219114/115 Programming I

Exercise 7: Objected-Oriented Programming (OOP)

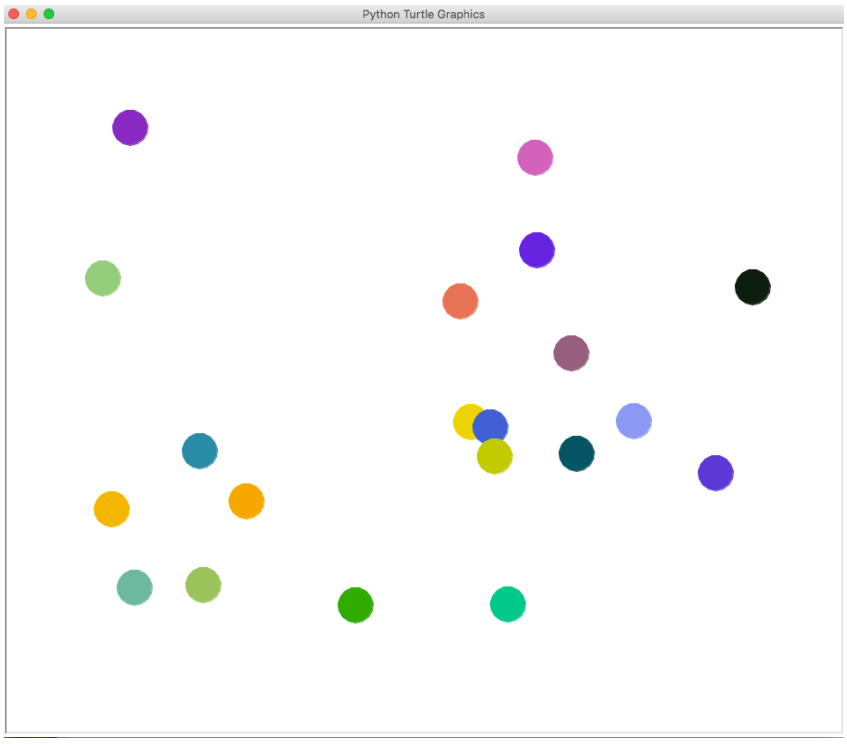
In this exercise, you will get a chance to code in OOP style given the original code written in procedural style. These sample procedural programs should be more appropriately coded in OOP style. You will see that, after you convert them into this style, the code looks more elegant and modular.

Download the zip file from the class website and unzip it to obtain the starting code.

1. Study and run the code given in ball.py and run_ball.py written in procedural style. Note that the code to run is located in run_ball.py and ball.py contains only function definitions. Then, convert it to ball_OO.py and run_ball_OO.py where these programs are written in OOP style. After the conversion, the programs' semantics must remain the same, i.e., the exact same behavior is observed when running either of set of programs written in two different styles.

Here are some guidelines:

- Create a class named Ball
- Provide the __init__ method to initialize each Ball object being created
- Provide the move method to update the Ball position
- Provide the draw method to draw the Ball
- For the code that uses this Ball class, create the number of Ball objects equal to the number of balls specified by a user, and put these objects in a list
- In a while (True) loop, update the state of all the Ball objects in the list



2. Study and run the code given in bank_account.py written in procedural style. Exercise all the choices in the banking system menu and make sure you fully understand all the functionalities of the given code. Convert this code into bank_account_OO.py where it produces the exact same result, but now written on OOP style. You can put class definitions and the running code in this same file.

Submission:

- ***Create StudentID_Firstname_ex7 folder, where StudentID is your KU ID and Firstname is your given name***
- ***Put the files to submit, ball_OO.py, run_ball_OO.py, and bank_account_OO.py into this folder***
- ***Zip the folder and submit the zip file to the course's Google Classroom before the due date***

Grading:

1. Correctness (50%); your code must run and produce correct outcomes; code that does not run because of, for example, syntax errors or name misspelling receives zero credit.

2. Good OOP style (40%); your code must define classes and create appropriate objects that interact to produce the desire result. Class design must be meaningful and fitting for the given problem

3. Cleanliness (10%): your code must be clean, following PEP 8 style guide; variable names must be meaningful, following PEP 8 convention; comments must be put in for others to be able to read and understand your code, again following PEP 8 convention.