HANS-GEORG SCHUMANN

PYTHON

FÜR KIDS

3. AUFLAGE

PROGRAMMIEREN LERNEN
OHNE VORKENNTNISSE

mitp

FÜR
JANNE, JULIA, KATRIN UND DANIEL

Hans-Georg Schumann

PYTHON FÜR KIDS

PROGRAMMIEREN LERNEN OHNE VORKENNTNISSE

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

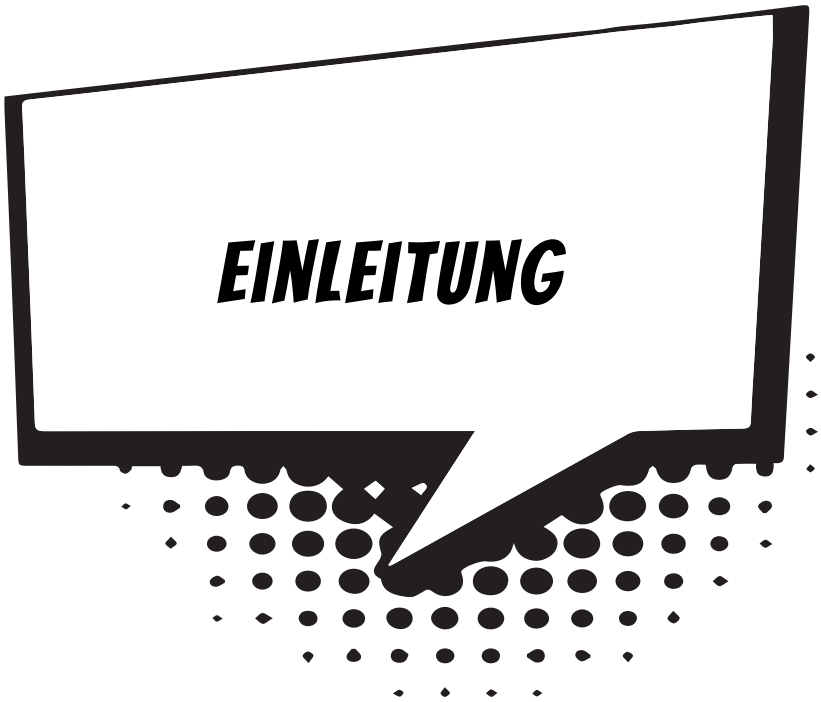
ISBN 978-3-7475-0620-2
3. Auflage 2023

www.mitp.de

E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Katja Völpel
Covergestaltung: Christian Kalkert
Satz: III-satz, Kiel, www.drei-satz.de

INHALT



EINLEITUNG

Python – wer denkt da nicht an eine Schlange? Eine, die zwar nicht giftig ist, aber einem sämtliche Knochen brechen und die Luft abschnüren kann. Du weißt natürlich, dass es hier bei Python um eine Programmiersprache geht.

Python wurde Anfang der 1990er-Jahre entwickelt. Der Name dieser Sprache geht jedoch nicht auf die gleichnamige Schlange zurück, sondern auf eine Truppe von englischen Komikern namens »Monthy Python«, die vor allem in den 70er-Jahren mit ihren skurrilen Filmen erfolgreich war (unter anderem mit »Das Leben des Brian«).

Natürlich hat Python viel von anderen Programmiersprachen übernommen. Sie hat sich einen Namen gemacht, weil sie als leicht erlernbar und übersichtlich gilt. In diesem Buch geht es um die bereits dritte Version von Python, die aktuell auch die neueste (und vielseitigste) ist.

WAS HEIßT EIGENTLICH PROGRAMMIEREN?

Wenn du aufschreibst, was ein Computer tun soll, nennt man das Programmieren. Das Tolle daran ist, dass du selbst bestimmen kannst, was getan werden soll. Lässt du dein Programm laufen, macht der Computer die Sachen, die du ausgeheckt hast.

Natürlich wird er dann dein Zimmer nicht aufräumen und dir auch keine Tasse Kakao ans Bett bringen. Aber beherrschst du erst mal das Programmieren, kannst du den Computer sozusagen nach deiner Pfeife tanzen lassen.

Allerdings passiert es gerade beim Programmieren, dass der Computer nicht so will, wie du es gerne hättest. Meistens ist das ein Fehler im Programm. Das Problem kann aber auch irgendwo anders im Computer oder im Betriebssystem liegen. Das Dumme bei Fehlern ist, dass sie sich gern so gut verstecken, dass die Suche danach schon manchen Programmierer zur Verzweiflung gebracht hat.

Vielleicht hast du nun trotzdem Lust bekommen, das Programmieren zu erlernen. Dann brauchst du ja nur noch eine passende Entwicklungsumgebung, und schon kann's losgehen.

WAS IST EINE ENTWICKLUNGSUMGEBUNG?

Um ein Programm zu erstellen, musst du erst einmal etwas eintippen. Das ist wie bei einem Brief oder einer Geschichte, die man schreibt. Das Textprogramm dafür kann sehr einfach sein, weil es ja nicht auf eine besondere Schrift oder Darstellung ankommt wie bei einem Brief oder einem Referat. So etwas wird **Editor** genannt.

Ist das Programm eingetippt, kann es der Computer nicht einfach lesen und ausführen. Jetzt muss es so übersetzt werden, dass der PC versteht, was du von ihm willst. Weil er aber eine ganz andere Sprache spricht als du, muss ein Dolmetscher her.

Du programmierst in einer Sprache, die du verstehst, und der Dolmetscher übersetzt es so, dass es dem Computer verständlich wird. So etwas heißt dann **Compiler** oder **Interpreter**.

Python bietet solche Dolmetscher gleich für mehrere Betriebssysteme. Dein Computer kann also ein Windows-PC oder ein Linux-PC sein, ein Macintosh oder irgendein anderes Computersystem. Ein und dasselbe Python-Programm kann so (eventuell mit kleinen Abweichungen) auf jedem beliebigen Computer funktionieren.

Schließlich müssen Programme getestet, überarbeitet, verbessert, wieder getestet und weiterentwickelt werden. Dazu gibt es noch einige zusätzliche Hilfen. Daraus wird dann ein ganzes System, die Entwicklungsumgebung.

WARUM GERADE PYTHON?

Leider kannst du nicht so programmieren, wie dir der Schnabel gewachsen ist. Eine Programmiersprache muss so aufgebaut sein, dass möglichst viele Menschen in möglichst vielen Ländern einheitlich damit umgehen können.

Weil in der ganzen Welt Leute zu finden sind, die wenigstens ein paar Brocken Englisch können, besteht auch fast jede Programmiersprache aus englischen Wörtern. Es gab auch immer mal Versuche, z.B. in Deutsch zu programmieren, aber meistens klingen die Wörter dort so künstlich, dass man lieber wieder aufs Englische zurückgreift.

Eigentlich ist es egal, welche Programmiersprache du benutzt. Am besten eine, die möglichst leicht zu erlernen ist. Wie du weißt, bekommst du es in diesem Buch mit der Programmiersprache Python zu tun, die mittlerweile sehr weit verbreitet ist. (Willst du mal in andere Sprachen hineinschnuppern, dann empfehle ich dir z.B. eines der anderen Kids-Bücher über C++ oder Java.)

Der Weg zum guten Programmierer kann ganz schön steinig sein. Nicht selten kommt es vor, dass man die Lust verliert, weil einfach gar nichts klappen will. Das Programm tut etwas ganz anderes, man kann den Fehler nicht finden und man fragt sich: Wozu soll ich eigentlich programmieren lernen, wo es doch schon genug Programme gibt?

Gute Programmierer werden immer gesucht, und dieser Bedarf wird weiter steigen. Und Python gehört dabei durchaus zu den erwünschten Sprachen. Wirklich gute Programmierer werden auch wirklich gut bezahlt. Es ist also nicht nur einen Versuch wert, es kann sich durchaus lohnen, das Programmieren in Python zu erlernen.

DIE ENTWICKLUNGSUMGEBUNG

Um eine Entwicklungsumgebung für Python musst du dich nicht weiter kümmern, wenn dir eine einfache reicht. Die nämlich bekommst du kostenlos mit dem Python-Paket (sie heißt IDLE = »Integrated Development and Learning Environment«). Die werden wir hier ausgiebig benutzen.

Das komplette Paket kannst du dir von dieser Seite herunterladen:

https://www.python.org/

Dabei muss es nicht unbedingt die neueste Version sein. Dieses Buch bezieht sich auf Python 3 (aktuelle Versionen 3.8 oder 3.9).

UND WAS BIETET DIESES BUCH?

Über eine ganze Reihe von Kapiteln verteilt lernst du

- das Basiswissen von Python kennen
- etwas über objektorientierte Programmierung
- mit Komponenten des Moduls `tkinter` zu arbeiten (das sind Bausteine, mit denen du dir viel Programmierarbeit sparen kannst)

- die grafischen Möglichkeiten von Python kennen
- etwas über den Umgang mit dem Spiele-Modul `pygame`
- wie man eigene `Game`- und `Player`-Klassen programmiert

Im **Anhang** gibt es dann noch zusätzliche Informationen und Hilfen, unter anderem über Installationen und den Umgang mit Fehlern.

WIE ARBEITE ICH MIT DIESEM BUCH?

Grundsätzlich besteht dieses Buch aus einer Menge Text mit vielen Abbildungen dazwischen. Natürlich habe ich mich bemüht, alles so zuzubereiten, dass daraus lauter gut verdauliche Happen werden. Damit das Ganze noch genießbarer wird, gibt es zusätzlich noch einige Symbole, die ich dir hier gern erklären möchte:

ARBEITSSCHRITTE

≫ Wenn du dieses Zeichen siehst, heißt das: Es gibt etwas zu tun. Damit kommen wir beim Programmieren Schritt für Schritt einem neuen Ziel immer näher.

Grundsätzlich lernt man besser, wenn man einen Programmtext selbst eintippt oder ändert. Aber nicht immer hat man große Lust dazu. Deshalb gibt es alle Projekte im Buch auch als Download:

http://www.mitp.de/0619

Und hinter einem Programmierschritt findest du auch den jeweiligen Namen des Projekts oder einer Datei (z. B. → PROJEKT1.PY). Wenn du also das Projekt nicht selbst erstellen willst, kannst du stattdessen diese Datei laden (zu finden im Ordner PROJEKTE).

AUFGABEN

Am Ende eines Kapitels findest du jeweils eine Reihe von Fragen und Aufgaben. Diese Übungen sind nicht immer ganz einfach, aber sie helfen dir, noch besser zu programmieren. Lösungen zu den Aufgaben findest du in verschiedenen Formaten ebenfalls im Verzeichnis PROJEKTE. Du kannst sie dir alle im Editor von Windows oder auch in deinem Textverarbeitungsprogramm anschauen. Oder du lässt sie dir ausdrucken und hast sie dann schwarz auf weiß, um sie neben deinen Computer zu legen. (Auch die Programme zu den Aufgaben liegen im Ordner PROJEKTE.)

NOTFÄLLE

Vielleicht hast du irgendetwas falsch gemacht oder etwas vergessen. Oder es wird gerade knifflig. Dann fragst du dich, was du nun tun sollst. Bei diesem Symbol findest du eine Lösungsmöglichkeit. Notfalls kannst du aber auch ganz hinten im Anhang B nachschauen, wo ein paar Hinweise zur Pannenhilfe aufgeführt sind.

WICHTIGE STELLEN IM BUCH

Hin und wieder findest du ein solch dickes Ausrufezeichen im Buch. Dann ist das eine Stelle, an der etwas besonders Wichtiges steht.

EXPERTENWISSEN

Wenn du ein solches »Wow« siehst, geht es um ausführlichere Informationen zu einem Thema.

WAS BRAUCHST DU FÜR DIESES BUCH?

Installiert wird Python mit einem Setup-Programm in ein Verzeichnis deiner Wahl, z. B. D:\PYTHON. Dort solltest du später auch deine Python-Projekte unterbringen.

Die Beispielprogramme in diesem Buch gibt es alle als Download von der Homepage des Verlages, falls du mal keine Lust zum Abtippen hast:

http://www.mitp.de/0619

Und auch die Lösungen zu den Fragen und Aufgaben sind dort untergebracht (alles im Ordner PROJEKTE).

BETRIEBSSYSTEM

Die meisten Computer arbeiten heute mit dem Betriebssystem Windows. Davon brauchst du eine der Versionen 7 bis 11. (Python gibt es unter anderem auch für Linux.)

SPEICHERMEDIEN

Auf jeden Fall benötigst du etwas wie einen USB-Stick oder eine SD-Card, auch wenn du deine Programme auf die Festplatte speichern willst. Auf einem externen Speicher sind deine Arbeiten auf jeden Fall zusätzlich sicher aufgehoben.

Gegebenenfalls bitte deine Eltern oder Lehrer um Hilfe.

HINWEISE FÜR LEHRER

Dieses Buch versteht sich auch als Lernwerk für den Informatik-Unterricht in der Schule. Dort setzt natürlich jeder Lehrer seine eigenen Schwerpunkte. Benutzen Sie an Ihrer Schule bereits ein Werk aus einem Schulbuchverlag, so lässt sich dieses Buch auch als Materialienband einsetzen – in Ergänzung zu dem vorhandenen Schulbuch. Weil dieses Buch sozusagen »von null« anfängt, ist ein direkter Einstieg in Python möglich – ohne irgendwelche anderen Programmierkenntnisse.

Ein wichtiger Schwerpunkt in diesem Buch ist die objektorientierte Programmierung (OOP). Auf die wichtigsten Eigenheiten (Kapselung, Vererbung und Polymorphie) wird ausführlich eingegangen. Ein anderer Schwerpunkt ist die Programmierung von Spielen. In den Projekten werden alle wesentlichen Elemente des Python-Wortschatzes wie auch die wichtigsten Grafik-Komponenten von `tkinter` eingesetzt. In den Lösungen zu den Aufgaben finden Sie weitere Vorschläge zur Programmierung.

ÜBUNGSMEDIEN

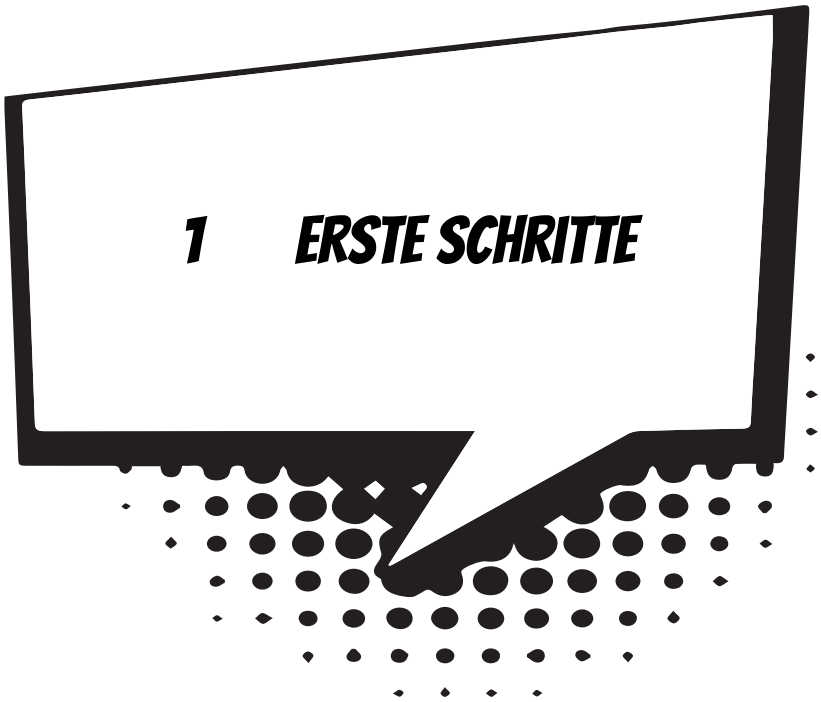
Für den Informatik-Unterricht sollte jeder Schüler ein eigenes externes Speichermedium haben, um darauf seine Programmierversuche zu sichern. So wird verhindert, dass sich auf der Festplatte des Schulcomputers mit der Zeit allerlei »Datenmüll« ansammelt. Außerdem dient der eigene Datenträger dem Datenschutz: Nur der betreffende Schüler kann seine Daten manipulieren.

AUF DIE DATEIEN ZUM BUCH VERZICHTEN?

Vielleicht ist es Ihnen lieber, wenn Ihre Schüler die Projekte alle selbst erstellen. Dann lassen Sie die Download-Dateien einfach (erst einmal) weg.

REGELMÄßIG SICHERN

Es kann nicht schaden, die Programmdateien, an denen gerade gearbeitet wird, etwa alle zehn Minuten zu speichern. Denn Computer pflegen gern gerade dann »abzustürzen«, wenn man seine Arbeit längere Zeit nicht gespeichert hat.



1 ERSTE SCHRITTE

Hier geht es gleich ans »Eingemachte«. Nachdem wir Python installiert und gestartet haben, machen wir unsere ersten Gehversuche. Um später auch größere Programmprojekte erstellen zu können, brauchen wir das passende Werkzeug. Wir richten uns so komfortabel ein, dass schließlich auch dein erstes Programm entsteht.

In diesem Kapitel lernst du

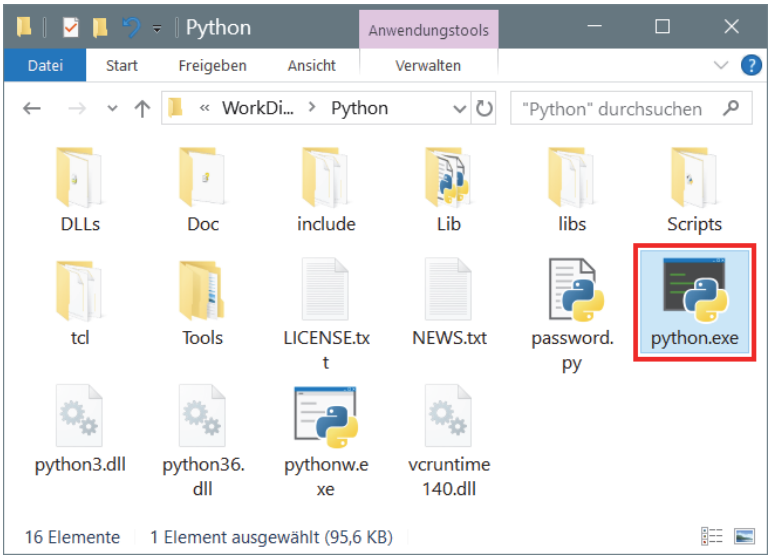
- wie man Python startet
- Anweisungen für Ausgabe und Eingabe kennen
- was Variablen sind
- den Typ String kennen
- etwas über den Einsatz von IDLE
- wie man ein Programm erstellt und speichert
- wie man Python beendet

MIT PYTHON LOSLEGEN

Bevor wir mit dem Programmieren anfangen können, muss **Python** erst installiert werden.

Die Installation übernimmt ein sogenanntes Setup-Programm. Genaues erfährst du im Anhang A. Hier musst du dir von jemandem helfen lassen, wenn du dir die Installation nicht allein zutraust. Eine Möglichkeit, Python zu starten, ist diese:

≫ Öffne den Ordner, in dem du Python untergebracht hast – z.B. C:\PROGRAMME\PYTHON oder D:\PYTHON.



≫ Hier suchst du unter den vielen Symbolen das mit dem Namen PYTHON.EXE heraus. Doppelklicke auf das Symbol.

Wenn du willst, kannst du auch eine **Verknüpfung** auf dem Desktop anlegen:

- Dazu klickst du mit der rechten Maustaste auf das Symbol für Python (PYTHON.EXE). Im Kontextmenü wählst du KOPIEREN.
- Dann klicke auf eine freie Stelle auf dem Desktop, ebenfalls mit der rechten Maustaste. Im Kontextmenü wählst du VERKNÜPFUNG EINFÜGEN.
- Es ist sinnvoll, für das neue Symbol auf dem Desktop den Text `python.exe - Verknüpfung` einfach durch `Python` zu ersetzen.

Von nun an kannst du auf das neue Symbol **doppelklicken**, um die Arbeitsumgebung von Python direkt zu starten.

Was dich nach dem Start erwartet, sieht etwa so aus:

Die ersten beiden Zeilen informieren dich unter anderem über die aktuelle Python-Version, aber du bekommst auch schon ein paar Befehle vorgeschlagen, die du hinter den drei spitzen Klammern (`>>>`) eintippen kannst.

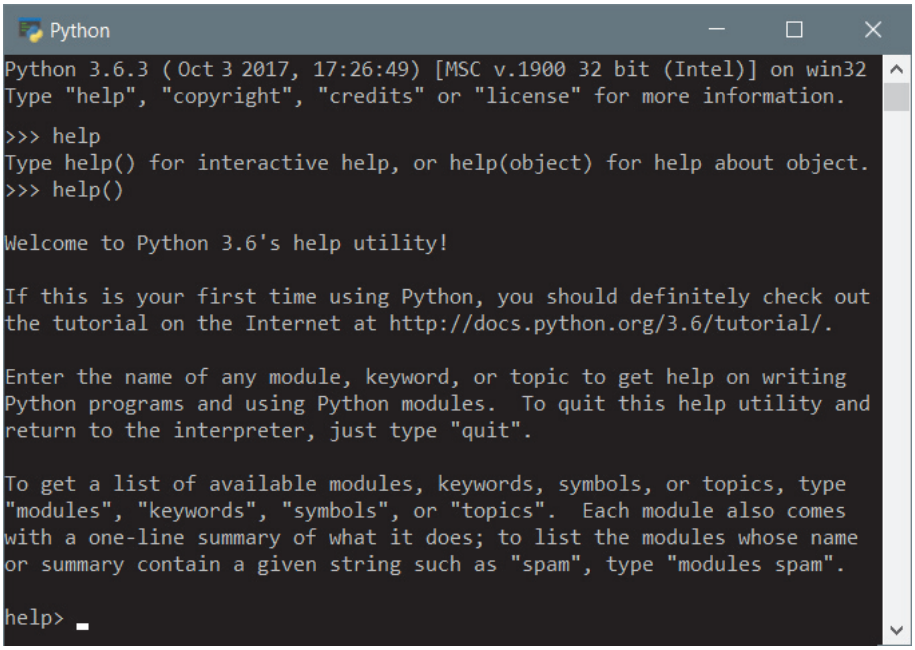
Die drei Zeichen werden hier auch **Prompt** genannt. Das ist eine Art Eingabeaufforderung, weil du dahinter etwas eingeben kannst (und musst, wenn es weitergehen soll).

≫ Probieren wir es doch gleich einmal mit »help«. Tippe dieses Wort ein.

Und prompt gibt es etwas zu meckern: Na ja, es ist eher ein netter Hinweis: Man muss `help` mit zwei runden Klammern dahinter eintippen.

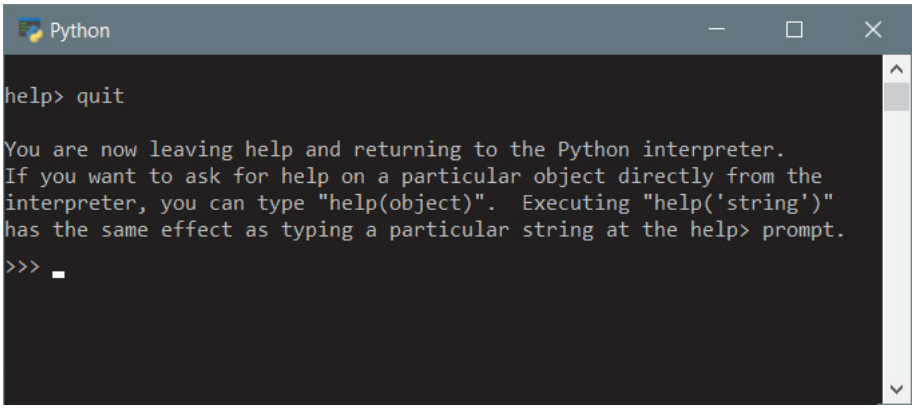
≫ Gib also `help()` ein.

Und du bekommst gleich eine ganze Menge Text serviert:



Nun steht da `help>` als Prompt. Du kannst dahinter ein Wort eingeben, und wenn es zum Python-Wortschatz gehört, bekommst du dazu eine (kurze) Erläuterung.

≫ Um zum ursprünglichen Prompt zurückzukehren, tippe `quit` ein.



Und du bist wieder zurück im Python-Interpreter.

Was ist ein **Interpreter**? Zuerst solltest du wissen, dass das, was du als Befehl hinter dem Prompt eintippst, für den Computer erst einmal völlig unverständlich ist. Normalerweise kann er also den jeweiligen Befehl gar nicht ausführen.

Ein Interpreter übersetzt die Befehlszeile in eine Sprache, die der Computer versteht, sodass er den Befehl ausführen kann – genannt **Maschinensprache**. Bei einem Programm, das aus einigen bis sehr vielen Zeilen bestehen kann, wird von einem Interpreter jede Zeile **einzeln** übersetzt und dann ausgeführt.

Im Gegensatz dazu gibt es **Compiler**, die das **gesamte** Programm in Maschinensprache übersetzen. Erst wenn das Programm komplett und fehlerfrei ist, kann es vom Computer ausgeführt werden. Für Python benutzen wir hier einen Interpreter, es gibt aber auch Python-Compiler.

ZAHLEN UND TEXT

Nun wollen wir aber endlich mal was ausprobieren.

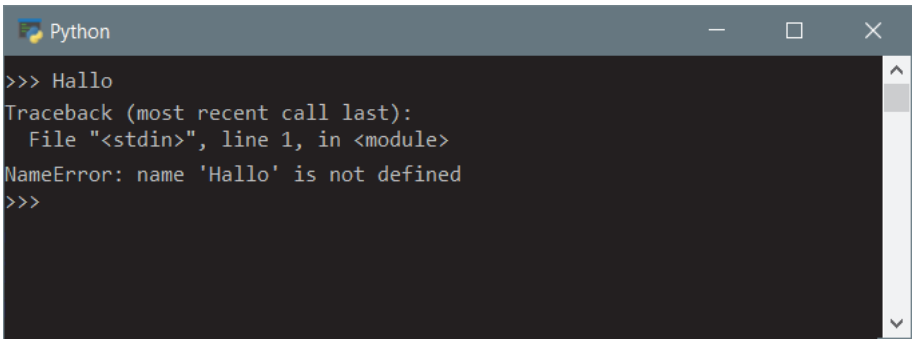
≫ Tippe also ein: `1+2+3` und drücke dann die ↵-Taste.

Und tatsächlich wird das Ergebnis dieser kleinen Matheaufgabe angezeigt.

≫ Du kannst gern ein paar weitere Aufgaben stellen und dabei auch die Operationszeichen für minus (-), mal (*) und geteilt durch (/) benutzen.

Na ja, als Taschenrechner scheint der Python-Interpreter ja gut zu funktionieren, aber natürlich erwartest du viel mehr als das.

≫ Versuchen wir es mal mit einem netten Gruß: Tippe `Hallo` ein und schließe das mit der ↵-Taste ab.



Na ja, irgendwie gibt es jetzt wirklich was zu meckern. »Error« heißt »Fehler«, demnach ist hier eindeutig etwas falsch. Was ist das Ziel? Ich möchte, dass der Computer ein freundliches »Hallo« sagt (bzw. schreibt).

≫ Dazu tippe jetzt mal folgende Zeile ein:

≫ `print("Hallo")`

Das geht. Dabei bedeutet `print()` hier anzeigen, ausgeben. Und in den runden Klammern dahinter steht das, was angezeigt werden soll. Das nennt man **Parameter**.

Natürlich geht das auch mit Zahlen:

≫ Probiere selber aus, was der `print`-Befehl alles ausgeben kann.

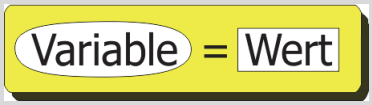
Nun wird es ein bisschen komplizierter. Bis jetzt haben wir immer nur einen Befehl eingegeben. Aber richtige Programme bestehen natürlich aus mehr als nur einer Zeile. Versuchen wir es mal mit diesem kleinen Programmstück:

```
Text = "Hallo"
print(Text)
```

≫ Gib diese beiden Zeilen ein. Wird angezeigt, was du erwartet hast?

Nach der ersten Zeile gibt es noch nichts anzuzeigen. Aber offenbar hat sich der Python-Interpreter gemerkt, was `Text` bedeutet. Und er weiß, welchen Wert `print()` als Parameter übernehmen soll.

Genauer: Bei `Text` handelt es sich um eine sogenannte **Variable**, der wird ein Wert zugewiesen, in diesem Fall ist das das Wörtchen `"Hallo"`. Und das Gleichheitszeichen (=) wird **Zuweisungsoperator** genannt.



In Python werden Variablen neu **erzeugt**, wenn ihnen zum ersten Mal ein Wert zugewiesen werden soll. Bei einer Zuweisung steht **immer** links die Variable und rechts der Wert, das Gleichheitszeichen hat also quasi die Bedeutung eines Pfeils:

`Variable ← Wert`

Variablen sind nützlich, weil sie Daten »aufheben«, sodass der Computer sich an einen Wert erinnern kann. Vor allem in größeren Programmen ist es wichtig, dass der Inhalt einer Variablen auch mehrmals benutzt werden kann. Einige Beispiele dafür wirst du noch kennenlernen.

Nun hat der Computer so schön »Hallo« gesagt, das könnte man doch noch um einigen Text erweitern:

```
Text = "Hallo, wer da?"
print(Text)
Name = input()
print(Name)
```

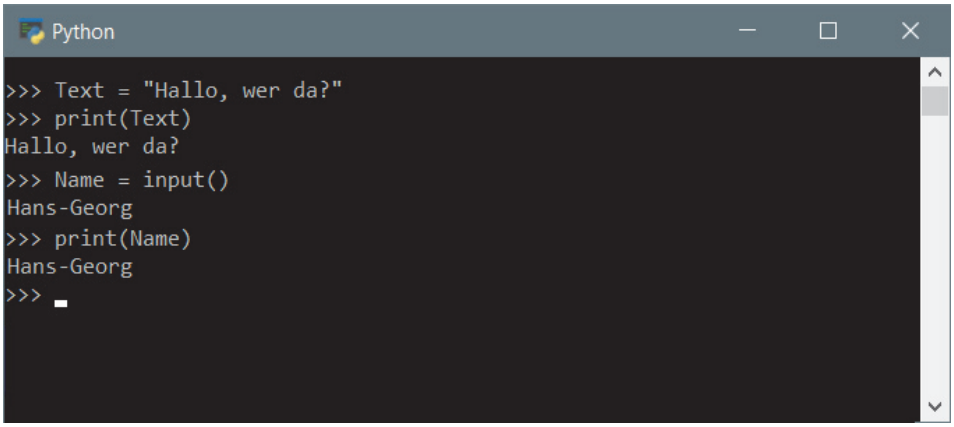
In Python muss ein Text wie unser Hallo-Gruß immer in **Anführungszeichen** gefasst werden. Ich benutze hier die doppelten ("), aber auch die einfachen (') sind erlaubt.

Diese beiden Zeilen sind also völlig gleichwertig:

```
Text = "Hallo, wer da?"
Text = 'Hallo, wer da?'
```

≫ Tippe alle diese Zeilen nacheinander ein. Beachte, dass du nach der `input`-Zeile erst selber deinen Namen eingeben musst, ehe es weitergeht.

Bei mir sieht das Ganze so aus:



Gar nicht so schlecht. Und dabei hast du gleich einen neuen Befehl kennengelernt: `input()` heißt hier eingeben. Damit lässt sich doch schon was anfangen. Wenn man oft genug `print()` und `input()` benutzt, kann man schon ein ansehnliches Gespräch mit dem Computer führen.

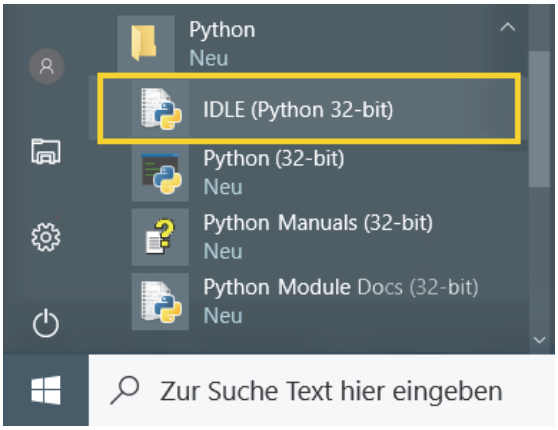
Aber etwas gefällt mir nicht. Zum Beispiel das dauernde Neueingeben. Man kann nicht einfach beliebig mit den Pfeiltasten im Programm herumwandern oder mit der Maus irgendwohin klicken und dort Text ändern. Besser wäre es doch, wenn man sich im Python-Fenster wie in einem Texteditor bewegen könnte.

Unvorstellbar, dass wir auf diese Weise größere Programmprojekte erstellen wollen. Dazu muss man auch die Möglichkeit haben, den mühsam eingetippten Text irgendwo als Datei zu speichern. Das aktuelle Werkzeug, das wir mit dem Eingabe-Fenster haben, reicht also offenbar nicht aus.

EINE ARBEITSUMGEBUNG NAMENS IDLE

Wir brauchen also ein Fenster, über das man eingegebenen Text auch speichern und dorthin wieder laden kann. Genannt Editor. Im Python-Paket ist ein solcher Editor bereits enthalten, man muss ihn nur finden.

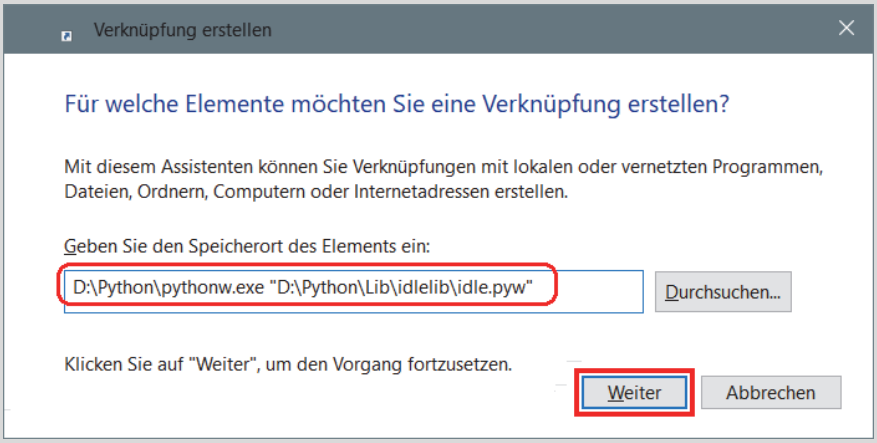
Wenn bei der Installation im START-Menü von Windows eine Verknüpfung zu Python eingerichtet wurde, dann findest du dort auch einen Eintrag wie IDLE (PYTHON).



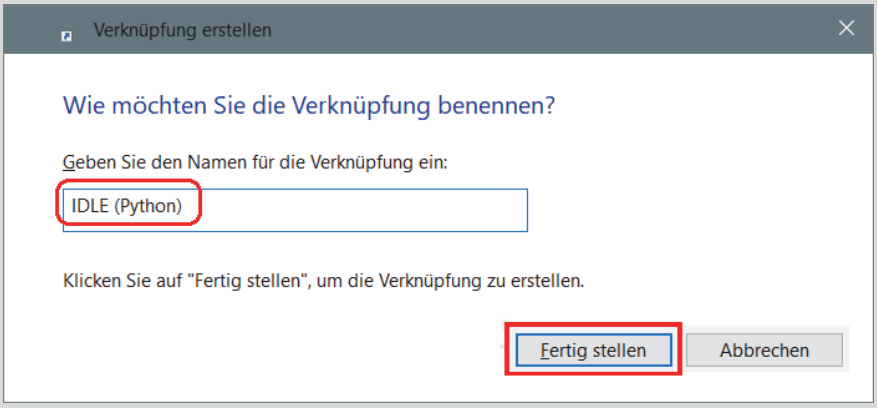
≫ Klicke darauf, um dieses Programm zu starten.

Wenn du diesen Eintrag nicht in deinem Start-Menü findest, dann kannst du dir selber eine Verknüpfung auf dem Desktop erstellen.

- Klicke mit der rechten Maustaste auf eine freie Stelle im Desktop und wähle im Kontextmenü NEU und dann VERKNÜPFUNG.
- Im Dialogfeld gibst du als SPEICHERORT DES ELEMENTS ein:
  `D:\Python\pythonw.exe "D:\Python\Lib\idlelib\idle.pyw"`

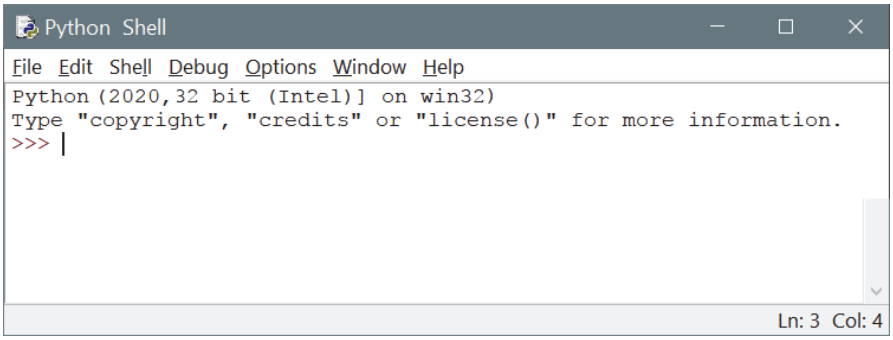


- Wichtig ist, dass D:\PYTHON auch das Verzeichnis ist, in das du Python installiert hast. Sonst musst du das entsprechend anpassen.
- Nenne das neue Symbol auf dem Desktop `IDLE (Python)`.



Damit kannst du von nun an den Python-Editor direkt von Desktop aus per **Doppelklick** starten.

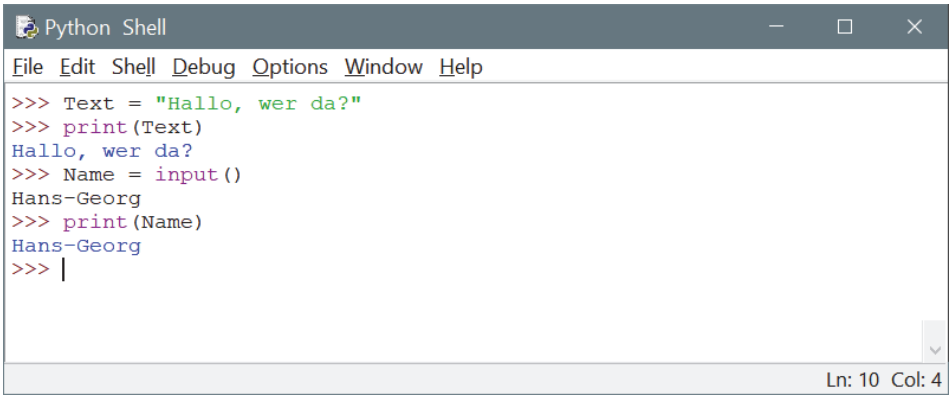
Nach dem Start von IDLE findest du dich in einem solchen Fenster wieder:



Sieht irgendwie ähnlich aus wie das Fenster des Python-Interpreters, und irgendwie auch anders. Nicht nur, dass es hier statt weiß auf schwarz umgekehrt zugeht, hier gibt es auch eine Menüleiste. Und das Ganze nennt sich »Shell«.

IDLE ist die Abkürzung für »Integrated Development and Learning Environment«, frei übersetzt ist das eine Umgebung in diesem Fall für das Entwickeln und Lernen von Python-Programmen. Der Begriff **Shell** bedeutet »Schale« und zielt in dieselbe Richtung.

Wir können auch in dieser Umgebung unsere Python-Befehle eintippen, dann bekommen wir dieselben Ergebnisse wie ganz oben (im »schwarzen« Fenster):



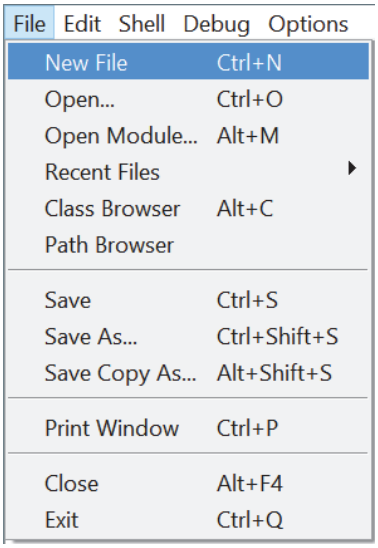
Allerdings sieht es hier etwas bunter aus. Wörter wie `print` und `input` werden farbig angezeigt, ebenso wie in Anführungsstriche gesetzter Text.

≫ Probiere das selber aus, indem du die Zeilen von oben auch hier noch mal eingibst.

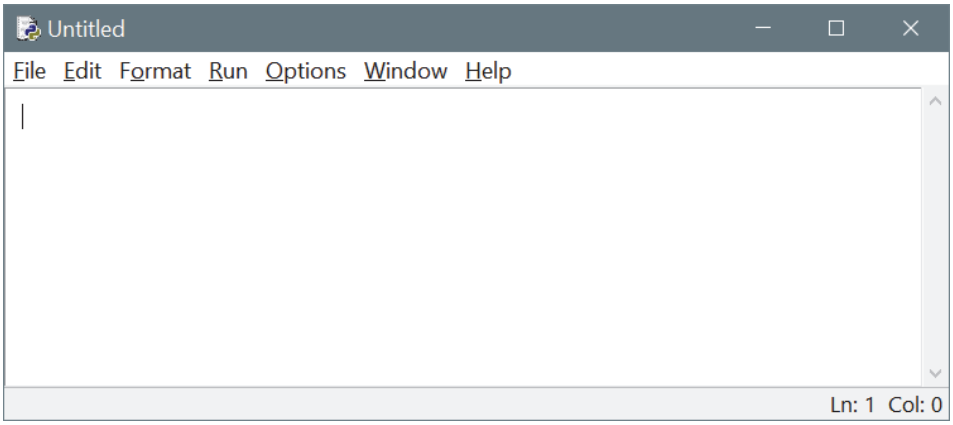
DIE ERSTE PY-DATEI

Wie kriegen wir es nun hin, dass aus den paar Zeilen ein komplettes Programm wird, das sich immer wieder laden und ausführen lässt?

≫ Klicke im Menü auf FILE und dann auf NEW FILE (oder drücke die Tastenkombination Strg + N).



Und schon haben wir ein weiteres (neues) Fenster mit dem Titel UNTITLED. Die Menüleiste ist ähnlich wie vorher, ansonsten ist das Fenster leer.

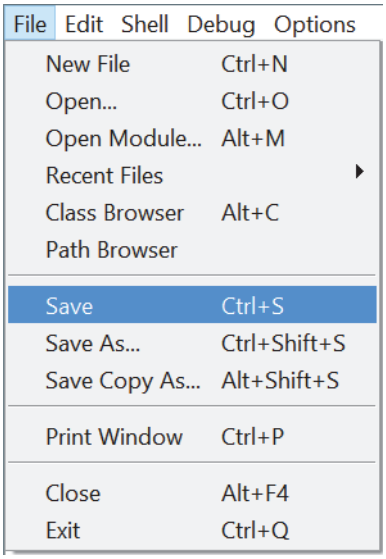


Hier können wir nun unser Programm eingeben, ohne dass irgendeine Zeile davon direkt vom Python-Interpreter ausgeführt wird.

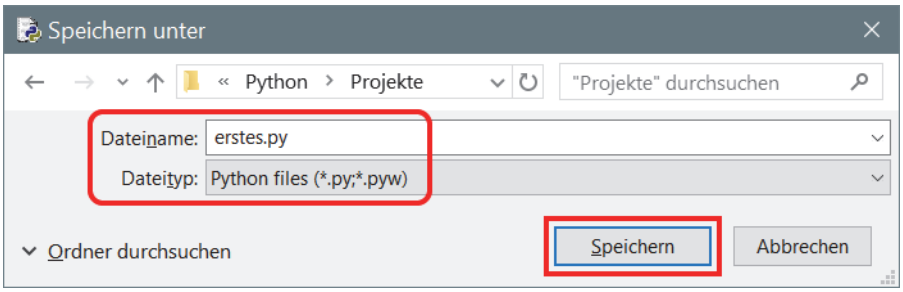
≫ Tippe also diese Zeilen ein:

```
Text = "Hallo, wer da?"
print(Text)
Name = input()
print(Name)
```

≫ Und damit du sie nicht wieder verlierst (nachdem du sie ja jetzt so oft eingeben musstest), klicke nun auf FILE und SAVE oder SAVE AS.

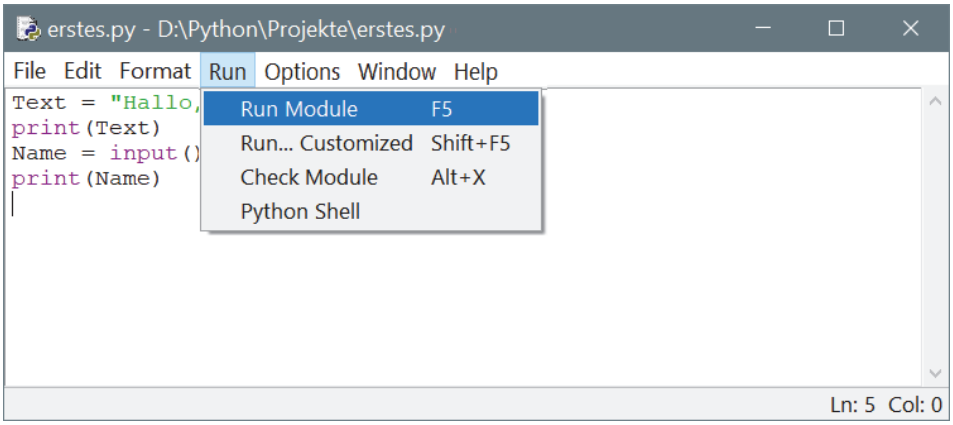


≫ Im Dialogfeld gibst du einen Namen für dieses Programm ein, z.B. `erstes` (oder was du willst). Das PY wird als Kennung für »Python« automatisch angehängt, wenn du es nicht angibst.



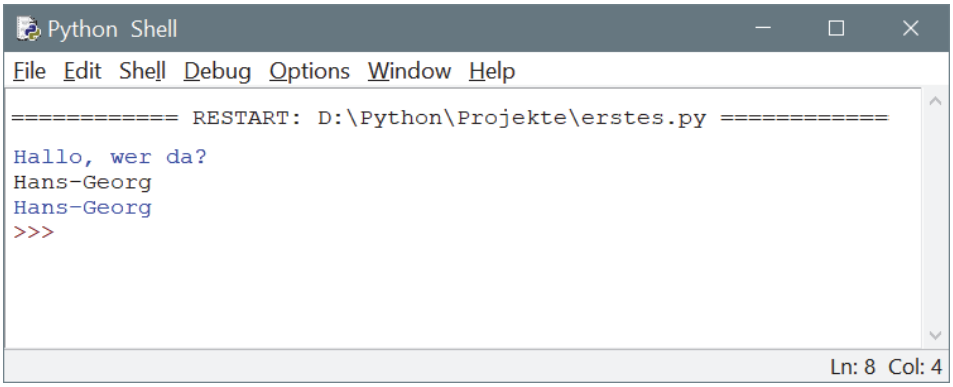
Ich habe im PYTHON-Verzeichnis einen Unterordner namens PROJEKTE erstellt und speichere meine Python-Projekte dort ab.

≫ Um das Programm zu starten, klickst du jetzt auf RUN und RUN MODULE. Oder du drückst die Taste F5.



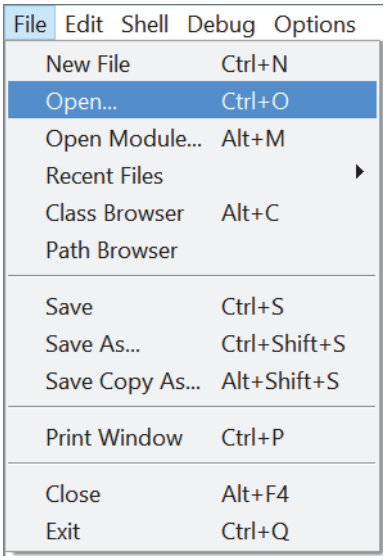
Und du landest wieder im ersten Fenster, wo dich der Gruß-Text »Hallo, wer da?« erwartet.

≫ Tippe nun deinen Namen ein und bestätige das mit der ↵-Taste. Dann könnte das Ergebnis so ähnlich aussehen:

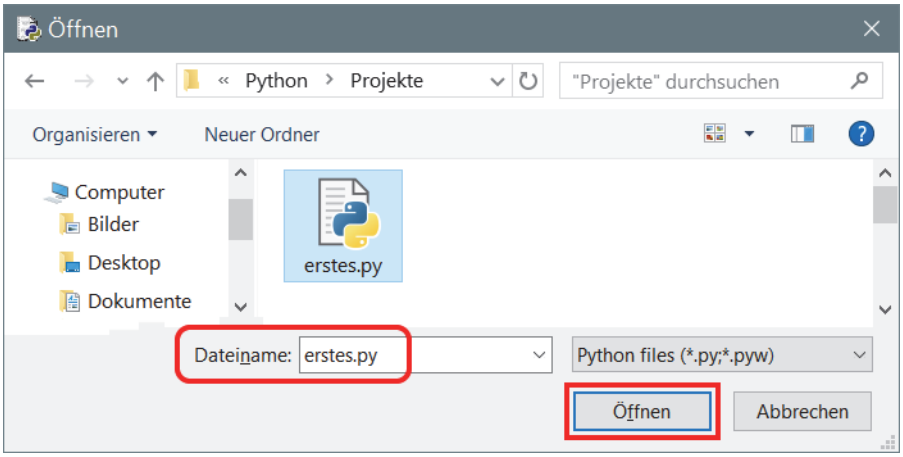


Wenn du nun dieses Fenster schließt, gibt es deine Programmdatei immer noch. Du musst sie dir einfach nur wieder zurückholen.

≫ Dazu klickst du auf FILE und OPEN.



≫ Im Dialogfeld wählst du die betreffende Datei (bei mir im Ordner PYTHON\PROJEKTE).



≫ Klicke auf ÖFFNEN, das Editor-Fenster öffnet sich und dein erstes Programm ist wieder verfügbar.

In Python wird ein Programm in einer Datei auch **Skript** genannt.

QUELLTEXT-SPIELEREIEN

Schauen wir uns jetzt diesen **Quelltext**, wie man die Summe der Textzeilen auch nennt, noch einmal näher an:

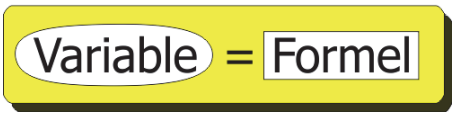
```
Text = "Hallo, wer da?"
print(Text)
Name = input()
print(Name)
```

Es gibt zwei Variablen `Text` und `Name`. In denen wird jeweils eine sogenannte Zeichenkette gespeichert, auch **String** genannt. Außerdem werden hier zwei Funktionen benutzt:

`print()`	Ausgabe von Zahlen und Strings auf dem Monitor
`input()`	Eingabe von Zahlen und Strings über die Tastatur

Wie du siehst, haben Funktionen immer runde Klammern als »Anhängsel«, in denen kann etwas drinstehen, sie können aber auch leer sein – je nach Art und Anwendung der Funktion.

Und wenn du noch etwas genauer hinschaust, dann fällt dir auf, dass `print()` direkt als Anweisung aufgeführt wird, `input()` aber wird in einer **Zuweisung** eingesetzt. Also wird über diese Funktion ein Wert an die Variable `Name` zugewiesen.



Dass ich hier die Bezeichnung »Formel« benutze, soll bedeuten, dass man auf der rechten Seite der Zuweisung außer Funktionen auch z.B. so etwas einsetzen kann:

```
Zahl = 1 + 2 * 3
Text = "Du bist also " + Name
```

Wobei das Plus (+) offenbar eine Doppelrolle hat: Man kann damit Zahlen addieren und Strings verketten.

Nachdem du jetzt weißt, wie man auch eine ganze Reihe von Programmzeilen sammeln und als Datei speichern kann, setzen wir doch gleich mal unser erstes Beispiel fort:

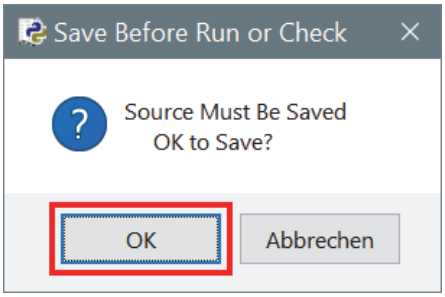
```
Text = "Hallo, wer da?"
print(Text)
Name = input()
Text = "Du bist also " + Name
print(Text)
print("Und wie geht es?")
Antwort = input()
print("Dir geht es also " + Antwort);
```

Wie du siehst, habe ich nicht überall Variablen benutzt, eigentlich ist das nur nötig, wenn man will, dass der Computer sich etwas merkt. Das betrifft in unserem Beispiel nur die Eingaben. Demnach kann unser Programmprojekt auch so aussehen:

```
print("Hallo, wer da?")
Name = input()
print("Du bist also " + Name)
print("Und wie geht es?")
Antwort = input()
print("Dir geht es also " + Antwort);
```

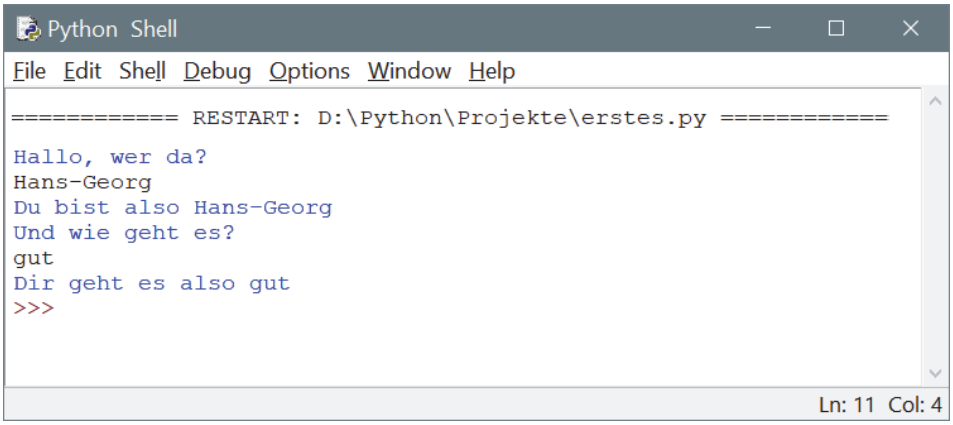
≫ Tippe erst den oberen Quelltext ein, probiere ihn über RUN und RUN MODULE (oder F5) aus. Dann ändere alles so, dass daraus die zweite Version wird. Lasse das Programm erneut laufen.

Bei jeder Änderung wirst du vor dem Programmstart aufgefordert, den Quelltext zu speichern:

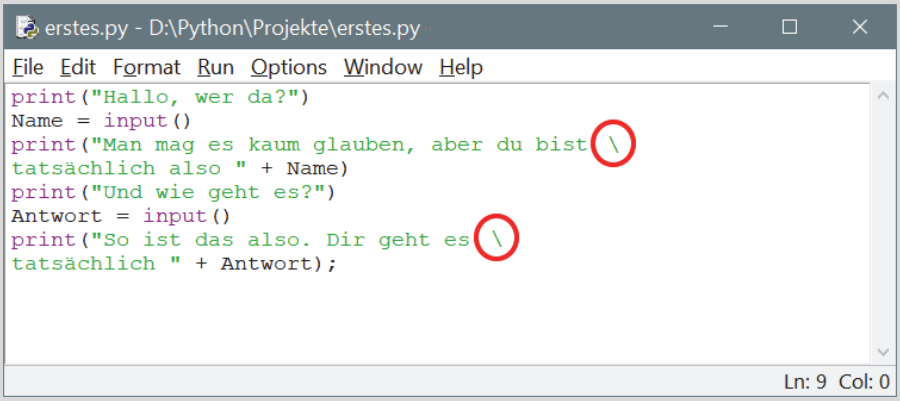


≫ Klicke dann auf OK.

Und so könnte unsere letzte Version im Python-Interpreter-Fenster ablaufen:



Wie du sehen kannst, ist Python zeilenorientiert: In **jeder** Zeile steht **eine** Anweisung (auch eine Zuweisung ist eine Anweisung). Man darf also Anweisungen nicht einfach auf zwei Zeilen verteilen. Und wenn das doch mal nötig sein sollte, muss am Ende der ersten Zeile ein sogenannter Backslash stehen, das ist ein umgekehrter Schrägstrich (\).



PYTHON VERLASSEN

Um die Python-Umgebung zu beenden, müssen alle offenen Fenster geschlossen werden:

≫ Das geht entweder über das jeweilige FILE-Menü und den Eintrag EXIT. Oder du klickst auf das kleine X oben rechts in der Titelleiste. Wenn du willst, kannst du auch die Tastenkombination Strg + Q benutzen.