Übungsblatt 4

Abgabe als ZIP-Datei **Deadline**: 27.10.2020 23:55 Uhr

Ablauf der Übung

Deadline: : 27.10.2020 23:55 Uhr

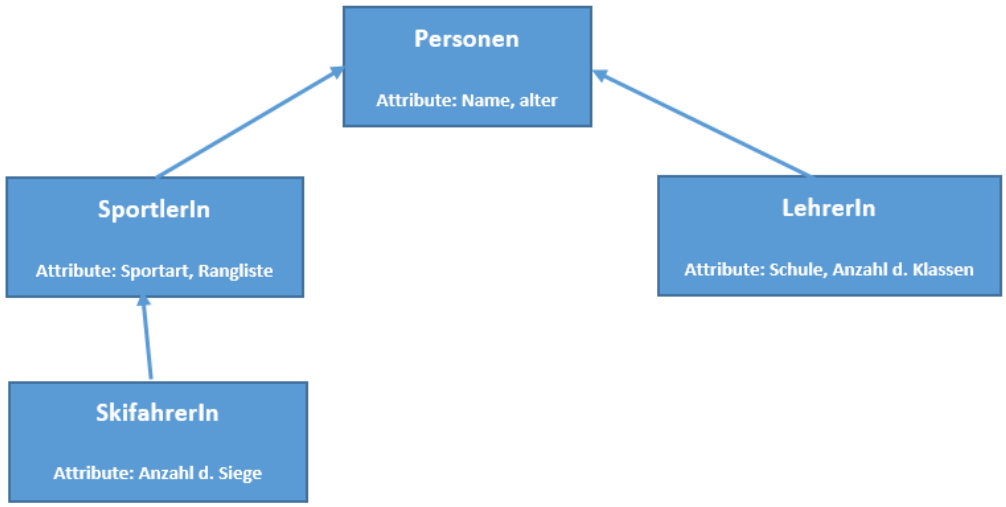
- Die finalen Python Dateien sollen am E-Campus in der Übungsabgabe als ZIP Datei hochgeladen werden.
- Jedes Python Programm enthält in den ersten Zeilen einen Header in dem der Name, Autor, eine kurze Beschreibung enthalten sind.
- Der Code soll sauber geschrieben und wichtige Code-Zeilen kommentiert werden. Der Output sollte ähnlich wie in der Angabe aussehen!

```
"""
<Name>
<Description>
<Mat-Nr>
"""
```

U4Bsp1

Erstellen Sie die Klasse „Person“. Leiten Sie davon 2-Subklassen ab und eine davon soll eine Spezialisierung haben.
Überlegen Sie sich sinnvolle Eigenschaften und Funktionen, die die jeweiligen Klassen besitzen
Welche Eigenschaften/Funktionen werden in der Baseklasse definiert?
Verwenden Sie auch die `__str__` Methode um einen freundlichen Output String des Klassenobjekts auszugeben.
Schreiben Sie eine `main()`, welche neue Objekte der Klassen erstellt und die Funktionen ausgibt
Hinweis: siehe Folien Klassen Vererbung und super()

z.B.:



Übungsblatt 4

Abgabe als ZIP-Datei **Deadline**: 27.10.2020 23:55 Uhr

```
def main():
    tina = SkifahrerIn('Tina', 18, 'skifahren', 2, 15)
    print(tina)
    print(type(tina))

    max = SportlerIn('Max', 20, 'Tennis', 200)
    print(max)
    print(type(max))

    moritz = LehrerIn('Moritz', 45, 'Real Gymnasium', 5)
    print(moritz)
    print(type(moritz))
```

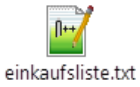
Output:

```
Name: Tina, Alter: 18, Sportart: skifahren, Rangliste: 2, Siege: 15
<class '__main__.SkifahrerIn'>
Name: Max, Alter: 20, Sportart: Tennis, Rangliste: 200
<class '__main__.SportlerIn'>
Name: Moritz, Alter: 45, Schule: Real Gymnasium, 5
<class '__main__.LehrerIn'>
```

U4Bsp2

Schreiben Sie ein Python Programm, das die Häufigkeit von Wörtern einer Textdatei ausgibt
Bsp:

- 10x Apfel
- 3x Birne
- usw.



Das Programm soll zwei Ausgabemodi/Funktionen besitzen:

- Ausgabe im Terminal (print)
- Ausgabe in Textdatei (output.txt)

Verwenden Sie Exception Handling um mögliche Fehler abzufangen
Tipps: Dictionary zum Zählen verwenden oder Counter

```
from collections import Counter
Counter(listElement)
```

Output im Terminal und im File:

```
2x -> Apfel
3x -> Bierdose
2x -> Steak
1x -> Reis
1x -> Birne
8x -> Leberkäsesemmel
```

U4Bsp3

Verwalten Sie die Kontaktdaten einer Firma. Datei: **people.json**
Erstellen Sie eine Klasse Person. Lesen Sie die JSON Datei ein und löschen Sie die gesamte Person, wenn diese ein leeres Attribut hat (z.B.: post code leer oder vname leer) und doppelte Personeneinträge sollen auch gelöscht werden.
Zusätzlich soll der Anfangsbuchstabe aller Attribute großgeschrieben werden (`str.capitalize()`)
Schreiben Sie alle „normalisierten“ Personeneinträge (gültige) in eine Liste (Liste von Klassenobjekten)
Finden Sie heraus wie viele Personeneinträge eingelesen wurden und wie viele nach dem Auslesen vorhanden sind.
Die fertigen Personeneinträge (Ergebnisse der vorigen Aufgaben) sollen ausgegeben werden.

Übungsblatt 4

Abgabe als ZIP-Datei **Deadline**: 27.10.2020 23:55 Uhr

[Optional] Nach „post code“ in aufsteigender Reihenfolge sortieren und ausgeben

Output:

```
Anzahl d. eingelesen Personen: 17
EMPTY ATTRIBUTES Niklas Wild, 2100
DUPLICATE Holly Moller, 6422 Stams
DUPLICATE Philipp Schober, 9470 Grönitz
EMPTY ATTRIBUTES Niki Benz, 5422
EMPTY ATTRIBUTES  Janser, 1090 Alsergrund
DUPLICATE Michael Schöpfer, 3372 Blindenmarkt
Endgültige Anzahl d. Personen: 10

Petra Rosenberger, 1050 Margareten
Manfred Zolles, 1210 Langenzersdorf
Michael Schöpfer, 3372 Blindenmarkt
Siegried Langrock, 3384 Markersdorf
Werner Dutter, 4654 Train
Holly Moller, 6422 Stams
Piper Perrish, 9361 Moserwinkl
Janice Griff, 9361 Moserwinkl
Philipp Schober, 9470 Grönitz
Janice Griff, 9470 Grönitz
```

U4Bsp4

Übergeben Sie ihrem Programm zwei int-Argumente (arg1, arg2) (mittels Programmargumente)
Implementieren Sie die Fehlerbehandlung bei falscher Übergabe (kein int oder zu viele/wenig Argumente)
Lassen Sie eine for-Schleife solange laufen bis der Bereich `arg1 – arg2` durchlaufen wurde
Geben Sie dabei jeden Wert aus.
Hinweis: Verwenden Sie dafür range(arg1, arg2)

Output (bei zu wenig/viele übergebene Argumente):

```
Falsche Parameter Eingabe
<myProgram.py> <arg1> <arg2>
```

Output (mit `python3 U4Bsp4.py 20 25`)

```
20
21
22
23
24
25
```

Übungsblatt 4

Abgabe als ZIP-Datei **Deadline**: 27.10.2020 23:55 Uhr

U4Bsp5

1. Erstellen Sie die Klasse Vehicle mit den Eigenschaften die Sie aus dem CSV Header entnehmen → **vehicle.csv**
    a) Zusätzliche Eigenschaft: **cnt_wins**
2. Für jedes Fahrzeug in der Liste soll eine Klassen Instanz (Objekt) erstellt werden und in eine Liste mit allen Vehicles (Objekte) geschrieben werden.
3. Programmieren Sie ein „Vergleichsspiel“
    a) Es werden jeweils 2 zufällige Fahrzeuge (Objekte) aus der Liste gewählt und ein zufälliges Attribut miteinander verglichen, z.B. mit `__getattribute__(attributsname)`

```
if self.__getattribute__(attr) > other.__getattribute__(attr):
    self.cnt_wins += 1
```

    b) Beim Gewinner (höher oder niedriger, kann auch zufällig bestimmt werden) wird cnt_wins um 1 erhöht
    c) Lassen Sie das Spiel x Mal (z.B. 10.000) durchlaufen und geben Sie alle Instanzen mit z.B. > 40 Siegen aus
    d) [optional] sortieren Sie den Output nach Siegen (bestes Auto zuerst)

Output (kann variieren – wegen Zufallsprinzip):

```
Fahrzeuge mit mehr als 40 Siege:
Name: Chevrolet Tahoe LT || Price: 41465 || Wins: 45
Name: BMW 545iA 4dr || Price: 54995 || Wins: 44
Name: Chrysler Town and Country Limited || Price: 38380 || Wins: 43
Name: Buick LeSabre Limited 4dr || Price: 32245 || Wins: 42
Name: Ford Excursion 6.8 XLT || Price: 41475 || Wins: 42
Name: Infiniti Q45 Luxury 4dr || Price: 52545 || Wins: 42
Name: Pontiac Bonneville GXP 4dr || Price: 35995 || Wins: 42
Name: Lincoln LS V8 Ultimate 4dr || Price: 43495 || Wins: 41
```

Eingelesene Fahrzeuge: 428