Python – Übung 2

Selbststudium (als Hausaufgabe)

👉 Lesen Sie die folgenden Konventionen zum Codestil, die Sie im PEP8-Dokument finden: `https://www.python.org/dev/peps/pep-0008/`

- Introduction
- A Foolish Consistency is the Hobgoblin of Little Minds
- Code Lay-out
- String Quotes
- Whitespace in Expressions and Statements
- When to Use Trailing Commas
- Comments

👉 Aktivieren Sie die Echtzeit-Codestilanalyse in Spyder (Version 4.x), indem Sie in den Einstellungen das Häkchen bei **Vervollständigung und Lint** → **Codestil und Formatierung** → **Linten des Codestils aktivieren** setzen. Die Codestilfehler werden in Spyder links von den Zeilennummern angezeigt.

👉 Folgende Einstellungen (in **Editor** → **Quellcode**) helfen bei der Vermeidung von trivialen Codestilfehlern:

- Nachgestellte Leerzeichen beim Speichern von Dateien automatisch entfernen
- Eine neue Zeile am Ende einfügen, wenn diese beim Speichern einer Datei nicht vorhanden ist
- Überzählige Zeilenumbrüche am Ende einer Datei beim Speichern kürzen

1 Verzweigungen & Schleifen

👉 Lösen Sie folgende Aufgaben:

- Gegeben sei folgendes Dictionary:

```
stock = {
    "Apple": 12,
    "Pear": 0,
    "Banana": 6,
    "Cherry": 20,
}
```

  Prüfen Sie, ob eine entsprechende Frucht, deren Name in der Variable `fruit` gespeichert ist, im Dictionary `stock` vorkommt. Falls die Frucht im Dictionary vorkommt, soll die Meldung `"in stock"` auf die Konsole ausgegeben werden, ansonsten soll die Meldung `"not in product range"` ausgegeben werden.

- Erweitern Sie die obige Lösung, so dass `"out of stock"` ausgegeben wird, falls die Frucht im Dictionary `stock` vorkommt und deren Anzahl gleich Null ist.
- Die personenspezifischen Informationen eines Reisenden wurden bereits erfasst und liegen in Variablen vor, z.B.:

  ```
  age = 25
  halbtax = False
  ```

  Berechnen Sie den Preis für ein Zugbillet von Zürich nach Bern, unter Berücksichtigung der folgenden Bedingungen:

  - Kinder bis 6 Jahren fahren gratis.
  - Kinder von 6 bis 16 Jahren bezahlen den halben Preis.
  - Erwachsene (ab 16 Jahren) bezahlen den vollen Ticketpreis von CHF 135.2, ausser sie haben ein Halbtax, dann bezahlen sie den halben Preis.
- Gegeben sei eine Liste, die beliebige Elemente enthalten kann, z.B.:

  ```
  stuff = [1, 4, 9, "hallo", (1, 0.5), 23.5]
  ```

  Iterieren Sie elementweise durch diese Liste und geben Sie nur diejenige Elemente aus, die vom Typ `int` oder `float` sind. Benutzen Sie die eingebaute Funktion `isinstance()`.
- Ergänzen Sie die obige Lösung, so dass sobald ein falscher Datentyp erkannt wird, die Iteration abgebrochen wird und folgende Meldung ausgegeben wird: `"invalid list"`. Falls alle Elemente vom richtigen Typ waren, soll am Schluss die Meldung `"valid list"` ausgegeben werden.

1.1 Fortgeschrittene Schleifen

Die Elemente des iterierbaren Objekts in einer `for`-Schleife können wiederum mehrere Elemente haben, z.B. eine verschachtelte Liste:

```
bom = [
    ["R1", 3.3e3, "Ohm"],
    ["R2", 1000, "Ohm"],
    ["C1", 10e-6, "Farad"],
    ["D1", "1N4148", ""],
]
```

Im obigen Beispiel besteht jedes Listenelement aus einer Liste von drei Unterelementen. Mit der `for`-Schleife können diese Unterelemente jeweils bei jeder Iteration direkt an drei Hilfsvariablen zugewiesen werden und innerhalb der Schleife benutzt werden:

```
>>> for name, value, unit in bom:
        print("Das Bauteil", name, "hat den Wert:", value, unit)
Das Bauteil R1 hat den Wert: 3300.0 Ohm
Das Bauteil R2 hat den Wert: 1000 Ohm
Das Bauteil C1 hat den Wert: 1e-05 Farad
Das Bauteil D1 hat den Wert: 1N4148
```

1.1.1 `.items()`-Methode

Falls man durch ein Dictionary iterieren will, kann man dies analog zum obigen Beispiel mit Hilfe der `.items()`-Methode implementieren.

```
bom_dict = {
    "R1": [3.3e3, "Ohm"],
    "R2": [1000, "Ohm"],
    "C1": [10e-6, "Farad"],
    "D1": ["1N4148", ""],
}
```

```
>>> for k, v in bom_dict.items():
        print("Das Bauteil", k, "hat den Wert:", v[0], v[1])
Das Bauteil R1 hat den Wert: 3300.0 Ohm
Das Bauteil R2 hat den Wert: 1000 Ohm
Das Bauteil C1 hat den Wert: 1e-05 Farad
Das Bauteil D1 hat den Wert: 1N4148
```

1.1.2 `enumerate()`-Funktion

Will man über eine Sequenz iterieren und gleichzeitig eine Zählvariable zur Verfügung haben, die die Elemente durchnummeriert? In diesem Fall kann man die eingebaute `enumerate()`-Funktion[1] benutzen:

```
>>> names = ["Laura", "Victor", "Charlie"]
>>> for n, name in enumerate(names):
        print(n, name)
0 Laura
1 Victor
2 Charlie
```

1.1.3 `zip()`-Funktion

Will man gleichzeitig über mehrere Sequenzen iterieren, dann kann man die eingebaute `zip()`-Funktion[2] benutzen:

```
starter = ["Suppe", "Salat", "Bruschetta"]
main_dish = ["Rösti mit Leberli", "Fischknusperli", "Spaghetti Carbonara"]
dessert = ["Caramelköpfli", "Glace", "Tiramisu"]
```

```
>>> for s, m, d in zip(starter, main_dish, dessert):
        print(s, m, d, sep=", ")
Suppe, Rösti mit Leberli, Caramelköpfli
Salat, Fischknusperli, Glace
Bruschetta, Spaghetti Carbonara, Tiramisu
```

[1] https://docs.python.org/3/library/functions.html#enumerate
[2] https://docs.python.org/3/library/functions.html#zip

2 Aufgaben aus dem Buch

👉 Lösen Sie folgende Aufgaben aus dem Buch:

Kapitel	Seiten	Aufgaben
Verzweigungen	74	2
Schleifen	83-85	1, 2
Funktionen	131	3, 4

3 Stundenplan

👉 Implementieren Sie die Funktion `current_lesson(timetable)`, welche die Informationen über der Lektion, die laut dem übergebenen Stundenplan (`timetable`-Argument) gerade durchgeführt wird, als String zurückgibt, z.B. `"Py-p13 in room 1.255"` oder `"no lesson"`.

Hinweise: Übertragen Sie zuerst die nötigen Stundenplan-Informationen in die `timetable`-Variable. Wählen Sie dafür einen geeigneten Datentypen. Die aktuelle Zeitinformation kann z.B. mittels der `time.localtime()`-Funktion[3] aus dem `time`-Modul ermittelt werden:

```
import time
t = time.localtime()
print(t.tm_year)
```

4 Primzahlen

👉 Implementieren Sie die Funktion `primes(n)`, welche eine Liste mit allen Primzahlen bis zur Zahl `n` zurückgibt.

Beispiel:

```
>>> primes(50)
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

[3] `https://docs.python.org/3/library/time.html#time.localtime`