Einführung in das Programmieren

Übungsblatt 5

Christoph Draxler, Vanessa Reichel

November 13, 2024

1 Schleifen

... bestehen aus drei Komponenten: der **Initialisierung**, dem **Schleifendurchlauf** und der **Rückgabe** des berechneten Wertes.
Wir unterscheiden zwei verschiedene Schleifen: **for-Schleifen** und **while-Schleifen**.
for-Schleifen sind **zählende** Schleifen. Dabei wird über eine mehrwertige Datenstruktur (z.B. eine Liste oder ein Dictionary) iteriert. Der folgende Code wird für jedes Element der Datenstruktur ausgeführt. Die Schleife endet, wenn die Datenstruktur, über die iteriert wird, einmal komplett durchlaufen wurde.
while-Schleifen sind **bedingte** Schleifen. Nach dem Ausdruck `while` wird im Code eine Bedingung formuliert. Der nachfolgende Code wird solange ausgeführt, bis die Bedingung nicht mehr erfüllt ist.

1.1 Häufigkeiten zählen

Zähle mithilfe einer for-Schleife, wie oft ein bestimmter Wert in einer Liste vorkommt.
Bei `haeufigkeit_zaehlen(["a", "b", "c", "a", "a", "c"], "a")` soll `3` zurückgegeben werden, da `"a"` dreimal in der Liste vorkommt.
Bei `haeufigkeit_zaehlen([10, 30, 20, 20, 30], 30)` soll `2` zurückgeben werden, da 30 zweimal in der Liste vorkommt.

1.2 Suchmaschine

Erstelle nun eine Funktion, die vom Anwender eine Zahl und eine Liste entgegennimmt und zurückgibt, wo in der Liste sich die Strings mit der gewünschten Zeichenanzahl befinden. Nenne die Funktion `vorkommen_in_liste(zahl, liste)`.

Beispiel: Beim Aufruf der Funktion `vorkommen_in_liste(4,["Heute", "ist", "nicht", "alle", "Tage", "ich", "komm", "wieder", "keine", "Frage"])` muss das Ergebnis `3 4 6` zurückgegeben werden.

1.3 Ratespiel

Schreibe eine Funktion `ratespiel()`, in der ein Spieler so lang eine gewürfelte Zahl erraten muss, bis er richtig liegt. Dabei soll die Funktion bei jedem falschen Raten `"Falsch!  Zahle 10 Cent!"` drucken und mitzählen wie viel man zahlen muss, bis die richtige Zahl geraten wurde. Damit sieht die Ausgabe beim Erraten wie folgt aus:
`"Richtig geraten!  In dieser Runde musstest du 40 Cent zahlen."`

Schreibe eine `while-Schleife`, welche so lang mitzählt, bis der Spieler die richtige Zahl errät. Du kannst erneut die bereits existierende Funktion `wuerfeln()` in die neue Funktion einbetten.
Tipp: Du benötigst den Befehl `input()`, mit dem eine Eingabe von der Konsole eingelesen werden kann. **Achtung**: `input()` gibt immer einen String zurück. Kann es zu Problemen kommen, wenn Du die Nutzereingabe mit dem Gewürfelten vergleichst?

1.4 Viel zu tun

Schreibe eine Funktion, die aus einer Liste alle doppelten Werte entfernt und die modifizierte Liste zurückgibt. Zum Beispiel soll die Funktion beim Aufrufen mit der Liste
`meine_to_dos = ["lernen", "aufraeumen", "kochen", "lernen", "aufraeumen" ]`
eine neue Liste mit den Werten `["lernen", "aufraeumen", "kochen"]` zurückgeben.

2 Dictonaries

In der letzten Stunde hast du Dictionaries als eine Datenstruktur kennengelernt, in der Werte als **Key-Value-Pairs** abgespeichert werden können. Ein Dictionary kann man sich als eine Tabelle mit zwei Spalten vorstellen: in der linken Spalte stehen die **Keys**, in der rechten Spalte die **dazugehörigen Values**.
Ein Beispiel:

```
aussaatplan = {
      "Frühling" :  ["Rote Beete", "Brokkoli", "Erbsen"],
      "Sommer" :  ["Radieschen", "Gelbe Rüben", "Tomaten", "Gurken"],
      "Herbst" :  ["Zwiebeln", "Kohlrabi", "Blaubeeren"],
      "Winter" :  ["Kresse", "Spinat", "Feldsalat"]
}
```

Die Befehle `keys()`, `values()` und `items()` kennst du bereits aus der Vorlesung.
Mit `dictname[key]` wird der zum Key gehörende Value ausgegeben.
`aussaatplan["Frühling"]` gibt `["Rote Beete", "Brokkoli", "Erbsen"]` zurück.
Nach dem selben Prinzip kann auch ein neuer Eintrag zu einem Dictionary **hinzugefügt** werden. `aussaatplan["Spätsommer"] = ["Rucola", "Rettich", "Kopfsalat"]` fügt

einen **neuen Key** (="Spätsommer") zum bestehenden Dictionary `aussaatplan` hinzu und schreibt `["Rucola", "Rettich", "Kopfsalat"]` als **Values** in die zweite Spalte.

2.1 Urlaubsplanung

Es ist Zeit für die Urlaubsplanung. In diesem Dictionary sind alle Gegenstände, die eingepackt werden müssen, als Keys gespeichert. Der Value ("ja" oder "nein") sagt aus, ob der Gegenstand schon eingepackt ist oder nicht.

```
urlaubsreif = {
    "t-Shirts" :  "ja",
    "Hosen" :  "ja",
    "Sonnencreme" :  "nein",
    "Snacks" :  "ja",
    "Perso" :  "nein",
    "Ladekabel" :  "nein",
    "Wanderschuhe" :  "ja"
}
```

- Überprüfe, ob "Sonnencreme" auf der Packliste steht
- Du packst den Perso ein. Ändere das Dictionary dementsprechend ab!
- Schreibe "Sonnenbrille" auf die Packliste. Der dazugehörige Value ist "ja".
- Anstelle der Wanderschuhe möchtest Du doch lieber Flipflops einpacken. Schreibe die Packliste entsprechend um!
- Wie viele Gegenstände stehen auf deiner Liste?
- Überprüfe, ob es außer `ja` und `nein` noch andere Values gibt.