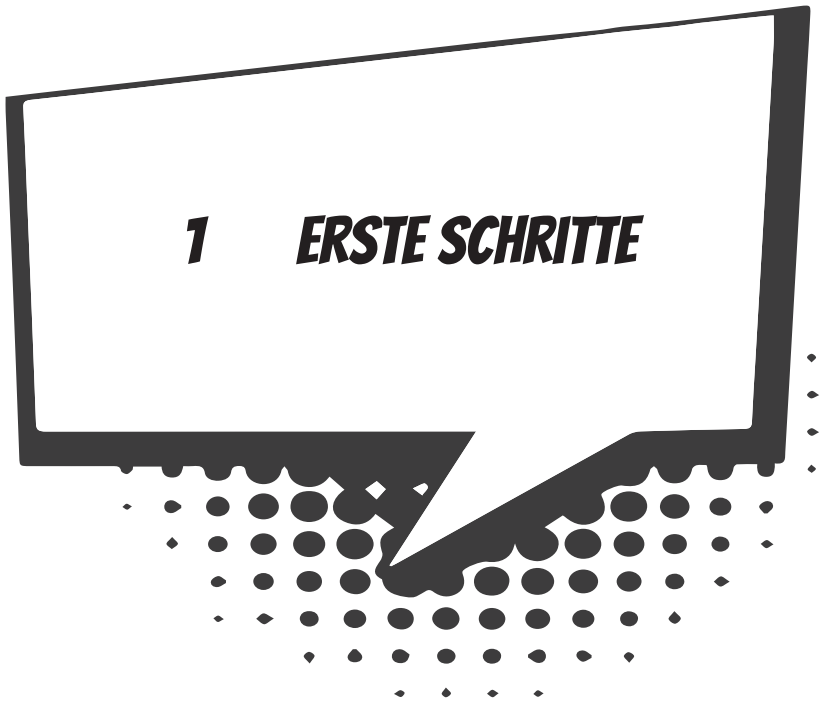
Python für Kids

Programmieren lernen ohne Vorkenntnisse



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

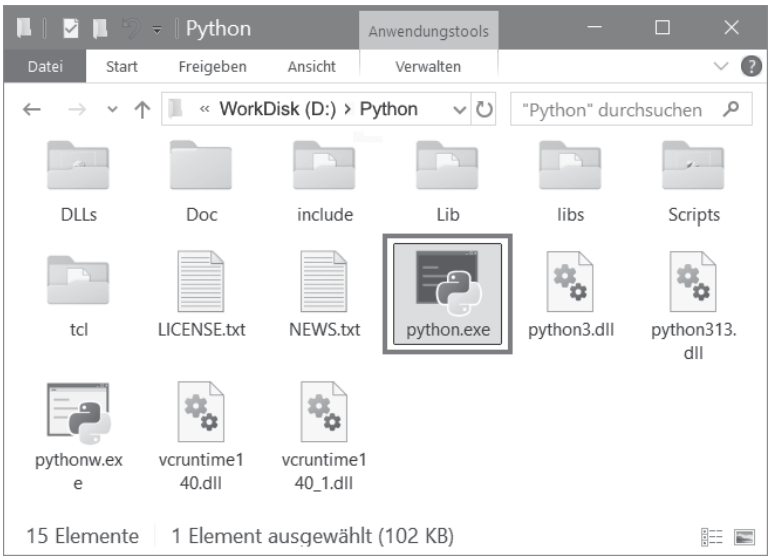
- wie man Python startet
- Anweisungen für Ausgabe und Eingabe kennen
- was Variablen sind
- den Typ String kennen
- etwas über den Einsatz von IDLE
- wie man ein Programm erstellt und speichert
- wie man Python beendet

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.



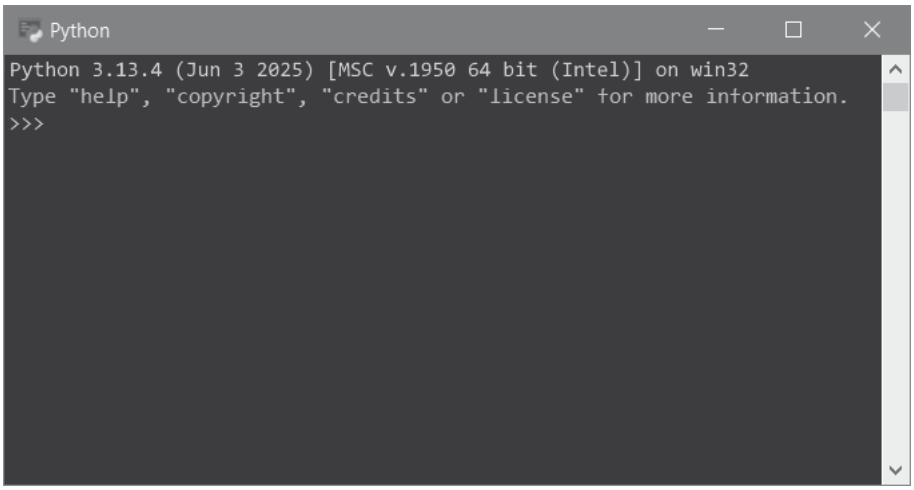
≫ Hier suchst du unter den vielen Symbolen das mit dem Namen PYTHON.EXE heraus. Doppelklicke auf das Symbol.

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.

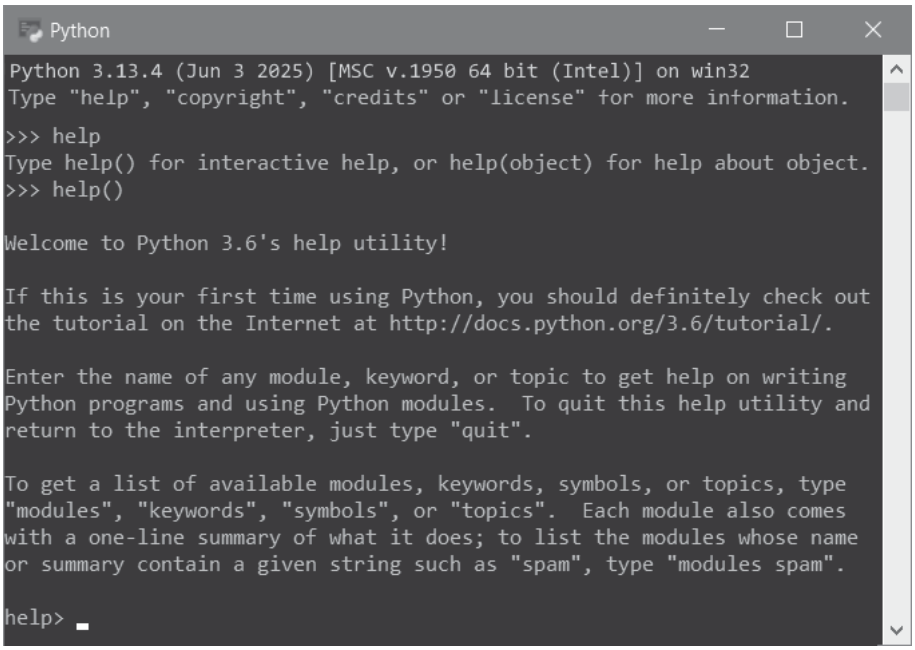
Die drei Zeichen werden hier auch **Prompt** genannt. Das ist eine Art Eingabeaufforderung, weil du dahinter etwas eingeben kannst (und musst, wenn es weitergehen soll).

≫ 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.

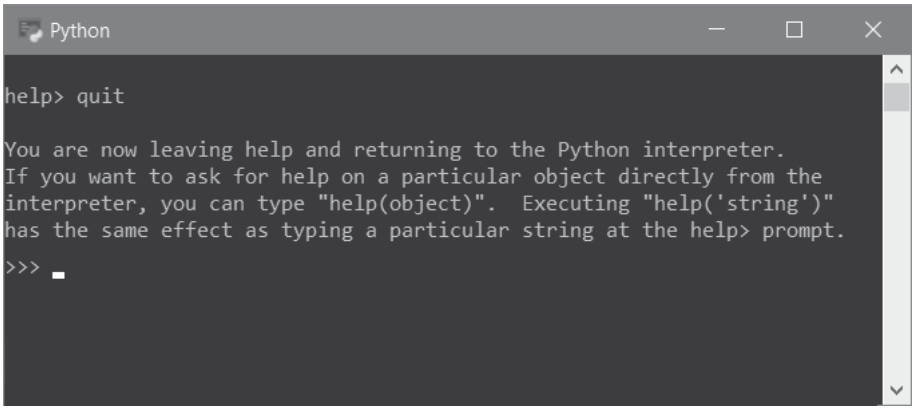
≫ Gib also `help()` ein.

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.

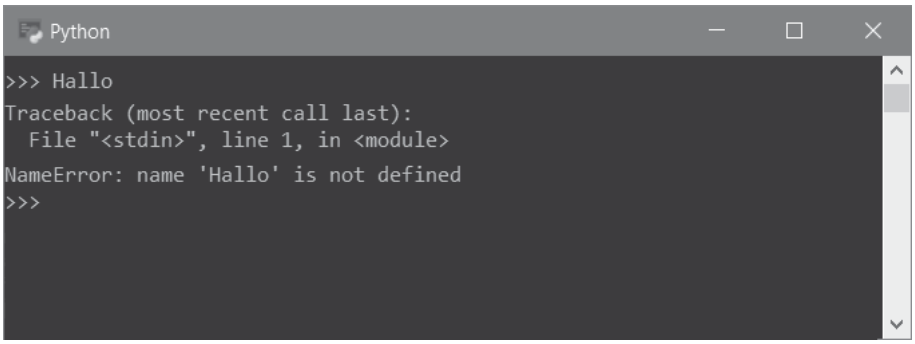
≫ Tippe also ein: `1+2+3` und drücke dann die ↵-Taste.

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.

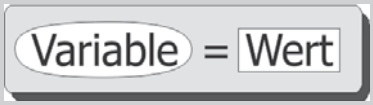
Nun wird es ein bisschen komplizierter. Bis jetzt haben wir immer nur einen Befehl eingegeben. Aber richtige Programme bestehen natürlich aus mehr als nur einer Zeile. Versuchen wir es mal mit diesem kleinen Programmstück:

```
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)
```

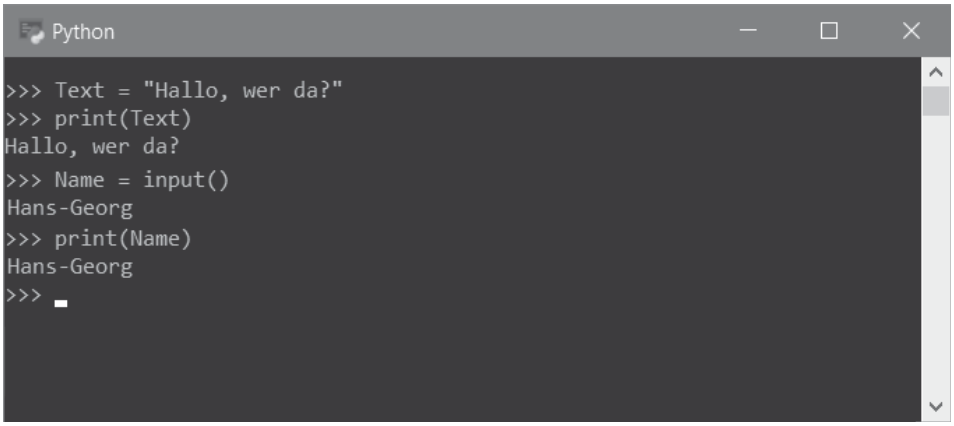
In Python muss ein Text wie unser Hallo-Gruß immer in **Anführungszeichen** gefasst werden. Ich benutze hier die doppelten ("), aber auch die einfachen (') sind erlaubt.

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.

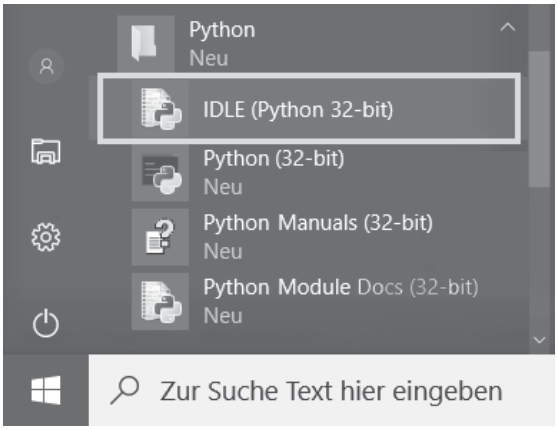
Aber etwas gefällt mir nicht. Zum Beispiel das dauernde Neueingeben. Man kann nicht einfach beliebig mit den Pfeiltasten im Programm herumwandern oder mit der Maus irgendwohin klicken und dort Text ändern. Besser wäre es doch, wenn man sich im Python-Fenster wie in einem Texteditor bewegen könnte.

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.

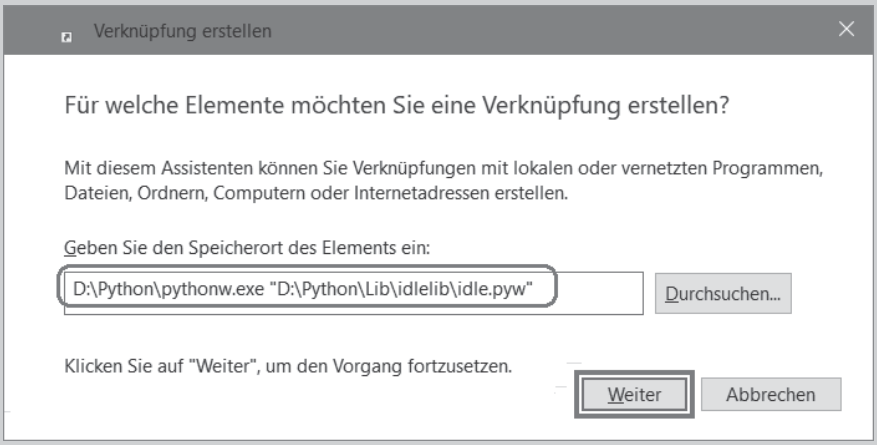
Wenn bei der Installation im START-Menü von Windows eine Verknüpfung zu Python eingerichtet wurde, dann findest du dort auch einen Eintrag wie IDLE (PYTHON).



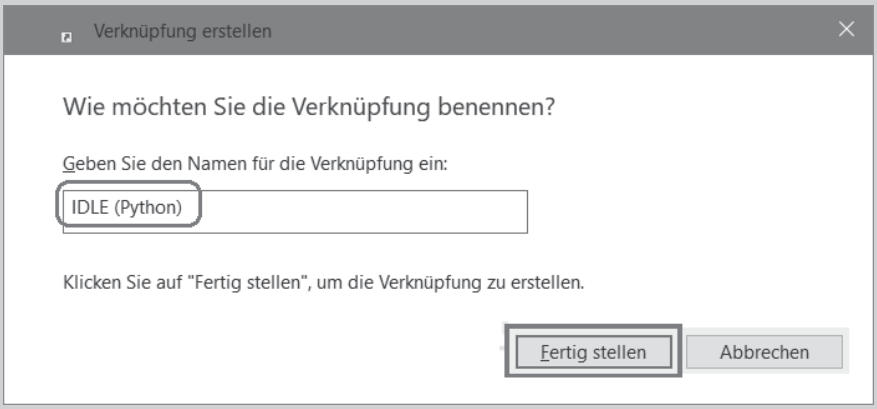
≫ Klicke darauf, um dieses Programm zu starten.

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"`

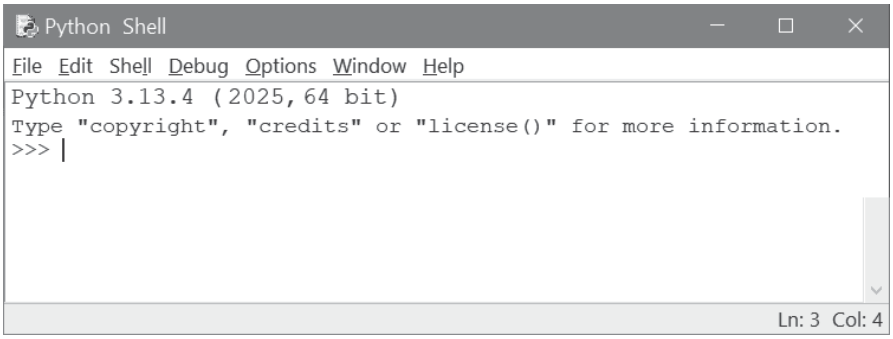


- Wichtig ist, dass D:\PYTHON auch das Verzeichnis ist, in das du Python installiert hast. Sonst musst du das entsprechend anpassen.
- Nenne das neue Symbol auf dem Desktop `IDLE (Python)`.



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

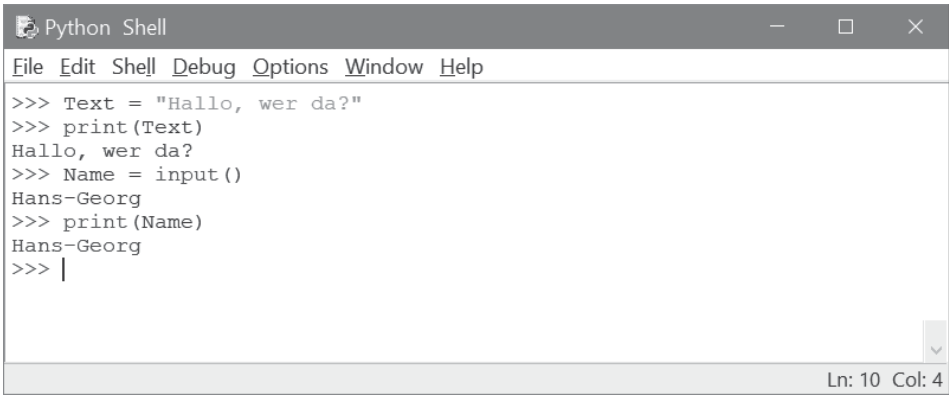
Nach dem Start von IDLE findest du dich in einem solchen Fenster wieder:



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.