Workshop Prep Materials

C-START Python PD Workshop 2017

This document outlines the materials to be completed before the first day of the program. Please complete the materials before attending, making note of any questions or interesting thoughts that arise.

If you have any questions or hit any roadblocks while completing these materials, please feel free to contact Jack Rosenthal (`jrosenth@mines.edu`).

1 Accessing Python Online

For the Python workshop, we will be using the online Python interface located at the URL below:

`https://repl.it/languages/python3`

You will need to complete these materials using this interface. If you create a free account, you can save your files online for later reference.

Type in the Python code below into the left pane of the interface, exactly as it is shown (omitting the line number).

```
1 print("Hello, World!")
```

Don't worry, we will explain what all of this code means later. For now, click the **Run** ▶ button to run the code. If all went well, your interface should look like the screenshot in Figure 1.

If you've made it this far without any issues, then congrats! You've just written your first Python program. In the next section, we will examine what makes this program work.

2 Examining the `Hello, World!` Program

Let's take a look at the different components of your program:

1. `print` is a **function**. Similar to a function in mathematics (such as $f(x) = x^2$), we use parentheses to evaluate the function at a value.

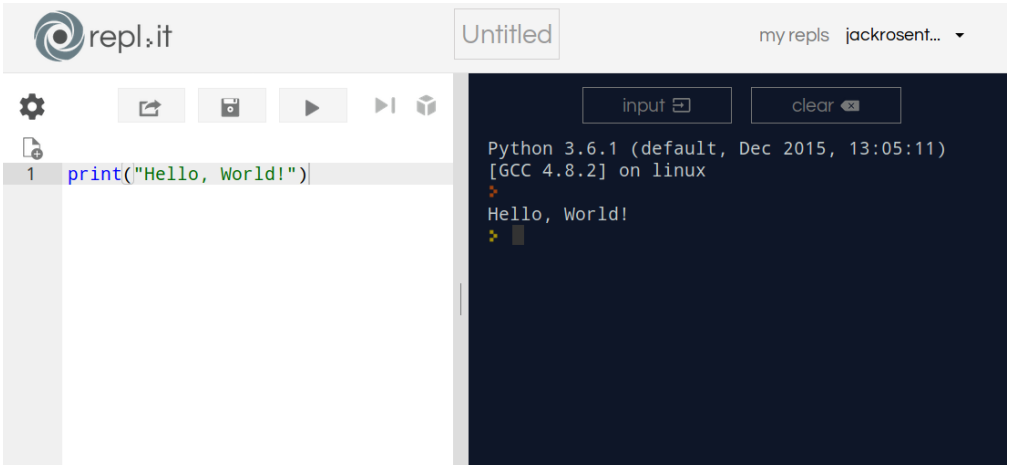

Figure 1: Running your "Hello, World!" program

2. `"Hello, World!"` is a **string**. To type a string in our Python program, we put quotation marks (") around the characters we are encapsulating in a string. A string is a data type, we will learn how to store data later on.
3. When we combine these two together, we print the string `"Hello, World!"` to our console.

3 Expressions

Python provides most the common operators you may use in a mathematical expression. Table 1 shows the common operators and their meaning.

Table 1: Common Operators in Python

+	Addition	/	Division
-	Subtraction	//	Integer Division
*	Multiplicaton	%	Modulus (division remainder)
**	Exponentation		

If you type an expression into Python's Interactive Interpreter (the right pane of the interface with the > prompt), it will evaluate the expression and show the result. This is shown in Figure 2. Try it in your own Interactive Interpreter: enter a few expressions and notice the results.

When we say **evaluate**, we mean simplify the expression to a single result following order of mathematical operations.

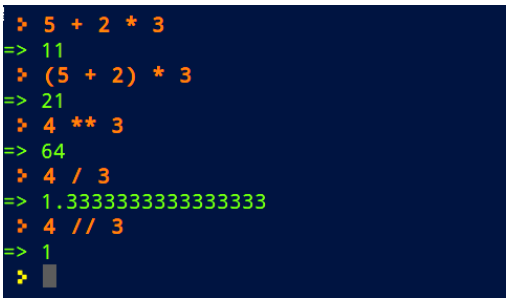

Figure 2: Entering Expressions into the Interactive Interpreter

Notice that we've used two more data types here: integers and floating point numbers: the former being whole-valued numbers, and the latter being numbers storing decimal places.

4 Variables

Variables are a way to store data for future use under a certain name. When we write a statement with an equals symbol (=), we *evaluate* the value on the right of the equals symbol and *assign* it to the variable name on the left. For example, consider the Python program below:

```
1 x = 12 + 1
2 y = x - 3
3 x = 15
4 print(x)
5 print(y)
```

Here's a breakdown of the program, line by line:

1. `12 + 1` evaluates to `13`, and that value is stored under the name `x`
2. `x - 3` evaluates to `13 - 3`, which evaluates to `10`, which is stored under the name `y`.
3. `x` is overwritten with the value `15`. It's important to note that this line does **not** change the value of `y`. Python only remembers the final evaluated result for variables, not the equation we used to get there.
4. Prints `15`, the current value of `x`.
5. Prints `10`, the current value of `y`.

4.1 Variable Names

You can use as many letters as you would like in your variable names, not just the `x` and `y` used in the previous example. You may even use digits (0 through 9) and underscores (_) in your variable names, *as long as your variable name does not start with a digit*. You may not use spaces in your variable names.

Variable names are case-sensitive; so `Jack` and `jack` are separate variables.

There are some names you cannot use for your variables, such as `in`, `if` or `for`. These words have a special meaning in Python; you'll learn how to make use of them during our workshop.

Practice: Valid Variable Names

Which of the following would be valid variable names?

1. `dogcow`
2. `clarus_the_dogcow`
3. `Clarus the Dogcow`
4. `8ball`
5. `_8ball`
6. `if`

Answer: 1, 2, and 5 are valid. 3 is not valid since it contains spaces, 4 is not valid since it starts with a digit, and 6 is not valid since it is a reserved word for Python.

5 More on Strings

Variables can hold more than just integers: they can hold strings as well. In fact, variables can hold any type of data. Consider the example below:

```
1 life = "42"
2 universe = "answer"
3 print(universe + life)
```

Notice there is something new: we are using + on two strings. When we use the + operator on strings, it will evaluate to the string with each string joined end-to-end. So in this example, `answer42` will be printed to the console.

Suppose we wanted to convert a string to an integer so we could do math on it. We can use the function `int` to do so. An example is below:

```
1 mynumber = 12
2 yournumber = "14"
3 result = int(yournumber) + mynumber
4 print(result)
```

Now, type in the example above, but change the `result` line to read:

```
result = yournumber + mynumber
```

Python will give you an error. What error does it give you? What do you think it is trying to tell you? Why do you think the designers of Python wouldn't let you add a string to an integer?

6 Collecting User Input

Suppose you've written a very simple Python program which greets you when you run it. For the author of this document, it would look like this:

```
1 print("Greetings, Jack!")
```

You loved your program so much that you wanted it to greet anybody who walked up to it, not just yourself. Python has a function called `input` which will help you complete this task. The function will ask the user a question, using a prompt message provided; then, after the user responds, the function will return what the user typed. You could use this in your program like so:

```
1 name = input("What is your name? ")
2 print("Greetings, " + name + "!")
```

You are guaranteed that `input` will always return a string. If you wanted the user to type a number, you would have to convert it using `int`. See the example below:

```
1 age = int(input("How many years old are you? "))
2 print("You must have been born in one of the following years:")
3 print(2017 - age)
4 print(2017 - age - 1)
```

Type in this program and run it. Was one of the years correct?

After doing so, modify the first line so it reads:

```
age = input("How many years old are you? ")
```

What error did you get running it? Again, why do you think Python gave you this error?

7 Project: BMI Calculator

Congrats. You've learned the basics of Python programming. Now you get the chance to write a program of your own.

Your program should:

1. Ask the user for their weight in pounds. Store this as a variable[1], make sure to convert to an integer. You may assume that the user will only enter a whole-valued number of pounds.
2. Ask the user for their height in inches. Store this as a different variable. You may also assume that they will only enter a whole-valued number of inches.
3. Convert their weight from pounds to kilograms. You can do so by multiplying by 0.4539.
4. Convert their height from inches to meters. You can do so by multiplying by 0.0254.
5. Calculate their BMI by taking their weight in kilograms and dividing it by their height in meters squared.
6. Print your BMI calculation.

Run your program and calculate your own BMI. Did your program calculate your BMI correctly?

[1] Having trouble deciding on a name? Naming your variable something that makes sense in context would be a good choice. For example, `lbs` or `weight_pounds` might be a good choice here.