Matthias Mruzek-Vering, M.Sc.

Hochschule Karlsruhe
University of
Applied Sciences

DSCB 230 - Aufgabenblatt 3

Die Aufgabenblätter werden in dieser Veranstaltung in Jupyter Notebooks veröffentlicht und bearbeitet. Diese finden Sie in der Github Organisation für Data Science 2 unter dem Repository *dscb230-tutorial* (https://github.com/hka-mmv/dscb230-tutorial). Die Musterlösung wird ebenfalls in Form eines Jupyter Notebook in Github hochgeladen.

Aufgabenteil 1

Aufgabe A: Boolesche Operatoren und Aussagenlogik

Ermitteln Sie das Ergebnis der folgenden Booleschen Operatoren. Verwenden Sie hierzu keinerlei Hilfsmittel!

1. a == 6
2. a == 7
3. a == 6 and b == 7
4. a == 7 and b == 7
5. not a == 7 and b == 7
6. a == 7 or b == 7
7. a == 7 or b == 6
8. not (a == 7 and b == 6)
9. not a == 7 and b == 6
10. not (a == 7) and not(b == 6)
11. not (not (a == 7) and not (b == 6))
12. not (not (a == 7) and b == 6)
13. not (not (a == 7) or b == 6)
14. not (not (a == 7) | b == 6)
15. not (not (a == 7) & b == 6)

Erklären Sie den Unterschied zwischen **&** und einem **and**.
Erklären Sie den Unterschied zwischen **|** und einem **or**.

Tim Ehmann
Fabian Juny
Raphaele Salvatore Licciardo, B.Sc.

Aufgabe B: Funktionale Programmierung

Sie haben folgende Temperaturen in Celsius gegeben: [45.2, 36.5, 36.3, 28, 37.2] rechnen Sie diese in Fahrenheit um. Sie sollten hierfür manche der folgende Funktionen verwenden. Die Fortgeschrittenen sollten alle Verwenden:

- list
  - https://python-reference.readthedocs.io/en/latest/docs/functions/list.html
  - Converts an object into a list
- map
  - https://python-reference.readthedocs.io/en/latest/docs/functions/map.html
  - Applies function to every item of an iterable and returns a list of the results
- float
  - https://python-reference.readthedocs.io/en/latest/docs/functions/float.html
  - Returns an expression converted into a floating point number
- format
  - https://python-reference.readthedocs.io/en/latest/docs/functions/format.html
  - Returns a formatted string
- enumerate
  - https://python-reference.readthedocs.io/en/latest/docs/functions/enumerate.html
  - Returns an enumerate object
- round
  - https://python-reference.readthedocs.io/en/latest/docs/functions/round.html
  - Returns a floating point number rounded to a specified number of decimal places
- zip
  - https://python-reference.readthedocs.io/en/latest/docs/functions/zip.html
  - Returns a list of tuples, where the i-th tuple contains the i-th element from each of the argument sequences or iterables
- lambda
  - https://python-reference.readthedocs.io/en/latest/docs/operators/lambda.html
  - Returns an anonymous function.

Die Ausgabe sollte wie folgt aussehen:

```
1) 45.20 °C -> 113.3600 °F
2) 36.50 °C -> 97.7000 °F
3) 36.30 °C -> 97.3400 °F
4) 28.00 °C -> 82.4000 °F
5) 37.20 °C -> 98.9600 °F
```

Zusätzliche Fortgeschrittenen Aufgabe: Die Musterlösung beinhaltet einen kleinen Fehler, der nur unter gewissen Konditionen auftritt. Es wurde eine Fahrenheit und eine Celsius Liste implementiert. Falls Sie dies nicht so getan haben, versuchen Sie es. So, nun sollten Sie eine Fahrenheit und eine Celsius Liste haben, Versuchen Sie die Fahrenheit Werte über 100 herauszufiltern. Was für einen Fehler entdecken Sie nun? Beheben Sie diesen ohne eine weitere Zeile hinzuzufügen. Tipp: Sie befinden Sich in der Funktionalen Programmierung, suchen Sie eine weitere Funktion welche ihnen hierbei helfen kann.

Aufgabe C: Quicksort

In dem ersten Aufgabenblatt haben Sie Quicksort implementiert. Falls Sie dies nicht getan haben sollte, ist hier eine Möglichkeit dies zu implementieren:

```
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quicksort(left) + middle + quicksort(right)

print(quicksort([3,6,8,10,1,2,1]))
```

Quicksort ist ein Sortieralgorithmus, der häufig verwendet wird. Quicksort ist ein Divide-and-Conquer-Algorithmus. Er erstellt zwei leere Listen, die Elemente kleiner als der Pivot-Wert und Elemente größer als der Pivot-Wert aufnehmen, und sortiert dann rekursiv die Unterliste. Der Algorithmus besteht aus zwei grundlegenden Operationen: dem Vertauschen von Elementen und der Partitionierung eines Bereichs des Listen.

Hierbei gibt es nun einen Optimierungsansatz, welcher ist das? Beschreiben Sie diesen.

Fortgeschrittenen Aufgabe: Versuchen Sie diesen gefundenen Optimierungsansatz nun zu Implementieren.

Aufgabenteil 2

Aufgabe 1 (reduce, zip, lambda)

1.1 Nutzen Sie die reduce() Funktion, in Verbindung mit einer __lambda__ Funktion, um das Produkt aller Zahlen in dem vorgegebenen Tupel zu berechnen.

Weiterführendes Material: https://python-reference.readthedocs.io/en/latest/docs/functions/reduce.html

```
liste = (2, 4, 5, 1, 6, 7, 4, 3, 7, 9, 4)
```

1.2 Ein Polizeirevier speichert Daten der Polizisten/Polizistinnen in verschiedenen Listen. Nun soll aus verwaltungstechnischen Gründen nur noch ein __Tuple__ zugehöriger Daten je Polizist*in in einer Liste gespeichert werden. z.B. [("Jake", 40, 9544), ("Amy", ...), ...]

Nutzen Sie die zip Funktion, um dies zu erreichen.

```
names = ["Jake", "Amy", "Rosa", "Charles", "Terry", "Hitchcock",
"Scully", "Gina"]
ages = [40, 38, 38, 47, 52, 60, 65, 41]
badge_no = [9544, 3263, 3118, 4262, 3781, 1023, 1893, None]
```

Aufgabe 2 Oscar-Verleihungen (filter, map, list comprehension)

In der Liste "best_actor" sehen Sie Informationen zu den diesjährigen Oscar-Nominierungen für den besten Schauspieler.
Nutzen Sie __lambda__ Funktionen innerhalb der filter/map Funktion.

2.1 Nutzen Sie die filter() Funktion, um eine Liste zu erzeugen, die die Informationen zu Schauspielern, über 50 Jahren beinhaltet

2.2 Lösen Sie 3.1 mit einer List Comprehension

2.3 Nutzen Sie die filter() Funktion, um eine Liste zu erzeugen, die die Informationen zu Schauspielern, die Chris Rock geohrfeigt haben, beinhaltet

2.4 Lösen Sie 3.3 mit einer List Comprehension

2.5 Nutzen Sie die map() Funktion, um eine Liste zu erzeugen, die nur die Namen aller Schauspieler beinhaltet

2.6 Lösen Sie 3.5 mit einer List Comprehension

```
best_actor = [
{"name" : "Benedict Cumberbatch", "age" : 45,
„slapped_Chris_Rock" : False},

{"name" : "Andrew Garfield", "age" : 38, "slapped_Chris_Rock" :
False},

{"name" : "Will Smith", "age" : 53, "slapped_Chris_Rock" : True},

{"name" : "Denzel Washington", "age" : 67, "slapped_Chris_Rock" :
False},

{"name" : "Javier Bardem", "age" : 53, "slapped_Chris_Rock" :
False}]
```

Aufgabe 3 Banknoten (basics, list comprehension)

Aus Philipp G. Freimann (https://www.programmieraufgaben.ch/aufgabe/pruefziffer-auf-euro-banknoten/fy4ohgdx)

Eine Bank bittet Sie, ein Programm zu schreiben, das prüft, ob die Seriennummer auf einer Euro-Banknote stimmt, oder nicht.

Auf allen Banknoten ist eine eindeutige Seriennummer aufgedruckt. Bei Euro-Scheinen haben diese Nummern das folgende Format:

SE8587985563

Dabei sind die ersten zwei Zeichen ein **Druckerei-Code** ("S" z. B. steht für Banca d'Italia, "E" dient der Kennzeichnung der Banknote innerhalb der Ausgaben der Druckerei). Es folgen 9 **Nutzziffern** und zuletzt die **Prüfziffer** (im Beispiel 8).

Die Prüfziffer wird wie folgt berechnet:

1. Schreibe anstelle des Druckerei-Codes die Position der Buchstaben im Alphabet (startend mit 1 für A). Hier 19 für "S" und 5 für "E". Die neue Seriennummer liest sich so **1958587985563**. (Nutzen Sie an dieser Stelle in ihrem Code eine **list comprehension**)

2. Bilde die Quersumme aus allen Stellen (außer der Prüfziffer). Hier also 1+9+5+8+5+8+7+9+8+5+5+6 liefert 76.

3. Bilde den Neunerrest: 76 : 9 = 8 rest 4.

4. Zähle den Rest von sieben ab. Ist das Resultat = 0, so ist die Prüfziffer gleich neun (9). Ist das Resultat = -1, so ist die Prüfziffer gleich acht (8). In allen anderen Fällen ist diese Differenz (7 - Rest) gleich der Prüfziffer. Hier 7 - 4 = 3.

Schreiben Sie ein Programm, das eine Euro-Seriennummer entgegen nimmt und prüft, ob die Seriennummer korrekt ist.

```
geldschein_1 = "FA5011913358" #echt
geldschein_2 = "UA3065954457" #echt
geldschein_3 = "SB8410052723" #blüte
```

Aufgabenteil 3

Aufgabe A:

Schreiben Sie ein Programm, welches eine Textdatei einliest und zählt, wie oft welcher Text-Charakter darin vorkommt (Groß- und Kleinschreibung soll egal sein). Hierbei ist das Counter Objekt aus dem collections Modul hilfreich.
https://docs.python.org/3/library/collections.html#collections.Counter
Sie können es aber auch ohne built-in Methode programmieren.

```
from collections import Counter
import pprint

sample_file = „sample.txt"
```

Aufgabe B:

Die Collatz-Vermutung ist eine mathematische Vermutung, welche 1937 von Lothar Collatz aufgestellt wurde. Diese sagt, dass jede natürliche Zahl im Zyklus 4-2-1 endet, wenn auf ihr folgende simple Rechnungen angewendet werden:

1. Ist n gerade, so wird n halbiert (n -> n/2)
2. Ist n ungerade, so wird n zu 3n+1 (n -> 3n+1)

Dies wird so lange wiederholt bis n 1 erreicht. Ab hier würde sich eine Schleife zwischen 1-4-2-1-4-2-… bilden. Bis heute ist es nicht gelungen, diese Vermutung zu beweisen.

Hier ein Video von „Veritasium“ über die Collatz-Vermutung:
https://www.youtube.com/watch?v=094y1Z2wpJg

Programmieren Sie nun ein Programm, welches für jeden Startwert n zwischen 1 und 100 dokumentiert, was die höchste Zahl, die beim Durchlaufen der oben genannten Schritte erreicht wird. Sortieren Sie danach die Startwert-Maximalwert Paare absteigend nach dem erreichten Maximalwert. Experimentieren Sie ruhig mit den Zahlen etwas herum, indem sie höhere Startwerte ausprobieren. (Bis 1000, 10000, 10000,…)

Aufgabe C:

Schreiben Sie ein Programm, welches überprüft, ob zwei Zahlen Primzahlzwillinge sind oder nicht. Primzahlzwillinge sind zwei Primzahlen, welche die Differenz 2 haben.