Programmieren mit Python

Workbook.

Ein 30-Tage Premium-Kurs von Edley.

Dein 30-Tage **Kursplan**

Einführung | Tage 1 - 2

Wer Programmieren lernen will, muss sich ganz zu Beginn eine schwierige Frage stellen:

Welche Programmiersprache sollte ich überhaupt wählen?

Wenn du dieses Workbook liest, dann hast du deine Entscheidung sicherlich bereits getroffen: Sie ist auf *Python* gefallen. Trotzdem soll sich die Einführung zu diesem Workbook mit der Frage befassen, warum es sinnvoll ist, das Programmieren zu erlernen und warum Python sehr wahrscheinlich die richtige Wahl ist.

Ganz zu Beginn dieses Workbooks steht die Frage, weshalb du überhaupt damit anfangen solltest, das Programmieren zu erlernen. Für viele Menschen lautet die Antwort: um die Karrierechancen zu verbessern. Programmierer erfreuen sich einer enormen Nachfrage, sodass die Löhne für diese Berufsgruppe ausgesprochen hoch sind. Doch selbst wenn du nicht als Programmierer arbeiten willst, sind Grundkenntnisse in diesem Bereich sehr wichtig. Die Digitalisierung umfasst immer weitere Wirtschaftsbereiche, sodass gute Computerkenntnisse mittlerweile in fast allen Branchen eine bedeutende Rolle spielen. Daher stellen Programmierkenntnisse in vielen Berufen eine hilfreiche Zusatzqualifikation dar, die man auf keinen Fall unterschätzen sollte.

Neben den Karrieremöglichkeiten stellt das Programmieren aber auch eine interessante intellektuelle Herausforderung dar: Wenn du auf ein Alltagsproblem triffst, ist es sehr interessant, ein Programm zu erstellen, um dieses zu lösen. Das ist eine hervorragende Denksportaufgabe und eröffnet auf viele Situationen eine völlig neue Sichtweise, die du im Laufe dieses Kurses immer wieder einnehmen wirst.

Python eignet sich sowohl für professionelle Programmierer als auch für Anfänger, die ihre erste Programmiersprache erlernen. Die Programmiersprache bietet einen sehr großen Funktionsumfang, sodass es möglich ist, damit Software zu erstellen, die allen Anforderungen der Wirtschaft entspricht. Daher kommt Python in diesem Bereich sehr häufig zum Einsatz. Gleichzeitig ist die Programmiersprache sehr einfach strukturiert, sodass sie sich hervorragend für Anfänger eignet.

Wenn du dieses Workbook liest und die praktischen Programme aus den zugehörigen Videos durchführst, lernst du von Grund auf, mit Python zu programmieren. So wirst du am Ende der 30 Tage in der Lage sein, für jede mögliche Problemstellung oder Idee eigenständig ein Programm zu coden oder die entsprechenden Ansätze kennen, um das Problem mithilfe des Internets und verschiedenen Programmier-Tools Schritt für Schritt zu lösen.

Vorteile von Python | Tag 1

Die Zahl an Programmiersprachen ist riesig. Namen wie *C*, *C++*, *Java*, *PHP*, *Ruby* oder *Swift* sind dir vielleicht bereits ein Begriff. Hinzu kommen hunderte weiterer Programmiersprachen. Daher stellt sich die Frage, weshalb du ausgerechnet mit Python anfangen solltest.

Nachfrage. Ein besonders wichtiger Grund für diese Entscheidung ist die hohe Nachfrage nach Python Programmierern. Python ist sehr mächtig und ermöglicht es, für fast alle Probleme im Bereich der Informatik eine passende Lösung zu entwickeln.

Effizienz. Zum anderen arbeiten Python-Programmierer sehr effizient. Die einfache Struktur des Programmcodes sorgt dafür, dass die Software vergleichsweise schnell entsteht.
Aufgrund dieser Vorteile wird Python als Programmiersprache in einer Vielzahl von Organisationen eingesetzt. Die CIA nutzt Python zum hacken, Google um Websites zu crawlen und Spotify, um seinen Nutzern Songs zu empfehlen. Sogar die NASA nutzt Python. Aber keine Sorge, mit Python zu programmieren ist bei weitem keine Raketenwissenschaft.

Einfachheit. Tatsächlich ist ein weiterer Grund, der für Python spricht, die Einfachheit und Flexibilität dieser Programmiersprache. Im Vergleich zu anderen Programmiersprachen musst dich bei Python deutlich weniger um strikte Formalitäten kümmern und kannst dich allein auf den Code fokussieren. Das führt zu schnellen Ergebnissen, mit relativ wenig Aufwand. Zum Vergleich:

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Einfaches Programm in *Java*

```python
print("Hello, World!")
```

Das gleiche Programm in *Python*

Verbreitung. All diese Vorteile führen dazu, dass auch erfahrene Programmierer Python immer öfter zur Programmiersprache ihrer Wahl machen. Python hat eine riesige Community (häufig fällt der Begriff *Pythonista*, um die Mitglieder der Community zu beschreiben) und es gibt für beinahe jede Problemstellung zahlreiche Lösungsansätze in den zahlreichen Python-Foren.

Relevanz. Python hat inzwischen Java als beliebteste Programmiersprache abgelöst und wird wahrscheinlich in wenigen Jahren auch die Programmiersprache sein, die am weitesten verbreitet ist. Diese Entwicklung ist einerseits durch die Effizienz und Einfachheit von Python begründet, aber auch durch die Vielfältigkeit der Einsatzgebiete der Programmiersprache. Verwendet wird Python unter anderem für die Entwicklung von Desktop-Programmen und Applikationen, den Aufbau von Web-Applikationen, für zahlreiche Projekte rund um den Bereich *Artificial Intelligence* (insbesondere Machine Learning) und bei der Automatisierung von Tasks. Das alles sind Technologien, die die Zukunft maßgeblich formen werden.

Philosophie hinter Python | Tag 1

Die im vorherigen Kapitel beschriebenen Aspekte hängen eng mit der Entwicklungsgeschichte von Python und der Philosophie, die hinter dieser Programmiersprache steckt, zusammen. Dabei handelt es sich um eine relativ junge Programmiersprache, die erst zu Beginn der 90er Jahre entstand. Für die Entwicklung verantwortlich war der Niederländer Guido van Rossum, der zu dieser Zeit am Centrum Wiskunde & Informatica in der niederländischen Stadt Amsterdam arbeitete. Um den Studenten das Programmieren beizubringen, verwendete dieses Institut damals die Programmiersprache *ABC*. Diese war sehr einfach strukturiert. Allerdings wies sie zu große Unterschiede zu den gängigen Programmiersprachen – insbesondere zu der älteren und bekannteren Programmiersprache C – auf. Dies führte dazu, dass die Akzeptanz von ABC nicht besonders hoch war.

Van Rossum beschloss daraufhin eine neue Programmiersprache, namens Python, zu entwickeln. Diese sollte, wie auch ABC, als Lehrsprache dienen und so einfach wie möglich aufgebaut sein. Python verwendet noch heute eine vergleichsweise unkomplizierte *Syntax*. (Syntax bezeichnet die formellen Regeln zur Gestaltung eines Programms.) Darüber hinaus ist Python-Code leicht verständlich, da sich die Befehle vorwiegend an der englischen Sprache orientieren. Das macht es gerade für Anfänger einfacher, den Sinn des Codes zu verstehen. Außerdem verzichtet Python so weit wie möglich auf Klammern und ähnliche Elemente, die bei vielen anderen Programmiersprachen für die Strukturierung der Programme notwendig sind. Hierdurch wird eine weitere häufige Fehlerquelle beseitigt.

Der Erfolg von Python war überwältigend. Als Ergebnis bildete sich sehr schnell ein großes Team aus freiwilligen Mitarbeitern heraus, die Python weiterentwickelten und dies bis heute tun, wodurch die Programmiersprache ständig verbessert wird und neueste Entwicklungen umsetzt. Van Rossum wirkte dabei lange Zeit in entscheidender Position mit – er hatte bei allen wichtigen Entscheidungen das letzte Wort. Im Juli 2018 gab er bekannt, sich von dieser Funktion zurückzuziehen. Dennoch arbeitet er nach wie vor an der Weiterentwicklung von Python mit.

Python 2 vs Python 3 | Tag 1

Seit die erste Vollversion von Python 1994 erschienen ist, hat sich viel getan. Die Entwicklergemeinschaft hat viele zusätzliche Funktionen eingefügt, die das Programmieren erleichtern. Mit der Version 2 von Python kam beispielsweise der *Garbage-Collector*, ein Feature zur Reduzierung des Speicherplatzes und zur Erhöhung der Effizienz, hinzu.

2008 erschien mit Python 3 die dritte Version der Programmiersprache. Dabei entschieden sich die Entwickler dazu, auf eine Kompatibilität zum bisherigen Code zu verzichten, um von den Vorteilen von Python 3 (Verbesserte Unterstützung von Unicode-Zeichen; True Division: Ergebnisse der Division werden bei Bedarf als Kommazahl dargestellt; Verbesserte Performance bei Computern mit mehreren Prozessoren, usw.) Gebrauch zu machen. Das führt dazu, dass Programme, die in Python 2 geschrieben sind, in Python 3 häufig nicht mehr ausführbar sind. Dieses Workbook verwendet mit Python 3 stets die aktuelle Version und wird regelmäßig aktualisiert.

Python 2 ist mittlerweile veraltet und kommt nur noch selten zum Einsatz. Wenn du im Internet oder in älteren Lehrbüchern nach weiterführenden Informationen suchst, dann solltest du diesen Unterschied trotzdem im Hinterkopf behalten. Wenn es sich bei den meisten entsprechenden Beispielen noch um Code in Python 2 handeln sollte, dann kannst du diesen mit einem Python-3-Interpreter nicht mehr ausführen. (Was ein *Python-Interpreter* ist, erfährst du im nächsten Kapitel.)

Download und Installation | Tag 2

Wenn du ein Code-Programm schreibst, dann besteht der geschriebene Code aus reinem Text. Ein Computer selbst arbeitet hingegen mit elektrischen Impulsen, die als Maschinensprache bezeichnet werden. Damit ein Programm ausgeführt werden kann, ist es notwendig, den Programmcode von der Textform in die Maschinensprache zu übersetzen.

Interpreter und Compiler

Für diese Übersetzung gibt es grundsätzlich zwei Möglichkeiten: Du kannst einen *Interpreter* oder einen *Compiler* verwenden.

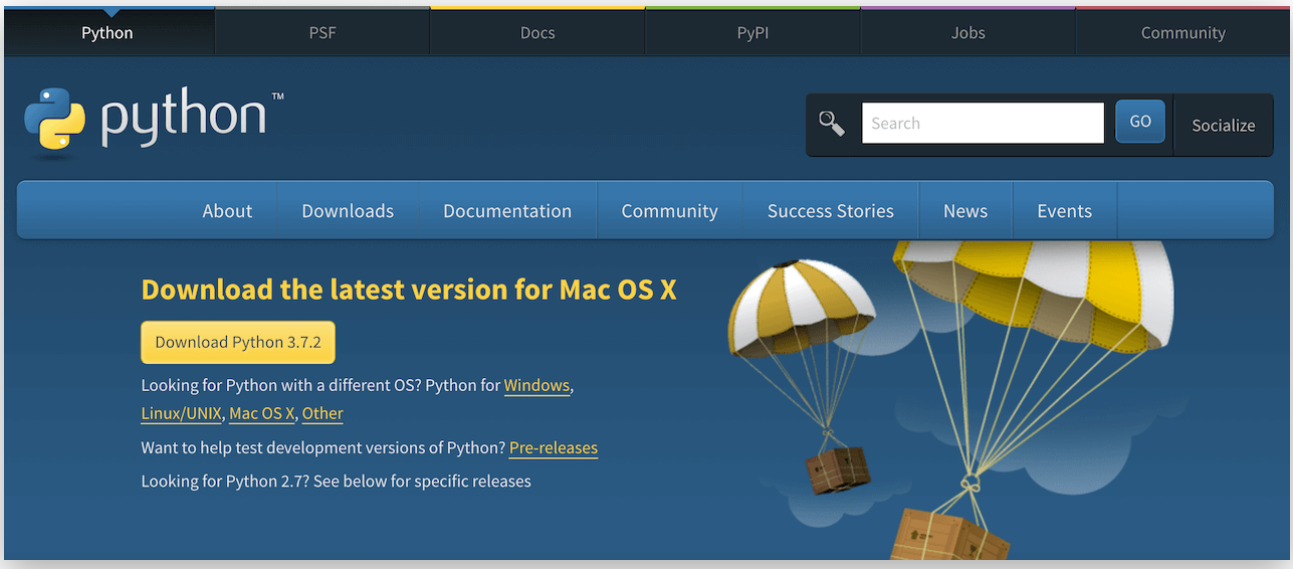
Ein **Compiler** übernimmt die Übersetzung ein einziges Mal und erzeugt daraufhin ein ausführbares Programm in Maschinensprache.

Ein **Interpreter** führt die Übersetzung hingegen jedes Mal aufs Neue aus, wenn du das Programm startest. Welche dieser beiden Alternativen zum Einsatz kommt, hängt von der Programmiersprache ab, die du verwendest.

Python ist eine interpretierte Programmiersprache. Das bedeutet, dass die Programme in Textform abgespeichert werden und bei jeder Ausführung erneut übersetzt werden. Das reduziert zwar die Effizienz, da diese Übersetzung etwas Zeit benötigt. Doch profitierst du davon, dass sich die Programme auf jedem Rechner ausführen lassen, auf denen ein entsprechender Interpreter installiert ist. Daher eignet sich Python hervorragend für plattformübergreifende Programme.

Um die Programme, die du mit Python schreiben wirst, selbst auszuprobieren, benötigst du also einen Python-Interpreter. Diesem kannst du auf folgender Seite kostenlos herunterladen:

https://www.python.org/downloads/



Screenshot: *Der Internetauftritt von Python*

Auf dieser Seite musst du lediglich auf die gelbe Schaltfläche (**Download Python**) klicken, um mit dem Download zu beginnen. Dabei findest du passende Versionen für alle gängigen Betriebssysteme: *Windows*, *macOS*, *Linux* und einige weitere. Die Voraussetzungen für dieses Programm sind nicht anspruchsvoll. Wenn du ein aktuelles Betriebssystem verwendest, sollte die Installation keine Probleme bereiten. Nachdem du die Schaltfläche betätigt hast, wird zunächst der Installations-Assistent heruntergeladen. Diesen musst du daraufhin anklicken, um mit der Installation zu beginnen.

Beim Fenster, das sich nun öffnet, ist es (für *Windows*-User) wichtig, ganz unten die Checkbox mit der Bezeichnung **Add Python X.X to PATH** auszuwählen. Tust du das nicht, ist Python nur in dem Verzeichnis zugänglich, in dem es installiert wurde. Da es jedoch nicht sinnvoll ist, alle Programme im gleichen Verzeichnis abzuspeichern, ist es notwendig, den Python-Interpreter auch aus anderen Bereichen zugänglich zu machen. Dafür musst du die entsprechende Pfadvariable hinzuzufügen.

Solltest du das Anklicken der Checkbox bei der Installation vergessen haben, ist es auch möglich, die Pfadvariablen manuell einzufügen. Da das jedoch etwas kompliziert ist, ist es empfehlenswert, das Programm wieder zu löschen und daraufhin erneut zu installieren – dieses Mal mit angeklickter Checkbox.

Screenshot: *Falls dein Betriebssystem Windows ist, solltest du die Checkbox am unteren Bildrand auf jeden Fall anklicken.*

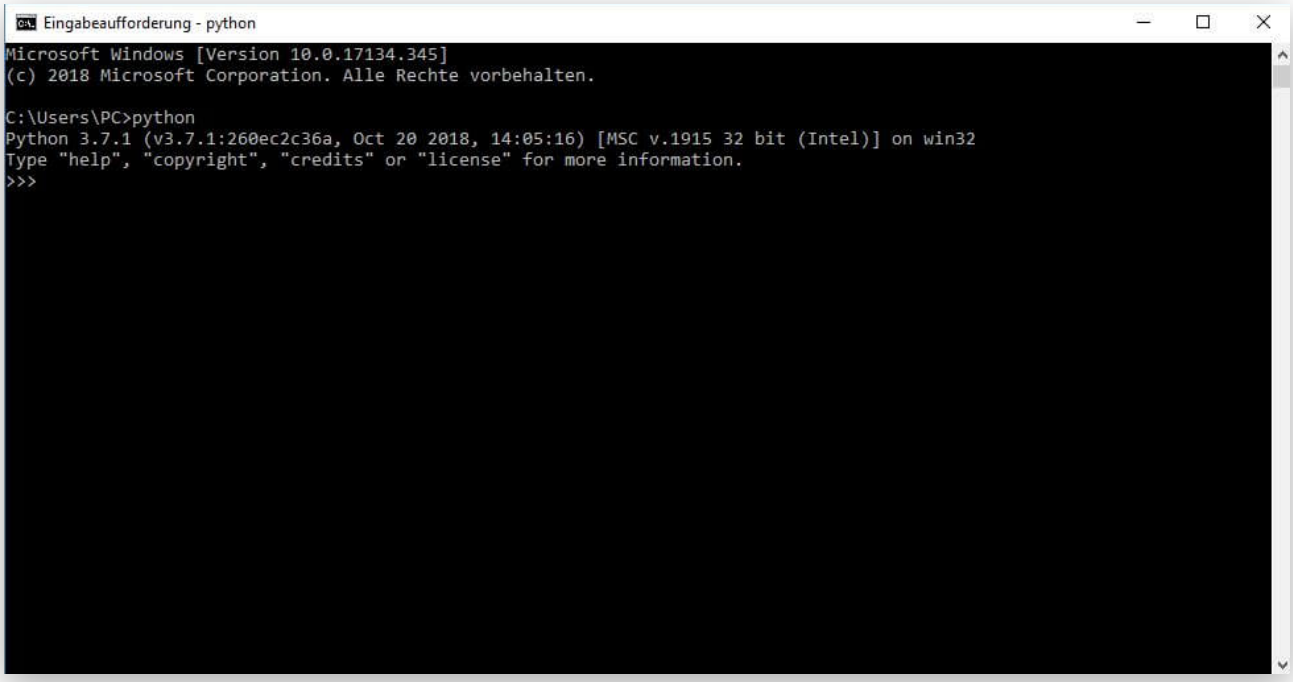
Um zu überprüfen, ob du alles richtig installiert hast, kannst du einen *Kommandozeileninterpreter* öffnen.

- Diesen findest du unter *Windows* im Startmenü im Ordner `Windows-System` unter der Bezeichnung **Eingabeaufforderung**.
- Solltest du einen *Linux*-Rechner verwenden, musst du nach dem **Terminal** suchen, das unter anderem über den Shortcut `Strg` + `Alt` + `T` erreichbar ist.
- Im Betriebssystem *macOS* trägt der Kommandozeileninterpreter ebenfalls die Bezeichnung **Terminal**. Öffne einfach die Suchfunktion (`cmd⌘` + `Leertaste`) und suche nach *Terminal*.

Um die korrekte Installation zu überprüfen, kannst du nun den Begriff `python` in den Kommandozeileninterpreter eingeben und mit `Enter↵` bestätigen. ***Hinweis:*** Die Eingabe bestätigst du immer mit `Enter↵` und springst dadurch automatisch in die nächste Zeile!

Ist alles erfolgreich verlaufen, sollte dieser die installierte Version anzeigen. Ist der Interpreter nicht verfügbar, kommt es hingegen zu einer Fehlermeldung.

Falls dir nach der Eingabe von `python` in den Kommandozeileninterpreter keine Version von **Python 3.X** angezeigt wird, sondern **Python 2.X** (insbesondere bei *macOS*), versuche es mit dem Befehl `python3` oder `python3 --version`. Jetzt sollte eine Version von **Python 3.X** angezeigt werden.
Falls es bei zukünftigen Eingaben mit dem Begriff `python` zu Problemen kommen sollte, versuche immer zuerst eine `3` direkt (also ohne Leerzeichen) anzuhängen.



Screenshot: *Wenn du den Interpreter richtig installiert hast, erscheint nach der Eingabe des Begriffs* `python` *bzw.* `python --version` *entsprechende Version.*

▶ *Python Installation und Hello World*

Editor

Computerprogramme bestehen aus Text. Wenn du nun ein normales Textverarbeitungsprogramm wie *Word* verwendest, dann ist es jedoch nicht möglich, das Programm auszuführen. Diese Programme speichern viele zusätzliche Informationen zur Gestaltung. Sie verwenden ein ganz eigenes Dateiformat, das sich nicht für die Programmierung eignet. Daher ist es notwendig, einen *Texteditor* zu verwenden, der lediglich den reinen Text abspeichert.

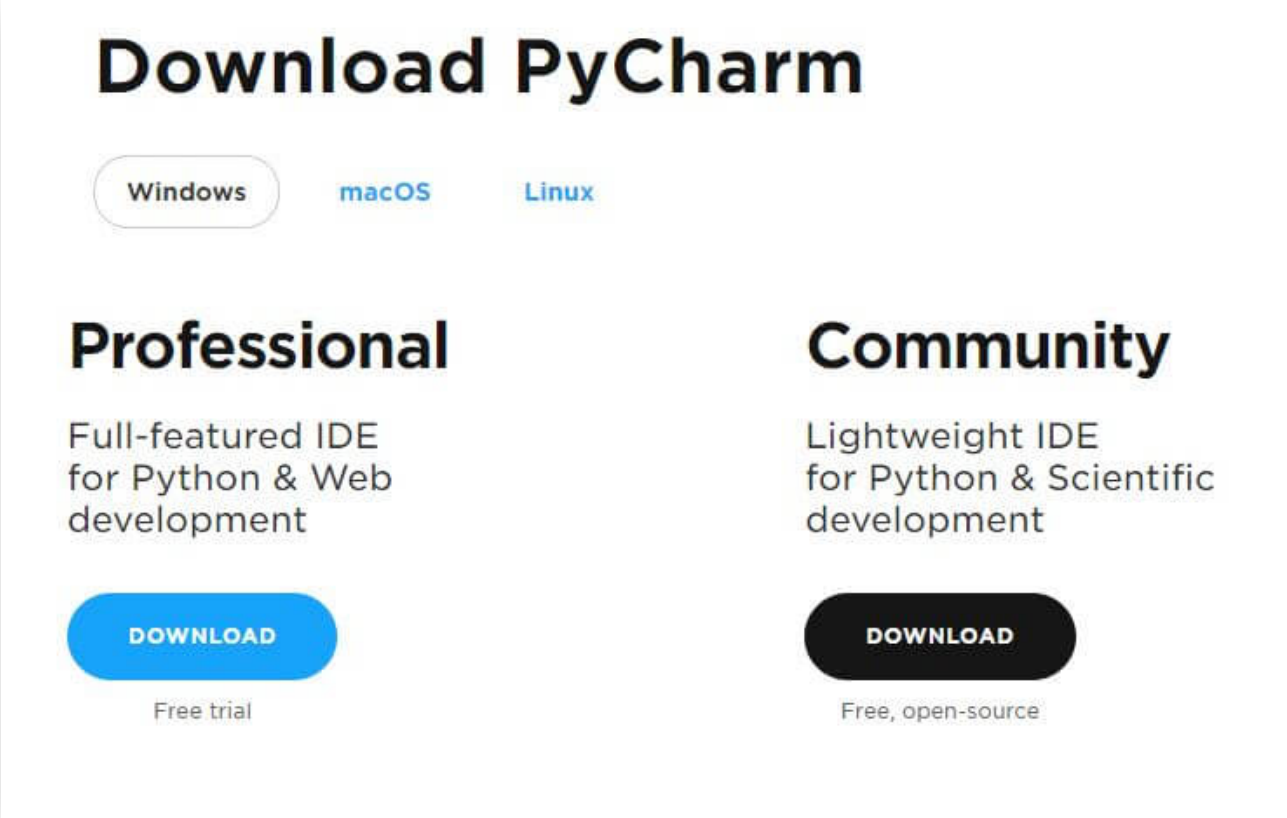
In den meisten Betriebssystemen ist bereits ein Texteditor installiert. Unter Windows entdeckst du beispielsweise den *Microsoft Editor*, der auch unter der Bezeichnung *Notepad* bekannt ist. Wenn du mit macOS arbeitest, solltest du über das vorinstallierte Programm *TextEdit* verfügen. Unter Linux ist meistens – je nach Distribution – entweder *gEdit* oder *Kate* vorinstalliert. Diese Programme eignen sich zwar alle für das Erstellen eines Computerprogramms. Ihr Funktionsumfang ist allerdings recht gering.

Eine der wesentlichen Funktionen eines Texteditors ist das *Syntax-Highlighting*. Das bedeutet, dass verschiedene Teile des Programmcodes in unterschiedlichen Farben dargestellt werden. Das verbessert die Übersichtlichkeit stark. Diese Funktion bietet weder der Microsoft Editor noch TextEdit von Apple.

Die beiden anderen Editoren verfügen zwar bereits über etwas mehr Funktionen, doch gibt es noch eine bessere Alternative: *PyCharm*. Hierbei handelt es sich nicht nur um einen reinen Texteditor – obwohl auch

ein Texteditor in PyCharm enthalten ist und dieser eine der wichtigsten Funktionen darstellt. PyCharm bietet auch die Möglichkeit, die Programme direkt auszuführen. Außerdem stehen viele weitere hilfreiche Tools zur Auswahl, die das Programmieren erleichtern. Daher wird eine solche Software als *IDE* (***I**ntegrated **D**evelopment **E**nvironment* oder auf deutsch *Integrierte Entwicklungsumgebung*) bezeichnet. Das Besondere an PyCharm ist außerdem, dass dieses Programm speziell auf das Programmieren mit Python abgestimmt ist und daher diese Aufgabe besonders einfach gestaltet. In der Basisversion ist es außerdem vollkommen kostenlos. Es ist sowohl für Windows als auch für macOS und Linux erhältlich. Du findest die Software unter folgender Adresse:

https://www.jetbrains.com/pycharm/download/



Screenshot: *Wähle dein Betriebssystem aus und lade die kostenfreie Community-Version herunter*

Du musst vor dem Download das passende Betriebssystem und für eine kostenfreie Nutzung die *Community-Edition* auswählen. Wenn du den entsprechenden Download-Link angeklickt hast, öffnet sich nach einigen Sekunden der Installations-Assistent. Hierbei kannst du die Standard-Einstellungen übernehmen.

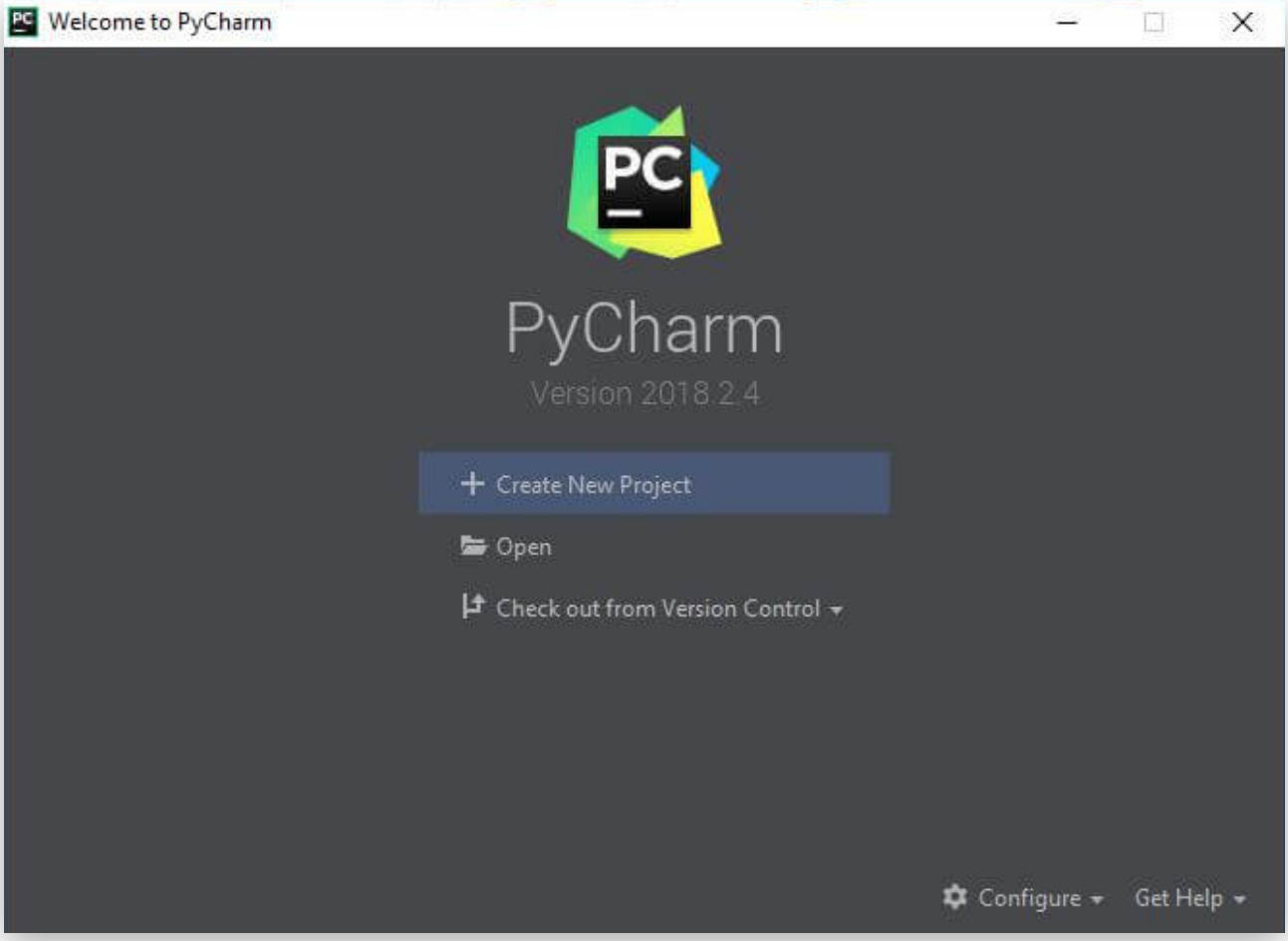
Jetzt ist PyCharm bereit zur Nutzung.

▶ *PyCharm Installation und Hello World*

Basics | Tage 3 - 8

Hello World! | Tag 3

Um mit dem Programmieren zu beginnen, ist es notwendig, die IDE, die im vorherigen Kapitel installiert wurde, zu öffnen. Daraufhin erscheint folgendes Fenster:

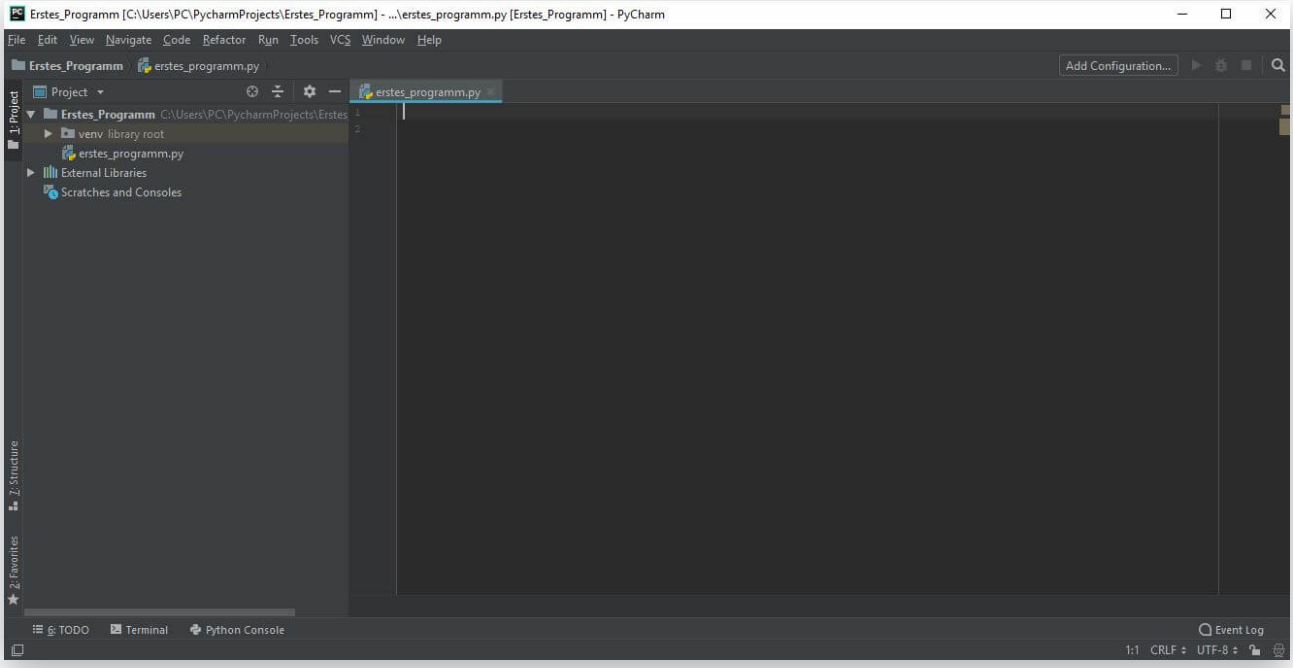


Screenshot: *Das Begrüßungsfenster von PyCharm*

Da wir in diesem Kapitel mit einem neuen Projekt beginnen, musst du **Create New Project** auswählen. Daraufhin musst du einen Ordner für das neue Projekt vorgeben.

In diesem Fall soll dieser den Namen `Erstes_Programm` erhalten. Beim ersten Öffnen wirst du zu einem kleinen Tutorial geleitet. Wenn du Lust hast, kannst du dieses anschauen.

Zu Beginn musst du links oben in der Menüleiste den Menüpunkt **File** und daraufhin auf **New** klicken. Danach öffnet sich ein neues Menüfeld, in dem du nochmals **File** auswählen musst. Danach erscheint ein kleines Fenster, in das du den Namen der neuen Datei eingeben musst. Diesen kannst du frei wählen. Allerdings ist es wichtig, dass er die Endung `.py` erhält. Für das erste Programm wählen wir folgende Bezeichnung: `erstes_programm.py`.



Screenshot: *die Benutzeroberfläche von PyCharm*

Nun erscheint die Benutzeroberfläche von PyCharm, die du auch im Screenshot erkennen kannst. Zunächst solltest du dich mit dieser Software vertraut machen.

- In der **Menüleiste** entdeckst du viele verschiedene Befehle, die für das Erstellen komplizierterer Programme nützlich sind. Diese sind jedoch am Anfang noch von keiner großen Bedeutung. Wenn sie im weiteren Verlauf des Workbooks notwendig werden, findest du an der entsprechenden Stelle eine kleine Erklärung dazu.
- Am **linken Bildschirmrand** entdeckst du eine Auflistung der verschiedenen Verzeichnisse, die zum Projekt gehören. Das wird insbesondere bei komplexen Programmen wichtig, die aus mehreren Dateien bestehen.
- Im Moment ist in erster Linie die **große freie Fläche im Zentrum** des Fensters von Bedeutung. Hier kannst du die Befehle für das Programm eingeben.

Das erste Programm soll so einfach wie möglich sein. Es soll nur eine kurze Begrüßung ausgeben. Um in Python Texte auszugeben, kommt der sogenannte *print-Befehl* zum Einsatz. Um diesen zu verwenden, musst du einfach den Begriff `print` in die Datei schreiben. Danach folgen eine Klammer und Anführungszeichen. Darin steht der Text, den du ausgeben willst:

```
print ("Herzlich Willkommen!")
```

Anmerkung: Ein wichtiger Unterschied zwischen Python 2 und Python 3 besteht in der Verwendung des print-Befehls. In vielen älteren Code-Beispielen findest du diesen ohne die Klammer. Das ist bei Python 3 aber nicht mehr möglich, es muss immer die runde Klammer und den auszugebenden Ausdruck gesetzt werden.

Wenn du diesen Befehl in PyCharm eingibst, dann bemerkst du auch sofort, wie nützlich dieser Editor ist. Wenn du die öffnende Klammer oder das erste Anführungszeichen eingibst, fügt er gleich den abschließenden Teil hinzu. Außerdem gibt er den verschiedenen Bereichen eine unterschiedliche Farbe, sodass du immer den Überblick behältst. Zur Erinnerung: Diese automatische Einfärbung des Codes nach Typ wird *Syntax-Highlighting* genannt. Das macht das Programmieren deutlich einfacher.

Nun musst du das Werk nur noch abspeichern. Das erledigst du über die Menüleiste (**File** > **Save All**) oder noch schneller mit dem Shortcut `Strg` + `S` (unter *macOS* mit `cmd⌘` + `S`). Damit ist das erste Programm bereits fertig.

Bei vielen anderen Programmiersprachen sind selbst für eine einfache Ausgabe mehrere Zeilen mit komplizierten Kommandos notwendig. In Python kannst du hingegen direkt mit den Befehlen loslegen.

Programme ausführen (run)

Nachdem du das erste Programm fertiggestellt hast, besteht der nächste Schritt darin, es mit dem *Python-Interpreter* auszuführen. Dafür gibt es mehrere Möglichkeiten.

Die klassische Methode besteht darin, den *Kommandozeileninterpreter* zu verwenden, der bereits in das Betriebssystem integriert ist. Diesen haben wir im Kapitel Interpreter und Compiler schon kennengelernt. Das folgende Beispiel bezieht sich auf den Kommandozeileninterpreter unter Windows. (Wenn du macOS oder Linux verwendest, ist die Verwendung sehr ähnlich – du musst lediglich die Pfadnamen an dein Betriebssystem anpassen.)

1. Öffne also deinen Kommandozeileninterpreter. Zur Erinnerung: Bei Windows ist der Kommandozeileninterpreter das Programm *Eingabeaufforderung*, bei macOS und Linux wird die Eingabeaufforderung *Terminal* genannt.
   Wenn du dieses Programm öffnest, erscheinen zunächst Angaben zur Version und zum Copyright. In der nächsten Zeile steht das Verzeichnis, in dem sich der Kommandozeileninterpreter im Moment befindet.

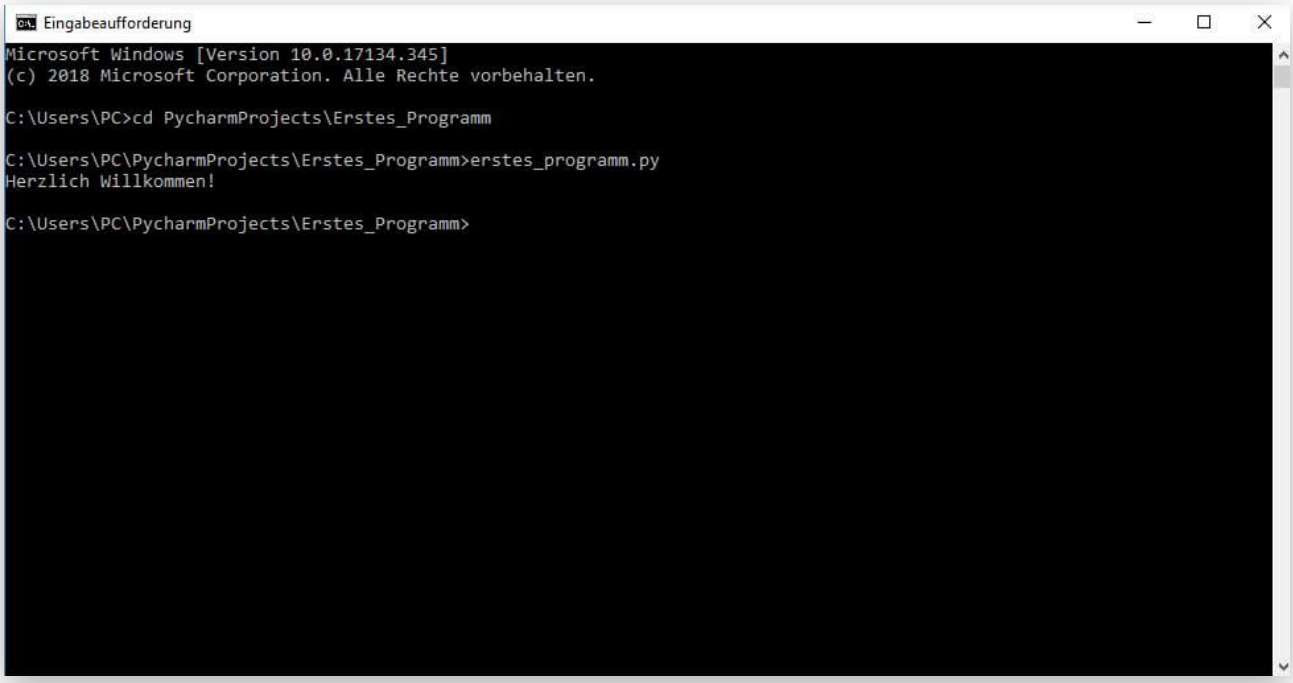
2. Um ein Programm auszuführen, musst du zunächst zum Verzeichnis wechseln, in dem du das Programm abgespeichert hast. Das geht mit dem Befehl `cd`. Du könntest dich damit Ordner für Ordner vorarbeiten, bis du im richtigen Verzeichnis angekommen bist. Schneller geht es jedoch, wenn du die Namen der Ordner einfach hintereinander schreibst und jeweils durch einen Schrägstrich (`/`) voneinander trennst. Bei älteren Windows-Versionen musst du die Ordner statt durch Schrägstriche gegebenenfalls durch Backslashes (`\`) voneinander trennen. Wenn du die Datei im vorgegebenen Standardverzeichnis mit dem Namen `Erstes_Programm` abgespeichert hast, wäre unter Windows folgende Eingabe in die Eingabeaufforderung notwendig:
   ```
   cd PycharmProjects\Erstes_Programm
   ```

Falls du ein anderes Verzeichnis verwendet hast, musst du den Pfad entsprechend anpassen. Solltest du dich nicht mehr genau an die Verzeichnisstruktur erinnern, siehst du diese in PyCharm ganz oben im Fenster.

3. Wenn du dich im richtigen Ordner befindest, musst du nun noch den Dateinamen eingeben:
   `erstes_programm.py`

Anmerkung: Solltest du eine Fehlermeldung erhalten oder nicht Windows verwenden, gib zusätzlich zum Dateinamen das Wort `python` ein, also in diesem Beispiel: `python erstes_programm.py`

Wenn du ihn mit Enter bestätigst, führt der Interpreter dein Programm aus.

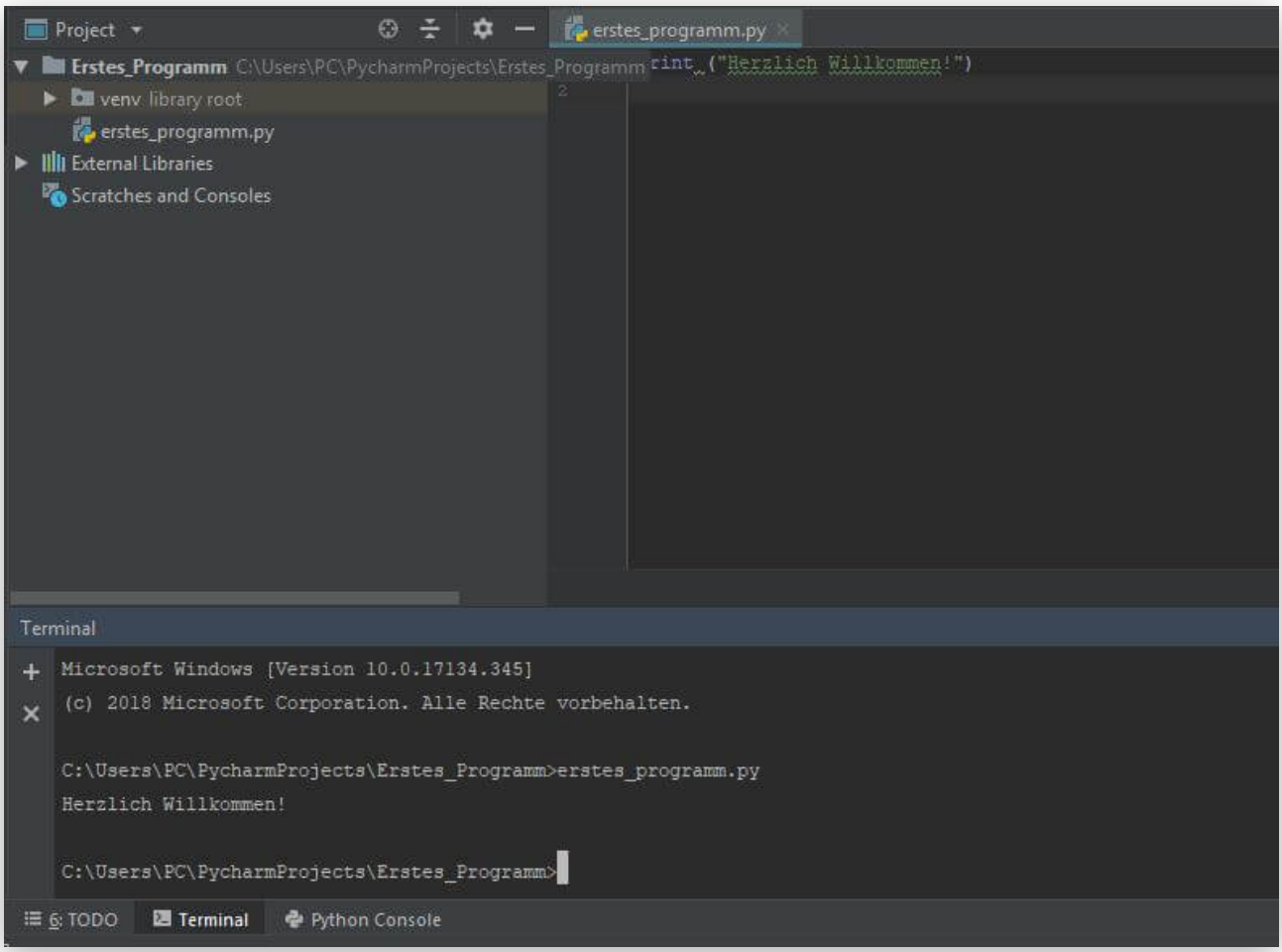


Screenshot: Die Ausgabe des ersten Programms im Kommandozeileninterpreter

Das Programm gibt nun den Text aus, den du in die Anführungszeichen des `print`-Befehls geschrieben hast. Diese Methode ist zwar nicht allzu schwierig, wenn du den Texteditor PyCharm verwendest, geht das Ganze aber noch etwas einfacher:

Dafür musst du einfach wieder zu dem Fenster im PyCharm wechseln, in dem du das Programm verfasst hast. Darin entdeckst du am unteren Rand den Begriff **Terminal**. Wenn du diesen anklickst, öffnet sich ein neuer Bereich, in dem genau der gleiche Text angezeigt wird, wie beim Öffnen des Kommandozeilen-interpreters. Der Grund dafür ist, dass es sich hierbei nicht um eine eigene Funktion von PyCharm handelt. Das Programm bindet lediglich den Kommandozeileninterpreter ein und bietet einen einfacheren Zugang. Der einzige Unterschied zum vorherigen Beispiel besteht darin, dass du dich nun bereits von Anfang an im richtigen Verzeichnis befindest. Daher kannst du dir das mühsame Eintippen des Pfadnamens sparen. Der Befehl, der zum Ausführen des Programms zum Einsatz kommt, ist wieder genau der gleiche:

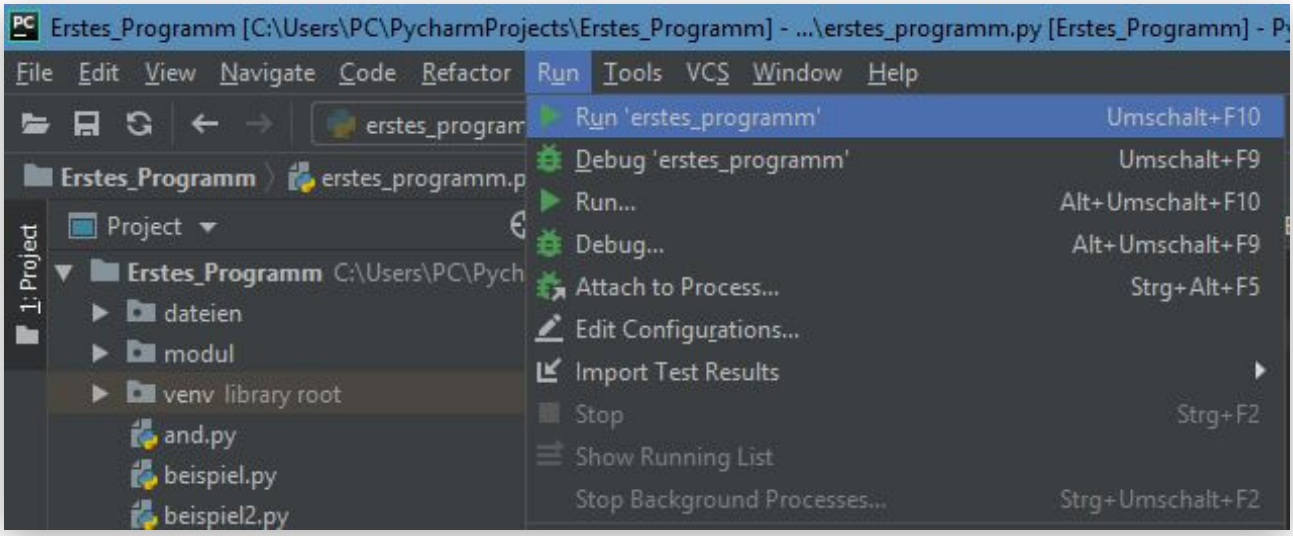
`erstes_programm.py` *bzw.* `python erstes_programm.py`



Screenshot: Die Ausführung des Programms direkt über PyCharm

Wie du siehst, ist der Effekt also genau der gleiche. Der Aufwand ist jedoch geringer. Außerdem erfolgt die Ausführung übersichtlich im gleichen Fenster. Aus diesen Gründen ist es sinnvoll, die Programme in Zukunft immer direkt über PyCharm auszuführen. Das Beispiel mit dem externen Kommandozeileninterpreter ist jedoch wichtig, um die Zusammenhänge und das Prinzip hinter der Ausführung von Programmen im Allgemeinen zu verstehen.

Darüber hinaus bietet PyCharm eine noch einfachere Möglichkeit: Du kannst die *Run-Funktion* aufrufen. Diese praktische Funktion ist auf mehrere Arten erreichbar. Du kannst beispielsweise mit der **rechten Maustaste** in den Code klicken. Daraufhin öffnet sich ein Menü, in dem du den Begriff **Run** auswählen musst. Eine weitere Möglichkeit besteht darin, die Menüleiste zu verwenden: über das Feld **Run** und anschließend über den Menüpunkt **Run 'erstes_programm'**. Besonders einfach ist es, wenn du auf das grüne Dreieck in der Werkzeugleiste klickst. Auch der Shortcut `Strg` + `Shift⇧` + `F10` (*Windows*) führt dich schnell ans Ziel. Jede dieser Alternativen führt dazu, dass PyCharm das Programm automatisch ausführt, ohne dass du dafür weitere Kommandos eintippen musst.



Screenshot: Der Aufruf der Run-Funktion über die Menüleiste

Am besten ist es, wenn du alle Alternativen ausprobierst und dich danach selbst entscheidest, welche dir am besten gefällt. Wenn du danach noch etwas üben willst, kannst du noch einige weitere Befehle in deine Programme einfügen und diese dann erneut ausführen. So lassen sich beliebige Nachrichten ausgeben.