In this lab you will create a program that does simple input and output.

Basic steps for creating a new program

- Type the program text in the Python editor window
- Save your program in a file, giving the file name a .py suffix.
- Test your program by running it
- If you have mistakes, fix them in the editor and re-run the program
- Add comments to your programs -- at least your name, date, and a general description of what the program does.

Getting Ready

- If you haven't already, create a CSC103 folder for saving your course work. You might want to create a subfolder for lab exercises.
- Start up IDLE and get to a new file window.

What should your program do?

- PART 1
- Create a program called yourname_lab2.py
- For all the following bits of part 1, write code in your python file, AND draw boxes on paper that represent the contents of the variables. You will be handing this paper with boxes in! Make sure your name is on it.
- If you replace the contents of the box with something else, for now SHOW me the original box with a line through it, and the new box with the new contents. Practice setting out boxes in legible ways - so it's easy to keep track of what value is in what variable.
- Remember that there are CONSTRUCTOR FUNCTIONS that change things from one data type to another. We've met int(), which makes integers, and float() that makes floats. There is also a string constructor, str(), that makes strings out of things.
- Answer any question below in COMMENTS in your code. Should there be errors, tell me what they are!

1. Write code that:
    - creates a variable called cat, and assigns it the integer 7
    - creates a variable called mouse, and assigns it the float 2.7
    - creates a variable called giraffe, and assigns it the string mouse
    - creates a variable called zebra, and assigns to it the contents of the variable cat
    - assigns the contents of the variable cat to now be 28
    - make a new variable called hippo, and assign it the value 2.3
    - now create a new variable called sausage, make it contain the results of applying the int() constructor to the variable mouse.
    - create a new variable called burger, make it contain the results of applying the string constructor to the variable hippo

    - create a new variable called balloon and make it contain the contents of the variable hippo
    - Print out the contents of each variable. NICELY.
2. Answer the following three questions in comments in your code. Be as specific as possible.
    - what if anything happens when you apply the string constructor (str) to the variable cat? That is, what do you see, and what actually happens to the contents of the variable cat.
    - what if anything happens when I apply the int constructor to the variable balloon? What do you see, and what happens to the contents of the variable.
    - what if anything happens when I apply the float constructor to the variable mouse?

REMEMBER you can try things out in the command window, to see what happens.

PART TWO

You do NOT need to do box tracing for the following parts (although of course it's useful if you run into problems)

1. In the same program file, write code that prompts the user to enter their name (HINT: Remember the input function) and then print a greeting, and then their name (HINT: How do you print a string, and then a variable?)
Sample Execution:
Please enter your name: nick
Hello nick
Another Sample Execution:
Please enter your name: Jillian
Hello Jillian
2. ADD to your program from above, so it also asks for a number.
Do NOT delete the original code, just add new code, so I should see the output from above, AND the output required below.
Print the name of the person the number of times the user enters.
Sample Execution:
Please enter your name: nick
Please enter a number: 3
Hello nicknicknick
3. ADD to your program so that it asks for first and then asks for last name, and creates a new variable called full, that is the *concatenation* of the two names. Make sure you add a space into the string. A space is just another string, with a space in it, like this " ".

Again, add this in to your existing code, making sure I can STILL see the previous output.

PART THREE

- Add new code to your file. This is separate from all the code above. Now, prompt the user to enter two numbers, using TWO calls to the input function, one after the other.
- Print the two numbers, their sum, difference, and product.

Sample Execution:

Please enter one number: 4
Please enter a second number: 6
The first number is 4
The second number is 6
Their sum is 10
Their difference is -2
Their product is 24

Another Sample Execution:

Please enter a number: 17
Please enter another number: 22
The first number is 17
The second number is 22
Their sum is 39
Their difference is -5
Their product is 374

- Now add new code that asks the user to enter FOUR numbers, *one at a time*
- Add up these numbers and print the result
- Print the average of the four numbers
- HINT: How many variables do you need?

PART FOUR

- From NEXUS download the code story.py (it should be at the bottom of this lab). Save it into your CSC103 folder. By default downloads end up in your Download folder. Move it from there to your CSC103 folder.
- RUN the code. You should see errors.
- EDIT the code, so that it works. This means you need to assign SOME variables with SOME values. Read the code to see if you can figure out HOW many variables you need to create, and WHAT THEY ARE CALLED.
- DO NOT EDIT ANY OF THE PRINT STATEMENTS IN THE CODE!
- Do this until the code works. When it works, it should generate a magnificent story. You get to choose the contents of the variables, but NOT the names of the variables.

- Write down in comments ANYTHING you notice about the output that makes it less than ideal - anything about punctuation, for example.
- Uncomment (delete the # symbols at the start of the line) the two lines of code beneath Part 2, and add a comment symbol (the #) before the print line AFTER Part 1. Re-run the code. You should see similar, but slightly different output. How is it different? Is it better? What about the code itself, which is 'better'? (i.e. easier to read, or write?)
- For the final part, comment out where you assign the variables values. DO NOT DELETE THEM - I WANT TO SEE THE VALUES YOU CHOSE.
- Instead, let's make this an interactive story. Write code using input commands, to ask the user to define their own values to use in this story. Ask the user appropriate questions, use input to store the values in the right variables. Test your code to make sure it works.

PART FIVE

In your original python file (yourname_lab2.py), use comments to write me the instructions to make a peanut butter and jelly sandwich.
You may assume that I live in a regular house, with a regular kitchen, and that I have all the necessary ingredients.
Write the instructions ONE per line...in the right order.
At the end, in a comment, tell me how many instructions it took.

TO HAND IN:

- ALL of your code, in order, into a single file for upload, plus my story.py - so TWO files in total. Add comments to make it clear which code answers which questions. Save this file as a .py file.

- Upload this both .py files to nexus

- Include the diagram sheet with boxes from part 1. Take a picture with your phone and upload that picture to nexus.

- IF you have not finished by the END of LAB - you will have until 10pm tonight to upload your code