4 Programmablauf-Steuerung

Wie wir im vorangehenden Kapitel gelernt haben, wird ein Programm zeilenweise von oben nach unten abgearbeitet. Nun gibt es Fälle, wo wir einen Programmblock nur unter einer Bedingung ausführen oder einige Anweisung mehrmals wiederholen wollen.
Es ist oft notwendig, einen ganzen Programmblock für jedes Element einer Liste zu wiederholen. Dies ist eine Form der Iteration. Wir sprechen von einer Iteration über den Elementen einer Liste. In Python wird dies mit einer for Schleife gemacht.[4] Wir werden in diesem Zusammenhang auch lernen, wie in Python ein Programmblock gekennzeichnet wird.
In anderen Fällen ist es notwendig, zu entscheiden, ob ein Programmblock ausgeführt werden soll oder nicht. Hier handelt es sich um eine Selektion. Dafür müssen wir zuerst lernen, wie in Python Bedingungen formuliert werden. Anschließend betrachten wir die if Verzweigung, welche uns das bedingte Ausführen eines Programmblocks ermöglicht.
Teilweise reicht diese sehr spezielle Form der Wiederholung nicht. Es kann auch notwendig sein, einen Block zu wiederholen, bis eine Bedingung nicht mehr wahr ist. Dies nennen wir eine bedingte Iteration.

4.1 Die for Schleife

Wie schon erwähnt, ist die for Schleife ein Spezialfall der Iteration, nämlich eine Iteration über einer Liste. Man kann alle Probleme, welche mit for gelöst werden können, auch mit der später besprochenen, allgemeineren while Schleife lösen. Das Iterieren über einer Liste ist aber eine so häufige Form der Iteration, dass for häufiger vorkommt als while.
Wir wollen ein erstes Beispiel einer solchen Schleife in Python betrachten:

```
1 for zahl in [4,5,7,11,21]:
2     summe = zahl + 2
3     print("Wenn ich zu", zahl, "zwei addiere, erhalte ich", summe)
4 print("Ich bin fertig!")
```

Hier nimmt die Variable zahl jeden Wert aus der Liste einmal an und durchläuft alle Anweisungen innerhalb der Iteration. Sobald Python beim letzten Element in der Liste angelangt ist, bricht die Schleife ab und es wird bei der ersten Anweisung nach der Schleife weitergefahren.
In diesem Fall gehörten zwei Anweisungen zum Programmblock, welcher wiederholt wird, nämlich die Berechnung der Variable summe sowie die Ausgabe mit print(). Diese Anweisung wurde nach innen gerückt, um Python mitzuteilen, dass sie zu der Schleife gehört.
Wenn du in PyCharm den Doppelpunkt am Ende der Zeile nach der Liste nicht vergisst, rückt PyCharm die nächste Zeile automatisch ein. Andernfalls kannst du dies mit der <TAB>-Taste selbst machen. Achte darauf, dass wirklich alle Zeilen, welche zum gleichen Block gehören, gleich stark eingerückt sind. Schon nur ein Leerzeichen unterschied wird zu einer Fehlermeldung führen. [5]
Möchte man nun zum Beispiel etwas für die Zahlen von 1 bis 100 ausführen, wird es mühsam, die Liste von Hand wie oben einzutippen. Dafür gibt es in Python den Befehl range(). So enthält range(100) alle Zahlen von 0 bis 99 (also insgesamt 100 Zahlen) und range(1, 101) alle Zahlen von 1 bis 100.

[4] Falls du schon andere Programmiersprachen kennst, kann es sein, dass dich die for Schleife in Python zuerst verwirrt. Sie ist nicht gleich aufgebaut wie etwa in C++ oder Java.

[5] Dies ist einer der Nachteile von Python. Andere Sprachen benutzen geschweifte Klammern und um zusammengehörende Blöcke zu kennzeichnen.

Bemerkung
Die letzte Zahl, welche wir dem range() Befehl übergeben, wird aus dem Bereich ausgeschlossen. Will man alle Zahlen bis 100, muss man als Ende des Bereichs 101 angeben. Der Anfang ist jedoch enthalten. Gibt man 1 als Anfang an, so ist auch die 1 noch im Bereich enthalten.

Mit list() kannst du solche Bereiche in wirkliche Listen konvertieren, um an der Python-Konsole zu betrachten, wie sie aussehen:

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1,11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Einen Bereich können wir nun wie folgt benutzen, um zum Beispiel alle Zahlen von 0 bis 99 aufzulisten.

```
1 for zahl in range(100):
2     print(zahl)
```

4.1.1 Aufgaben

1. Gegeben ist der folgende Anfang eines Programms:

   ```
   animals = ["tiger", "mouse", "bird", "python", "elephant"]
   # ...
   ```

   Ergänze das Programm so, dass für jedes Tier aus der Liste animals der Satz „... ist ein Tier“ in der Konsole ausgegeben wird. Benutze dafür die print() Funktion.

2. Mit append() kannst du einer Liste ein Element hinten anfügen. Schreibe ein Programm, welches den Benutzer mit input() fünf mal nach einem Wort fragt und alle diese Worte in einer Liste abspeichert. Anschliessend werden alle Worte aus der Liste mit print() zurückgegeben.

   Erweiterung: Frage den Benutzer zuerst, wie viele Worte er eingeben will und frage dann genau so viel mal nach einem Wort.

3. In der Mathematik gibt es die Schreibweise

   $$n! := n \cdot (n-1) \cdot (n-2) \cdots 3 \cdot 2 \cdot 1$$

   So wird zum Beispiel $5!$ durch $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$ berechnet. Dies wird in der Mathematik als Fakultät von 5 bezeichnet.

   Schreibe ein Programm welches $n!$ für ein vom Benutzer vorgegebenes n berechnet. Benutze dafür eine for Schleife.

4. Schreibe ein Programm, welches die folgende Ausgabe bis zu einer vom Benutzer gewählten Zahl fortsetzt:

```
1*
2**
3***
4****
5*****
6******
7*******
...
```

Zusatz: Passe das Programm so an, dass es einen Weihnachtsbaum aus Sternchen zeichnet, dessen Höhe vom Benutzer gewählt werden kann.

5. Passe das in einer früheren Aufgabe erweiterte Notenprogramm so an,dass der Benutzer wählen kann, wie viele Noten er eingeben möchte.

4.2 Die if Verzweigung

Wie oben erklärt, kommt es vor, dass wir einen Programmblock nur unter einer Bedingung ausführen wollen. Falls die Bedingung nicht erfüllt ist, führen wir entweder nichts oder einen anderen Block aus.

```
1 antwort = input("Möchtest du Feierabend: ")
2
3 if antwort in ["Ja", "ja", "jep"]:
4     print("Sehr gut!")
5 else:
6     print("Ich glaube dir nicht!")
```

Mit dem Wort if wird eine Bedingung abgefragt. In diesem Fall ist dies die Frage, ob die als antwort gespeicherte Zeichenkette in der Liste enthalten ist oder nicht. (Dieses Enthalten-Sein wird mit dem Wort in überprüft.)
Der Programmblock, welcher nach dem Wort else folgt, wird ausgeführt, falls die Bedingung oben falsch ist. Die wichtigen Zeilen sind also die Zeilen

```
if antwort in ["Ja", "ja", "jep"]:
```

und

```
else:
```

Achte auch hier darauf, wie die print() Befehle eingerückt wurden, um Python mitzuteilen, welche Zeilen zu den Blöcken gehören, welche nur bedingt ausgeführt werden sollen.

4.2.1 Aufgaben

1. Für Vergleiche, ob etwas grösser oder kleiner ist, können die in der Mathematik üblichen Zeichen < und > benutzt werden. Schreibe ein Programm, welches die folgenden Schritte ausführt:

   (a) Es wird vom Benutzer per input() eine Zahl eingelesen.

   (b) Es wird geprüft, ob die Zahl größer oder kleiner als 10 ist.

   (c) Mit dem Befehl print() wird entsprechend entweder „Die Zahl ist größer als 10“ oder „Die Zahl ist kleiner als 10“ ausgegeben.

2. Schreibe ein Programm, welches vom Benutzer 10 Zahlen einliest und diese in einer Liste speichert. Anschließend soll das Minimum und das Maximum der Zahlen aus der liste bestimmt und ausgegeben werden.

3. Ob ein Jahr im Gregorianischen Kalender ein Schaltjahr ist, kann nach den folgenden Regeln entschieden werden:

   - Jahre sind Schaltjahre, falls ihre Jahrzahl durch 400 teilbar ist.
   - Jahre sind Schaltjahre, falls ihre Jahrzahl durch 4 teilbar ist, außer die Jahrzahl ist durch 100 teilbar.

   Schreibe ein Programm, welches den Benutzer nach einer Jahreszahl fragt und anschließend prüft, ob es sich um ein Schaltjahr handelt.

4.3 Die while-Schleife

Es gibt Fälle, wo eine for Schleife nicht ausreicht, weil wir keine Liste von Elementen haben und auch nicht zu Beginn wissen, wie oft etwas ausgeführt werden soll.
Die while Schleife wird solange ausfeführt wie eine Bedinung wahr ist. Sie hat die folgende Struktur:

```
while Bedingung:
    Befehl 1
    Befehl 2

Befehl 3
```

Dabei werden die Befehle 1 und 2 solange ausgeführt, wie die Bedingung wahr (also True) ist. Sobald die Bedinung falsche (False) wird, springt Python zur nächsten nicht eingerückten Zeile, also Befehl 3. Zur Formulierung der Bedingung können dieselben Formen wie bei der if Abfrage benutzt werden.
Man könnte sich zum Beispiel vorstellen, dass wir in einem Programm dem Benutzer eine Frage gestellt haben, welche mit „Ja“ oder „Nein“ zu beantworten ist. Wenn der Benutzer nicht entweder J für Ja oder N für Nein eingibt, ist die Eingabe ungültig und wir müssen nochmals nachfragen. Dies würde in Python wie folgt umgesetzt:

```
antwort = input('Beantworte die Frage mit Ja oder Nein (J/N): ')

while antwort not in ['j', 'J', 'n', 'N']:
    print('Eingabe ungültig!')
    antwort = input('Beantworte die Frage mit Ja oder Nein (J/N): ')

print('Vielen Dank!')
```

Der eingerückte Codeblock wird so lange wiederholt, wie die Bedingung antwort not in [’j’, ’J’, ’n’, ’N’] wahr ist. Das heißt, so lange die Antwort nicht in der Liste mit gültigen Antworten enthalten ist.

4.3.1 Aufgaben

1. Schreibe ein Programm, welches alle Zahlen von 1 bis 100 auf den Bildschirm schreibt, ohne dafür eine for Schleife zu verwenden.

2. Aus dem Kapitel über die for Schleife kennst du das folgende Beispielprogramm:

   ```
   for zahl in [4,5,7,11,21]:
       summe = zahl + 2
       print("Wenn ich zu", zahl, "zwei addiere, erhalte ich", summe)
   print("Ich bin fertig!")
   ```

   Schreibe das Programm so um, dass es eine while Schleife anstelle der for Schleife benutzt.

3. Die Folge aus Fibonacci-Zahlen wird wie folgt gebildet:

   - Das erste und das zweite Element sind 0 und 1.
   - Jedes folgende Element wird gebildet, in dem die letzten zwei Elemente addiert werden.

Das heisst, die Folge sieht so aus: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \cdots$

Schreibe ein Programm, welches die Fibonacci-Zahlen bis zu einer vom Benutzer gewählten Zahl ausgibt.

4. Der Wert einer Immobilie steigt jedes Jahr um p Prozent an. Schreibe ein Programm, welche den Zeitwert dieser Immobilie so lange berechnet, bis der Zeitwert (mehr als) doppelt so groß ist wie der heutige Anfangswert. Frag den Benutzer nach dem Prozentsatz.

5. Eine Firma kauft eine Maschine (der Einkaufspreis ist einzugeben), welche innerhalb einiger Jahre (die Anzahl Jahre ist einzugeben) linear abgeschrieben wird.

   Beispieleingabe:

   20000 (Euro Einkaufspreis), 5 (Jahre)

   Beispielausgabe:

   Jahr 1: 16000.00 Jahr 2: 12000.00 Jahr 3: 8000.00 Jahr 4: 4000.00 Jahr 5: 0.00

   Schreibe ein Programm, das diese Berechnung mit der passenden Ausgabe durchführt.

6. Herr Spar legt sich jedes Jahr einen festen Betrag (z.B. 800.00 Euro) auf sein Konto, welches beispielsweise $2,5\%$ Zins trägt. Er tut dies so lange, bis er einen bestimmten Betrag überschreitet. Dieser Betrag ist auch einzugeben, z.B. 5000.00. Schreibe dazu ein Programm.

7. Ein Gegenstand hat den Wert k (dieser Wert ist eine Eingabe). Erstelle ein Programm, welches für die nächsten Jahre (die Anzahl Jahre ist ein Eingabewert) den Zeitwert dieses Gegenstands berechnet, wenn jedes Jahr p Prozent des Werts abgeschrieben werden. (Der Abschreibungssatz ist auch einzugeben.) Diese Art Abschreibung heißt degressive Abschreibung.

8. Wie lange muss man einen Würfel werfen, bis (zum ersten Mal) eine 6 erscheint? Schreibe ein Programm, welches so lange würfelt, bis eine 6 erschienen ist.

   Eingabe: keine

   Ausgabe: alle geworfenen Zahlen und die Anzahl Würfe.

9. Ein Würfel wird so lange geworfen, bis die Summe aller Würfe 100 oder mehr beträgt. Zum simulieren eines Würfels kannst du die Funktion randrange() aus dem Modul random benutzen.

   (a) Bestimme mit einem Programm, wie viele Würfe dazu nötig sind.

   (b) Schreibe ein Programm, welches mit Hilfe einer Liste zählt, wie oft welche Augenzahl geworfen wurde. Am Ende soll die Liste mit print() ausgegeben werden.

   (c) Ändere das Programm so ab, dass so lange gewürfelt wird, bis zum ersten Mal eine Zahl 50 Mal geworfen wurde. Gib an, wie oft die anderen Zahlen geworfen wurden.

10. Der Computer „würfelt“ eine Zufallszahl zwischen 1 und 6 (Grenzen inklusive). Der Spieler versucht nun, die gewürfelte Zahl zu erraten. Bestimme mit einem Programm, wie viele Versuche dazu nötig sind.

11. Schreibe ein Programm, welches mit input() zwei Zahlen vom Benutzer einliest und den grössten gemeinsamen Teiler der beiden Zahlen mit print() ausgibt.

    Dazu kannst du den Euklidischen Algorithmus benutzen, welchen du entweder aus dem Mathematikunterricht kennst oder sonst sicher im Internet findest.

4.4 Kombinieren von Kontrollstrukturen

Wie du sicherlich schon festgestellt hast, kommt man nie mit einer einzelnen Kontrollstruktur aus, die meisten Problemstellungen benötigen ein geschicktes Kombinieren von if, for und while. Dazu kommt meistens noch die Wahl einer Datenstruktur: Macht es Sinn, einzelne Variablen zu benutzen oder ist etwa eine Liste geeigneter? Halte dich beim Programmieren an folgende Grundsätze:

- Nummeriere keine Variablen. Wenn du in deinem Programm etwa Variablen-Namen x_wert_1, x_wert_2 und x_wert_3 hast, ist dies ein Hinweis darauf, dass du vermutlich besser eine Liste verwenden würdest.
- Benutze wenn immer möglich ein elif, wenn du mehrere if-Abfragen hintereinander ausführst.
- Natürlich gilt weiterhin: Schreibe am besten zuerst die Kommentare für dein Programm und dann den Programmcode. Falls du dies lieber nicht machst, achte auf jeden Fall darauf, dass du jeden logisch zusammenhängenden Programmblock mit einem oder mehreren -Kommentarzeilen erklärst.

Löse die folgenden Aufgaben unter Beachtung der genannten Grundsätze.

4.4.1 Aufgaben

1. Schreibe ein Programm, welches den Benutzer mit Hilfe von input() nach einem Satz fragt.
   (a) Das Programm soll anschließend die Anzahl Worte im Satz sowie die Anzahl Buchstaben ausgeben.
   (b) Erweitere das Programm so, dass es auch angibt, wie viele Buchstaben davon Großbuchstaben sind.
2. Schreibe ein Programm, welches den Benutzer nach einer Zahl fragt und anschliessend prüft, ob diese Zahl eine Primzahl ist.
3. Schreibe ein Programm, welches alle Primzahlen zwischen 1 und einer vom Benutzer gewählten oberen Grenze ausgibt.

   Zusatzaufgabe: Versuche im Internet herauszufinden, wie man dieses Problem möglichst effizient (d.h. so, dass es wenig Zeit braucht) löst und schreibe dein Programm um, damit es schneller wird.
4. Im Zahlenlotto werden sechs Zahlen aus 49 zufällig ausgewählt, wobei keine doppelt vorkommen darf. Schreibe ein Programm, welches 6 Zahlen aus 49 zufällig wählt. Verwende dafür wiederum Funktionen aus dem random Modul.
   (a) Benutze ausschliesslich die randrange() Funktion, um dies zu erreichen.
   (b) Lies die Dokumentation des random Moduls und versuche jene Funktion zu finden, mit welcher du die Aufgabe am einfachsten lösten kannst. (Das heisst mit möglichst wenig Programmcode)
5. Das Collatz-Problem[6] gehört zu den ungelösten Problemen der Mathematik. Es besagt, dass jede Abfolge von Zahlen, welche nach der folgenden Regel gebildet wird, irgendwann bei der Zahl 1 landet:

[6] https://de.wikipedia.org/wiki/Collatz-Problem

- Die Folge beginnt bei einer beliebigen Zahl.
- Ist die momentane Zahl n gerade, so ist die nächste Zahl die Hälfte von dieser, also $\frac{n}{2}$.
- Ist die momentane Zahl n ungerade, so ist die nächste Zahl um eins größer als das dreifache der Zahl, also $3n + 1$.

Bilden wir beispielsweise für die Zahl 23 diese Folge, so lautet sie:

$$23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1$$

Schreibe ein Programm, welches diese Folge für eine vom Benutzer gewählte Zahl ausgibt und mit der Berechnung stoppt, sobald 1 erreicht wurde.

6. Schreibe ein Programm, welches eine Liste von Zahlen sortiert und gib das Ergebnis in der Konsole aus. Für die Liste [1,4,3,3,5,7,2] sollte das Ergebnis die Liste [1,2,3,3,4,5,7] sein.

   (a) Speichere die Liste in der ersten Zeile als Variable ab und sortiere sie anschliessend.

   (b) Frage den Benutzer nach der Liste von Zahlen. Überlege dir, wie du es am besten lösen kannst, dass der Benutzer eine vorher nicht bekannte Anzahl Zahlen eingeben kann.

7. **Zusatz:** Schreibe ein Programm, welches eine Liste von vom Benutzer eingegebenen Worten alphabetisch sortiert. Der Benutzer gibt etwa die Worte Zweck, schneiden, Granit, Gewinn, Schubkarre und entflammen ein und dein Programm gibt folgendes aus:

```
entflammen
Gewinn
Granit
schneiden
Schubkarre
Zweck
```

8. **Zusatz:** Schreibe ein Programm, welches den Benutzer nach einem oder mehreren deutschen Sätzen fragt. Anschliessend verändert das Programm den Text so, dass bei jedem Wort der erste und der letzte Buchstabe stehen gelassen werden und alle Buchstaben dazwischen jeweils wie Jasskarten gemischt werden. Das heisst, dass keine Buchstaben dazu kommen und auch keine entfernt werden, jedoch die Reihenfolge verändert wird. Gib anschliessend den so entstandenen Text aus. Was stellst du fest?

Abgewandelt von `https://pythonbuch.com/kontrollstrukturen.html`