1 Einführung

1.1 Python im Überblick

Python wurde Anfang der 1990er-Jahre von Guido von Rossum als Skriptsprache entwickelt und 1994 in der Version 1.0 veröffentlicht. Mittlerweile hat Python zwar schon mehr als 25 Jahre auf dem Buckel, wird aber nicht altersschwach, sondern kontinuierlich gepflegt und weiterentwickelt. Beispielsweise hat die bereits 2008 erschienene Hauptversion Python 3 diverse Aktualisierungen erfahren und liegt im Sommer 2021 in Version 3.9.6 vor. Für Oktober 2021 ist dann Python 3.10 geplant.

Hinweis: Python 2

Tatsächlich findet man immer noch Projekte und Informationen zur Vorgängerversion Python 2. Das neuere Python 3 ist an einigen Stellen nicht rückwärtskompatibel, bietet aber modernere Konzepte und sollte für Neuentwicklungen bevorzugt werden.

In den letzten Jahren wurde Python immer beliebter und ist nun eine der populärsten Programmiersprachen. Mittlerweile (im August 2021) hat Python laut TIOBE-Popularitätsindex[1] den 2. Platz erobert und damit Java als langjährigen Zweitplatzierten überholt. Python wird vielleicht auch bald den Spitzenreiter C überflügeln, wenn sich der Aufwärtstrend weiter so fortsetzt. Eine wichtige Rolle spielt vermutlich die freie Verfügbarkeit für alle relevanten Betriebssysteme. Zudem ist Python recht einfach zu erlernen (zum Einstieg deutlich einfacher als Java und insbesondere als C oder C++). Auch in der Forschung und Lehre wird Python zunehmend populärer und gewinnt an Bedeutung. Schließlich ermöglicht Python sowohl die objektorientierte als auch die funktionale Programmierung, sodass man je nach Einsatzzweck geeignet wählen kann.

Außerdem ist Programmieren ein wunderbares Hobby sowie ein faszinierender Beruf und es macht zudem noch jede Menge Freude, fördert die Kreativität und den Gestaltungswillen.

Wie Sie sehen, sprechen viele gute Gründe für einen Einstieg in die Programmierung mit Python. Das Wichtigste ist jedoch der Spaß am Programmieren, Tüfteln und Ausprobieren. Lassen Sie uns starten!

[1] `https://www.tiobe.com/tiobe-index/`

Bestandteile von Python-Programmen

Python als Programmiersprache besitzt wie eine natürliche Sprache auch eine Grammatik und feststehende Begriffe / Wörter. Man spricht dabei von Syntax und Schlüsselwörtern (vgl. Anhang A).

Python-Programme werden textuell verfasst. Das wird ***Sourcecode*** genannt. Schauen wir uns zum Einstieg zwei einfache Python-Programme an. Wir beginnen mit der skriptbasierten (Zeile für Zeile abgearbeiteten) Variante des kürzestmöglichen HelloWorld-Programms, der Ausgabe eines Grußes auf der Konsole:

```
print("Hello world!")
```

Diese Funktionalität lässt sich auch ein wenig komplizierter folgendermaßen im objektorientierten Stil mithilfe einer Klasse verfassen:

```
class HelloWorld:
    def greet(self):
        print("Hello world!")


greeter = HelloWorld()
greeter.greet()
```

Keine Sorge, Sie müssen das Ganze noch nicht verstehen, wir werden das alles Stück für Stück erlernen. Hier ist zunächst nur wichtig, dass Sie elementare Bestandteile von Python-Programmen grob einordnen können. Dazu gehören die Schlüsselwörter, also Python-Befehle oder -Anweisungen, hier etwa `class`, `def` und Funktions- bzw. Methodenaufrufe wie `print()` und `greet()`. Ebenso wie die Begriffe in einer Sprache tragen diese reservierten Wörter eine besondere Bedeutung, ganz analog etwa zu Auto, Haus, Tür usw. im Deutschen.

Ebenso wie im Deutschen können (oder besser sollten) die Begriffe nicht einfach wahllos miteinander verknüpft werden, denn dann ergibt dies keinen Sinn. Um einen gültigen Satz zu formulieren, bedarf es der Einhaltung einiger Regeln. Diese werden als Grammatik bezeichnet. Auch in Python existiert eine solche. Damit wird etwa festgelegt, dass es `def greet()`, aber nicht `greet() def` heißen muss.

Zudem sehen wir Einrückungen. Diese kann man sich wie Absätze in einem Text vorstellen. In Python bündeln diese Einrückungen Anweisungen. Man spricht dann auch von Blöcken.

Zunehmende Popularität von Python

Wie schon angedeutet, wird Python immer populärer. Beleuchten wir ein paar Gründe für diesen Trend. Zwei wesentliche sind sicher das breite Einsatzspektrum sowie die recht flache Lernkurve beim Einstieg: Erste Experimente gehen oftmals schnell von der Hand. Dabei hilft die auf das Wesentliche reduzierte Syntax (die wenigen Schlüsselwörter und Anweisungen) in Kombination mit einigen eleganten Sprachfeatures. Zum einfachen Ausprobieren existiert ein interaktiver Modus. Diesen starten Sie von der Kommandozeile mit dem Aufruf `python` (Windows) bzw. `python3` (MacOS). Dann sollte Python gestartet werden und dies in etwa wie folgt protokollieren:

```
Python 3.9.6 (default, Jun 29 2021, 06:20:32)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Die drei `>>>` zeigen an, dass Python nun auf Ihre Befehle wartet. In diesem Modus können Sie kleinere Programme Zeile für Zeile eingeben und direkt die Resultate sehen, etwa folgendermaßen:

```
>>> 7 * 6
42
>>> print("Willkommen zum Python-Buch")
Willkommen zum Python-Buch
```

Weitere Vorteile Die gut verständliche Formatierung und Gestaltung von Blöcken mit Einrückungen statt der in anderen Programmiersprachen üblichen geschweiften Klammern erleichtern den Einstieg. Neben dieser optischen Hilfe und Strukturierung besitzt Python nur eine überschaubare Anzahl an Befehlen (Schlüsselwörter genannt).

Damit eine Programmiersprache eine gewisse Popularität erreichen kann, muss sie fast zwangsläufig für alle gängigen Betriebssysteme wie Windows, MacOS und UNIX verfügbar sein. Das ist für Python gegeben.

Eine weitere Rolle spielt das sogenannte Ökosystem, also die Menge an Tools und Frameworks sowie Bibliotheken, die für eine Programmiersprache existieren. Lange Zeit war hier Java vorbildlich und extrem stark. Python holt diesbezüglich aber stetig auf und es gibt diverse gute Entwicklungstools und weitere Bibliotheken, viele vor allem im Bereich AI (Artificial Intelligence) und ML (Machine Learning).

Zwischenfazit

Genug der vielen Informationen. Nachfolgend werden wir die Dinge gründlich und detailliert besprechen und didaktisch immer ein neues Themengebiet ergründen, bis wir schließlich einen guten Einstieg in die Python-Programmierung vollzogen haben werden. Vorab werden wir aber erst einmal Python selbst und die IDE namens PyCharm installieren und erste Schritte ausführen, um für unsere weitere Entdeckungsreise gewappnet zu sein.

1.2 Los geht's – Installation

Im ersten Teil dieses Buchs wird ein Hands-on-Ansatz verfolgt, bei dem wir Dinge oftmals in Form kleinerer Python-Codeschnipsel direkt ausprobieren. Sie benötigen vorab keine tiefgreifenden Programmiererfahrungen, allerdings schaden diese natürlich nicht, ganz im Gegenteil. Hilfreich wäre allerdings, wenn Sie sich einigermaßen mit dem Installieren von Programmen und grundlegend mit der Kommandozeile auskennen.

Damit Sie die nachfolgend beschriebenen Python-Programme ausführen können, benötigen Sie eine aktuelle Python-Installation. Beginnen wir also damit.

1.2.1 Python-Download

Auf der Webseite `https://www.python.org/` ist die aktuelle Python-Version frei verfügbar.

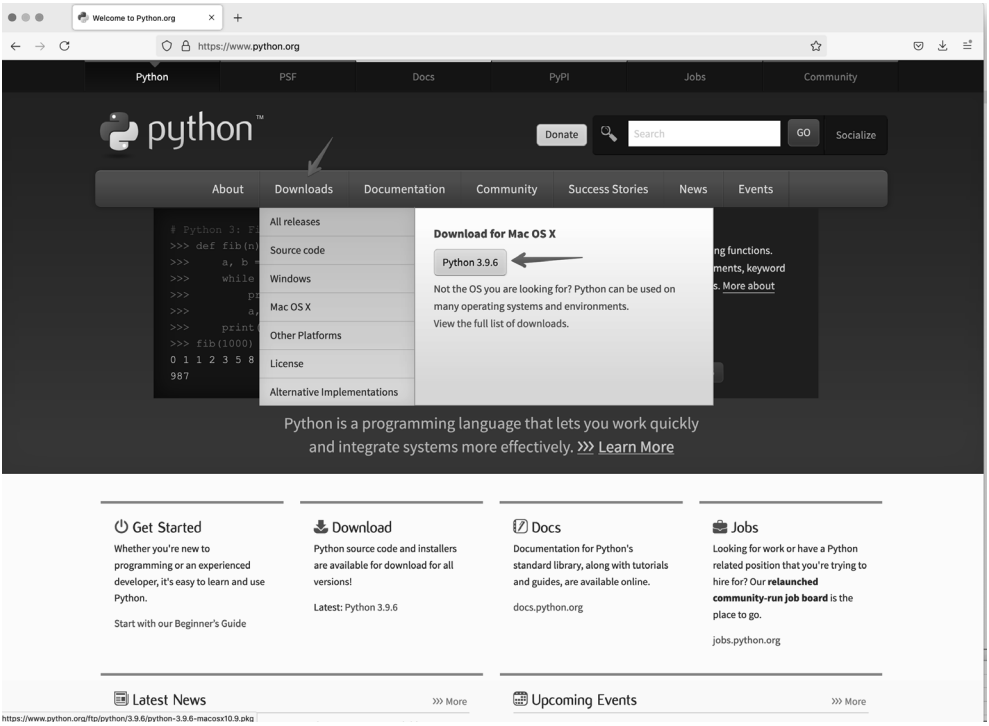

Abbildung 1-1 *Python-Download-Seite*

Nachdem Sie auf den Link zum Python-Download geklickt haben, startet der Download. Wenn dieser abgeschlossen ist, fahren Sie wie im Anschluss beschrieben fort.

1.2.2 Installation von Python

Die Installation von Python wird nachfolgend für die weitverbreiteten Betriebssysteme Windows und MacOS beschrieben. Falls Sie ein Unix-Derivat nutzen, dann finden Sie dort oftmals schon eine aktuelle Version von Python vorinstalliert.

Python-Installation für Windows

Für Windows doppelklicken Sie bitte auf die `.exe`-Datei, die nach erfolgreicher Installation gelöscht werden kann. Führen Sie also das heruntergeladene Installationsprogramm aus (z. B. `python-3.9.6-amd64.exe`). Damit wird Python installiert. Akzeptieren Sie die Standardeinstellungen und befolgen Sie die Anweisungen während der Installation. In einem Schritt kann auch angeklickt werden, dass der Pfad gesetzt werden soll. Das sollten Sie unbedingt anklicken, um spätere Aktionen und Änderungen an den Systemeinstellungen zu vermeiden.

Abbildung 1-2 *Python-Installation für Windows*

Python-Installation für MacOS

Unter MacOS doppelklicken Sie auf die `.pkg`-Datei, um die Installation zu starten. Folgen Sie den Aufforderungen. Möglicherweise müssen Sie das Administrator-Passwort eingeben, um fortzufahren. Nachdem die Installation abgeschlossen ist, können Sie die `.pkg`-Datei löschen, um Speicherplatz zu sparen.

1.2.3 Nacharbeiten nach der Python-Installation

Damit Python bei Ihnen nach der Installation auch in der Konsole korrekt funktioniert, sind mitunter noch ein paar Nacharbeiten nötig. Dazu müssen wir Python zur leichteren Handhabung in den Pfad aufnehmen. Dies wird nachfolgend für die weitverbreiteten Betriebssysteme Windows und MacOS beschrieben. Falls Sie ein Unix-Derivat nutzen, dann finden Sie dort oftmals schon eine aktuelle Version von Python vorinstalliert.

Nacharbeiten für Windows

Sofern nicht bereits während der Installation angeklickt, muss noch das Installationsverzeichnis in die Umgebungsvariable `PATH` aufgenommen werden. Diese können Sie unter »Umgebungsvariablen« ändern. Drücken Sie die Win-Taste und geben Sie dann »umgeb« ein, bis »Umgebungsvariablen für dieses Konto bearbeiten« erscheint. Mit Enter öffnet sich der Dialog »Umgebungsvariablen«. Hier wählen Sie den Eintrag »Path« aus und klicken auf den Button »Bearbeiten«. In die nun erscheinende Liste fügen Sie das Installationsverzeichnis ein. Dazu klicken Sie rechts auf den Button »Neu« und fügen dort zwei Verzeichnisse hinzu: Zuerst das Installationsverzeichnis, das bei einer Standardinstallation ungefähr so aussieht: `C:\Users\MINDEN\AppData\Local\Programs\Python\Python39`. Für das Unterverzeichnis Scripts lautet es ungefähr wie folgt: `C:\Users\MINDEN\AppData\Local\Programs\Python\Python39\Scripts`. Nach dem Schließen des Dialogs mit »OK« sollte das Ganze dann in etwa so aussehen:

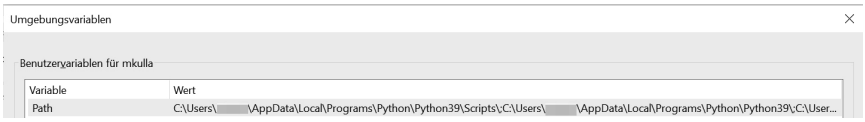

Abbildung 1-3 *Umgebungsvariablen bearbeiten*

Beachten Sie bitte Folgendes: Bestätigen Sie die gesamten Dialoge bitte immer mit »OK«, sodass die Variablen gesetzt sind. Eventuell geöffnete Konsolen müssen geschlossen und neu geöffnet werden, um die geänderten Variablen wirksam werden zu lassen.

Weitere Informationen finden Sie unter `https://docs.python.org/3/using/windows.html`.

Nacharbeiten für MacOS

Auch unter MacOS empfiehlt es sich, einen Verweis auf Python im Pfad in der jeweiligen Shell (dem Terminal) passend zu setzen. Normalerweise geschieht dies bei einer Installation automatisch. Ansonsten können Sie dies von Hand ausführen bzw. in das Startskript Ihrer Shell eintragen, etwa `~/.bash_profile` oder neuer `~/.zshrc`:

```
export PYTHON_HOME=/Library/Frameworks/Python.framework/Versions/3.9/
export PATH=$PYTHON_HOME/bin:$PATH
```

1.2.4 Python-Installation prüfen

Nachdem Sie die obigen Schritte ausgeführt haben, sollte Python auf Ihrem Rechner installiert und von der Konsole startbar sein und Sie damit bereit sein für die nächsten Schritte.

Öffnen Sie eine Konsole und geben Sie das unten stehende Kommando ein – im folgenden Text nutze ich immer $ zur Kennzeichnung von Eingaben auf der Konsole, also der Windows-Eingabeaufforderung (`python`) und dem Terminal bei MacOS (`python3`) wie folgt:

```
$ python3
Python 3.9.6 (default, Jun 29 2021, 06:20:32)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
```

Sofern Sie ähnliche Meldungen erhalten (möglicherweise mit kleinen Abweichungen bei den Versionsangaben), können wir uns auf die Entdeckungsreise zur Python-Programmierung machen. Wir beenden den interaktiven Modus wie folgt, um noch die skriptbasierte Ausführung kurz kennenzulernen.

```
>>> exit()
```

1.2.5 Python-Programm als Skript ausführen

Wie schon erwähnt, kann man Python-Programme auch Zeile für Zeile ausführen lassen. Dazu kann man die Python-Anweisungen zunächst mit einem Texteditor eingeben, beispielsweise folgende Anweisungen:

```
print("Herzlich Willkommen zu 'Einfach Python'")
print("Viel Spaß wünscht Ihnen Michael Inden")
```

Speichern Sie dies als `welcome.py` ab. Danach können Sie die Anweisungen folgendermaßen starten:

```
$ python3 welcome.py
```

Python lädt und analysiert das Programm aus der Datei `welcome.py` und gibt als Folge diese Meldungen aus:

```
Herzlich Willkommen zu 'Einfach Python'
Viel Spaß wünscht Ihnen Michael Inden
```

1.3 Entwicklungsumgebungen

Zum Schreiben von umfangreicheren Python-Programmen (also zum Bearbeiten von viel Sourcecode) empfehle ich den Einsatz einer IDE anstelle von Texteditoren oder anstatt rein auf der Konsole zu arbeiten. Für kleine Experimente ist aber gerade der Python-Kommandozeileninterpreter ein wunderbares Hilfsmittel.

Für Änderungen an größeren Python-Programmen kann man zwar auch mal einen Texteditor nutzen, aber dieser bietet nicht die Annehmlichkeiten einer IDE: In IDEs laufen verschiedene Aktionen und Sourcecode-Analysen automatisch und im Hintergrund ab, wodurch gewisse Softwaredefekte direkt noch während des Editierens erkannt und angezeigt werden können, etwa in einer To-do-/Task-Liste. IDEs bereiten zudem vielfältige Informationen auf. Des Weiteren werden diverse Annehmlichkeiten wie Quick Fixes zur Korrektur kleinerer Probleme sowie automatische Transformationen und Änderungen von Sourcecode, sogenannte ***Refactorings***, unterstützt.

Für Python existieren verschiedene IDEs. Momentan besitzt PyCharm ganz klar eine Vorreiterrolle. Es basiert auf dem für Java populären IntelliJ IDEA. Praktischerweise gibt es PyCharm als kostenlose Community Edition sowie als kostenpflichtige Ultimate Edition. Mehr Informationen zu PyCharm sowie zum Download finden Sie im Anschluss und unter `https://www.jetbrains.com/pycharm/`.

Wenn Sie bereits etwas Erfahrung haben, dann sind Sie natürlich frei, sich andere IDEs anzuschauen und diese auszuprobieren. Vieles geht über persönliche Präferenzen.

1.3.1 Installation von PyCharm

Gehen Sie auf die Seite `https://www.jetbrains.com/pycharm/`. Diese präsentiert sich ähnlich zur folgenden Abbildung. Dort finden Sie oben rechts einen Download-Button, den Sie bitte drücken.

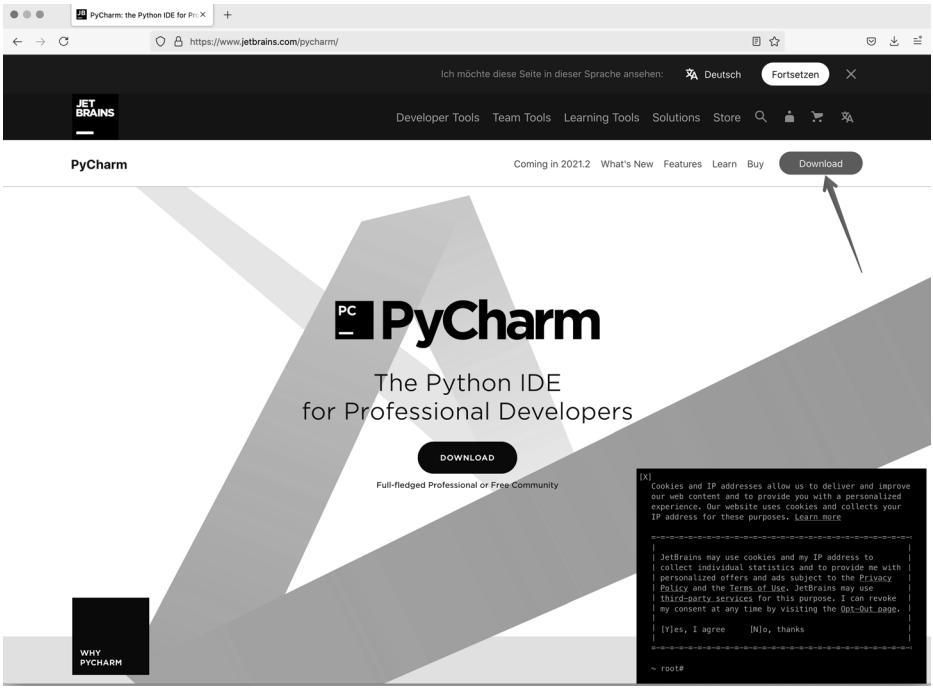

Abbildung 1-4 *PyCharm-Hauptseite zum Download*

Hinweis: Möglicherweise leicht abweichende Screenshots

Denken Sie bitte daran: Bei Ihnen können die nachfolgend gezeigten Abbildungen leicht abweichen, insbesondere, wenn Sie Windows verwenden. Die Screenshots wurden unter MacOS Big Sur erstellt.

Dadurch wird die Download-Seite mit ein paar Auswahlmöglichkeiten geöffnet. Hier können Sie unter anderem die Version für das gewünschte Betriebssystem sowie die kostenlose Community Edition oder die kostenpflichtige Ultimate Edition auswählen. Ein Klick auf den jeweiligen Download-Button startet dann den Download.

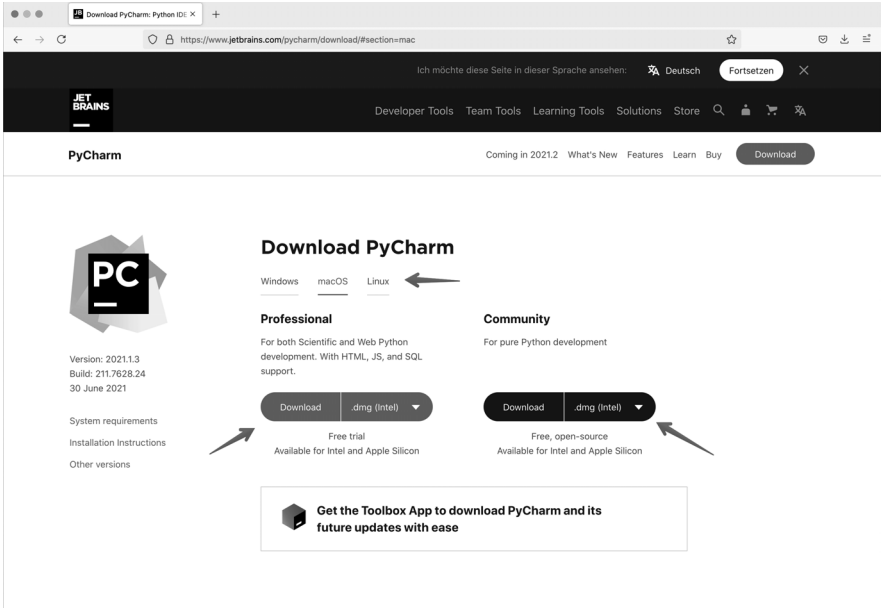

Abbildung 1-5 *PyCharm-Hauptseite zum Download (Fortsetzung)*

Nachdem der Download abgeschlossen ist, entpacken oder starten Sie bitte das jeweilige heruntergeladene ZIP, DMG oder die Datei im jeweiligen Linux-Format. Für MacOS wird ein Fenster geöffnet, wo Sie das Programm per Drag and Drop in den Programme-Ordner verschieben können. Im Falle von Windows installieren Sie PyCharm bitte über die `.exe`-Datei. Der Einfachheit halber klicken Sie alle Optionen unter »Installation Options« an (64-bit launcher, add launchers dir to the PATH etc.). Ansonsten können Sie die vorgeschlagenen Werte übernehmen. Nach der Installation (und einem Reboot) können Sie PyCharm über das Startmenü oder das Desktop-Icon öffnen.

1.3.2 PyCharm starten

Nach den beschriebenen Installationsschritten sollte Ihnen nun PyCharm als Programm im Startmenü bzw. der Programmauswahl zur Verfügung stehen. Starten Sie es bitte, etwa durch einen Doppelklick auf das Programm-Icon.

Bei MacOS erhalten Sie gegebenenfalls noch einen Warnhinweis, dass es sich um ein aus dem Internet heruntergeladenes Programm handelt. Dies können Sie ignorieren und durch einen Klick auf Öffnen fortfahren.

Abbildung 1-6 *Warnmeldung (MacOS)*

Als Nächstes öffnet sich der Startbildschirm von PyCharm, der sich je nach Version leicht unterschiedlich präsentiert. Hier werden verschiedene Optionen angeboten, etwa das Anlegen neuer Projekte oder das Öffnen bestehender:

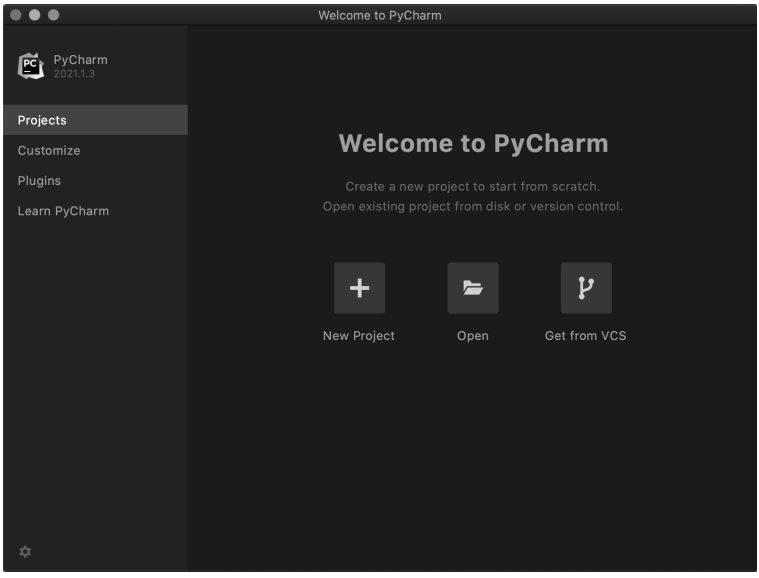

Abbildung 1-7 *Projekt anlegen oder öffnen*

Das Importieren wird separat in der Beschreibung zum Download der Sourcen zu diesem Buch thematisiert. Wir schauen uns nachfolgend das Anlegen eines Projekts mitsamt einer einfachen Klasse sowie deren Start an.

Wenn Sie bereits Projekte erstellt oder importiert haben, dann sehen Sie in etwa folgende Darstellung:

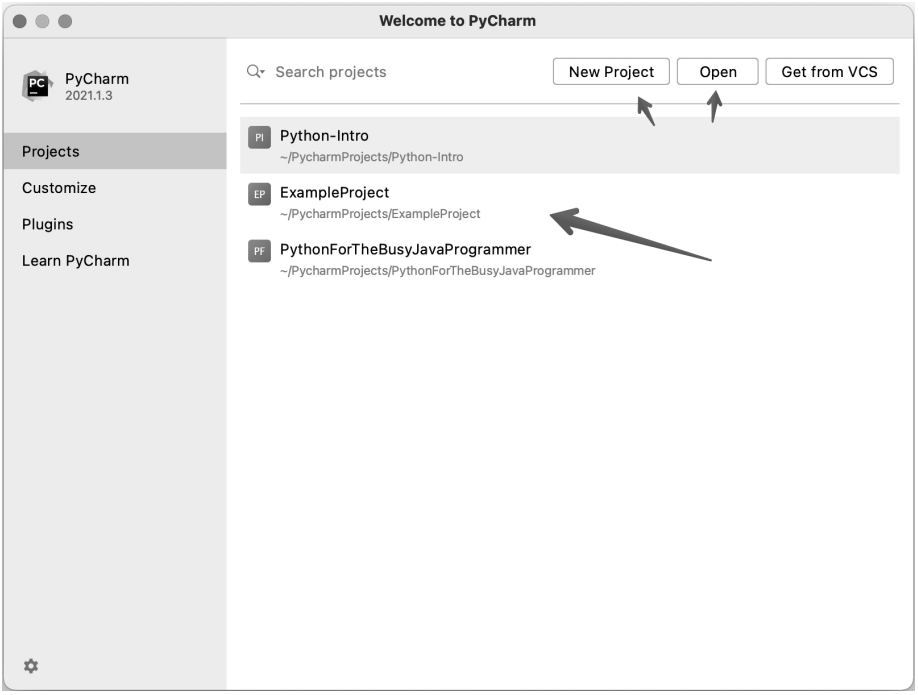

Abbildung 1-8 *Projektliste mit Auswahl zum Projekt anlegen oder öffnen*

Nun wollen wir uns dem Anlegen eines Projekts und eines ersten Python-Moduls sowie einer ersten Klasse widmen, um die Abläufe exemplarisch einmal durchzuspielen, die für spätere Aktionen notwendig sind. Keine Sorge, Sie müssen diese Schritte nicht im Detail verstehen. Es baut sich aber ein erstes Verständnis auf, das sich dann weiter vertieft, wenn Sie dies häufiger machen.

Beginnen wir mit dem Anlegen eines Python-Projekts. Klicken wir zunächst auf den Button `New Project`.

1.3.3 Erstes Projekt in PyCharm

Zum Anlegen eines Projekts öffnet sich folgender Dialog, in dem Sie unter `Location` am Ende den gewünschten Namen des Projekts, hier `MyFirstPythonProject`, eintragen. Zudem müssen Sie noch die zu verwendende Python-Installation angeben und dazu gegebenenfalls das passende Installationsverzeichnis wählen. Zum Schluss empfiehlt es sich, für erste Experimente noch das Erzeugen eines initialen Python-Moduls `main.py` per Checkbox zu aktivieren. All dies ist durch Pfeile in der Abbildung gekennzeichnet.

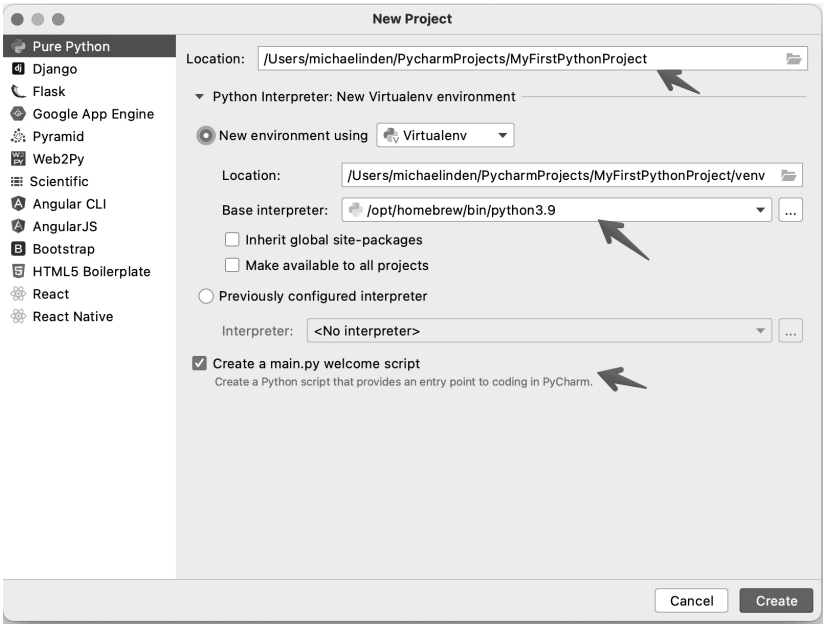

Abbildung 1-9 *Dialog »New Project«*

1.3.4 Erstes Modul in PyCharm

Unser erstes Python-Projekt ist jetzt angelegt und ist bereit, um mit Leben in Form von Modulen und Klassen gefüllt zu werden. PyCharm sollte sich in etwa wie folgt präsentieren. Praktischerweise findet sich schon ein einfaches Grundgerüst als Datei `main.py`, wo Sie mit ersten Experimenten beginnen können.

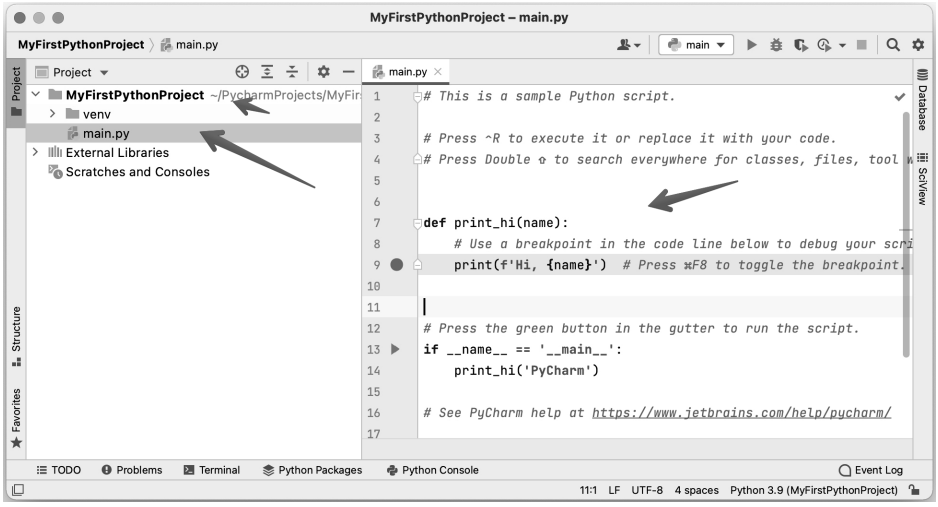

Abbildung 1-10 *Neu angelegtes Projekt*

Neues Modul anlegen

Ausgangspunkt zum Anlegen eines neuen Moduls ist die Baumdarstellung auf der linken Seite im Project Explorer. Dort öffnet man (durch einen Rechtsklick oder Ctrl + Klick) ein Kontextmenü unter anderem mit dem Eintrag `New > Python File`.

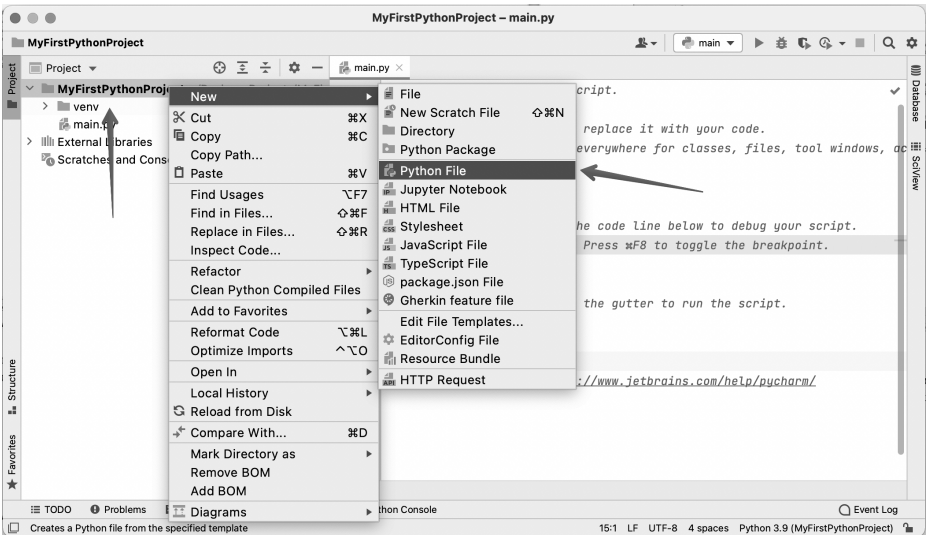

Abbildung 1-11 *Kontextmenü zum Anlegen einer Python-Datei*

Durch Auswahl des Kontextmenüs öffnet sich folgender Dialog zum Erzeugen einer neuen Python-Datei.

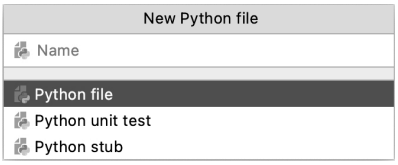

Abbildung 1-12 *Dialog zum Erstellen einer neuen Python-Datei*

Im Dialog muss man den gewünschten Dateiname eingeben, woraufhin diese Datei dann erzeugt wird. Wir nutzen `myfirstpythonmodul` als Namen.

Sourcecode editieren

Nachdem das Grundgerüst steht, können Sie dann den Sourcecode aus den Beispielen im Editor einfügen bzw. abtippen. Generell können Sie viele der hier gezeigten Aktionen auch zum Nachvollziehen der für den Python-Kommandozeileninterpreter gezeigten Programmschnipsel nutzen, indem Sie die Programmzeilen einfach im Editorfenster eintippen.

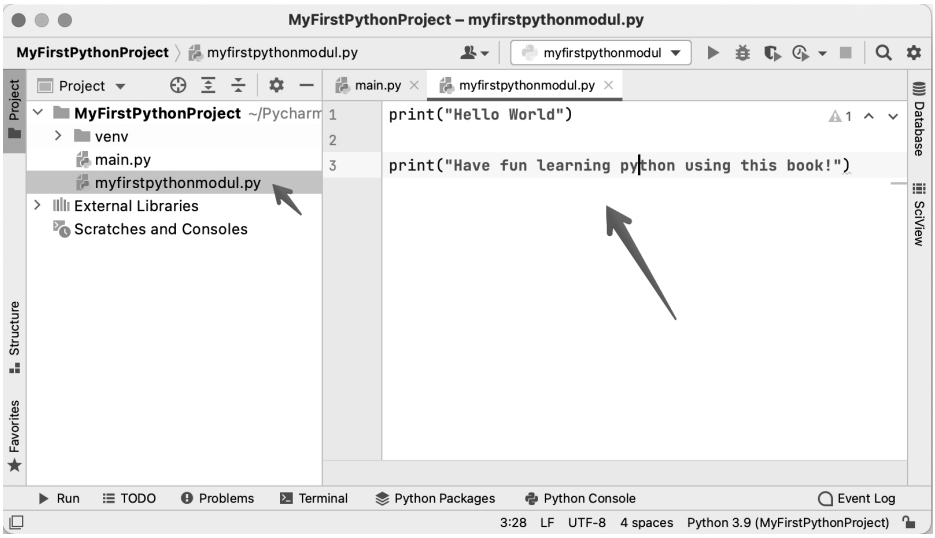

Abbildung 1-13 *Sourcecode editieren*

Klasse ausführen

Schließlich wollen Sie sicherlich das so entstandene Python-Programm auch mal in Aktion erleben. Dazu ist eine `main()`-Funktion hilfreich, wie sie im vorherigen Beispiel zu sehen war. Die `main()`-Funktion ist jedoch optional. Wird diese nicht definiert, so werden die Python-Anweisungen im bereits vorgestellten Skriptmodus Zeile für Zeile ausgeführt.

Mithilfe des Kontextmenüs oder des grünen Play-Pfeils kann man die Ausführung in beiden Varianten starten. Allerdings muss beim ersten Mal immer über das Kontextmenü gestartet werden, weil dann eine Startkonfiguration angelegt wird. Diese erlaubt es, danach auch über den grünen Pfeil zu starten.

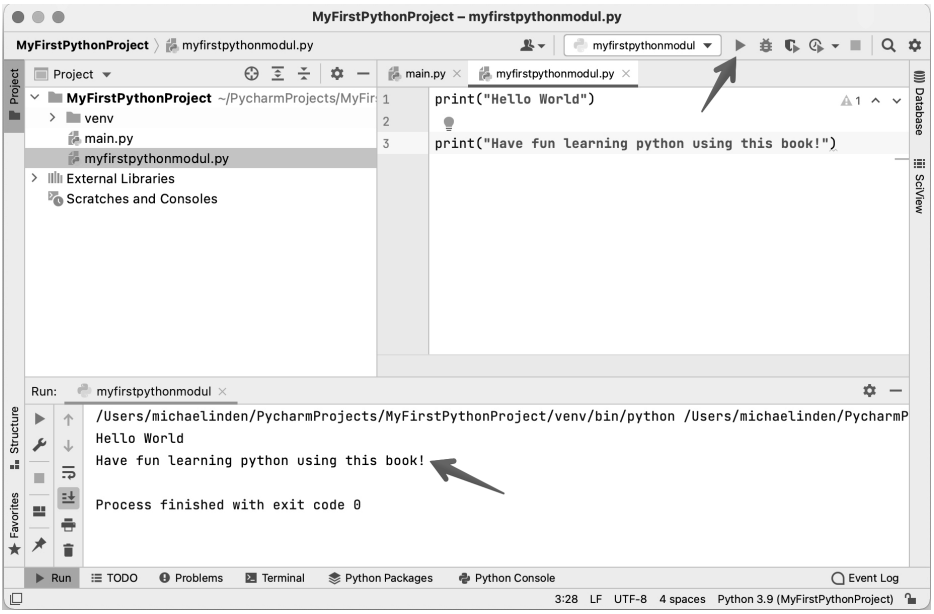

Abbildung 1-14 *Programm ausführen*