COMP 1005/1405
Summer 2017 - Tutorial #1

Objectives

- Logging in to the SCS undergrad computers
- Practice writing Python code in a text editor
- Practice running Python code from a command line
- Practice creating a zip file and submitting to cuLearn
- Practice basic problem solving

Expectations

- To receive full grades you must write the attendance quiz on time and work on the problems until you are finished or have been checked off by a TA. If you do not have time to complete the entire tutorial before the end of the session, you should finish it afterward for your own benefit.

Submit

For this tutorial, you will submit a single zip file called **tutorial01.zip** to the cuLearn page of the course. The file name must be exactly as described, with no additional characters or different cases and it must be a zip file.

Grading Scheme

Tutorials are 5% of your course grade and will be determined by the work you complete in tutorials 1 to 5. Each of these tutorials will contribute 1% of your final grade. You will receive full marks for each tutorial if you complete the attendance quiz on time, work on the problems throughout the tutorial, and are checked off by a TA before leaving. If you miss the attendance quiz, you will still receive part marks for working on the tutorial and being checked by a TA.

Problem 1 (Lab Accounts and Attendance)

Everyone who is registered in the course has access to the computers in the tutorial room (HP 4155) and other SCS lab rooms. For information about all SCS computer labs see

https://www.scs.carleton.ca/technical-support/computer-laboratories

In order to use these computers, however, you will need to set up your password for your account. If you already have a Carleton University cmail email account you can set up your SCS (School of Computer Science) account from another computer (from anywhere) by following the instructions found at

https://secure.scs.carleton.ca/scs_authentication/newacct-policy-form.php

You can also set up your account when you come to your first tutorial. The TAs will be around to help you with this process.

Once you have an account set up, log on to a computer in the tutorial room (HP 4155). Open a web browser and log into the cuLearn page for **your tutorial section**. Start and complete the attendance quiz for Tutorial 1. You will do this at the start of every tutorial. The tutorial quiz must be completed in the first 15 minutes of the tutorial and it must be done using a PC in the tutorial room. You cannot access the quiz from your laptop or from outside of the tutorial room (you will be given extra time this first tutorial to complete the attendance quiz). Once you have completed the attendance quiz, you can use a lab computer or your own laptop to complete the tutorial assignment.

Problem 2 (Your First Python Program)

A similar process can be used to solve most programming problems. The steps below outline a process that you can apply throughout this course when working to solve programming problems:

1. Analyze the problem
2. Create a plan to solve the problem
3. Design an algorithm (in short, a precise set of steps/actions to solve the problem)
4. Test the algorithm in your head or by tracing through execution on paper
5. Write the Python program code in a text editor
6. Execute and test your Python code/program from a command prompt. Possibly repeat any of the previous steps until the program works correctly

The first program you will complete is a simple program that prints a phrase ("Hello World!") to the console, which is the window in which the user interacts with the program. This program is simple enough that we can largely skip steps 1-4 of the above process. The algorithm for this program is one step: print "Hello World!" to the console.

The next step of the process involves writing the necessary code inside a text editor and saving it as a file on the computer. Open the program called Notepad++, or a similar text editor if you are not using a lab computer (Atom is a good program for Mac users) and set the language to Python using the Language drop-down menu in Notepad++.

Type the following in the text editor (try to **avoid cut & paste** from a PDF file to text file, it generally does not work very well)

```
# my first python program
print("Hello World!")
```

The first line of this code starts with #, which is used to tell Python that you are writing a comment. This means that everything to the right of the # is ignored by the Python interpreter. Comments can be used to explain what a block of code is doing, make notes of your thought process, or include any other information that may be useful in understanding the code.

The second line contains a print command. This command tells the Python interpreter to print whatever text is inside the brackets, which in this case is "Hello World!".

Now that you have written some code in the text editor, you need to save it in a .py file before you can execute it and see the result. If you are using a lab computer, save the file as "t1.py" in your Z-drive. When you click save-as, you will find your Z-drive as one

of the places to save your code. Select the Z-drive and specify the name of the file as t1.py. If you are using your own computer, you can save the file wherever you prefer.

To execute Python programs, we will use the Windows Powershell command prompt program. Open the Windows Powershell program, type **ls** (that is, LS in lower case letters) and hit Enter to see which directory (folder) you are currently in and what files are in that directory. The output of the command should start with something like

```
    Directory: C:\Users\yourname
```

OR

```
    Directory: Z:\
```

You need to change the directory to be the same as the one where you saved your t1.py file. To change directories within Powershell, you use the **`cd`** command. Typing **`cd Z:`** and hitting enter will change the directory to the Z drive. Use the ls command to confirm you have changed to the Z drive and see what files are there. You should see your t1.py file listed – if you do not, make sure you saved it to the directory you are currently in.

Once you are in the same directory as your saved file, type **`python t1.py`** and hit Enter to run your program in the Python interpreter (Mac users will likely have to type **python3 t1.py**). Doing this runs a separate program, called the Python interpreter, and tells it to execute your Python file t1.py. The Python interpreter will then read and execute each of the instructions you have included inside the specified file (i.e., t1.py). If you see “Hello World!” printed in the Powershell window, your program works as expected!

That’s all there is to it! You have now created and run a Python program.

Problem 3 (Submitting in cuLearn)

In this course all assignment submissions, unless otherwise stated, will be a single zip file. You must be extremely careful with the details of your submissions.

For example, when you are asked to submit an assignment called **assignment2.zip**, your submission will lose marks if the filename is not exactly as specified. If you submit Assignment2.zip, assignment02.zip, MyAssignment2.zip, a2.zip, Circle.py, or anything that is not exactly assignment2.zip, you will lose marks. Please note that case matters. A and a are not the same!

When you are asked to submit a zip file, you must submit a zip file. This means that if you use 7zip, rar or any other programs to create a compressed archive of your files you will lose marks. You can create a zip file as follows:

Windows: Select all files you would like in the zip (hold the Ctrl button down to select multiple files). Right-click on the selected files, select **Send to**, and then **Compressed (zipped) folder**. This will create a zip file of all of the files you had selected. You may have to rename this zip file.

OS X: Put the files you want in your zip file in a directory. Right-click on the directory name and select **Compress "DirectoryName"**,where DirectoryName will be the name of the directory you want to create the zip with.

Create a directory called tutorial01 and copy/move your t1.py file to it. There are commands in powershell to let you do this (create a directory and move files) but you can also do this as you normally would (using the file explorer). Enter this directory, select all of the files you want to submit (just t1.py in this case), and create a zip file containing these files as described above. Rename the file so that it is called tutorial01.zip.

Submit your tutorial01.zip file to cuLearn. There will be a submission link in your tutorial section's cuLearn page. After you submit the file, download your submission from cuLearn and confirm that it is a zip file called tutorial01.zip that contains the single file t1.py.

Problem 4 (Input)

In the previous problem, you used the print command to print out text. You can perform 'user input' by reading what the user types into the console. The Python command for reading user input from the console is:

```
input("Prompt the user")
```

Whenever you include this command inside your code, the Python interpreter executing the code will print out the prompt included inside the quotation marks (in this case, 'Prompt the user') and then wait for the user to type something in and hit Enter. Whatever text the user types into the console before hitting enter is 'returned' by this command, a concept we will discuss later. For now, all you need to know is that you can save what the user typed so it can be used later in the program like this:

```
text = input("Prompt the user")
```

This takes the text the user typed and stores it in a 'variable' called *text* (similar to a mathematical expression like x=5). You can think of this as associating the name *text*

with whatever the user typed in. So any time later in your code where you use the name *text*, you will retrieve the value associated with the name *text*. To see how this works, add the following two lines of code to your t1.py file (note: there are no quotation marks around name in the second line):

```
name = input(“Enter your name: ”)
print(name)
```

Save the file and execute it again using Powershell the same way you did for the previous problem. The program should ask you to enter your name and then print your name once you have hit Enter.

Problem 5 (Basic Calculations)

You can also perform mathematical calculations using Python. For example, we could create a conversion program that takes a measurement in pounds and prints the equivalent in kilograms:

```
pounds = 22
kilograms = pounds / 2.2
print(kilograms)
```

Another thing we could calculate is the y coordinate of a line given an x value. The general equation for a line is y = m*x + b, where m is the slope and b is the y-intercept of the line. Add code to your t1.py file that sets values for m, b and x and prints out the associated y value. Try changing the values and checking the correct y value is printed each time.

For a list of mathematical operations that you can use, check out:
http://en.wikibooks.org/wiki/Python_Programming/Basic_Math#Mathematical_Operators

Try adding some additional calculations that use the different mathematical operators to your program.

Problem 6 (Basic Problem Solving and Algorithms)

The following few questions will let you practice some simple problem solving and algorithm development.
1) The sum of 2 consecutive odd numbers is 44. What are the two numbers?
2) Assuming that you did not just guess the numbers exactly correct immediately, write out the steps that you used to solve the problem.
3) Now assume you are given some number *x*. Write out general instructions for finding the 2 consecutive odd numbers that sum to *x* or to determine if no solution exists.
4) Assume you are given a list of numbers [8, 3, 5, 1, 2, 9, 7, 6, 4]. Explain in detail the steps you would take to find the largest number. Remember to be specific about things like what numbers you are considering at each step and what actions you are taking (i.e., be much more specific than 'look through the list and find the largest number')
5) Explain in detail the steps you would use to find the largest number in a general list of *n* numbers. What about finding the second largest number?
6) Explain in detail the process for drawing a house or some other relatively simple object.

Reminders

Save Your Work:

Always save all of your work for all the tutorial problems you solve (or attempt) so that you can show the TAs at the end of the tutorial session (and you can also refer to them later if needed). You can use Dropbox, Google Drive, your Z-drive, email it to yourself, or any other way to make a copy that is external to the PC in the tutorial room. Get used to backing up your work often as you do your assignments.

Tutorial Computers (HP 4155):

Here are some helpful reminders about using the Windows machines in HP4155.

http://carleton.ca/scs/scs-windows/

In particular, be careful where you **save** your data while working on the school machines. Lost data on the school machines, because you did not move your data to your Z drive, Dropbox, Google drive, USB memory stick, etc., before you logged off, will

not be accepted as an excuse for late or missed tutorials/assignments. Please take care that you save your data somewhere that is external to the lab computer.