LAB - PART 2

TOPICS: first program, output, data types, operators, keyboard input, branching, looping

5 First program, output

- Create a program that outputs a recipe by writing it on the computer screen.
- The goal of this exercise is to make sure that you know how to perform the basic operations in Python within the Spyder development environment.
- Your program should be saved in a Python file.
- Your output should have two sections and be easy to read:

1. ingredients
2. instructions

- The two sections should be clearly labelled.
- Each instruction should be numbered.
- Each instruction should be described simply and should contain only one thing to do.

If you are copying the recipe from a cookbook, this may mean that you have to divide some of their steps into multiple instructions for the purpose of this exercise.

A Python program consists of variables (ingredients) and actions (instructions), and each action must be described clearly and simply -- keep this in mind when you create your recipe.

- Here's an example:

```
Recipe for Boiled Water
-----------------------

Ingredients
-----------
2 C water
1   kettle

Instructions
------------
1.  Pour the water into the kettle.
2.  Set the kettle on the stove.
3.  Turn the stove on "high".
4.  It is done when the water is bubbling and steaming, so when that happens, turn off
the stove.
```

6 Storing data

Copy and run the following program:

```
myname = 'Elizabeth'
print 'Hello' + myname
```

The symbol myname is a *variable*. It's value, in this example, is "Elizabeth." In the print statement, the value of the variable is *concatenated* to the *constant* or *String literal* "hello".

- Modify the program as follows:

1. Replace my name with your first name.
2. Define a second variable and assign its value to a String containing your surname.
3. Modify the print statement to print both of your names.

7 Keyboard input

- Programs tend to be much more interesting if users can interact with them. Users can provide **input** to programs through text-based interfaces or graphical interfaces. To keep things simple for now, we will use a text-based interface enabled through the Spyder console window. With various Python extensions, one can build graphical interfaces that open windows, include pull-down menus, slider bars, etc.

The code below provides an example of how to read one line of input:

```
yourname = raw_input( 'what is your name? ' )
print( 'hello ' + yourname )
```

- Create a new file and copy this code into the file using the Spyder editor. Run it. In the Spyder console window, you'll be asked to enter your name. The program should output "hello" addressed to you!

- Modify this program as follows:

1. Change the prompt so that the program asks for the user's first name.
   Note that you may also want to change the name of the variable to indicate that it will contain the user's first name only.
2. Add another input statement that asks the user for their surname and stores it in another variable.
3. Modify the print statement so that it displays a greeting to the user, addressing her by her first name and her surname.

- Create a new file and copy the code below into the file using the Spyder editor:

```
n = raw_input( 'enter a number: ' )
print( 'you entered ' + n )
n3 = n * 3
print( 'three times your number = ' + n3 )
```

What do you think this program will do? Run the code. **Does it do what you expected?**

- When you use the raw_input() built-in function to read data from the keyboard, **the input is treated as String data** -- even if the user enters digits. So, in order to perform arithmetic on the input, you need to *convert* the String data into a numeric form. You can use the int() built-in function to convert from String to integers or the float() built-in function to convert to floating point (real) numbers. For example:

```
1   x = '7'
2   print( 'x = ' + x )
3   y = int( x )
4   y = y * 3
5   print( 'y = ' )
6   print( y )
7
```

- Modify the program at the bottom of the previous page so that it behaves as expected, i.e., it performs arithmetic.

- You may have noticed in the listing above that the final output is split across two print statements (lines 5 and 6), unlike the listing on the previous page (line 4) where the output is done with one print statement. This is because you cannot mix String and numeric input using the concatenate (+) operator. (You may have found this yourself when modifying the code from the previous page.)

You can convert numeric input to String format using the str() built-in function. So, the last two lines from the listing above can be combined into:

```
print( 'y = ' + str( y ))
```

If necessary, modify your program so that the final output is produced with one print statement, as above.