EUROPA-FACHBUCHREIHE
Programmierung für die
IT-Ausbildung

Python für IT-Berufe

```
print("Basisband")

Kapitel = input()

if (Kapitel  == "Teil 1"):

    print("Strukturierte Programmierung mit Python")

elif (Kapitel  == "Teil 2"):

    print("Objektorientierte Programmierung mit Python")

elif (Kapitel  == "Teil 3"):

    print("Fortgeschrittene Programmierung mit Python")

elif (Kapitel  == "Teil 4"):

    print("Aufgabenpool")

elif (Kapitel  == "Teil 5"):

    print("Lernsituationen")
```

VERLAG EUROPA-LEHRMITTEL · Nourney, Vollmer GmbH & Co. KG
Düsselberger Str. 23 · 42781 Haan-Gruiten

Europa-Nr.: 31781

Verfasser:
Dirk Hardy, 46049 Oberhausen

1. Auflage 2022

Druck 5 4 3 2 1

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Korrektur von Druckfehlern identisch sind.

ISBN 978-3-7585-3178-1

www.europa-lehrmittel.de
Satz: Reemers Publishing Services GmbH, 47799 Krefeld
Umschlag: braunwerbeagentur, 42477 Radevormwald
Umschlagbilder: Gina Sanders-stock.adobe.com, refresh(PIX)-stock.adobe.com, newb1-stock.adobe.com
Druck: Plump Druck & Medien GmbH, 53619 Rheinbreitbach

Vorbemerkung

Die Programmiersprache Python ist eine relativ junge Sprache. Sie wurde Anfang der 1990er Jahre von einem niederländischen Informatiker (Guido van Rossum) entwickelt. Das primäre Ziel war die Entwicklung einer einfachen und flexiblen Programmiersprache, die schnell zu erlernen und in verschiedenen Bereichen einsetzbar ist. Python kann als Skriptsprache genutzt werden, aber auch (wie Java oder C#) in einen plattformübergreifenden Bytecode übersetzt werden, der dann auf verschiedenen Plattformen ausgeführt werden kann. Die Beliebtheit von Python ist in den letzten Jahren enorm gestiegen und die Sprache erreichte im Jahr 2021 sogar einen Platz unter den ersten drei Programmiersprachen mit der größten Popularität. Das verdankt Python nicht nur seiner Einfachheit, sondern auch den vielfältigen Möglichkeiten der Anwendung (als Skriptsprache, als Web-Anwendung, als Desktop-Anwendung, in der Netzwerkprogrammierung oder auch in mathematisch-wissenschaftlichen Anwendungen).

Aufbau des Buches

Der vorliegende Basisband möchte die Sprache Python möglichst anschaulich, praxis- und unterrichtsnah vermitteln. Damit verfolgt dieses Buch einen **praktischen Ansatz**. Es ist die Ansicht des Autors, dass gerade in der schulischen Ausbildung der Zugang zu den komplexen Themen der Programmierung verstärkt durch anschauliche und praktische Umsetzung vorbereitet werden muss. Anschließend können allgemeine und komplexe Aspekte der Programmierung oder auch der Softwareentwicklung besser verstanden und umgesetzt werden.

Das Buch ist in **fünf Teile** getrennt. Die ersten drei Teile des Buches dienen als **Informationsteil** und bieten eine systematische Einführung in die **strukturierte Programmierung als auch in die objektorientierte Programmierung mit Python**. Abgerundet wird diese Einführung mit einem **Einstieg in fortgeschrittene Themen der Python-Programmierung**.

Der **vierte Teil** des Buches ist eine **Sammlung von Übungsaufgaben**. Nach der Erarbeitung der entsprechenden Kenntnisse aus dem Informationsteil können die Aufgaben aus diesem Teil zur weiteren Auseinandersetzung mit den Themen dienen und durch verschiedene Schwierigkeitsgrade auch die Differenzierung im Unterricht ermöglichen.

Der **fünfte Teil** des Buches beinhaltet **Lernsituationen** basierend auf dem Rahmenlehrplan für alle IT-Berufe. Lernsituationen konkretisieren sich aus den Lernfeldern und sollen im Idealfall vollständige Handlungen darstellen (Planen, Durchführen, Kontrollieren). Aus diesem Grund werden die Lernsituationen so angelegt, dass neben einer Planungsphase nicht nur die Durchführung (Implementation des Programms) im Blickpunkt steht, sondern auch geeignete Testverfahren zur Kontrolle des Programms bzw. des Entwicklungsprozesses in die Betrachtung einbezogen werden. Die Lernsituationen können aber auch als **Projektideen** verstanden werden.

Das Buch ist als Grundlagenbuch für alle berufsbezogenen Ausbildungsgänge im IT-Bereich konzipiert. Durch die differenzierten Aufgabenstellungen kann es in allen IT-Berufen, aber auch von den informationstechnischen Assistenten/innen genutzt werden. **In weiteren Vertiefungsbänden werden dann spezifische Inhalte der einzelnen IT-Berufe berücksichtigt.**

Als Entwicklungswerkzeug wird in diesem Buch die ***Community-*** *Edition Visual Studio 2019* von Microsoft genutzt. Diese Entwicklungsumgebung ist kostenfrei als Download im Internet verfügbar.

Dirk Hardy Frühjahr 2022

1

Teil
Strukturierte Programmierung mit Python

1 Wissenswertes zu Python

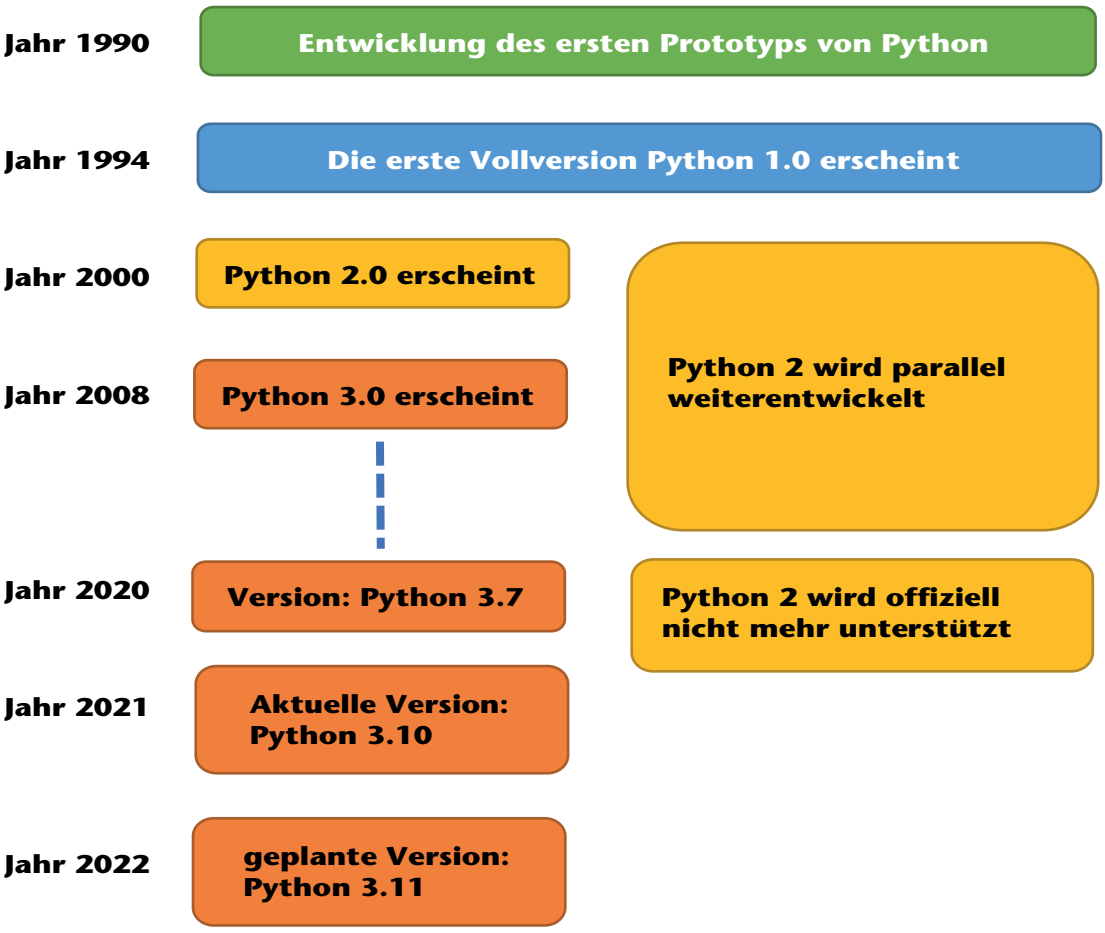
1.1 Entstehung der Sprache Python

Anfang den 90-er Jahren entwickelte der niederländische Informatiker Guido van Rossum die Idee, eine Programmiersprache zu entwerfen, die einerseits die Programmierung selbst deutlich vereinfacht, und andererseits flexibel in vielen Gebieten eingesetzt werden kann – dabei sollte auch eine umfassende Standardbibliothek helfen.

Diese Programmiersprache nannte er ***Python***, wobei der Name nicht an die bekannte Schlangenfamilie angelehnt ist, sondern an die britische Comedygruppe „Monty Pythons“, die van Rossum damals sehr schätzte.

Van Rossum arbeitete vor seiner Entwicklung von Python mit der Programmiersprache ABC, die eher für die Lehre des Programmierens entwickelt wurde. ABC sollte solche Sprachen wie BASIC ablösen, die weit verbreitet waren, aber zu einem sehr unstrukturierten Programmierstil führten. ABC war aber nicht für den professionellen Einsatz konzipiert und van Rossum wollte mit Python eine Sprache entwickeln, die die Vorteile von ABC mit professionellen Einsatzmöglichkeiten verbinden sollte. Für die professionelle Seite von Python nahm van Rossum die Programmiersprache C zum Vorbild und viele Schlüsselworte (Befehle) in Python sind auch identisch mit den entsprechenden Schlüsselworten in C. Python arbeitet sogar direkt mit Modulen der Programmiersprache C und inzwischen auch C++ zusammen.

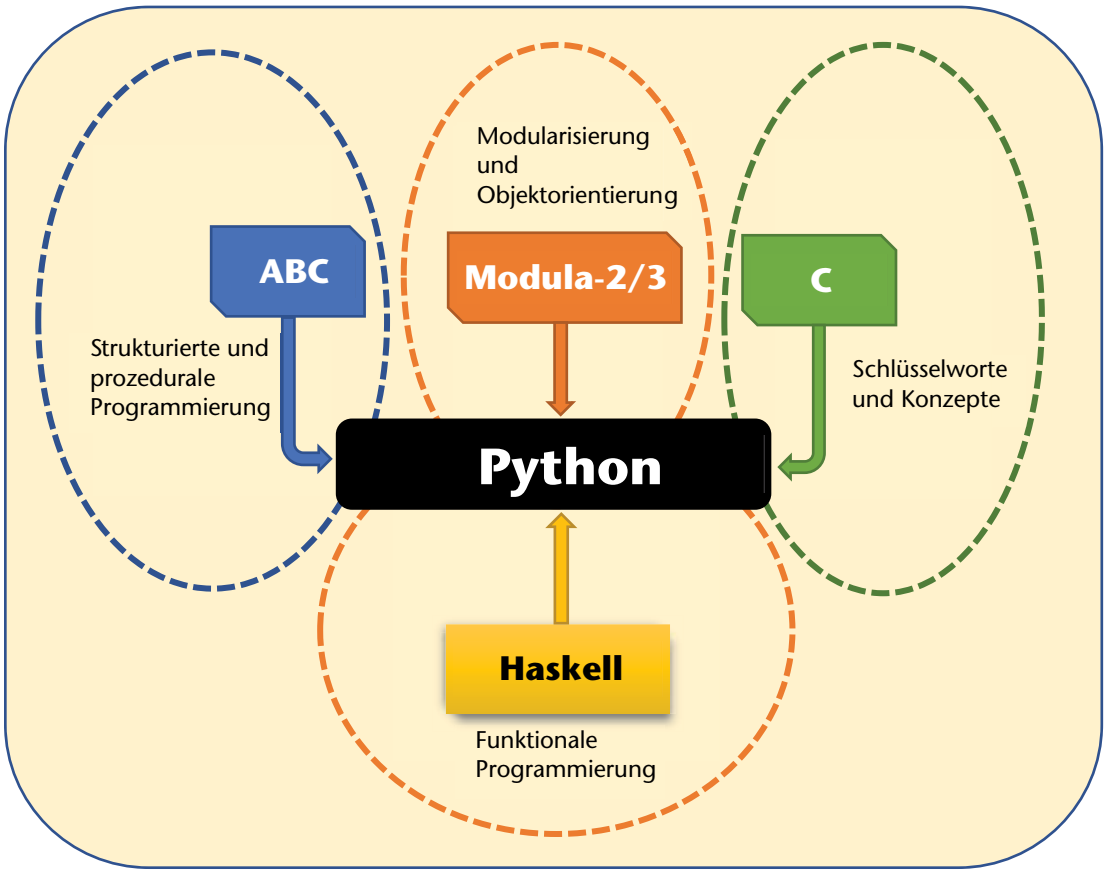
Die folgende Grafik zeigt den zeitlichen Verlauf der Python-Entwicklung:



1.2 Einordnung der Sprache Python

Python ist eine Programmiersprache, die verschiedene Programmier-Paradigmen[1] unterstützt. Neben der strukturierten und prozeduralen Programmierung wird auch die objektorientierte Programmierung unterstützt. Darüber hinaus kann mit Python auch funktional programmiert werden.

Die Eigenschaften hat Python von verschiedenen Vorbildern „geerbt". Dazu zählen die Programmiersprache **ABC** (strukturierte und prozedurale Programmierung), **Modula-2** und **Modula-3** (Modularisierung und Objektorientierung) sowie die Programmiersprache **C** als Vorbild für Schlüsselworte, Operatoren und Basis-Implementierungskonzepte wie die Socket-Programmierung. Das Konzept der funktionalen Programmierung (basierend auf dem Lambda-Kalkül) übernahm Python von der Programmiersprache **Haskell**.



1.3 Strukturierte, prozedurale, objektorientierte und funktionale Programmierung

In der Einordnung der Sprache Python wurde darauf hingewiesen, welche Programmier-Paradigmen Python umsetzt. Zum besseren Verständnis werden diese Paradigmen anhand von Beispielen kurz erläutert:

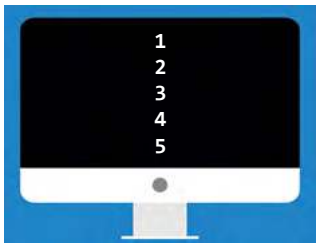
Strukturierte Programmierung

Die strukturierte Programmierung zeichnet sich durch Kontrollstrukturen wie die **`Selektion`** (**`IF-ELSE`**) oder die **Iteration** (**`FOR`**, **`WHILE`**) aus. Damit erhält ein Programm eine nachvollziehbare Struktur.

1 Paradigma kommt aus dem Griechischen und heißt so viel wie Muster oder Vorbild.

Beispiel:

```
FÜR Var := 1 BIS 5 MIT SCHRITTWEITE 1
    SCHREIBE AUF BILDSCHIRM Var
```



Das Beispiel zeigt eine Iteration (Wiederholung) in so genanntem Pseudocode[2]. Dieser Code beschreibt den Ablauf des Programmes, ohne allerdings auf eine spezielle Programmiersprache einzugehen. In dem Beispiel wird eine Variable **Var** so lange um 1 erhöht, bis der Wert 5 erreicht ist. Jeder Wert der Variablen wird dann auf dem Bildschirm ausgegeben.

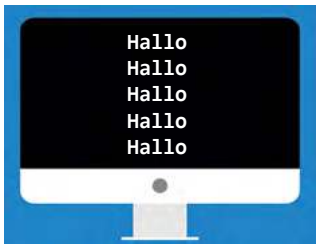
Prozedurale Programmierung

Die prozedurale Programmierung teilt Programme in kleine Einheiten (Prozeduren oder Funktionen), die für bestimmte Aufgaben verantwortlich sind. Sind diese Prozeduren einmal geschrieben und getestet, dann können sie immer wieder benutzt werden – das spart Entwicklungszeit und führt auch zu einer besseren Lesbarkeit des Programms.

Beispiel:

```
PROZEDUR Ausgabe
        SCHREIBE AUF BILDSCHIRM "Hallo"
ENDE

FÜR Var := 1 BIS 5 MIT SCHRITTWEITE 1
        AUFRUF Ausgabe
```



Das Beispiel in Pseudocode zeigt eine Prozedur mit dem Namen **Ausgabe**. Diese Prozedur hat eine Anweisung, die das Wort „Hallo“ auf den Bildschirm schreibt. Die bereits bekannte Wiederholung aus dem Beispiel vorher läuft dann 5-mal und ruft jedes Mal die Prozedur *Ausgabe* auf. Damit steht 5-mal das Wort „Hallo“ auf dem Bildschirm.

Objektorientierte Programmierung

Die objektorientierte Programmierung möchte Objekte der realen Welt in einem Programm abbilden. Damit sollen Problemstellungen aus beliebigen Bereichen (Geschäftsprozesse, wissenschaftliche Untersuchungen usw.) geeigneter als mit den anderen Programmierparadigmen in Programme umgesetzt werden können.

Im Mittelpunkt der objektorientierten Programmierung steht die **Klasse**, aus der dann konkrete Objekte gebildet werden. Diese Objekte haben bestimmte Eigenschaften (Attribute) und so genannte Methoden, mit denen diese Eigenschaften beispielsweise verändert werden können.

2 Pseudocode ist eine Art Sprache, mit der der Ablauf eines Programmes beschrieben wird. Pseudocode zeichnet sich dadurch aus, dass er näher an der natürlichen Sprache als eine Programmiersprache ist. Ein Programm, das in Pseudocode geschrieben ist, kann problemlos in jede Programmiersprache übersetzt werden.

Beispiel:

```
KLASSE Kunde
    Name
    Telefon
ENDE

BILDE OBJEKT K1 VON Kunde
K1.Name    := "Maier"
K1.Telefon := "123456"

SCHREIBE AUF BILDSCHIRM K1.Name und K1.Telefon
```

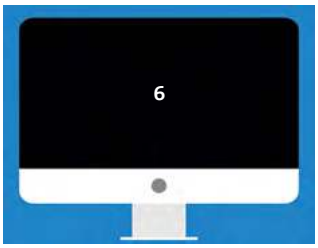
In dem Beispiel wird ein Klasse Kunde definiert. Von dieser Klasse können dann konkrete Objekte wie K1 (für Kunde 1) gebildet werden. Die Eigenschaften des Objektes (Name, Telefon) können dann mit Werten belegt werden. In diesem Beispiel erhält das Objekt K1 den Namen „Maier“ und die Telefonnummer „123456“. Anschließend werden Name und Telefon des Objektes auf den Bildschirm geschrieben.

Funktionale Programmierung

In der funktionalen Programmierung basiert alles auf Funktionen. Berechnungen werden mithilfe von Funktionen definiert, die wiederum Funktionen aufrufen oder deren Rückgabewerte nutzen. Funktionen können auch anonym definiert werden und können damit sehr flexibel eingesetzt werden. Auch Elemente der strukturierten Programmierung wie die Iteration werden durch Funktionen ersetzt – in diesem Fall müssen dann so genannte rekursive Funktionen genutzt werden (das sind Funktionen, die sich selbst aufrufen).

Beispiel:

```
FUNKTION ggT (Übergabewerte x, y)
       FALLS x = y
            RÜCKGABE von: x
       SONST
            FALLS x < y
                RÜCKGABE von: ggT (x, y-x)
            SONST
                RÜCKGABE von: ggT (x-y, y)
ENDE
SCHREIBE AUF BILDSCHIRM ggT (12, 42)
```



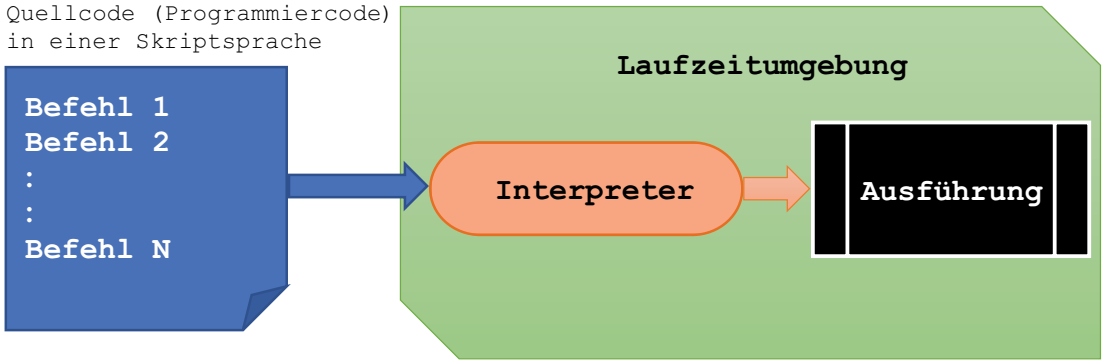
In dem Beispiel wird eine Funktion **ggT** definiert, die den größten gemeinsamen Teiler von zwei Zahlen berechnet. Dabei ruft die Funktion sich selbst auf und übergibt angepasste Zahlen, je nachdem, ob die erste übergebene Zahl größer oder kleiner der zweiten übergebenen Zahl ist. Das entspricht dem euklidischen Algorithmus, der von dem berühmten Mathematiker Euklid stammt (griechischer Mathematiker, 3. Jahrhundert vor Christus). Ein solches Prinzip nennt sich ***Rekursion***.

1.4 Skriptsprache vs. Compilersprache

Vor der Einordnung von Python als Skript- oder Compilersprache sollen diese beiden Möglichkeiten dargestellt werden.

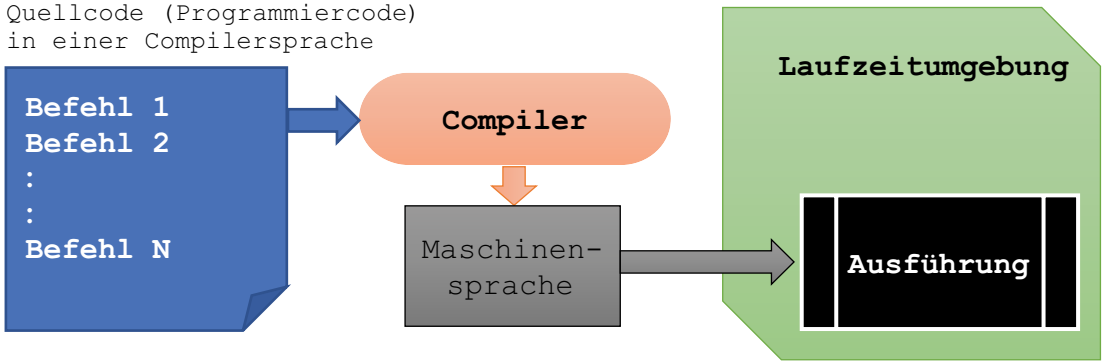
Skriptsprache (Interpretersprache)

Eine Skriptsprache ist definiert als eine Sprache, die von einem Interpreter Zeile für Zeile übersetzt und auf der jeweiligen Plattform ausgeführt wird. Bildlich kann das so dargestellt werden:

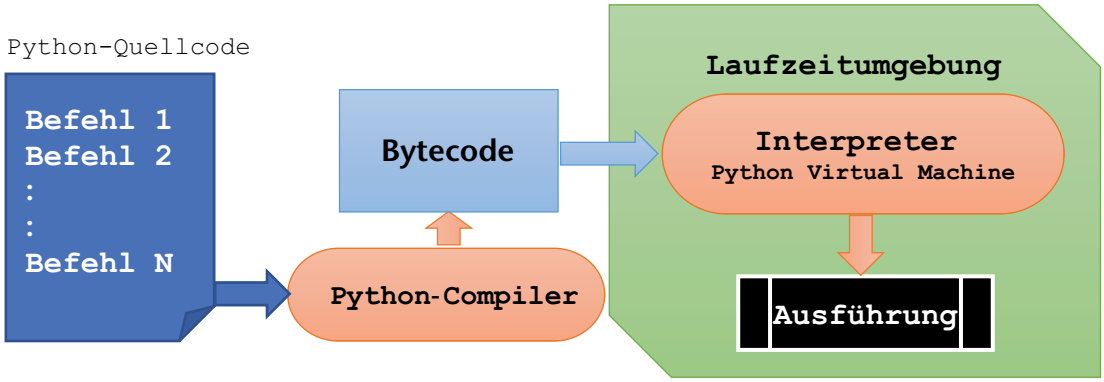


Compilersprache

Bei einer Compilersprache wird der Quellcode zuerst von einem Compiler in Maschinensprache der jeweiligen Plattform übersetzt. Dieses Maschinensprachen-Programm kann dann direkt auf dieser Plattform ausgeführt werden. Bildlich kann das so dargestellt werden:



Python geht nun einen Mittelweg zwischen Skript- und Compilersprache. In manchen Fällen wird Python tatsächlich als reine Skriptsprache eingesetzt (Makroprogrammierung bei LibreOffice oder in der interaktiven Konsole von Python), in den meisten Fällen wird Python aber zu einer Art plattformunabhängigem Zwischencode übersetzt, der dann in der entsprechenden Laufzeitumgebung interpretiert und ausgeführt wird (durch eine ***Python Virtual Machine***). Das Prinzip findet sich auch bei Programmiersprachen wie Java oder C#, die ebenfalls zuerst in einen Zwischencode (Bytecode, Intermediatecode) übersetzt werden.



Der Vorteil bei dieser Vorgehensweise ist ein sehr kompakter und schnell ausführbarer Zwischencode, der die Sprache Python damit plattformunabhängig macht. Auf der jeweiligen Plattform muss nur die ***Python Virtual Machine*** vorhanden sein, um diesen Bytecode zu interpretieren und auszuführen.

1.5 Einsatzgebiete der Sprache Python

Python ist eine Programmiersprache, die durch ihre Konzeption in verschiedenen Bereichen eingesetzt werden kann.

Die Sprache hat eine sehr umfangreiche Standard-Bibliothek, mit der bereits viele Programmieraufgaben gelöst werden können. Weiterhin ist eine Vielzahl freier Zusatz-Module verfügbar. Die folgende Auflistung zeigt die weiteren Einsatzmöglichkeiten:

Einsatzgebiet	Beispiele
Grafische Benutzeroberflächen (GUI)	Mithilfe folgender Bibliotheken können grafische Benutzeroberflächen (GUI) entwickelt werden: PyQt5, Tkinter, Kivy, wxPython, Libavg, PySimpleGUI, PyForms u.a.
Netzwerkprogrammierung	Python unterstützt TCP- und UDP-Kommunikation sowie Web-Client/-Server-Programmierung und CGI-Programme. Ebenso werden E-Mail-Protokolle (SMTP, POP, IMAP) unterstützt.
Datenbankanbindung	Mit Python können eine Vielzahl von Datenbanken angebunden werden (MongoDB, MariaDB, MySQL u.a.). Besonders einfach ist der Zugriff auf die integrierte ***SQLite-Datenbank***.
Professionelle Programmierung	Durch einen vollen Zugriff auf das Betriebssystem und Dateisystem sowie die Möglichkeiten der parallelen Programmierung sind professionelle Anwendungen möglich.
Reguläre Ausdrücke	Python verfügt über leistungsstarke reguläre Ausdrücke, mit denen sehr effiziente Suchen durchgeführt werden können.
Web-Anwendungen	Durch eine Vielzahl von Web-Frameworks ist die professionelle Entwicklung von Web-Anwendungen möglich – speziell durch das ***Django-Framework***.
Dateiverarbeitung	Umfangreiche Funktionalitäten zur Dateiverarbeitung inkl. der gängigen Formate wie XML, CSV oder JSON.
Einbindung von externen Modulen	Python bietet einige Schnittstellen, um beispielsweise Module aus der Programmiersprache C/C++ einzubinden.
Einsatz als integrierte Skriptsprache	Python kann beispielsweise als Skriptsprache in LibreOffice oder auch GIMP eingesetzt werden, um Makros zu programmieren.
Mathematische und wissenschaftliche Anwendungen	Durch Zusatzmodule wie ***numpy, scipy*** oder auch ***pandas*** kann Python besonders gut für mathematisch-wissenschaftliche Anwendungen eingesetzt werden. Vor allem auch bei statistischen Fragestellungen.

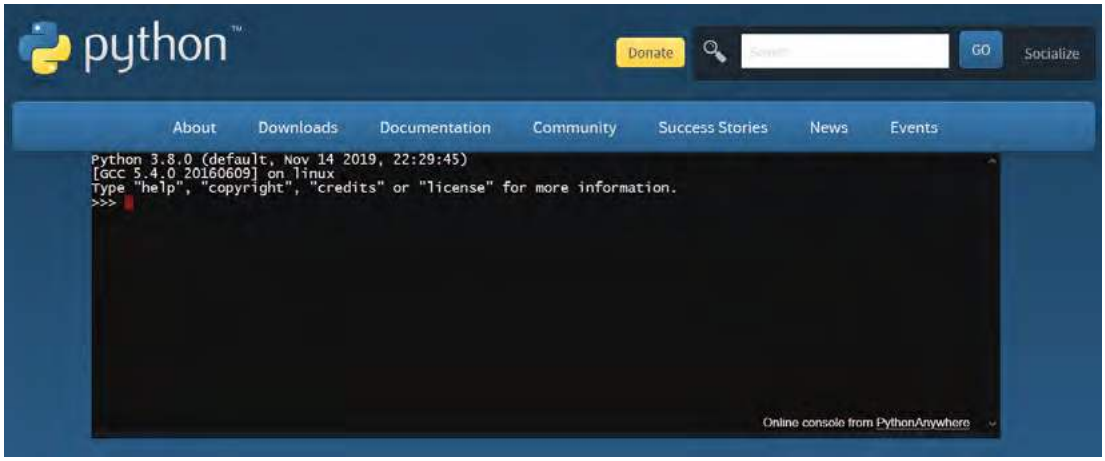
2 Das erste Python-Programm

2.1 Der Python-Interpreter

2.1.1 Einen Python-Interpreter online nutzen

Das Besondere an Python ist die Möglichkeit, sofort mit der Programmierung zu starten. Da Python eine Skriptsprache ist, braucht es keinen grundsätzlichen Aufbau eines Programmes wie in den Sprachen C++ oder Java, sondern die Eingabe eines einzigen Befehls wird sofort interpretiert und ausgeführt. Natürlich können mit Python auch größere und komplexe Programme wie mit den anderen modernen Programmiersprachen geschrieben werden. Dazu wird aber in der Regel eine Entwicklungsumgebung genutzt, die alle Quellcode-Dateien verwaltet und auch die Übersetzung und Ausführung übernimmt. Eine solche Entwicklungsumgebung wird in dem nächsten Unterkapitel vorgestellt. Allerdings soll nun an erster Stelle die einfache Möglichkeit gezeigt werden, wie Python-Befehle (oder auch ein ganzes Python-Programm) mit einem Online-Interpreter direkt ausgeführt werden können. Es gibt eine Vielzahl von solchen Interpretern, hier wird der Interpreter der ***Python.org*** – Webseite vorgestellt. Diese Webseite wird von der Python Software Foundation (PSF) herausgegeben. Diese Organisation besitzt die Markenrechte an der Programmiersprache Python.

Der Aufruf der Seite „https://www.python.org/shell/“ startet einen Online-Interpreter, den die Python Software Foundation zur Verfügung stellt:



Die drei Größer-Zeichen „>>>“ sind der so genannte Eingabeprompt (Eingabeaufforderung) – an dieser Stelle ist der Interpreter bereit für Eingaben, die mit einem „`Return`“ abgeschlossen werden müssen. Ausgaben des Interpreters werden nicht mit diesen drei „>>>“ eingeleitet. Mit den Pfeiltasten können bereits eingegebene Befehle wieder angezeigt und neu verwendet werden.

2.1.2 Erste Eingaben mit Python

Die Möglichkeit dieser Online-Interpreter-Eingabe dient vor allem dem schnellen Testen von einzelnen oder wenigen Python-Anweisungen. Die Eingabe kann aber auch als eine Art Taschenrechner-Ersatz dienen. Die folgenden Beispiele zeigen die Verwendung dieser einfachen Option:

Beispiel 1: Addition der Zahlen 10 und 20

```
Python 3.8.0 (default, Nov 14 2019, 22:29:45)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>> 10 + 20
30
```

Beispiel 2: Verknüpfung (Verkettung) von zwei Zeichenketten

```
Python 3.8.0 (default, Nov 14 2019, 22:29:45)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> "Hallo" + "Python"
'Hallo Python'
```

Beispiel 3: Importieren einer Mathematik-Bibliothek und Berechnung der Quadratwurzel von 9

```
Python 3.8.0 (default, Nov 14 2019, 22:29:45)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> math.sqrt(9)
3.0
```

Diese ersten Beispiele zeigen die Verwendung dieser Eingabemöglichkeit. Wenn das Wissen über die Programmiersprache Python weiter fortgeschritten ist, dann ist der Einsatz dieser Möglichkeit deutlich interessanter.

2.2 Einsatz einer Entwicklungsumgebung

2.2.1 Ein Python-Projekt in Visual Studio anlegen

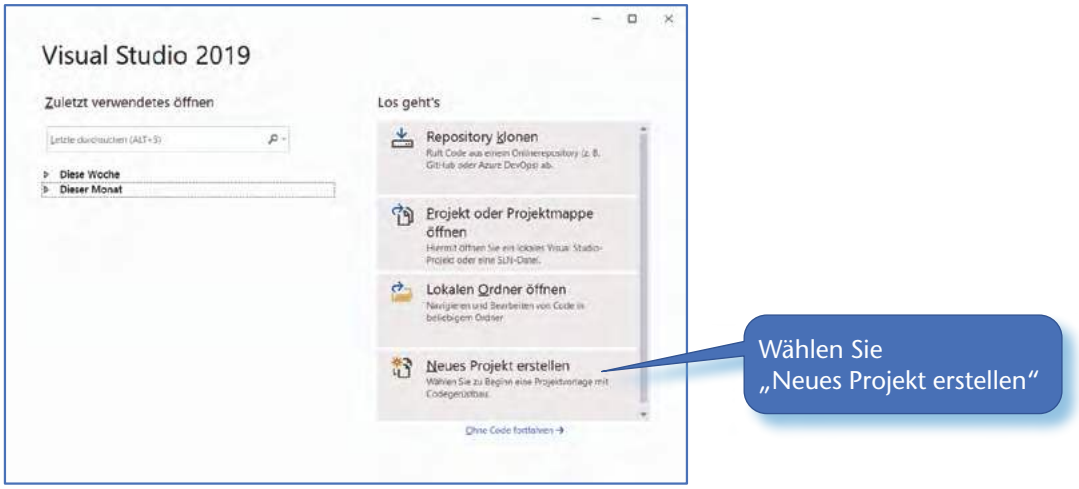
Die integrierte Entwicklungsumgebung Visual Studio ist eine komfortable Umgebung, um Python-Programme zu entwickeln. Besonders erfreulich ist der Umstand, dass die Umgebung kostenfrei als *Community Edition* im Internet bereitsteht. Ein Python-Programm besteht aus einer oder mehreren Quellcodedateien. Diese Dateien werden in einem Projekt organisiert. Visual Studio unterscheidet verschiedene Projektarten:

- Python-Anwendung (Konsolenanwendung)
- Web-Projekt (Grundgerüst für Web-Anwendungen)
- Django Web-Projekt (Web-Anwendungen mit dem Django-Framework)
- IronPython-Anwendungen (Konsolen-, Forms- oder WPF-Anwendungen mit Python)

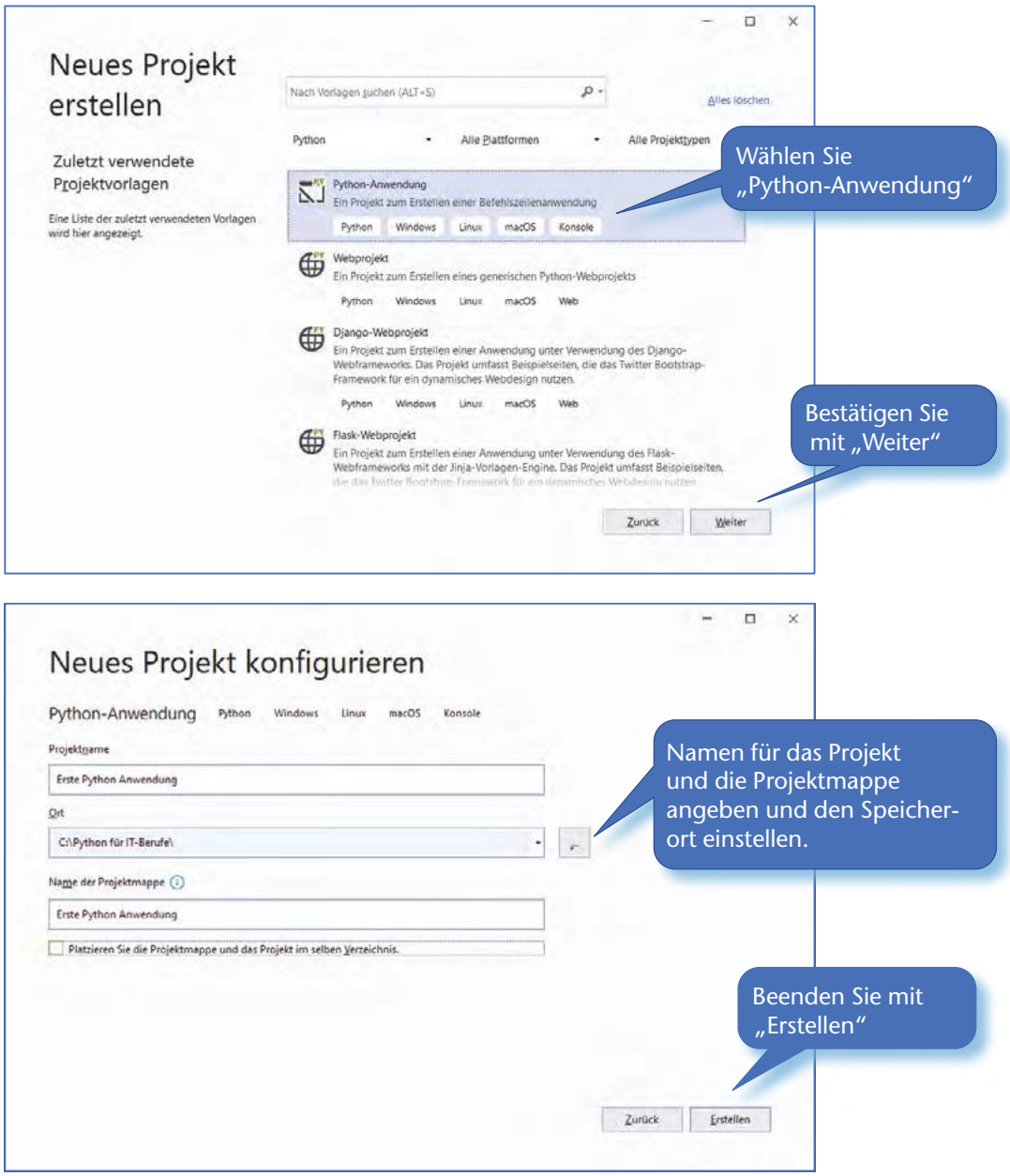
In diesem Buch wird nur eine Projektformen verwendet: die Python-Anwendung (**Konsolenanwendung).** Die Konsolenanwendung ist ausreichend, um eine einfache Ein- und Ausgabemöglichkeit für die Python-Programme zu haben und alle wesentlichen Inhalte des Buches umzusetzen. Eine Konsolenanwendung ist natürlich nicht so ansprechend wie eine grafische Benutzeroberfläche, aber, um die Grundlagen der Sprache zu lernen, völlig ausreichend.

Anlegen eines neuen Projektes:

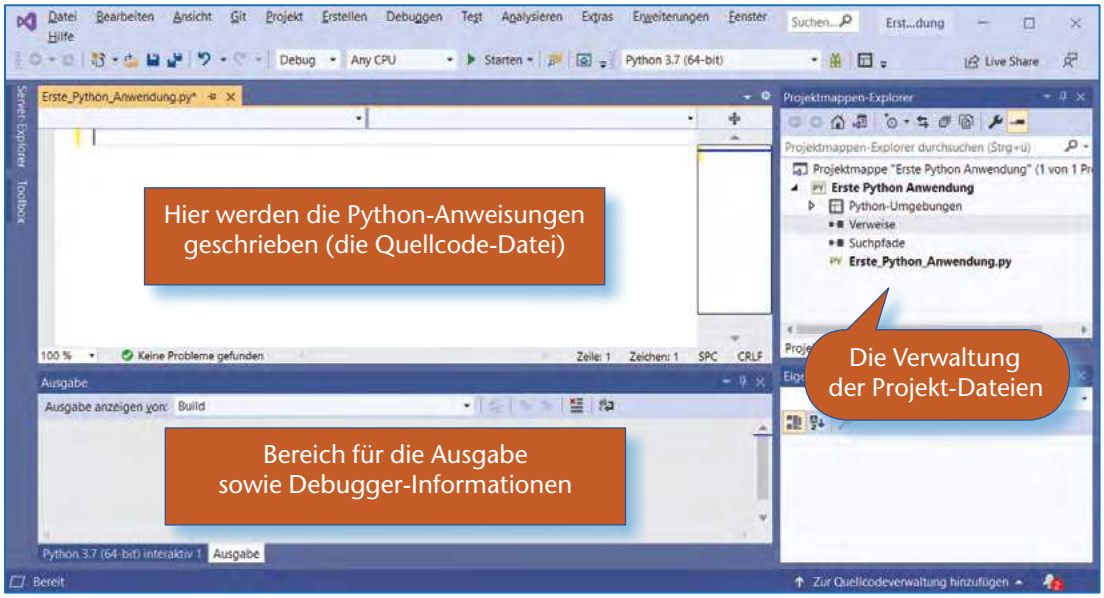
- Starten Sie Visual Studio.



- Anschließend erscheint eine Auswahl der möglichen Projektformen:



- Nach der Bestätigung wird ein neues Projekt angelegt und in der Entwicklungsumgebung angezeigt.



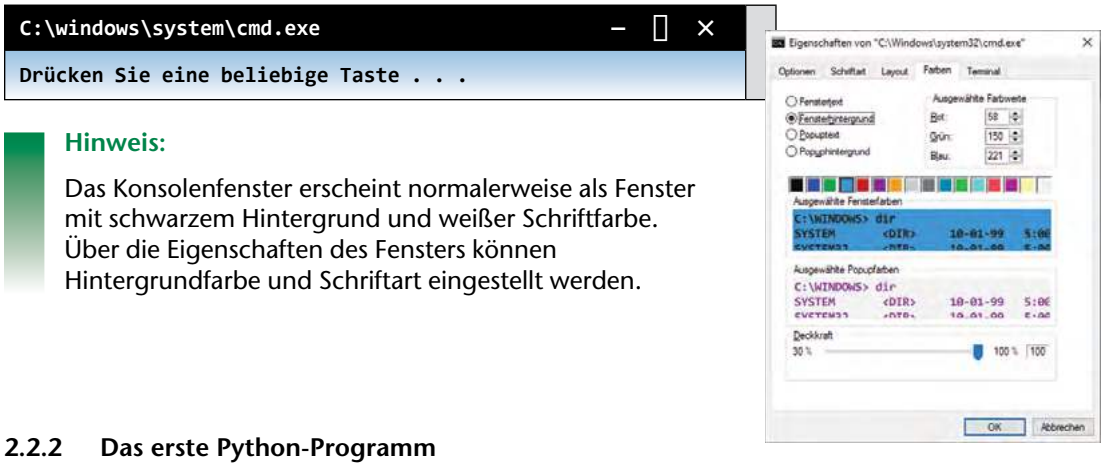
Die Entwicklungsumgebung hat ein Projekt mit dem gewählten Namen (hier „Erste Python Anwendung“) angelegt. Zusätzlich zum Projekt wurde eine Projektmappe mit demselben Namen angelegt. Innerhalb dieser Projektmappe können beliebig viele weitere Projekte angelegt werden. Der Projektmappenname kann auch anders benannt werden (auf die rechte Maustaste über dem Namen klicken und „Umbenennen“ wählen). Innerhalb des Projektes sind Python-Umgebungen, Verweise und die Quellcode-Datei „Erste_Python_Anwendung.py“ angelegt.

Ausführen eines Python-Programms

Um das Programm zu übersetzen und anschließend auszuführen, gibt es verschiedene Möglichkeiten unter Visual Studio:

- Menüpunkt: Debuggen → Starten ohne Debugging
- Tastenkombination: STRG + F5

Fortgeschrittene werden später das Starten mit Debugging (Menüpunkt: Debuggen → Debugging starten oder F5 drücken) verwenden, wenn kompliziertere Programme analysiert werden müssen. Für den Anfang ist jedoch die oben beschriebene Vorgehensweise völlig ausreichend. Nach dem Starten des obigen ersten Programms erscheint dann folgendes Fenster:



Hinweis:

Das Konsolenfenster erscheint normalerweise als Fenster mit schwarzem Hintergrund und weißer Schriftfarbe. Über die Eigenschaften des Fensters können Hintergrundfarbe und Schriftart eingestellt werden.

2.2.2 Das erste Python-Programm

Da Python als Skriptsprache kein Programmgerüst braucht, um eine erste Anweisung auszuführen, ist das erste Python-Programm sehr einfach:

```
print("Das erste Python-Programm!")
```

Nach dem Starten erscheint das folgende Konsolenfenster:

Der `print`-Befehl schreibt (wie der Name schon andeutet) einen Inhalt auf den Bildschirm. Dieser Befehl und ein Befehl zum Einlesen von Werten über die Tastatur werden am Ende des Kapitels noch einmal genau beleuchtet.

Zum Vergleich wird hier das erste Programm in der Programmiersprache C# gegenübergestellt.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Erstes_Programm
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Console.WriteLine("Das erste C#-Programm!");
        }
    }
}
```

Nach dem Starten erscheint dieses Konsolenfenster:

Es ist erkennbar, dass für das erste C#-Programm deutlich mehr Aufwand betrieben werden muss. In anderen Sprachen wie Java oder C++ ist es ähnlich. Das zeigt die Einfachheit einer Sprache wie Python. Am Ende einer Konsolenanwendung wartet das Programm auf das Drücken einer beliebigen Taste. Erst danach wird das Fenster geschlossen.

2.3 Grundlegende Konventionen in Python

2.3.1 Schlüsselworte in Python

Ein Python-Programm ist nichts anderes als eine Reihe von Python-Anweisungen, die hintereinander abgearbeitet werden. Dabei gibt es Möglichkeiten, die Reihenfolge diese Abarbeitung zu steuern (mit Kontrollstrukturen wie der Selektion und Iteration) oder Teile der Anweisungen in Prozeduren oder Funktionen zu definieren sowie im Sinne der objektorientierten Programmierung Klassen anzulegen und Objekte zu erstellen. Diese Themen werden im Laufe des Buches ausführlich beschrieben. Die Grundlage aller dieser Möglichkeiten sind die Schlüsselworte der Programmiersprache Python. Es sind klar definierte Begriffe, die nur für den vorgesehenen Zweck eingesetzt werden dürfen. Python hat eine sehr überschaubare Anzahl dieser Schlüsselworte – deshalb gilt Python auch als einfach zu erlernende Sprache.