CSC148, Lab #1, winter 2018

Introduction

The goals of these lab exercise are:

- To give you practice **designing a class**. Object-oriented analysis was discussed in last week's lectures, and there is an exercise on designing and building a Point class.
- To give you practise **implementing a class**. This will make you review lots of the material from CSC108: creating simple classes, writing methods, and basic data structures in Python.
- To make people practice the function design recipe, especially students who have not used it in a previous courses. The function design recipe was used in class last week. It is a strategy for writing functions that makes them easier to write correctly and leaves behind an excellent docstring.
- Use standard Python style. You should consult CSC108 style guidelines and pep 8. In CSC148, we follow pep 8, and not CSC108, style in preferring to use parentheses for long lines. We also don't leave a blank line between the docstring and the function body (the syntax highlighter effectively distinguishes these elements).

In addition, this lab will make sure that:

- students who have not used the PyCharm IDE get familiar with it, or the IDE they plan to use in this course, and

Don't hesitate to make use of other resources, such as course notes or How to Think Like a Computer Scientist to read up on any topic that you've forgotten about.

Start on these **early** so that you'll have plenty of time to talk to a TA or ask other students if you get stuck, and please be generous in helping others.

Getting started with Pycharm

1. Log in to your teach.cs account.
2. Follow the instructions to configure Pycharm, and create a **csc148** directory structure for your work. Do **not** try to install PyCharm on teach.cs — it's already there, under the menu for **First Year teach.cs**. Although we recommend PyCharm, you are allowed to use any IDE you want in this course. In what remains, we will refer to your IDE, rather than PyCharm.
3. Download the file specs.txt and save it in the `lab1` subdirectory you created above.
4. Open the file `specs.txt` in your IDE. This is the specification for the code you have to write in the rest of this lab.

Designing a class

1. Create a new file called **lab1.py**.
2. Perform an object-oriented analysis of the specifications in specs.txt, following the same recipe we used in class for Point:
   (a) choose a class name and write a brief description in the class docstring.

(b) write some examples of client code that uses your class

(c) decide what services your class should provide as public methods, for each method declare an API[1] (examples, header, type contract, description)

(d) decide which attributes your class should provide without calling a method, list them in the class docstring

This analysis will require a fair amount of thought. Don't worry about getting it completely right: you need to leave enough time to get to the next steps where you will implement your design. If you're stuck, you may look at an example of Point class API.

Implementing a class

1. write the body of special methods `__init__`, `__eq__`, and `__str__`
2. write the body of other methods

If you're stuck, you might look at the code for completed Point class.

Using your class

1. Create a new file named **lab1tester.py** where you will carry out the following tasks, under `if __name__ == "main":`
   - from `lab1` import `NameOfYourClass` (but replace `NameOfYourClass` with the actual name of the class you wrote). You may need to review how to import names of Python objects such as classes.
   - Create a race registry.
   - Register the following runners: gerhard@mail.utoronto.ca (with time under 40 minutes), tom@mail.utoronto.ca (with time under 30 minutes), toni@mail.utoronto.ca (with time under 20 minutes), margot@mail.utoronto.ca (with time under 30 minutes), and gerhard@mail.utoronto.ca (with time under 30 minutes — he's gotten faster).
   - Report the runners in the speed category of under 30 minutes.
2. Run your file to make sure everything works... But don't panic if it doesn't!
3. If you have to, use the rest of your time to debug your code, with the help of other students, and your TA.

Congratulations: you're done with the first lab!

Additional practice

As well as the race registry, here are some additional exercises in designing and implementing classes. We set up each one so that one appropriate solution involves just a single class.

We'll have a brief quiz at the end of the lab, which will involve an exercise similar to the race registry or one of the additional exercises.

How did you do?

Make sure you are off to a solid start in csc148 by identifying any concepts that caused you difficulty and mastering them before the course moves on. Use the resources linked to in this handout, as well as course office hours and the Computer Science Help Centre to get any help you need.

[1] use the CSC108 function design recipe