## Instruction Schedule – Based on Part - Time Cohort

• 60 minutes: welcome and content read

• 60 minutes: lecture and demo lessons

• 30 minutes: work on challenges

• 30 minutes: walk through challenge solutions and wrap up

## Instruction Schedule – Based on Full - Time Cohort

### Morning session:

• 60 minutes: welcome and content read

• 60 minutes: lecture and demo lessons

• 30 minutes: work on challenges

• 30 minutes: walk through challenge solutions and wrap up

Break – 30 mins

### Afternoon session:

• 60 minutes: recap and content read

• 60 minutes: lecture and demo lessons

• 30 minutes: work on challenges

• 30 minutes: walk through challenge solutions and wrap up

## PYTHON Instructor Guide Introduction

Welcome to Python! The Python workshop will introduce students to the field of data analytics and the basics of Python. Your role as the instructor is critical to the student learning experience. You will be meeting with students in a synchronous environment three times a week and should be prepared with the teaching goals, strategies, and structure for each class. This is where the instructor guide comes in! Each synchronous class covers one module, and you will be provided an instructor guide for every module. The modules are further subdivided into lessons. The guide is meant to offer you the goals and key considerations for each module as whole, but also each lesson. We hope this will help and make you feel supported in your very important role.

### Considerations to Keep in Mind

* Some students may come with no experience in Python whereas others might already be proficient.
* There is a workbook for the entire Python workshop. Encourage students to use the workbook to take notes.
* Each class will start with a lesson video that will introduce key concepts and examples for the designated modules and how data analysts use Python.
* Each module will end with a challenge activity.

# Module 1 – Instructor guide

## Introduction to Python

### Module Learning Outcomes

In this module students will,

### 1. Define Python as a programming language.

### 2. Identify the different variable types.

### 3. Define key terminology.

### 4. Describe what Python can do.

### 5. Apply basic code in Python.

### Module Overview Description

This module will introduce students to the basics of Python programming language, key terminology, and fundamentals. The module wraps up with a Challenge activity to review what you learned by answering several foundational questions and creating Python code.

### Lesson 1: Welcome to Python

* Describe a few things that Python can do with your data.
* Explain what sets Python apart from SQL and other programming languages.
* Define the following key terminology of Python.
  + Python
  + Syntax
  + Variable
  + Expression
  + Script
  + Float
  + String
  + Print
  + Boolean
  + Loop
  + Command Line
  + Identifier
  + Value
  + Describe how Python relies on indentation.
  + Describe the difference in how double and single quotes are used in Python.

### Lesson 2: Variable Types

* Walk students through and guide them through the following:
  + Identify the types of variables that will be most common to you as an analyst.
  + Describe the parts of a variable.
  + Describe case sensitivity in Python.
  + Identify the rules to follow to declare descriptive variables.
  + Provide examples of acceptable variable names.
  + Define casting.

### Lesson 3: Multiple Variables

* Build the student’s knowledge on the following areas:
  + Write code for assigning values to more than one variable at a time.
  + Write code that assigns the same value to more than one variable on the same line.
  + Define tuple.
  + Identify the steps involved in creating a list in Python and some important tips.
  + Write code for a list in Python.

### Lesson 4: Output Variables

* Build the student’s knowledge on the following areas:
  + Write code that combines both text and variables in Python.
  + Explain the method behind the above code.
  + Identify what happens when you try to combine a string and a numeric in Python.

# Module 2 – Instructor guide

## Python and Basic Operations

### Module Learning Outcomes

In this module students will,

### 1. Define Python statements.

### 2. Identify the different data types.

### 3. Define the different number data types.

### 4. Demonstrate what operators can do.

### 5. Apply basic operations.

### Module Overview Description

### This module will get students familiar with Python statements, data types, numbers, and operators. The module wraps up with a Challenge activity to review what you learned by answering several foundational questions and creating Python code.

### Lesson 1: Statements

* Build student’s knowledge and guide them through the following:
  + Define assignment statement.
  + Define multi-line statements.
  + Write code for the following;
    - Multiple statements surrounded by parenthesis.
    - Multiple statements surrounded by brackets.
    - Multiple statements surrounded by curly brackets.
    - Multiple statements surrounded by semicolons.
  + Describe why comments are valuable in Python and the symbol that is used to begin a comment.
  + Write code for a comment in Python.

### Lesson 2: Data Types

* Build student’s knowledge and guide them through the following:
  + Define constructor function.
  + Define the following data types and provide examples using code.
    - Text type:
    - Numeric:
    - Sequence:
    - Mapping:
    - Set:
    - Boolean:
  + Describe when parenthesis are used in Python versus when brackets are used in Python.

### Lesson 3: Numbers

* Guide students through how to define the following numeric types and write code in Python:
  + Float
  + Integer
  + Complex
  + Write code for defining number variables and values, using the type () function.
  + Write code that combines the type() function with output variables using print statement and declare variables.
  + Describe what it means to convert numbers from one to another using the type() function.
  + Use the code your previously wrote to change your numbers to be converted. Also include comments to indicate what is being done.

### Lesson 4: Operators

* Build student’s knowledge and guide them through the following:
  + Identify the arithmetic operators in Python.
  + Write an example of how you would apply mathematical operators in python.
  + Define logical operators in Python.
  + Write an example of how you would apply logical operators in Python.
  + Define identify operators.
  + Write an example of how you apply identity operators in Python.

This module has the [*Python Fundamentals*](https://content.bridgepointeducation.com/curriculum/file/da053006-be45-488b-85a0-2e03a367c5a0/1/Python%20Fundamentals.zip/story.html)interactive available for students to interact with and test their newly acquired skills. This interactive contains digital flashcards with basic terminology and syntax examples. There is also a challenge that requires completing syntax arguments.

# Module 3 – Instructor guide

**Python Simple Calculations and Collections**

### Module Learning Outcomes

In this module students will,

### 1. Define Python calculations using string and integers.

### 2. Identify different types of data collections.

### 3. Describe what data collections can do.

### 4. Apply basic calculations.

### Module Overview Description

This module will get students familiar with how to use Python as a calculator and identify different types of data collections. The module wraps up with a Challenge activity to review what they have learned by performing calculations creating data collections using Python code.

### Lesson 1: Simple Calculations Using Integers

* Walk students through the following areas:

### Identify what expression syntax includes.

### Write code for the following:

### Addition

### Subtraction

### Division

### Multiplication

### Power calculation

### Identify the data type that a division produces in Python.

### Lesson 2: Simple Calculations Using Strings

* Walk students through the following areas:
  + Identify how to access string elements in Python.
  + Define string literals.
  + Identify how strings are concatenated.
  + Identify how to add a space between two words in a string.
  + Write example code for combining a literal and a variable.
  + Define slicing and give an example.
  + Define indexing and give an example.
  + Identify how to find the length of a variable and provide an example.​

### Lesson 3: Lists

* Build student’s knowledge and guide them through the following:

### Identify the types of data that can be contained in a list.

### Identify how lists are indicated in Python and write an example.

### Identify how strings are indicated in Python and write an example.

### Write example code for calling a list with a specific index number.

### Identify what happens when you call a list with an index number that is out of range.

### Write example code to concatenate string items at their index number using the + operator.

### Write example code to delete Items in a list using the remove() method.

### Write example code to add an item at the end of a list using the append() method.

### Write example code to add an item at the end of a list using the insert() method.

### Write example code to determine how many items a list holds using the len() function.

### Lesson 4: Python Collections

* Build student’s knowledge and guide them through the following:
  + Define the following collection data types in Python.
    - Lists
    - Tuples
    - Sets
    - Dictionaries
  + Define the following keywords for data collection type.
    - Ordered
    - Unordered
    - Unchangeable
    - Allow duplicates
    - No duplicates
  + Describe the difference between tuples and lists.
  + Write example code for a tuple in Python.
  + Identify how sets are indicated in Python.
  + Describe how dictionaries store data values and how they are indicated in Python.
  + Write example code for adding another key to an existing dictionary.

# Module 4 – Instructor guide

**Python Control Flow and Statements**

### Module Learning Outcomes

In this module students will,

### 1. Define control flow statements.

### 2. Identify the different types of control flow statements.

### 3. Identify the purpose of loops.

### 4. Recognize what functions and arguments can do.

### 5. Apply function, argument, and loop code in Python.

### Module Overview Description

This module will get students familiar with control flow statements, loops, functions, and arguments. The module wraps up with a Challenge activity to review what they have learned by answering foundational questions and creating Python code.

### Lesson 1: IF ELSE ELIF Statements

* Build student’s knowledge and guide them through the following:

### Identify the logical condition symbols that Python recognizes.

### Write sample code for an IF statement in Python.

### Identify what happens if an indentation is not used with an IF statement.

### If an expression is false and does not return the specified condition, what statement is returned?

### Write sample code for an ELIF statement in Python.

### Write sample code that contains IF, ELIF, and ELSE statements.

### Lesson 2: AND OR Statements

* Build student’s knowledge and guide them through the following:

### Define the following statements:

### AND

### OR

### NOT

* + Write example code using the NOT statement.
  + Write example code using the AND statement.
  + Write example code using the OR statement.

### Lesson 3: LOOP Statements

* Walk students through how to:
  + Define a loop.
  + Define a while loop statement.
  + Write example while loop code.

### Lesson 4: Functions and Arguments

* Walk students through how to:
  + Define functions.
  + Define user defined functions.
  + Write sample function syntax.
  + Identify the proper components that should be included when creating and writing functions.
  + Define arguments and how to write them.

# Module 5 - Instructor guide

### Python Anonymous Functions, Classes and Objects

### Module Learning Outcomes

In this module students will,

### 1. Define key terminology.

### 2. Identify the different types of classes and objects.

### 3. Identify the purpose of modules.

### 4. Describe how to recall scopes.

### 5. Apply dir() code in Python.

### Module Overview Description

This module will get students familiar with lambda functions, classes, objects, scopes, and working with modules. The module wraps up with a Challenge activity to review what they have learned by answering foundational questions and creating Python code.

### Lesson 1: IF ELSE ELIF Statements

* Build student’s knowledge and guide them through the following:

### Define Lambda functions.

### Define anonymous functions.

### Write sample lambda syntax.

### Identify how functions should be written if needing a return statement.

### Identify the number of expressions and arguments that can exist in Lambda code.

### 

### Lesson 2: Classes and Objects

* Build student’s knowledge and guide them through the following:

### Describe the value and features of classes.

### Define objects and what they consist of.

### Identify the function that every class has.

### Write sample code for creating a class.

### Lesson 3: Scopes

* Walk students through how to:
  + Define namespace.
  + Define the scope concept, local.
  + Define the scope concept, enclosed.
  + Define the scope concept, global.
  + Write code for the above scope concepts.

### Lesson 4: Modules

* Walk students through how to:
  + Define a module in Python.
  + Write code for recalling a module.
  + Write code that adds to the module.
  + Write code for recalling a module using an alias.

# Module 6 - Instructor guide

### Python Anonymous Functions, Classes and Objects

### Module Learning Outcomes

In this module students will,

### 1. Define key terminology for file handling.

### 2. Identify the different types files.

### 3. Identify how to read, append, and delete files.

### 4. Show how to import modules.

### Module Overview Description

This module will get students comfortable with how to handle text file types, append existing files, and open and create modules. The module wraps up with a Challenge activity to review what they have learned by answering foundational questions and creating Python code.

### Lesson 1: Open File Mode Arguments

* Build student’s knowledge and guide them through the following:

### Identify the key function when working with files in Python.

### Identify the two parameters needed to open files.

### Formulate example syntax for opening files in Python.

### Define the different mode arguments below:

### Read “r”

### Append “a”

### Write “w”

### Create “x”

### Identify the default mode that Python assumes if a mode argument is not used.

### Formulate example syntax for the read mode argument if you need to extract a string that contains all characters.

### Formulate example syntax for the read mode argument if you need to include a file path for a text file.

### Lesson 2: Reading Files

### Build student’s knowledge and guide them through the following:

* + Formulate example syntax using the readline() method.
  + Identify what using the readline() method twice does.
  + Formulate example syntax using the for argument and describe what this does.
  + Formulate example syntax that closes Python files.
  + Explain why you need to clean up the output when presenting data.
  + Define line splitting.
  + Write an example of a splitlines()statement.

### Lesson 3: Append Write Create Files

* Build student’s knowledge and guide them through the following:
  + Define appending a file.

### Describe what the following append modes do:

* + - ‘a’
    - ‘a+’
    - Formulate example syntax for the append mode.
    - Formulate example syntax for writing a file ovewrite.
    - Explain why one should be caution when overwriting a file.
    - Formulate example syntax for creating a new file using the open() method and one of the mode parameters.

### Define the following Python operators:

### r

### rb

### r+

### rb+

### w

### wb

### w+

### wb+

### a

### ab

### a+

### ab+

### Lesson 4: Delete Files and JSON Types

* Walk students through how to:
  + Write an example of syntax that removes a text file.
  + Can files be retrieved once they are deleted?
  + Write example syntax that checks if a file exists and then sets the code to delete it.
  + Identify an alternative and quicker way to determine if a file exists.
  + Define JSON
  + Identify the method needed to read in JSON files.
  + Describe what the json.load() does.
  + Write an example json syntax.
  + Identify the two ways to return objects using the loads() method.
  + Identify when you will work with JSON files.
  + For each of the Python objects listed below, write in the JSON object.

|  |  |
| --- | --- |
| Python | JSON |
| dict |  |
| list |  |
| Str |  |
| None |  |
| In |  |
| Float |  |
| True |  |
| False |  |

# Module 7 - Instructor guide

### Introduction to Pandas

### Module Learning Outcomes

In this module students will,

### 1. Define key terminology for Pandas.

### 2. Demonstrate how to install pandas

### 3. Recognize different series.

### 4. Identify how to create datasets and dataframes.

### 5. Identify how to read, append, and delete files.

### Module Overview Description

This module will introduce students to Pandas, creating datasets, dataframes, and csv files in Python. The module wraps up with a Challenge activity to review what they have learned by answering foundational questions and creating Python code.

### Lesson 1: Importing Pandas and Series

* Build student’s knowledge and guide them through the following:

### Define Pandas.

### Identify what Pandas does.

### Describe how to import Pandas.

### Describe the benefits of setting a series to your data.

### Identify what a series consists of.

### Identify the constructors that series class provides.

### Write example code that creates a basic series from a list.

### Write example code that sets index labels to the series.

### Write code to print a value of your choice in the index.

### Lesson 2: Dictionary, List Series

### Build student’s knowledge and guide them through the following:

* + Write code that uses keys and object values to create a series using a list index.
  + Write code that creates a series from a dictionary and assigns values with a colon.
  + Write code that extracts a few items from a dictionary using the pd.Series argument.
  + Write code that uses the index () argument to assign the index structure for a list.
  + Identify what NAN means.

### Lesson 3: Data Frames

* Build student’s knowledge and guide them through the following:
  + Define data frames.
  + Discuss the difference between a series and a data frame.
  + Identify what can make up a data frame.
  + Identify how to create a dataframe using Pandas.
  + Identify the steps needed to create datasets in Python.
  + Identify the steps needed to create an empty frame.
  + Write the code needed to create an empty frame.
    - Explain the code for creating an empty frame.
  + Write example syntax for creating a set of data that holds employee information.
    - Explain the example syntax code for creating a set of data that holds employee information.
  + Write example syntax for creating a dataframe from two different datasets.
    - Explain the example code for creating a dataframe from two different datasets.
  + Write example syntax for using the loc attribute to locate and return one or more specific rows from a dataframe.

### Lesson 4: CSV and JSON Files

* Walk students through how to:
  + Write the syntax for reading in a csv file into Python.
    - Explain what the syntax is telling Python.
  + Write the syntax for loading a csv file using the file path name.
  + Identify how to print the entire set of data when reading in a csv file, rather than just the first 5 rows.
  + Write example syntax for reading in JSON files using Pandas.

# Module 8 - Instructor guide

### INTRODUCTION TO NUMPY

### Module Learning Outcomes

In this module students will,

### 1. Define key terminology for NumPy.

### 2. Demonstrate how to install NumPy.

### 3. Demonstrate how to create arrays.

### 4. Demonstrate how to slice arrays.

### 5. Identify how to read and index arrays.

### Module Overview Description

This module will introduce students to NumPy and arrays. The module wraps up with a Challenge activity to review what they have learned by answering foundational questions and creating Python code.

### Lesson 1: Importing Pandas and Series

* Build student’s knowledge and guide them through the following:

### Define NumPy and what it is used for.

### Define an array.

### Identify how to determine the rank of the array.

### Identify what defines the shape of an array.

### Describe how NumPy and Pandas are different.

### Identify the array class of NumPy and what that means.

### Describe what homogenous data types means.

### Create code for a specified data type with an array.

* + Identify the meaning of the following Python single characters:
    - i
    - b
    - u
    - f
    - c
    - m
    - M
    - O
    - S
    - U
    - V
* Identify how Python indicates a data type.
* Identify what happens when you mix multiple data types in one code.
* Identify how to convert a data type to another data type.
* Write code for a multidimensional array.

### Lesson 2: Array Shapes

### Build student’s knowledge and guide them through the following:

### Write example syntax for a one-dimensional array.

### Write example syntax for a two-dimensional array.

### Write example syntax for a three-dimensional array.

### Describe why Python may convert an array to a list.

### Describe advantages of using arrays.

### Describe important things to know about lists over arrays.

### Lesson 3: Array Indexing

* Build student’s knowledge and guide them through the following:
  + Define array indexing.
  + Write example code that shows how to index from a created array.
  + Write example code that shows how to index and then add values together.
  + Describe what negative indexing allows you to do and write sample code for this.
  + Identify the difference between a copy and an array.
  + Write example code for creating a copy of an original array.

### Lesson 4: Array Slicing

* Walk students through how to:
  + Define slicing.
  + Describe why you would slice.
  + Describe what should be considered when slicing.
  + Write example code that shows slicing an array element from an index to the end of the array.
  + Write code that shows negative slicing

This module has the [*Python Pandas and NumPy*](https://content.bridgepointeducation.com/curriculum/file/13b6cb01-845a-415a-bbaf-80c7f8ac1aa4/1/Python%20Pandas%20and%20Numpy.zip/story.html)interactive available for students to interact with and test their newly acquired skills. This interactive challenges students to use Python to create lines of code for specific purposes.

# Module 9 - Instructor guide

### PYTHON INTRODUCTION TO VISUALS

### Module Learning Outcomes

In this module students will,

### 1. Define key terminology for matplotlib.

### 2. Demonstrate how to install matplotlib.

### 3. Demonstrate how to create basic charts.

### 4. Demonstrate how to create comparison charts.

### 5. Identify how to read chart results.

### Module Overview Description

### This module will introduce students to matplotlib and how to create visuals. The module wraps up with a Challenge activity to review what they have learned by creating Python code using matplotlib to create basic charts.

### Lesson 1: Import MATPLOTLIB

* Build student’s knowledge and guide them through the following:
  + Define Matplotlib and describe what it does.
  + Describe how you use matplotlib and NumPy together.
  + Write code for how you import the pyplot module of the matplot library.
  + Write the code for importing NumPY and Matplotllib together.
  + Describe what pyplot does and how it is used.
  + Describe the following call signatures and data labels:
    - x,y
    - fmt
    - data
* Identify what you should keep in mind while working through visuals in Python.
* Identify the steps to successfully create a plot using Matplotlib.

### Lesson 2: Plotting

### Build student’s knowledge and guide them through the following:

### Describe the steps for potting a diagram using matplotlib.

### Write code for plotting a line using matplotlib.pyplot as plt

### Write code for plotting points on a chart that coincide with the data elements but does not draw a connecting line.

### Write code for plotting an array with multiple points.

### Write code for plotting a graph that does not specify the x-axis points.

### Lesson 3: Scatter Plot

* Build student’s knowledge and guide them through the following:
  + Write example code for creating a scatter plot using the scatter() function.
  + Write example code for comparing data from two different datasets using the scatter() function syntax.

### Lesson 4: Bar Pie Charts

* Walk students through how to:
  + Define the bar()function and describe what it does.
  + Describe the steps to use the bar()function.
  + Write example code using the bar()function.
  + Write example code that created a bar chart with horizontal columns.
  + Write example code that creates a pie chart.

This module has the [*Python Introduction to Visuals*](https://content.bridgepointeducation.com/curriculum/file/c569f4aa-b24c-4ac5-b122-c0333aa9ffe7/1/Python%20Introduction%20to%20Visuals.zip/story.html)interactive available for students to interact with and test their newly acquired skills. This interactive will challenge students to type in the correct syntax were prompted or select the correct chart type.

# Module 10 - Instructor guide

### PYTHON FORMATTING VISUALS

### Module Learning Outcomes

In this module students will,

### 1. Define key terminology related to formatting visuals in Python.

### 2. Demonstrate how to create markers, labels, and titles.

### 3. Demonstrate how to create subplots.

### 4. Demonstrate how to create color schemes.

### Module Overview Description

### This module will introduce students to multiple formatting options for their visuals. Using markers, labels, titles, and color schemes will keep your audience engaged in their data presentation. The module wraps up with a Challenge activity to review what they have learned by creating Python code using matplotlib to create basic charts and apply formatting concepts.

### Lesson 1: Import MATPLOTLIB

* Build student’s knowledge and guide them through the following:
  + Define markers and what they are used for.
  + Describe how to add markers to a chart in Python.
  + Write example code for adding markers to a chart in Python.
  + Identify the syntax for the following colors in Python:
  + Red
  + Green
  + Blue
  + Cyan
  + Magenta
  + Yellow
  + Black
* Write example code for making a plot with a green dotted line.
* Describe how to change the size and color of markers in a chart using Python.
* Write code for creating larger markers and a different marker face color on a chart.
* Describe how to add labels to a chart in Python.
* Write code for adding labels to a chart in Python.

### Lesson 2: Formatting Color Schemes

### Build student’s knowledge and guide them through the following:

### Describe how to create a bar chart with bars of a variety of colors.

### Write example code for creating a bar chart with a variety of colors and is labeled.

### Write example code for creating a bar chart with labels, colors, and a title.

### Lesson 3: Creating Multiple Plots

* Build student’s knowledge and guide them through the following:
  + Define the subplot() function and describe what it does.
  + Describe what is needed to use the subplot() function.
  + Write example code for two plots generated from a single syntax with parameters of your choice.
    - Identify parameters.
  + Add titles to the above code.

### Lesson 4: Color Bars

* Walk students through how to:
  + Describe why you would add a color bar to a scatter plot chart.
  + Describe how to add color bars to charts in Python.
  + Write example code for adding a color bar to a chart in Python.

This module has the [*Python Formatting Charts*](https://content.bridgepointeducation.com/curriculum/file/33f02522-5bad-482e-8e9d-a07356358038/1/Python%20Formatting%20Charts%20and%20Visuals%20Using%20Matplotlib.zip/story.html)interactive available for students to interact with and test their newly acquired skills. This interactive challenges students to type in the correct syntax where prompted.

# Module 11 - Instructor guide

### Python Data Cleaning

### Module Learning Outcomes

In this module students will,

### 1. Identify how data is cleaned using Python syntax.

### 2. Demonstrate how to identify and remove duplicate data in a data series.

### 3. Demonstrate how to reverse a string.

### 4. Demonstrate how to import and plot in Seaborn.

### Module Overview Description

### This module will introduce students to multiple syntax arguments in how to clean and transform their data. Through data transformation they will be able to plot their data using Seaborn. The module wraps up with a Challenge Activity to review what they have learned by creating Python code using matplotlib to create basic charts and apply formatting concepts.

### Lesson 1: Missing Data

* Build student’s knowledge and guide them through the following:
  + Explain why objects will be excluded when working with data in Pandas.
  + Define NaN
  + Identify what to do if you choose to include NaN values when presenting.
  + Identify the argument needed to fill in the missing data and avoid discarding important data.
    - Explain what this will do.
  + Identify the import argument needed to import the nan library from NumPy.
    - Describe how nan functions and what it can do.
  + Write code for the data output with NaN data.
  + Identify the argument needed to tell Python to drop the missing data.
  + Write example code for Python to drop missing data and clean the data series.
  + Break down what the code is telling us.
  + Identify how to return the cleaned data output.

### Lesson 2: Data Transformation

### Build student’s knowledge and guide them through the following:

### Identify what the duplicate() argument returns in a dataframe.

### Describe the options you have with duplicate dataframes and the pros and cons of each.

### Write example code for dropping duplicate data.

### Describe why it is important to include command line comments to syntax when dropping duplicate data.

### Lesson 3: Transforming Data

* Build student’s knowledge and guide them through the following:
* Identify the arguments needed to convert all data characters in a data frame to be lowercase.
* Write example code for converting data characters to lowercase.
* Describe how to create a new column of data within a dataframe.
* Describe how to map the data to the new column category.
* Write example code for creating a new column and mapping data to that new column.

### Lesson 4: SEABORN

* Walk students through how to:
  + Describe what you can do using Seaborn in Python.
  + Describe how to use seaborn in Python.
  + Identify how to import Seaborn into Python and what is best practice.
  + Write example code for basic distribution plot syntax.
  + Describe what a normal distribution is.
  + Identify what the mean and standard deviation are.

# Module 12 - Instructor guide

### Python Machine Learning

### Module Learning Outcomes

In this module students will,

### 1. Identify the importance of machine learning.

### 2. Demonstrate how to identify percentiles.

### 3. Demonstrate how to read distribution plots.

### 4. Demonstrate how to work with random, big datasets.

### Module Overview Description

### This module will introduce students how to run and understand statistics while working in both small and large datasets. Using Machine learning applications, they will be able to plot their data and understand how the data compares to itself. The module wraps up with a Challenge Activity to review what they have learned by creating Python code using NumPy and SciPy to run statistical analysis.

### Lesson 1: Getting Started

* Build student’s knowledge and guide them through the following:
  + Define machine learning.
  + Describe how businesses are using machine learning.
  + Identify an example of an array.
  + Define the following data types:
    - Numerical
    - Categorical
    - Ordinal
  + Define the following statistical values:
    - Mean
    - Median
    - Mode
  + Describe how to use NumPy to calculate the mean of a dataset.
  + Write example code to calculate the mean of a dataset.
  + Describe how to use NumPy to calculate the median of a dataset.
  + Write example code to calculate the median of a dataset.
  + Describe how to use scipy to calculate the mode of a dataset.
  + Write example code to calculate the mode of a dataset.

### Lesson 2: Standard Deviation

### Build student’s knowledge and guide them through the following:

### Define standard deviation.

### Describe how to find the standard deviation using std() method.

### Write example code to find standard deviation in Python.

### Describe how to plot a distribution using the sns.distplot() method.

### Write example code for plotting a distribution using the sns.distplot() method.

### Define variance.

### Describe the relationship between variance and the standard deviation.

### Explain what variance tells us about performance.

### Write example code that uses the var() method to find variance.

### Lesson 3: Percentiles and Data Distribution

* Build student’s knowledge and guide them through the following:
* Define percentiles in statistics.
* Describe the following quartiles:
  + Quartile 1
  + Quartile 2
  + Quartile 3
* Write example code for using the NumPy percentile() method to find a specified percentile.
* Write example code using the percentile() method that shows how the data compares against the mean.

### Lesson 4: Data Distribution

* Walk students through how to:
  + Describe what the random() method allows you to do.
  + Explain why when using the random() method a different result will return each time.
  + Write example code for using the random.choice() method.
  + Describe what the random.random() syntax is used for.
  + Write code for creating a random large dataset using random.uniform().
  + Write example code for creating a histogram in addition to a random dataset using the random.uniform and plt.hist() methods.
  + Write example code that creates a normal distribution by using the random.normal() argument.
  + Write example code that creates a scatter plot using the scatter() argument.