



Übungszettel 6b - Python OOP

Falls Sie heute lieber noch Aufgaben von den vorherigen Übungszetteln bearbeiten möchten, dürfen Sie das natürlich auch gerne tun.

Aufgabe 1: Objektorientierte Programmierung

Betrachten Sie folgende Klasse “Student”.

```
1 class Student:
2     'contains students, email, name ...'
3
4     def __init__(self, name, kurse=[]):
5         self.name = name
6         self.kurse = kurse
7         print("Klasseninstanz angelegt für: ", name)
8         return
9
10    def printDetails(self):
11        print("Name:", self.name)
12        print("Kurse: ", self.kurse)
13        return
14
15
16    def einschreiben(self, kurs):
17        self.kurse.append(kurs)
18        return
```

- Füge weitere Attribute hinzu (Telefonnummer, e-mail Adresse,...). Du kannst default-Werte für Parameter angeben, indem du ein Gleichheitszeichen (=) benutzt, wie es im Konstruktor der Klasse “Student” beim Parameter “Kurse” gemacht wurde.
- Schreibe ein Programm, welches eine Studentin namen “Marie” anlegt. Marie ist bereits in den Kurs mit Kursnummer “K374” eingeschrieben, möchte aber noch weitere Kurse besuchen. Gib dem Nutzer also die Möglichkeit weitere Kurse einzutragen. Wenn alle Kurse eingetragen sind, soll das Programm alle Daten von Marie auf dem Bildschirm ausgeben.
- Lege eine Liste von Studenten an. Der Nutzer soll neue Studenten zur Liste hinzufügen können. Das Programm soll den Nutzer nach dem Namen eines Studenten fragen, und dann dessen Daten ausgeben.
- Füge eine Methode creditPoints hinzu, die die Creditpoints eines Studenten berechnet, unter der Annahme, dass jeder Kurs 3CP hat.
- Programmiere eine Klasse “Arbeitnehmer”, die Informationen über Namen, Alter und Position enthält. Welche Methoden und weitere Attribute könnten für diese Klasse nützlich sein?

Solution:

```
(a) class Student:
2     'contains students, email, name ...'
3
```

```

4     def __init__(self, name, kurse=[], phone = 'none given', email='no
      email'):
5         self.name = name
6         self.kurse = kurse
7         self.phone = phone
8         self.email = email
9         print("Klasseninstanz angelegt für: ", name)
10        return
11
12    def printDetails(self):
13        print("Name:",self.name)
14        print("Kurse: ", self.kurse)
15        print("Phone: ", self.phone)
16        print("Email: ", self.email)
17        return
18
19
20    def einschreiben(self, kurs):
21        self.kurse.append(kurs)
22        return
23
24    def creditPoints(self):
25        return(len(self.courses)*3)

```

```

(b) from student import Student
2
3 student1 = Student("Marie", ["K374"])
4
5 print("Geben Sie die Kurse ein in die ", student1.name, "
      eingeschrieben ist. ")
6 newcourse = input("Geben Sie eine Kursnummer oder 'stop' ein: ")
7
8 while newcourse != 'stop':
9     student1.einschreiben(newcourse)
10    newcourse = input("Geben Sie eine Kursnummer oder 'stop' ein: ")
11
12 student1.printDetails()

```

```

(c) from student import Student
2
3 students = []
4 again = "yes"
5
6 while again != "nein":
7     name = input("Geben Sie den Namen des Studierenden ein: ")
8     phone = input("Geben Sie die Telefonnummer des Studierenden ein: ")
9     email = input("Geben Sie die E-mail Adresse des Studierenden ein: "
10    )
11    student = Student(name, [], phone, email)
12    students.append(student)
13
14    print("Geben Sie die Kurse ein, die ", student.name, "belegt.")
15    newcourse = input("Geben Sie eine Kursnummer oder 'stop' ein: ")
16
17    while newcourse != "stop":
18        student.einschreiben(newcourse)
19        newcourse = input("Geben Sie eine Kursnummer oder 'stop' ein: "
20    )
21
22    again = input("Möchten Sie noch mehr Studierende eingeben? ")

```

```

21
22 for student in students:
23     student.printDetails()

```

- (d) Ist oben schon drin
- (e) Sieht bei jedem anders aus.

Aufgabe 2: OOP-Komposition

- (a) Implementiere eine Klasse "Autor", die die drei privaten Attribute `name`, `geschlecht` und `email`, welche im Konstruktor initialisiert werden, besitzt. Es soll auch möglich sein, Autoren ohne E-mail-Adresse anzulegen (Tipp: default-Wert setzen). Ferner soll die Klasse die Methoden `getName()`, `getEmail()`, `setEmail(emailadresse)` und `setGender()`, sowie eine Methode `info()` haben, die den String "AutorName (Geschlecht) at AutorEmail" zurückliefert, besitzen.
- (b) Schreibe ein Programm "testAutor.py" das den Konstruktor und die öffentlichen (public) Methoden der Klasse testet; E-mail Adresse ändern, Namen ausgeben, etc.
- (c) Implementiere eine Klasse "Buch" (die die Autorklasse verwendet), welche die privaten Attribute `titel`, `autor`, `preis` besitzt. Alle drei Attribute sollen im Konstruktor initialisiert werden. `titel` und `autor` sind unveränderbar, der Preis soll sich aber ändern können. Die Klasse soll folgende öffentliche Methoden besitzen: `getName()`, `getAutor()`, `getPreis()`, `setPreis()`, `info()`. Die `info()`-Methode soll einen String "buchName von autorName (geschlecht) at email" zurückgeben. (Beachte, dass die `info()`-Methode der Klasse "Autor" "AutorName (geschlecht) at AutorEmail" liefert).
- (d) Schreibe ein Programm "testBuch.py" um den Konstruktor und die öffentlichen Methoden der Klasse "Buch" zu testen. Beachte, dass du zunächst eine Instanz der Klasse "Autor" anlegen musst.
- (e) Beachte, dass sowohl "Buch" als auch "Autor" ein Attribut `name` besitzt. Sie können durch die referenzierende Instanz unterschieden werden. Für eine Buchinstanz `buch1` z.B. liefert der Aufruf `buch1.name()` den Namen des Buches, während für die Autoreninstanz `autor1`, der Aufruf `autor1.name()` den Namen des Autors liefert.
Versuche die Werte `name` und `email` des Autors von der Buchinstanz aus aufzurufen.
(Hinweis: nutze das Attribut `autor`)
- (f) Erweitere die Klasse "Buch" um die Methoden `getAutorName()`, `getAutorEmail()`, `getAutorGender()`, um `name`, `email` und `geschlecht` des Autors des Buches zurückzugeben.

Solution:

```

(a) #Autor und Buch Klassen
2
3 class Autor:
4     'Autor mit Name, e-mail und Gelschlecht'
5
6     def __init__(self, name, gender, email='no email'):
7         self.__name = name
8         self.__gender = gender
9         self.__email = email
10        print("Autor angelegt: ", name)
11        return
12
13    def setEmail(self,email):
14        'speicher e-mail-Adresse'
15        self.__email = email
16        return
17

```

```

18     def getName(self):
19         return(self.__name)
20
21     def getEmail(self):
22         return(self.__email)
23
24     def getGender(self):
25         return(self.__gender)
26
27     def info(self):
28         return(self.__name+' (' +self.__gender + ') at ' + self.__email
29             )
30 class Buch:
31     'Buch mit Titel, Autor, Preis'
32
33     def __init__(self, name, autor, preis):
34         self.__name = name
35         self.__autor = autor
36         self.__preis = preis
37         print('Buch angelegt: ', name)
38         return
39
40     def getName(self):
41         return(self.__name)
42
43     def getAutor(self):
44         return(self.__autor)
45
46     def getPreis(self):
47         return(self.__preis)
48
49     def setPreis(self, preis):
50         self.__preis = preis
51         return
52
53     def info(self):
54         return(self.__name + ' von ' + self.__autor.info())
55
56     def getAutorName(self):
57         return(self.__autor.getName())
58
59     def getAutorEmail(self):
60         return(self.__autor.getEmail())
61
62     def getAutorGender(self):
63         return(self.__autor.getGender())

```

```

(b) from autor import Autor
2
3 a1 = Autor('Swaroop', 'm')
4
5 print(a1.info())
6 a1.setEmail("swaroop@me.com")
7
8 print(a1.info())
9 print("Der Name des Autors ist: ", a1.getName())
10 print("Der Autor ist: ", a1.getGender())
11 print("Die Email-Adresse lautet: ", a1.getEmail())

```

(c) oben schon drin

```
(d) from autor import Autor, Buch
    2
    3 a1 = Autor('Swaroop', 'm')
    4
    5 print(a1.info())
    6 a1.setEmail("swaroop@me.com")
    7
    8 print(a1.info())
    9 print("Der Name des Autors ist: ", a1.getName())
   10 print("Der Autor ist: ", a1.getGender())
   11 print("Die Email-Adresse lautet: ", a1.getEmail())
   12
   13 b1 = Buch("A byte of Python", a1, 17.90)
   14
   15 print(b1.info())
   16 b1.setPreis(12.95)
   17 print(b1.info())
   18 print("Der Titel des Buches lautet: ", b1.getName())
   19 print("Der Autor des Buches ist: ", b1.getAutor().info()) # Zugriff auf
      Autor über Buch
   20 print("Das Buch kostet: ", b1.getPreis())
   21 print("Der Name des Autors lautet: ", b1.getAutorName())
   22 print("Die E-Mail-Adresse des Autors lautet: ", b1.getAutorEmail())
   23 print("Der Autor ist: ", b1.getAutorGender())
```

Aufgabe 3: Rock, Paper, Scissors

schwierig

“Rock, Paper, Scissors” ist ein bekanntes rundenbasiertes Knobelspiel.

Jeder Spieler wählt zu Beginn einer Runde eines der folgenden Objekte Rock, Paper, Scissors. Wählen beide Spieler das gleiche Objekt, gilt die Runde als unentschieden. Sonst wird der Gewinner nach folgenden Regeln bestimmt:

- Scissors cuts paper
 - Paper covers rock
 - Rock crushes scissors
- (a) Setze dieses Spiel in Python um. Der Nutzer soll über eine textuelle Benutzungsschnittstelle gegen den Computer knobeln können.
- (b) Erweitere das Spiel um die Objekte Lizard, Spock.
Die Regeln zur Bestimmung des Gewinners müssen um die folgenden erweitert werden:
- Lizards posions spock
 - Spock smashes scissors
 - Scissors decapitates lizard
 - Lizard eats paper
 - Paper disproves Spock
 - Spock vaporizes rock
 - Rock crushes lizard

Solution:

- (a) Hier eine beispielhafte Implementierung des Programms für Aufgabe 2:

```

1  #!/usr/bin/env python
2  # encoding: utf-8
3  """
4  RockPaperScissors.py
5
6  Created by Johannes Seitz on 2010-07-30.
7  """
8
9  import random
10
11 VALID_OPTIONS = ("scissors", "paper", "rock", "lizard", "spock")
12 RULES = ["scissors cuts paper",
13          "paper covers rock",
14          "rock crushes lizard",
15          "lizard poisons spock",
16          "spock smashes scissors",
17          "scissors decapitates lizard",
18          "lizard eats paper",
19          "paper disproves spock",
20          "spock vaporizes rock",
21          "rock crushes scissors"]
22
23 def main():
24     """Mainloop"""
25     user_score = computer_score = 0
26     while True:
27         user_choice = input("Choose your weapon: ").lower().strip()
28         if user_choice in VALID_OPTIONS:
29             user_score, computer_score = \
30                 play_game(user_choice, user_score, computer_score)
31         elif user_choice == "exit":
32             exit()
33         elif user_choice == "rules":
34             for rule in RULES:
35                 print("\t%s" %rule.capitalize())
36         else:
37             print("\tType one of the following:\n \
38                 \t%s, %s, %s, %s, %s, rules or exit." %VALID_OPTIONS)
39
40 def play_game(users_choice, user_score, computer_score):
41     """The actual game"""
42     computers_choice = random.choice(VALID_OPTIONS)
43     if computers_choice == users_choice:
44         print("\t%s ties with %s".capitalize() \
45             %(users_choice, computers_choice))
46     else:
47         rule = rule_for_combination(users_choice, computers_choice)
48         print("\t%s" %rule.capitalize())
49         if rule.startswith(computers_choice):
50             computer_score += 1
51             print("\tYOU LOSE!"),
52         else:
53             user_score += 1
54             print("\tYOU WIN!"),
55         print("You: %s, Computer: %s" %(user_score, computer_score))
56     return (user_score, computer_score)
57
58 def rule_for_combination(choice1, choice2):
59     """Looks up the rule for a given combination"""
60     for rule in RULES:

```

```
61     if choice1 in rule and choice2 in rule:
62         return rule
63
64 if __name__ == '__main__':
65     main()
```

Viel Erfolg!