CSC148, Lab #1, winter 2017

Introduction

The goals of this lab are:

- To give you practise **designing classes**. 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 give everyone practise using the function design recipe, especially students who have not used it in a previous course. 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. In CSC148, we follow pep 8, and not CSC108, style by preferring to use parentheses for long lines rather than "\". We also don't leave a blank line between the docstring and the function body (the syntax highlighter effectively distinguishes these elements).

  Although you will get plenty of warnings from **PyCharm**'s inspector, you can be confident that we will use only **python_ta** to automatically check the style in your code.

In addition, this lab will make sure that:

- students who want to use the PyCharm IDE become familiar with it, or the IDE they plan to use in this course, and
- everyone can successfully submit files on MarkUs, the system where you'll later submit assignments.

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

If you are doing these exercises in a lab period, show your work to your TA frequently, so that they can make sure you are on the right track. Feel free to ask your TA questions — that's why they're here! We also encourage you to ask other students if you get stuck, and please be generous in helping others.

Getting started with Pycharm

1. Log in to your cdf account.
2. Follow the instructions to configure Pycharm, and create a **csc148** directory structure for your work. Do **not** try to install PyCharm on Teaching Labs — it's already there, under the menu for **First Year Teaching labs**. 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.

   We encourage you to use **PyCharm** as your IDE, since it provides a lot of help in producing structurally correct code. Part of the installation instructions for **PyCharm** include the modules **hypothesis** and **python_ta**. If you decide to use another IDE, you will find that these modules are available on the Teaching Labs machines.
3. In your IDE, navigate to **csc148/Labs/lab1**
4. Start the browser and navigate to the lab1 page on the CSC148H1 web site:

http://www.teach.cs.toronto.edu/~csc148h/winter/Labs/lab1

For each lab, you will find handouts and other resources linked from this page.

5. Download the file specs.txt from

   http://www.teach.cs.toronto.edu/~csc148h/winter/Labs/lab1/

   and save it in the `lab1` subdirectory you created above.

6. 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.

Before moving on to the next section, show your work to your TA.

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, or the (much longer) class recipe used for building the Course class

3. Using your Teaching Labs username and password, log on to MarkUs at the following URL:

   https://markus.teach.cs.toronto.edu/csc148-2017-01/

   Find the Lab 1 submission page, and submit your file `lab1.py`. Get help from other students or your TA if you're not sure how to do this or if you run into any problems! You won't be graded on **what** you submit for this lab, only for the fact **that** you submitted.

Before moving on to the next section, show your work to your TA.

[1] use the CSC108 function design recipe

Implementing a class

1. write the body of special methods `__init__`, `__eq__`, and `__str__`
2. write the body of other methods
3. Save your modified file **lab1.py**, then re-submit it on MarkUs. (Get in the habit of submitting your work multiple times to MarkUs — every time that you complete some feature in your code. New submissions simply replace old ones, and in addition, MarkUs keeps track of previous versions so that it is possible to go back to previous versions if needed.) If you are working with a partner, make sure each of you submit at least one version of a file on Markus.

Before moving on to the next section, show your work to your TA. If you're stuck, you might look at the code for the completed Point class.

Before moving on to the next section, show your work to your TA.

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. Submit your file **lab1tester.py** on MarkUs.
3. Run your file to make sure everything works... But don't panic if it doesn't!
4. If you have to, use the rest of your time to debug your code, with the help of other students, and your TA.

Show your work to your TA one last time. 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 certainly don't expect you to do this many exercises in the lab, but they are here for additional practice. 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.