Exercise Sheet 07 – Sorting and Object Oriented Programming

Sebastian Höffner Aline Vilks

Deadline: Mon, 22 May 2017 08:00 +0200

Submission

By the end of this sheet you will have a number of different files to submit. In Stud.IP you will have a directory for your own group, please upload them there. It is easier for you if you just archive (preferably zip) all files and upload your archive, but it is okay if you upload them one by one.

Exercise 1: Sorting Bond movies

During the lecture we implemented a simple sort algorithm called bubble sort. As your homework, it is your task to change bubble sort to sort some Bond movies. You can find a `bond.csv` file inside the accompanying zip file. It is a simplified list of Wikipedia's List of James Bond films[1].

Create a class `Movie` which models a movie:

- A movie has a release `year`.
- A movie has lead `actor`.
- A movie has a `budget` (a float, in US dollars).
- A movie has a box office `revenue` (a float, in US dollars).
- Implement the `__str__` method such that printing a film shows the title followed by the year and the lead actor in parentheses:

  ```
  From Russia with Love (1963, Sean Connery)
  ```

- Implement a method `income` which calculates the "income" of a movie (while this is technically not correct, we will assume that the income is the

[1] https://en.wikipedia.org/wiki/List_of_James_Bond_films#Box_office_and_budget

difference between the box office revenue and the budget of a movie – also this is not adjusted to account for inflation).

Inside the zip file for this homework you will find an implementation of bubble sort (in `movies.py`). Change it so that it becomes `movie_sort` and sorts a list of movies by their release years.

Bonus: Change `movie_sort` such that you can pass a function `key` which selects the attribute to be sorted on, that is: `def movie_sort(movies, key=get_movie_year)` and sorts the list accordingly. *Hint*: A cool idea for keys is to use lambda functions[2], e.g. `key=lambda movie: movie.year`. You can find some examples in the `bonus` function.

Make all your changes inside the `movies.py` and submit your result.

Exercise 2: Castles crashed... again!

Remember the little knights from exercise sheet 2[3]? Now we can model them much better! Modify the file `knights.py` as explained below.

Define a class `Knight`, where each knight has `level`, `strength`, `magic`, `defense`, and `agility`. Knights have a `normal_attack(self, other)`, a `strong_attack(self, other)`, a `throw_attack(self, other)`, and an `arrow_attack(self, other)`. Each of those attacks deals damage according to the `other` knight's `take_damage(self, attack_damage)` function. For example the `strong_attack(self, other)` looks like this:

```
    def strong_attack(self, other):
        damage = (5 + 1.15 * self.strength + 0.1 * self.level) // 1
        other.take_damage(damage)
```

Remember to add some current `health` to each knight. When the current health is less than or equal to 0, a function `is_alive(self)` should return `False`. When instantiating a knight, it starts at full health (`maximum_health(self)`).

Gather the information about formulae you still need from sheet 2, you may of course also check its example solution.

Your output should look like this:

```
Round: 9
Red alive? True HP: 173.0
Blue alive? False HP: -24.0
```

[2] https://docs.python.org/3/reference/expressions.html#lambda

[3] https://shoeffner.github.io/monty/files/BPP-02_VariablesAssignmentsFunctions-Sheet.pdf