# Kapitel 1
# Auf die Plätze …

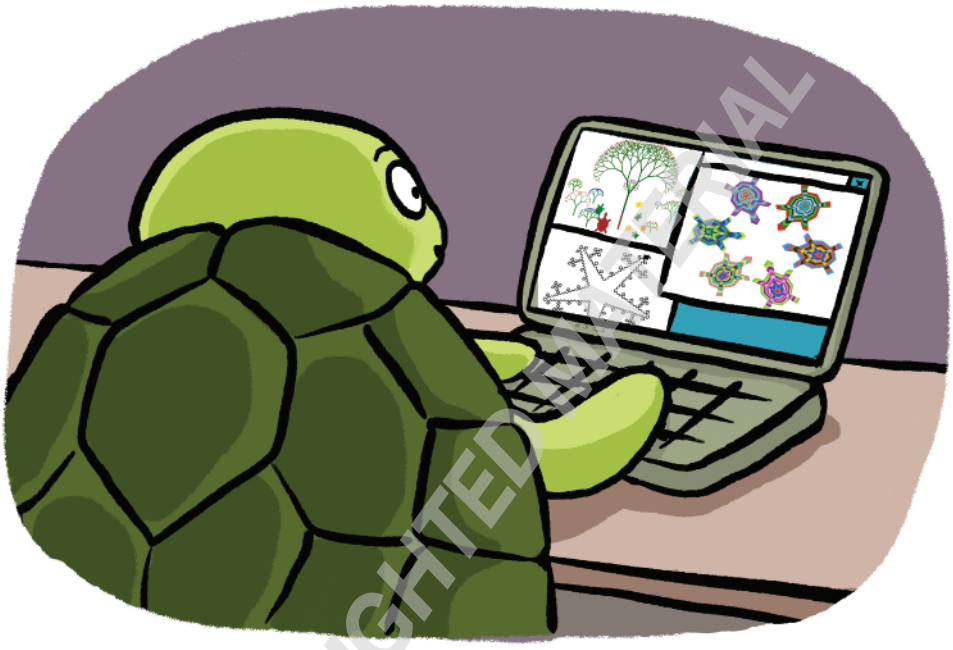


Illustration: Leonard Ermel



**Computer können überhaupt nichts von alleine. Alles muss man ihnen sagen.** Die Sprachen, in denen wir Computern Anweisungen erteilen, heißen *Programmiersprachen*. Sie unterscheiden sich ziemlich stark von den Sprachen, in denen Menschen miteinander reden. Der erste auffällige Unterschied ist, dass Programmiersprachen nicht gesprochen, sondern geschrieben werden. Und zwar von *Programmierern*. Die schreiben auf, was der Computer in welcher Reihenfolge tun soll. Man nennt eine Reihe solcher Anweisungen ein *Programm* oder eine *App* oder einfach *Code* oder *Software*.

## Computer und Programmiersprachen

Computer unterstützen uns beim Schreiben von Texten, beim Bildermalen, beim Komponieren von Musik und beim Suchen und Einkaufen im Internet, sie können 3D-Filme zeigen, Roboter auf dem Mars steuern, . . . Bestimmt fallen dir noch weitere Dinge ein, die wir mit Computern tun.

Da ist es ganz natürlich, dass mehrere Programmiersprachen praktischer sind als eine einzige für alles. Die wichtigsten Programmiersprachen, die heute verwendet werden, heißen Java, C, C++, C#, Python, PHP und JavaScript. Es gibt aber noch viel, viel mehr! Programmierer streiten sich manchmal, welche Sprache für welche Art von Aufgaben am besten geeignet ist. Für uns ist wichtig: Sooo unterschiedlich sind die Programmiersprachen erst einmal gar nicht. Auf jeden Fall ist es leichter, eine Programmiersprache zu lernen als eine echte Fremdsprache. Und wenn du erst eine Programmiersprache kennengelernt hast, dann lernst du die nächste gleich noch einmal so schnell.

Damit ein Computer die Anweisungen in einem Programm befolgen kann, braucht er ein paar Geräte (man nennt sie die *Hardware*). Also auf jeden Fall einen Bildschirm, der Bilder und Texte anzeigt, und einen Lautsprecher für die Töne. Bildschirm und Lautsprecher nennt man Ausgabegeräte.

Illustration: ekzarkho – stock.adobe.com

Manchmal braucht der Computer auch noch Informationen von uns, die nicht von vornherein im Programm stehen: Welchen Text willst du ausdrucken, welche Zahlen sollen addiert werden, wie groß soll der Kreis sein, den du zeichnen möchtest? Solche Informationen teilen wir dem Computer dann über die Tastatur oder die Maus oder den Touchscreen mit. Das sind dann – logisch – Eingabegeräte.

Illustration: ekzarkho – stock.adobe.com

Damit Programmierer auf der ganzen Welt die Anweisungen in Programmiersprachen verstehen und benutzen können, sind die wichtigsten Anweisungen immer auf Englisch. Möchtest du also den Text »HALLO WELT!« auf dem Bildschirm anzeigen lassen, dann benutzt du in vielen Programmiersprachen das Wort `print` (auf Deutsch »drucke«) und schreibst zum Beispiel:

`print("HALLO WELT!")` in Python,

`printf("HALLO WELT!\n");` in C oder

`System.out.println("HALLO WELT!");` in Java.

Schön, jetzt verstehen also wir Programmierer die Anweisungen und können ein Programm schreiben. Aber der Computer soll unsere Anweisungen ja auch verstehen, und der kann doch kein Englisch, oder?

Um zu verstehen, wie der Computer Anweisungen verarbeitet, schauen wir uns mal an, wie er aufgebaut ist. Von Ein- und Ausgabegeräten hast du schon gehört, ein wichtiges Stück Hardware fehlt aber noch: die *Zentraleinheit* (auf Englisch *CPU*, für *Central Processing Unit*). Sie ist das Herz des Computers und ist mit den Ein- und Ausgabegeräten über Stromleitungen verbunden.

Die CPU ist ein elektronischer Chip und enthält enorm viele kleine Schalter (sogenannte *Transistoren*), die entweder geschlossen sein können (dann fließt Strom) oder offen (dann kann kein Strom fließen). Unser Programm steuert nun, welche von den Schaltern geöffnet und welche geschlossen werden. Dafür reicht der CPU eine lange Liste, die für jeden Schalter sagt, ob er `offen` oder `geschlossen` sein soll. Ein Programm, wie die CPU es versteht, also in *Maschinensprache*, kannst du dir so vorstellen (die Nummern sind die einzelnen Schalter):

```
1. offen
2. geschlossen
3. offen
4. offen
5. geschlossen
6. offen
7. offen
8. offen
```

Weil das ein bisschen lang wird, hat man sich darauf geeinigt, stattdessen die beiden Zeichen `0` (Null) für `offen` und `1` (Eins) für `geschlossen` zu verwenden, und die Nullen und Einsen in Schalterreihenfolge als Kette anzuordnen. Das Programm oben kann dann viel kürzer geschrieben werden, nämlich als Kette `0100 1000`. Der Effekt der Kette `0100 1000` ist, dass der Buchstabe »H« auf unserem Bildschirm erscheint. Für den Rest unseres Textes »HALLO WELT!« braucht die CPU noch viele weitere Nullen und Einsen.

## Von Bits und Bytes

Eine Kette aus Nullen und Einsen, also eine Zahl, die nur aus zwei unterschiedlichen Ziffern aufgebaut ist, nennt man *binäre Zahl* (*bini* ist das lateinische Wort für *je zwei*). Die einzelnen Stellen der Zahl heißen *Bit*. Eine binäre Zahl aus acht Stellen, also aus acht Bits, wird *Byte* genannt.

Jetzt bleibt nur noch die Frage, wie unser Programm mit seinen englischen Anweisungen in Maschinensprache übersetzt wird, also in Ketten von Nullen und Einsen.

Bei der Übersetzung muss sehr sorgfältig vorgegangen werden, denn eine einzige falsch übersetzte Eins oder Null kann das ganze schöne Programm kaputt machen. Wenn bei einer Aufgabe sorgfältig vorgegangen werden muss und etwas genau nach Vorschrift getan werden soll, bietet es sich an, einen Computer zu verwenden und für die Aufgabe ein Programm zu schreiben.

Es gibt daher besondere Übersetzerprogramme (genannt *Compiler* oder *Interpreter*), die das für uns lesbare Programm (den *Quellcode*) in Maschinensprache übersetzen (den *Binärcode*) und das übersetzte Programm dann ausführen. Diese Übersetzerprogramme brauchen wir nicht selbst zu schreiben, es gibt sie für jede Programmiersprache. Du wirst im nächsten Abschnitt erfahren, wo du den Interpreter für Python im Internet findest und herunterladen kannst.

## Compiler und Interpreter

Die beiden Übersetzervarianten Compiler und Interpreter unterscheiden sich in der Reihenfolge von Übersetzen und Ausführen.

Ein *Compiler* übersetzt erst das gesamte Programm in *Binärcode* und speichert es als Datei auf dem Computer. Danach, wenn der Compiler fertig übersetzt hat, kann das Programm ausgeführt werden. Ein fertig übersetztes Programm kann immer wieder aufgerufen werden und muss nicht jedes Mal neu übersetzt werden. Die Programmiersprachen C und C++ verwenden Compiler.

Im Gegensatz dazu übersetzt der *Interpreter* einzelne Anweisungen nicht bevor, sondern während das Programm läuft. Er liest also eine Anweisung aus dem Quellcode, zum Beispiel `print("HALLO WELT!")`, übersetzt sie in Binärcode und sorgt dafür, dass diese Anweisung sofort auf dem Computer ausgeführt wird. Danach liest er die nächste Anweisung, zum Beispiel `print("TSCHÜSS WELT!")`, übersetzt sie und führt sie aus. Und so weiter. Der Interpreter erzeugt keine Datei mit Binärcode. Jedes Mal, wenn das Programm laufen soll, muss er wieder loslegen und das Programm neu übersetzen und ausführen. Python ist ein Beispiel für eine Programmiersprache mit Interpreter.

Wenn du also dein Programm ausführst (auf dem Computer laufen lässt), dann startest du in Wirklichkeit erst das Übersetzerprogramm und erzeugst Binärcode aus deinem Quellcode. Mit dem Binärcode werden dann die Schalter geschaltet, und du siehst das Ergebnis auf deinem Ausgabegerät.

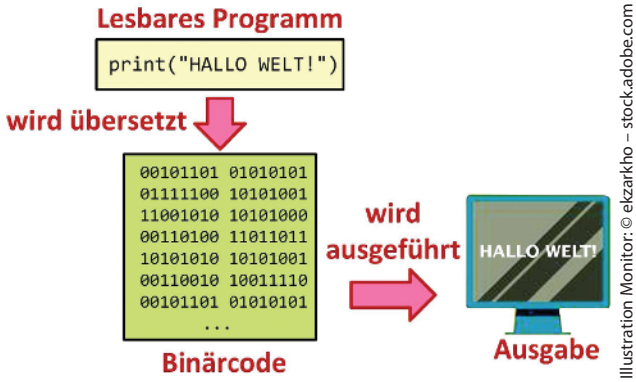



Illustration Monitor: © ekzarkho – stock.adobe.com

## Python auf deinem Computer

Damit du nun an deinem Computer mit dem Programmieren in Python loslegen kannst, brauchst du zwei Dinge: einen *Python-Interpreter*, der deinen Code in Binärcode übersetzen und ausführen kann, und einen *Editor*. Das ist ein Programm zum Schreiben von Programmen. Sowohl der Interpreter als auch der Editor können von der Python-Homepage aus dem Internet heruntergeladen werden, und das sogar kostenlos. Das wollen wir jetzt tun.

### Warum heißt die Programmiersprache Python?

Programmiersprachen haben manchmal seltsame Namen. Meistens entscheiden ihre Erfinder, wie sie heißen sollen. Der niederländische Erfinder von Python, Guido van Rossum, ist ein Fan der englischen Komikergruppe *Monty Python* aus den 1970er-Jahren, die mehrere Kinofilme drehte und eine Fernsehserie hatte (*Monty Python's Flying Circus*).

Van Rossum hat Python entwickelt, weil er eine möglichst einfache und übersichtliche Programmiersprache haben wollte. Das ist ihm gelungen, Python war im Jahr 2022 vor C und Java die beliebteste Programmiersprache!

## Python installieren

Wir installieren jetzt den Python-Interpreter und -Editor unter Windows (am besten Windows 8, 10 oder 11).

1. **Gehe auf die Internetseite von Python: https://www.python.org/downloads**
2. **Klicke dort auf das gelbe Feld Download Python 3.10.5 (bzw. die gerade aktuelle Version von Python).**
3. **Speichere die Datei (am besten im Ordner Downloads).**
4. **Starte die Installation mit einem Doppelklick auf den Dateinamen und folge den Installationsanweisungen.**

   Du bekommst einen neuen Ordnereintrag Python 3.10 unter Alle Programme im Windows-Startmenü.

*Sollte dein Computer kein Windows-Computer sein, sondern unter Linux/Unix oder macOS oder einem anderen Betriebssystem laufen, dann klickst du auf der Internetseite von Python nicht auf das gelbe Feld, sondern auf den Link für dein Betriebssystem, lädst dir die neueste passende Version herunter und folgst den Installationsanweisungen. Lass dir hier eventuell von deinen Eltern helfen.*

## Python-Symbol auf dem Bildschirm erstellen

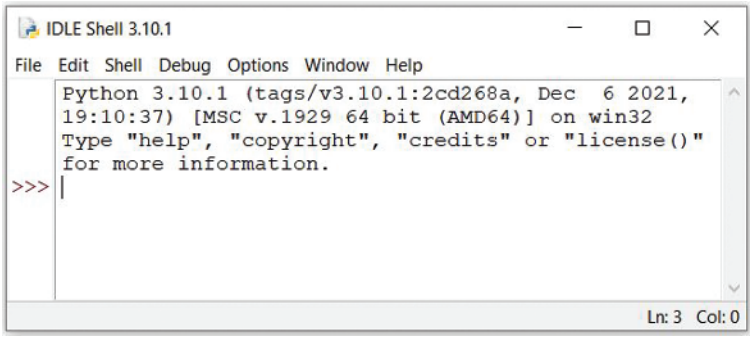
Um ein Python-Symbol auf dem Bildschirm zu erstellen, befolge die folgenden Schritte:

1. **Klicke mit der linken Maustaste auf Start (das Windows-Startsymbol links unten), scrolle dann durch die alphabetisch sortierten Programme und klicke auf das Ordnersymbol Python 3.10.**

**2** **Halte mit der linken Maustaste den Eintrag IDLE (Python 3.10) gedrückt, ziehe ihn aus dem Startmenü auf den Desktop und lasse ihn dort los.**

Jetzt gibt es ein Symbol mit dem Namen IDLE (Python 3.10) auf deinem Bildschirm.

Mit einem Doppelklick auf das Symbol öffnest du ein neues Fenster, die sogenannte Python-Shell (auf Deutsch »Schale/Hülle«).




## Dein erstes Python-Programm

Wie der Name schon sagt, ist die Python-Shell erstmal nur eine leere Schale, die nichts enthält. Ganz oben stehen ein paar technische Informationen über die Version von Python, die uns hier nicht interessieren. Wichtig sind die drei Zeichen `>>>` ganz unten: Sie stellen Pfeile dar und zeigen an, dass dahinter die nächste Eingabe von dir erwartet wird. Dort kannst du jetzt direkt Programmier-Anweisungen schreiben und dann den Interpreter darauf ansetzen. Und das geht so:

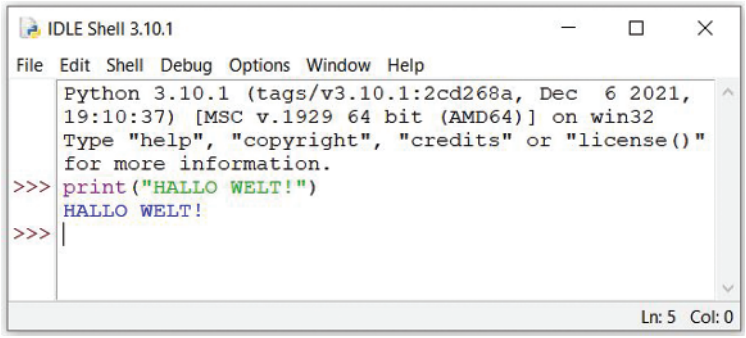
**1** **Klicke in das Fenster.**

**2** **Schreibe `print("HALLO WELT!")`.**

**3** **Drücke die Eingabetaste ↵.**

Fertig. Dein erstes Programm wird sofort übersetzt und ausgeführt, und das Ergebnis ist direkt eine Zeile tiefer zu bewundern. Wie du siehst, erscheinen erneut die drei Zeichen `>>>` und signalisieren dir, dass Python für deine nächste

Anweisung bereit ist. Natürlich kannst du statt »HALLO WELT!« auch einen anderen Text schreiben lassen, probiere es aus . . .




## Python als Taschenrechner

Das Rechnen in Python geht auch ganz einfach. Wir werden wieder die Python-Shell benutzen. Die Grundoperationen `+`, `-`, `*`, `/` funktionieren so wie bei einem Taschenrechner. Auch Klammern kannst du setzen. Gib einfach einen Rechenausdruck ein, zum Beispiel `(5+6)*3+8*3-37`, und drücke ↵, um die Eingabe abzuschließen. Das Ergebnis ist 20. Rechne mal nach . . .

Wenn du die Klammern weglässt, erhältst du ein ganz anderes Ergebnis (nämlich 10). Denn ohne Klammern gilt auch für Python das Prinzip: »Punktrechnung vor Strichrechnung«.

Jetzt kannst du die Python-Shell auch als Taschenrechner benutzen, zum Beispiel, wenn die Batterie von deinem Taschenrechner leer ist.

## Programme für die Ewigkeit

Bisher sind wir mit der Python-Shell gut zurechtgekommen. Aber wenn deine Programme länger werden, wirst du bestimmt immer mal wieder Fehler machen. Das ist ganz normal. Aber jedes Mal musst du wieder von vorne mit dem Tippen anfangen (oder, wenn du schlau bist, die alte Anweisung kopieren, unten einfügen, verbessern und dann erst auf ↵ drücken). Aber das ist viel Arbeit.

Außerdem willst du deine Programme vielleicht auch irgendwann mal wiederverwenden, also vielleicht nach drei Monaten oder einem Jahr. Die ganze Programmiererei soll sich ja auch lohnen!

## Programme speichern

Programme wären nicht besonders nützlich, wenn du sie jedes Mal Zeile für Zeile wieder aufs Neue eintippen müsstest. Du brauchst also eine Möglichkeit, deine Programme zu speichern, damit du sie immer wieder aufrufen und laufen lassen kannst.

Dafür gibt es den *Python-Editor*. Um ihn in Betrieb zu nehmen, gehst du folgendermaßen vor:

**1** **Klicke in der Python-Shell oben links in der sogenannten *Menüzeile* auf File und wähle New File.**

Es öffnet sich ein neues leeres Fenster. Das ist der Editor. Hier kannst du Programmcode schreiben, zum Beispiel:

```
print ("HALLO WELT")
print ("Ich rechne mit Python")
print ("Eine Aufgabe mit Klammern: (5+6)*3+8*3-37 = ")
print ((5+6)*3+8*3-37)
print ("jetzt aber ohne Klammern: 5+6*3+8*3-37 = ")
print (5+6*3+8*3-37)
```

*Es ist sehr nützlich, das eigene Programm mit* Kommentaren *zu versehen. Das sind Bemerkungen, die dir selbst und anderen helfen zu verstehen, warum du was programmiert hast. In den ersten Zeilen einer Programmdatei sollte am besten stehen, was das Programm macht und wie es heißt. Kommentare werden am Beginn mit* # *gekennzeichnet, damit der Python-Interpreter weiß, dass dieser Text nicht zum eigentlichen Programm gehört. Beim Übersetzen wird der Rest der Zeile hinter einem* #*-Zeichen einfach ignoriert.*

**2** **Speichere dein Programm, indem du zuerst File und dann Save klickst.**

Du wirst in einem neuen Fenster nach einem Ordner und dem Dateinamen gefragt.

**3** **Wähle den Ordner aus, gib einen Namen ein, zum Beispiel `HalloWelt`, und klicke auf Speichern.**

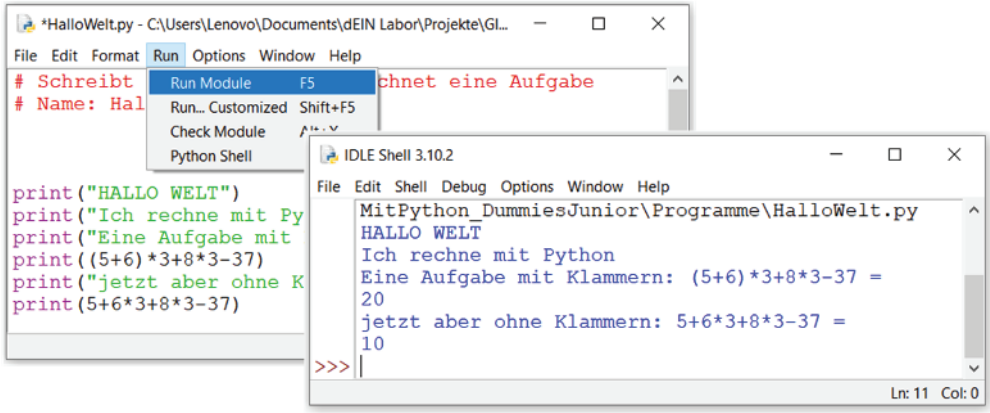
Du hast jetzt eine neue Datei *HalloWelt.py* auf deinem Computer. Die Endung *.py* wird automatisch beim Speichern ergänzt. Sie bedeutet, dass diese Datei ein Python-Programm enthält.

## Programme ausführen

Nun kannst du dein Programm ausführen, so oft du möchtest.

1 **Klicke einfach in der Menüzeile vom Editor-Fenster auf Run und dann auf Run Module.**

Das Fenster der Python-Shell springt nach vorne und zeigt die Ausgabe deines Programms in den letzten Zeilen.




*Hast du bemerkt, dass es print-Anweisungen mit und ohne Anführungszeichen gibt? Die Anführungszeichen setzt man immer dann, wenn genau dieser Text ausgegeben werden soll. Bei dem Rechenausdruck brauchst du die Anführungszeichen nicht, weil der Computer nicht die Rechnung, sondern das Rechenergebnis ausgeben soll.*

Wenn du anschließend noch eine Änderung in deinem Programm vornimmst, wirst du vor der Ausführung gefragt, ob das Programm gespeichert werden soll. Klicke dann auf OK. Probiere das mal aus.

## Altes Programm laden

Und was musst du tun, um dein Programm Tage (oder Jahre) später zu bearbeiten oder einfach nur wieder einmal auszuführen?

1 **Klicke im Dateimanager (Windows Explorer) mit der rechten Maustaste auf den Programmnamen.**

**2** **Wähle Edit with IDLE und anschließend Edit with IDLE 3.10 aus.**

Dann öffnet sich wieder das Editor-Fenster mit dem Programm.

Eine andere Möglichkeit, ein gespeichertes Programm im Editor zu öffnen, besteht darin, erst die Python-Shell zu öffnen (über das Python-Symbol auf dem Bildschirm), dort File und Open anzuklicken und die Datei auszuwählen.

*Wenn es dir schwerfällt, das Shell-Fenster und das Editor-Fenster zu unterscheiden und zu finden, schau einfach auf die Titelleiste. Im Shell-Fenster steht* IDLE Shell 3.10.1, *im Editor-Fenster steht der Dateiname.*

## Das kannst du jetzt

In diesem Kapitel hast du Folgendes gelernt:

- wie du Python auf deinem Computer installierst
- wie du Texte in die Python-Shell schreiben lässt
- wie du in der Python-Shell mit Zahlen rechnest
- wie du Python-Programme im Editor schreibst
- wie du Kommentare zu Programmen hinzufügst
- wie du Programme speichern, öffnen und ausführen kannst