3.1 Lesson Summary - Let's Begin Python

**Python** is a powerful, widely used, and versatile programming language. Its elegance, portability, and relative simplicity have led to its use in a variety of computational activities. It’s used to power major companies’ web servers, analyze the human genome, run microcomputers like the Raspberry Pi and everything in between. More relevant to our purposes Python offers an excellent set of tools for collecting, processing, analyzing, visualizing, and sharing data. Python is a cornerstone of contemporary data analytics.

Concept: In order for Python to run it requires an interpreter application to convert the code into instructions the computer can understand. We run our interpreter by using a text base command line style application either **git-bash** on Windows machines or **terminal** on Macs. Both of these applications are generally referred to as **Terminal**. Terminal works by allowing the user to enter commands and displaying text output. Because of their simplicity and versatility terminal provides an ideal environment for running our Python interpreter and other activities like creating and copying files. Common tasks in terminal can be automated by running a sequence of commands.

* Activity: 01-Ins\_Terminal, 02-Stu\_TerminalTest
* Suppl link: [Basic Terminal Commands](https://github.com/coding-boot-camp/DataViz-Lesson-Plans/blob/master/01-Lesson-Plans/03-Python/1/Activities/01-Ins_Terminal/Solved/CommonCommands.txt)

Concept: Python is an awesome programming language but it becomes even more powerful when paired with tools to assist in writing code, analyzing, and visualizing data. **Anaconda** combines Python and the most commonly used data analytics tools into one easy to install package. Bundled with Anaconda is the **conda** package manager which manages the software packages used by Python. To check that you have a version of Anaconda installed you can run the following code: *conda --version*

Concept: Python often relies on a number of other software packages. If the correct version of these packages is not installed there can be conflicts with other packages or code behavior can differ between package versions. conda makes sure there will be no conflicts between packages and offers **Virtual Environments** to ensure that only a certain set of packages are available within that environment.

You can create a conda environment named “PythonData” using the following code:

*conda create -n PythonData python=3.6 anaconda*

To activate the PythonData virtual environment you can run the following code:

*source activite PythonData*

Concept: Like any other programming language **Python** allows you to store data in **Variables**. Unlike VBA Python doesn’t require you to declare the type of your variable, it automatically detects it. To create a variable with the string “Hello World” you could use the following code:

*hello\_world\_variable = “Hello World”*

To have your code display text in the terminal you can use the following:

*print(“Display This Text Please”)*

* Activity: 03-Ins\_Variables, 04-Stu\_HelloVariableWorld

Concept: Displaying data in the terminal is very useful and you also have the ability to read in text of the user types in the terminal. To store a user’s **input** into a variable you could use the following code:

*favorite\_integer = int(input(“What is your favorite number?”))*

* Activity: 05-Ins\_Prompts, 06-Stu\_DownToInput

Concept: **Conditional** if-then statements allow you to control the logical flow of your code. To display a message if the variable test\_var is true you could use the following code:

*If test\_var:*

*print(“That variable is true!”)*

* Activity: 07-Ins\_Conditionals, 08-Stu\_ConditionalConundrum

Concept: Data can be grouped together in Python using a **List**. To create a List of numbers you could use the following code:

*my\_favorite\_numbers = [1, 7, 22, 9]*

To add to the list *my\_favorite\_numbers* you could use the following code:

*my\_favorite\_numbers.append(4)*

To print the 2nd entry in a list you could use the following code:

*print(my\_favorite\_numbers[2])*

Being able to add or subtract from the list slows down the processing of list data. A **tuple** is a list that you cannot add or subtract from once it has been created making it faster to create and access in most cases. You could create a tuple with the following code:

*my\_numbers\_tuple = (1, 7, 22, 9)*

* Activity: 09-Ins\_List, 10-Stu\_RockPaperScissors

Concept: To perform some actions multiple times in Python you can use a **loop**. A **for** loop runs through a specified sequence. To run a loop that prints the numbers 0 through 5 you could use the following code:

*For x in range(5):*

*print(x)*

To loop code until a condition is satisfied you can use a **while** loop. To run the loop until a user inputs yes you could use the following code:

*while run == "y":*

*run = input("To run again. Enter 'y': ")*

* Activity: 11-Ins\_Loops, 12-Stu\_NumberChain

Python is an extremely powerful tool to use for data analytics. In this course mastering Python goes hand in hand with mastering data analytics.