Advanced Programming in Python, Group B
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: JTSK-350112
April 26th, 2018

Assignment 3 - More on Classes, Inheritance and Exceptions

- The problems of this assignment must be solved in Python.
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/350112/2018_1gB/Grading-Criteria-Python.pdf

Problem 3.1 *Compare students* (1 point)

Presence assignment, due by 18:30 h today
Download the file:
https://grader.eecs.jacobs-university.de/courses/350112/python/student.py
Add methods to the `Student` class such that two `Student` objects can be compared. Equality (==) and non-equality (!=) should check for all the data stored in the objects (meaning names, scores, etc.). The other relational operators (<, <=, >, >=) need to compare only the names using the alphabetical order. Write a simple test program called `comp_students.py` to test the correctness of your methods.
Upload two files: `student.py` and `comp_students.py`.

Problem 3.2 *Sort students* (1 point)

Presence assignment, due by 18:30 h today
Modify and extend your solution to **Problem 3.1** such that it creates 10 students and places them in a list. Then the list should be shuffled (use the `shuffle()` method from the `random` module) and printed on the screen. Afterwards use the `sort()` method to order the students in descending order based on their names.
Upload two files: `student.py` and `sort_students.py`.

Problem 3.3 *Pickling objects* (1 point)

Write two programs called `pickling.py` and `unpickling.py`. The first program should create a student object with some properties (chosen by you), then print on the screen student and the size of the object in bytes by using the built-in function `sys.getsizeof()`, then "pickle" the object and write the result into a file called "`object.txt`" and close the file. The second program should open the file "`object.txt`", then "unpickle" the data and print resulting the student object on the screen and close the file. After this, reopen the same file, read the content of the file line by line, print the lines on the screen and sum the sizes of the lines using again `sys.getsizeof()` and finally print the summed size and close the file.
Upload three files: `student.py`, `pickling.py` and `unpickling.py`.

Problem 3.4 *Deadoralive* (1 point)

Download the file:
https://grader.eecs.jacobs-university.de/courses/350112/python/blackjack.py
By using the class `Blackjack` as a source for modification implement the game *Deadoralive*. The game works like in the following: Player and Dealer draw a card, the higher rank wins. If the ranks are equal, the order is given as in `SUITS`: spades, hearts, diamonds and clubs. Count the number of wins and losses for the player and play until the deck is empty. Print the results of the game on the screen.
Upload one file: `deadoralive.py`.

Problem 3.5 *Raise exceptions* (1 point)

Write a function called `something()` which gets an integer as input from the keyboard. If the parameter is `1` then an exception with the value `999` should be raised, if the parameter is `2` then an exception with the value "`Error has happend`" should be raised, if the parameter is `3` then an exception with the value `1.23` should be raised. For raising the exceptions use three different exception classes (written by you) derived from the built-in exception class `Exception`.

Call the function 5 times, provide different inputs and catch the correspondingly raised exceptions and print their values on the screen. Use one `try` block, multiple `except` blocks, an `else` and a `finally` block. In the case of the last two print some meaningful message on the screen.
Upload one file: `raise_exc.py`.

Problem 3.6 *Catch exceptions* (1 point)

Download the file:
https://grader.eecs.jacobs-university.de/courses/350112/python/robustness.py
The program in the file is not very robust. You can easily make it crash. Observe each function and see how to make it fail, or see why it will fail the way it is called. Extend the program with catching exceptions, printing messages and making the functions more robust to erroneous input/data. Consider the following exceptions: `ZeroDivisionError`, `ValueError`, `IndexError`, `TypeError`, `FileNotFoundError` which are all built-in exceptions.
Upload one file: `robustness.py`.

Problem 3.7 *Complex numbers* (2 points)

Write a `Complex` class for operations with complex numbers. You need to implement constructor, overload the operators `+`, `-`, `*`, `/`, `==` and `!=`. Write a program called `test_complex.py` which tests the behavior and operations using at least two `Complex` objects. Print the results on the screen.
Upload two files: `complex.py` which contains the class and `test_complex.py` which tests the behavior.

Addition of two complex numbers:

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

Subtraction of two complex numbers:

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

Multiplication of two complex numbers:

$$(a + bi) * (c + di) = (ac - bd) + (ad + bc)i$$

Division of two complex numbers:

$$\frac{a + bi}{c + di} = \frac{ac + bd}{c^2 + d^2} + \frac{bc - ad}{c^2 + d^2}i$$

How to submit your solutions

Name the programs a3_*x*.py.
Each program **must** include a comment on the top like the following:

```
# JTSK-350112
# a3_1.py
# Firstname Lastname
# myemail@jacobs-university.de
```

You have to submit your solutions via *Grader* at
`https://grader.eecs.jacobs-university.de`.

If there are problems (but only then) you can submit the programs by sending mail to
`k.lipskoch@jacobs-university.de` **with a subject line that starts with `JTSK-350112`**.

Please note, that after the deadline it will not be possible to submit solutions. It is useless to send solutions then by mail, because they will not be accepted.

Your code must compile without any warning under python3.x.

This assignment is due by Tuesday, May 1st, 10:00 h