

# Appendix C: Python for Data Professionals

Rev# 20180128

## Table of Contents

Objectives.....	C-1
Module Questions .....	C-2
Lesson 1 .....	C-4
Built-In Commands .....	C-5
Demonstration 1.....	C-6
Lesson 2.....	C-7
Demonstration 2.....	C-8
Lesson 3.....	C-9
Demonstration 3.....	C-10
Lesson 4.....	C-11
Demonstration 4.....	C-12
Lesson 5.....	C-13
Demonstration 5.....	C-14
Lesson 6.....	C-15
Demonstration 6.....	C-16
Lesson 7.....	C-17
Demonstration 7.....	C-18
Review.....	C-19
Lab C (2 – 4 hours) .....	C-21
Exercise 1: Running Commands.....	C-22
Exercise 2: Creating Variables.....	C-23
Exercise 3: Creating Functions .....	C-24
Exercise 4: Use Control Flow Statements.....	C-25
Exercise 5: Working with Dates and Times.....	C-26
Exercise 6: Working with Data .....	C-27

Exercise 7: Visualizing Data.....	C-28
Exercise 8: Skills Project (Optional) .....	C-29





## Objectives

### Module Objectives

- Running Commands
- Creating Variables
- Creating Functions
- Flow Control Statements
- Working with Dates and Times
- Working with Data
- Visualizing Data

The Python logo, consisting of two interlocking snakes, one blue and one yellow, set against a black square background.

There is a wealth of free material available to teach anyone how to use Python. The documentation on <http://www.python.org>, for example, can help anyone regardless of their level of experience. The purpose of this tutorial is to focus on helping a specific audience, data professionals with little or no experience using Python. The information here will focus on specific tasks that are commonly done and use exercises to test and reinforce those skills.

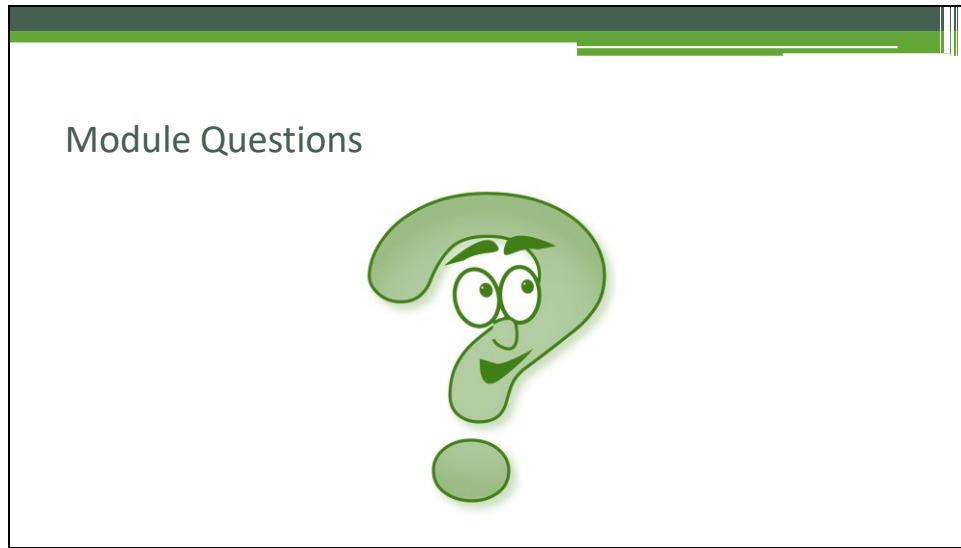
If you are new to python, you will learn how to do the following tasks:

1. Run Commands
2. Create Variables
3. Create Functions
4. Flow Control Statements
5. Working with Dates and Times
6. Working with Data
7. Visualizing Data

The goal is to have students learn python data related operations quickly by briefly explaining them and using them in lab exercises. Each lesson to follow will teach you a different skill and show you how to use what you have learned to complete a specific exercise. Complete the exercise for a lesson before moving on to the next one. At the end of the tutorial, you will be given a project that brings all of those lessons together to further reinforce your new skills. The whole tutorial should take 2 to 4 hours to complete. Exercises that require you to graphically represent your data (e.g. using Matplotlib), will require a local computer or virtual machine.

If you are not using this tutorial as a part of course 552241A or 552242A, then you will have to install and configure python on your own. The <http://www.python.org> web-site will provide all the documentation needed as well as information on choosing tools to run python or ipython (e.g. jupyter).

## Module Questions



This tutorial is designed to teach basic python mathematical and visualization skills in a few hours. Other skills, focusing on the management of Azure resources, will be learned during the 552241A and 552242A classes. By the end of the tutorial, students should be able to answer the following questions:

1. True or False. Python commands are case-sensitive.
2. What command can be run to get documentation on an object?
3. What command will make functions and methods in the Numpy library available in a session?
4. What command is used to verify the version of python running in a console?
5. What command can you use to verify the attributes available for the datetime module?
6. What command can be used to install the Numpy module in a PowerShell or Bash Shell?
7. True or False. Numpy arrays are useful when mathematical calculations must be performed on elements in a list.

8. True or False. Tuples cannot be modified after they are created.
9. How would you convert the int1 variable (int1=200) to a string value stored in a variable named str1?
10. What command will allow you to append the number 6 to a list named list2?
11. Why would you use the return statement in a function definition?
12. How would you define a function variable so it is available after the function runs?
13. How can you stop a looping operation before the test condition evaluates to false?
14. What type of statement will you use to perform an operation for each element in a list?
15. True or False. The else statement is used to test a different condition from the original if statement.
16. What command can you run to find attributes available with the datetime module after it is imported with an alias of dt?
17. What command can you use to assign a date that is 60 days from today to a variable named enddate?
18. True or False. A Pandas dataset can be manipulated like a table or spreadsheet.
19. What command will export a Pandas dataframe named df2 to a CSV file named df2.csv?
20. What command will allow you to list the style sheets available for a Matplotlib chart?

## Lesson 1



As with most programming language, Python allows you to run commands interactively or in a script. After installing the compiler, you will type “python” to use the interactive interface. The commands will work the same from Linux or Windows shells. Python scripts are created as text documents and normally have an extension of py.

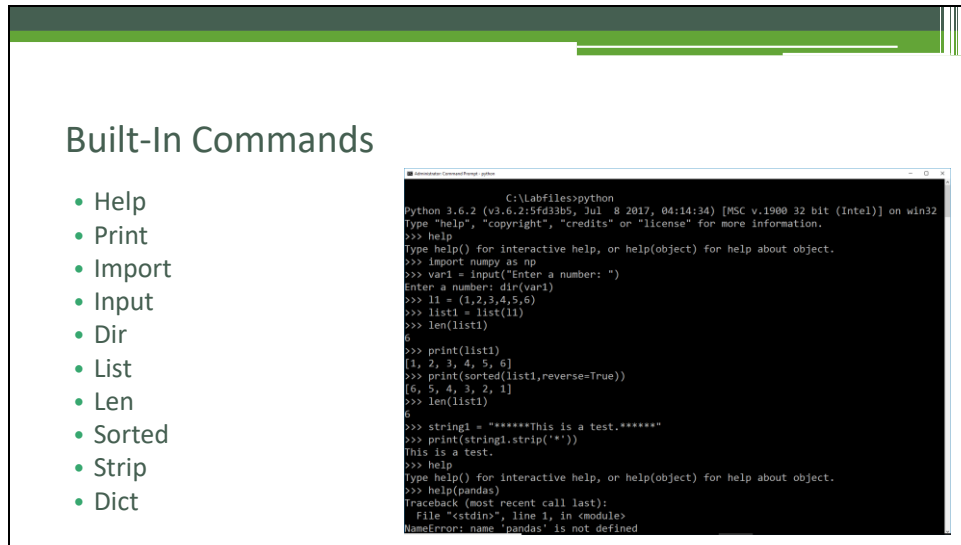
The functionality of the language can be increased by importing libraries. These libraries are responsible for the popularity of Python among data scientist and enthusiasts and some allow you to visualize your information (e.g. numpy, pandas, scipy or matplotlib). For those who use R, access to the R programming components can be accessed using the rpy2 library. These modules are not installed by default and must be added before their functionality can be used. Pip is a popular tool for doing this from a shell (e.g. pip install scipy). It can also be used to install a python environment for running commands (e.g. pip install jupyter ; pip install pyqt5).

As new versions of python come out, the functionality and syntax of some commands will change. It is always a good idea to know what version you are using to verify what you will be able to do on a system. The “python –version” command will verify the version of python you are working on.

**Note:** If you are using the Azure Cloud Shell, keep in mind that PowerShell and Bash use different versions of python. This will affect the syntax you use for some of your commands. For example, the print statement may require parentheses in some cases but not in others.



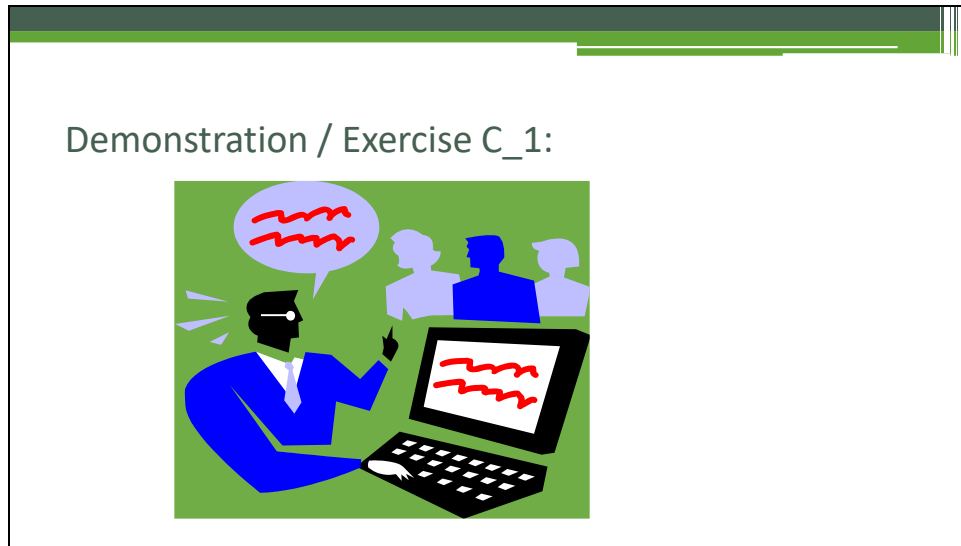
## Built-In Commands



Some of the more popular python functions and methods are listed here. This is a good starting point for those new to the environment. All commands are case-sensitive and usually lower-case. Here is a brief description of each of them:

- **Help:** This will provide documentation on an object, method or module (e.g. `help(pandas)`).
- **Print:** Will display the value assigned to a variable or text to the screen (e.g. `print("Hello World")`).
- **Import:** Used to add libraries to the environment so their methods and functions can be accessed (e.g. `import pandas`).
- **Input:** Can be used to accept interactive input, often from the keyboard which is usually stored in a variable (e.g. `input1 = input("Please enter a value: ")`).
- **Dir:** To get a listing of the attributes available on an object that is already imported or defined in the environment (e.g. `dir(numpy)`).
- **List:** This function will convert any sequence of values into a python list that is more easily manipulated (e.g. `sequence1 = (1,2,3,4,5) ; list1 = list(sequence1)`).
- **Len:** Provides the number of elements in a list (e.g. `len(list1)`).
- **Sorted:** Sorts a list in ascending or descending order (e.g. `sorted(list1, reverse=True)`).
- **Strip:** This method removes specified characters from the beginning and end of a string (e.g. `string1 = "+++This is a test.+++" ; print(string1.strip('+'))`).
- **Dict:** Dictionary functions are used to store key-value pairs (e.g. `dict1 = {'name':'Bill Williams'} ; type(dict1)`).

### Demonstration 1



Note: If you are not using a python IDE, make sure the console (Linux or Windows) has administrator or root privileges before running python to open the shell. The **C:\Labfiles\Python\labC\_1.py** script can be used for the demonstration or exercise.

Any text editor can be used to create or edit the scripts. Students should be encouraged to use a popular Python IDE (PyCharm, PyDev, IDLE, e.t.c.)

Demonstration: Show students how to use each of the commands mentioned in this lesson, then have them do Exercise 1 before moving to the next lesson.

## Lesson 2

### Lesson 2: Creating Variables

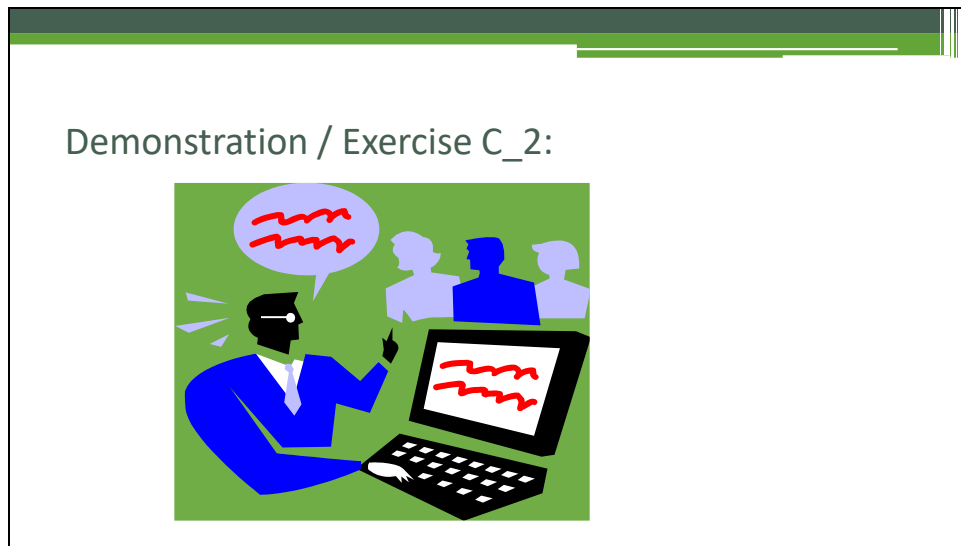
- Create Variables
- Convert Variables
- Variable Data Types
- Lists, Tuples & Arrays

All programming languages use variables to store values that may be used repeatedly and Python is no different. Creating one is simple and takes the format [Variable Name] = [Variable Value] ( `var1 = "This is a test."` ). The value of the variable can be retrieved by typing its name or putting it in a print statement ( `print(var1)` ).

Python variables support a number of data types such as strings, numbers, lists and others. The data type can be assigned when it is created or modified afterwards. Convert to integers with `int()`, to strings with `str()` or to float with `float()`. The conversion options depend on the value of the variable ( `var2 = 4 + 5` ; `var2 = str(var2)` ). The `type()` function is used to verify the data type of a variable ( `type(var2)` ).

Lists, tuples and arrays are used regularly by those working with large amounts of data in python. Lists are very flexible in terms of manipulating the data in them ( `list1=[1,2,3,4,5]` ). Values can be changed and elements added or removed ( `list1.append(6)` ). Tuples cannot be modified after they are created ( `tuple1 = (1,2,3,4,5)` ). Python arrays are useful for working with large datasets but do not provide the ability to perform mathematical operations on the data. Numpy arrays provide these capabilities, but require that all elements in the array have the same data type. Python lists do not have this limitation and can be modified much more easily than numpy arrays. As with python arrays, lists do not have the ability to perform the mathematical operations available with numpy arrays.

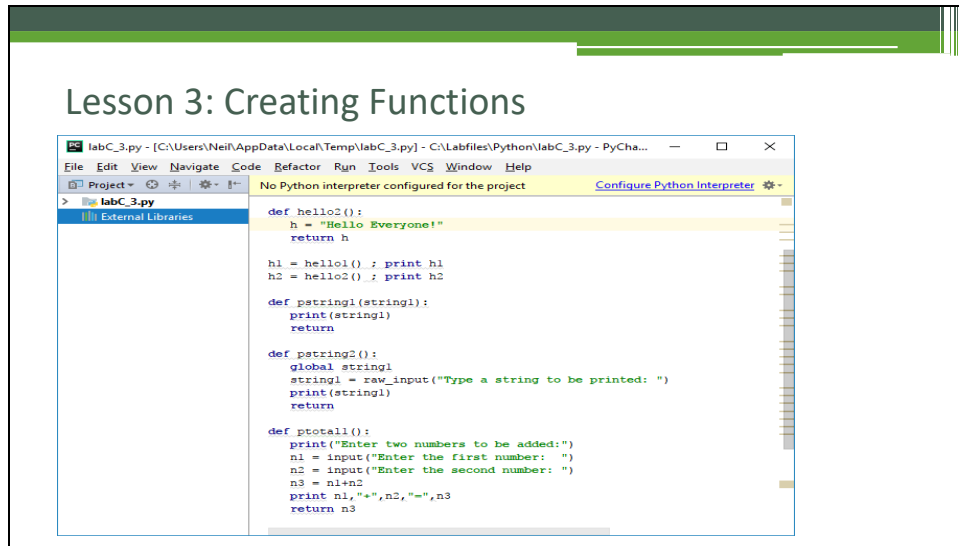
## Demonstration 2



Note: If you are not using a python IDE, make sure the console (Linux or Windows) has administrator or root privileges before running python to open the shell. The **C:\Labfiles\Python\labC\_2.py** script can be used for the demonstration or exercise.

Demonstration: Show students how to create variables and numpy arrays. Students should complete Exercise 2 before going to the next lesson.

## Lesson 3



Functions are used to run a single statement that performs multiple operations or commands. It allows you to assign a name to a stored list of commands. The syntax for creating a function simply involves using the `def` keyword, a unique name, a colon followed by one or more statements you want to execute. A simple function to print "Hello" to the screen could be written as follows:

```
def printh():
    print("Hello")
```

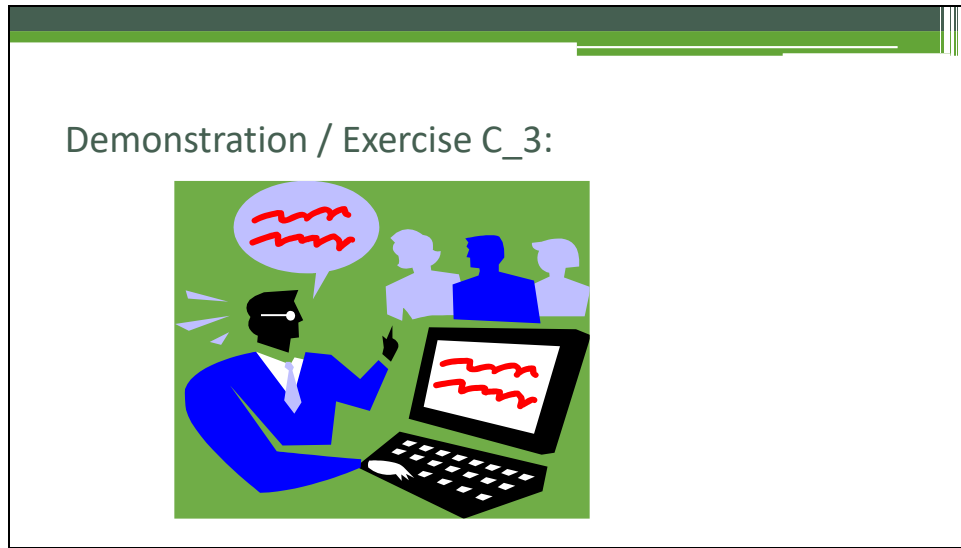
After it is defined, simply running `printh()` would print "Hello". Of course, the same function could be redefined to do other things. Input parameters can be used to pass values and variables to the function ( `def printh(var1):` ).

The `input()` function may even be used to accept values that you want to use in the function. An alternative is to use the `raw_input()` statement. `input()` will evaluate the statement given to see if it is an integer or string. `Raw_input()` will evaluate the data given as a string, regardless of what information is given.

If you need to pass a value from the function to the shell, the return statement can be used. All values generated in the function are local to it and are not available afterwards. The return statement allows you to pass one or more values from the function to the shell.

If a function's local variable must pass its value to the shell, this can be done by changing its scope to global ( `global var1` ). The variable and its value will then be available after the function is finished running.

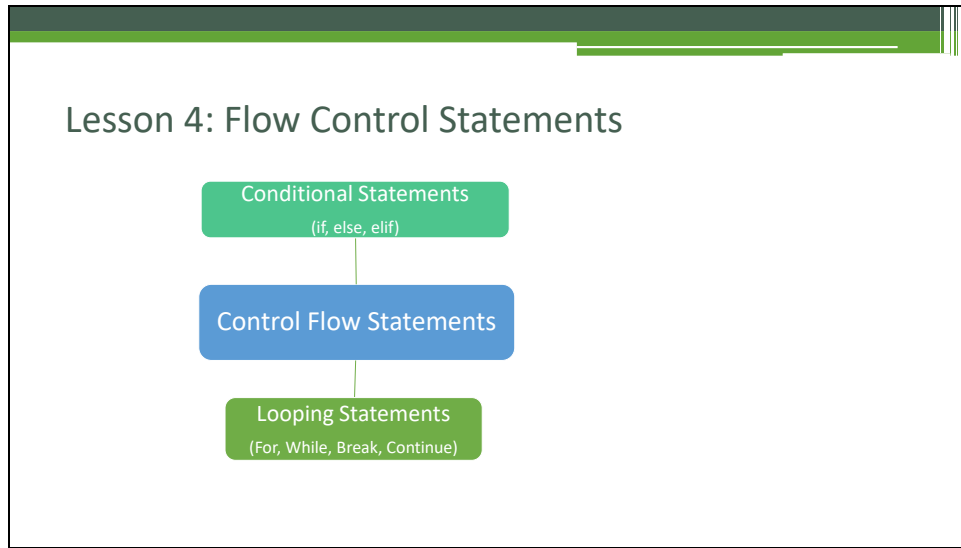
### Demonstration 3



Note: If you are not using a python IDE, make sure the console (Linux or Windows) has administrator or root privileges before running python to open the shell. The **C:\Labfiles\Python\labC\_3.py** script can be used for the demonstration or exercise.

Demonstration: Show students how to create functions that use input parameters, the input statement, the return statement and global variables. Students should complete Exercise 3 before going to the next lesson.

## Lesson 4



When the normal flow of a script must be changed, control flow statements are used to modify the order in which code is executed. Using flow and looping operations, code can be executed based on any criteria you choose.

We will discuss three common statements used to manage the order in which code is executed:

- If statements
- For statements
- While statements

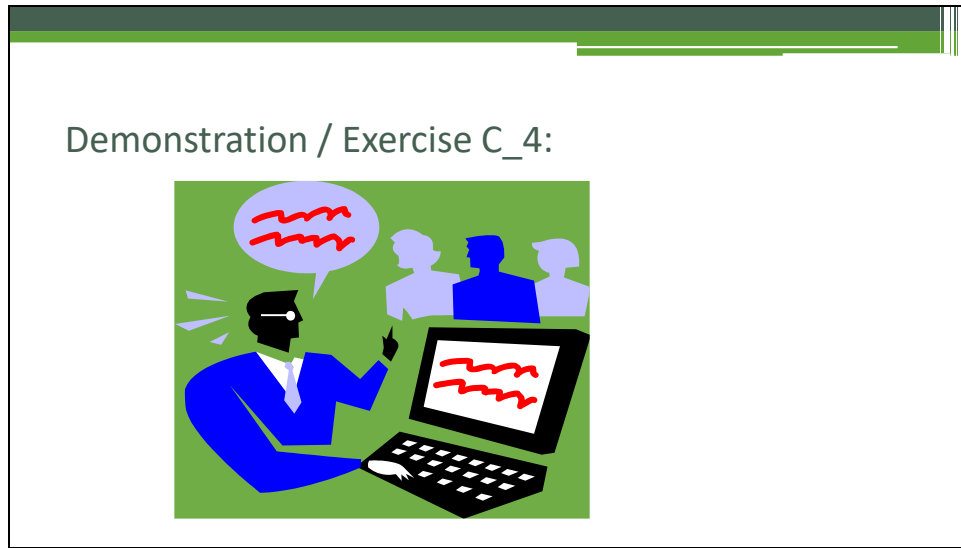
An if statement tests an expression and only performs the specified operation if it is true (if variable1 == 1: print("variable1 is equal to 1") ). If other code must be executed when the statement is false, the else: statement is used to specify what statements should be run. You may include one or multiple elif: statements that tests other expressions and assign code to be executed if any of them are true.

The for statement performs specified operations for each item in a sequence. A list or string can be used to specify the elements used in the sequence ( for [element] in [list of values]: print([element]) ). The statement will be executed for each element in the list.

While statements are similar to if statements in that they test whether a statement is true. The looping operation will continue until the statement is false ( while var1 <= 5: print(var1) ). An expression is normally included that will change the testing condition to prevent an endless loop ( var1 = var1 + 1 ).

When more complex conditions require that a looping or sequence operation end, the break statement is used to stop the current operation and continue to the next statement (if [True]: break ). The continue statement is used to prevent the rest of the operations in the sequence from running and return to the beginning of the looping operation.

#### Demonstration 4

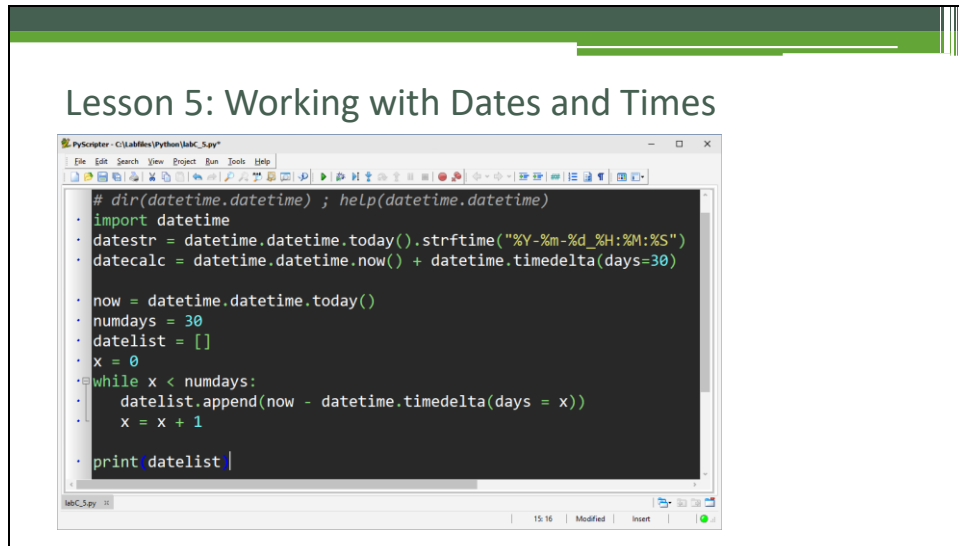


Note: If you are not using a python IDE, make sure the console (Linux or Windows) has administrator or root privileges before running python to open the shell. The **C:\Labfiles\Python\labC\_4.py** script can be used for the demonstration or exercise.

Demonstration: Show students how to create control flow statements using if, for and while constructs. Also demonstrate the use of break and continue statements in a while construct. Students should complete Exercise 4 before going to the next lesson.



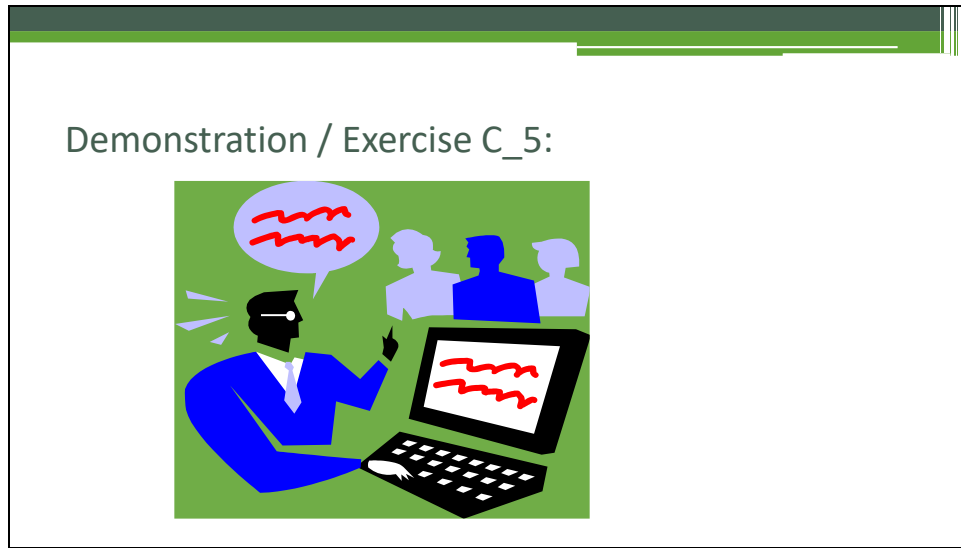
## Lesson 5



We often analyze data to find trends or compare information from different periods of time. The ability to use and manipulate time-based data is therefore important. Python provides modules that give this capability to coders and we will discuss the basics of using some of them.

We will use the `datetime` module for this lesson. The `datetime` module provides options for manipulating and representing dates and times ( `datetime.datetime.today().strftime("%Y-%m-%d_%H:%M:%S")` ). Mathematical operations may be performed on date-based variables such as adding or subtracting day ( `datetime.datetime.now() + datetime.timedelta(days=30)` ). Using looping operations, you may also create a range of dates and store them in a string or list.

### Demonstration 5



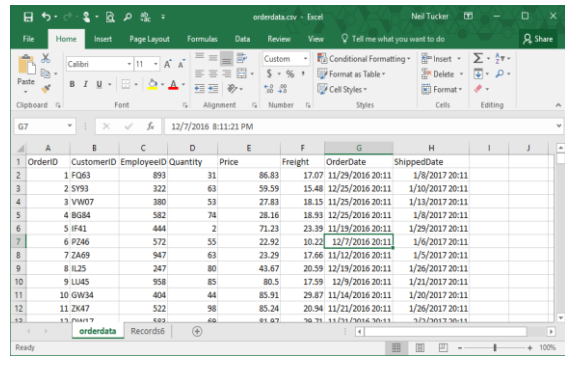
Note: If you are not using a python IDE, make sure the console (Linux or Windows) has administrator or root privileges before running python to open the shell. The **C:\Labfiles\Python\labC\_5.py** script can be used for the demonstration or exercise.

Demonstration: Show students how to create variables that store dates. Use a looping operation to create a list that stores date information in a specified range. Students should complete Exercise 5 before going to the next lesson.

## Lesson 6

### Lesson 6: Working with Data

- Creating Datasets
- Importing data from files
- Exporting data



OrderID	CustomerID	EmployeeID	Quantity	Price	Freight	OrderDate	ShippedDate
1	FO63	893	31	86.83	17.07	11/29/2016 20:11	1/8/2017 20:11
2	SY93	322	63	59.59	15.48	12/25/2016 20:11	1/10/2017 20:11
3	VW07	380	53	27.83	18.15	11/25/2016 20:11	1/13/2017 20:11
4	B084	582	74	28.16	18.93	12/25/2016 20:11	1/9/2017 20:11
5	IF41	444	2	71.23	23.39	11/19/2016 20:11	1/29/2017 20:11
6	P246	572	55	22.92	10.22	12/7/2016 20:11	1/6/2017 20:11
7	ZA69	947	63	23.29	17.66	11/12/2016 20:11	1/5/2017 20:11
8	IL25	247	80	43.67	20.59	12/19/2016 20:11	1/26/2017 20:11
9	LU45	958	85	80.5	17.59	12/9/2016 20:11	1/21/2017 20:11
10	GW34	404	44	85.91	29.87	11/14/2016 20:11	1/20/2017 20:11
11	TD47	322	98	85.34	20.94	11/21/2016 20:11	1/24/2017 20:11
12	PH47	683	48	61.67	18.71	11/31/2016 20:11	1/23/2017 20:11

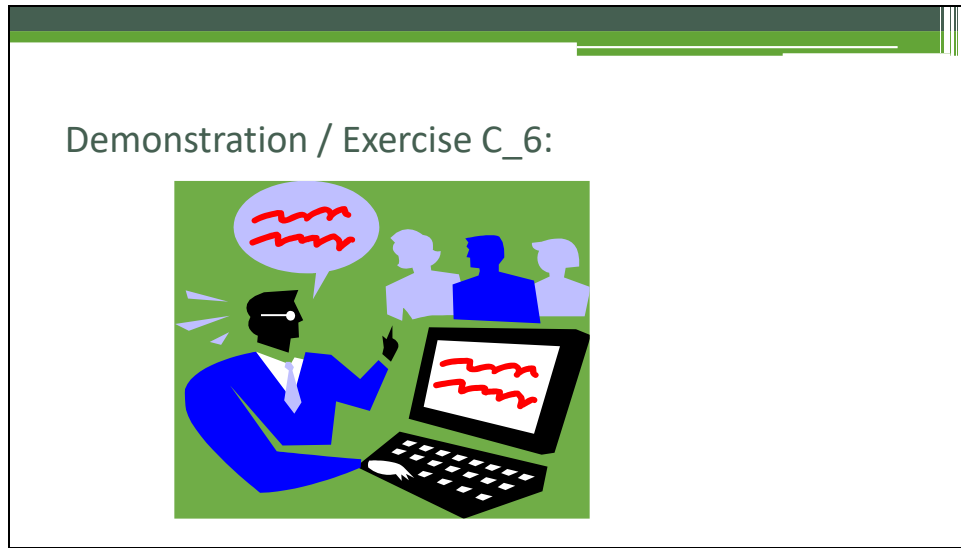
One of the main reasons for python's popularity is its power to manipulate data and the modules often used for these tasks are numpy and pandas. This lesson will focus mainly on what can be done with pandas. This is a powerful module for analyzing and modeling data, but we will focus on three areas in this section:

- Creating Datasets
- Importing data from files
- Exporting data to files

Many datasets begin as lists that are then converted into pandas dataframes to allow more complex mathematical operations ( `list1 = [1,2,3,4,5]` ; `df1 = pandas.DataFrame(list1)` ). Multiple lists can be combined to create a "table" of related information ( `df2 = pandas.DataFrame(list(zip(list2,list3,list4)))` ). Additional "columns" can be added to the dataset as needed ( `df3 = df1.join(df2)` ). Using the different options available for pandas DataFrames, datasets can be manipulated in much the same way as a database table using concatenate, merge, sort and other operations.

Pandas also makes it relatively simple to import and export data from text files, spreadsheets & databases into and out of datasets. A common format used for exports is comma-delimited files ( `df3.to_csv('df3.csv')` ). Data can also be exported in a JSON format ( `df3.to_json('df3.json')` ). Options are available for managing headers and delimiters for both file types. The `read_csv` & `read_json` methods are used to create datasets from existing files ( `df3_csv = pandas.read_csv('df3.csv')` ).

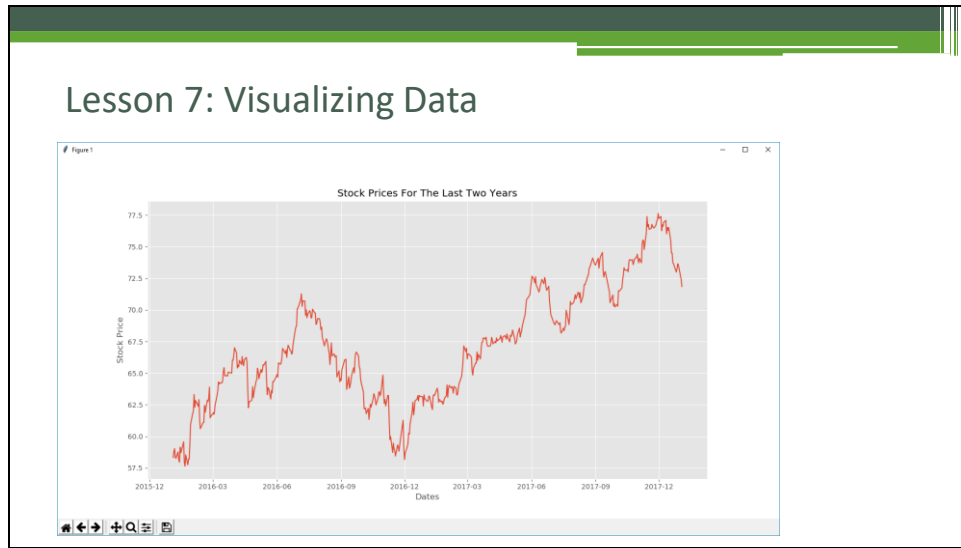
## Demonstration 6



Note: If you are not using a python IDE, make sure the console (Linux or Windows) has administrator or root privileges before running python to open the shell. The **C:\Labfiles\Python\labC\_6.py** and **RandomizedDataset.py** scripts can be used for the demonstration or exercise.

Demonstration: Show students how to create and manipulate datasets using the pandas module. Export datasets to csv files and reverse the process. Students should complete Exercise 6 before going to the next lesson.

## Lesson 7

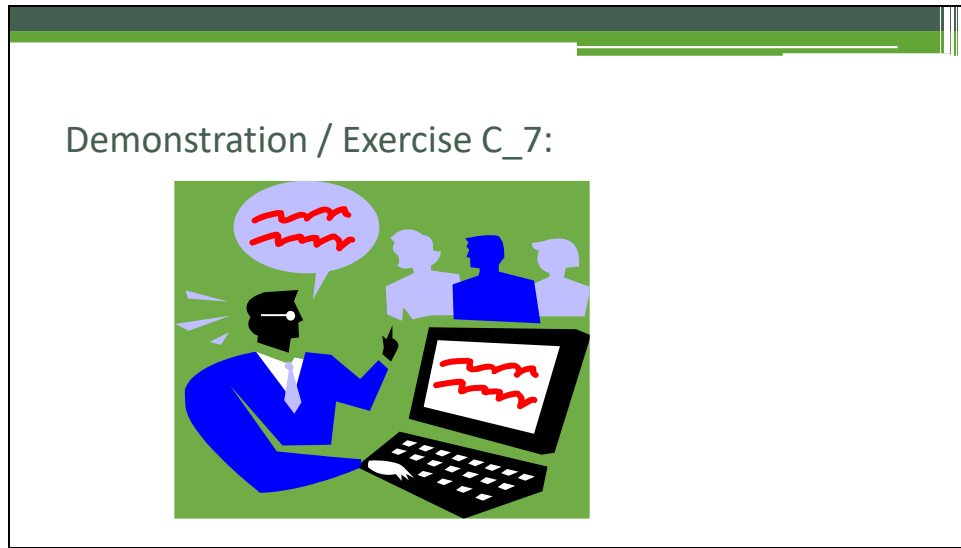


Our final lesson will look at using matplotlib to visualize data. Matplotlib was designed to emulate MATLAB which is used for modelling, simulation and visualization. We will use the matplotlib.pyplot collection to graphically represent information from different sources.

The simplest setup will use two pyplot options. Plot and show (import matplotlib.pyplot as plt ; plt.plot([34,21,11,89,43,28,89,92,62,94]) ; plt.show() ). The plot function can be used multiple times to display multiple charts in the same graph. The x and y axis parameters can be managed with customized settings. The show function is executed last after all other methods are configured, such as those that configure labels and titles ( plt.title('Chart Title') ; plt.xlabel('Time') ; plt.ylabel('Price') ; plt.legend() ).

Other chart options allow you to control the colors, type of chart, date ranges represented, whether to use grid lines or what format to save the graph in. Preconfigured style sheets are also available to automatically set options to have a specified type of design ( plt.style.use('ggplot') ). The available style sheets can be viewed with the plt.style.available function.

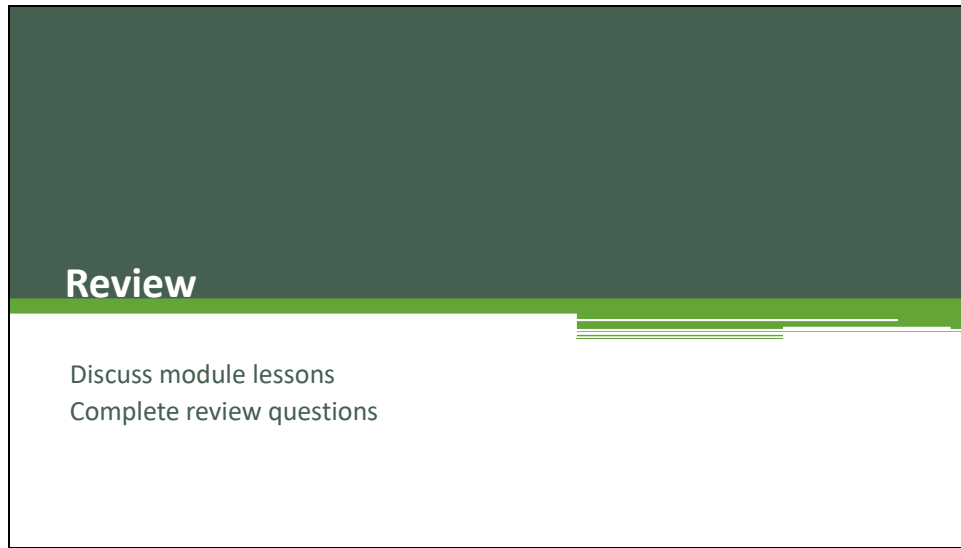
### Demonstration 7



Note: If you are not using a python IDE, make sure the console (Linux or Windows) has administrator or root privileges before running python to open the shell. The **C:\Labfiles\Python\labC\_7.py** script can be used for the demonstration or exercise.

Demonstration: Show students how to use matplotlib to create charts. Show the different options available for style sheets and displaying or saving charts. Students should complete Exercise 7 before doing exercise 8.

## Review



1. True or False. Python commands are case-sensitive.  
**True**
2. What command can be run to get documentation on an object?  
**help**
3. What command will make functions and methods in the Numpy library available in a session?  
**Import numpy**
4. What command is used to verify the version of python running in a console?  
**python --version**
5. What command can you use to verify the attributes available for the datetime module?  
**dir(datetime)**
6. What command can be used to install the Numpy module in a PowerShell or Bash Shell?  
**pip install numpy**
7. True or False. Numpy arrays are useful when mathematical calculations must be performed on elements in a list.  
**True**
8. True or False. Tuples cannot be modified after they are created.  
**True**
9. How would you convert the int1 variable (int1=200) to a string value stored in a variable named str1?  
**str1 = str(int1)**

10. What command will allow you to append the number 6 to a list named list2?  
**list2.append(6)**
11. Why would you use the return statement in a function definition?  
**To return a value to the shell after the function is finished running**
12. How would you define a function variable so it is available after the function runs?  
**By using the global statement (global variable1)**
13. How can you stop a looping operation before the test condition evaluates to false?  
**Use the break statement**
14. What type of statement will you use to perform an operation for each element in a list?  
**The for statement**
15. True or False. The else statement is used to test a different condition from the original if statement.  
**False. This is done with the elif statement.**
16. What command can you run to find attributes available with the datetime module after it is imported with an alias of dt?  
**dir(dt)**
17. What command can you use to assign a date that is 60 days from today to a variable named enddate?  
**enddate = datetime.datetime.now() + datetime.timedelta(60)**
18. True or False. A Pandas dataset can be manipulated like a table or spreadsheet.  
**True**
19. What command will export a Pandas dataframe named df2 to a CSV file named df2.csv?  
**df2.to\_csv('df2.csv')**
20. What command will allow you to list the style sheets available for a Matplotlib chart?  
**print(matplotlib.pyplot.style.available)**



## Lab C (2 – 4 hours)

### Lab C: Python for Data Professionals

- Exercise 1: Running Commands
- Exercise 2: Creating Variables
- Exercise 3: Creating Functions
- Exercise 4: Flow Control Statements
- Exercise 5: Working with Dates and Times
- Exercise 6: Work with Data
- Exercise 7: Visualizing Data
- Exercise 8: Skills Project

Note: Make sure that the Shell that you use to run Python commands gives you Administrator privileges on Windows systems or Root access on Linux / Unix based computers. The exercises are of varying complexity, but try to take no more than 2 – 3 hours on exercises 1 – 7. Exercise 8 is optional and should take no more than 1 to 2 hours. Take full advantage of the reference scripts if you need help in writing your code.

Objective for Lab C\_1: Learn to run Python commands

Objective for Lab C\_2: Create and Use Variables

Objective for Lab C\_3: Create and Use Functions

Objective for Lab C\_4: Create and Use Flow Control Statements

Objective for Lab C\_5: Work with Date and Time variables

Objective for Lab C\_6: Work with Data

Objective for Lab C\_7: Visualizing Data

Objective for Lab C\_8: Skills Project (Optional)

### Exercise 1: Running Commands

The objective of this exercise is to learn to run basic Python commands and get information about objects. You will be able to work with lists, get help, print information and import modules. Reference **C:\Labfiles\Python\LabC\_1.py**.

1. Import the numpy module
2. Get information about the numpy module using `dir()`
3. Use the print function to send a sentence to your display screen
4. Create a variable and assign an integer value to it using the input statement
5. Create three variables that store an array, list and dictionary

**Exercise 2: Creating Variables**

The objective of this exercise is to learn how to create variables, arrays & lists. You will also learn how to modify the data values in them and their data types. Reference **C:\Labfiles\Python\LabC\_2.py**.

1. Import the numpy module
2. Create two integer variables and two string variables
3. Use a third integer variable to store the sum of the first two
4. Use a third string variable to store the concatenated value of the first two
5. Use the print() and type() functions to get information about your variables
6. Create an array and a list that store the same integer values.
7. Print the array and list
8. Add new values to the list
9. Create a new array that uses the values from the updated list
10. Print the new values of the list and the new array
11. Multiply the list and two arrays by an integer value
12. Print the new values of the two arrays and list

### Exercise 3: Creating Functions

The objective of this exercise is to learn how to create a function. You will learn how to use the return statement, input parameters and global variables. Reference **C:\Labfiles\Python\LabC\_3.py**.

1. Create and run a function that print the value of a preconfigured variable
2. Create and run a function that prints the value of a variable defined in the function itself
3. Create and run a function that prints a value input from the keyboard
4. Create a function that prints and returns a value assigned to a variable

**Exercise 4: Use Control Flow Statements**

The objective of this exercise is to learn how to create if, for & while statements. You will also learn how to use the break and continue clauses in looping operations. Reference **C:\Labfiles\Python\LabC\_4.py**.

1. Create an if statement that prints a statement verifying the value of a variable
2. Create an if statement that uses the else condition
3. Create an if statement that uses the elif condition
4. Create a while statement that prints the changing values of a variable
5. Create a while statement that uses the break statement
6. Create a while statement that uses the continue statement

### Exercise 5: Working with Dates and Times

The objective of this exercise is to learn how to use the datetime module. You will create variables and lists that store date-based information. Reference **C:\Labfiles\Python\LabC\_5.py**.

1. Import the datetime module
2. Assign today's date to a variable
3. Assign today's date to a variable in the form of a string
4. Create a list that stores 30 consecutive dates with today being the last
5. Print the list

**Exercise 6: Working with Data**

The objective of this exercise is to learn the basics of using the pandas module. Students should be able to create and manipulate datasets, export/import data to/from text files. Reference **C:\Labfiles\Python\LabC\_6.py** and **RandomizedDataset.py**.

1. Import the pandas & os modules
2. Create 6 lists that each store 5 values for first names, last names, ages, phone numbers, email addresses and id numbers
3. Create a DataFrame for each of the 6 previous lists
4. Merge the data into a single DataFrame
5. Sort and index the DataFrame
6. Change the working directory to a non-system folder
7. Export the data to csv and json files

### Exercise 7: Visualizing Data

The objective of this exercise is to learn how to visualize data using matplotlib. Students should be able to create datasets, customize charts and save chart images. Reference **C:\Labfiles\Python\LabC\_7.py**.

1. Import the os, pandas, pandas\_datareader, datetime and matplotlib modules
2. Use pandas.DataReader to download three stock values for one year
3. Change the working directory to a non-system folder
4. Export the data for each stock to a separate csv file
5. Read "Date" and "Closing Price" columns from each csv file and assign them to variables
6. Use the variables to plot a line chart
7. Create x and y axis labels
8. Save and Display the chart



**Exercise 8: Skills Project (Optional)**

The objective of this exercise is to apply the skills learned in the previous seven lessons. Lessons from each of the exercises will be included. Reference **C:\Labfiles\Python\LabC\_8.py**.

1. Create a function that generates a dataset with information for an orders table with randomly generated data. It should have at least 10,000 records and contain the following columns:
  - Order ID
  - Customer ID
  - Price of Item ordered
  - Quantity ordered
  - Date of Order
  - Save the file in a CSV format with the name ordersdata.csv.
2. Extract 1,000 rows from the ordersdata.csv file with only the following columns:
  - Order ID
  - Price
  - Quantity
  - Total (Price \* Quantity)
  - Date of Order
  - Save the file in a CSV format with the name orders.csv
3. Create a list that stores 60 dates starting from the current date.
4. Visualize stock data for the three largest utility companies for the last two years. Export the stock pricing data to CSV files and save the chart as a PDF file. Include the date in the file names so you have a record of when the data was tabulated.

