CSC148, Lab #1
week of September 16th, 2013

To earn your lab mark you must seriously participate in the lab. You are graded on *participation*, not results. Please note that participation includes showing up on time (10 minutes after the hour, at U of T).

You are encouraged to begin work on this lab once you arrive, and not earlier. If you do happen to finish some of it earlier, bring no notes or electronic files, so that you can participate with a fresh perspective during the lab. Here's why:

- The goal of the labs is to help you learn by practicing — think of it as a workout session for your brain! Yes, this is something you could do on your own... But we want stronger students to help out weaker students, and weaker students to have the benefit of discussing material with stronger students.
- We also want everyone to get the benefit of having a TA who can answer your questions. Your TA will be providing help that will get you to learn at least as much, and in less time than it would take on your own.
- Stronger partners learn from weaker partners, by teaching them. Weaker partners learn from stronger partners by asking guidance.
- As a future computer scientist, you'll be working in teams, so now's a good time to start.

Activities for this lab:

1. Preliminaries
   (a) Icebreaker
   (b) Pair Programming Model
2. Programming
   (a) Getting started with Wing
   (b) Designing a class
   (c) Implementing a class

All of these activities should be very fast to complete, except for the last two: you should spend most of your time (at least 3/4 of it) on those last two activities.

1 Preliminaries

Icebreaker

You will be spending lots of hours with your TA and fellow classmates, so you will start by getting to know them. Your TA will get you going on this.

Pair programming model

In every lab, you will be working in the pair programming model. Obviously, two programmers (in rare cases three) are involved, and we call these two people the driver and the navigator. Here are the definitions of the two roles:

driver: The person typing at the keyboard.

navigator: The person (two people, in the rare case of a trio) watching for mistakes, and thinking ahead.

And here is the most important rule for every lab:

- The navigator must **never** touch the keyboard. (Don't make us enforce this by giving a grade of 0 to navigators who are found typing!)

Throughout the lab, you will be switching back and forth between the driver and navigator roles. The instructions will tell you when to do this.

2 Programming

To begin, decide who will start out as driver and who will start out as navigator.

The navigator will read the lab handout and give instructions to the driver about what to do. The navigator is not allowed to carry out the instructions on their own.

The driver The driver will carry out the navigator's instructions. The driver is not allowed to decide what to do on their own.

These restrictions are supposed to make it easy to split up the work between the two roles. But they are not meant to be overly restrictive: you are supposed to discuss what you are doing with your partner throughout the lab!

During the course of the lab, show your work to your TA frequently, and feel free to ask questions — that's why they're here! Don't hesitate to also ask other students if you get stuck, and please be generous in helping others.

Getting started with Wing

Remember, these instructions are meant for the current driver...

- Log in to your account.
- Start Wing by double-clicking on its icon.
- Under your home directory, create a new directory named `csc148`.

- Inside this directory, create another directory named `lab01` (you can make new directories using the `mkdir` command in the terminal, or through the File menu in Wing — create a new file, then right click the window that pops up and create a new folder).
- Start the browser and navigate to the lab01 page on the CSC148H1 web site:

  http://www.cdf.toronto.edu/~heap/148/F13/Labs/lab01

  For each lab, you will find handouts and other resources linked from this page.
- Download the files specs.txt, sample1.txt, and sample2.txt from

  http://www.cdf.toronto.edu/~heap/148/F13/Labs/lab01/

  and save them in the `lab01` subdirectory you created above.
- Open the file `specs.txt` in Wing, using the File/Open... menu item. This is the specification for what you have to do in the rest of this lab.

Before moving on to the next section, **show your work** to your TA and switch driver and navigator roles. You should use **the same** CDF account open on your workstation, since you are working in a pair (perhaps in a trio).

Designing a class

You've remembered to switch driver and navigator roles, right?... OK, once again, these instructions are for the driver.

- Create a new file in Wing, using the File/New menu item.
- Perform an object-oriented analysis of the specifications in specs.txt. (Object-oriented analysis was discussed in last week's lectures; check out How to Think Like a Computer Scientist.
- This will require a fair amount of discussion between the driver and navigator! Don't worry about getting it completely right: the important thing at this point is to come up with some reasonable design (even if it's not "perfect"), so that you can move on to the next steps and have a chance to implement your design. In your new file:
  - For every major noun you obtained from your analysis, write the header for an appropriate class definition, along with a short docstring for the class. (Hint: you can do this with just one new class: you don't need to try to capture every concept using a class; keep it simple.)
  - For every verb you obtained from your analysis, write the header for an appropriate method, along with a short docstring for the method. (Again, keep it simple by doing this only for the most important verbs.)
  - For every adjective or minor noun you obtained from your analysis, in the docstring for your class(es), write a description of the attributes shared by instances of the class.
- Save your file. Name it `lab01.py` (without the quotes around the name, of course). Using your CDF username and password, log on to MarkUs at the following URL:

  https://markus.cdf.toronto.edu/csc148-2013-09/

Find the Lab 1 submission page, and submit your file `lab01.py`. Get help from your partner, other students, or your TA if you're not sure how to do this or if you run into any problems! Please note that you aren't graded on what is actually submitted to MarkUs, we just want to verify that you can indeed make a submission. This guarantees that you know what to do later on, when you need to submit your work on your own.

Before moving on to the next section, **show your work** to your TA and switch driver and navigator roles.

Implementing a class, part 1

Once again, these instructions are for the driver.

- Write the body of the initializer for your main class. Your initializer should take one argument (in addition to self, of course): the name of a text file. Your initializer will read in the words from the file, and store them in some appropriate internal data structure, through one (or more) instance variable(s).
  - This will make you review lots of the material from CSC108: working with files, creating simple classes, writing functions, simple string manipulations, and basic data structures in Python. Don't hesitate to make use of the reference How to Think Like a Computer Scientist to read up on any topic that you've forgotten about.
  - In particular, there is a specific built-in data structure in Python that would be most suited to store the collection of key words along with their frequencies. Figure out what this is (discuss it with your partner and, if necessary, with other students and your TA), before you start to write code for your `__init__` method.
- Save your file, then log on to `https://markus.cdf.toronto.edu/csc148-2013-09` with your account (not the account of the current navigator that you used to submit previously). Find the Lab 1 submission page, and submit your file `lab01.py`. Now both you and your partner have confirmed that you are able to submit work on MarkUs!

Before moving on to the next section, **show your work** to your TA and switch driver and navigator roles.

Implementing a class, part 2

I shouldn't have to remind you at this point, but just in case... These instructions are for the driver!

- Write the body of your method that finds the "top n" words in a collection of key words.
- Save your file, then 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.
- Write the body of your method that compares two collections of key words and that removes from both collections every word that appears in both collections.
- Again, save your file then submit it on MarkUs. At this point, there is one important thing that we have not reminded you to do. Hint: It involves the other files you downloaded earlier... That's right: you should be testing your code!
- Create a new file named `lab01tester.py` where you will carry out the following tasks:

  - from `lab01` import `NameOfYourClass` (where you should replace `NameOfYourClass` with the actual name of the class you wrote, of course)
  - create two instances of your class: one from file `sample1.txt` and the other from file `sample2.txt`
  - call your comparison method between the two objects (it does not matter which one you use as the caller)
  - call your "top n" method with n = 10 on each object and print the results

- Submit your file `lab01tester.py` on MarkUs, to have a record of the work you've done.
- Run your file to make sure everything works... But don't panic if it doesn't!
- If you have to, use the rest of your time to debug your code, with the help of your partner, other students, and your TA.

Show your work to your TA one last time. Congratulations: you're done with the first lab!

A final reminder, because it's important: show your work to your TA frequently, and feel free to ask questions —that's why they're here! Don't hesitate to also ask other students if you get stuck, and please be generous in helping others.