

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

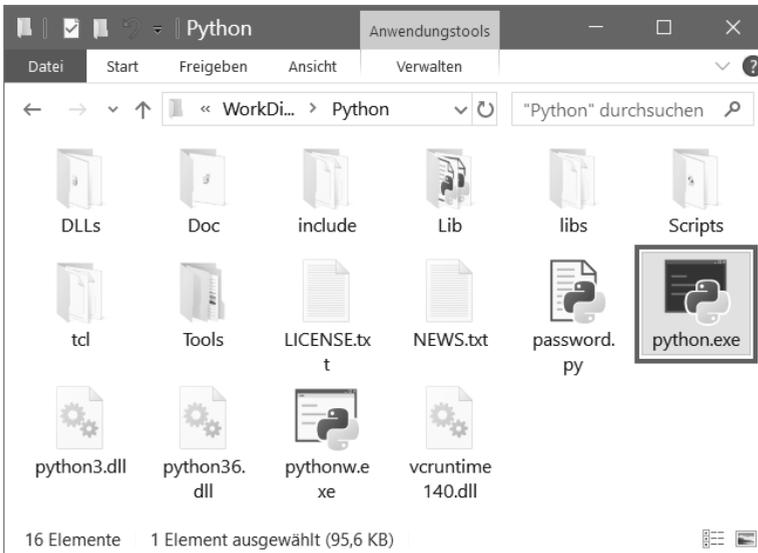
- ⊙ wie man Python startet
- ⊙ Anweisungen für Ausgabe und Eingabe kennen
- ⊙ was Variablen sind
- ⊙ den Typ String kennen
- ⊙ etwas über den Einsatz von IDLE
- ⊙ wie man ein Programm erstellt und speichert
- ⊙ wie man Python beendet

MIT PYTHON LOSLEGEN

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

Die Installation übernimmt ein sogenanntes Setup-Programm. Genaueres 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.

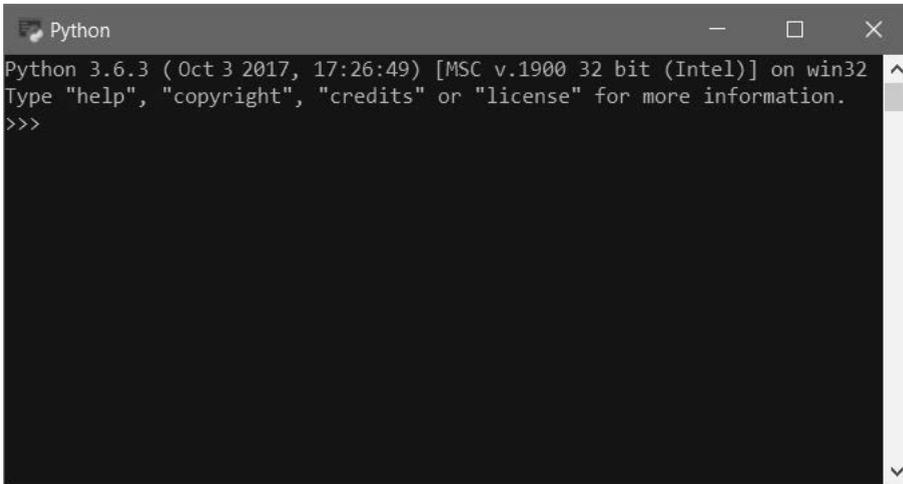
Wenn du willst, kannst du auch eine **Verknüpfung** auf dem Desktop anlegen:

- ◆ 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:



```
Python 3.6.3 (Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

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).

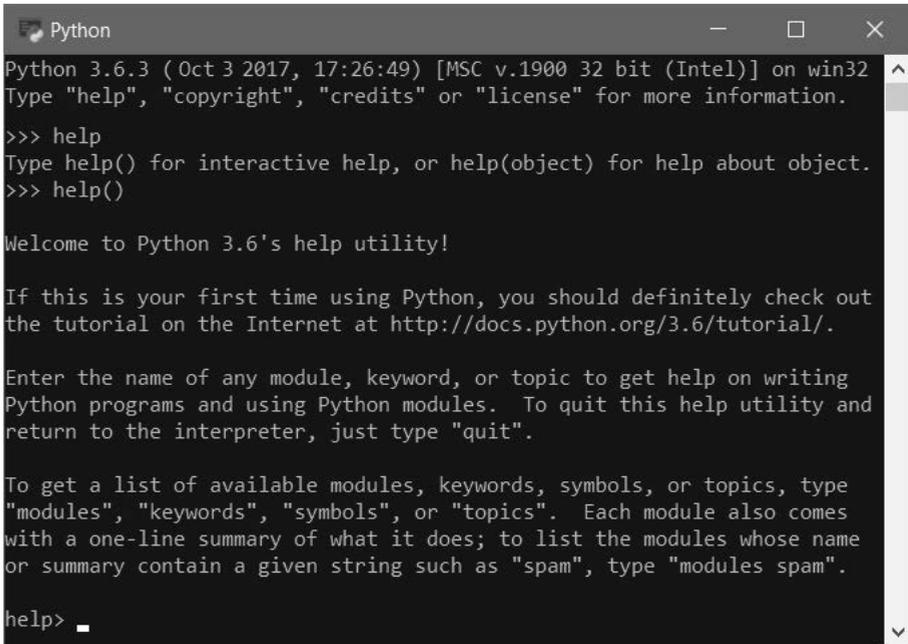


➤ Probieren wir es doch gleich einmal mit »help«. Tippe dieses Wort ein.

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:



```
Python 3.6.3 (Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> help
Type help() for interactive help, or help(object) for help about object.
>>> help()

Welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.6/tutorial/.

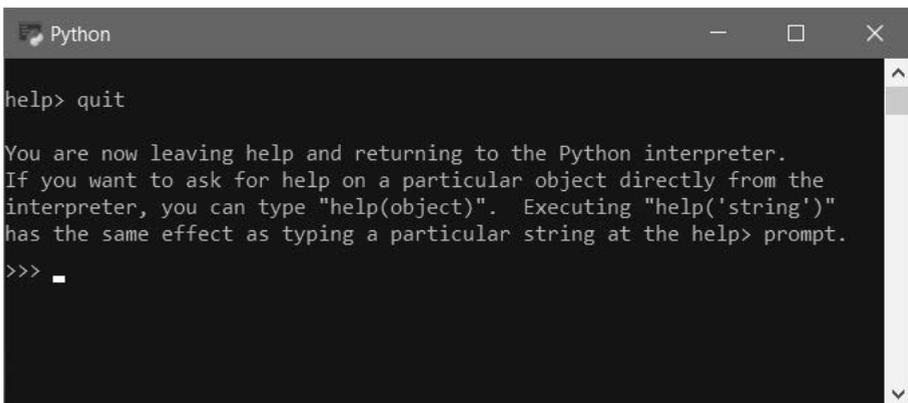
Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> _
```

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.



```
Python 3.6.3 (Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> help
Type help() for interactive help, or help(object) for help about object.
>>> help()

Welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.6/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> quit

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)". Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.

>>> _
```

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 **einzel**n 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.

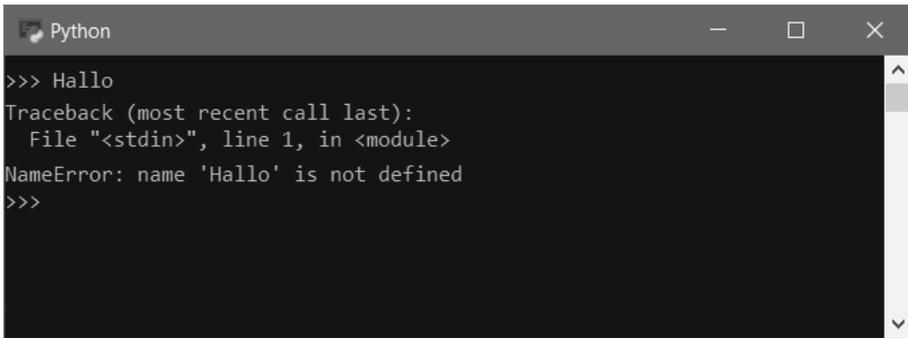
```
Python
>>> 1+2+3
6
>>>
```

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 `Ha1lo` ein und schließe das mit der `↵`-Taste ab.



```
Python
>>> Hallo
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Hallo' is not defined
>>>
```

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")`



```
Python
>>> print("Hallo")
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:



```
Python
>>> print(55)
55
>>>
```

➤ 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?



```
Python  
>>> Text = "Hallo"  
>>> print(Text)  
Hallo  
>>> _
```

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.

Variable = Wert

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:

```
Python
>>> Text = "Hallo, wer da?"
>>> print(Text)
Hallo, wer da?
>>> Name = input()
Hans-Georg
>>> print(Name)
Hans-Georg
>>> _
```

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 Eingabefenster 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.

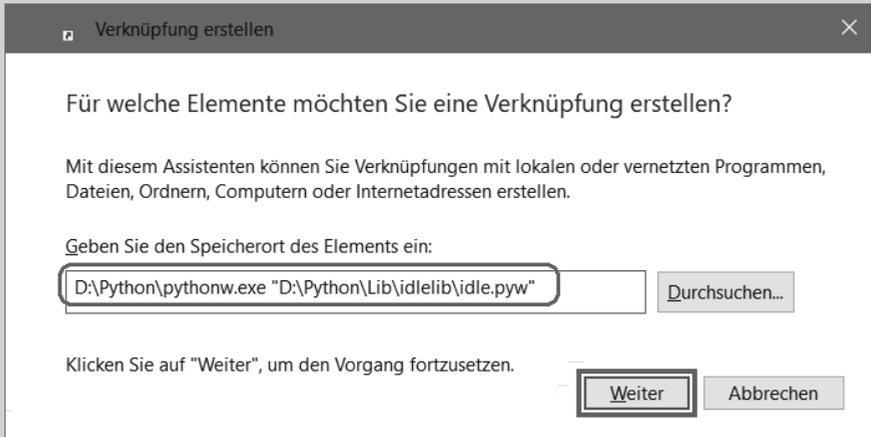
Wenn bei der Installation im START-Menü von Windows eine Verknüpfung zu Python eingerichtet wurde, dann findest du dort auch einen Eintrag wie IDLE (PYTHON).



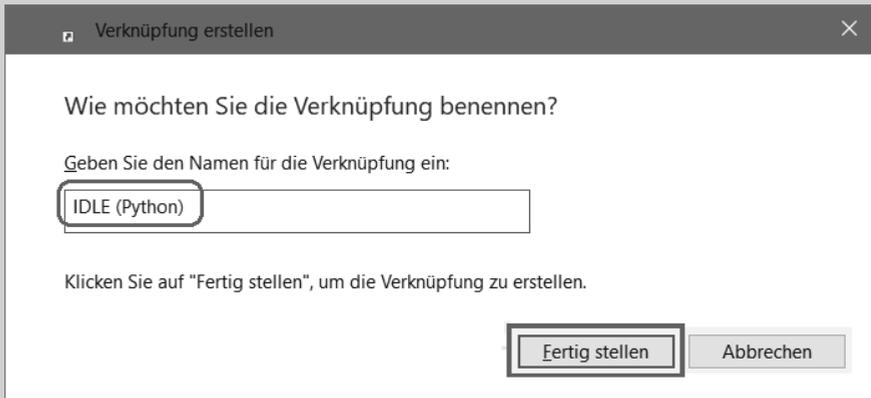
➤ Klicke darauf, um dieses Programm zu starten.

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"

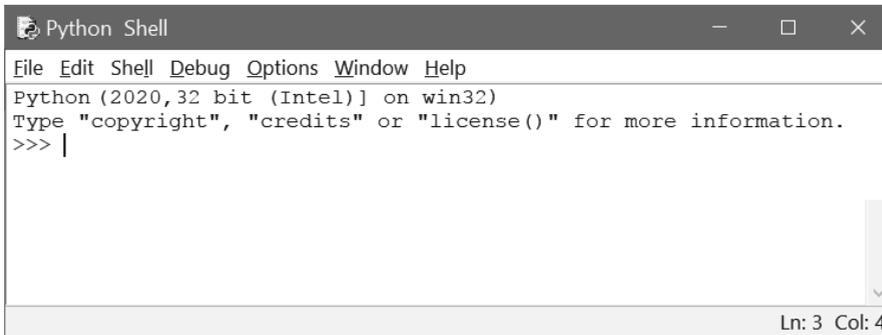


- ◇ Wichtig ist, dass D:\PYTHON auch das Verzeichnis ist, in das du Python installiert hast. Sonst musst du das entsprechend anpassen.
- ◇ Nenne das neue Symbol auf dem Desktop IDLE (Python).



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:



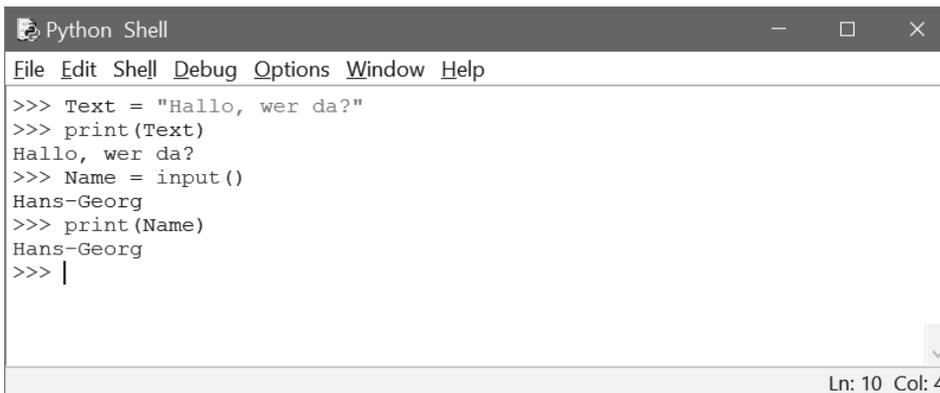
```
Python Shell
File Edit Shell Debug Options Window Help
Python (2020, 32 bit (Intel)] on win32)
Type "copyright", "credits" or "license()" for more information.
>>> |
Ln: 3 Col: 4
```

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):



```
Python Shell
File Edit Shell Debug Options Window Help
>>> Text = "Hallo, wer da?"
>>> print(Text)
Hallo, wer da?
>>> Name = input()
Hans-Georg
>>> print(Name)
Hans-Georg
>>> |
Ln: 10 Col: 4
```

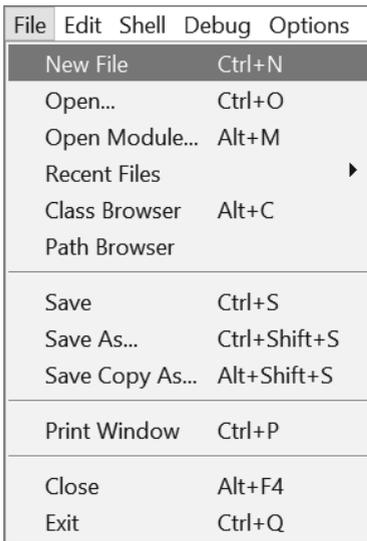
Allerdings sieht es hier etwas bunter aus. Wörter wie `print` und `input` werden farbig angezeigt, ebenso wie in Anführungsstriche gesetzter Text.

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

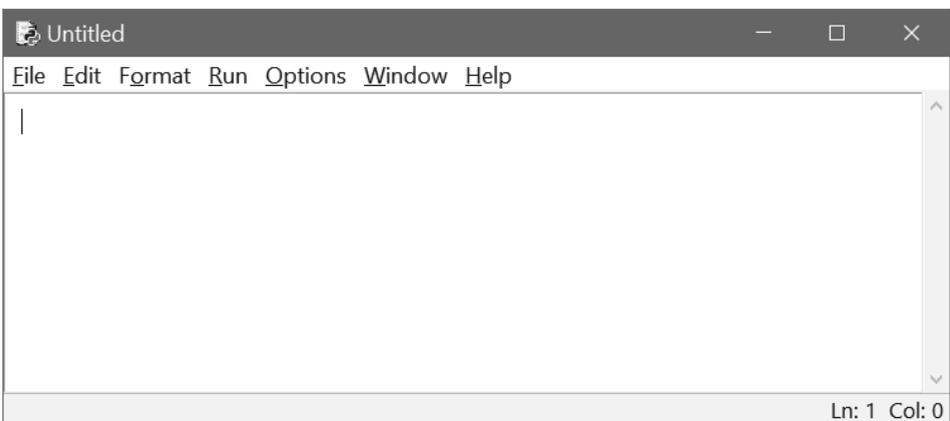
DIE ERSTE PY-DATEI

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.



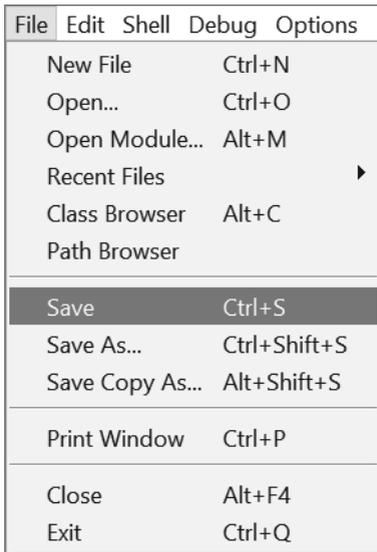
Hier können wir nun unser Programm eingeben, ohne dass irgendeine Zeile davon direkt vom Python-Interpreter ausgeführt wird.

DIE ERSTE PY-DATEI

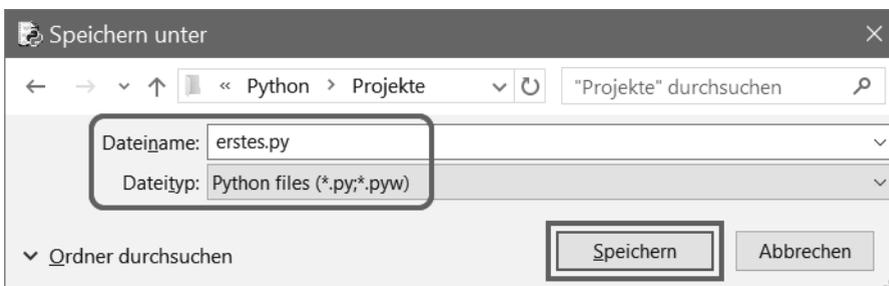
➤ Tippe also diese Zeilen ein:

```
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.



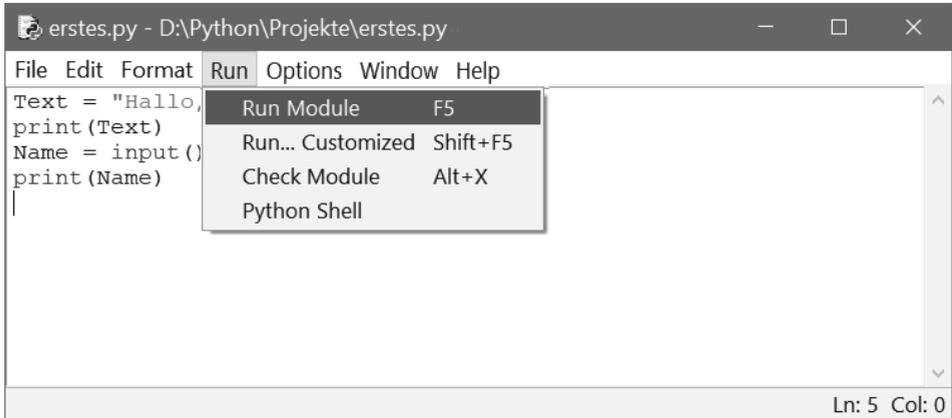
➤ Im Dialogfeld gibst du einen Namen für dieses Programm ein, z.B. erstes (oder was du willst). Das PY wird als Kennung für »Python« automatisch angehängt, wenn du es nicht angibst.



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

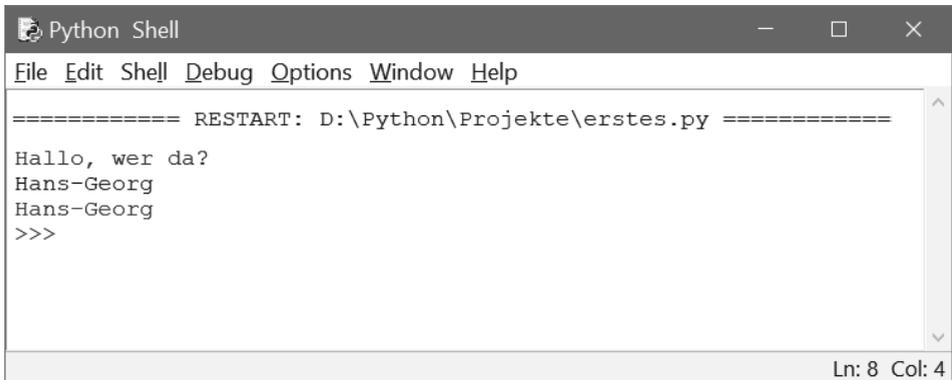


- Um das Programm zu starten, klickst du jetzt auf RUN und RUN MODULE. Oder du drückst die Taste `F5`.



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:

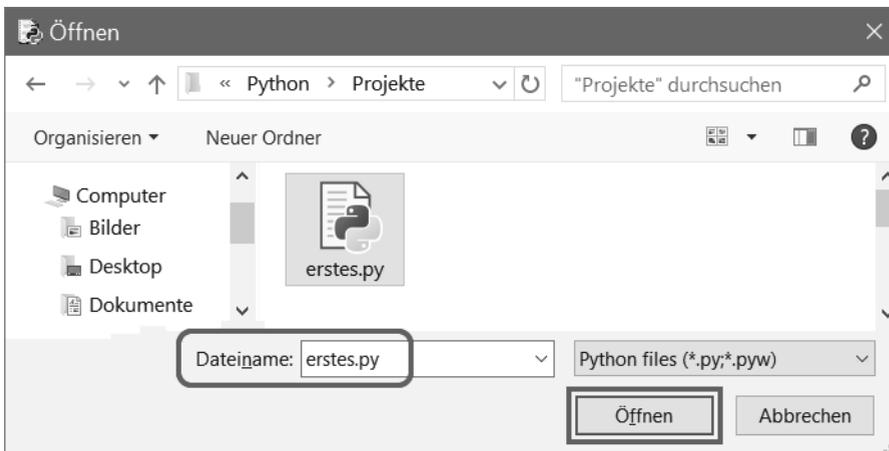


Wenn du nun dieses Fenster schließt, gibt es deine Programmdatei immer noch. Du musst sie dir einfach nur wieder zurückholen.

- Dazu klickst du auf FILE und OPEN.

File	Edit	Shell	Debug	Options
New File			Ctrl+N	
Open...			Ctrl+O	
Open Module...			Alt+M	
Recent Files				▶
Class Browser			Alt+C	
Path Browser				
Save			Ctrl+S	
Save As...			Ctrl+Shift+S	
Save Copy As...			Alt+Shift+S	
Print Window			Ctrl+P	
Close			Alt+F4	
Exit			Ctrl+Q	

➤ 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:

<code>print()</code>	Ausgabe von Zahlen und Strings auf dem Monitor
<code>input()</code>	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.

Variable = Formel

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:

```

Python Shell
File Edit Shell Debug Options Window Help
===== RESTART: D:\Python\Projekte\erstes.py =====
Hallo, wer da?
Hans-Georg
Du bist also Hans-Georg
Und wie geht es?
gut
Dir geht es also gut
>>>
Ln: 11 Col: 4

```

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 (`\`).



```

erstes.py - D:\Python\Projekte\erstes.py
File Edit Format Run Options Window Help
print("Hallo, wer da?")
Name = input()
print("Man mag es kaum glauben, aber du bist
tatsächlich also " + Name)
print("Und wie geht es?")
Antwort = input()
print("So ist das also. Dir geht es
tatsächlich " + Antwort);
Ln: 9 Col: 0

```

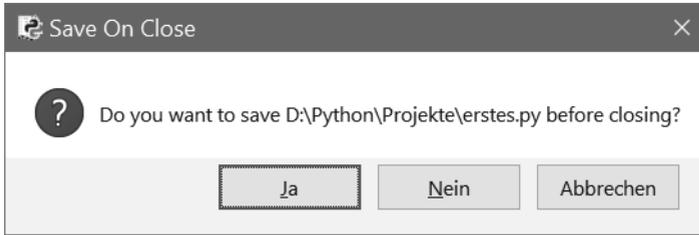
PYTHON VERLASSEN

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.

ZUSAMMENFASSUNG

Solltest du vorher noch etwas am Text geändert haben, dann erscheint zuerst dieses Meldefenster:



➤ Je nach Lage klickst du auf einen der Buttons. (Mit ABBRECHEN kehrst du zum Editor zurück.)

ZUSAMMENFASSUNG

Mit deinem ersten Python-Projekt hast du schon einiges geschafft. Mal sehen, was du von diesem Kapitel behalten hast. Da wären zuerst mal ein paar Operationen im Umgang mit der Python-Shell (IDLE):

IDLE starten	Doppelklicke auf das Symbol für IDLE (Python)
Datei speichern	Klicke auf FILE/SAVE
Datei neu speichern	Klicke auf FILE/SAVE AS
Datei öffnen/laden	Klicke auf FILE/OPEN
Programmprojekt starten	Klicke auf RUN/RUN MODULE
Hilfesystem aufrufen	help() eintippen (mit quit verlassen)
Python beenden	Klicke auf FILE/EXIT

Und ein klein wenig vom Python-Wortschatz hast du auch schon kennengelernt:

print()	Funktion für die Anzeige von Zahlen und Text
input()	Funktion für die Eingabe von Zahlen und Text
=	Zuweisungsoperator
+	Operator für Addition oder Verkettung
\	Zeilen-Trenner

EIN PAAR FRAGEN ...

1. Welche Funktionen sind für Eingabe und Ausgabe zuständig?
2. Was ist der Unterschied zwischen Compiler und Interpreter?
3. Ist eine Zuweisung dasselbe wie eine Gleichung?

... ABER NOCH KEINE AUFGABE

Diese Leseprobe haben Sie beim
 edv-buchversand.de heruntergeladen.
Das Buch können Sie online in unserem
Shop bestellen.

[Hier zum Shop](#)