CSC148 Summer 2016, Lab #01

Introduction

The goals of this lab are:

To give you practise designing a class. Object-oriented analysis was discussed in last week's lectures; and there is an exercise on designing and building a Rectangle 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 function design recipe, especially students who have not used it in a previous course. (Note that the function design recipe that we used is slightly revised, compared to what you used in CSC108.) 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.

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 CDF—it's already there, under the menu for First Year CDF. 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. In your IDE, navigate to csc148/Labs/lab01
4. Start the browser and navigate to the lab01 (this document) page on the CSC148H1 web site.

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

5. Download the file specs.txt

and save it in the lab01 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 lab01.py.
2. Perform an object-oriented analysis of the specifications in specs.txt, following the same recipe we used in class for Rectangle:

   (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 (header, type contract, example, 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 Rectangle class API, or the (much longer) class recipe used for building the Course class

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

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 lab01.py.

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

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

Using your class

1. Create a new file named lab01tester.py where you will carry out the following tasks, under

   if __name__ == "main":

from lab01 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.

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.