CSCI 141
Fall 2019
Lab 1: Introduction to Python and the Thonny IDE
Due Date: Friday, October 4th at 9:59pm

Lab and Homework Assignments

The purpose of this lab is meant to introduce you to Python and Thonny. In lecture, we've talked about what programming is about and seen a few examples of very short Python programs. In this lab, you'll expand on upon that example, and get some hands-on experience in writing and running your own Python program.

You should have enough time to complete this lab during the lab session. If you do not, be sure to upload your submission to Canvas by the due date. If you have questions, be sure to ask the TA. Ask questions often. Labs are your opportunity to get personalized help!

Collaboration in Lab

You are encouraged to work together with your peers in completing the labs; you can discuss problems with a lab partner, or even work together as you learn how to write Python code. In lab, you are allowed to look at other students' code and discuss solutions with your peers. **However, the Python submission that you upload to Canvas must be your own.** If by the time you're done, you could not sit down and write the code you've submitted by yourself without help from anyone else, you have both violated academic honesty and done your future self a disservice.

Notation

There are two basic fonts in use in this document (this is common to many other help documents and computer books you will use). One is the font that I have used so far in this document and then there is `this font` which is a fixed width font that I will use to denote exact things you should type, exact names of menus to use, website addresses and so on. If you have questions about this notation ask your TA for clarification.

Log into your CS Account

You should have activated your CS account at the following URL prior to coming to lab: http://password.cs.wwu.edu. If you have not done so, please have the TA help you access this address and activate your account.

If for some reason it doesn't look like your lab computer is on the Windows login screen, please refer to Section 5 of this lab for instructions on how to log into a specific operating system on a lab computer. Ask your TA for help as needed.

Log in using your CS account credentials. If only an image appears on your screen, click one of the mouse buttons to view the login prompt. The first time that you log into your Windows CS account you may experience a wait time of a few minutes while your account files are created.

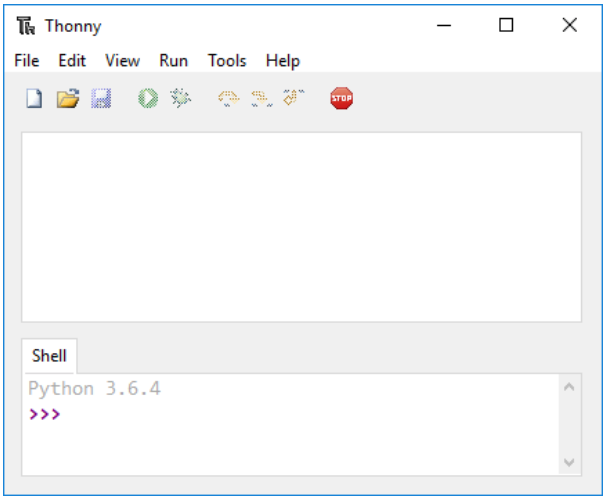

Figure 1: The Thonny IDE

Create Folders

You should get into the habit of organizing and placing your files into folders. Also, it is a good idea to save everything to your N drive, which is accessible on any CS lab computer you log into. Create a folder called csci141 in your N drive, and inside that make another folder named lab1. Save all of your work in that location. Your N drive is for you to use as you like.

To access your N drive from a CS lab Windows machine, launch the File Explorer Desktop app. Under Network Locations, you'll see your N drive. Double mouse click to expand the N drive, where you can create folders.

1 Getting Started with Thonny

The lab machines all have Thonny installed already. If you want to use it on your own computer, you can download it from http://www.thonny.org. Once you've logged into Windows on a lab machine, search for and start Thonny using the search box at the bottom left. If there are multiple versions, use the most recent version that is installed. If you've launched Thonny correctly, a screen similar to what is shown in Figure 1 should appear. Don't worry if the version number after "Python" in the Shell tab differs; as long as the major (first) version number is 3, you're in good shape.

There are two different ways that Thonny can be used. You can use the interpreter (or shell) directly, which will cause each line of code that you type into the Shell section to be executed after you press return, or you can create and save a python program file, and then run (execute) the file in its entirety. Because you'll be submitting your python program via Canvas, all instructions in labs and homework assignments will ask you to save a python program file.

Select New from the File menu, which will begin a new file. Then select Save As from the file menu, and save your file as *lastName_firstName_lab1.py* in your lab1 folder.

As a first step, you'll recreate the quintessential Hello World program that we saw in class. An important of programming well is placing comments throughout your code to document your work. Comment lines among python code are ignored and not executed when the program is run. To insert a comment, begin the line with a hash (or pound) symbol, #. You saw in lecture how to use the `print` function to output text to the console.

2 Hello World

Type into your editor window of python (edit your file) the following (use your name and the current date):

```
# Author: Scott Wehrwein
# Date: September 26, 2019
# Description: Code for CSCI 141 Lab 1
print("Hello World")
```

Notice how the IDE colors different parts your code differently. Comments are grey, and the words *Hello World*, which are enclosed in double quotes, are colored green, to specify that those words are a string literal. You'll learn more about strings later in the course, but now just think of it as text: the double quotes tell Python that what's inside them shouldn't be interpreted as more code, but as a piece of data representing some text.

That's it! Your first python program. It contains comments (which will be ignored by the python interpreter), and a single use of the `print` function, which will print the phrase Hello World to the console.

Save your program to your lab1 folder (File -¿ Save as..., Ctrl+S), then run the program by either selecting Run Current Script from the Run menu, click on the green circle with the right-pointing arrow, or press the F5 button on your keyboard. If your code has no syntax errors, you should get something that is similar to the what is shown in Figure 2.

Figure 2: Hello World Output

3 Input, len, and variables

One important feature of most programs is that they somehow interact with the user and allow the user to input data. The `input` function in Python is one way to accomplish this: the program to prompts the user and then waits for some input to be entered using the keyboard.

You saw in lecture that we can call `input` function with no arguments to effectively pause the program until the user presses enter, as in

```
input()
```

If we provide a string as an argument to `input`, it will print the string as a prompt before waiting for input.

Once the user provides input, you'll often want the program to to store whatever the user entered somewhere in memory so that you can refer to that data when needed. Programming languages rely on variables as place holders that "remember" where a piece of data is stored in the computer. The concept of a variable will be discussed extensively in lecture. For the time being, think of a variable as an easy-to-remember name for a piece of data. To set the value of a variable use the assignment operator, `=`.

You've learned the basic use of two functions so far: `input` and `print`. Here's another one: the `len` function, which is short for *length*, calculates the length (number of characters) of whatever is inside the parentheses to the right of `len`.

Add to your python program the below lines of code. Type the text exactly—double quotes, commas, parentheses, line breaks, etc. – all of it:

```
name = input("What is your name?")
name_length = len(name)
print("Hello", name)
print("You have", name_length, "characters in your name.")
```

When executed, the four lines of code print to the console *What is your name?*, and then wait for the user's input. Once the user provides input and presses return (enter), then the user's input is placed into the variable `name`. The `len` function is invoked to calculate the length of the data in the variable `name`. The output of the `len` function is placed into the variable `name_length`. The second-to-last `print` function causes the program to print the phrase *Hello* followed by the value that is stored in the variable `name`. Lastly, the count of characters in the user's name (as calculated by the `len` function) is also output to the console.

Run your program. If your code has no syntax errors, you should get something that is similar to what is in Figure 3.

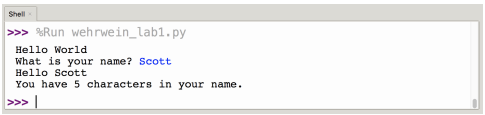

Figure 3: Example output for the full program.

4 Errors

Learning how to code means being able to find and fix syntax errors. Moreover, even if your python code has correct syntax, there may be other errors that may cause the program to crash. Knowing how to read error messages is an important skill because they inform you where in the code there is an error.

In this section, you'll introduce some syntax errors on purpose to get familiar with what you'll see when something goes wrong. Intentionally change `print` to `printt`, or `len` to `LEN`, or omit the parentheses around *What is your name* in the call to the `print` function. Any of these will generate an error when the program is run.

Save and run your program, and look at the red error message. A sample output is provided in (Figure 4). Assuming you had mistyped your code unintentionally, what information does the error message provide that you can use to troubleshoot your code? Notice that the error message tells you what line of code has the problem. Moreover the python interpreter tells you what exactly on a specific line of code it could not understand.

Play around with small modifications to your code and get a feel for the different error messages. There are many, many rules about what is valid Python syntax. You will see lots of examples of correct Python code (and you'll encounter your fair share of incorrect code, as well!), but many questions are best answered by trying things out. Get creative about adding or removing spaces and newlines, changing capitalization, and so on.

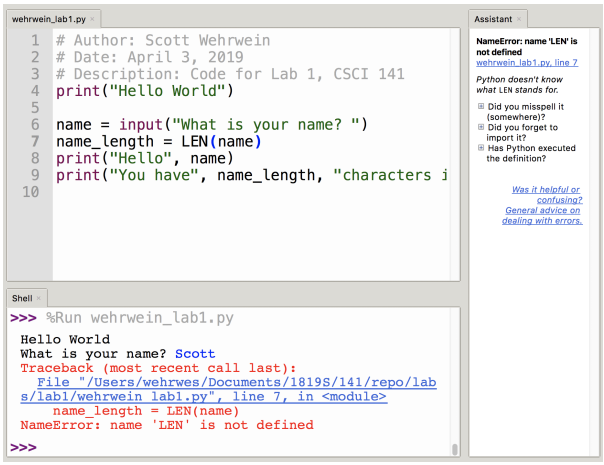

Figure 4: Sample Error Message output by python

When you get an error message, pay close attention to how to interpret what the error messages say about *where* the error is coming from. This will come in very useful later, when you need to locate an error you've made unintentionally! You'll also notice that Thonny provides some helpful suggestions for locating the error in the Assistant panel on the right.

Submission

That's it for this week's lab—don't worry if this seemed simple: future labs will be more involved.

Before you submit, undo the errors in your code so that your python program runs correctly. Save your .py file, and upload it to Canvas. For this lab, Canvas has been configured to permit only .py submissions. If you have never used Canvas, log into your canvas account, proceed to CSCI141, and you should see an option for items that are open for submission. Click on Lab 1, and select to upload a file.

5 Windows and Linux on the Lab Computers

The computers in the CS building are dual boot machines. That means that when the computer boots up, you have the option to launch and log into a Windows account, or a Linux account. This lab was done in Windows, while the next one must be done using Linux. After that, you will be able to choose whichever one you are most comfortable with—both the Linux and Windows accounts on the CS computers have Thonny installed.

To switch operating systems, you must reboot the computer. While the computer boots, you will see a screen listing a bunch of choices; one will say Ubuntu (that's the version of Linux installed on the machines), while another will say Windows. You can choose which you'd like using the arrow keys and press enter to select it.

Rubric

Top of program file contains comments, including your name	3 points
The program does not contain any syntax errors and runs as intended	2 points
The program uses `input`, `len`, and `print`	5 points
Total	10 points