Einführung in das Programmieren mit Python

M. Sabath

6. März 2017

Inhaltsverzeichnis

1 Überlegungen

Bevor wir nun in eine Programmiersprache eintauchen, solltest du einige Vorüberlegungen anstellen.

1. Was ist für dich "programmieren"?
2. Wozu braucht man denn das Programmieren?
3. Was macht man mit einer Programmiersprache?
4. Was muss eine Programmiersprache können?
5. Überlege dir konkrete Sachverhalte, die eine Sprache können muss und begründe diese.
6. Überlege dir ein sehr einfaches Beispiel, was du mit der Programmiersprache umsetzten willst.
7. Formuliere dein Programm aus Punkt 6 in Worten, so das eindeutig klar ist, was du möchtest. Schreibe dein "Programmäuf, d.h. tue so, als ob der Computer "Deutsch"versteht und formuliere so, dass er dein Programm ausführt. Vergiss nicht, der Computer ist "dumm", also wähle deine Anweisungen sorgfältig.
8. Suche dir einen Partner nachdem du die Fragen beantwortet hast und tauscht euch 10 Minuten aus.
9. Geht danach mit einer anderen zweier Gruppe zusammen und tauscht auch hier eure Ergebnisse 10 Minuten aus.
10. Bereitet eine beliebige Präsentation eurer Ergebnisse vor.

2 Hinweise

2.1 Bezugsquellen

Python kann von dieser Homepage geladen werden: https://www.python.org/.

Python ist eine vollwertige Programmiersprache, die den Ruf hat, sehr einsteigerfreundlich zu sein. Sie ist für alle gängigen Betriebssysteme erhältlich. Im Netz gibt es eine Vielzahl von Tutorials, Foren und Wissensdatenbanken, die einen guten Anlaufpunkt bei Problemen bieten.

Hier zwei Beispiele: http://py-tutorial-de.readthedocs.io/de/python-3.3/appetite.html
http://python4kids.net/how2think/index.htm

Als Vorlage zu diesem Material diente das Buch von *Jamie Chan: Learn Python in one day and Learn It Well. LearnCoding Fast, Kindle Edition, 2014*

2.2 Python auch ohne Installation

Im Netz gibt es sogar die Möglichkeit Python-Programme ganz ohne Installation laufen zu lassen: https://www.pythonanywhere.com/. Hier sind jedoch keine grafischen Oberflächen möglich, welche für den Anfang wohl eher uninteressant sind.

2.3 Ausführbares Programm erstellen

Um ein Python-Programm laufen zu lassen, muss Python auf dem Zielrechner installiert sein. Da das nicht immer der Fall ist, gibt es Drittanbieter Software, die aus deiner Python-Datei ein lauffähiges Programm erstellen. Z.B. Py2exe (http://www.py2exe.org/) oder Pyinstaller (http://www.pyinstaller.org/).

Teste, ob du aus deinen Programmen eine .exe erstellen kannst.

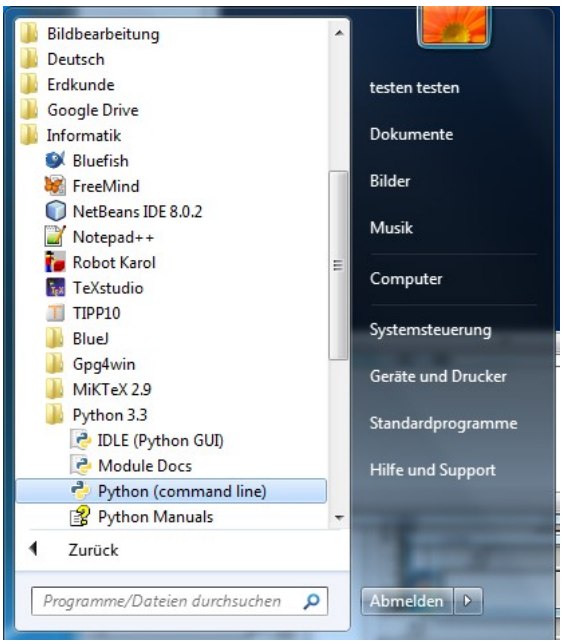
3 Erste Schritte

Nach der Installation verfügt der Rechner über einen sogenannte Python Shell für den **interaktiven Modus** und den Editor **IDLE**. Im interaktiven Modus kannst du Python-Befehle testen. Für eine richtiges Programm musst du deine Befehle in eine Textdatei mit der Endung **.py** schreiben.

3.1 Interaktiver Modus

Es gibt verschieden Möglichkeiten den interaktiven Modus zu starten:

- Tippe in der Shell den Befehl ‘python‘ ein.
- Starte die Python (command line)



Öffne die Python-Shell um in den interaktiven Modus zu gelangen. Tippe dort folgende Zeilen ein:

```
1 12-8
```

```
2+7
18*215
3<4
3>4
print("IGS Kandel")
sdf
```

Gratuliere, du hast soeben programmiert und deine ersten Befehle in Python geschrieben.

Ein Computer bietet sich an mathematische Probleme zu lösen. Finde heraus, wie die Befehle für folgende Aufgaben lauten:

```
1. 3*3*3=
2. 7+14=
3. 3+3*10= # (rechne zuerst im Kopf, was stellst du fest?)
4. 12:4 =
# (rechne ab jetzt zuerst im Kopf.)
5. 100:-10 =
6. 2 hoch 3, 3 hoch 2, 10 hoch 10
7. Wurzel aus 4, 16, 91, 100, -4
8. dritte Wurzel aus 8, 27, 100
```

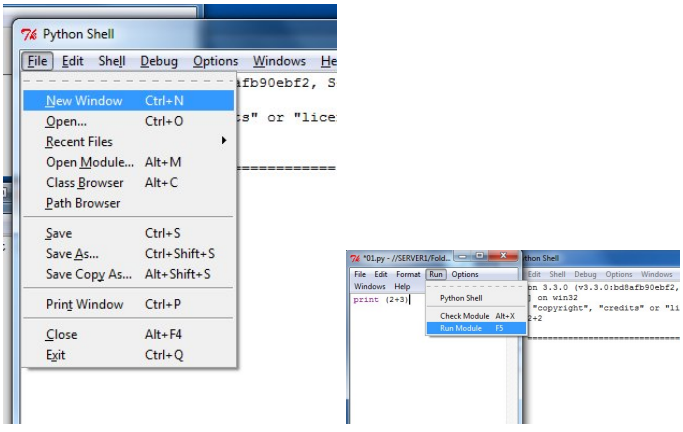
3.2 Der Editor IDLE

Starte IDLE und erstelle darin eine neue Datei *erstesProgramm.py*. **Beachte, dass Python Programmdateien immer die Endung .py haben müssen.**

Füge folgenden Befehl ein:

```
print(2+3)
```

Führe es aus ‘F5‘.



Das war nun dein erstes Python-Programm, gratuliere.

3.3 Die Sache mit den Kommentaren

Gewöhne dir direkt an, alle deine Programme mit reichlich Kommentaren zu versehen. Nimm das nicht auf die leichte Schulter. Es kommt beim Programmieren sehr schnell vor, dass man schon kurze Zeit später nicht mehr nachvollziehen kann, weshalb man denn so manche Programmzeile geschrieben hat.

3.3.1 Einzeilige Kommentare

Einzeilige Kommentare beginnen mit einem Raute Zeichen

```
1 # Das ist ein einzeiliger Kommentar
```

3.3.2 Mehrzeilige Kommentare

```
1 """ Ein  mehrzeiliger Kommentar wird
2 von drei einfachen oder drei doppelten Anfuehrungszeichen
3 eingefasst."""
```

3.3.3 Das fertige Programm

So könnte dein auskommentiertes Programm aussehen.

```
1 '''Der Print Befehl gibt hier die Worte
2 IGS Kandel auf dem Bildschirm aus.'''
3 print("IGS Kandel")
```

4 Variablen

Eine Variable kannst du dir als Schachtel vorstellen, in der du alle möglichen Dinge speichern kannst, um später im Verlauf deines Programms damit arbeiten zu können.

Eine Variable in einem Programm einzuführen, ist einfach. Alles, was du tun musst, ist deine "Schachtel" mit einem Namen zu versehen und mit einem Inhalt füllen.

```
# hier fuehre ich eine Variable ein und belege sie mit der Zahl 100
meineVariable = 100
```

```
# weitere Beispiele zum Einfuehren von Variablen

variable1 = 70
variable2 = "Knoedel"
variable3 = "Schranktuer"

# mehrere in einer Zeile geht auch
variable4, variable5 = 30, "Federtier"

'''das ist identisch zu:
variable4= 30
variable5 = "Federtier"
'''
```

Fachbegriffe:
Wird eine Variable mit Namen eingeführt, nennt man das **deklarieren**. Wird sie mit einem Inhalt/Wert gefüllt, nennt man das **initialisieren**. Bei Python passiert beides gleichzeitig.

Aufgabe 1: (Lösung S. 21)
Erstelle eine Datei variablenNamen.py und tippe die Beispiele von oben ab.

Wenn du die Datei ausführst passiert nichts. Woran liegt das?

Speichere die Datei unter dem Namen **variablenNamen-ausgeben.py** und lass dir die Inhalte folgender Variablen von den Beispielen weiter oben ausgeben:

- variable1
- variable2 und gleichzeitig variable3
  siehe dazu Kapitel 4.3.1 Mehrere Parameter auf S.11
- variable4 (addiere aber direkt bei der Ausgabe +5)

Wie du siehst ist der ‘print()‘ Befehl sehr flexibel.

4.1 Namen von Variablen

Bei der Namensvergabe von Variablen sind ein paar Dinge zu beachten.

- Es sind nur Buchstaben, Zahlen und der Unterstrich (_) erlaubt.
- Ein Name darf nicht mit einer Zahl beginnen.
- Namen unterscheiden zwischen Groß- und Kleinschreibung.
- Sogenannte reservierte Wörter dürfen nicht verwendet werden. Das sind Wörter, die in Python schon verwendet werden. Z.B. ’print, input, if, ... ’

4.2 Variablen mit einem Wert belegen

Weiter oben hast du Variablen schon mehrfach mit einem Wert belegt. Es ist bei Programmiersprachen üblich, dass einer Variablen über das = (Gleichheitszeichen) ein Wert zugewiesen wird. Das Gleichheitszeichen hat somit hier eine andere Bedeutung, als in der Mathematik. Dort bedeutet es, dass links und rechts das “Gleiche“ seht. In Python bedeutet es, dass der Variablen der Wert rechts vom Gleichheitszeichen zugewiesen wird.

Variablen können sowohl Text, Zahlen, Liste, ... enthalten. Du musst Python nicht mitteilen, von welchem Datentyp eine Variable ist. Programmiersprachen, die sich so verhalten, nutzen eine **schwache Typisierung** (loose typing). Das ist nicht bei allen Programmiersprachen so. Bei Java musst du dem Programm mitteilen von welchem Datentyp deine Variable ist. Dieses Prinzip nennt man dann **starke Typisierung** (strong typing).

Beachte: Bei Dezimalzahlen musst du mit einem Punkt arbeiten. Das kommt aus dem Englischen. Schreibe kein Komma!
3 : 2 = 1.5 und nicht ~~1,5~~

Unter Python ist ein beliebiger Wechsel des Datentyps einer Variablen zulässig:

```
1 # die Variable enthaelt eine Zahl
```

```
2  meineSchachtel = 70
3
4  # die Variable enthaelt eine Dezimalzahl
5  meineSchachtel = 70.789
6
7  # die Variable enthaelt nun Text
8  meineSchachtel = "Hallo"
9
10 # die Variable enthaelt nun eine Liste
11 meineSchachtel = [70, 25, "Hallo", 23.45, "alles klar"]
```

Streng typisierte Programmiersprachen geben hier eine Fehlermeldung aus.

4.3 Der Print() Befehl

Wie du festgestellt hast, brauchst du im Moment noch den *print()* Befehl, um dir Informationen von deinen Programmen anzeigen zu lassen. Deshalb an dieser Stelle einige nützliche Informationen zum *print()* Befehl.

4.3.1 Mehrere Parameter

Als Parameter bezeichnet man in der Informatik Übergabewerte. Damit sind hier die Werte gemeint, die dem *print()* Befehl in der Klammer "übergeben" sprich mitgeteilt werden. In Python ist es möglich dem *print()* Befehl beliebig viel, durch Komma getrennte Parameter zu übergeben.

Aufgabe 2: (Lösung S. 21)
Erstelle einen Datei **parameter-print.py**. Führe dort folgende Variablen mit den entsprechenden Werten ein:

- alter ← 16
- lieblingsTier ← Hund
- name ← Fritz

Lass dir nun folgenden Satz mit **einem** *print()* Befehl ausgeben, indem du die drei Variablen nutzt:
Fritz hat im Alter von 16 einen Hund geschenkt bekommen.
Wie verhält es sich mit den Leerzeichen zwischen den Wörtern?

```
alter=16
lieblingsTier= "Hund"
name="Fritz"

ptint(name, "hat im Alter von", 16, "einen", lieblingsTier, ↩
    "geschenkt bekommen.")
```

4.4 Mathematische Operationen

Wie schon erwähnt, sind Computer super geeignet um mit Zahlen umzugehen. In Python sind einige mathematischen Rechenarten direkt nutzbar. Hier wird dir gezeigt, wie du diese einsetzen kannst.

Beispiele: Es sei x=7 und y=2

- Addition:
  x+y = 9
- Subtraktion:
  x-y = 6
- Multiplikation:
  x*y = 14
- Division:
  x/y = 3.5
- Abrunden beim Dividieren:
  x//y = 3
- Modulo (Restklasse / welcher Rest bleibt?):
  x%y = 1
- Potenz:
  x**y = 49

4.5 Moglichkeiten der Wertzuweisung bei Variablen

Du hast gelernt, dass du Variablen mit dem Gleichheitszeichen einen Wert zuweisen kannst. x=7, x="Hund", y=14, ... Wird der Variablen eine Zahl als Wert zugewiesen, gibt es noch eine Möglichkeit der Vereinfachung.

Beispiele:
Ausgangslage ist *x=10*

- x = x + 2 (x hat nun den Wert 12) einfacher: x+= 2
- x = x - 2 (x hat nun den Wert 8) einfacher: x-= 2
- x = x * 2 (x hat nun den Wert 20) einfacher: x*= 2
- x = x / 2 (x hat nun den Wert 5) einfacher: x/= 2
- x = x //3 (x hat nun den Wert 3) einfacher: x//= 3
- x = x%3 (x hat nun den Wert 1) einfacher: x%= 3
- x = x**3 (x hat nun den Wert 1000) einfacher: x**= 3

Aufgabe 3: (Lösung S. 21)
Erstelle einen Datei **wertzuweisung.py**. Führe dort folgende Wertzuweisungen durch. Überlege welchen Wert x hat, bevor du das Programm ausführst.

- x = 16
- x = 11+3
- x = 11/2
- x = 11//2
- x = 11%2
- x = 3 danach x% =7
- x = 3 danach x// = 3
- x = 3 danach x** = 3

4.6 Arbeiten mit Strings

Zeichenketten, auch Strings genannt, sind nichts anderes als Wörter oder einen Aneinanderreihung von Buchstaben und Zahlen, sofern diese nicht als Zahlen, sondern als Zeichenkette verwendet werden sollen. Das Arbeiten mit Strings ist in vielen Programmiersprachen vorgesehen, so auch in Python.

4.6.1 Wertzuweisung

Strings können einer Variablen mit einfachen oder doppelten Anführungszeichen zugewiesen werden. Somit sind folgende Ausdrücke identisch:
meineZeichenkette="Hund"
meineZeichenkette='Hund'

4.6.2 Konkatenation

Zeichenketten können über den + Operator miteinander verkettet werden. Man nennt diesen Vorgang auch konkatenieren. Aus *"Haus"+"tier"* wird *"Haustier"*.

4.6.3 Formatieren von Strings (Sternchen-Abschnitt)

Mithilfe des % Operators können Zeichenketten sehr flexibel eingesetzt werden. Der % Operator innerhalb eines Strings kann als Platzhalter gesehen werden. Beim Abarbeiten des Programms wird der % Operator entsprechend der in ( ) angegebenen Vorgaben durch die Werte ersetzt.

- %s: das s steht für einen String.
- %d: das d steht für eine Ganzzahl.
- %5d: das d steht für eine Ganzzahl mit der entsprechenden Anzahl (hier 5) an Stellen.
- %f: das f steht für eine Dezimalzahl.
- %5.3 f: das f steht für eine Dezimalzahl. Hier mit eine Gesamtlänge von 5 Stellen inklusive 3 Nachkommastellen und dem Punkt.

Aufgabe 4: (Lösung S. 22)
Erstelle eine Datei **strings.py** und füge folgende Zeilen hinzu und lass dir jede Nachricht direkt anzeigen.

```
1 nachricht1 ="Nachricht 1: Heute haben wir %s viel gelernt. Gestern ↩
    habe ich %d geuebt und dabei %f Liter Wasser getrunken"% ↩
    ("schon",5,1.5)
2
3 nachricht2 = "Nachricht 2: Heute haben wir %s viel gelernt. Gestern ↩
    habe ich %d geuebt und dabei %3.2f Liter Wasser ↩
    getrunken"%("schon",5,1.5)
4
5 nachricht3 = "Nachricht 3: Heute haben wir %s viel gelernt. Gestern ↩
    habe ich %10d geuebt und dabei %3.2f Liter Wasser getrunken"% ↩
    ("schon",5,1.5)
6
7 nachricht4 = "Nachricht 4: Heute haben wir %s viel gelernt. Gestern ↩
    habe ich %10d geuebt und dabei %10.2f Liter Wasser getrunken"% ( ↩
    "schon",5,1.5)
```

Wie du feststellst, bist du beim Arbeiten mit %s, %d oder %f sehr unflexibel, da du stets auf die Einhaltung der Reihenfolge achten musst. Python hat auch hier eine passende Antwort zur Hand. Die Methode *format()*. Hier hast du Einfluss, an welcher Stelle im Text du welchen Parameter einsetzen möchtest. Bei dieser Methode arbeitest du nicht mit %s, %d oder %f, sondern mit geschweiften Klammern {*Position des Parametes:Formatierung* } in denen du angibst, wo as stehen soll. Beachte, die Positionen werden beginnend mit der 0 gezählt.

Beispiel:
Heute haben wir {*2:s*} viel gelernt. Gestern habe ich {*0:5d*} geuebt und dabei {*1:5.3f*} Liter Wasser getrunken".format(5,1.5,“schon")

Füge das Beispiel der Datei *strings.py* hinzu. Erklärung: Da die Zählweise bei 0 beginnt, steht die 5 an erster Stelle und wird bei {*0:5d*} eingesetzt. Die 1.5 steht an zweiter Stelle und wird somit bei {*1:5.3f*} entsprechend der Formatierung eingesetzt. Zugunterletzt fehlt noch der dritte Parameter, der an der Stelle {*2:s*} eingesetzt wird. Lässt du die Positionsangaben weg, verhält sich die *format*-Methode wie die einfache Formatierung und setzt die Parameter entsprechend ihrer Reihenfolge ein.

Aufgabe 5: (Lösung S. 23)
Erstelle eine Datei **strings2.py** und löse folgende Ausgaben:

Setze folgende Parameter an der richtigen Stelle in den Text entsprechend der gewünschten Formatierung ein. Ein o steht für ein Leerzeichen.Die Reihenfolge der Parameter darfst du nicht verändern.

1.
Parameter: (2,2.5, "Division")
Text: Bei der XXXXXXX von 5 : oX lautet das Ergebnis oX.XX

2.
Parameter: ("fahrtuechtig",100.0,"Wert","Blut",1)
Text: ooX Promille im XXX ist ein sehr hoher XXX. Zu oXXX.XX Prozent ist man nicht mehr XXXXXX. Also nie mit X.X Promille unterwegs sein!

4.6.4 Sonstiges

Python bietet noch viele weitere String Funktionen, wie z.B. die Funktion *upper()*. So wird aus *"Haustier".upper()* → *"HAUSTIER"*

4.7 Datentypen

4.7.1 int, float, string

Die einfachen Datentypen hast du schon kennengelernt.

- int: Ganzzahlen
- float: Dezimalzahlen
- String: Zeichenketten

Es gibt noch weitere Datentypen in Python, die an entsprechender Stelle eingeführt werden. Eine Auflistung von den in Python vorhandenen Datentypen findest du hier:
https://www.hdm-stuttgart.de/~maucher/Python/html/Datentypen.html
https://de.wikibooks.org/wiki/Python_unter_Linux:_Datentypen

4.7.2 Datentyp Boolean

Der Computer arbeitet intern mit dem Binärsystem. D.h. dort gibt es nur zwei Ziffern, Zustände. Identisch zu deinem Alltag, auch dort findest du oft Dinge, die sich mit zwei Zuständen beschreiben lassen: Strom an/aus, Fußgängerampel: grün/rot, das Auto fährt/steht, er hat Geburtstag: ja/nein. Um diese Ereignisse in Python verarbeiten zu können gibt des den Datentyp *Boolean* mit den beiden Wahrheitswerten True, False.

4.7.3 Datentypen vergleichen

Datentypen können teilweise miteinander verglichen werden. Die nutzbaren Vergleichsoperatoren sind:

- == gleich
- != ungleich
- < kleiner
- > größer
- <= kleiner gleich
- >= größer gleich

Python gibt dann abhängig vom Ergebnis des Vergleichs *True oder False* zurück. Vorsicht: Vergleichst du Strings, musst du beachten, dass Python Großbuchstaben immer vor Kleinbuchstaben einsortierst. Um das zu verhindern, musst du eine String erst komplett in Klein- bzw. Großbuchstaben umwandeln. Auch bei Umlauten verhält sich Python nicht nach der DIN Norm, bei der entweder ü=u ist bzw. ü=ue. Python sortiert die Umlaute nach den Vokalen ein.

4.7.4 Umwandeln von Datentypen

Du hast mit Ganzzahlen, Dezimalzahlen und Strings schon einige Datentypen in Python kennengelernt. Wie du weißt, ist es nicht notwendig, dass du Python mitteilst, von welchem Typ deine Variable ist, wenn du ihr einen Wert zuweist (schwache Typisierung). Nun kann es vorkommen, dass du direkten Einfluss auf den Datentyp einer Variablen nehmen und diesen umwandeln möchtest. Dazu stehen dir die Methoden

- int()
  Wandelt in eine Ganzzahl um.
  Zahl1 = int(5.34567) ergibt den Wert 5 für die Variable Zahl1.
- float()
  Wandelt in eine Dezimalzahl um.
  Zahl2 = float(3) oder Zahl2 = float("3") ergibt den Wert 3.0 für die Variable Zahl2
- str()
  Wandelt in einen String um.
  Zahl3 = str(3) oder Zahl3 = str(3.0) ergibt den Wert "3"bzw. "3.0"für die Variable Zahl3

Aufgabe 6: (Lösung S. 24)
Erstelle eine Datei **umgang-mit-datentypen.py** und löse darin folgende Aufgaben:

1. Es sind die beiden Dezimalzahlen zahl1=1.23 und zahl2=4.56 gegeben. Füge die beiden Variablen zu den Ergebnissen ergebins1=1.234.56 und ergebnis2=123456 zusammen.
2. Schreibe ein Programm, das testet ob **zahl1 = drei hoch 9** kleiner ist als **zahl2 = die vierte Wurzel aus 6000**. Der Wert von zahl1 und zahl2 interessiert dabei nicht.
3. Wird Meier im Alphabet vor Musterer einsortiert?
4. Teste, ob die Dezimalzahl 2.00 den gleichen Wert hat wie die Ganzzahl 2.

5 Programme smarter machen

5.1 Benutzereingaben

Jetzt hast du lange genug mit starren Programmen gearbeitet. Nun schauen wir uns an, wie du mit deinen Programmen interagieren kannst. Informationen kannst du ja schon über die *print()* Methode ausgeben. Für Benutzereingaben gibt es die *input()* Methode.

```
1 benutzerEingabe1 = input()
2 benutzerEingabe2 = input("Bitte gib etwas ein:")
```

Beachte, dass Nutzereingaben immer als String gelesen werden.

Aufgabe 7: (Lösung S. **25**)
Erstelle eine Datei **benutzereingaben.py** und löse darin folgende Aufgaben:

1. Schreibe ein Programm, dass dich nach deinem Namen fragt und dich danach freundlich begrüßt.
2. Erstelle ein Additionsprogramm, das zwei beliebige Zahlen, die du über die Tastatur eingibst addiert.

5.2 Fallunterscheidung

So langsam kommen wir mit dem Programmieren in die Gänge. Was auf jeden Fall fehlt, ist abhängig von der Benutzereingabe reagieren zu können. Dein Programm fragt den Nutzer, ob Mann oder Frau und danach begrüßt dein Programm entsprechend. Z.B. *Hallo Frau Meier, ...*

An diesem Beispiel wird klar, dass dein Programm Entscheidungen treffen können muss, abhängig von bestimmten Vorgeben. Da *Python* aus dem Englischen kommt, lautet das *reservierte Wort* zur Fallunterscheidung **for**. Eng damit verbuden sind **elif** für "*else if* und **else**.

Diese Wörter alleine reichen nicht. Dein Programm muss in der Lage sein gegen eine Bedingung zu prüfen. Im Sprachgebrauch macht es ja auch keinen Sinn, wenn jemand

nur sagt: „Wenn du.“ Da gehört dann schon eine Bedingung dazu: „Wenn du jetzt nicht aufräumst.“

In den Sprachgebrauch übertragen:

if	→	wenn
elif	→	sonst, wenn
else	→	sonst

Um bei dem Beispiel mit der Begrüßung zu bleiben, müsste das Programm in etwas so ablaufen:

- „Wie heißt du?“ (Die Eingaben speicherst du dann in eine Variable)
- „Bist du ein Mann oder eine Frau?“ (Die Eingaben speicherst du dann in einer anderen Variable)
- Wenn das Geschlecht „Mann“ ist (Hier prüfst)
  - dann sage: „Hallo Herr ...“ (Hier regierst du)
- Wenn das Geschlecht nicht „Mann“, sondern “Frau“ ist (Hier prüfst erneut, wenn die erste Prüfung negativ war.)
  - dann sage: „Hallo Frau ...“ (Hier regierst du)
- Wenn die Eingabe weder „Mann“, noch „Frau“ war (Hier prüfst erneut, wenn die erste Prüfung negativ war.)
  - dann sage: „Ein Fehler ist aufgetreten.“ (Hier regierst du)

In Python sieht der Aufbau so aus:

```
1  variable1=5
2  variable2=5
3
4  if␣variable1␣<␣variable2:␣#␣nach␣if␣folgt␣die␣zu␣pruefende␣Bedingung
5  ␣␣␣␣print(variable1,"ist␣kleiner␣als",variable2)
6  ␣␣␣␣print("das␣war␣es")
7
8  elif␣variable1␣>␣variable2:␣#␣hier␣folgt␣die␣zweite␣zu␣pruefende␣Bedingung
9  ␣␣␣␣print(variable1,"ist␣groesser␣als",variable2)
10 ␣␣␣␣print("fertig")
11
12 else:␣#␣trifft␣nichts␣zu,␣wird␣das␣else␣ausgefuehrt
13 ␣␣␣␣print(variable1,"ist␣weder␣kleiner␣noch␣groesser␣als",variable2)
14 ␣␣␣␣print("fertig")
```

Beachte, dass die Blöcke nach der Prüfung **genau vier** Leerzeichen weit eingerückt ist. So weiß Python, weche Befehle zu einem Block gehören und nach erfolgreicher Prüfung ausgeführt werden müssen.

Hier nochmals der grundlegende Aufbau:

```
wenn Bedingung 1 wahr:
␣␣␣␣dann tue
ansonsten, wenn Bedingung 2 wahr:
␣␣␣␣dann tue
  (usw.)
ansonsten:
␣␣␣␣dann tue
```

Auf Seite 16 kannst du nochmals nachlesen, welche Operatoren dir zur Verfügung stehen, um eine Bedingung zu stellen.

Aufgabe 8: (Lösung S. **25**)
Erstelle ein Programm in der Datei **begruessung.py**, welches den Nutzer nach dem Nachnamen und dem Geschlecht fragt. Abhängig von der Antwort, begrüßt dein Programm den Nutzer.

6 Lösungen zu den Aufgaben

Aufgabe 1:
Kapitel 1 Variablen auf S.9
Wenn du die Datei 02-variablenNamen.py ausführst passiert nichts. Woran liegt das? Das liegt daran, dass das Programm von dir noch keine Anweisung erhalten hat, den Inhalt einer Variablen anzuzeigen. Dies kannst du mit dem *print()* Befehl nachholen.

```
''' weitere Beispiele zum Einfuehren von Variablen'''
# weitere Beispiele zum Einfuehren von Variablen
variable1 = 70
variable2 = "Knoedel"
variable3 = "Schranktuer"

# mehrere in einer Zeile geht auch
variable4, variable5 = 30, "Federtier"

print(variable1)
print(variable2, variable3)
print(variable4+5)
```

Aufgabe 2:
Kapitel 2 Mehrere Parameter auf S.11
Zwischen den einzelnen Parametern wird automatisch ein Leerzeichen eingefügt.

```
# Variablen werden deklariert (eingefuehrt) und initialisiert (mit einem Wert belegt)
# mehrere Variablen in einer Zeile einfuehren geht auch
alter, lieblingsTier = 16, "Hund"
name = "Fritz"

# der Print() Befehl mit mehreren Parametern, incl. Variablen.
print(name,"hat im Alter ↩
    von",alter,"einen",lieblingsTier,"geschenkt bekommen.")
```

Aufgabe 3:
Kapitel 3 Moglichkeiten der Wertzuweisung bei Variablen auf S.13
Hier das Programm. Teste, ob du richtig gelegen bist.

```
#1.
x = 16
print ("Bei x = 16 hat x den Wert:",x)

#2.
x = 11+3
print ("Bei x = 10+3 hat x den Wert:",x)

#3.
x = 11/2
print ("Bei x = 11/2 hat x den Wert:",x)

#4.
x = 11//2
print ("Bei x = 11//2 hat x den Wert:",x)

#5.
x = 11%2
print ("Bei x = 11%2 hat x den Wert:",x)

#6.
x=3
x %= 7
print ("x= 3. Bei x %= 7 hat x den Wert:",x)

#7.
x=7
x //= 3
print ("x= 7. Bei x //= 3 hat x den Wert:",x)

#8.
x=3
x **= 3
print ("x= 3. Bei x **= 3 hat x den Wert:",x)
```

Aufgabe 4:

Kapitel 4 Formatieren von Strings (Sternchen-Abschnitt) auf S.14

Arbeiten mit Strings

```
# Erg: Heute haben wir schon viel gelernt. Gestern habe ich 5 geuebt und dabei ➥
    1.500000 Liter Wasser getrunken
nachricht1 = "Nachricht 1: Heute haben wir %s viel gelernt. Gestern ➥
    habe ich %d geuebt und dabei %f Liter Wasser getrunken" % ➥
    ("schon",5,1.5)
```

```
print(nachricht1)

# Erg: Heute haben wir schon viel gelernt. Gestern habe ich 5 geuebt und dabei 1.50 Liter Wasser getrunken
nachricht2 = "Nachricht 2: Heute haben wir %s viel gelernt. Gestern habe ich %d geuebt und dabei %3.2f Liter Wasser getrunken" % ("schon",5,1.5)
print(nachricht2)

# Erg: Heute haben wir schon viel gelernt. Gestern habe ich     5 geuebt und dabei 1.50 Liter Wasser getrunken
nachricht3 = "Nachricht 3: Heute haben wir %s viel gelernt. Gestern habe ich %10d geuebt und dabei %3.2f Liter Wasser getrunken" % ("schon",5,1.5)
print(nachricht3)

# Erg: Heute haben wir schon viel gelernt. Gestern habe ich     5 geuebt und dabei 1.50 Liter Wasser getrunken
nachricht4 = "Nachricht 4: Heute haben wir %s viel gelernt. Gestern habe ich %10d geuebt und dabei %10.2f Liter Wasser getrunken" % ("schon",5,1.5)
print(nachricht4)

# Erg: Heute haben wir schon viel gelernt. Gestern habe ich     5 geuebt und dabei 1.50 Liter Wasser getrunken
nachricht5 = "Heute haben wir {2:s} viel gelernt. Gestern habe ich {0:5d} geuebt und dabei {1:5.3f} Liter Wasser getrunken".format(5,1.5,"schon")
print(nachricht5)
```

Aufgabe 5:
Kapitel 5 Formatieren von Strings (Sternchen-Abschnitt) auf S.15
Arbeiten mit Strings

```
nachricht1="Bei der {2:s} von 5 : {0:2d} lautet das Ergebnis {1:5.2f}".format(2,2.5, "Division")
print(nachricht1)
#Ergebnis: Bei der Division von 5 : 2 lautet das Ergebnis 2.50

nachricht2="{4:3d} Promille im {3:s} ist ein sehr hoher {2:s}. Zu{1:7.2f} Prozent ist man nicht mehr {0:s}. Also nie mit {4:3.1f} Promille unterwegs sein!".format("fahrtuechtig",100.0,"Wert","Blut",1)
print(nachricht2)
```

```
#Ergebnis:  1 Promille im Blut ist ein sehr hoher Wert. Zu 100.00 Prozent ist man nicht mehr fahrtuechtig. Also nie mit 1.0 Promille unterwegs sein!
```

Aufgabe 6:
Kapitel 6 Umwandeln von Datentypen auf S.17
Umgang mit Datentypen

```
# Aufgabe 1
#ergebnsi1=1.234.56
zahl1 = 1.23
zahl2 = 4.56
ergebnis1 = str(zahl1) + str(zahl2)
print(ergebnis1)

zahl1 *= 100
zahl2 *= 100
print(zahl2)
# du siehst, dass Python nicht ganz genau rechnet. Anstatt 456 gibt Python 455.99999999 zurueck. Deshalb wenden wir einen kleinen Trick an:

# wir schiessen etwas ueber das Ziel hinaus
zahl2 += 1
print(zahl2)

# nun klappt es mit 123456
ergebnis2 = str(int(zahl1)) + str(int(zahl2))
print(ergebnis2)
''' Dieses Beispiel zeigt, dass nicht immer alles so funkeioniert, wie man es sich erwartet.
Es koennte auchsein, dass auf einem anderen System oder in einer anderen Versione dieser Fehler so nicht auftaucht.'''

# Aufgabe 2: zahl1 ist nicht kleiner als zahl2
zahl1 = 3 ** 9
zahl2 = 6000 ** (1/4)
print(zahl1 < zahl2)

# Aufgabe3: Antwort, ja
wort1 = "Meier"
wort2 = "Musterer"
print(wort1<wort2)

# Aufgabe4: Antwort, ja
zahl1 = 2.00
```

```
zahl2 = 2
print(zahl1 == zahl2)
```

Aufgabe 7:
Kapitel 7 Benutzereingaben auf S.18
Benutzereingaben

```
# Benutzereingaben
# 1
# Es wird nach dem Namen gefragt.
# nutze nicht name als Variable, da dieses Wort in Python schon vergeben sein koennte
derName = input("Wie heisst du? ")
print("Hallo", derName, ", schoen dass du da bist")

# 2
'''Da Nutzereingaben immer als String gelesen werden,
muessen wir diese erst in eine Zahl umwandeln, hier eine Ganzzahl'''
zahl1 = int(input("Bitte gib eine Zahl ein: "))
zahl2 = int(input("Bitte gib noch eine Zahl ein: "))

ergebnis = zahl1 + zahl2

print("Das Ergbnis der Addition von", zahl1, "und", zahl2, ↩
    "lautet:",ergebnis)
```

Aufgabe 8:
Kapitel 8 Fallunterscheidung auf S.20
Benutzereingaben

```
# Der Nachname und das Geschlecht werden vom Nutzer abgefragt
Nachname=input("Wie heist du mit Nachnamen? ")
Geschlecht=input("Gib bitte dein Geschlecht als m oder w ein. ")

if Geschlecht == "m": # ein Mann
    print("Hallo Herr", Nachname)

elif Geschlecht == "w": # eine Frau
    print("Hallo Frau", Nachname)

else: # trifft nichts zu, wird das else ausgefuehrt
    print("Willkommen", Nachname)
```