1 Einführung

1.1 Python im Überblick

Python wurde Anfang der 1990er-Jahre von Guido von Rossum entwickelt und 1994 in Version 1.0 veröffentlicht. Mittlerweile hat Python zwar schon rund 30 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 seit Oktober 2023 in der Version 3.12 vor. Im Oktober 2024 ist Python 3.13 erschienen.

Hinweis: Python 2

Tatsächlich findet man vereinzelt immer noch Projekte und Informationen zur Vorgängerversion Python 2. Das neuere Python 3 ist an einigen Stellen nicht rückwärtskompatibel, etwa bei `print()`, das in Python 2 noch ein Kommando war und mittlerweile eine Funktion ist, oder aber bei Aktionen auf Listen, wie Filterungen, die im Gegensatz zu Python 2 nicht mehr direkt ausgeführt werden, sondern ein Ergebnis zum Weiterverarbeiten liefern. Insgesamt bietet Python 3 modernere Konzepte und sollte für Neuentwicklungen bevorzugt werden. Deswegen wird Python 2 in diesem Buch nicht weiter behandelt.

In den letzten Jahren wurde Python immer beliebter und ist mittlerweile die populärste Programmiersprache. Stand November 2024 hat Python laut TIOBE-Popularitätsindex[1] den 1. Platz inne und diesen bereits vor über einem Jahr erobert. Eine wichtige Rolle spielt vermutlich die freie Verfügbarkeit für alle relevanten Betriebssysteme. Zudem ist Python recht leicht 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.

Darüber hinaus 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 ebenso wie eine natürliche Sprache 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. Zum Einstieg betrachten wir zwei einfache Python-Programme. Wir beginnen mit der ***skriptbasierten*** (Zeile für Zeile abgearbeiteten) Variante des kürzestmöglichen Hello-World-Programms, der Ausgabe eines Grußes auf der Kommandozeile:

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

Diese Zeile speichern Sie etwa in einer Datei `helloworld.py` und können diese später ausführen. Das schauen wir uns in Kürze an.

Die gezeigte Funktionalität lässt sich auch etwas komplizierter folgendermaßen im objektorientierten Stil mithilfe einer Klasse (vgl. Kapitel 4) verfassen – solche Darstellungen von Sourcecode nennt man auch ***Listing***:

```
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 den Nomen Auto, Haus, Tür usw. oder den Verben laufen, knobeln etc. im Deutschen.

Genau wie im Deutschen können (oder besser sollten) Begriffe nicht wahllos miteinander verknüpft werden, denn dann ergibt dies höchstwahrscheinlich 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()` heißen muss, aber nicht `greet() def`. Man spricht dann davon, dass das Programm syntaktisch korrekt ist, wenn es den Regeln der Grammatik entspricht.

Zudem sehen wir Einrückungen. Diese kann man sich wie Absätze in einem Text vorstellen. In Python bündeln diese Einrückungen Anweisungen. Man spricht auch von Blöcken oder Suites. Einrückungen sind im Gegensatz zu anderen Programmiersprachen wie Java äußerst wichtig. Wird in einem Python-Programm falsch eingerückt oder werden Einrückungen ganz weggelassen, so lässt sich das Programm nicht ausführen.

Gründe für die 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.

Einfaches Experimentieren und erste Schritte Erste Experimente mit Python 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 weniger Anweisungen existiert ein interaktiver Modus. Dieser steht nach der im Anschluss (vgl. Abschnitt 1.2) beschriebenen Installation von Python automatisch zur Verfügung.

Im folgenden Text nutze ich immer $ zur Kennzeichnung von Eingaben auf der Kommandozeile (eine Kurzeinführung finden Sie in Abschnitt 1.3). Von dort starten Sie Python im interaktiven Modus mit dem Aufruf `python` (Windows) bzw. `python3` (macOS). Dies sollte in etwa wie folgt protokolliert werden:

```
$ python3
Python 3.13.0 (v3.13.0:60403a5409f, Oct  7 2024, 00:37:40) [Clang 15.0.0 (clang
      -1500.3.9.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

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

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

Wie bereits angedeutet ist es alternativ ebenfalls möglich, die abgebildeten Zeilen in einer Datei mit der Endung `.py`, etwa als `intro.py`, zu speichern – dabei müssen Berechnungen dann entweder einer Variablen zugewiesen, `result = 7 * 6`, oder wie im nachfolgenden Listing gezeigt per `print()` ausgegeben werden:

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

Eine solche Datei mit Python-Anweisungen lässt sich dann wie folgt ausführen – mehr Details zu dieser skriptbasierten Ausführung behandelt Abschnitt 1.2.5:

```
$ python3 <dateiname>.py
```

Konkret würde ein Aufruf so aussehen und die nachfolgenden Ausgaben produzieren:

```
$ python3 intro.py
42
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 eine überschaubare Anzahl an Befehlen (***Schlüsselwörter***).

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 externen Tools und Frameworks sowie Bibliotheken, die für eine Programmiersprache existieren. Lange Zeit war hier Java vorbildlich und extrem stark. Python spielt in den Bereichen Künstliche Intelligenz (KI) und Machine Learning (ML) eine Vorreiterrolle und holt in anderen Bereichen stetig auf: Es gibt diverse gute Entwicklungstools und weitere Bibliotheken, herausragend sind eben die in den Bereichen KI und ML.

Als Vorgeschmack sei auf die Kapitel 11 und 12 hingewiesen. Im ersten von beiden beschäftigen wir uns mit verschiedenen Bibliotheken zur Verarbeitung von Bildern und zum Ausführen von Web-Requests. Das zweite genannte Kapitel liefert einen Einstieg in KI und beleuchtet einige Möglichkeiten von Large Language Models (LLMs), etwa die Generierung von Texten, Audioausgaben in MP3 und Bildern.

Zwischenfazit

Genug der vielen Informationen. Nachfolgend werden wir die Dinge gründlich besprechen und didaktisch immer wieder neue Themengebiete ergründen, bis wir schließlich einen guten Einstieg in die Python-Programmierung vollzogen haben werden. Vorab wollen wir Python selbst und optional die IDE namens PyCharm[2] 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 dazu keine tiefgreifenden Programmiererfahrungen, allerdings schaden diese natürlich nicht, ganz im Gegenteil. Hilfreich wäre jedoch, 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.

Hinweis: Möglicherweise leicht abweichende Screenshots

Denken Sie bitte daran: Bei Ihnen können die Abbildungen aufgrund von unterschiedlichen Versionen leicht abweichen, insbesondere, wenn Sie Windows verwenden. Die Screenshots wurden unter macOS Sonoma erstellt.

2 `https://www.jetbrains.com/pycharm`

1.2.1 Python-Download

Auf der Webseite `https://www.python.org` ist die aktuelle Python-Version frei verfügbar (vgl. Abbildung 1-1).

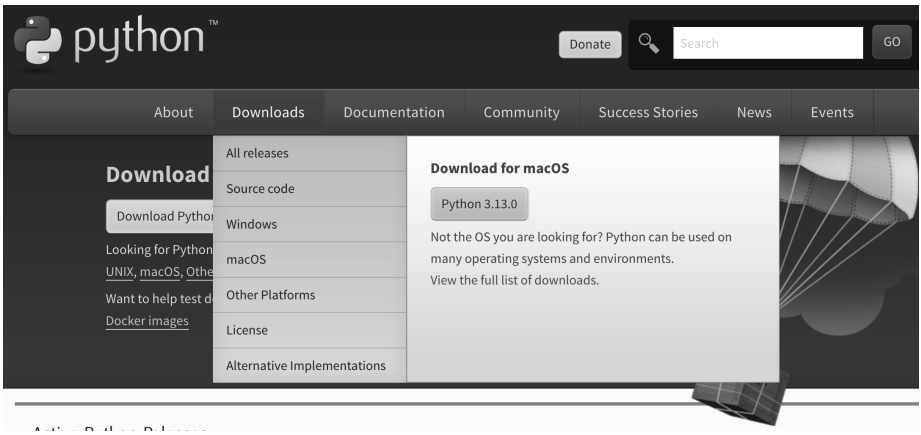

Abbildung 1-1 *Python-Download-Seite*

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 das heruntergeladene Installationsprogramm aus (z. B. `python-3.13.0-amd64.exe`). Damit wird Python installiert. Akzeptieren Sie die Standardeinstellungen und befolgen Sie die Anweisungen während der Installation. In einem Schritt kann angeklickt werden, dass Python der Umgebungsvariablen `PATH` hinzugefügt werden soll. Das sollten Sie unbedingt auswählen, um spätere Aktionen und Änderungen an den Systemeinstellungen zu vermeiden.

Exemplarisch ist die Python-Installation für Windows inklusive der auszuwählenden Aktionen in der nachfolgenden Abbildung 1-2 gezeigt.

Abbildung 1-2 *Python-Installation für Windows*

Python-Installation für macOS

Für macOS sollten Sie eine `.pkg`-Datei (z. B. `python-3.13.0-macos11.pkg`) heruntergeladen haben, die Sie doppelklicken, 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 in der Kommandozeile korrekt funktioniert, sind mitunter ein paar Nacharbeiten nötig. Dazu sollten wir Python zur leichteren Handhabung in den Pfad aufnehmen. Dies wird nachfolgend für die Betriebssysteme Windows und macOS beschrieben. Falls Sie ein Unix-Derivat nutzen, dann finden Sie dort oftmals eine aktuelle Version von Python vorinstalliert und passend konfiguriert.

Nacharbeiten für Windows

Sofern nicht bereits während der Installation angeklickt, sollte das Installationsverzeichnis in die Umgebungsvariable `PATH` aufgenommen werden. Diese können Sie unter »Umgebungsvariablen« (engl. Environment Variables) 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 auf den Button »Neu« und fügen dort zwei Verzeichnisse hinzu: Zum einen das Installationsverzeichnis, das standardmäßig etwa so aussieht: `C:\Users\<NUTZERNAME>\`

`AppData\Local\Programs\Python\Python313`. Zum anderen das Unterverzeichnis Scripts, für das der Eintrag ungefähr wie folgt lautet: `C:\Users\<NUTZERNAME>\AppData\Local\Programs\Python\Python313\Scripts`. Nach dem Schließen des Dialogs mit »OK« sollte das Ganze dann in etwa so aussehen:

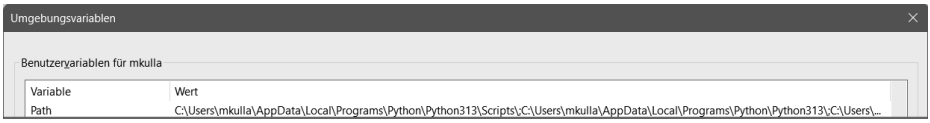

Abbildung 1-3 *Umgebungsvariablen bearbeiten*

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

Weitere Informationen finden Sie auf der Webseite `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 folgende Kommandos eintragen, etwa `~/.bash_profile` oder neuer `~/.zshrc`:

```
$ export PYTHON_HOME=/Library/Frameworks/Python.framework/Versions/3.13/
$ 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 Kommandozeile startbar sein und Sie damit bereit für die nächsten Schritte. Öffnen Sie eine Kommandozeile – falls Sie sich damit noch etwas schwertun, finden Sie in Abschnitt 1.3 eine Kurzeinführung in die Kommandozeile. Von dort starten Sie Python im interaktiven Modus mit `python` (Windows Eingabeaufforderung) oder `python3` (macOS Terminal). Nochmals zur Erinnerung: Im folgenden Text nutze ich $ zur Kennzeichnung von Eingaben auf der Kommandozeile, um es unabhängig vom jeweiligen Betriebssystem einheitlich darstellen zu können:

```
$ python3
Python 3.13.0 (v3.13.0:60403a5409f, Oct  7 2024, 00:37:40) [Clang 15.0.0 (clang
      -1500.3.9.4)] 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 mit `exit()` oder seit Python 3.13 kürzer mit `exit`, um anschließend noch die skriptbasierte Ausführung kennenzulernen.

```
>>> exit
```

1.2.5 Python-Programm als Skript ausführen

Wie schon erwähnt, kann man Python-Anweisungen nicht nur interaktiv auf der Kommandozeile direkt eingeben und ausführen, sondern auch in Form einer Textdatei bereitstellen und die dortigen Anweisungen Zeile für Zeile ausführen lassen. Dazu müssen die gewünschten Python-Anweisungen zunächst mit einem Texteditor eingegeben werden, beispielsweise folgende zwei Zeilen:

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

Speichern Sie diese Zeilen etwa als Datei `welcome.py` ab. Danach können Sie die Anweisungen als Python-Programm unter macOS folgendermaßen starten – für Windows nutzen Sie `python` zum Aufruf statt `python3` (macOS).

```
$ python3 welcome.py
```

Dieser Aufruf startet Python, das dann die Anweisungen aus der angegebenen Datei, hier `welcome.py`, lädt und analysiert und diese schließlich ausführt. Für dieses Beispiel werden als Folge diese Meldungen auf der Kommandozeile ausgegeben:

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

1.3 Arbeiten mit der Kommandozeile (Kurzeinführung)

Gerade habe ich bereits das eine oder andere Mal die Kommandozeile erwähnt. Möglicherweise haben Sie noch nicht so viel Erfahrung und tun sich bei deren Start noch etwas schwer. Um diese Einstiegshürde zu überwinden, möchte ich eine kurze Einführung geben, damit Sie dann sicher Python im interaktiven Modus von der Kommandozeile starten können.

Nachfolgend zeige ich Ihnen zuerst die Schritte für macOS und danach die für Windows.

1.3.1 Für macOS

Terminal starten

Unter macOS heißt das Programm für die Kommandozeile »Terminal«. Um dieses zu starten, führen Sie bitte alternativ eine der beiden folgenden Varianten aus.

Variante 1 – Dock und Launchpad Klicken Sie im Dock auf das Symbol des Launchpad (vgl. Abbildung 1-4).

Abbildung 1-4 *Dock und Launchpad*

Daraufhin erscheint eine Iconliste mit allen Programmen. Im oberen Teil befindet sich ein Suchfeld. Dort geben Sie bitte `Terminal` ein. Während der Eingabe wird die Liste der dem Suchbegriff entsprechenden Programme immer stärker eingeschränkt und es sollte das zum `Terminal` gehörende Programmsymbol erscheinen. Dieses können Sie anklicken, wie es nachfolgend in Abbildung 1-5 durch den Pfeil angedeutet ist.

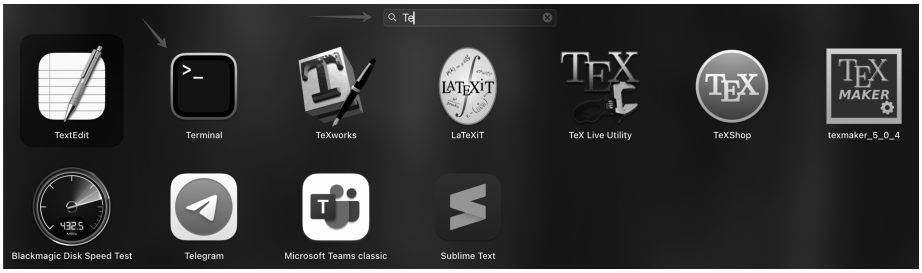

Abbildung 1-5 *Dienstprogramme mit Filterung auf »Te«*

Variante 2 – Dock und Finder Klicken Sie im Dock auf das Symbol des Finder (vgl. Abbildung 1-6).

Abbildung 1-6 *Dock und Finder*

Öffnen Sie dort den Ordner `/Programme/Dienstprogramme` und doppelklicken Sie dann auf `Terminal` wie es nachfolgend in Abbildung 1-7 durch den unteren Pfeil angedeutet ist.

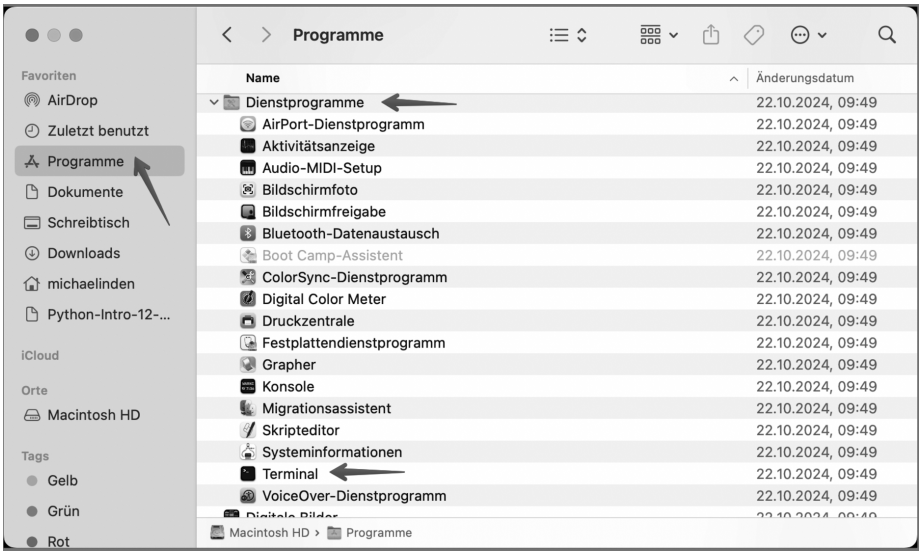

Abbildung 1-7 *Dienstprogramme im Finder*

Resultat: Terminal-Fenster Unabhängig von den beiden beschriebenen Wegen sollte sich ein Terminal-Fenster, also die Kommandozeile, analog zu dem in Abbildung 1-8 gezeigten öffnen – Datumsangaben und Rechnername werden bei Ihnen vermutlich abweichen. :-)

Abbildung 1-8 *Terminal*

Für das Terminal verwende ich im Buch als Abkürzung immer das $-Zeichen. Hier geben Sie Kommandos ein oder führen die Installation von Python-Bibliotheken und Tools durch.

Python im Terminal starten

Um die im Text abgebildeten Python-Kommandos ausführen zu können, müssen Sie den Python-Kommandozeileninterpreter (auch REPL für Read-Eval-Print-Loop genannt) für macOS durch Eingabe von `python3` starten. Nachfolgend ist das Ganze in Abbildung 1-9 nach dem Start von Python gezeigt. Alle im Buch mit `>>>` markierten Aktionen finden dann dort statt.

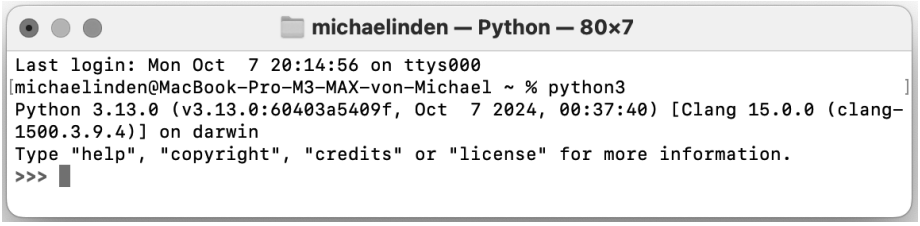

Abbildung 1-9 *Python im Terminal starten*

1.3.2 Für Windows

Eingabeaufforderung starten

Unter Windows heißt das Programm für die Kommandozeile »Eingabeaufforderung« (`cmd.exe`). Um diese zu starten, führen Sie bitte alternativ eine der beiden folgenden Varianten aus.

Variante 1 – Run-Menü Eine Möglichkeit ist die Tastaturkombination `Win + R` zum Öffnen des Run-Menüs. Dort können Sie `cmd` eintippen und den Dialog mit `OK` bestätigen. Daraufhin wird die Eingabeaufforderung gestartet (vgl. Abbildung 1-10).

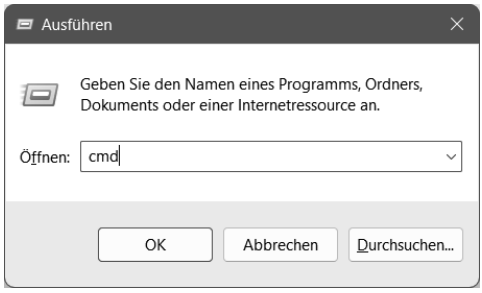

Abbildung 1-10 *Run-Menü und Eingabeaufforderung*

Variante 2 – Startmenü Öffnen Sie das Startmenü des Computers. Klicken Sie dazu auf das Windows-Icon (oftmals in der unteren linken Ecke des Desktops) oder drücken Sie die Win-Taste. Es erscheint das Startmenü, in dem Sie `cmd` eintippen. In der Liste sollte die Eingabeaufforderung als oberster Eintrag angezeigt werden. Mit Enter oder durch Anklicken mit der Maus können Sie die Anwendung starten.

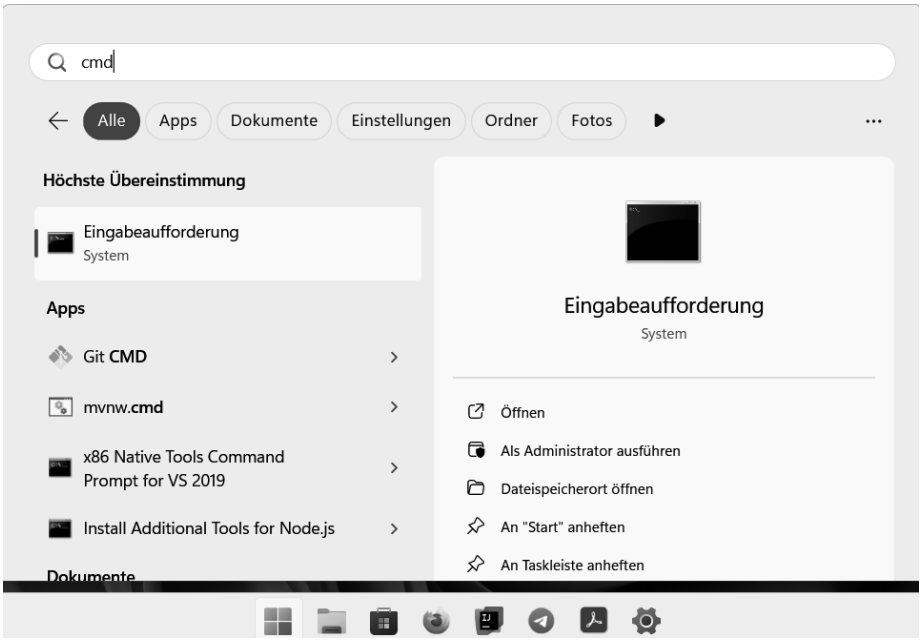

Abbildung 1-11 *Startmenü und Eingabeaufforderung*

Resultat: Eingabeaufforderungfenster Unabhängig von den beiden Wegen sollte sich ein Eingabeaufforderungfenster ähnlich zu dem in Abbildung 1-12 gezeigten öffnen. Nochmals zur Erinnerung: Für diese Eingabeaufforderung verwende ich als Abkürzung immer das $-Zeichen. Hier führen Sie Kommandos wie das Starten von Python oder die Installation von Python-Bibliotheken und Tools durch.

Abbildung 1-12 *Eingabeaufforderung*

Python in der Eingabeaufforderung starten

Um die im Text abgebildeten Python-Kommandos ausführen zu können, müssen Sie den Python-Kommandozeileninterpreter (auch REPL für Read-Eval-Print-Loop genannt) für Windows durch Eingabe von `python` starten. Nachfolgend ist das Ganze in Abbildung 1-13 nach dem Start von Python gezeigt. Alle im Buch mit >>> markierten Aktionen finden dann dort statt.

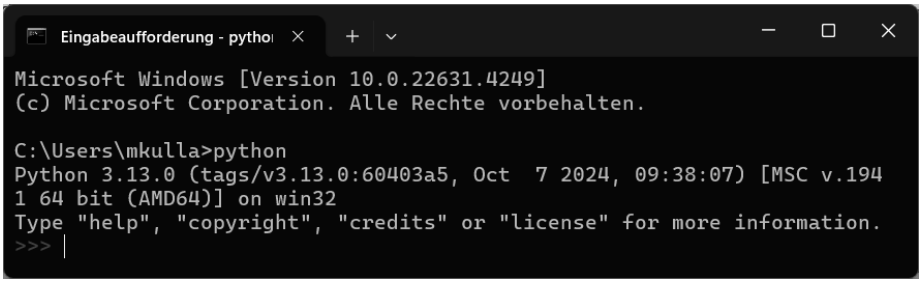

Abbildung 1-13 *Python in der Eingabeaufforderung starten*

1.3.3 IDLE als Alternative

Sowohl für macOS als auch für Windows wird in einer Python-Installation das Programm IDLE (Integrated Development and Learning Environment) mitgeliefert, das eine (sehr) einfache grafische Oberfläche besitzt und mehr oder weniger ein Fenster rund um die Python-Kommandozeile ist. Nachfolgend zeige ich für die IDLE wiederum zuerst die Schritte für macOS und danach die für Windows.

Für macOS

Um die IDLE für macOS zu starten, folgen Sie dem bereits beschriebenen Vorgehen zum Starten des Terminal-Programms mithilfe von Finder. Wechseln Sie dort in das Verzeichnis `Programme`, wie es in Abbildung 1-14 angedeutet ist.

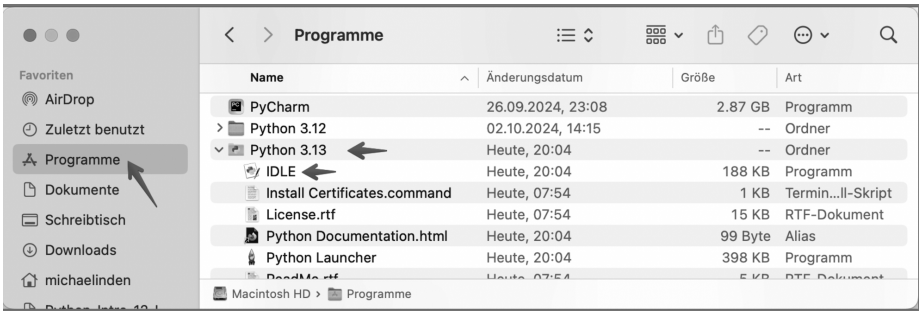

Abbildung 1-14 *IDLE im Programme-Verzeichnis*

Danach wählt man das zur gewünschten Python-Version korrespondierende Verzeichnis, hier `Python 3.13`. Dort befindet sich dann ein Programm namens `IDLE`, dass sich durch Doppelklick starten lässt.

Nach dem Starten der IDLE (vgl. Abbildung 1-15) kann man Experimente analog zum Python-Kommandozeileninterpreter durchführen.

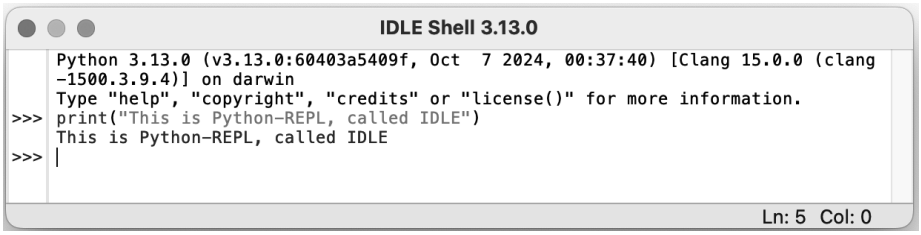

Abbildung 1-15 *IDLE unter macOS*

Für Windows

Um die IDLE für Windows zu starten, folgen Sie dem zuvor beschriebenen Weg für das Startmenü, wo Sie im Suchfeld `IDLE` eintippen. In der Liste sollte das Programm als oberster Eintrag erscheinen, wie es Abbildung 1-16 zeigt.

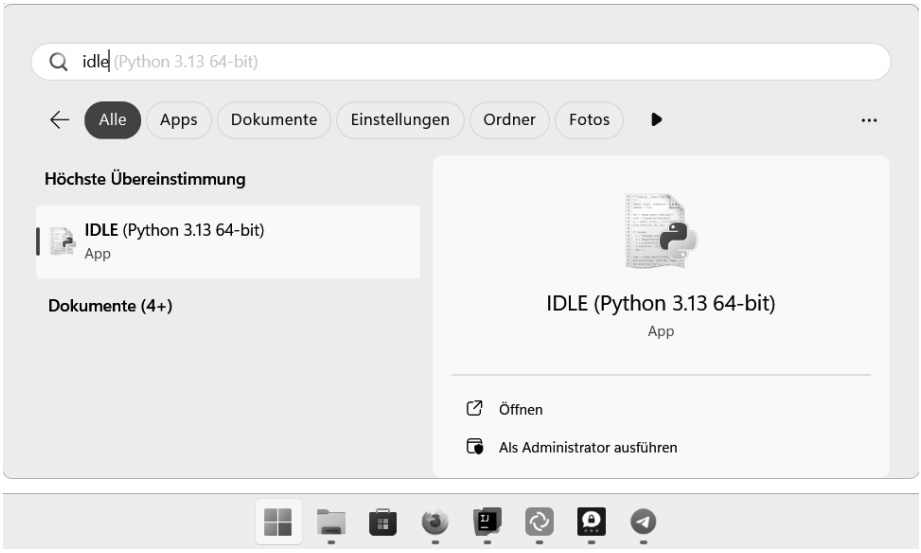

Abbildung 1-16 *IDLE im Programme-Verzeichnis*

Mit Enter oder durch Anklicken können Sie die IDLE starten (vgl. Abbildung 1-17) und Experimente analog zum Python-Kommandozeileninterpreter durchführen.

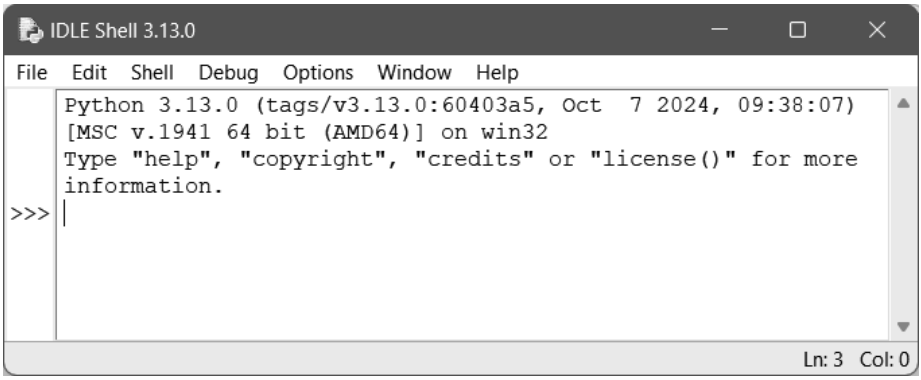

Abbildung 1-17 *IDLE unter Windows*

Hinweis

Abschließend sei noch angemerkt, dass das Bearbeiten mit der IDLE nicht wirklich komfortabel ist. Diese Möglichkeit, Python-Programme zu bearbeiten, erwähne ich nur der Vollständigkeit halber. Je intensiver Sie Python nutzen wollen, desto mehr empfiehlt sich der Einsatz einer Entwicklungsumgebung, auch IDE genannt. Das Thema behandle ich im Anschluss.

1.4 Entwicklungsumgebung PyCharm (Optional)

Bevor es mit dem Thema Entwicklungsumgebung, auch IDE (Integrated Development Environment) genannt, losgeht, sei noch auf Folgendes hingewiesen: Generell werden wir uns in diesem Buch nicht allzu stark mit IDEs beschäftigen. Somit ist sowohl die Installation einer IDE als auch die Lektüre dieses Unterkapitels optional. Sie können es daher zunächst überspringen und bei Bedarf später hierher zurückkehren.

Bislang haben wir zum Ausführen von Python-Programmen folgende Varianten kennengelernt: den Python-Kommandozeileninterpreter, die Python-IDLE sowie die Ausführung als Skript mit vorherigem Erstellen und Bearbeiten von Python-Anweisungen in einem Texteditor. Tatsächlich reichen diese Varianten für viele Experimente und für einen unbeschwerten Start vollkommen aus. Das gilt ebenfalls für viele der im Buch enthaltenen Beispiele. Nachfolgend möchte ich Ihnen das Thema Entwicklungsumgebung/IDE trotzdem schon einmal einführend näherbringen, weil Sie nach den ersten Schritten im Python-Kommandozeileninterpreter vermutlich mehr Komfort und bessere Unterstützung beim Entwickeln suchen.

Je intensiver Sie sich mit Python und dem Programmieren auseinandersetzen, desto mehr werden die Komplexität und der Umfang Ihrer Programme wachsen. Ab einem gewissen Punkt steht der Komfort, mit einer IDE zu arbeiten, in keinem Vergleich zu Texteditoren oder dem Programmieren ausschließlich über die Kommandozeile. Allerdings ist das zugegebenermaßen für den ersten Einstieg noch nicht nötig. Denn für

kleine Experimente ist gerade der Python-Kommandozeileninterpreter ein wunderbares Hilfsmittel, das wir zum Einstieg bevorzugt nutzen werden.

Ab Kapitel 4 werden die Programme tendenziell etwas umfangreicher. In solchen Fällen lohnt es sich zur Bearbeitung eher, eine IDE einzusetzen. Die folgende Tabelle zeigt Ihnen, in welchen Kapiteln für die Beispiele vorwiegend der Python-Kommandozeileninterpreter (man spricht auch von REPL für Read-Eval-Print-Loop) und/oder die IDE genutzt bzw. die Programme dort ausgeführt werden.

Kapitel	**Einsatz REPL / IDE**
Kapitel 2 Schnelleinstieg	REPL
Kapitel 3 Strings	REPL
Kapitel 4 Klassen und Objektorientierung	REPL + IDE
Kapitel 5 Collections	REPL
Kapitel 6 Ergänzendes Wissen	REPL
Kapitel 7 Verarbeitung von Dateien	REPL + IDE
Kapitel 8 Collections Advanced	REPL + IDE
Kapitel 9 Fehlerbehandlung	REPL + IDE
Kapitel 10 Datumsverarbeitung	REPL + IDE
Kapitel 11 Bildverarbeitung und Kontakt zur Außenwelt	REPL + IDE
Kapitel 12 LLMs mit Python im Kurzüberblick	REPL + IDE

Vorteile von Entwicklungsumgebungen

Für Änderungen an größeren Python-Programmen kann man zwar auch einmal einen Texteditor nutzen, aber dieser bietet nicht den Komfort einer IDE: In IDEs laufen verschiedene Sourcecode-Analysen automatisch und im Hintergrund ab, wodurch gewisse Defekte direkt während des Modifizierens von Sourcecode (auch ***Editieren*** genannt) erkannt und passend angezeigt werden können. Es liegt an Ihnen, diese dann auch zeitnah zu beheben. :-) Darüber hinaus bieten IDEs weitere Annehmlichkeiten wie Quick Fixes zur Korrektur kleinerer Probleme sowie automatische Transformationen und Änderungen von Sourcecode, sogenannte ***Refactorings***.

Momentan besitzt PyCharm eine Vorreiterrolle bei den Python-IDEs. Es basiert auf dem für Java populären IntelliJ IDEA. Praktischerweise bietet PyCharm eine kostenlose Community Edition, die für die Aufgabenstellungen dieses Buchs vollkommen ausreicht. Das gilt sogar teilweise noch für professionelles Arbeiten – nur bei hohen Ansprüchen empfiehlt sich die kostenpflichtige Ultimate Edition. Mehr Informationen zu PyCharm sowie zum Download finden Sie im Anschluss.

Wenn Sie bereits ein wenig Erfahrung haben, dann sind Sie natürlich frei, auch andere IDEs auszuprobieren. Vieles geht über persönliche Präferenzen.

1.4.1 Installation von PyCharm

Öffnen Sie die Seite `https://www.jetbrains.com/pycharm`. Diese präsentiert sich ähnlich zur folgenden Abbildung 1-18. Dort finden Sie oben rechts sowie unten links je einen Download-Button, von denen Sie bitte einen drücken.

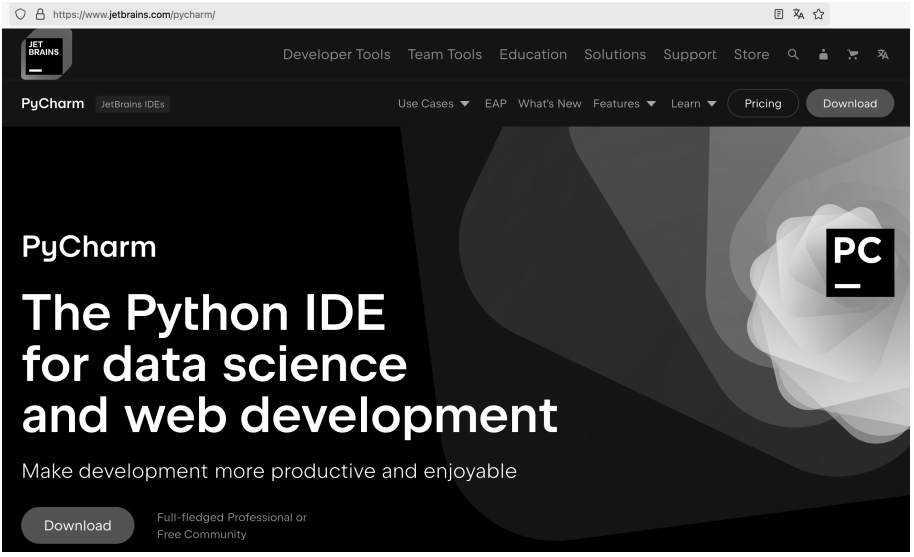

Abbildung 1-18 *PyCharm-Hauptseite zum Download*

Dadurch wird die Download-Seite geöffnet (vgl. Abbildung 1-19).

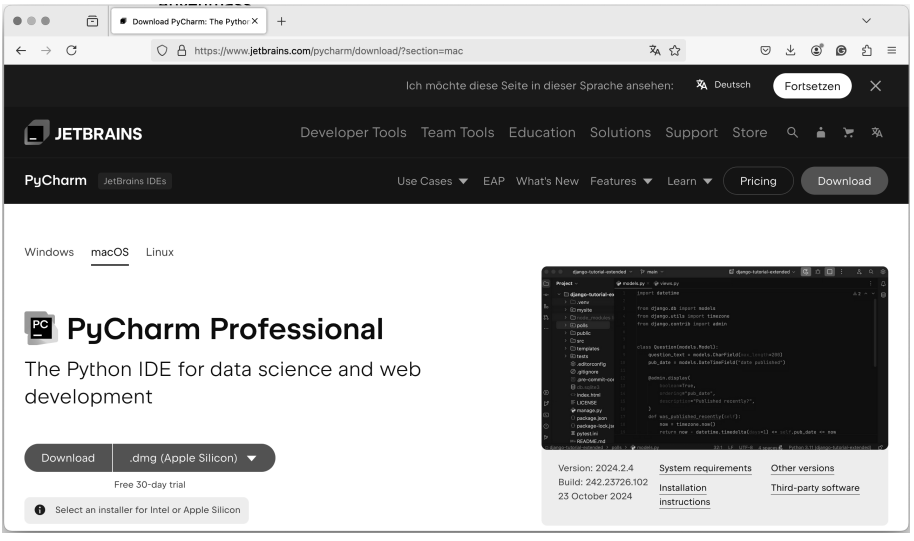

Abbildung 1-19 *PyCharm-Hauptseite zum Download (Fortsetzung)*

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.

Nachdem der Download abgeschlossen ist, entpacken oder starten Sie bitte das heruntergeladene ZIP, DMG oder die Datei im jeweiligen Unix-Format. Für macOS wird ein Fenster geöffnet, in dem 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 gegebenenfalls einem Neustart) lässt sich PyCharm über das Startmenü oder das Desktop-Icon öffnen.

1.4.2 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 einen Warnhinweis, dass es sich um ein aus dem Internet heruntergeladenes Programm handelt. Diesen können Sie ignorieren und wie in Abbildung 1-20 dargestellt durch einen Klick auf Öffnen fortfahren.

Abbildung 1-20 *Warnmeldung (macOS)*

Als Nächstes öffnet sich der Startbildschirm von PyCharm (vgl. Abbildung 1-21), der sich je nach Version leicht unterschiedlich präsentiert. Hier werden verschiedene Optionen angeboten, etwa das Anlegen neuer Projekte oder das Öffnen bestehender.

Begrifflichkeiten

Beginnen wir mit ein paar Begrifflichkeiten: Ein ***Projekt*** bündelt verschiedene Dateien, insbesondere solche mit Python-Kommandos, aber auch Texte, Bilder usw. Eine Datei, die Python-Kommandos, beispielsweise Variablendefinitionen, Funktionen und weitere Anweisungen enthält, nennt man ***Modul*** und im Dateisystem enden diese standardmäßig mit `.py`. Der zugehörige Modulname ergibt sich aus dem Dateinamen, allerdings ohne die Endung `.py`.

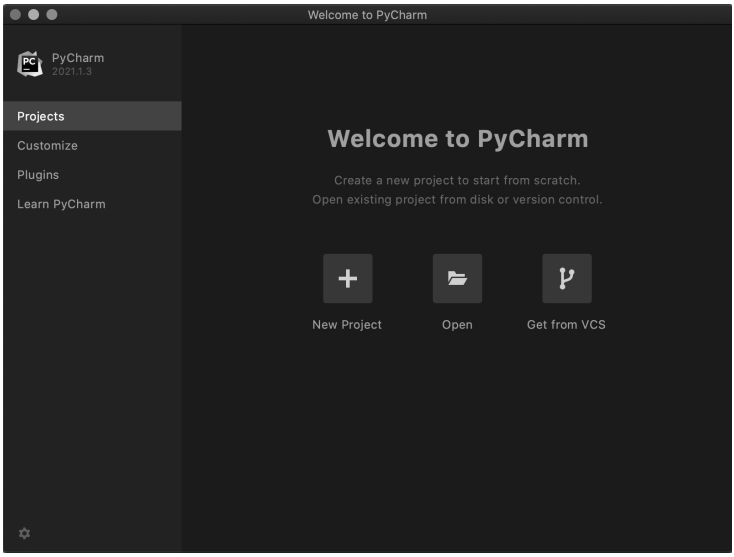

Abbildung 1-21 *Projekt anlegen oder öffnen*

Wenn Sie bereits Projekte erstellt oder importiert haben, dann sehen Sie statt der vorherigen Abbildung in etwa folgende Darstellung (vgl. Abbildung 1-22).

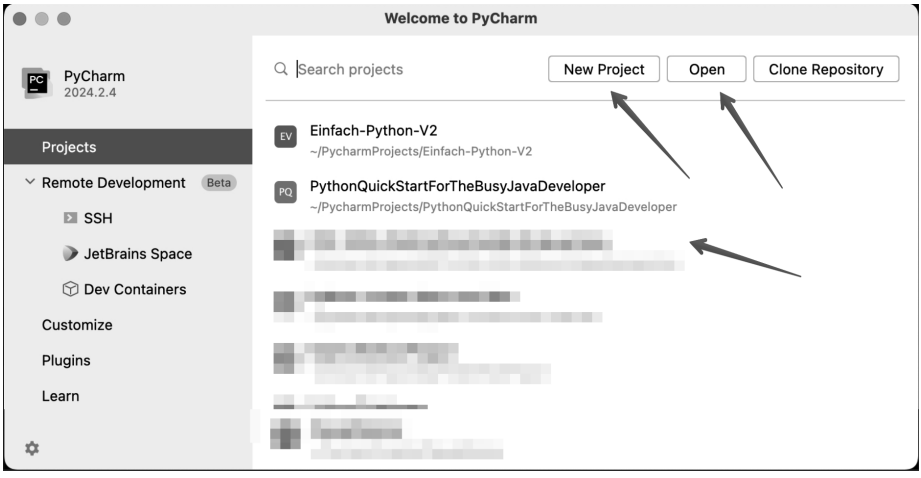

Abbildung 1-22 *Projektliste mit Auswahl zum Projekt anlegen oder öffnen*

Im Anschluss wollen wir uns dem Anlegen eines Projekts und eines ersten Python-Moduls widmen, um die Abläufe exemplarisch einmal durchzuspielen, die für spätere Aktionen notwendig sind.

1.4.3 Erstes Projekt in PyCharm

Beginnen wir mit dem Anlegen eines Python-Projekts. Klicken Sie zunächst auf den Button `New Project`. Zum Anlegen eines Projekts öffnet sich folgender Dialog, in dem Sie im Textfeld `Name` den gewünschten Namen des Projekts, hier `MyFirstPythonProject`, eintragen. Zudem müssen Sie die zu verwendende Python-Installation angeben und dazu gegebenenfalls das passende Installationsverzeichnis wählen. Zum Schluss empfiehlt es sich, für erste Experimente das Erzeugen eines initialen Python-Moduls (`Create a welcome script`) per Checkbox zu aktivieren. All dies ist durch Pfeile in der Abbildung 1-23 gekennzeichnet.

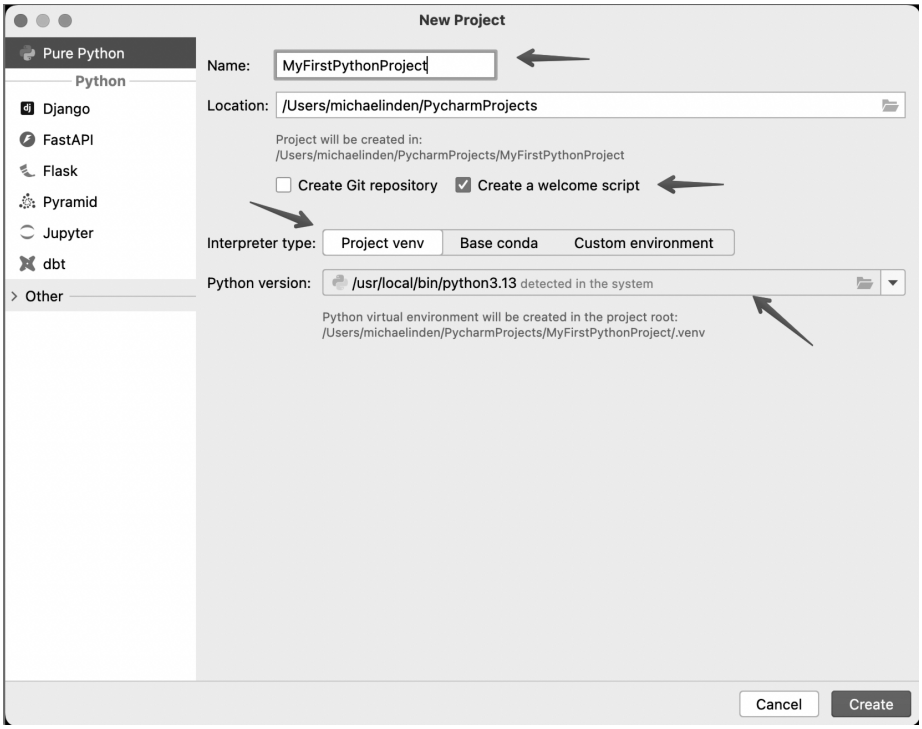

Abbildung 1-23 *Dialog »New Project«*

1.4.4 Erstes Modul in PyCharm

Unser erstes Python-Projekt ist jetzt angelegt und bereit, um mit Leben in Form von Modulen, also Dateien mit der Endung `.py`, die Python-Anweisungen enthalten, gefüllt zu werden. PyCharm sollte sich in etwa wie in Abbildung 1-24 präsentieren. Praktischerweise findet sich schon ein einfaches Grundgerüst als Datei `main.py`, die bereits ein paar Python-Kommandos enthält (siehe rechte Seite der Abbildung). Mit diesen können Sie Ihre ersten Experimente beginnen.

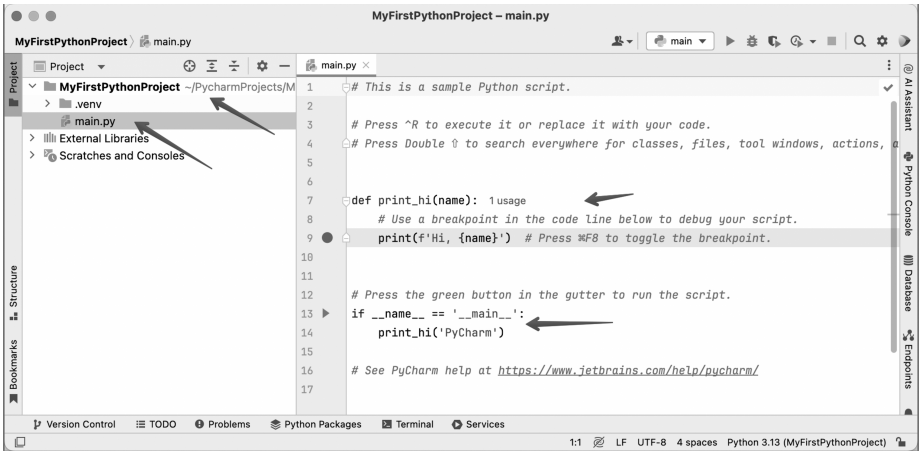

Abbildung 1-24 *Neu angelegtes Projekt*

Neues Modul (Python-Datei) anlegen

Der Ausgangspunkt zum Anlegen eines neuen Moduls / einer Python-Datei ist die Baumdarstellung des Project Explorer auf der linken Seite in Abbildung 1-25. Dort öffnen Sie (durch einen Rechtsklick oder Ctrl + Klick) ein Kontextmenü und klicken darin auf den Eintrag `New > Python File`.

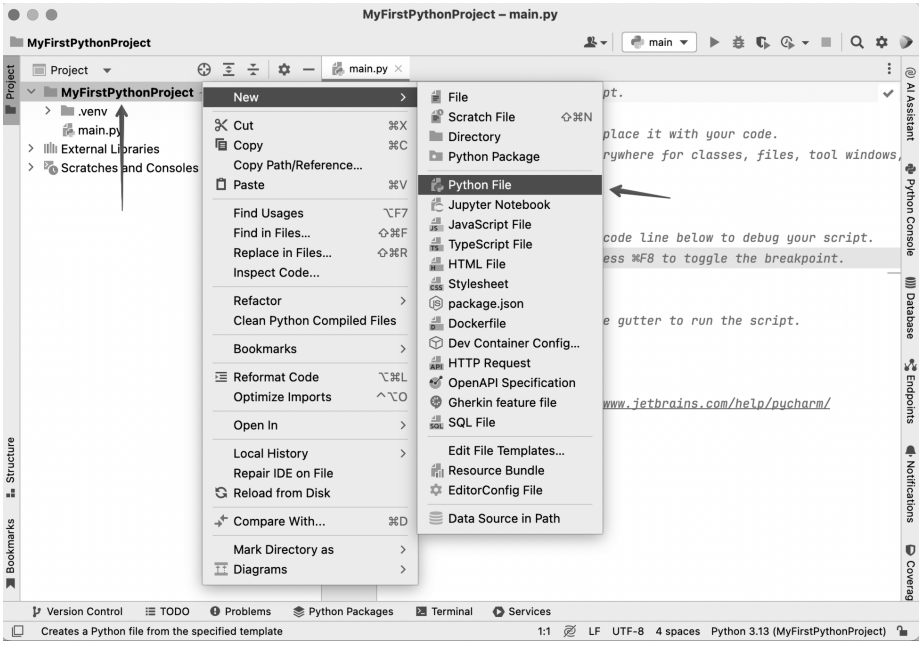

Abbildung 1-25 *Kontextmenü zum Anlegen einer Python-Datei*

Durch Auswahl des Kontextmenüs öffnet sich folgender Dialog zum Erzeugen einer neuen Python-Datei (vgl. Abbildung 1-26):

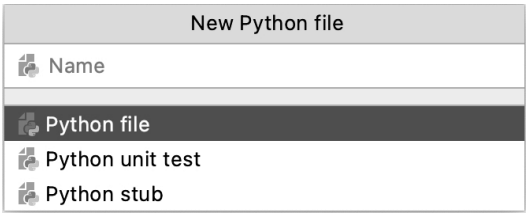

Abbildung 1-26 *Dialog zum Erstellen einer neuen Python-Datei*

Im Dialog gibt man den gewünschten Dateinamen ein, woraufhin diese Datei dann erzeugt wird. Wir nutzen `myfirstpythonmodule` als Namen.

Sourcecode editieren

Nachdem das Grundgerüst steht, können Sie den Sourcecode aus den Beispielen in das Editorfenster auf der rechten Seite einfügen bzw. dorthin abtippen. Generell können Sie Programmschnipsel, die mit dem Python-Kommandozeileninterpreter demonstriert und ausgeführt wurden, alternativ ausprobieren, indem Sie die Programmzeilen im Editorfenster eintippen, wie es durch den großen Pfeil und die beiden Ausgaben mit `print()` in Abbildung 1-27 angedeutet ist.

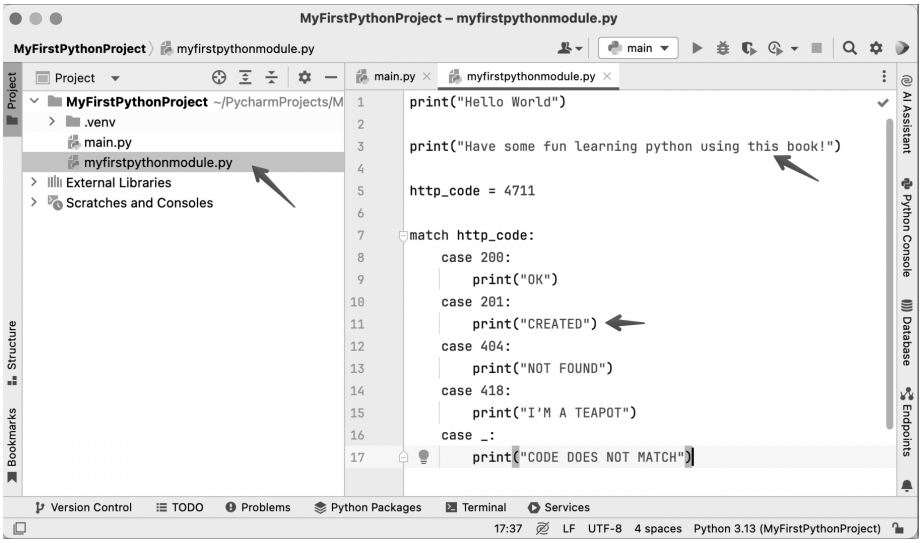

Abbildung 1-27 *Sourcecode editieren*

Programm ausführen

Schließlich wollen Sie bestimmt die entwickelten Python-Programme in Aktion erleben. Dazu ist ein Einstiegspunkt mit einer Prüfung `if __name__ == '__main__':` hilfreich, wie dies im Beispiel in Abbildung 1-24 zu sehen war. Ein solcher Einstiegspunkt ist jedoch optional. Wird dieser nicht definiert, so werden die Python-Anweisungen einfach im Skriptmodus Zeile für Zeile von oben nach unten ausgeführt, wie dies bereits in Abschnitt 1.2.5 vorgestellt wurde.

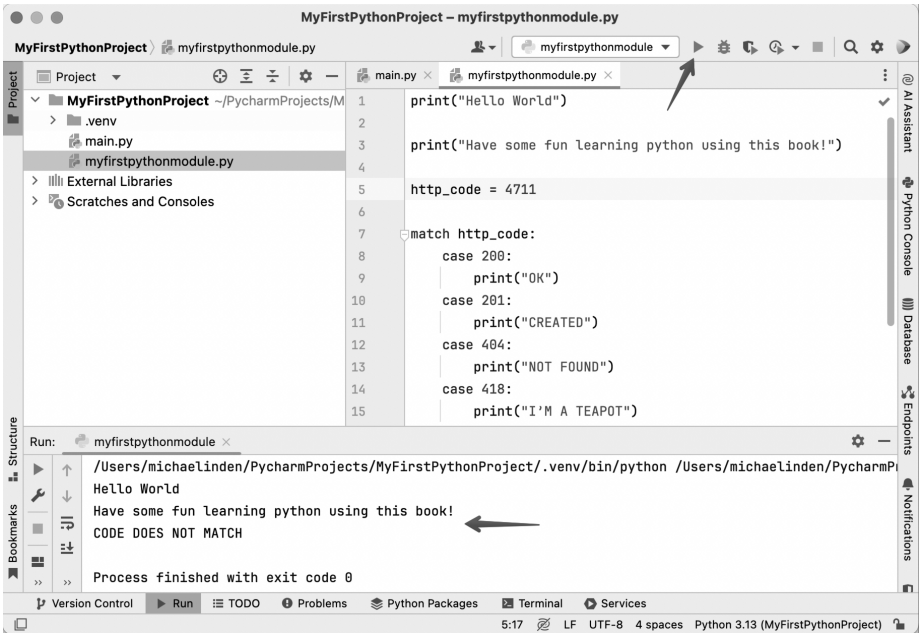

Abbildung 1-28 *Programm ausführen*

Mithilfe des Run-Eintrags im Kontextmenü oder mithilfe des grünen Play-Pfeils (vgl. Abbildung 1-28) kann man die Ausführung 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.

1.5 Ausprobieren der Beispiele

Bevor wir unsere Entdeckungsreise starten, möchte ich nochmals auf das Ausprobieren der Beispiele kurz eingehen.

Vielfach können Sie die abgebildeten Sourcecode-Schnipsel einfach in den Python-Kommandozeileninterpreter oder mithilfe eines Texteditors bzw. besser noch einer IDE eingeben. Nachfolgend motiviere ich ein wenig, wann die eine oder andere Art zu bevorzugen ist. Damit diese Formen leichter und direkt von Ihnen erkannt werden können, markiere ich die Eingaben in den Python-Kommandozeileninterpreter besonders.

1.5.1 Python-Kommandozeileninterpreter

Wie bereits bekannt, können Sie die Python-Anweisungen einfach in den Python-Kommandozeileninterpreter eintippen. Das ist normalerweise dadurch angedeutet, dass die einzugebenden Zeilen mit `>>>` markiert sind:

```
>>> print("Hello Python-REPL")
Hello Python-REPL
>>>
>>> def multiply(a, b):
...     return a * b
...
>>> multiply(7, 2)
14
```

Auch spezielle Python-Funktionalitäten lassen sich mithilfe von `import` einbinden, etwa das Modul `random` für Zufallswerte (Details folgen in Kapitel 6):

```
>>> import random
>>>
>>> random.randrange(1, 49)
42
>>> random.choice(["Sekt", "Selters", "Apfelschorle"])
'Selters'
```

Dieses Vorgehen praktiziere ich bis maximal ca. 20 Zeilen. Danach empfiehlt es sich, zum Editieren einen Texteditor oder besser eine IDE zu verwenden.

Hinweis

Um konzeptionell etwas zu verdeutlichen, verzichte ich mitunter auf die exakte Darstellung im Python-Kommandozeileninterpreter, da die Anweisungen durch den Mix von Protokollierung mit `>>>` und den Zwischenergebnissen unübersichtlich wird. Dann nutze ich eine verkürzte Darstellung – die Anweisungen sind dabei einzeln in den Python-Kommandozeileninterpreter einzugegeben oder aber in einen Texteditor oder eine IDE.

```
import random

print(random.randrange(1, 49))

print(random.choice(["Sekt", "Selters", "Apfelschorle"]))
```

1.5.2 Texteditor und direkte Ausführung

Werden die Programme länger, so gestaltet sich das Editieren mit dem Python-Kommandozeileninterpreter immer mühsamer. Hier kann ein Texteditor Abhilfe schaffen. Programme sind in dem Fall generell ohne >>> in etwa wie folgt abgebildet:

```
import random


def simplified_lotto_numbers():
    result = []
    for i in range(6):
        result += [random.randrange(1, 49)]
    print(result)


simplified_lotto_numbers()
```

Diese Zeilen müssen dann exakt so (inklusive Einrückungen, die penibel beachtet werden müssen) abgetippt und als Python-Datei mit beliebigem Namen, aber inklusive der Endung `.py`, also hier etwa als `random_lotto_example.py`, abgespeichert werden.

Wird das Programm noch größer, so hilft es, eine `main()`-Funktion einzuführen und die zuvor definierten Funktionalitäten dort aufzurufen – hier ist das nur exemplarisch gezeigt. Das Einführen einer `main()`-Funktion ist vor allem auch dann sinnvoll, wenn auf die in dem Python-Modul bereitgestellten Funktionalitäten von anderen Modulen zugegriffen werden soll.

```
def main():
    simplified_lotto_numbers()


if __name__ == "__main__":
    main()
```

Ausführung

Die Anweisungen lassen sich durch einen Aufruf von `python3 <dateiname>` ausführen. Betrachten wir das exemplarisch für das obige einfache Beispielprogramm:

```
$ python3 random\_lotto\_example..py
```

Dann erhalten wir etwa diese Ausgabe:

```
[34, 18, 47, 32, 23, 3]
```

1.5.3 IDE

Wie bereits angedeutet, ist eine IDE die empfehlenswerte Variante, den Sourcecode zu pflegen, insbesondere, wenn dieser umfangreicher wird.

Wenn Sie mit einer IDE arbeiten, dann können Sie wie im vorherigen Unterkapitel beschrieben ein Projekt sowie einzelne Python-Dateien anlegen. Programme, die ohne >>> abgebildet sind, sollten Sie bevorzugt in einer IDE eingeben und nur selten auf die zuvor gezeigte Variante mit Texteditor zurückgreifen, oder gar auf den Python-Kommandozeileninterpreter.

1.5.4 Abschließender Hinweis

Obwohl die IDE am meisten Komfort bietet, möchte ich noch einmal auf Folgendes hinweisen: In diesem Buch können Sie leicht zwischen zeilenweiser, interaktiver

```
>>> print("interactive style")
interactive style
```

und skriptbasierter Ausführung mit in Dateien gespeicherten Anweisungen

```
print("script based style")
```

durch die Art und Weise der Darstellung, eben mit >>> und ohne diese Zeichen, unterscheiden.

Die Anweisungen aus Listings zur skriptbasierten Ausführung können Sie ebenso gut im Editorfenster Ihrer IDE eingeben und dort ausführen, wie wir es bereits kennengelernt haben.

Inhaltsverzeichnis