Übungszettel 6b - Python OOP

Falls du heute lieber noch Aufgaben von den vorherigen Übungszetteln bearbeiten möchtest, darfst du das natürlich auch gerne tun.

Aufgabe 1: Objektorientierte Programmierung

Betrachte folgende Klasse "Student".

```python
class Student:
    'contains students, email, name ...'

    def __init__(self, name, kurse=None):
        self.name = name
        if kurse == None:
            self.kurse = []
        else:
            self.kurse = kurse
        print("Klasseninstanz angelegt für: ", name)
        return

    def printDetails(self):
        print("Name:",self.name)
        print("Kurse: ", self.kurse)
        return


    def einschreiben(self, kurs):
        self.kurse.append(kurs)
        return
```

(a) Füge weitere Attribute hinzu (Telefonnummer, e-mail Adresse,...). Du kannst default-Werte für Parameter angeben, indem du ein Gleicheitszeichen (=) benutzt, wie es in der init-Methode der Klasse "Student" beim Parameter "Kurse" gemacht wurde.

(b) 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 den Nutzer*innen also die Möglichkeit weitere Kurse einzutragen. Wenn alle Kurse eingetragen sind, soll das Programm alle Daten von Marie auf dem Bildschirm ausgeben.

(c) Lege eine Liste von Studierenden an. Die Nutzer*innen sollen neue Studierende zur Liste hinzufügen können. Das Programm soll die Nutzer*innen nach dem Namen einer*s Studierenden fragen, und dann die Daten der*s Studierenden ausgeben.

(d) Füge eine Methode creditPoints hinzu, die die Creditpoints eines Studierenden berechnet, unter der Annahme, dass jeder Kurs 3CP hat.

(e) Programmiere eine Klasse "Angestellte", die Informationen über Namen, Alter und Position enthält. Welche Methoden und weitere Attribute könnten für diese Klasse nützlich sein?

Aufgabe 2: OOP-Komposition

(a) Implementiere eine Klasse "Author", die die drei privaten Attribute `name`, `geschlecht` und `email`, welche im Konstruktor initialisiert werden, besitzt. Es soll auch möglich sein, Autor*innen ohne E-mail-Adresse anzulegen (Tipp: default-Wert setzen). Ferner soll die Klas-

se die Methoden `getName(), getEmail(), setEmail(emailadresse)` und `setGender()`, sowie eine Methode `info()` haben, die den String "AuthorName (Geschlecht) at AuthorEmail" zurückliefert, besitzen.

(b) Schreibe ein Programm "testAuthor.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 Authorklasse verwendet), welche die privaten Attribute `titel, author, preis` besitzt. Alle drei Attribute sollen im Konstruktor initialisiert werden. `titel` und `author` 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 "Author" ein Attribut `name` besitzt. Sie können durch die referenzirende Instanz unterschieden werden. Für eine Buchinstanz `buch1` z.B. liefert der Aufruf `buch1.name()` den Namen des Buches, während für die Authorinstanz `author1`, der Aufruf `author1.name()` den Namen der*s Author liefert.
Versuche die Werte `name` und `email` des Author von der Buchinstanz aus aufzurufen.
(Hinweis: nutze das Attribut `autor`)

(f) Erweitere die Klasse "Buch" um die Methoden `getAuthorName(), getAuthorEmail(), getAuthorGender()`, um `name, email` und `geschlecht` des Authors des Buches zurückzugeben.

Aufgabe 3: Rock, Paper, Scissors schwierig

"Rock, Paper, Scissors" ist ein bekanntes rundenbasiertes Knobelspiel.

Jede*r Spielende wählt zu Beginn einer Runde eines der folgenden Objekte Rock, Paper, Scissors. Wählen beide Spielende das gleiche Objekt, gilt die Runde als unentschieden. Sonst wird nach folgenden Regeln bestimmt, wer gewonnen hat:

- Scissors cuts paper
- Paper covers rock
- Rock crushes scissors

(a) Setze dieses Spiel in Python um. Die Nutzer*innen sollen über eine textuelle Benutzungsschnittstelle gegen den Computer knobeln können.

(b) Erweitere das Spiel um die Objekte Lizard, Spock.
Die Gewinnregeln müssen um die folgenden Regeln erweitert werden:

- Lizards posions spock
- Spock smashes scissors
- Scissors decapitates lizard
- Lizard eats paper
- Paper disproves Spock
- Spock vaporizes rock
- Rock crushes lizard

Viel Erfolg!