Informatik I: Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Dr. Daniel Büscher, Hannes Saffrich
Wintersemester 2019

Universität Freiburg
Institut für Informatik

Übungsblatt 7 – Lösungen

Abgabe: Montag, 9.12.2019, 9:00 Uhr morgens

Aufgabe 7.1 (Filme; Datei: `movies.py`; Punkte: 3+3+4+4+4)

(a) Implementieren Sie eine Klasse `Movie`, welche als Attribute den Titel `title` (str), das Erscheinungsjahr `year` (int), sowie die durchschnittliche Nutzerbewertung `rating` (float) besitzt. Beim Erzeugen eines `Movie`-Objekts sollen Titel, Erscheinungsjahr und Bewertung als Argumente übergeben werden. Hierzu ist es notwendig, eine `__init__` Funktion innerhalb der `class`-Anweisung zu definieren (siehe Vorlesung). Beispiel:

```
>>> m = Movie("Groundhog Day", 1993, 8.0)
>>> m.title, m.year, m.rating
('Groundhog Day', 1993, 8.0)
```

Lösung:

```
class Movie:
    """Class representing a movie.

    Attributes:
        title (str): The movie title.
        year (int): Release year.
        rating (float): Average rating, a score between 1 and 10.
    """

    def __init__(self, title, year, rating):
        """Set attributes to their initial values.

        Args:
            title (str): The movie title.
            year (int): Release year.
            rating (float): Average rating, a score between 1 and 10.
        """
        self.title = title
        self.year = year
        self.rating = rating
```

(b) Implementieren Sie eine Funktion `movie_str(movie: Movie) -> str`, welche ein `Movie`-Objekt als Argument erhält und es, analog zur Vorlesung, in einen informativen String umwandelt.

```
>>> movie_str(m)
"Movie('Groundhog Day', 1993, 8.0)"
```

Lösung:

```
def movie_str(movie: Movie) -> str:
    """Return a string representation of a Movie object.

    Args:
        A Movie object.

    Result:
        String representing the movie.
    """
    return "Movie(" + repr(movie.title) + ', ' + repr(
        movie.year) + ', ' + repr(movie.rating) + ")"

# for convenience (not required)
Movie.__str__ = movie_str
```

(c) Implementieren Sie eine Funktion `avg_score(movies: list) -> float`, welche eine Liste von `Movie`-Objekten als Argument erhält und die durschnittliche Bewertung der enthaltenen `Movie`-Objekte zurückgibt. Wir nehmen hierzu an, dass die übergebene Liste mindestens ein Element enthält.

```
>>> movies = [
        Movie("Groundhog Day", 1993, 8.0),
        Movie('The Life Aquatic with Steve Zissou', 2004, 7.3),
        Movie('Tootsie', 1982, 7.4)]
>>> abs(avg_score(movies) - 7.566666) < 1e-4
True
```

Lösung:

```
def avg_score(movies: list) -> float:
    """Calculate the average score of a list of movies.

    Args:
        movies: A list of Movie objects.

    Result:
        The average score.
    """
    res = 0
    for movie in movies:
        res += movie.rating
    return res / len(movies)
```

(d) Implementieren Sie eine Klasse `Actor` zur Repräsentation eines Schauspielers. `Actor`-Objekte besitzen die Attribute: `firstname` (str), `lastname` (str) und `movies` (Liste von `Movie`-Objekten). Beim Erzeugen einer Instanz von `Actor` sollen die Attribute als Argumente übergeben werden.

```
>>> bill = Actor('Bill', 'Murray', [Movie('Tootsie', 1982, 7.4)])
>>> bill.firstname, bill.lastname, len(bill.movies)
('Bill', 'Murray', 1)
```

Lösung:

```
class Actor:
    """Class representing an actor/actress.

    Attributes:
        firstname (str): The actors first name.
        lastname (str): The actors last name.
        movies (list): A list of movies the actor appeared in.
    """

    def __init__(self, firstname, lastname, movies):
        """Set attributes to their initial values.

        Args:
            title (str): The movie title.
            year (int): Release year.
            rating (float): Average rating, a score between 1 and 10.
        """
        self.firstname = firstname
        self.lastname = lastname
        self.movies = movies
```

(e) Implementieren Sie eine Funktion `worst_actor(actors: list) -> Actor`, welche eine Liste von `Actor`-Objekten als Argument erhält und denjenigen Schauspieler zurückgibt, dessen Filme die niedrigste durchschnittliche Bewertung haben. Wir nehmen an, dass die Schauspielerliste, sowie die Filmlisten der einzelnen Schauspieler jeweils mindestens ein Element enthalten. Beispiel:

```
>>> actors = [
        Actor('Bill', 'Murray', [
            Movie('Groundhog Day', 1993, 8.0),
            Movie('The Life Aquatic with Steve Zissou', 2004, 7.3),
            Movie('Tootsie', 1982, 7.4)]),
        Actor('Steven', 'Seagal', [
            Movie('Black Dawn', 2005, 3.9),
            Movie('Exit Wounds', 2001, 5.5),
            Movie('The Patriot', 1998, 4.1)]),
```

```
        Actor('Ben', 'Kingsley', [
            Movie('Sexy Beast', 2000, 7.3),
            Movie('Lucky Number Slevin', 2006, 7.8)])]
>>> worst_actor(actors).firstname
'Steven'
```

Lösung:

```
def worst_actor(actors: list):
    """Find and return the actor with the lowest average movie score.

    Args:
        actors: A list of Actor objects.

    Result:
        The Actor object with the lowest average movie score.
    """
    worst, worstscore = actors[0], avg_score(actors[0].movies)
    for actor in actors:
        score = avg_score(actor.movies)
        if score < worstscore:
            worst = actor
            worstscore = avg_score(actor.movies)
    return worst
```

Aufgabe 7.2 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet07` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, Bezug zur Vorlesung, Interessantes, benötigter Zeitaufwand, etc.).