Serie 08 - Standardmodule verwenden

> **Programmieren für Naturwissenschaften FS 2023**
> Gruppe: Sofia Kessler, Florian Mohaupt, Lukas Batschelet

Aufgabe 1: Standardmodul `random`

Aufgabe

Gegeben seien die folgenden Anweisungen, welche eine oder mehrere Zufallszahlen generieren:

1. `random.randrange(1, 100, 2)`
2. `random.choice([2, 3, 4, 5, 9, 10])`
3. `random.randint(0, 100)`
4. `random.uniform(50, 200)`
5. `random.sample([11, 12, 13, 14, 15, 16, 17], 3)`
6. `random.random()`

Geben Sie für jede Anweisung das Intervall an, in welchem die Zufallszahlen gewählt werden und welcher Datentyp die Zufallszahlen jeweils aufweisen.

Lösung

1. `random.randrange(1, 100, 2)` generiert eine Zufallszahl zwischen 1 und 100 (exklusive) mit einer Schrittweite von 2 (also nur ungerade Zahlen). Der Datentyp ist `int`.
2. `random.choice([2, 3, 4, 5, 9, 10])` generiert eine Zufallszahl aus der Liste. Der Datentyp ist `int`.
3. `random.randint(0, 100)` generiert eine Zufallszahl zwischen 0 und 100 (inklusive). Der Datentyp ist `int`.
4. `random.uniform(50, 200)` generiert eine Zufallszahl zwischen 50 und 200 (inklusive). Der Datentyp ist `float`.
5. `random.sample([11, 12, 13, 14, 15, 16, 17], 3)` generiert eine Liste mit 3 Zufallszahlen aus der Liste. Der Datentyp ist `list`.
6. `random.random()` generiert eine Zufallszahl zwischen 0 und 1 (exklusive). Der Datentyp ist `float`.

Aufgabe 2: Standardmodul `math`

Aufgabe

Was ist das Resultat der folgenden Ausdrücke?

1. `math.pow(4, 3)`
2. `math.cos(math.pi)`
3. `math.floor(4.5)`
4. `math.ceil(math.sqrt(99))`
5. `math.log(2)`

Lösung

1. `math.pow(4, 3)` ist gleich $4^3 = 64$
2. `math.cos(math.pi)` ist gleich $\cos(\pi) = -1$
3. `math.floor(4.5)` ist gleich $\lfloor 4.5 \rfloor = 4$
4. `math.ceil(math.sqrt(99))` ist gleich $\lceil \sqrt{99} \rceil = 10$
5. `math.log(2)` ist gleich $\ln(2) = 0.6931471805599453$

Aufgabe 3: Standardmodlu `statistics`

Aufgabe

Sehen Sie sich den Datensatz aus der Datei `passagierfrequenz.csv` genauer an. Die Daten stammen von dem Open Data Angebot der SBB und können unter https://data.sbb.ch/pages/home/ heruntergeladen werden. Die Bezeichnungen sind wie folgt definiert:

- DTV = Durchschnittlicher täglicher Verkehr (Montag bis Sonntag) im Jahr 2018.
- DWV = Durchschnittlicher werktäglicher Verkehr (Montag bis Freitag) im Jahr 2018.
- DNWV = Durchschnittlicher nicht-werktäglicher Verkehr (Samstage, Sonntage, Feiertage) im Jahr 2018.

Schreiben Sie ein Programm, welches die Datei einliest. Dann sollen mithilfe des Moduls `statistics` die folgenden statistischen Auswertungen durchgeführt werden:

- Berechnen Sie den arithmetischen Mittelwert von den Werten aus den Spalten `DTV`, `DWV`, `DNWV`.
- Ermitteln Sie den Median der Werte aus den Spalten `DTV`, `DWV`, `DNWV`.
- Finden Sie den Bahnhof mit dem kleinsten und dem grössten Wert `DTV`

Mögliche Lösung

Bemerkung

Das ist sicher einfacher und übersichtlicher zu lösen. Das Programm sollte so aber funktionieren.

```
import statistics

file_name = "passagierfrequenz.csv"

# Datei lesen
with open(file_name, "r") as file:
    content = file.read()
    print(content)

# leere listen für die werte aus der Datei erstellen
dtv_values = []
dwv_values = []
dnwv_values = []
name_values = []

with open(file_path, "r") as file:
    # Die erste Zeile lesen (Header)
    header_line = file.readline()
    # Header-Zeile am Komma aufteilen
    header = header_line.split(';')

    # Den Index der gewünschten Spalten finden
    dtv_index = header.index("DTV")
    dwv_index = header.index("DWV")
    dnwv_index = header.index("DNWV")
    name_index = header.index("Bahnhof_Haltestelle")

    # Zeilen lesen
    for line in file:
        # Zeilen am Komma aufteilen
        row = line.strip().split(';')
        # Werte in die entsprechende Liste einfügen
        dtv_values.append(float(row[dtv_index]))
        dwv_values.append(float(row[dwv_index]))
        dnwv_values.append(float(row[dnwv_index]))
        name_values.append(row[name_index])
```

```
    # Stationen mit maximaler und minimaler Frequenz finden
        # grösste Werte in der Liste suchen
        max_dtv = max(dtv_values)
        min_dtv = min(dtv_values)
        # index dieser Werte suchen
        max_index = dtv_values.index(max_dtv)
        min_index = dtv_values.index(min_dtv)
        # stationsname mit hilfe des index festlegen
        name_max = name_values[max_index]
        name_min = name_values[min_index]

dtv_mean = statistics.mean(dtv_values)
dwv_mean = statistics.mean(dwv_values)
dnwv_mean = statistics.mean(dnwv_values)
dtv_median = statistics.median(dtv_values)
dwv_median = statistics.median(dwv_values)
dnwv_median = statistics.median(dnwv_values)

print("DTV \t= Durchschnittlicher täglicher Verkehr (Montag bis Sonntag)")
print("DWV \t= Durchschnittlicher werktäglicher Verkehr (Montag bis Freitag)")
print("DNWV \t= Durchschnittlicher nicht-werktäglicher Verkehr (Samstage, Sonntage, Feiertage)")
print("Durchschnittswerte:")
print("\t DTV pro Bahnhof (2018): \t", round(dtv_mean, 2))
print("\t DWV pro Bahnhof (2018): \t", round(dwv_mean, 2))
print("\t DNWV pro Bahnhof (2018): \t", round(dnwv_mean, 2))
print("Median:")
print("\t DTV pro Bahnhof (2018): \t", round(dtv_median, 2))
print("\t DWV pro Bahnhof (2018): \t", round(dwv_median, 2))
print("\t DNWV pro Bahnhof (2018): \t", round(dnwv_median, 2))
print(name_max, "ist die Station mit dem höchsten DTV, und zwar: \t", max_dtv)
print(name_min, "ist die Station mit dem geringsten DTV, und zwar: \t", min_dtv)
```