**Hello World**

You know how we do! Gotta start it off right with the Hello World!

**Examples**

**Hello World!**

**Code Comments**

Like every single programming language I am aware of, Python has a way for you to comment your code by marking text in such a way that it hides the text from the interpreter.

Python, however, does not inherently have a way to make multi-line comments like other languages. Below I will teach you a way to work around that limitation.

**Examples**

**Single Line Comment**

Single line comments simply start with a hashtag.

**Multi-Line Comment**

Multi-line comments can be created by wrapping your text in a set of triple quotation marks. As you will see, some IDEs like to automatically pair your quotes so creating just three can be something of a challenge.

**Variables**

Variables in Python are fun.

Variables are data in programs that either change on the fly or are used to represent some value that you set once, and then use the variable in your code in place of that value.

Clearly variables are what make your program work.

Python is a strong and dynamically typed language. That is computer science talk for a language where the data types of variables do not change without explicit conversion (strongly typed), and that data type is decided at runtime (dynamically typed).

That is all well and good, but what all that really means is that you do not have to specify a data type for a variable when you declare it.

However, you do have to assign a variable a value when you create one. That initial variable definition is what decides the data type of the variable.

**Examples**

**Simple Variable Declaration**

Here we create a variable and assign it a value.

**Using A Variable To Simplify Code**

It is often the case that variables hold values that can be a bit unwieldy like file paths or URLs. In this case, we create a variable and assign it the unwieldy value so later in our code, we can use the simpler representation of the value.

**Changing The Value Of A Variable**

Here is an example of changing the value of a variable.

**String Concatenation**

One way you can change the value of a variable if it is a string value is to take it and smoosh it together with another string. This is called string concatenation and it comes in handy when you need to do things like build file paths.

**Smoosh!**

When you assign a string value to a variable, you let Python know it is a string by surrounding it in quotes. You can use single quotes or double quotes. Just be sure to be consistent. As you can see below, spaces are signified by a space in between two quotes.

**Building File Paths**

At some point you are probably going to need to move files around on a machine. Some Python methods require that you specify a file name while others just need a directory.

In either case, you want to build a string out of your file path in such a way that you get maximum flexibility out of the whole string. You do that by breaking up the file path into things that change and things that stay the same.

In the variables tutorial, you saw the sophisticated way to do this. I am going to show you a more brute force approach.

See how we break the string down so it is three component parts? In the real world, you will frequently write processes where the file name will change as you loop through a directory. I have simulated that here by reassigning the variable file\_name to a new value.

Also notice how I have written the double backslashes. That is because the backslash has special meaning in Python. The backslash is used to denote that a special character is coming like \n which represents the ASCII linefeed value. The backslash escapes the n so the Python interpreter knows that \n means LF and not just the literal string ‘\n’.

Since backslash is the escape character, putting another backslash in front of a backslash “escapes” the first backslash and now the Python interpreter will know to treat the initial backslash like any other string.

**Arithmetic Operators**

The arithmetic operators in Python should be just as familiar to you as they were in grade school. However, there are three unusual operators that we need to talk about real quick.

The first strange operator is integer division which is performed by a double forward slash between the numerator and denominator. Integer division rounds down to the nearest integer value.

Modulo, which is performed by the percent sign between the numerator and denominator, returns the remainder value from a division operation.

Exponent you should be familiar with. The only odd thing about it is exponent operations are usually performed by ^ in other languages. In this case, it is performed with two asterisks.

I want to point out two things. The first is the print statements.

Do you see how each variable is preceded by str? That is because those variables are either integers or floats. If we want to smoosh a number together with a string, we have to cast the number as a string and that is what the str method does.

The second thing is the value of the division operation. If you are sharp you will see that we divided two integer values. Despite the fact that the result was an integer, the data type of division resulted in a float.