linux zum Programmieren lernen gut geeignet.

GRATIS UND OPEN-SOURCE

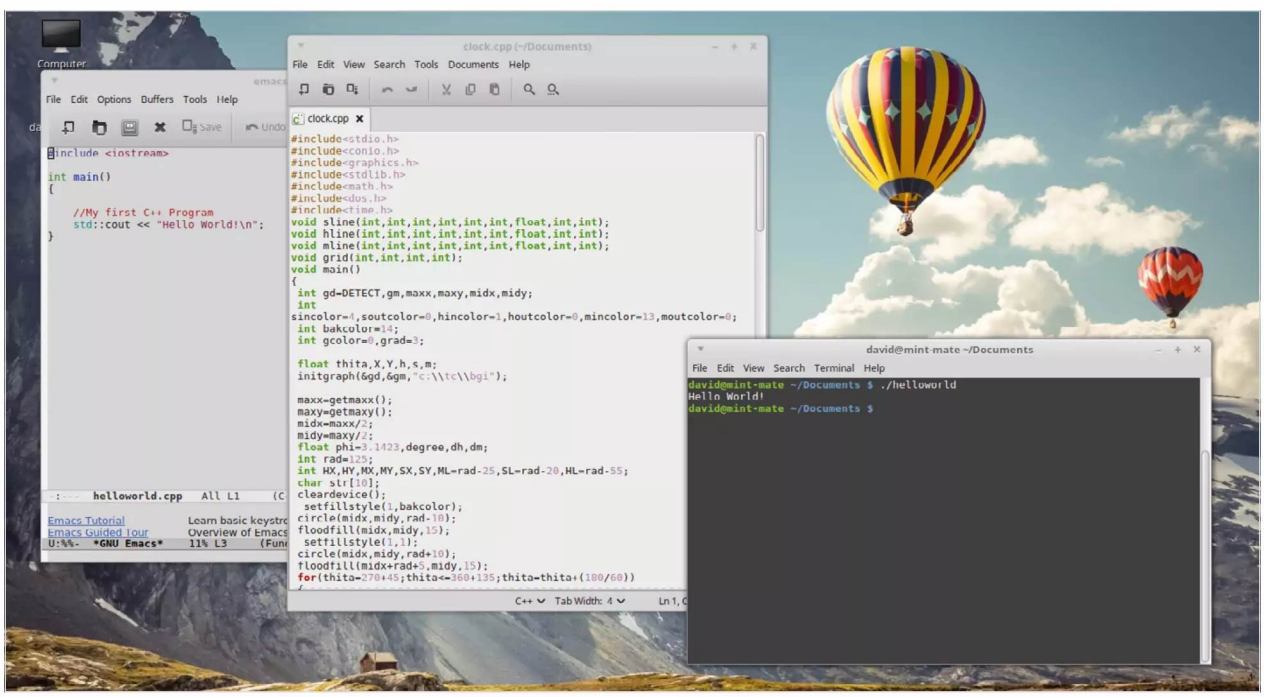
Linux eignet sich hervorragend für alle, die plattformübergreifenden Code entwickeln möchten. Die Effizienz des Systems, die Verfügbarkeit von Anwendungen und seine Stabilität sind nur einige gute Gründe.

Zunächst sei erwähnt, dass es kein Linux-Betriebssystem gibt. Linux ist der Betriebssystem-Kernel, die Hauptkomponente eines Betriebssystems. Wenn über Linux gesprochen wird, ist eine der vielen Distributionen gemeint, die den Linux-Kernel verwenden. Zweifellos haben Sie von mindestens einer bereits gehört: Ubuntu, Linux Mint, Fedora, openSUSE, Debian usw. Jede dieser Distributionen bietet dem Benutzer etwas anderes. Sie alle basieren zwar auf einem Linux-Kernel, haben jedoch unterschiedlich aussehende Desktop-Umgebungen, verschiedene vorinstallierte Anwendungen, verschiedene Möglichkeiten zur Systemaktualisierung und zum Installieren weiterer Apps, und jede bietet dem Benutzer schlicht und einfach ein anderes Gefühl bei der Anwendung. Im Zentrum befindet sich aber Linux, und deshalb ist auch von Linux die Rede.

Linux Funktionsweise unterscheidet sich deutlich von Windows und macOS. Linux kann kostenlos auf beliebig vielen Computern heruntergeladen und installiert sowie unbegrenzt verwendet werden. Auch das Upgrade und die Programme und Anwendungen für Erweiterungen sind gratis. Das Element der kostenlosen Verfügbarkeit ist einer der größten Anziehungspunkte für Entwickler. Während eine Windows-Lizenz um die 150 € und eine Mac-Lizenz noch höher sein kann, kann ein Entwickler hier schnell eine Distribution herunterladen und innerhalb weniger Minuten mit dem Programmieren beginnen.

Neben dem Gratis-Aspekt gibt es einen Grad an Freiheit, das System an die eigenen Bedürfnisse anzupassen. Jede der im Internet verfügbaren Distributionen verfügt über etwas Spezielles; einige bieten

Das Linux-Betriebssystem eignet sich großartig, um das Programmieren zu lernen.



erhöhte Sicherheit, andere einen toll aussehenden Desktop, andere wiederum eignen sich gut für Spiele und andere für Studierende. Diese Erweiterbarkeit macht Linux zu einer begehrten Plattform für das Erlernen des Programmierens, da das System leicht in eine Entwicklungsbasis geformt werden kann, u. a. auch für die vielen verschiedenen IDEs für z. B. Python, die Webentwicklung, C++, Java usw.

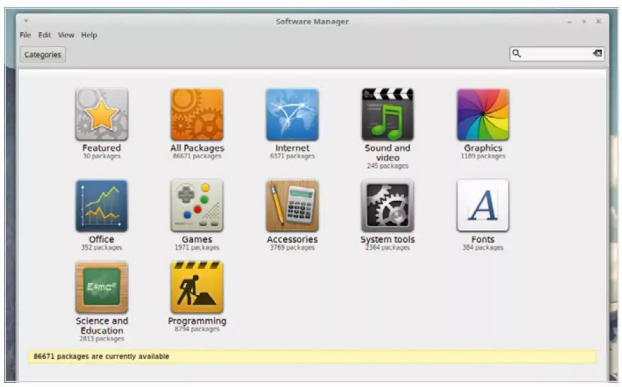
Ein weiterer bemerkenswerter Vorteil ist, dass Linux die meisten der gängigen Programmierumgebungen enthält. Sowohl Python als auch C++ sind in vielen verfügbaren Linux-Distributionen vorinstalliert. Das bedeutet, dass Sie mit dem Programmieren beginnen können, sobald Sie das System installiert haben und zum ersten Mal starten.

Im Allgemeinen beansprucht Linux nicht so viele Systemressourcen wie Windows oder macOS. Unter Systemressourcen verstehen wir Speicher, Festplattenspeicher und CPU. Der Linux-Code wurde rationalisiert und ist frei von „Bloatware" von Drittanbietern, die diese Ressourcen überlastet. Ein effizienteres System bedeutet natürlich mehr verfügbare Ressourcen für die Porgrammier- und Testumgebung sowie die von Ihnen erstellten Programme. Ein reduzierter Ressourcenverbrauch bedeutet auch, dass Sie Linux mit älterer Hardware verwenden können, die normalerweise die neuesten Versionen von Windows oder macOS nicht ausführen kann. Anstatt einen alten Computer wegzuwerfen, kann er mit einer Linux-Distribution wiederverwendet werden.

Es geht jedoch nicht nur um C++, Python oder eine der anderen gängigen Programmiersprachen. Mit der Befehlszeile von Linux, die auch als Terminal bezeichnet wird, können Sie Shell-Skripte erstellen. Hierbei handelt es sich um Programme, die über die Befehlszeile ausgeführt werden und aus Skriptsprachen bestehen. Sie dienen hauptsächlich der Automatisierung von Aufgaben und bieten dem Benutzer eine Form der Ein- und Ausgabe für einen bestimmten Vorgang. Sie sind überraschend leistungsstark – dazu in Kürze mehr.

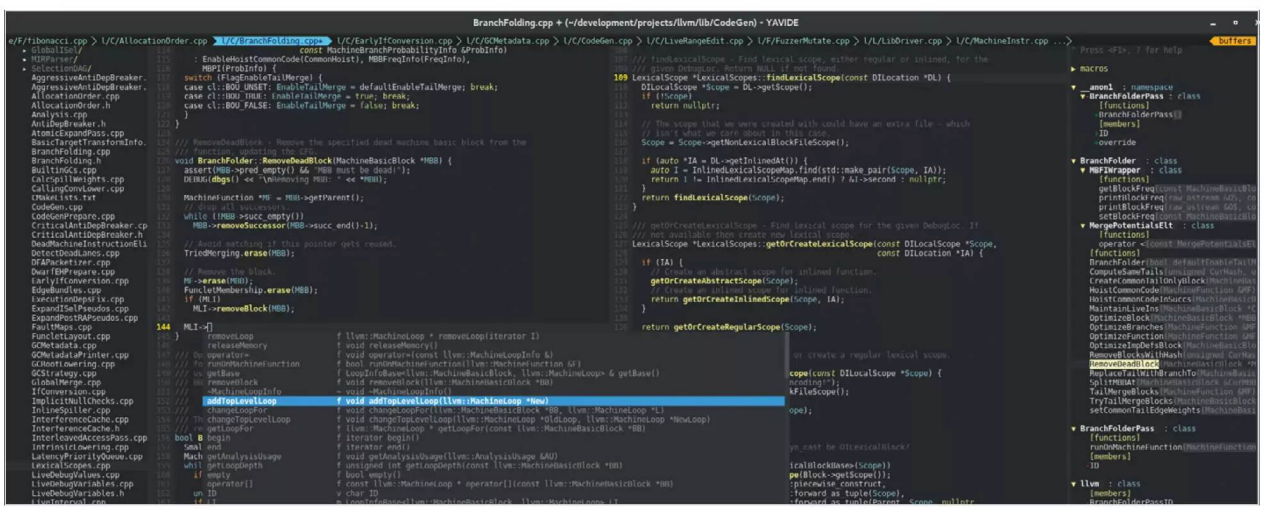
Neben vielen weiteren Vorteilen gibt es auch unzählige kostenlose Programme und Apps, die nahezu jeden Aspekt der Informatik abdecken. Zum Zeitpunkt des Schreibens gibt es allein für Linux Mint über 8 700 spezifische Programmieranwendungen.

Linux bietet daher eine tolle Programmier-Ressource und -Umgebung. Linux ist für Entwickler ideal und wird ständig verbessert und weiterentwickelt. Wenn Sie es ernsthaft mit dem Programmieren meinen, ist Linux einen Versuch wert.



Es gibt Tausende von kostenlosen Paketen für das Programmieren unter Linux.

Jede Distribution bietet dem Benutzer etwas Einzigartiges, aber hinter jeder steckt Linux.



Eine Linux-Programmierumgebung kann so einfach oder komplex sein, wie Sie es wünschen.

Die besten Linux-Distributionen

Es gibt viele Versionen von Linux, die als Distributionen bekannt sind. Jede hat ein anderes Ethos und einen anderen Ansatz. Hier sind fünf tolle Distributionen zum Ausprobieren, einschließlich der Website, auf der sie erhältlich sind.

LINUX MINT – LINUXMINT.COM

Die mit Abstand beliebteste Distribution ist Linux Mint. Mint wurde im Jahr 2006 als Alternative zur damals populärsten Distribution Ubuntu entwickelt. Obwohl Linux Mint auf Ubuntus Langzeitunterstützung basiert, hat es eine andere Richtung eingeschlagen und bot dem Benutzer eine bessere Erfahrung.

Linux Mint bietet mit jeder neuen Version des Betriebssystems drei primäre Desktop-Umgebungen an. Derzeit ist die Cinnamon Desktop-Umgebung das Aushängeschild von Linux Mint. Die anderen beiden sind MATE und Xfce.

Cinnamon ist eine grafisch aufwendig gestaltete Desktop-Umgebung. MATE verwendet weniger hochwertige Grafiken und ist auf einer Vielzahl von Desktop-Systemen stabiler. Xfce ist eine sehr rationalisierte Desktop-Umgebung, die auf Geschwindigkeit und ultimative Stabilität ausgelegt ist.

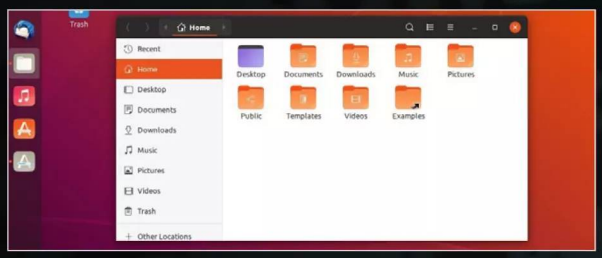
In dieser Ausgabe werden wir Cinnamon verwenden, Sie können aber auch andere Desktop-Umgebungen ausprobieren.

Es wird auch empfohlen, dass Sie verschiedene Umgebungen und verschiedene Distributionen ausprobieren, um herauszufinden, welche für Sie und Ihren Computer am besten geeignet ist.

UBUNTU – UBUNTU.COM

Die zweitbeliebteste Distribution ist Ubuntu, das ein altes afrikanisches Wort für „Menschlichkeit gegenüber anderen“ ist. Die Popularität von Ubuntu hat sich im Laufe seines 14-jährigen Daseins verändert. Ubuntu war einst mit Abstand das meistgenutzte Linux-basierte Betriebssystem der Welt, aber falsche Entscheidungen hinsichtlich der Darstellung und kontroverse Elemente, die den Datenschutz betrafen, haben Ubuntu von der Spitze verdrängt.

Ubuntu hat seitdem allerdings vieles wieder gut gemacht und sich langsam in die Liste der Linux-Spitzenreiter zurückerarbeitet. Die neuesten Versionen des Betriebssystems verwenden die beeindruckende GNOME 3-Desktop-Umgebung, die für ehemalige Windows-Benutzer jedoch etwas verwirrend sein kann. Sie belastet



auch die Systemressourcen, insbesondere auf älteren Computern.

Ubuntu ist trotz all seiner Fehler eine gute Linux-Distribution, mit der sich das Experimentieren lohnt. Ubuntu hat eine klare Oberfläche, ist einfach zu bedienen und zu installieren und bietet dem Benutzer das komplette Linux-Erlebnis.

ARCH – ARCHLINUX.ORG

Arch ist eine der am längsten laufenden Linux-Distributionen und bildet die Basis für viele andere Linux-Versionen. Warum dann aber Mint oder Ubuntu installieren? Während Arch zwar von vielen Benutzern verwendet wird, ist sie für Anfänger nicht gut geeignet. Ubuntu und Mint hingegen bieten einen einfacheren Installationspfad und werden mit Softwarepaketen geliefert, die den Einstieg erleichtern.

Arch ist eher eine Art „Barebone"-Angelegenheit. Die Arch-Distribution hat sich der freien Software verschrieben und ihre Paketquellen enthalten über 50 000 Apps. Auch mit Arch können mehrere verschiedene Desktop-Umgebungen installiert und verwendet werden.

Arch ist für erfahrene Linux-Anwender. Es wird direkt in der Befehlszeile begonnen. Von dort aus muss die Festplatte manuell partitioniert, der Speicherort der Installationsdateien festgelegt, ein Benutzer angelegt, das Gebietsschema des Betriebssystems festgelegt und letztendlich eine Desktop-Umgebung mit den gewünschten Apps installiert werden.

All das hat jedoch den Vorteil, dass man am Ende eine speziell für sich selbst erstellte Distribution hat, ohne all die unnötigen Dateien und Apps, die von anderen vorinstalliert wurden.

OPENSUSE – OPENSUSE.ORG

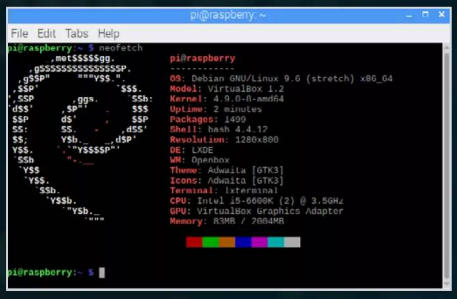
Die meisten Linux-Distributionen fallen in zwei Kategorien. Es gibt solche mit den neuesten Funktionen und Technologien wie Ubuntu und Mint, und solche mit wenigen neuen Funktionen, die aber eine solide Zuverlässigkeit aufweisen, wie Debian.

openSUSE versucht beide Lager abzudecken. openSUSE Leap ist ein grundsolides System. Es wurde offen von einer Gemeinschaft zusammen mit SUSE-Mitarbeitern entwickelt, die ein Betriebssystem auf Unternehmensebene entwickelten: SUSE, auf das u. a. auch die Londoner Börse setzt. Es wurde für missionskritische Umgebungen entworfen, in denen es keinen Spielraum für Instabilität gibt. Wem das zu ausgereift ist, für den gibt es openSUSE Tumbleweed, eine kontinuierliche Softwareentwicklung, die alle neuesten Funktionen enthält; und gelegentlich auch abstürzt.

openSUSE ist eine hoch angesehene Linux-Distribution und viele wichtige Mitwirkende arbeiten am Linux-Kernel, LibreOffice, Gnome und in anderen wichtigen Linux-Bereichen mit.

RASPBERRY PI DESKTOP – RASPBERRYPI.ORG/DOWNLOADS/RASPBERRY-PI-DESKTOP

Sie haben sicherlich schon vom Raspberry Pi gehört, dem bemerkenswerten, winzigen Computer, der seit seiner Einführung vor sieben Jahren die Technologiewelt im Sturm erobert hat.



Der Raspberry Pi hat mehrere Aspekte, die ihn in der Computerwelt so begehrt machen. Zum einen ist er billig – man erhält praktisch einen voll funktionsfähigen Computer für ca. 30 €. Er ist kaum größer als eine Kreditkarte und kann mit seiner vollständig programmierbaren Schnittstelle für den Bau von Elektronikprojekte eingesetzt werden. Er kommt mit Raspbian, seinem eigenen maßgeschneiderten, auf Debian basierenden Betriebssystem, das neben vielen verschiedenen Programmiersprachen und Bildungsressourcen auch eine Office-Suite enthält.

Raspbian war ursprünglich exklusiv für die Pi-Hardware , da der Raspberry Pi einen ARM-Prozessor für die Stromversorgung verwendet. Mittlerweile hat die Raspberry Pi Foundation jedoch eine PC-Version von Raspbian veröffentlicht: Raspberry Pi Desktop.

Genau wie die Pi-Version enthält auch Raspberry Pi Desktop alle Programmier-, Lern- und anderen Apps, die Sie benötigen. Sie ist schnell, stabil und funktioniert hervorragend. Wenn Sie Ihre Linux-Erfahrung ausbauen möchten, sollten Sie diese Distribution auf jeden Fall in Betracht ziehen.

Benötigtes Zubehör

Linux Mint gilt unter den vielen verschiedenen Distributionen als eine der besten für Anfänger sowie Fortgeschrittene. Sie ist eine ausgezeichnete Programmierplattform mit vielen integrierten Sprachen. Hier erfahren Sie, was Sie für die Arbeit mit Linux Mint benötigen.

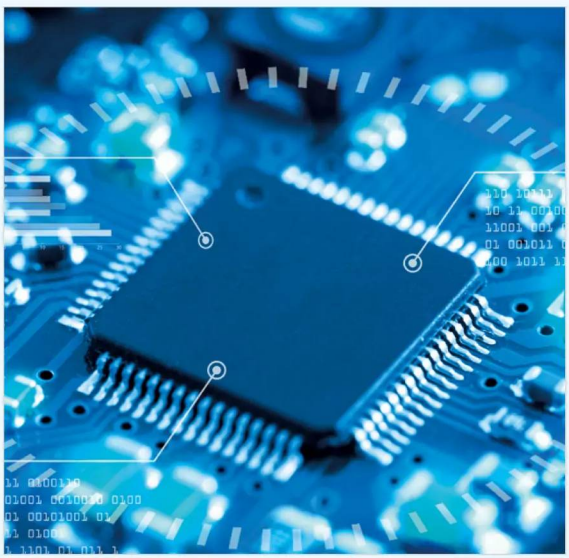
DURCH FREIHEIT ZUR ELEGANZ

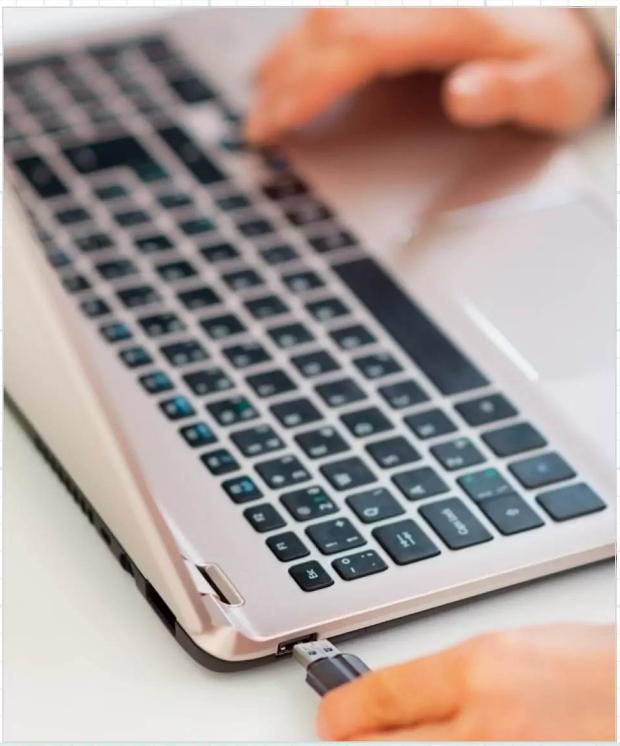
Dank der Vielseitigkeit von Mint stehen Ihnen mehrere Optionen zur Installation dieser Distribution zur Verfügung. Nehmen Sie sich Zeit und schauen Sie, welche Methode für Sie am besten geeignet ist.

SYSTEMVORAUSSETZUNGEN

Je besser Ihr System ist, desto besser und schneller ist natürlich die Anwendung. Die Mindestsystemanforderungen für Linux Mint 18 lauten wie folgt:

CPU	700 MHz
RAM	512 MB
Festplattenspeicher	9 GB (20 GB empfohlen)
Monitor	1024 x 768 Auflösung



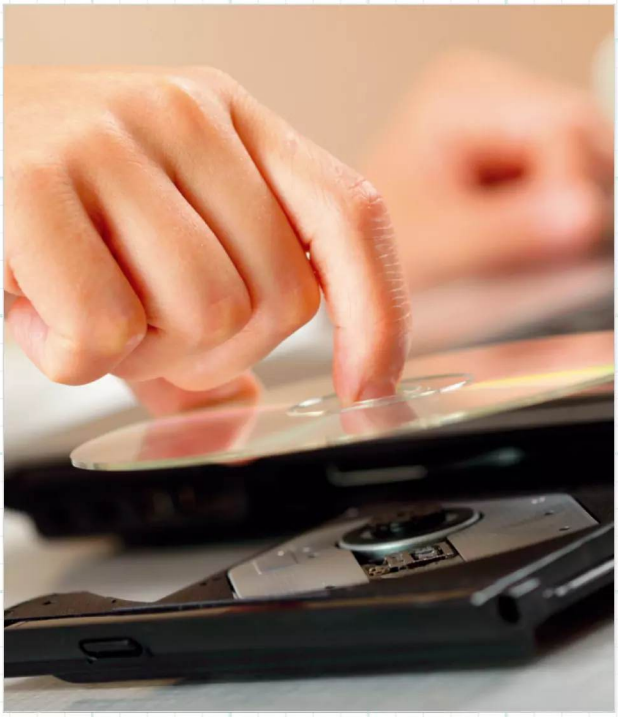


USB-INSTALLATION

Sie können Linux Mint über USB oder DVD auf Ihrem Computer installieren. Wir werden uns später noch beide Optionen einzeln ansehen. Für eine USB-Installation benötigen Sie für die Linux Mint ISO-Datei einen USB-Speicherstick mit einem Minimum von 4 GB.

DVD-INSTALLATION

Für die DVD-Installation von Linux Mint benötigen Sie eine leere DVD-R-Disk. Natürlich brauchen Sie auch einen DVD-Brenner, um das ISO-Abbild auf die Disk übertragen zu können.



VIRTUELLE UMGEBUNG

Die Installation in eine virtuelle Umgebung ist eine beliebte Methode zum Testen und Anwenden von Linux-Distributionen. Linux Mint arbeitet hervorragend in einer virtuellen Umgebung – dazu später mehr. Es gibt viele verschiedene Apps für virtuelle Umgebungen, in dieser Ausgabe verwenden wir jedoch Virtualbox von Oracle. Die neueste Version kann auf **www.virtualbox.org** heruntergeladen werden.

INTERNETVERBINDUNG

Es versteht sich von selbst, dass eine Internetverbindung unerlässlich ist, nicht nur um sicherzustellen, dass Linux Mint mit den neuesten Updates und Patches ausgerüstet ist, sondern auch, um weitere Software installieren zu können. Für die Anwendung von Linux Mint ist zwar keine Internetverbindung erforderlich, es entgeht Ihnen aber eine Reihe kostenloser Software, die für diese Distribution erhältlich ist.

MAC-HARDWARE

Linux Mint kann zwar auf einem Mac installiert werden, es gibt aber ein Denkmodell, das Mac-Besitzern empfiehlt, eine virtuelle Umgebung wie Virtualbox oder Parallels zu verwenden. Und warum auch nicht, macOS ist bereits ein hervorragendes Betriebssystem. Wenn Sie einem älteren Mac neues Leben einhauchen möchten, stellen Sie sicher, dass es sich um ein Intel-CPU-Modell und nicht um ein PowerPC-Modell handelt.

Linux-Installer unter Windows erstellen

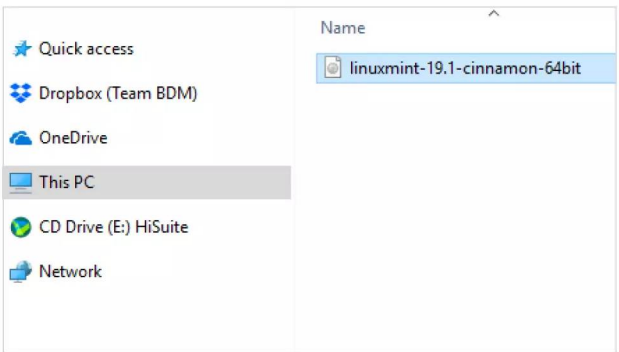
Vor der Installation müssen Sie die heruntergeladene Linux ISO-Datei auf eine DVD oder einen USB-Stick übertragen. Dies wird eine Live-Umgebung sein, in der Sie das OS vor der Installation testen können. Zuerst müssen Sie aber das bootfähige Medium erstellen.

BOOTFÄHIGE DVD-MEDIEN

Wir verwenden zur Übertragung der ISO-Datei auf eine DVD einen Windows 10-PC. Solange Sie eine Windows-Version ab 7 verwenden, ist der Vorgang sehr einfach.

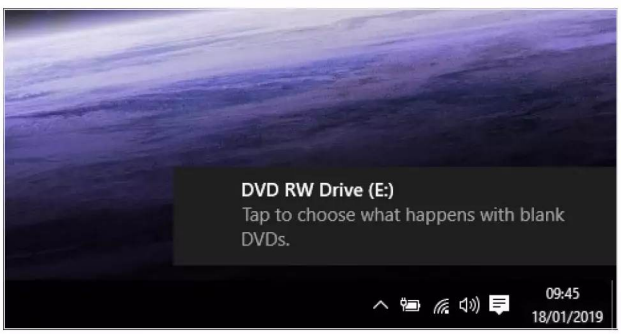
SCHRITT 1

Suchen Sie zuerst nach dem bereits heruntergeladenen ISO-Abbild von Linux. Auf PCs mit Windows 7, 8.1 und 10 finden Sie es in der Regel im Downloads-Ordner, es sei denn, Sie haben beim Speichern einen anderen Speicherort gewählt.



SCHRITT 2

Legen Sie nun eine beschreibbare DVD in das optische Laufwerk Ihres Computers ein. Die DVD wird gelesen und nach ein paar Sekunden erscheint von Windows die Anfrage, was Sie mit der eingelegten DVD machen möchten. Ignorieren Sie dies, da wir die eingebaute Brennfunktion verwenden werden.

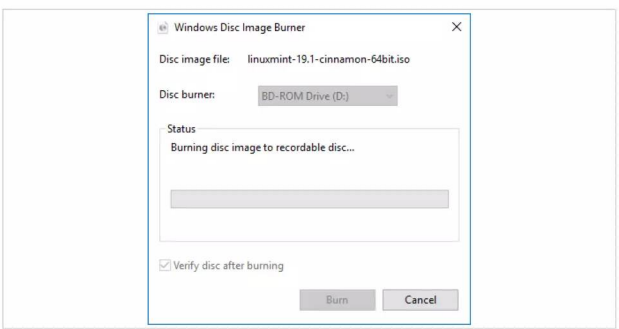


SCHRITT 3

Machen Sie auf der Linux ISO-Datei einen Rechtsklick und wählen Sie „Datenträgerabbild brennen". Je nach PC-Geschwindigkeit kann es einige Sekunden dauern, bis etwas passiert. Sollte es aber länger als eine Minute dauern, lohnt es sich, den PC neu zu starten und es noch einmal zu versuchen. Mit etwas Glück sollte der „Windows-Brenner für Datenträgerabbilder" starten.

SCHRITT 4

Klicken Sie im geöffneten Dialogfeld „Windows-Brenner für Datenträgerbilder" auf „Datenträger nach dem Brennen überprüfen" und dann auf „Brennen". Der Vorgang wird je nach Geschwindigkeit Ihres optischen Laufwerks einige Minuten dauern. Nach Beendigung wird der Datenträger überprüft. Die DVD sollte anschließend automatisch ausgeworfen werden.

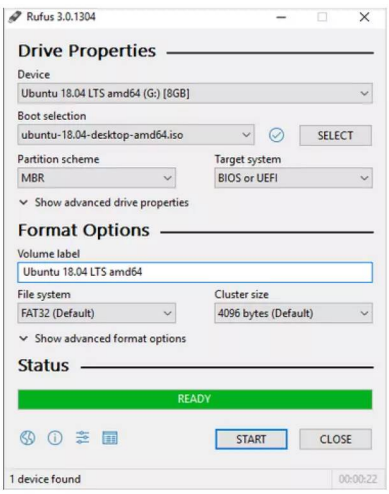


BOOTFÄHIGE USB-MEDIEN

USB-Medien sind schneller als DVDs und oftmals auch praktischer, da die meisten modernen PCs keine optischen Laufwerke mehr haben. Für die Übertragung des ISO-Abbilds benötigen Sie eine Drittanbieter-App und einen mindestens 4 GB großen USB-Stick.

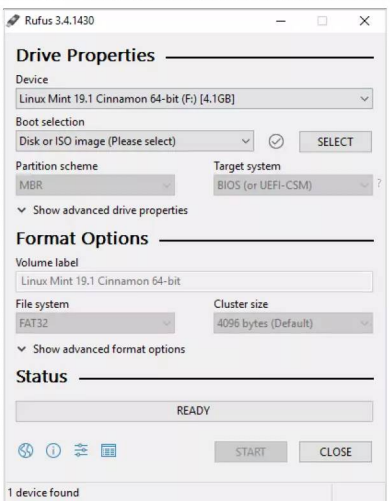
SCHRITT 1

Öffnen Sie einen Webbrowser und gehen Sie zu **https://rufus.ie/**. Scrollen Sie runter bis Sie zum Download-Abschnitt kommen. Hier finden Sie die neueste Version von Rufus. Durch Anklicken wird der Download gestartet.



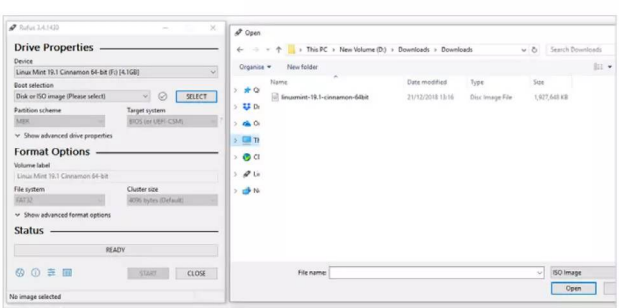
SCHRITT 2

Doppelklicken Sie auf die heruntergeladene Rufus-Datei und bestätigen Sie die Windows-Sicherheitsfrage sowie auch die Suche nach Aktualisierungen mit „Ja". Nachdem Rufus hochgefahren ist, sollte die App Ihren angeschlossenen USB-Stick erkennen können; falls nicht, entfernen und schließen Sie ihn neu an.



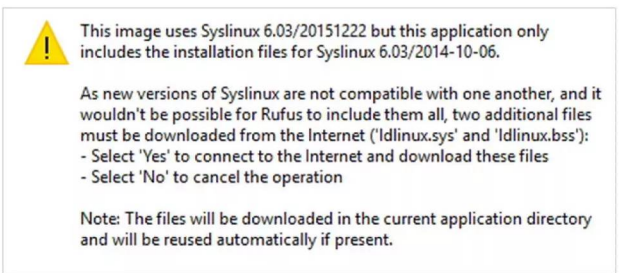
SCHRITT 3

Auf den ersten Blick erscheint die Rufus-Oberfläche etwas verwirrend, aber Rufus ist wirklich sehr einfach. Klicken Sie zu Beginn neben dem Startart-Auswahlmenü auf die Schaltfläche AUSWAHL. Dadurch wird das Windows Explorer-Fenster geöffnet. Wählen Sie darin die Linux ISO-Datei aus.



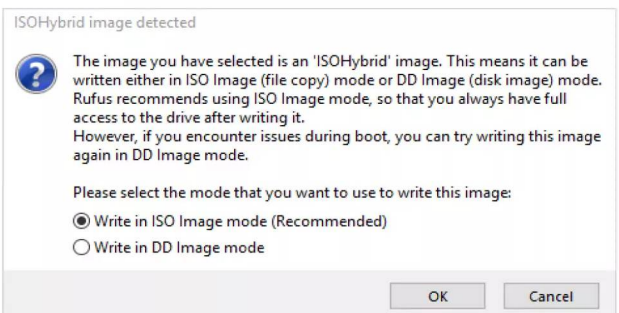
SCHRITT 4

Wenn Sie so weit sind, klicken Sie unten in der Rufus-App auf die Starttaste. Dadurch wird ein weiteres Dialogfeld geöffnet, das Sie bittet, eine neue Version von SysLinux herunterzuladen und zu verwenden. Dabei handelt es sich um eine Auswahl von Bootloadern, die einem modernen PC ermöglichen, auf ein USB-Laufwerk zuzugreifen und darüber hochzufahren. Dies ist notwendig, klicken Sie daher auf „Ja", um fortzufahren.



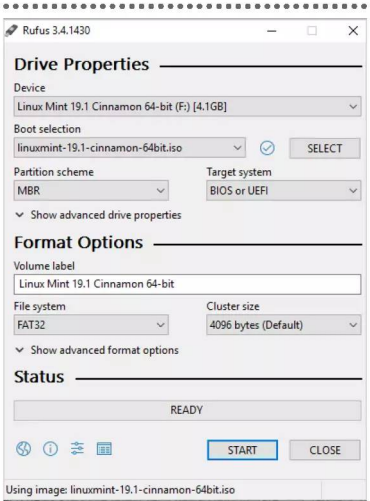
SCHRITT 5

Im nächsten Schritt werden Sie gefragt, in welchem Abbild-Modus die Linux ISO-Datei auf den USB-Stick geschrieben werden soll. Beide Methoden funktionieren in unterschiedlichen Situationen, aber im Allgemeinen ist der empfohlene ISO-Abbild-Modus populärer. Stellen Sie sicher, dass dieser Modus vorab gewählt ist, und klicken Sie auf OK.



SCHRITT 6

Die Linux ISO-Datei wird nun auf den USB-Stick übertragen. Dies sollte nicht allzu lange dauern, es hängt halt von der Geschwindigkeit des USB-Sticks und des PCs ab. Evtl. öffnet Rufus während des Vorgangs automatisch das USB-Laufwerk im Windows Explorer. Dies ist kein Grund zur Sorge und Sie können es minimieren oder schließen. Klicken Sie nach Beendigung des Vorgangs auf „Schließen".



Linux-Installation auf einem PC

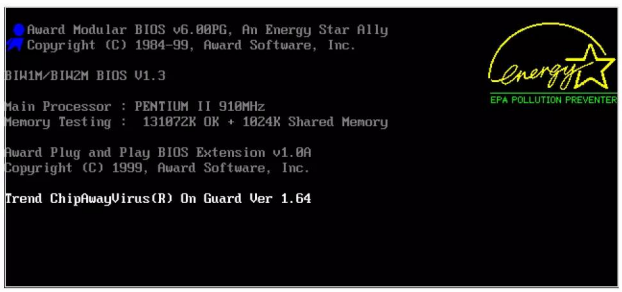
Die meisten Linux-Distributionen sind Live-Umgebungen. Das bedeutet, dass Sie direkt von der soeben erstellten DVD oder dem USB-Stick in eine voll funktionsfähige Distribution booten können. Hier erfahren Sie, wie das funktioniert.

UEFI BIOS

Die Unified Extensible Firmware Interface (UEFI) dient zur Identifikation von Hardware und zum Schutz des PCs während des Bootvorgangs. Sie ersetzt das traditionelle BIOS, kann aber bei der Installation von Linux Probleme verursachen.

SCHRITT 1

Legen Sie Ihre DVD ein bzw. schließen Sie Ihren USB-Stick an und, falls Sie es noch nicht getan haben, fahren Sie Windows herunter. Wir benutzen einen USB-Stick, aber der Vorgang ist praktisch für beide Boot-Medien identisch. Starten Sie den PC und drücken Sie bei Aufforderung die entsprechende Taste für das BIOS bzw. SETUP, z. B. **F2**, **Entf** oder sogar **F12**.



SCHRITT 2

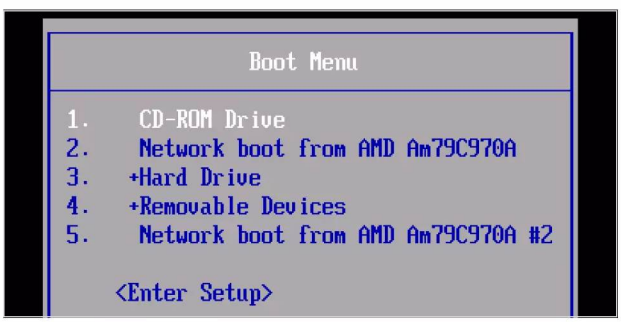
Es gibt verschiedene Versionen des UEFI BIOS, es ist daher unmöglich, alle abzudecken. Suchen Sie nach dem Abschnitt, der die **Bootsequenz** oder den **Boot-Modus** beschreibt. Hier haben Sie die Möglichkeit, UEFI auszuschalten und **Legacy** auszuwählen oder die **Secure Boot**-Funktion zu deaktivieren. Die meisten Distributionen funktionieren zwar mit UEFI, das Hochfahren kann aber schwierig sein.

SCHRITT 3

Nachdem UEFI in den Legacy-Modus gesetzt wurde, gibt es nun zwei Möglichkeiten, mit denen die Live-Umgebung hochgefahren werden kann. Die Erste ist über das BIOS, in dem Sie sich bereits befinden. Suchen Sie nach der Bootsequenz und ändern Sie die Originaleinstellung des ersten Boot-Geräts, die meist Internal HDD oder ähnlich ist, in: **USB Storage Device** für die USB-Option oder **DVD Drive** für die DVD-Option.

SCHRITT 4

Alternativ können Sie das Boot-Optionsmenü verwenden. Hier erhalten Sie durch Drücken von **F12** (bzw. der entsprechenden Taste) eine Liste von Boot-Medienoptionen, aus denen Sie die passenden Boot-Medien auswählen können. Sie können das BIOS nun speichern und verlassen, indem Sie zur Option **Save & Exit** navigieren und **Save Changes** and **Exit** wählen.



LINUX INSTALLIEREN

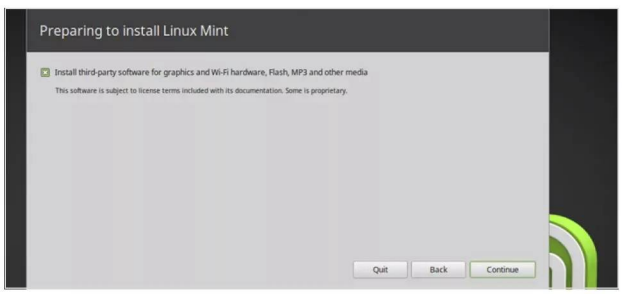
Nachdem die Live-Umgebung gestartet wurde, können Sie die Distribution auf Ihrem Computer installieren. Schauen Sie sich um und wenn Sie so weit sind, suchen Sie auf dem Desktop nach der Installationsoption.

SCHRITT 1

Vorausgesetzt Sie haben eine aktive Internetverbindung und sind in der Live-Umgebung, können Sie den Installationsvorgang durch Doppelklicken des Desktop-Symbols **Linux Mint installieren** einleiten. Andere Distributionen zeigen natürlich ihr eigenes Symbol an, aber der Vorgang ist gleich. Klicken Sie anschließend auf **Weiter**.

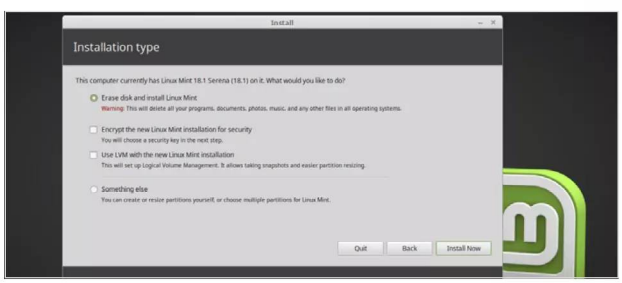
SCHRITT 2

Die Installation der meisten Linux-Distributionen verläuft ähnlich, bei einigen erscheinen während der Installation jedoch unterschiedliche Fragen. In der Regel sind sie nicht allzu schwierig oder technisch, aber einige, z. B. die Frage nach dem Installieren von Drittanbieter-Software, können verwirrend sein. Sie können auf **Weiter** klicken, aber wenn Sie unsicher sind, halten Sie ein mit dem Internet verbundenes Gerät für Fragen bereit.



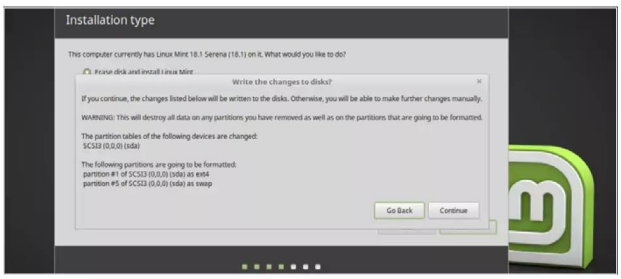
SCHRITT 3

Bei der Installation eines neuen Betriebssystems wird empfohlen, das alte zu löschen und durch das neue zu ersetzen. Wenn Sie diesen Punkt des Installationsvorgangs erreichen, stellen Sie sicher, dass die Option „Festplatte löschen und Linux installieren ...“ ausgewählt ist. HINWEIS: **Windows 10 wird dadurch vollständig gelöscht. Stellen Sie daher sicher, dass alle persönlichen Dateien und Daten gesichert sind.**



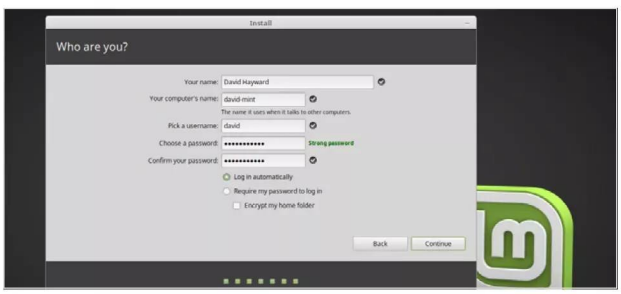
SCHRITT 4

Vor der Installation werden Sie gefragt, ob Sie Ihre Festplatte tatsächlich löschen möchten. Dies ist Ihre letzte Gelegenheit, um es abzubrechen. Wenn Sie sicher sind, dass alles gelöscht werden kann und Sie mit Linux Mint neu anfangen möchten, klicken Sie auf **Weiter**. Wenn Sie Ihre Daten sichern möchten, entfernen Sie die Linux-DVD bzw. den USB-Stick, machen Sie einen Neustart, sichern Sie Ihre Daten und beginnen Sie von vorne.



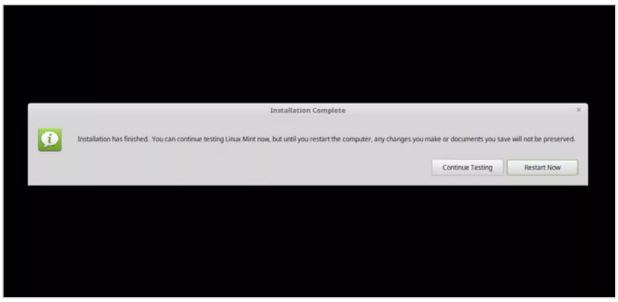
SCHRITT 5

Letztendlich müssen Sie Ihren Linux-Benutzernamen und Ihr Passwort einrichten. Geben Sie zuerst Ihren Namen ein, dann den Namen des Rechners, d. h. der Name, mit dem er vom Netzwerk erkannt wird. Wählen Sie anschließend einen Benutzernamen und ein gutes Passwort aus. Sie können die Option **Automatische Anmeldung** aktivieren, Ihre persönlichen Daten sollten Sie aber verschlüsselt lassen.



SCHRITT 6

Der Installationsvorgang kann schnell ablaufen und möglicherweise werden weitere Fragen gestellt. Sie werden anschließend gefragt, ob Sie die Live-Umgebung weiter testen oder einen Neustart ausführen möchten, um das neu installierte Betriebssystem zu nutzen. Wenn Sie mit Linux beginnen möchten, klicken Sie auf **Jetzt neu starten**.



Installation einer virtuellen Umgebung

Eine virtuelle Umgebung ist ein simuliertes Computersystem. Auf einer virtuellen Maschine kann ein Standard-PC nachgeahmt und ein gesamtes Betriebssystem installiert werden, ohne dass dabei das auf Ihrem Computer installierte Betriebssystem beeinträchtigt wird. Sie bietet eine tolle Möglichkeit, um Linux zu testen.

VIRTUELLE MASCHINE

Eine virtuelle Maschine (VM) beansprucht auf Ihrem Computer Arbeitsspeicher und Festplattenspeicher, nutzt den Prozessor usw. Stellen Sie daher sicher, dass vor Beginn von allem genug vorhanden ist.

SCHRITT 1

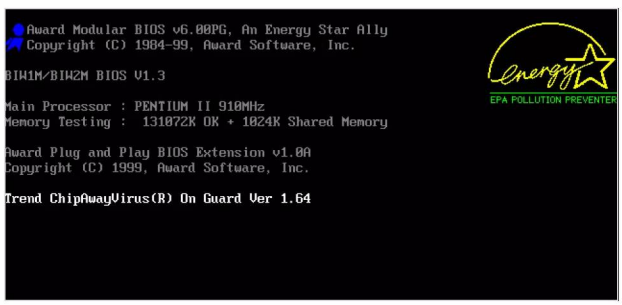
Wir verwenden hier VirtualBox, da sie eine der einfachsten virtuellen Umgebungen für den Einstieg ist. Gehen Sie auf **www.virtualbox.org** und klicken Sie auf **Download VirtualBox**. Dadurch gelangen Sie auf die eigentliche Download-Seite. Suchen Sie nach dem korrekten Host für Ihr System, Windows oder Mac, und klicken Sie auf den Download-Link.

SCHRITT 2

Suchen Sie ebenfalls auf der VirtualBox-Download-Seite nach dem Link fürs VirtualBox Extension Pack. Dieses Erweiterungspaket unterstützt USB-Geräte und enthält viele weitere Extras, die helfen können, die virtuelle Maschinenumgebung zu einer akkurateren Nachahmung eines „echten" Computers zu machen.

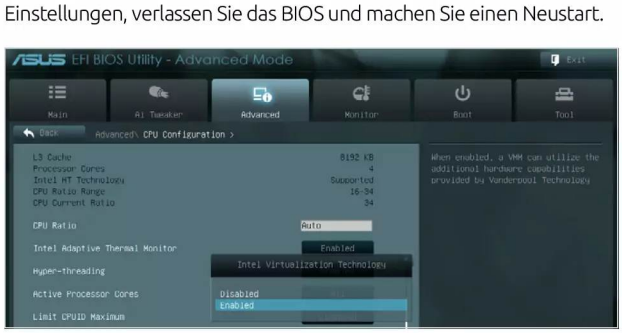
SCHRITT 3

Nachdem Sie die korrekten Pakete heruntergeladen haben, müssen Sie vor der Installation sicherstellen, dass auf dem Computer auch eine virtuelle Maschine laufen kann. Machen Sie dazu einen Neustart und gehen Sie ins BIOS. Drücken Sie während des Hochfahrens **Entf**, **F2** bzw. die entsprechende Taste, mit der das Set-up aufgerufen wird.



SCHRITT 4

Da jedes BIOS anders ausgelegt ist, ist es sehr schwierig zu bestimmen, wo man in jedem einzelnen Beispiel nachschauen muss. Als Daumenregel gilt jedoch, nach der Intel Virtualisation Technology bzw. einfach nur nach Virtualisation zu suchen. In der Regel findet man dies im erweiterten BIOS-Bereich. Wenn Sie es gefunden haben, aktivieren Sie es, speichern Sie die Einstellungen, verlassen Sie das BIOS und machen Sie einen Neustart.

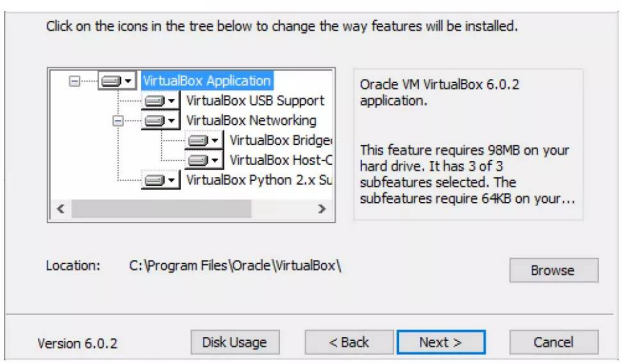


SCHRITT 5

Wenn der Computer hochgefahren ist, suchen Sie nach der heruntergeladenen VirtualBox-Anwendung und beginnen Sie die Installation per Doppelklick. Um fortzufahren, klicken Sie auf „Next".

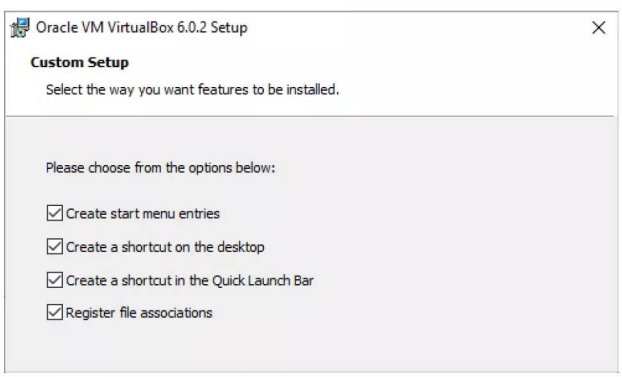
SCHRITT 6

Der standardmäßige Installationsspeicherort der VirtualBox sollte akzeptabel sein, wenn Sie aber einen anderen Speicherort nehmen möchten, klicken Sie auf „Browse" und ändern Sie den Installationsordner. Stellen Sie auch sicher, dass alle Symbole im VirtualBox-Verzeichnisbaum ausgewählt sind und keines ein rotes X daneben hat. Klicken Sie anschließend auf „Next".



SCHRITT 7

Wenn Sie möchten, können Sie die Standardeinstellungen in diesem Abschnitt lassen, wie sie sind. Sie machen den Umgang mit virtuellen Maschinen einfacher, besonders wenn es sich um heruntergeladene virtuelle Maschinen handelt. Klicken Sie erneut auf „Next", um fortzufahren.



SCHRITT 8

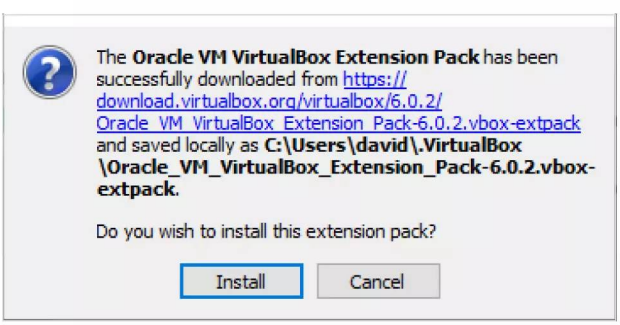
Beim Installieren von VirtualBox wird Ihre Netzwerkverbindung kurzfristig unterbrochen. Dies liegt daran, dass VirtualBox eine verknüpfte, virtuelle Netzwerkverbindung herstellt, damit jede virtuelle Maschine über die bereits bestehende Netzwerkverbindung des Computers auf das Internet und die Ressourcen Ihres Heimnetzwerks zugreifen kann. Klicken Sie auf „Yes" und dann auf „Install", um die Installation zu beginnen.

SCHRITT 9

Eventuell fragt Windows an, ob Sie Sicherheitsbenachrichtigungen akzeptieren. Klicken Sie auf „Ja". Klicken Sie auch bei der Anfrage, ob Sie der Installation von Oracle vertrauen auf „Ja", und akzeptieren Sie die Installation der VirtualBox-Anwendung. Wenn die Installation fertig ist, klicken Sie auf „Finish", um VirtualBox zu starten.

SCHRITT 10

Sie können nun das VirtualBox-Erweiterungspaket installieren. Suchen Sie nach dem heruntergeladenen Add-on und machen Sie einen Doppelklick darauf. Evtl. gibt es eine kurze Unterbrechung, während VirtualBox das Paket analysiert, aber letztendlich werden Sie die Meldung erhalten, dass Sie es installieren können. Klicken Sie auf „Install", scrollen Sie auf der nächsten Seite herunter und akzeptieren Sie die Vereinbarungen.



Linux-Installation in einer virtuellen Umgebung

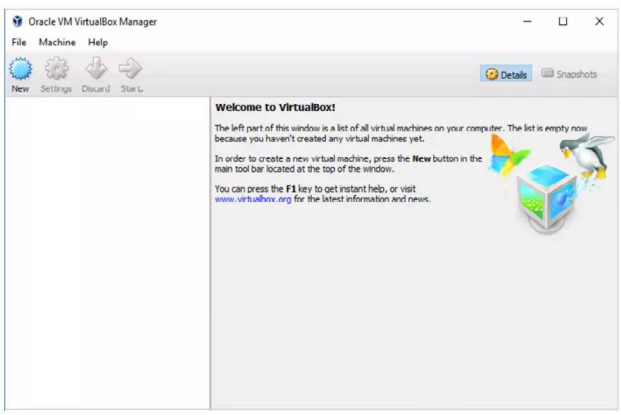
Nachdem Oracles VirtualBox nun eingerichtet und betriebsbereit ist, erstellen wir nun die virtuelle Maschinenumgebung, in der Linux installiert wird. Dieser Vorgang hat keinen Einfluss auf Ihr derzeit installiertes Betriebssystem.

VIRTUELLE MASCHINE ERSTELLEN

Zum Erstellen einer virtuellen Maschine stehen uns viele Möglichkeiten zur Auswahl. Wir wollen zunächst jedoch eine leistungsfähige virtuelle Maschine einrichten, die Mint Cinnamon ausführen kann.

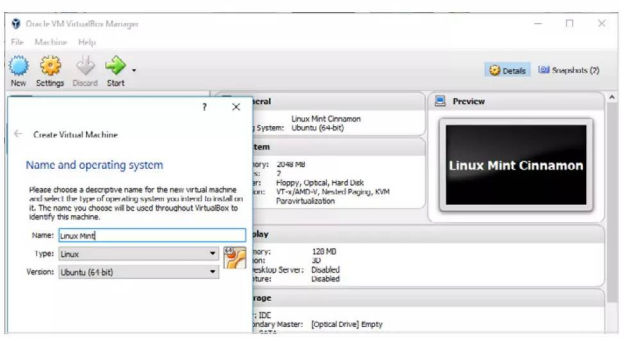
SCHRITT 1

Klicken Sie bei geöffneter VirtualBox oben links auf das Symbol **Neu**. Dadurch wird der neue VM-Assistent geöffnet.



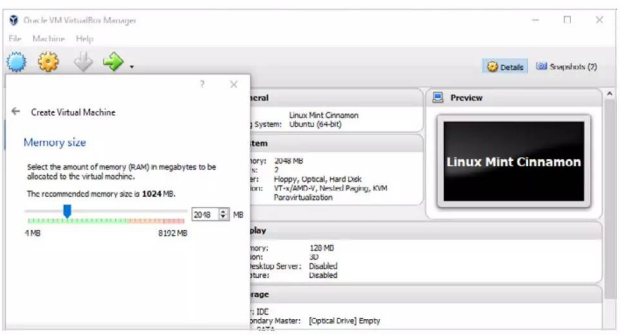
SCHRITT 2

Geben Sie im Namensfeld **Linux Mint** ein. VirtualBox sollte als Typ automatisch Linux und als Version Ubuntu (64-Bit) wählen. Wenn nicht, wählen Sie aus dem Drop-down-Menü die korrekten Einstellungen aus. Denken Sie daran, dass Mint auf Ubuntu basiert. Wenn Sie fertig sind, klicken Sie auf **Weiter**.



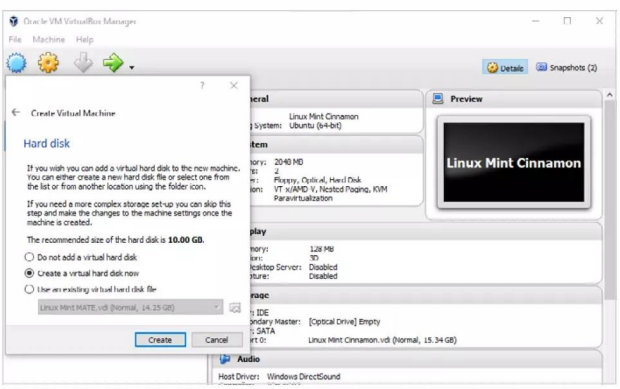
SCHRITT 3

Der nächste Abschnitt definiert die an die VM zugewiesene Größe des Systemspeichers (RAM). Beachten Sie, dass diese Speichermenge von der verfügbaren Speichermenge, die in Ihrem Computer installiert ist, abgezogen wird, geben Sie der VM daher nicht zu viel. Unsere Speichergröße beträgt z. B. 8 GB und wir geben 2 GB an die VM. Klicken Sie anschließend auf **Weiter**.



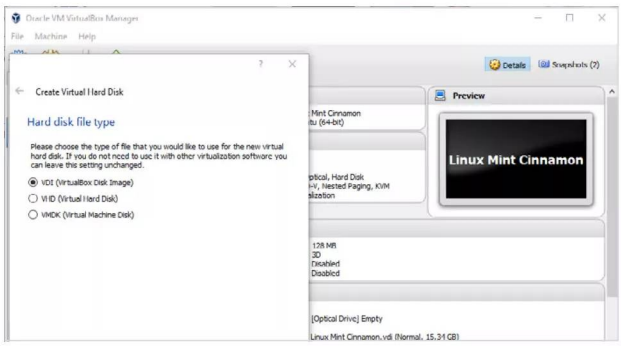
SCHRITT 4

In diesem Abschnitt erstellen Sie die virtuelle Festplatte, auf der die virtuelle Maschine Mint installieren wird. Wir verwenden die Standardoption „Festplatte erzeugen". Klicken Sie auf **Erzeugen**, um fortzufahren.



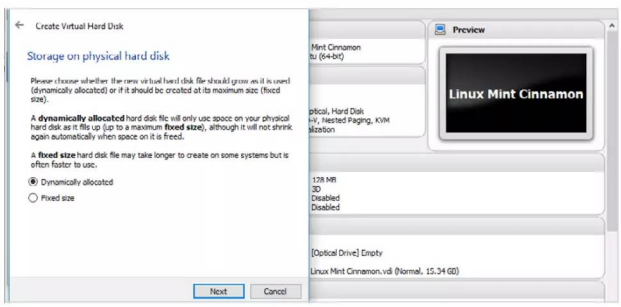
SCHRITT 5

Im darauffolgenden Fenster müssen Sie den Typ der neuen virtuellen Festplatte angeben. Benutzen Sie in diesem Fall die Standardoption VDI (VirtualBox Disk Image), da die anderen oft dazu verwendet werden, virtuelle Maschinen von einer VM-Anwendung zur anderen zu verschieben. Stellen Sie sicher, dass VDI ausgewählt ist, und klicken Sie auf **Weiter**.



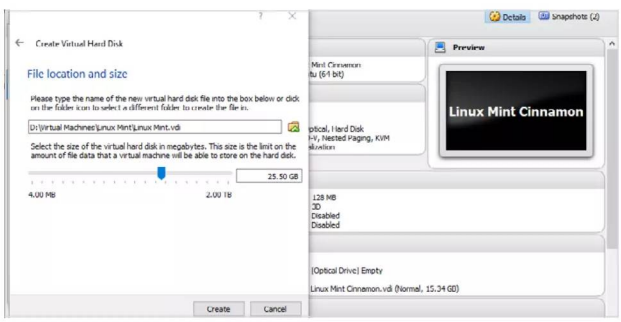
SCHRITT 6

Die Frage, ob man sich für eine dynamisch allozierte virtuelle Festplatte oder eine mit fester Größe entscheiden soll, kann für den Einsteiger etwas verwirrend sein. Im Grunde ist eine dynamisch allozierte virtuelle Festplatte eine flexiblere Option zur Speicherverwaltung. Sie nimmt zu Beginn auch nicht allzu viel Speicherplatz auf Ihrer physischen Festplatte ein. Stellen Sie sicher, dass **dynamisch alloziert** ausgewählt ist, und klicken Sie auf **Weiter**.



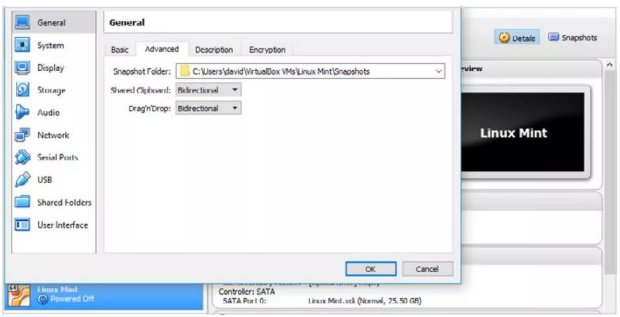
SCHRITT 7

Die virtuelle Festplatte besteht aus einem einzelnen Ordner, dessen Größe Sie in diesem Abschnitt bestimmen. Stellen Sie sicher, dass der Speicherort Ihrer virtuellen Festplatte auf Ihrem Computer genügend freien Speicherplatz hat. Wir haben z. B. eine größere Speicheroption auf unserem D:\Laufwerk verwendet, haben es Linux Mint benannt und der virtuellen Festplatte Speicherplatz in Höhe von 25,5 GB zugewiesen.



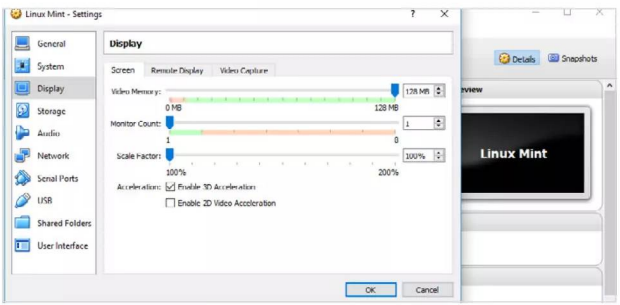
SCHRITT 8

Nachdem auf „Erzeugen" geklickt wurde, ist die Erstinstallation der virtuellen Maschine abgeschlossen. Sie sollten nun die neu erstellte VM innerhalb der Virtual-Box-Anwendung sehen. Bevor Sie loslegen, klicken Sie auf **Ändern** und wählen Sie unter **Allgemein** den Tab **Erweitert**. Wählen Sie mithilfe des Drop-down-Menüs **bidirektional**, sowohl für Gemeinsame Zwischenablage als auch Drag'n'Drop.



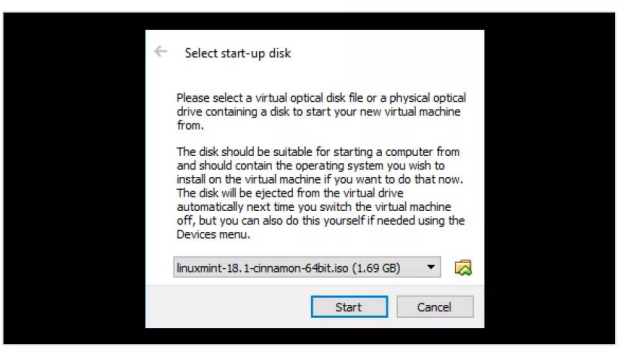
SCHRITT 9

Klicken Sie nun unter **System** auf den Tab **Prozessor**. Weisen Sie, abhängig von Ihrer CPU, so viele Einheiten wie Sie können zu ohne jedoch Ihr Host-System zu schädigen; wir haben uns für zwei CPUs entschieden. Gehen Sie in den Bereich **Anzeige**, schieben Sie den Regler für den Grafikspeicher bis zum Maximum und aktivieren Sie die Option **3D-Beschleunigung aktivieren**. Klicken Sie auf OK, um die neuen Einstellungen zu übernehmen.



SCHRITT 10

Klicken Sie auf **Starten** und suchen Sie mithilfe der Explorer-Taste (Ordner mit grünem Pfeil) im Startup-Disk-Fenster die heruntergeladene ISO-Datei von Mint. Klicken Sie auf **Starten**, um die VM in der Linux Mint Live-Umgebung hochzufahren. Sie können Linux nun gemäß den Standardanforderungen für die PC-Installation installieren.



Programmieren unter Linux – auf geht's!

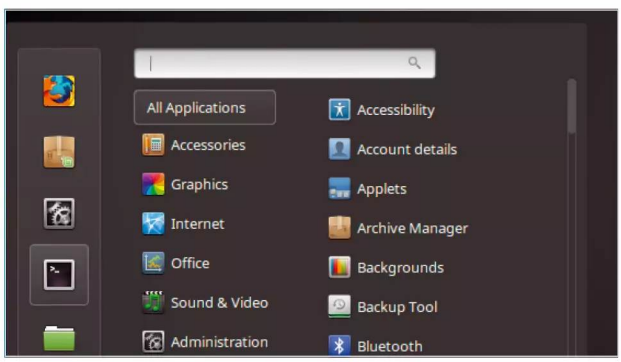
Das Programmieren in Linux erfolgt meistens im Terminal bzw. in der Befehlszeile. Das Terminal ist eine äußerst leistungsfähige Umgebung und bevor Sie mit dem Programmieren beginnen, sollten Sie wissen, wie es funktioniert.

ÜBERNEHMEN SIE DAS KOMMANDO

Das Programmieren über die Befehlszeile, die den Kern von Linux bildet, nennt sich Scripting. Es handelt sich dabei um eigenständige Programme, die für eine Ausführung im Terminal erstellt werden.

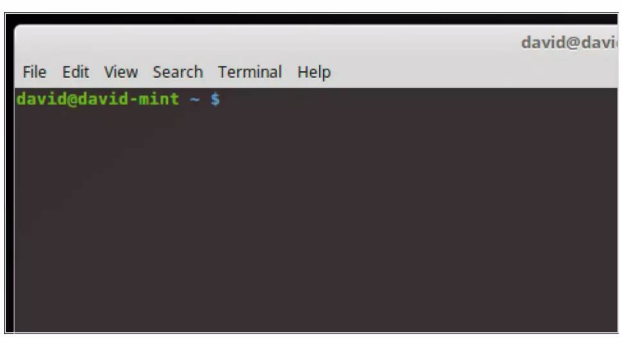
SCHRITT 1

Das Terminal ist der Ort, an dem Sie Ihre Reise mit Linux beginnen. Dies geschieht über die Befehlszeile und den darin erstellten Skripte. In Linux Mint kann das Terminal im Menü durch Anklicken seines Symbols gestartet werden. Alternativ können Sie im Suchfeld „Terminal" eingeben.



SCHRITT 2

Das Terminal bietet Zugriff auf die Linux Mint Shell, die sich BASH nennt; damit können Sie auf das zugrunde liegende Betriebssystem zugreifen, was Scripting zu solch einer leistungsstarken Sprache macht. Alles in Mint und Linux als Ganzes, einschließlich des Desktops und der GUI, ist ein Modul, das von der Befehlszeile aus läuft.

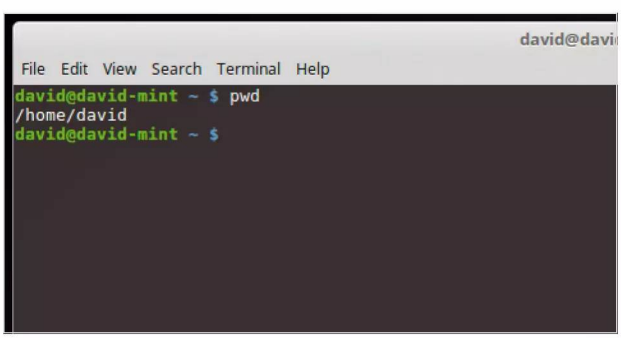


SCHRITT 3

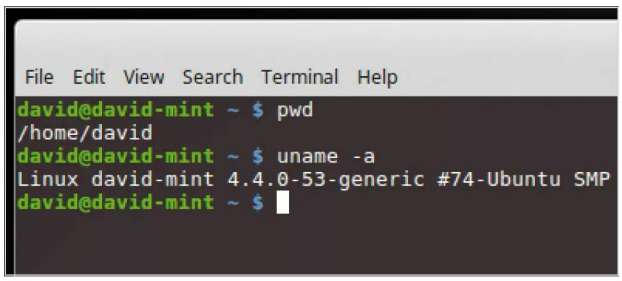
Im Terminal sehen Sie zu Beginn Ihren Anmeldenamen gefolgt vom Namen des Rechners, den Sie während der ersten Installation von Mint eingegeben haben. Die Zeile endet mit dem aktuellen Ordnernamen; dieser besteht zunächst nur aus der Tilde ~, die für Ihr Home-Verzeichnis steht.

SCHRITT 4

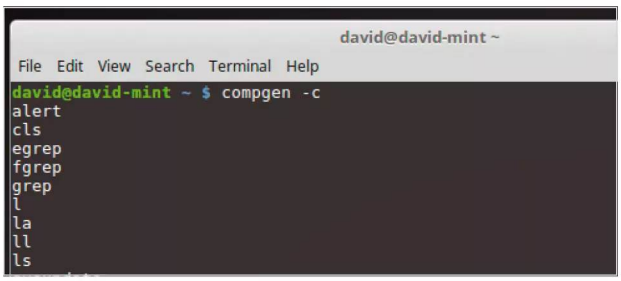
Der blinkende Cursor am Zeilenende ist die Stelle, an der Sie Ihre textbasierten Befehle eingeben. Beginnen Sie, indem Sie mit einem einfachen Befehl etwas experimentieren: Print Working Directory (pwd); damit wird das aktuelle Verzeichnis auf dem Bildschirm ausgegeben. Geben Sie pwd ein und drücken Sie die Eingabetaste.



SCHRITT 5 Alle Befehle funktionieren auf die gleiche Weise. Sie geben den Befehl ein, einschließlich aller Parameter, mit denen der Befehl erweitert wird, und drücken zur Ausführung die Eingabetaste. Geben Sie `uname -a` ein und drücken Sie die Eingabetaste. Damit werden Systeminformationen zu Linux Mint angezeigt. Beim Scripting können Sie alle Befehlszeilenbefehle von Linux in Ihren eigenen Skripten anwenden.



SCHRITT 6 Die Liste der verfügbaren Linux-Befehle ist riesig. Einige zeigen lediglich das aktuelle Arbeitsverzeichnis an, während andere das gesamte System in einem Moment löschen können. Das Lernen der Befehle gehört beim Scripting dazu. Ein falscher Befehl, und der Computer könnte gelöscht werden. Geben Sie `compgen -c` ein, um die verfügbaren Befehle anzuzeigen.

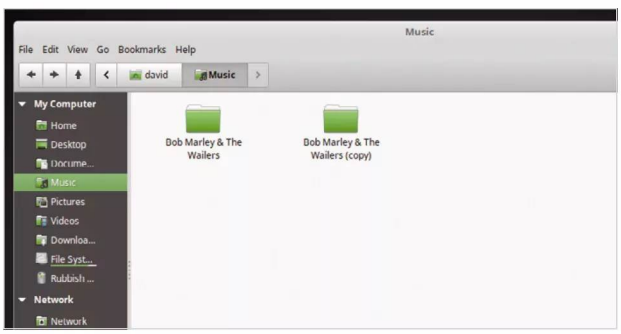


AUF NUMMER SICHER GEHEN

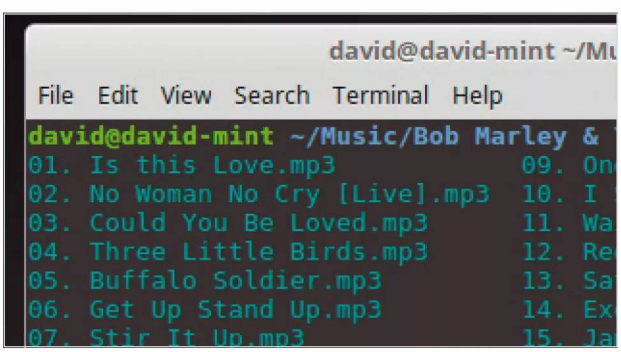
Ein Mythos im Internet besagt, dass ein Mitarbeiter von Disney Pixar den Animationsfilm Toy Story beinahe ruiniert hätte, indem er versehentlich den falschen Linux-Befehl eingab und das gesamte System löschte, in dem der Film gespeichert war.

SCHRITT 1 Wenn Sie für die Arbeit mit dem System auf das Terminal zugreifen, umgehen Sie die GUI-Desktop-Methode. Das Terminal ist eine weitaus leistungsfähigere Umgebung als der Desktop, der mehrere Sicherheitsvorkehrungen hat, für den Fall, dass Sie versehentlich Ihre gesamte Arbeit löschen, z. B. einen Papierkorb, über den Sie gelöschte Dateien wiederherstellen können.

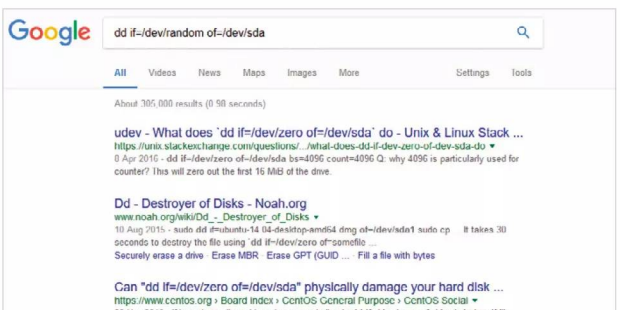
SCHRITT 3 Es lohnt sich daher immer, im Terminal auf zwei Arten zu arbeiten. Machen Sie zunächst im Desktop regelmäßig Sicherungskopien der Verzeichnisse, mit denen Sie im Terminal arbeiten. Sollte etwas schief gehen, haben Sie somit schnell ein Backup zur Hand.



SCHRITT 2 Im Terminal gibt es diesen Luxus jedoch nicht. Wenn Sie über das Terminal auf ein Verzeichnis mit Dateien zugreifen und dann den Befehl `rm *.*` eingeben, werden alle Dateien in diesem Verzeichnis sofort gelöscht. Sie erscheinen auch nicht im Papierkorb – sie sind endgültig weg.



SCHRITT 4 Überprüfen Sie auch einen im Internet gefundenen Befehl, bevor Sie ihn eingeben. Wenn Sie z. B. den Befehl `sudo dd if=/dev/random of=/dev/sda` in einem Skript verwenden, werden Sie die Aktion bald bereuen, da der Befehl die gesamte Festplatte löscht und mit zufälligen Daten füllt. Googeln Sie daher einen Befehl, um zu sehen, was er bezweckt.



Bash-Skripte erstellen – Teil 1

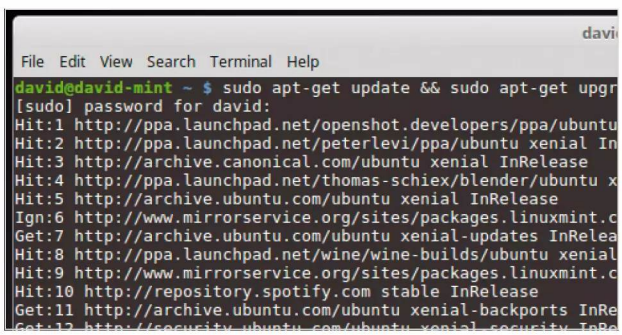
Wenn Sie mit Linux Mint erfahrener geworden sind, wollen Sie sicher Ihre eigenen automatisierten Aufgaben und Programme erstellen. Dies sind Bash-Shell-Skripte, die auf die gleiche Weise wie eine DOS Batch-Datei bzw. jede andere Programmiersprache funktionieren.

LOS GEHT'S MIT SCRIPTING

Ein Bash-Skript besteht aus einer Reihe von Befehlen, die Mint durchläuft, um eine bestimmte Aufgabe auszuführen. Je nach Situation können die Befehle einfach oder komplex sein.

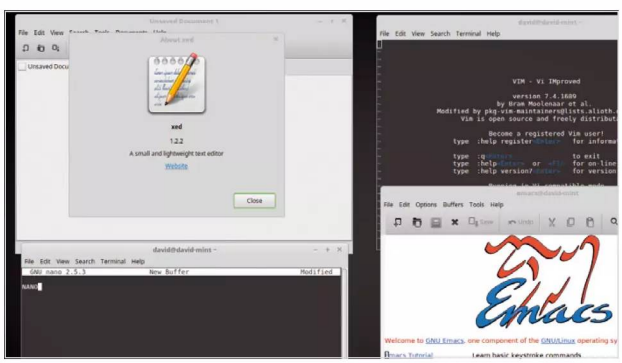
SCHRITT 1

Auf den nächsten Seiten werden Sie mit dem Terminal und einem Texteditor arbeiten. Es gibt Alternativen zum Texteditor, die wir in uns in Kürze ansehen werden, aber der Einfachheit halber nehmen wir für unsere Beispiele Xed. Bevor Sie beginnen, schauen Sie, ob es Aktualisierungen gibt: `sudo apt-get update && sudo apt-get upgrade`.



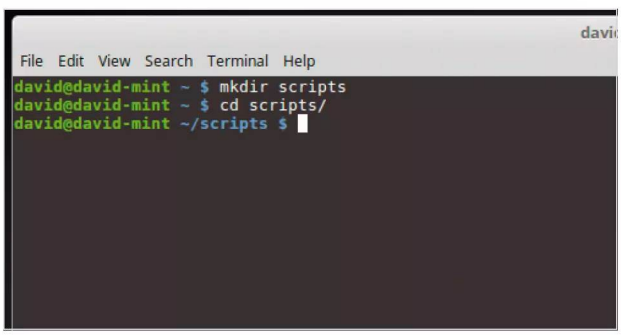
SCHRITT 2

Es gibt mehrere Texteditoren zum Erstellen von Bash-Skripten: Xed, Vi, Nano, Vim, GNU Emacs usw. Am Ende kommt es halt auf den persönlichen Geschmack an. Wir verwenden Xed lediglich, um die Skripte in den Abbildungen lesbarer zu machen.



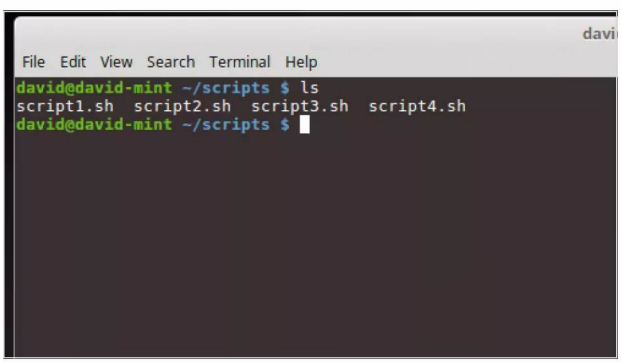
SCHRITT 3

Bevor Sie mit dem Schreiben von Skripten beginnen, müssen Sie ein Verzeichnis erstellen, in dem Sie Ihre Skripte aufbewahren. Geben Sie dazu `mkdir scripts` ein und wechseln Sie mit `cd scripts/` zum Verzeichnis. Dies wird Ihr Arbeitsverzeichnis sein, in das Sie auch Unterverzeichnisse anlegen können, falls Sie für jedes Skript ein eigenes Verzeichnis möchten.



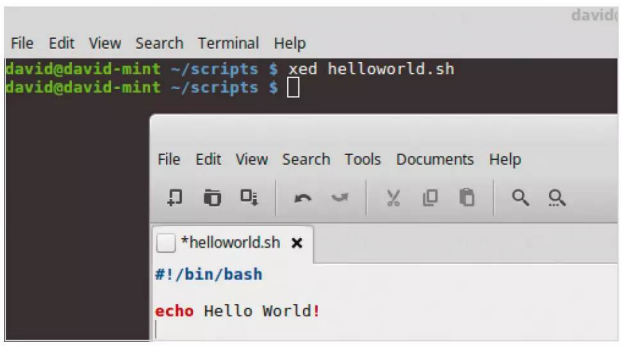
SCHRITT 4

Windows-Nutzer werden wissen, dass eine Batch-Datei eine .BAT-Dateierweiterung haben muss, damit sie funktionieren und das enthaltene Programm ausführen kann. Linux ist ein Betriebssystem ohne Erweiterungen, aber es besteht die Übereinkunft, Skripten eine .sh-Erweiterung zu geben.



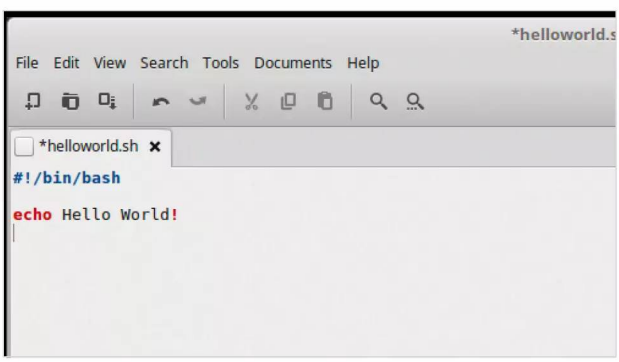
SCHRITT 5

Wir beginnen mit einem einfachen Skript, mit dem etwas im Terminal ausgegeben wird. Geben Sie `xed helloworld.sh` ein. Dadurch wird Xed gestartet und eine Datei namens helloworld.sh. erstellt. Geben Sie in Xed Folgendes ein: `#!/bin/bash` und dann in einer neuen Zeile: `echo Hello World!`.



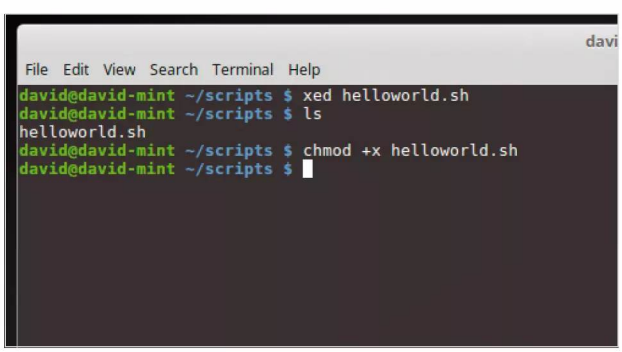
SCHRITT 6

Die Zeile `#!/bin/bash` sagt dem System, welche Shell Sie verwenden werden, in diesem Fall ist es die Bash-Shell. Das Rautenzeichen (#) kennzeichnet einen Kommentar, der vom System ignoriert wird. Das Ausführungszeichen bedeutet, dass der Kommentar umgangen wird, und zwingt das Skript, die Zeile als Befehl auszuführen. Dies ist auch als Hash-Bang bekannt.



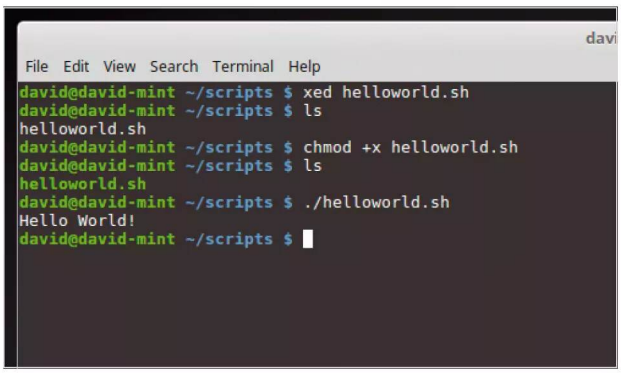
SCHRITT 7

Speichern Sie die Datei (File > Save), schließen Sie sie und kehren Sie ins Terminal zurück. Die Eingabe von `ls` zeigt das Skript im Verzeichnis an. Damit jedes Skript ausgeführt werden kann, müssen Sie mit `chmod +x helloworld.sh` seine Zugriffsrechte ändern. Sie müssen dies mit jedem Skript machen, das Sie erstellen.



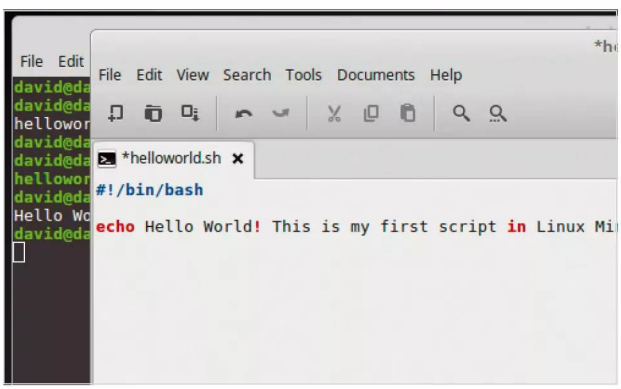
SCHRITT 8

Wenn Sie erneut `ls` eingeben, werden Sie feststellen, dass das helloworld.sh-Skript nun grün anstatt weiß ist, was bedeutet, dass es eine ausführbare Datei ist. Um das Skript auszuführen, d. h. um es zu veranlassen, die darin eingegebenen Dinge auszuführen, geben Sie `./helloworld.sh` ein.



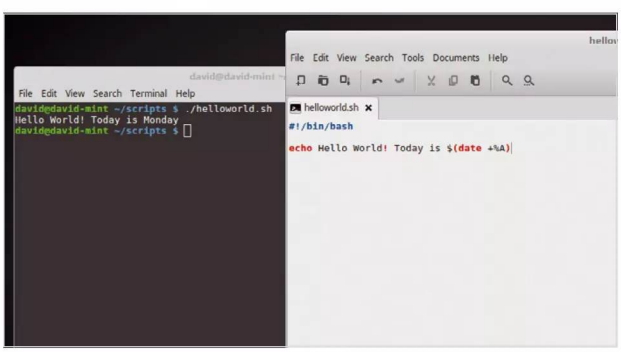
SCHRITT 9

Sie sind zwar nicht sehr spannend, aber die Wörter „Hello World!" sollten nun im Terminal angezeigt werden. Der echo-Befehl sorgt dafür, dass die nachstehenden Wörter im Terminal ausgegeben werden; Sie können ihn auch dazu veranlassen, an andere Quellen auszugeben.



SCHRITT 10

`echo` ist mit dem alten BASIC-Ausgabebefehl PRINT vergleichbar. Er zeigt Text, Zahlen oder jegliche Variablen an, die im System gespeichert sind, z. B. das aktuelle Systemdatum. Probieren Sie das folgende Beispiel aus: `echo Hello World! Today is $(date +%A)`. „$(date +%A)" ruft die Systemvariable ab, die den aktuellen Wochentag speichert.



Bash-Skripte erstellen – Teil 2

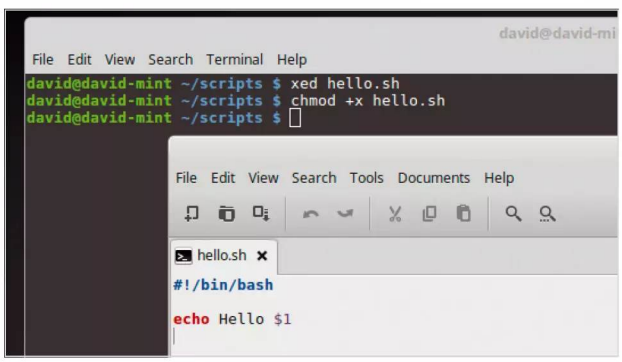
Im letzten Abschnitt haben wir uns mit dem Erstellen Ihres ersten Bash-Skripts, Hello World, und dem Hinzufügen einer Variablen befasst. Wir werden das Ganze nun erweitern und uns anschauen, was Sie mit Ihren eigenen Variablen machen können.

VARIABLEN

Genau wie in jeder anderen Programmiersprache kann ein Bash-Skript bestimmte Variablen speichern und abrufen. Dabei kann es sich um allgemeine oder vom Benutzer erstellte Variablen handeln.

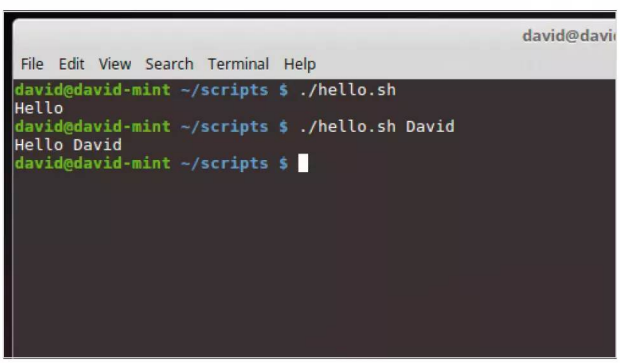
SCHRITT 1

Beginnen Sie, indem Sie ein neues Skript namens hello.sh erstellen: `xed hello.sh`. Geben Sie darin `#!/bin/bash` und dann `echo Hello $1` ein. Speichern Sie die Datei und schließen Sie Xed. Machen Sie im Terminal das Skript mit `chmod +x hello.sh` ausführbar.



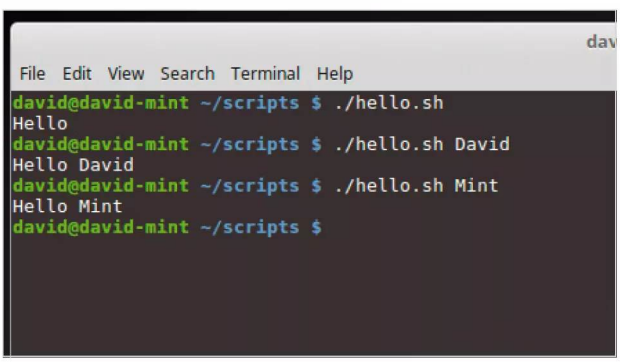
SCHRITT 2

Führen Sie das Skript nun mit `./hello.sh` aus. Wie Sie sicherlich vermutet haben, wird ein einfaches „Hello" im Terminal angezeigt. Interessant wird es jedoch, wenn Sie den Befehl nun mit einer Variablen eingeben. Probieren Sie folgendes Beispiel aus: `./hello.sh David`.



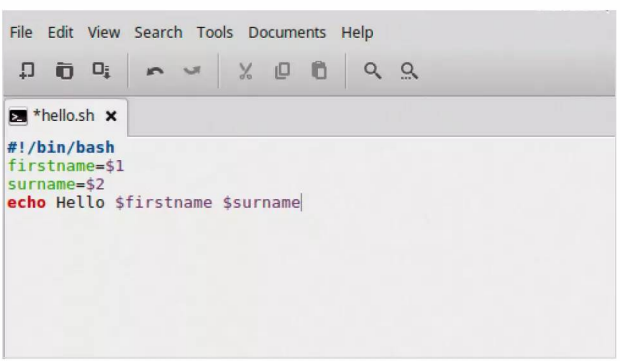
SCHRITT 3

Es wird nun „Hello David" ausgegeben. Das liegt daran, dass Bash Variablen automatisch dem Benutzer zuordnet, die dann beibehalten und ans Skript weitergeleitet werden. Die Variable „$1" enthält nun den Wert „David". Geben Sie etwas anderes ein, um die Variable zu ändern: `./hello.sh Mint`.



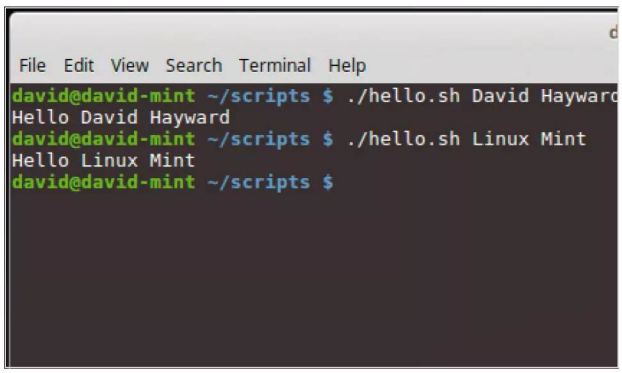
SCHRITT 4

Sie können Variablen auch umbenennen. Ändern Sie das hello.sh-Skript wie folgt: `firstname=$1, surname=$2, echo Hello $firstname $surname` (geben Sie jede Anweisung in eine neue Zeile ein). Speichern und schließen Sie das Skript und kehren Sie ins Terminal zurück.



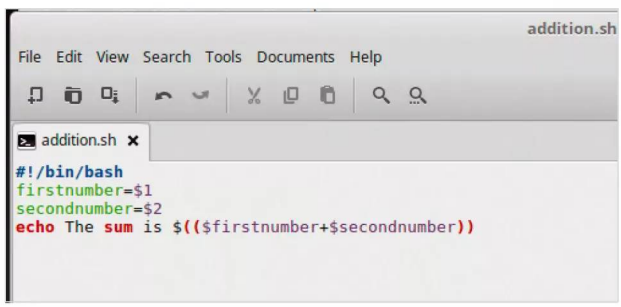
SCHRITT 5

Wenn Sie nun das Skript ausführen, haben Sie zwei benutzerdefinierte Variablen: `./hello.sh David Hayward`. Sie müssen sie natürlich gegen Ihren eigenen Namen austauschen. Momentan wird der Inhalt nur ausgegeben, wir werden daher die Nutzung der zweifachen Variablen erweitern.



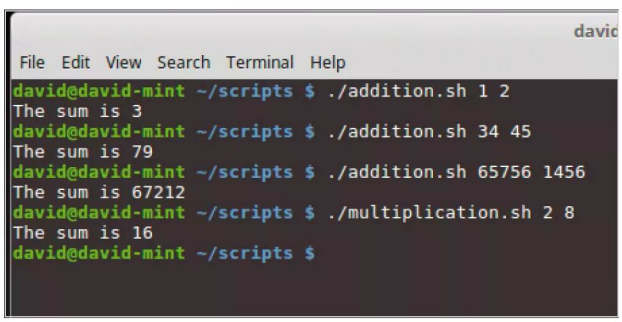
SCHRITT 6

Erstellen Sie ein neues Skript namens `addition.sh`. Benutzen Sie das gleiche Format wie beim hello.sh-Skript, ändern Sie aber die Namen der Va-riablen. Geben Sie `firstnumber` und `secondnumber`, und geben Sie mithilfe des echo-Befehls, dem Sie eine mathematische Funktion zufügen, eine einfache Rechenaufgabe aus: `echo The sum is $(($firstnumber+$secondnumber))`. Speichern Sie das Skript und machen Sie es ausführbar: `(chmod +x addition.sh)`.



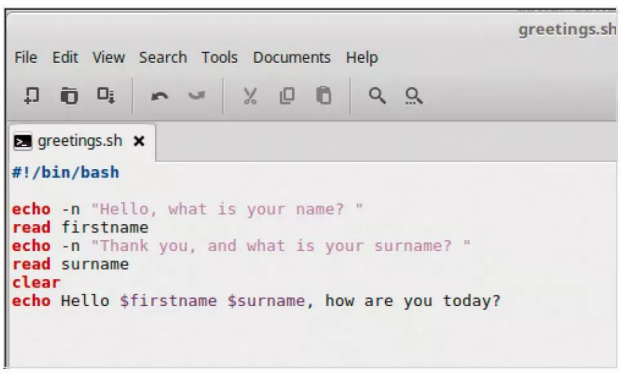
SCHRITT 7

Wenn Sie das addition.sh-Skript ausführen, können Sie zwei Nummern eingeben: `./addition.sh 1 2`. Das Ergebnis wird hoffentlich 3 sein. Probieren Sie verschiedene Nummern aus und schauen Sie, was passiert. Versuchen Sie auch, das Skript zu ändern, indem Sie Multiplikationen und Subtraktionen einfügen und das Skript dementsprechend umbenennen.



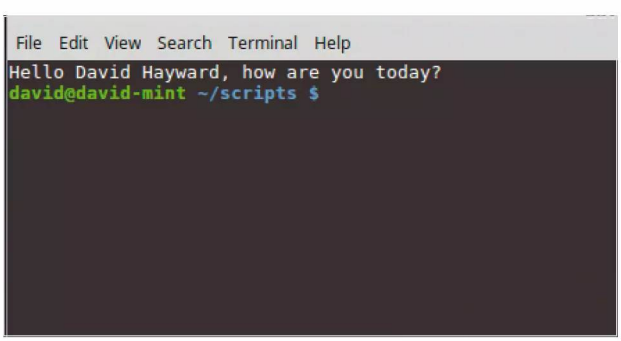
SCHRITT 8

Sie können das Ganze weiter ausbauen. Erstellen Sie ein neues Skript namens `greetings.sh`. Geben Sie das Skript wie in der Abbildung zu sehen ist ein, speichern Sie es und machen Sie es mit dem `chmod`-Befehl ausführbar. Wie Sie sehen, hat das Skript nun ein paar neue Ergänzungen.



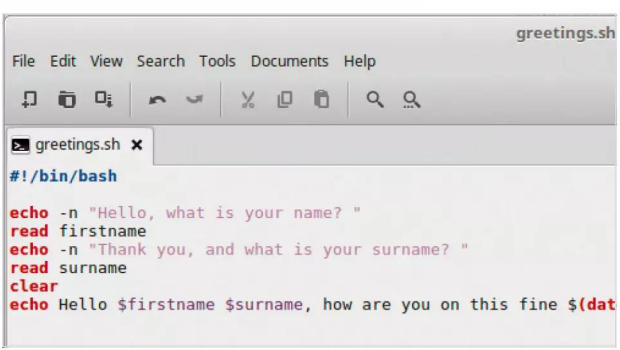
SCHRITT 9

Wir haben hier dem echo-Befehl ein `–n` angehangen, wodurch der Cursor in der gleichen Zeile wie die Frage bleiben wird, anstatt in eine neue Zeile zu gehen. Der `read`-Befehl speichert die Eingaben des Nutzers als die Variablen firstname (Vorname), und surname (Nachname), die später in der letzten echo-Zeile ausgegeben werden. Der `clear`-Befehl löscht den Bildschirm.



SCHRITT 10

Geben Sie zuletzt die Datumsvariable ein, die Sie im letzten Abschnitt verwendet haben. Ändern Sie die letzte Zeile des Skripts wie folgt: `echo Hello $firstname $surname, how are you on this fine $(date +%A)?`. Die Ausgabe sollte den aktuellen Wochentag wiedergeben, der von einer Systemvariablen abgerufen wird.



Bash-Skripte erstellen – Teil 3

Auf den vorherigen Seiten haben wir uns ein paar sehr einfache Bash-Skripte angeschaut, die Text und Eingaben des Nutzers enthielten, die gespeichert und auf dem Bildschirm ausgegeben wurden, und denen wir mithilfe des date-Befehls eine System-Variable hinzugefügt haben. Wir werden das Gelernte nun mit Schleifen kombinieren.

IF, THEN UND ELSE

Bei den meisten Programmstrukturen wird der Moment kommen, an dem die eingegebenen Befehle eine Schleife durchgehen sollen, um die Funktionalität und im Endeffekt das Programm zu verbessern.

SCHRITT 1

Wir schauen uns nun die IF-, Then- und Else-Anweisungen an, die bei korrekter Ausführung eine Reihe von Befehlen vergleichen und herausfinden, wenn (IF) etwas vorhanden ist, dann (THEN) wird etwas ausgeführt, ansonsten (ELSE) wird etwas anderes ausgeführt. Erstellen Sie ein neues Skript namens `greeting2.sh` und geben Sie den Text in der unteren Abbildung ein.

*greetings.sh

```
#!/bin/bash

echo -n "Hello, what is your name? "
read firstname
echo -n "Thank you, and what is your surname? "
read surname
clear
if [ "$firstname" == "David" ]
then echo "Awesome name, " $firstname
else echo Hello $firstname $surname, how are you on this fine
fi
```

SCHRITT 2

Greeting2.sh ist eine Kopie von greeting.sh, die aber einen kleinen Unterschied in Form einer Schleife aufweist, die bei der IF-Anweisung beginnt. Dies bedeutet, wenn (IF) die eingegebene Variable mit „David" übereinstimmt, dann (THEN) passiert Folgendes; in diesem Fall wird auf dem Bildschirm „Awesome name" ausgegeben, gefolgt von der Variablen (David).

```
david@davi
File  Edit  View  Search  Terminal  Help
Awesome name,  David
david@david-mint ~/scripts $
```

SCHRITT 3

Die nächste Zeile, ELSE, gibt an, was passiert, wenn die Variable nicht mit „David" übereinstimmt. In diesem Fall wird nur das Wort „Hello" ausgegeben. Die letzte Zeile, die FI-Anweisung, ist der Befehl, der die Schleife beendet. Fehlt bei einem IF-Befehl der FI-Befehl, erscheint eine Fehlermeldung.

```
david@david
File  Edit  View  Search  Terminal  Help
Hello Pink Floyd, how are you on this fine Wednesday?
david@david-mint ~/scripts $
```

SCHRITT 4

Sie können mit dem Skript natürlich etwas herumspielen, z. B. die Namen der Variablen ändern, die eine Antwort auslösen, oder eine Antwort erstellen, für den Fall, dass die Variablen des Vor- als auch des Nachnamens mit einer bestimmten Variable übereinstimmen.

greetings2.sh

```
#!/bin/bash

echo -n "Hello, what is your name? "
read firstname
echo -n "Thank you, and what is your surname? "
read surname
clear
if [ "$firstname" == "David" ] && [ "$surname" == "Hayward" ]
then echo "Awesome name, " $firstname $surname
else echo Hello $firstname $surname, how are you on this fine
fi
```

NOCH MEHR SCHLEIFEN

Sie können Daten mit den FOR, WHILE und UNTIL-Anweisungen durchlaufen. Diese sind praktisch, wenn für das mehrfache Benennen, Kopieren oder Ausführen eines Skripts ein Zähler benötigt wird.

SCHRITT 1

Erstellen Sie ein neues Skript namens `count.sh`. Geben Sie den Text der unteren Abbildung ein, speichern Sie das Skript und machen Sie es ausführbar. Dadurch wird die Variable „count" erstellt, die zu Beginn des Skripts null entspricht. Starten Sie dann die WHILE-Schlaufe, die, während „count" kleiner als 100 ist, den aktuellen Wert von „count" im echo-Befehl ausgibt.

count.sh

```
#!/bin/bash

count=0

while [ $count -lt 100 ];do
echo $count
let count=count+1
done
```

SCHRITT 2

Durch die Ausführung des count.sh-Skripts werden die Zahlen 0 bis 99 auf dem Terminalbildschirm aufgelistet; bei 100 wird das Skript beendet. Durch die Änderung des Skripts mit der FOR-Anweisung funktioniert es auf ähnliche Weise. Um sie in Ihrem Skript anzuwenden, fügen Sie den Text in der Abbildung in das count.sh-Skript ein.

*count.sh

File Edit View Search Tools Documents Help

*count.sh

```
#!/bin/bash

for count in {0..100}; do
echo $count
let count=count+1
done
```

SCHRITT 3

Die Ergänzung hier lautet: `for count in {0..100}; do`. Dies bedeutet, dass für (FOR) die Variable „count" die Zahlen 0 bis 100 eingezählt (count IN) werden und dann die Schleife gestartet wird. Der Rest des Skripts ist gleich. Führen Sie das Skript aus; im Terminal sollte das gleiche Ergebnis ausgegeben werden.

*count.sh

```
#!/bin/bash

for count in {0..100}; do
echo $count
let count=count+1
done
```

SCHRITT 4

Die UNTIL-Schleife funktioniert auf ähnliche Weise wie die WHILE-Schleife, meistens allerdings umgekehrt. Wenn wir also mit UNTIL bis 100 zählen, würde dies wie folgt aussehen: `until [ $count -gt 100 ]; do`. Der Unterschied besteht darin, dass die Schleife weiterläuft, bis (UNTIL) die Variable „count" größer als (-gt) 100 ist.

*count.sh (

File Edit View Search Tools Documents Help

*count.sh

```
#!/bin/bash

until [ $count -gt 100 ]; do
echo $count
let count=count+1
done
```

SCHRITT 5

Sie sind nicht auf die Zahlen von 0 bis 100 eingeschränkt. Innerhalb der Schleife können Sie beliebig viele Befehle einfügen und diese so oft ausführen, wie Sie möchten. Sie können eine Million Dateien umbenennen, fünfzig Ordner erstellen usw. Dieses Skript wird z. B. mit der FOR- Schleife zehn Ordner namens folder1 bis hin zu folder10 erstellen.

*count.sh (

File Edit View Search Tools Documents Help

*count.sh

```
#!/bin/bash

for count in {0..10};do
mkdir Folder$count
let count=count+1
done
```

SCHRITT 6

Wenn Sie die FOR-Anweisung erneut anwenden, können Sie die Zählsequenz durch Ändern des {0..100}-Teils ausführen. Dieser Abschnitt des Codes bedeutet Start, Ende, Inkrement {START..END..INCREMENT}. Gibt es kein Inkrement, besteht die Sequenz bis zum Ende nur aus einer einzigen Ziffer. Sie können die Schleifen z. B. folgendermaßen in Zweierschritten bis 1000 zählen lassen: `for count in {0..1000..2}; do`.

*count.sh

File Edit View Search Tools Documents Help

*count.sh

```
#!/bin/bash

for count in {0..1000..2};do
echo $count
let count=count+1
done
```

Bash-Skripte erstellen – Teil 4

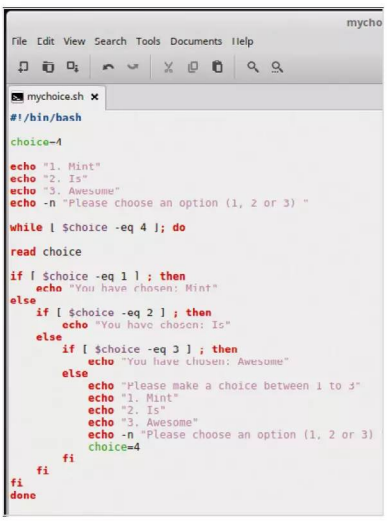
Wir sind bereits auf Benutzerinteraktionen gestoßen und haben uns auch das Erstellen von Schleifen innerhalb eines Skripts angeschaut, mit denen gezählt oder eine Aufgabe mehrmals wiederholt wurde. Hier kombinieren und erweitern wir nun das Ganze.

DER CHOICE-BEFEHL

Hier ist ein weiterer Befehl, CHOICE, sowie ein paar verschachtelte IF- und ELSE-Anweisungen. Beginnen Sie, indem Sie ein neues Skript namens `mychoice.sh` erstellen.

SCHRITT 1

Das mychoice.sh-Skript sieht etwas komplexer aus. Wir haben hier eine Liste mit vier Auswahlmöglichkeiten (choices) und drei möglichen Optionen. Die Optionen Mint, Is und Awesome werden angezeigt, wenn der Benutzer die korrekte Optionstaste drückt. Wenn nicht, wird das Menü wieder erscheinen (die vierte Auswahl).



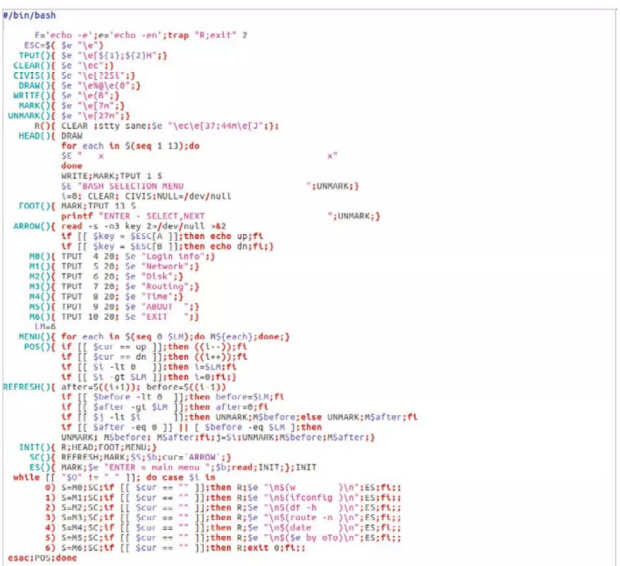
SCHRITT 2

Wenn Sie dem Skript folgen, werden Sie, basierend auf dem, was wir bereits abgedeckt haben, bald den Dreh raushaben und verstehen, was passiert. WHILE, IF und ELSE durchlaufen die Optionen und bringen Sie zum Anfang zurück, wenn Sie die falsche Option wählen, während die FI-Anweisung die Schleife schließt.

```
david@david-mint ~/scripts $ ./mychoice.sh
1. Mint
2. Is
3. Awesome
Please choose an option (1, 2 or 3) 1
You have chosen: Mint
david@david-mint ~/scripts $ ./mychoice.sh
1. Mint
2. Is
3. Awesome
Please choose an option (1, 2 or 3) 2
You have chosen: Is
david@david-mint ~/scripts $ ./mychoice.sh
1. Mint
2. Is
3. Awesome
Please choose an option (1, 2 or 3) 3
You have chosen: Awesome
david@david-mint ~/scripts $ ./mychoice.sh
1. Mint
2. Is
3. Awesome
Please choose an option (1, 2 or 3) 4
```

SCHRITT 3

Sie können natürlich die Anzahl der Auswahlmöglichkeiten erhöhen, müssen dabei aber sicherstellen, dass sie mit der Anzahl der Auswahlmöglichkeiten in den IF-Anweisungen übereinstimmen. Das Skript kann den Bildschirm recht schnell ausfüllen. Dieses lange Skript ist eine weitere Methode zum Anzeigen des Menüs und hat auch ein nettes Farbschema.



SCHRITT 4

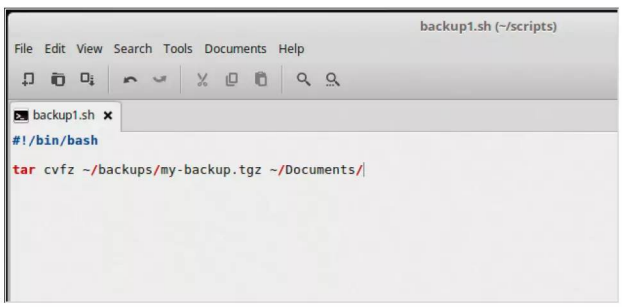
Im Skript können Sie mit den Pfeil- und der Eingabetaste das Menü bedienen. Jede Auswahlmöglichkeit ist ein externer Befehl, der verschiedene Informationen liefert. Experimentieren Sie mit den Befehlen und Auswahlmöglichkeiten und schauen Sie, was dabei herauskommt. Es geht etwas über das hinaus, womit wir uns befasst haben, zeigt aber, was erreicht werden kann.

ERSTELLEN EINES BACKUP-SKRIPTS

Ein viel genutztes Beispiel zum Schreiben von Bash-Skripten ist das Erstellen einer Backup-Routine, mit der die Aufgabe automatisiert und nebenbei noch ein paar Anpassungen hinzugefügt werden können.

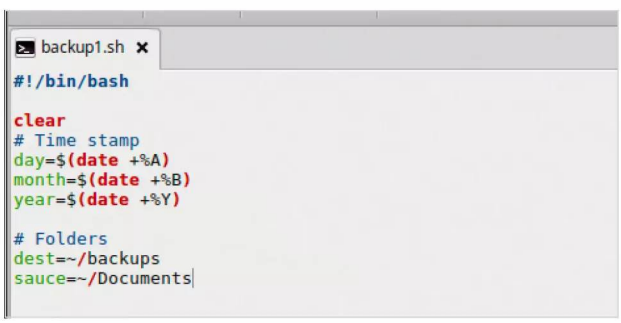
SCHRITT 1

Ein sehr einfaches Backup-Skript würde in etwa wie folgt aussehen: `#!/bin/bash, then, tar cvfz ~/backups/my-backup.tgz ~/Documents/`. Dadurch wird ein komprimiertes Datei-Backup des ~/Documents-Verzeichnisses erstellt und in einem Verzeichnis namens /backups unter dem Namen „mybackup.tgz“ abgelegt.



SCHRITT 2

Das Skript ist zwar in Ordnung so wie es ist, Sie können es jedoch interaktiver gestalten. Beginnen Sie, indem Sie einige Variablen definieren. Geben Sie den Text in der Abbildung in ein neues backup.sh-Skript ein. Wir haben das Wort „source“, also Quelle, absichtlich als „sauce“ falsch geschrieben, da es bereits einen eingebauten Befehl namens „source“ gibt.

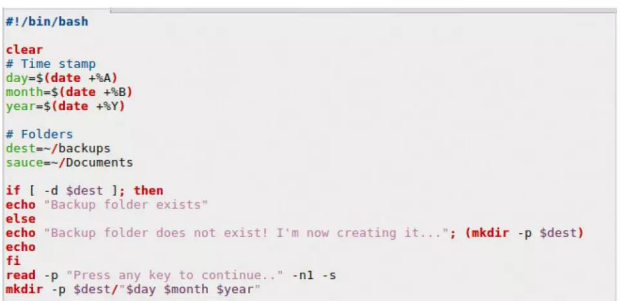


SCHRITT 3

Das vorherige Skript ermöglicht es, einen Zeitstempel zu erstellen, damit Sie wissen, wann das Backup ausgeführt wurde. Sie haben auch eine „dest“-Variable erstellt, das Verzeichnis, in dem das Backup erstellt wird (~/backups). Sie können nun einen Codeabschnitt hinzufügen, mit dem zuerst geprüft wird, ob ein ~/backups-Verzeichnis existiert und, wenn nicht, eins erstellt wird.

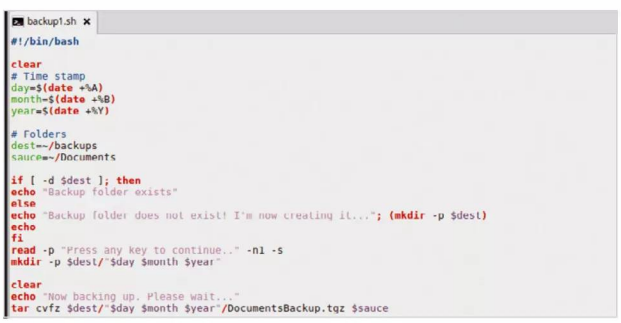
SCHRITT 4

Nachdem das ~/backups-Verzeichnis erstellt wurde, können Sie darin nun ein neues Unterverzeichnis erstellen, das auf den Zeitstempel-Variablen basiert, die Sie zu Beginn festgelegt haben. Geben Sie `mkdir -p $dest/”$day $month $year”` ein. Hier werden die Backup-Dateien abgelegt, die für den Tag/Monat/Jahr relevant sind.



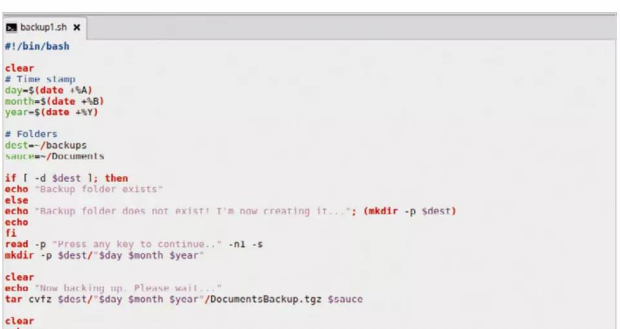
SCHRITT 5

Nachdem nun alles vorhanden ist, können Sie die eigentliche Backup-Routine eingeben, die auf dem tar-Befehl in Schritt 1 basiert. Zusammen mit den Variablen haben wir Folgendes: `tar cvfz $dest/”$day $month $year” /DocumentsBackup.tgz $sauce`. Mit dem echo-Befehl können Sie nützliche Kommentare wie “Backup wird ausgeführt“ einfügen.



SCHRITT 6

Zu guter Letzt können Sie mit dem echo-Befehl auch eine Bestätigungsnachricht einfügen, z. B. `echo “Backup ist abgeschlossen”`. Das Skript ist nicht zu komplex und lässt sich leicht an Ihre eigenen Verzeichnisse in Ihrem Home-Bereich sowie dem gesamten Home-Bereich an sich anpassen.



Bash-Skripte erstellen – Teil 5

Das Backup-Skript, das wir uns zuvor angesehen haben, kann um weitere Auswahlmöglichkeiten erweitert werden, z. B. Wahl des Speicherorts der Backup-Datei durch Benutzerinteraktion etc. Die Automatisierung von Aufgaben ist einer der Hauptvorteile der Bash-Skripte.

EINFACHE AUTOMATISIERUNG UND PRAKTISCHE SKRIPTE

Müssen Sie Zeile für Zeile Befehle eingeben, um Systeminformationen abzurufen, eine Datei zu suchen oder einen Stapel von Dateien umzubenennen? Ein Skript wäre da eine bessere Lösung.

SCHRITT 1

Wir beginnen mit der Erstellung eines Skripts zur Anzeige der Mint-Systeminformationen. Erstellen Sie ein neues Skript namens `sysinfo.sh` und geben Sie das Folgende in Xed oder einen Texteditor Ihrer Wahl ein.

```
sysinfo.sh
#!/bin/bash

#  Hostname information:
echo -e "\e[31;43m***** HOSTNAME INFORMATION *****\e[0m"
hostnamectl
echo ""

# -File system disk space usage:
echo -e "\e[31;43m***** FILE SYSTEM DISK SPACE USE *****\e[0m"
df -h
echo ""

# -Free and used memory:
echo -e "\e[31;43m***** FREE AND USED MEMORY *****\e[0m"
free
echo ""

# -System uptime and performance load:
echo -e "\e[31;43m***** SYSTEM UPTIME AND LOAD *****\e[0m"
uptime
echo ""

# -Users currently logged in:
echo -e "\e[31;43m***** CURRENT USERS *****\e[0m"
who
echo ""

# -Top five processes being used by the system:
echo  e "\e[31;43m***** TOP 5 MEMORY CONSUMING PROCESSES *****\e[0m"
ps -eo %mem,%cpu,comm --sort=-%mem | head -n 6
echo ""
echo -e "\e[1;32mDone.\e[0m"
```

SCHRITT 2

Wir haben ein paar zusätzliche Befehle in dieses Skript eingefügt. Der Erste ist die Erweiterung -e für echo, d. h. er ermöglicht die echo-Interpretation zusätzlicher Instanzen einer neuen Zeile sowie anderer Sonderzeichen. Das Element „31;43m“ aktiviert die Farbe für den Vorder- und Hintergrund.

```
david@david-mint ~/scripts $ ./sysinfo.sh
***** HOSTNAME INFORMATION *****
   Static hostname: david-mint
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 5ab3c275b7304ed3b8aeef9ffcc37eb4
           Boot ID: 61ce1baadf934f649cf5ac809abe7e18
    Virtualization: oracle
  Operating System: Linux Mint 18.1
            Kernel: Linux 4.4.0-53-generic
      Architecture: x86-64

***** FILE SYSTEM DISK SPACE USE *****
```

SCHRITT 3

Jeder Abschnitt führt einen anderen Terminalbefehl aus. Die Ergebnisse werden unter der entsprechenden Überschrift ausgegeben. Sie können Weiteres hinzufügen, z. B. die aktuellen Aliasnamen, die im System verwendet werden, Uhrzeit und Datum usw. Außerdem können Sie all diese Informationen auch in eine praktische HTML-Datei weiterleiten, die in einem Browser angezeigt werden kann.

```
david@david-mint ~/scripts
File  Edit  View  Search  Terminal  Help
david@david-mint ~/scripts $ ./sysinfo.sh > sysinfo.html
david@david-mint ~/scripts $
```

SCHRITT 4

Obwohl es einfache Terminalbefehle gibt, mit denen Sie nach einer bestimmten Datei oder einem bestimmten Ordner suchen können, macht es oft mehr Spaß, ein Skript dafür zu erstellen. Erstellen Sie ein neues Skript namens `look4.sh` und geben Sie den Inhalt der unteren Abbildung ein.

```
look4.sh (~
File  Edit  View  Search  Tools  Documents  Help
look4.sh
#!/bin/bash

target=~/

read name

output=$( find "$target" -iname "*.$name" 2> /dev/null )

if [[ -n "$output" ]]; then
    echo "$output"
else
    echo "No match found"
fi
```

SCHRITT 5

Nach der Ausführung wartet das Skript auf Eingaben des Benutzers, in diesem Fall auf die Dateierweiterung wie jpg, mp4 usw. Wir wollen es aber etwas benutzerfreundlicher machen und fügen kurz vor dem read-Befehl folgende Aufforderung ein: `echo -n "Please enter the extension of the file you're looking for: "`.

*look4.sh

```
#!/bin/bash

target=~/

echo -n "Please enter the extensions of the file you're looking for: "
read name

output=$( find "$target" -iname "*.$name" 2> /dev/null )

if [[ -n "$output" ]]; then
    echo "$output"
else
    echo "No match found"
fi
```

SCHRITT 6

Hier ist ein Skript, das die App espeak verwendet. Installieren Sie espeak mit `sudo apt-get install espeak`, und geben Sie den Text in der Abbildung in ein neues Skript namens `speak.sh` ein. Wie Sie sehen, ist es eine Neuauflage des ersten Begrüßungsskripts, das wir ausgeführt haben. Allerdings verwendet es dieses Mal die Variablen von espeak.

speak.sh

```
#!/bin/bash

echo -n "Hello, what is your first name? "
read firstname
echo -n "Thank you, and what is your surname? "
read surname
clear
espeak "Hello $firstname $surname, how are you on this fine $(date +%A)?"
```

SCHRITT 7

Wir haben uns schon kurz die farblichen Skript-Ausgaben angeschaut. Es würde zu lange dauern, um die Farboptionen zu erläutern, aber es gibt ein Skript, das die verfügbaren Daten ausgibt. Erstellen Sie ein neues Skript namens `colours.sh` und geben Sie den unteren Text ein.

colours.sh

```
#!/bin/bash

clear
echo -e "Normal \e[1mBold"
echo  c "Normal \e[2mDim"
echo -e "Normal \e[4mUnderlined"
echo -e "Normal \e[5mBlink"
echo -e "Normal \e[7minverted"
echo -e "Normal \e[8mHidden"
echo
echo -e "\e[0mNormal !ext"
echo

echo -e "Default \e[39mDefault"
echo -e "Default \e[30mBlack"
echo -e "Default \e[31mRed"
echo -e "Default \e[32mGreen"
echo -e "Default \e[33mYellow"
echo -e "Default \e[34mBlue"
echo -e "Default \e[35mMagenta"
echo -e "Default \e[36mCyan"
echo -e "Default \e[37mLight gray"
echo -e "Default \e[90mDark gray"
echo -e "Default \e[91mLight red"
echo -e "Default \e[92mLight green"
echo -e "Default \e[93mLight yellow"
echo -e "Default \e[94mLight blue"
echo -e "Default \e[95mLight magenta"
echo  c "Default \e[96mLight cyan"
echo -e "Default \e[97mWhite"
echo
echo -e "Default \e[49mDefault"
echo -e "Default \e[40mBlack"
echo -e "Default \e[41mRed"
echo -e "Default \e[42mGreen"
echo  c "Default \e[43mYellow"
echo -e "Default \e[44mBlue"
echo -e "Default \e[45mMagenta"
echo -e "Default \e[46mCyan"
echo -e "Default \e[47mLight gray"
echo -e "Default \e[100mDark gray"
echo -e "Default \e[101mLight red"
echo -e "Default \e[102mLight green"
echo -e "Default \e[103mLight yellow"
echo -e "Default \e[104mLight blue"
echo -e "Default \e[105mLight magenta"
echo -e "Default \e[106mLight cyan"
echo -e "Default \e[107mWhite"
```

SCHRITT 8

Die Ausgabe von colours.sh kann natürlich auch kombiniert werden, was je nach Wunsch unterschiedliche Auswirkungen auf die Ausgabe hat, z. B. weißer Text, der vor rotem Hintergrund aufblinkt. Leider funktioniert der Blinkeffekt nicht in allen Terminals, evtl. müssen Sie daher zu einem anderen Terminal wechseln.

```
Normal Text

Default Default
Default Black
Default Red
Default Green
Default Yellow
Default Blue
Default Magenta
Default Cyan
Default Light gray
Default Dark gray
Default Light red
Default Light green
Default Light yellow
Default Light blue
Default Light magenta
Default Light cyan
Default White

Default Default
Default Black
Default Red
Default Green
Default Yellow
Default Blue
Default Magenta
Default Cyan
Default Light gray
Default Dark gray
Default Light red
Default Light green
Default Light yellow
Default Light blue
Default Light magenta
Default Light cyan
Default
david@david-mint ~/scripts $
```

SCHRITT 9

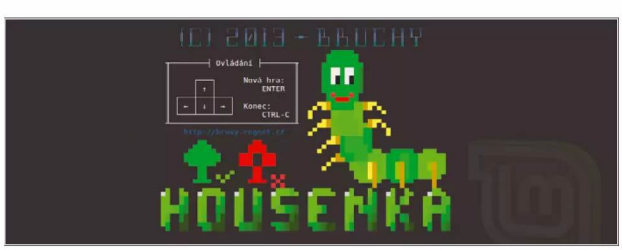
Mit Zenity können Sie eine grafische Benutzeroberfläche ausgeben. Geben Sie dazu den unteren Text in ein neues Skript namens `mmenu.sh` ein. Machen Sie es ausführbar und führen Sie es aus. Es sollten einige Dialogfelder angezeigt werden, gefolgt von einer abschließenden Meldung.

mmenu.sh

```
#!/bin/bash

firstname=$(zenity --entry --title="Your Name" --text="What is your first name?")

surname=$(zenity --entry --title="Your Name" --text="What is your first surname?")

zenity --info --title="Hello!" --text="Welcome to Linux Mint.\n\n Have fun, $firstname $surname."
```

SCHRITT 10

Das Gaming in einem Bash-Skript kommt zwar nicht häufig vor, ist aber durchaus möglich. Wenn Sie Lust auf ein Spiel haben, geben Sie `wget http://bruxy.regnet.cz/linux/housenka/housenka.sh` ein, machen Sie das Skript ausführbar und führen Sie es aus. Es ist in Polnisch und wurde von Martin Bruchanov geschrieben, aber wir sind sicher, dass Sie es ändern können. Hinweis: Der Titelbildschirm basiert auf Base64.



Kurzreferenz zur Befehlszeile

Wenn Sie gänzlich auf Linux umsteigen, werden Sie schnell feststellen, dass sich die grafischen Oberflächen von Ubuntu, Mint usw. zwar für viele, aber nicht alle Aufgaben eignen. Wenn Sie die Anwendung der Befehlszeile verstehen, werden Sie nicht nur Linux besser verstehen, sondern auch Ihr Wissen über das Programmieren verbessern. Unsere Kurzreferenz zur Befehlszeile hilft Ihnen, Linux schneller zu meistern.

DIE TOP 10 DER BEFEHLE

Es mag sich dabei vielleicht nicht um die gängigsten Befehle für jedermann handeln, von Linux- und Befehlszeilen-Benutzern wenden sie jedoch häufig angewandt.

cd

Den `cd`-Befehl werden Sie in der Befehlszeile unter Linux am häufigsten verwenden. Sie können damit zu Ihrem Arbeitsverzeichnis wechseln und sich innerhalb der Hierarchie Ihres Dateisystems bewegen. Sie können auch chdir verwenden.

ls

Der `ls`-Befehl listet die Dateien im aktuellen Verzeichnis auf. In Verbindung mit bestimmten Optionen zeigt er die Dateigröße, das Erstellungsdatum und die Dateiberechtigungen an; z. B. zeigt `ls ~` die im Home-Verzeichnis befindlichen Dateien an.

cp

Der `cp`-Befehl kopiert Dateien und Verzeichnisse. `cp Dateiname sub` erstellt z. B. eine exakte Kopie der Datei, deren Namen Sie eingegeben haben, und benennt die Kopie „sub". Die erste Datei bleibt jedoch mit ihrem ursprünglichen Namen erhalten.

pwd

Der `pwd`-Befehl gibt den vollständigen Pfadnamen des aktuellen Arbeitsverzeichnisses aus (pwd steht für „print work directory"). Beachten Sie, dass das GNOME-Terminal diese Informationen auch in der Titelleiste des Fensters anzeigt.

clear

Der `clear`-Befehl löscht den Bildschirm. Er sucht in der Umgebung nach dem Terminaltyp und dann in der Terminfo-Datenbank, wie der Bildschirm gelöscht wird. Dies entspricht der Eingabe von Control-L in der Bash Shell.

mv

Der `mv`-Befehl verschiebt eine Datei oder benennt sie um. `mv Dateiname sub` benennt z. B. die Originaldatei in „sub" um. `mv sub ~/Desktop` verschiebt die Datei „sub" in Ihr Desktop-Verzeichnis, benennt sie aber nicht um. Sie müssen zum Umbenennen den neuen Dateinamen angeben.

chown

Der `chown`-Befehl ändert den Benutzer- und/oder Gruppeneigentümer einer Datei. Ist nur ein Eigentümer (ein Benutzername/eine numerische Benutzer-ID) angegeben, ist dieser Benutzer der Eigentümer dieser Datei; die Dateigruppe wird nicht geändert.

cmod

Der `chmod`-Befehl ändert die Berechtigungen für die aufgelisteten Dateien. Sie können Berechtigungen für Benutzer, Gruppe und dem Rest der Welt festlegen und bestimmen, ob jeder die Datei lesen, schreiben oder ausführen kann.

rm

Der `rm`-Befehl löscht Dateien und Verzeichnisse. Beim Löschen wird ein Dateiname in einem Dateisystem von den Daten auf dem Speichergerät getrennt und dieser Speicherplatz für zukünftige Schreibvorgänge als nutzbar markiert. Kurz gesagt erhöht das Löschen den verfügbaren Speicherplatz auf der Festplatte.

mkdir

Mit `mkdir` (kurz für „make directory") werden Verzeichnisse in einem Dateisystem erstellt, wenn das angegebene Verzeichnis noch nicht vorhanden ist. Z. B. erstellt `mkdir work` ein Arbeitsverzeichnis. Mit „mkdir" können mehrere Verzeichnisse angegeben werden.

NÜTZLICHE HILFE-/INFO-BEFEHLE

Die folgenden Befehle sind nützlich, um mehr über das System oder Programm zu erfahren, mit dem Sie in Linux arbeiten. Auch wenn Sie sie wahrscheinlich nicht täglich benötigen werden, können sie sich bei der Anwendung als äußerst hilfreich erweisen.

free

Der `free`-Befehl zeigt die Gesamtmenge des freien und verwendeten physischen Speichers und des Swap-Speichers im System an. `free -m` gibt z. B. die Informationen in Megabyte an.

df

Der `df`-Befehl zeigt den Festplattenspeicherplatz des Dateisystems für alle Partitionen an. `df -h` ist wahrscheinlich der nützlichste Befehl (-h bedeutet, dass es für den Menschen lesbar ist).

top

Das `top`-Programm bietet eine dynamische Echtzeitansicht eines laufenden Systems. Es kann eine Zusammenfassung der Systeminformationen sowie eine Liste von Prozessen anzeigen.

uname-a

Der `uname`-Befehl gibt zusammen mit der Option `-a` alle Systeminformationen aus, einschließlich Computername, Kernelname, Version sowie einige weitere Infos.

ps

Der `ps`-Befehl zeigt alle auf dem Computer ausgeführten Prozesse an. Die ps-Version unterscheidet sich geringfügig auf jedem Betriebssystem, aber sie führen alle dasselbe aus.

grep

Mit dem `grep`-Befehl können Sie in einer Reihe von Dateien nach einem bestimmten Suchmuster suchen und dann passende Zeilen ausgeben. Ein Beispiel wäre `grep Suchwort Dateiname`.

sed

Der `sed`-Befehl öffnet einen Stream-Editor. Mit einem Stream-Editor werden Textumwandlungen in einem Eingabestream (Datei oder Eingabe aus einer Pipe) ausgeführt.

adduser

Der `adduser`-Befehl fügt einen neuen Benutzer zum System hinzu. Auf ähnliche Weise fügt der Befehl `addgroup` eine neue Gruppe zum System hinzu.

deluser

Der Befehl `deluser` entfernt einen Benutzer aus dem System. Um die Dateien und das Basisverzeichnis des Benutzers zu entfernen, müssen Sie die Option `-remove-home` hinzufügen.

delgroup

Der Befehl `delgroup` entfernt eine Gruppe aus dem System. Sie können keine Gruppe entfernen, die als primäre Gruppe von Benutzern dient.

man man

Der `man man`-Befehl ruft den manuellen Eintrag für den `man`-Befehl auf, der für den Einstieg äußerst praktisch ist.

man intro

Der `man intro`-Befehl ist besonders nützlich. Er zeigt die Einführung in die Benutzerbefehle an, eine gut geschriebene, recht kurze Einführung in die Linux-Befehlszeile.

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
```

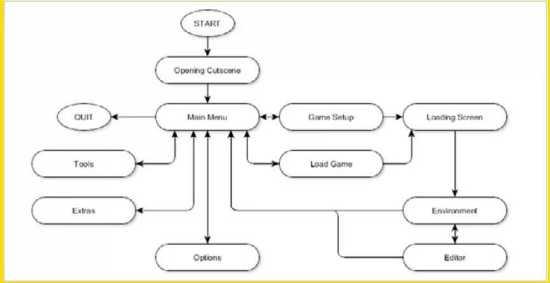
EINFACHE VARIABLEN

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 eines Morgens mit dem Gedanken auf, einen Code für ein klassisches Textabenteuer zu schreiben, aber ohne einen guten Plan ist das nicht sinnvoll. Kleine Codeabschnitte 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 oft 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 klare Anweisungen, 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
12323423423423423 4
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

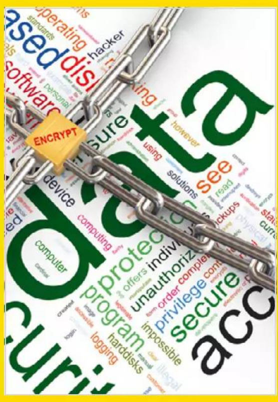
Machen Sie immer ein Backup Ihrer Arbeit sowie ein sekundäres Backup für alle Änderungen, die Sie ausgeführt haben. Fehler können behoben werden, wenn ein gutes Backup vorhanden ist, auf das zurückgegriffen werden kann, wenn etwas schief geht. Es ist viel einfacher, an der Stelle zu beginnen, an der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

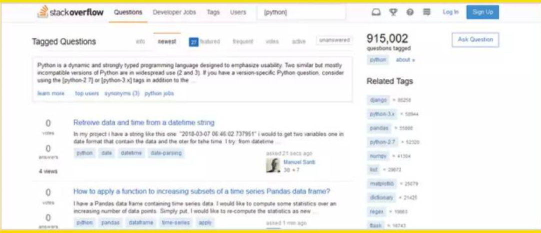
DATEN SICHERN

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 Programme erzeugt haben, was je nach Funktion des Codes verheerende Auswirkungen haben kann. Überprüfen Sie daher Ihre Codegleichungen 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 dieser Funktion ein Leben? Schreiben Sie einen Kommentar, damit Sie und auch andere sehen, was die Funktion bezweckt.

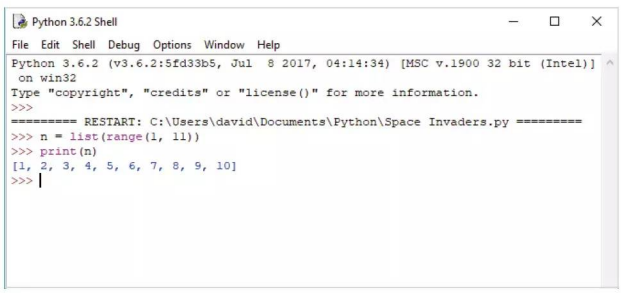
```
# set up pygame
pygame.init()
mainClock = pygame.time.Clock()

# 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:

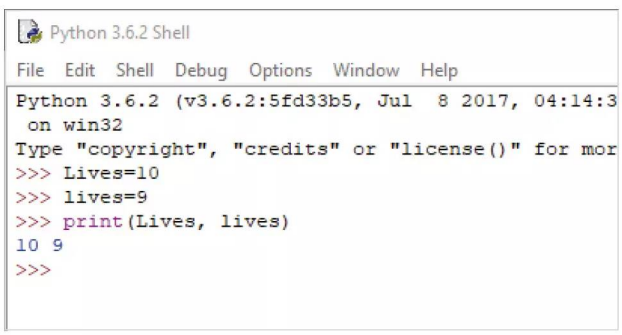
```
n = list(range(1, 11))
```

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

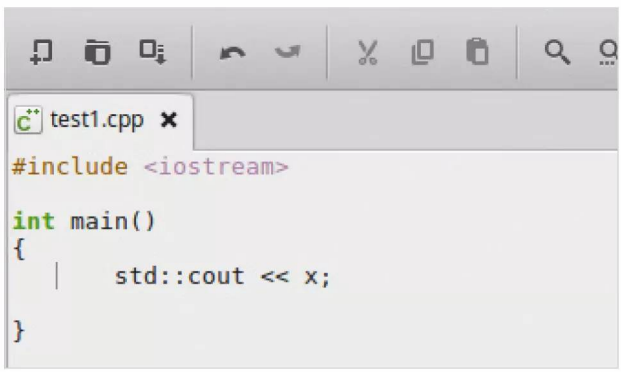
Es gibt viele Fallgruben, auf die der C++-Entwickler stoßen kann, zumal diese Sprache auch komplexer und oft unnachgiebiger ist. Anfänger sollten C++ schrittweise erlernen und erst das Gelernte verarbeiten, bevor sie mit dem nächsten Abschnitt weitermachen.

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 ODER 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.

```
david@mint-mate ~/Documents
File  Edit  View  Search  Terminal  Help
david@mint-mate ~/Documents $ gcc test1.cpp -o test
/tmp/ccA5zhtg.o: In function `main':
test1.cpp:(.text+0x2a): undefined reference to `std::cout'
test1.cpp:(.text+0x2f): undefined reference to `std::ostream::
/tmp/ccA5zhtg.o: In function `__static_initialization_and_dest
)':
test1.cpp:(.text+0x5d): undefined reference to `std::ios_base:
test1.cpp:(.text+0x6c): undefined reference to `std::ios_base:
collect2: error: ld returned 1 exit status
david@mint-mate ~/Documents $ g++ test1.cpp -o test
david@mint-mate ~/Documents $
```

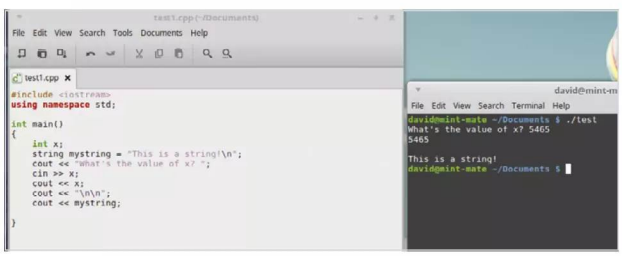
KOMMENTARE

Und wieder sind wir beim Problem der fehlenden Kommentare. Wie zuvor schon erwähnt ist es ohne lesbare Bezeichner 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 einen 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 null 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

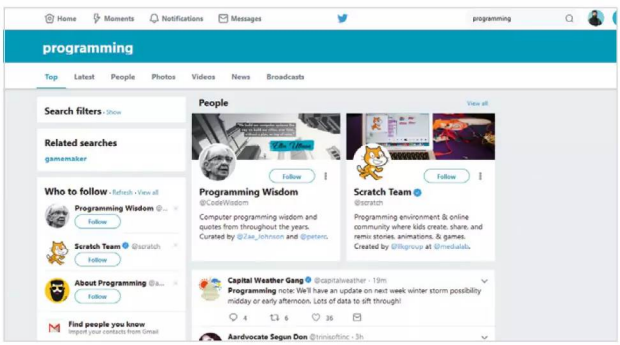
Das Programmieren ist eine kontinuierliche Lernerfahrung. Sie mögen zwar kein Anfänger mehr sein, müssen aber weiterhin Ihren Code testen, Tricks und Tipps lernen, um ihn effizienter zu gestalten, und sogar eine andere Programmiersprache lernen.

#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?

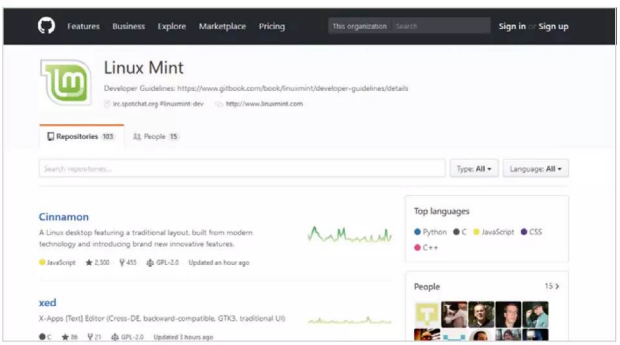
TWITTER

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 können 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-Projekten, 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 Aktualiserungen für den Code handelt, so ist es doch eine ehrenwerte Aufgabe für Programmierer.



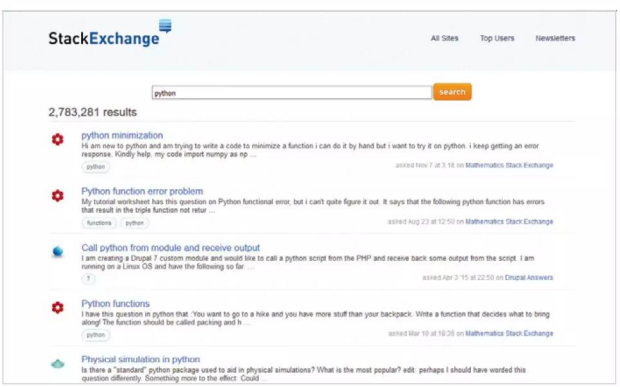
IMMER WEITER PROGRAMMIEREN

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;
};
```

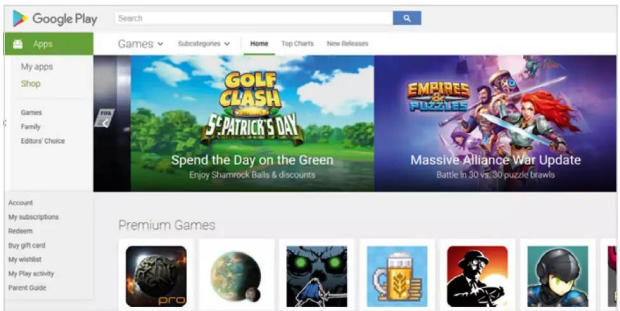
VONEINANDER LERNEN

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.



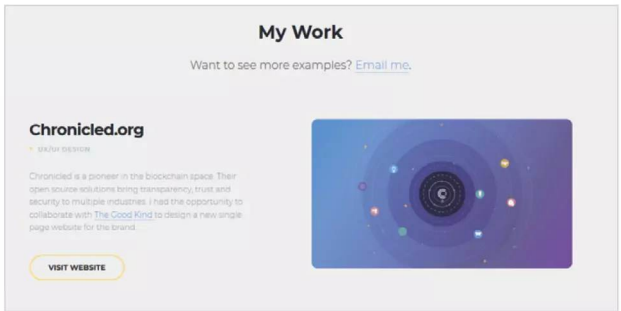
MOBILFUNKMARKT

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.



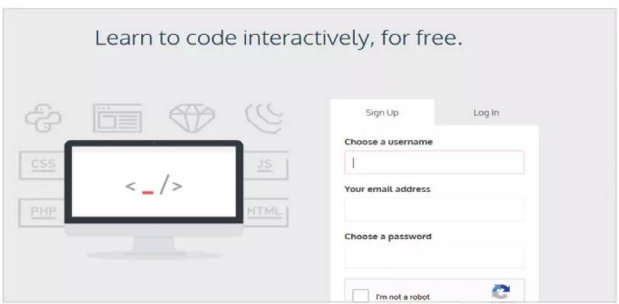
PORTFOLIOS

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.



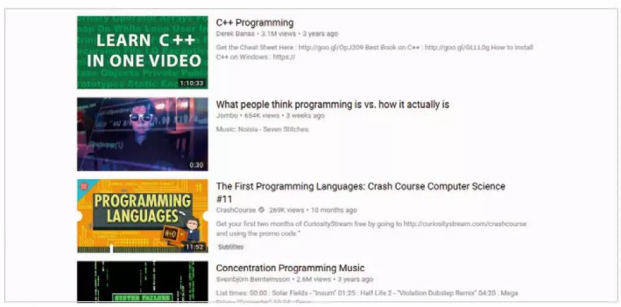
ONLINE LERNEN

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.



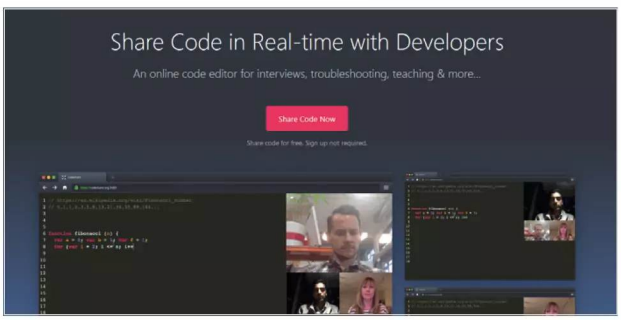
CODE LEHREN

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

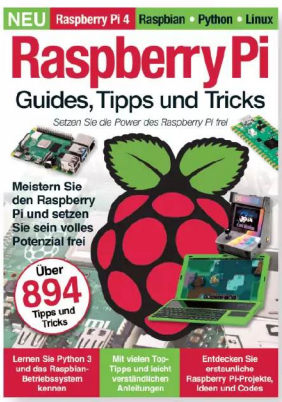
Teilen Sie Ihren Code mit anderen, auch wenn Sie der Meinung sind, dass er nicht sehr gut ist. Die Kritik, Ratschläge und Kommentare, die Sie erhalten, helfen Ihnen, Probleme in Ihrem Code zu beheben. Vielleicht ist Ihr Code aber auch absolut fantastisch, was Sie aber nicht wissen werden, wenn Sie ihn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

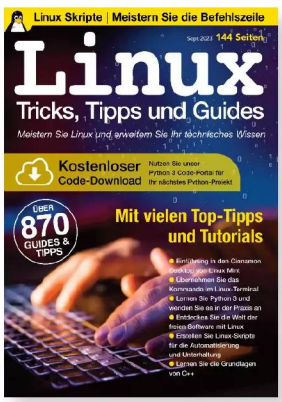


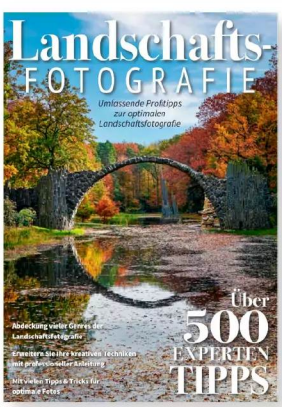
HARDWARE-PROJEKTE

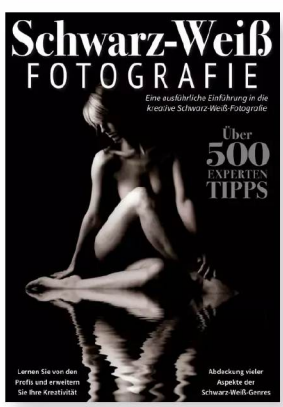
Die Beteiligung an Hardware-Projekten ist eine tolle Möglichkeit, um seinen eigenen Code unter Beweis zu stellen und von anderen Mitwirkenden zu lernen. In vielen Entwicklerforen werden Projekte angeboten, für die sich Programmierer bewerben können, und in denen mit einzigartigen Codes das Beste aus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

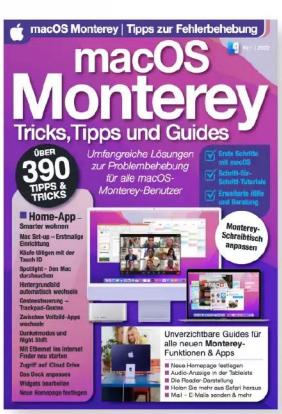
Entdecken Sie noch heute auf Readly unsere weiteren Expertenmagazin!











Papercut

www.pclpublications.com

Coding Tricks, Guides und Tipps
Januar-2024

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: enquiries@pclpublications.com
Vertrieb: Readly AB