CPS109 Lab 1

Getting Started in Python

Learning Objectives

This lab is intended to introduce you to the computer lab environment in the CS department at TMU and give you a chance to practice some Python basics.

1. Familiarize yourself with the computer lab environment.
2. Familiarize yourself with the Spyder IDE.
3. Evaluate arithmetic and logical Python expressions in the interactive shell.
4. Write a Python script that solves several more complicated mathematical problems.

Before the lab:

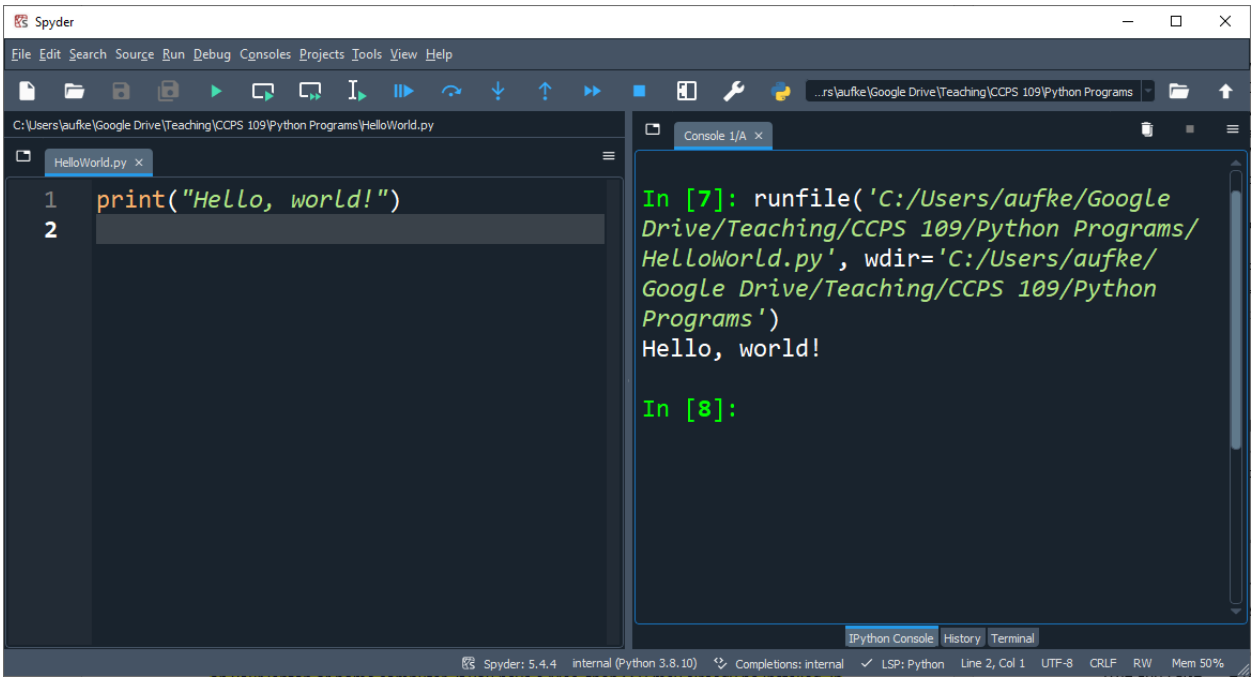
Make sure you know your CS ID and can log into the lab computers. You should have done this in week 1. If you didn't, you can ask your TA for help during the lab this week.

During the lab (warmup):

1) In this first lab, you will get familiar with the computing environment in the Department of Computer Science at TMU. If you haven't already done so, the system administrators and teaching assistants (TAs) will give you your usernames and your default password.
   You will change your password to one that only you know. Please remember this password, as neither your TA nor your professor can help you if you forget it. You'll have to make the walk of shame to the system administrators and ask them to reset it for you.

2) If you haven't already installed an Integrated Development Environment (IDE) for Python 3 on your personal computer, you can download one now (ask your TA if you need help with this). One good and simple IDE, which can be used on Windows, Mac, or Linux, is Spyder: **https://www.spyder-ide.org/**. Spyder does not come with the Python language, so if you haven't installed Python before you'll have to do this as well by going to the following link: **https://www.python.org/downloads/**. Python and Spyder are already installed on the lab computers. If you installed Anaconda, you already have both.

3) Write the simplest program, HelloWorld, using your IDE. To do this in Spyder, after opening the application, you can just start writing a program in the editor window on the left. Save the file, either using File->Save as, or with a keyboard shortcut like Control-S or Control-Shift-S. Click on Run->Run, or press F5, or click the green arrow in the GUI (there are many ways to do everything!). You will see the output in the interactive Shell window. If you did not have a syntax error, then "Hello World" will be printed in the console.



4) In the interactive Shell, try typing the expressions below. The interactive shell if good for evaluating single-line expressions very quickly, without having to write a whole new program. Type the expression, hit enter, and the shell prints the result. Notice that you do not need to call the print function in the shell – it prints automatically. Here are some expressions to try, but you should make up more of your own too. If you do not understand why the output is what it is, you can ask your TA.

1/3	# note that division returns floating point
7/3	# you can even type these comments along with the expression
7//2	# note integer division
33 % 5	# modulus gives the remainder of the integer division
type(10)	# type() is a useful function to check the type of an object
3 ** 9	# exponentiation
type(True)	# bool
True and False	# try other Boolean operators **or** and **not**
True and 5	# just something weird to think about – “truthiness”
5 and True	# more truthiness
True + 5	# also weird, not something you would normally do
a = 7	# assignment to a variable, binds the variable to the object
type(a)	# type of the variable is type of the object to which it is bound
10 + 3 * 2	# note the precedence is what you expect

(10 + 3) * 2	# as usual, parentheses can guide the precedence
'Hi' + 'there'	# + operator is overloaded to be concatenation for strings
3 * 'blue'	# * operator is overloaded to be repetition for strings
'blue' * 3	# repetition from either side
'blue' + 10	# syntax error, cannot concatenate integer to string
print('Blue')	# string using single quotes
print(''Red'')	# string with double quotes
print("O'Henry")	# single quote inside double quotes
print('Blue Red')	# syntax error -- you cannot do a newline inside single or double quotes
print('''Blue Red''')	# triple quotes allows the string to contain newline characters
a = '''Blue Moon'''	# the triple quotes are three single quotes
a	# 'Blue\nMoon' since \n means newline
print(a)	# Notice how the print goes on two lines
a = 'Blue \ Moon'	# Notice the backslash lets you continue on the next line
print('Blue', 'red')	# Notice that the print statement can take any number of arguments # and prints the values of the arguments with a space between them
a, b = 3, 4	# tuple assignment
c = a * b	# arithmetic expression
print('c is', c)	
3 * 6 // 2 == 9	# The == comparison operator checks for equality
3 * 6 // 2.0 == 9	# Is the result what you expect?
3 == 3.000	# Is the result what you expect?
3 * 7 // 2 == 3 * (7 // 2)	# Is the result what you expect?
(11 + 2 // 3) ** 3 / 9 % 5	# using several arithmetic operators, can you guess the output?
5 > 2	# comparison operators
7 >= 5	
2 < 3	
2 <= 2	
4 == (2 ** 2)	
(2 < 3) and (1 > 3) or ((12 < 13) and not (5 >= 7))	# using Boolean operators

5) The following program illustrates binding the value of an expression to a variable by using the '=' operator. Note that changing the value of radius after its use in the assignment to area, does not change the value of area.

```
pi = 3
radius = 2
area = pi * (radius ** 2)
print('area = ', area)
radius = 200
print('area =', area)              # did changing radius change the calculated area?
pi = 3.1415                        # another text example to show that
diameter = 11                      # meaningful names make errors easier to spot
area = pi * (diameter ** 2)        # what is the error in this statement
print('area =', area)

# this is the same program, but the error is not easily spotted
a = 3.1415
b = 11
c = a * (b ** 2)      # make your programs readable by using variable names that are
print('c =', c)       # meaningful.
```

During the lab (to be submitted):

Solve each of the questions below in a single Python file called "cps109_lab1.py". Label each question clearly using a comment. When your file is executed, your answer for each question should execute one at a time, in order.

1. Read in a temperature from the user in Celsius. Calculate and print the temperature in Fahrenheit, and Kelvin. Think about what type seems most appropriate for representing temperature, and why. Read the value using the input() function, but don't forget that input() returns string! You will have to convert accordingly.

```
F = C*(9/5) + 32      K = C + 273.15
```

2. Request three values from the user (once again, choose your types appropriately). These three values represent polynomial coefficients (a, b, c in the equation below). Using these coefficients, evaluate the quadratic formula and print the roots. Notice what happens when the discriminant (b^2-4ac) is negative.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

3. Request three more values from the user. If these three values could represent the sides of a triangle, print True. Otherwise, print False. You do **not** need if/else for this!

   Hint #1: Try and draw a triangle whose side lengths are 1cm, 2cm, and 5cm. Use a ruler.
   Hint #2: The max() function, when given three arguments, will return the largest.

4. Read in a single value from the user. This value represents the side length of a regular pentagon. Calculate and print the area of this pentagon using the side length. Use the formula below, where 'a' is the side length. Notice the nested square roots in the equation.

$$A = \frac{1}{4}\sqrt{5\left(5+2\sqrt{5}\right)}\,a^2$$

5. Read in an integer value from the user, called n. Using the golden ratio (Φ), calculate and print the n^{th} Fibonacci number using the equation below.

$$f(n) = \frac{2+\emptyset}{5}\emptyset^n + \frac{3-\emptyset}{5}\emptyset^{-n} \qquad \emptyset = \frac{\sqrt{5}+1}{2}$$

   Hint: Since Φ appears in several places, perhaps you should calculate it once and store it in a variable. Don't calculate something four times when once is enough!

Submission

You will submit your code for the previous five questions from the "to be submitted" section. You are still expected to try out the examples in the warmup, since they help you get started with the computing environment at TMU, downloading an IDE for Python, and trying out the interactive shell. However, there is nothing to submit for the warmup section.

For this lab, and all future labs, you will be submitting your code on D2L. Submit your Python file, cps109_lab1.py, to D2L under Assessment->Assignments->Lab #1. If you're having any trouble accessing or submitting to D2L, please ask your TA for assistance.