These are class notes to complement the daily instruction in class. In here, you will find many of the concepts explained and clarified. (These are written with the help of students – past and current.)

O&G Digital Foundations Handbook

Full Course Notes

Ram Narasimhan

Short Table of Contents

Detailed Table of Contents

Digital Foundations

Full Class Notes

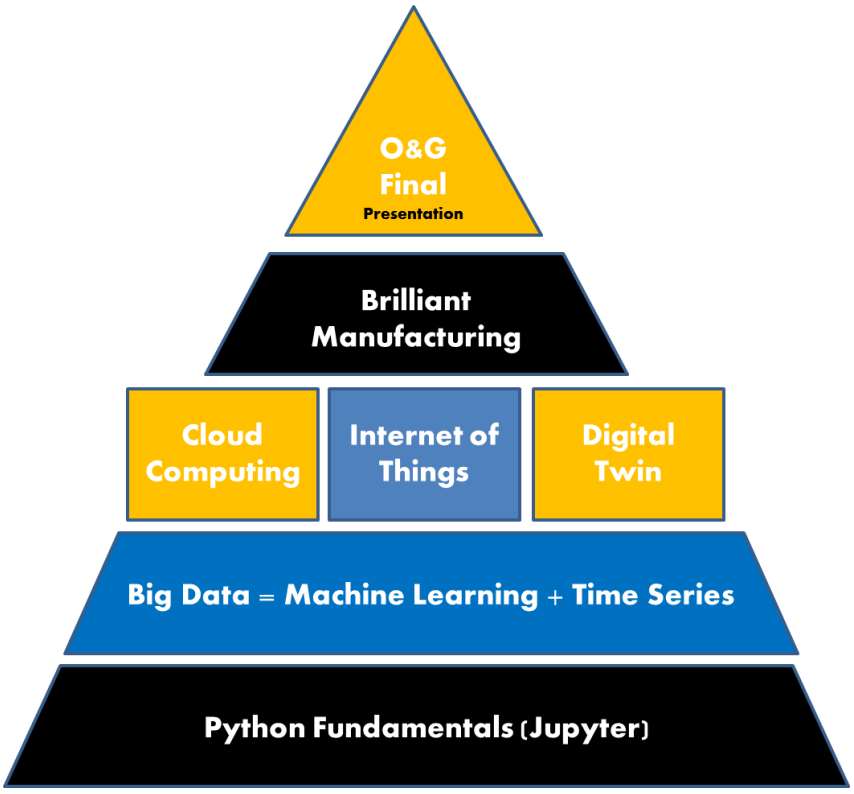
March 2019

Notes and Questions

Python Related

Math and Statistics Related

Digital Concepts Covered



1 Chapter 1 Intro to Python and Jupyter

Topics Covered in Chapter 1

- Introduction to Jupyter
- Taking Python for a test drive
    - Introduction to Python
- Basics of Python
    - Arithmetic Operators
    - String Operations
    - Type Conversions
- Lists in Python

1.1 Jupyter Basics

Jupyter is the interface for Python – This talks to the python 3.6 kernal (or source code) and python includes other modules that we can import.

- **What is Juypter**
    - A Juypter notebook is an interactive document containing code, output, text, plots and images. A notebook is saved in a file with the **.ipynb extension** which is a plain text file storing a JSON (JavaScript Object Notation) data structure.
    - In the Juypter notebook, the dashboard has three (3) tabs which are **Files** (shows files & notebooks), **Running** (shows all kernel open) and **Clusters** (parallel computing can take place).
- **Cells**
    - These are the basic blocks of a Jupyter Notebook. Inside each cell, you can enter text (Python Code) and when you execute the code (Run it), you will see the cell output.
    - All Python cells (code cells) have a sequence number.
    - Each cell runs by itself
    - A cell is one Jupyter unit.
    - When a cell is busy this mean the programme is still calculating an output
- **Key steps before starting to use Juypter**
    - To check the version of your system, run the following steps in separate cells: **cell 1: import sys ; cell 2: sys.version**
    - To check where the directory is to obtain the jupyter files, run the following steps in separate cells: **cell 1: assert sys.version_info >= (3,0) ; cell 2: import os ; cell 3: print(os.getcwd())**
    - To check how many files are stored in the directory run the following action : **os.listdir()**
- **Types of Cells**

1. **Code Cell:** This writes your software code which is to be executed by the Kernel. This type of cell does the coding and processing. The markdown do not have the "In" and "bracket". The programming language => the kernel's language.
2. **Markdown Cell**: It is rich text. It is a decorator eg. bold, color, headings, etc. It is mostly English or images. In addition to classic formatting options like bold or italics, we can add links, images, HTML elements, mathematical equations etc.

- **Type of Modes**
  0. **Edit Mode (Green Border):** To write code and the pen icon appears
  1. **Command Mode (Blue Border):** To operate on cells. It has a grey border and operates on the cell.
- **Kernel-** A **kernel** is a process running an interactive session. When using Jupyter, this kernel is a Python process. There are kernels in many languages other than Python. In Jupyter, notebooks and kernels are strongly separated. **A notebook is a file, whereas a kernel is a process.** The kernel receives snippets of code from the Notebook interface, executes them, and sends the outputs and possible errors back to the Notebook interface. This is what Jupyter communicates with.
- Logical units is kept in one notebook
- **Types of Languages**
  0. **Compile-** Run the entire programme
  1. **Python**- Interpreted language. It runs line by line.

- Jupyter Notebook – "editor" for language.
- A notebook is saved in a file with the extension with the .ipynb extension
- Answers to all the exercises can be found on the web.

Jupyter Notebook

- Each notebook is self-contained.
- Star (figure) means the cell is working, if takes a long of time press stop.
- Green cell means active cell.
- Blue cell means selected cell.
- IPython vs Jupyter??? More on this?
- "Green" cell implies it is running. "Black" cell implies it is "sitting".
- In Jupyter, the color of the words are related to commands (green and black).
  - Green is a usually a Python keyword
- All Python cells (code cells) have a sequence number.
- 2 main types of cells: Code Cell & Markdown Cell.
- Use Help => User Interface Tour
- You don't need the internet to run kernels in Jupyter Notebook

Jupyter Shortcuts

- “Esc” + “m” = markdown cell
- “Esc” + “y” = to change from markdown to code
- “Esc” + “x” = delete cell
- “Esc” + “b” = create new cell below
- “Esc” + “a” = create new cell above
- “c” (highlighting entire cell) = to copy
- “v” (highlighting entire cell) = to paste
- “Ctrl” + “s” = to save
- Escape + m ; to convert a code markdown
- Escape + b ; to create a new line below
- Escape + a; to create a new line above
- Escape + x ; to eliminate lines
- Escape + c ; to copy a line
- Escape + v ; to paste a line

- Jupyter Notebook – “editor” for language.
- A notebook is saved in a file with the extension with the .ipynb extension
- Answers to all the exercises can be found on the web.
- “Green” cell implies it is running. “Black” cell implies it is “sitting”.
- All Python cells (code cells) have a sequence number.
- 2 main types of cells: Code Cell & Markdown Cell.
- IPython vs Jupyter??? More on this?
- Use Help => User Interface Tour
- You don’t need the internet to run kernels in Jupyter Notebook
- ***Note: It is not necessary to get access to the internet to use Jupyter.***

1.2 Starting out with Python

- Python is the language of choice.
- In Python, "#" is a comment indicator, meaning anything after "#" will be ignored for that line and not the next line(s).
- Python is case-sensitive, i.e. OIL is not the same as oil. Places is not the same as places.
- Download Iphone app – Steps; and Moment. Download Android app – Bandicoot.

You can check your Python version at the command line by running `python --version`.

1.2.1.1 Introduction

- Python is the language of choice.
- IDE (interactive development environment)

1.2.1.2 Downloading and Installing Anaconda (and Python)

Python Basics

- In Python, "#" is a comment indicator, meaning anything after "#" will be ignored for that line and not the next line(s).
- Python is case-sensitive, i.e. OIL is not the same as oil. Places is not the same as places.

- Python is an interpreted language so the commands can be typed and executed line by line.
- In Python, anything after # is ignored (that is not executed) because it is a comment and it is case sensitive.
- White space is important.
- Indentation- It can be used to distinguish a block of code.
- Len- shows how many symbols are in the string and how many elements are in the list.
- One cell can have only one output.
- Int- drop the decimal value. Note, you cannot find integer of a string. It must be converted to a float first.

1.3 Basics of Python

1.3.1 Arithmetic Operations in Python

Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	a - b will give -10
*	Multiplication - Multiplies values on either side of the operator	a * b will give 200
/	Division - Divides left hand operand by right hand operand	b / a will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a will give 0
**	Exponent - Performs exponential (power) calculation on operators	a**b will give 10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	9//2 is equal to 4 and 9.0//2.0 is equal to 4.0

Figure 1 Arithmetic Operations in Python

http://www.emcu.it/Python/Python.html

- 27//4 = 4. The "//" cuts off any other numbers after 4 (the answer should have been 4.5).
- 2**3 = 8. That is, $2^3 = 8$.
- 25%7 = 4. That is 3x7 = 21; 25 – 21 = 4.
- BODMAS (Hierarchy/Priority of calculation). E.g. 5 – 4 * 3 = – 7. Use brackets to remove confusion. Should be written as 5 – (4 * 3) = – 7.
- It is cleaner coding to use () to complete different math functions.

- **floor division -** Mathematical division that rounds down to nearest integer. The floor division operator is //. For example, the expression 11 // 4 evaluates to 2 in contrast to the 2.75 returned by float true division. Note that (-11) // 4 is -3 because that is -2.75 rounded *downward*.
- // will truncate the division (integer division), aka. Quotient (29//6 = 4 or 15//7 = 2)
- % shows the remainder/modulus (opposite of //) (25%7 = 4)
- ** indicates power (exponential) (2**3 = 8)
- Python follows BODMAS

1.3.2 String Operations in Python

Strings in Python

- Are a sequence of more than one characters which can be letters, numbers punctuation, and other symbols
- " " This is a string because of the two quotes.
- A string is not a number
- Exercise explained: round(3.223883432, 4) : This mean round off to four decimal places.
- **Note: For print, int, float, range, pop, reversed, sorted, pow, and append the ( ) brackets are used.**

3 ways of writing strings:

- In python you can use both (single and double quotes).
- Single quotes ('Hello')
- Double quotes ("Hey there")
- Use of triple quotes (""""this is a very long multi-line String""")

- In order to close a string the initial and the final quotations have to be the same.

- Single, double and triple quotes can be used. Why we used these difference ways is because one type of quote can be used inside another.
- The length of the strings considers all the characters (including spaces, period, commas).

Note that a string is not a number! ('3' and 3 are not the same in Python)

1.3.3 Type Conversions

Converting strings to float

- Text- String: To convert to a string: eg. str (11) would become str('11').
- Float-numbers. **Research: Float** is a term used in various programming languages to define a variable with a fractional value. Numbers created using a float variable declaration will have digits on both sides of a decimal point. This is in contrast to the integer data type, which houses an integer or whole number.
- A list of strings must be converted to a list of numbers in order to perform an action. Run the following example:
    - oil_temp = ["1", "2" ,"3"] then let
    - oil_temp = [float(x) for x in oil_temp]
    - print (oil_tem)
    - Note, for float, .0 is added to the number if it doesn't have a value. For example, if the string is "83" then the number would be 83.0 when converting to a float.

To round off use: round (value to be round off, #of decimal place) eg. round(6.580600, 2) would become 6.58.

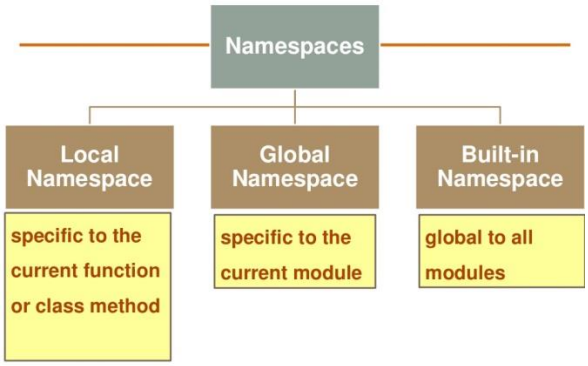
- str(x)
- float(x)
- int(x)

1.4 The idea of Modules

Using Math Functions

- import math
  from math import sqrt
  from math import *

1. What is the difference between:
   - import math: content is pulled from a module, needs to be recalled every time
   - math import*: content is locally stored, done once



2. What is a module? A list of functions that has been prepared by users in the Python community and is available for use in an open source platform.

An alias is a convention

- Using an alias = import math as m (then use m instead of math)

```
Import pandas as pd
```

- is common nomenclature

Questions

3. round(3.332,2)? result 3.33

round(3.5,2)? result 3.5

4. What is the difference between round() and //? // rounds down

1.5 Getting Help

Tips of the day:

- Tip of the day: Press "*Tab*" to populate a list of possible functions available (this will give you the options for the remaining text.
- Tip: To see all of the python os functions

  import os

  dir(os)
- Tip: To see the directory where you have saved the your files (if you need to email/share/send the file)

  !cd
- Tip: In Jupyter, double click the left side of output to hide it.
- Tip: String multiplied/divided/subtracted by a string is error and strings can only be added.
- Tip: *help(print*) will list a good description of print

1. **import** math
   math.
   This will automatically fill the dropdown list. (use the tab button)
2. Built-in functions: useful not to be duplicated by the user

Using Math Functions

```
import math
from math import sqrt
from math import *
```

- Each notebook is self-contained.
- ***Import*** Only works for each notebook

1.6 Data structures in Python

Python includes several built-in container types: lists, dictionaries, sets, and tuples.

1. List : list [1, 2, 3] Ordered collection This is a fundamental structure in Python
2. Tuple : tuple (1, 2, 3) Immutable ordered collection
3. Dictionary : dict {'a':1, 'b':2, 'c':3} Unordered (key, value) mapping
4. Set: set {1, 2, 3} Unordered collection of unique values

1.7 Lists in Python

A Python list is a "container" that holds an array of items in sequence. These items have indices which can be used to access them. A Python list is 'mutable' (that is, it can be mutated) by adding, removing and replacing items.

To define a list, we use square brackets and each item is separated by a comma.

List_a = [12, 14, 15, 'abc']

10 things about Python Lists

1. Lists are ordered.
2. Lists can contain any arbitrary objects.
3. List elements can be accessed by index
4. List indices are integers
5. Lists are mutable.
6. You can have a list within a list
7. Strings are Python lists of characters
8. The last element of a list can be accessed with index -1
9. Any list has its length defined by len()
10. Python list indexes start with 0, 1, 2... (the first element is at index 0)

```
xs = [3, 1, 2]    # Create a list
print(xs, xs[2])  # Prints "[3, 1, 2] 2"
print(xs[-1])     # Negative indices count from the end of the
list; prints "2"
```

```
xs[2] = 'foo'     # Lists can contain elements of different types
print(xs)         # Prints "[3, 1, 'foo']"
xs.append('bar')  # Add a new element to the end of the list
print(xs)         # Prints "[3, 1, 'foo', 'bar']"
x = xs.pop()      # Remove and return the last element of the list
```

```
print(x, xs)      # Prints "bar [3, 1, 'foo']"
```

1.7.1 Interacting with lists

- Note- List has a beginning and an ending, eg. List[begin:end]. Therefore, it would list up to but not including the end.
- If the length of the list is unknown, and you want to access the last two elements use list_name[-2:]
- To list every elements in the slice use list_name[0::1] where the 1 represent in steps of 1.
- To list every other elements from 1 to infinity use list_name[1::2]
- To list elements in steps of three use list_name[::3]
- To list the element starting with element 2 and going to the of the list in steps of three use list_name[2::3]
- To replace an element in the list use list_name[element position], eg. list_name[4] = Apple would replace the 4th element with Apple
- To add an element to the list use list_name.append("Apple"). This would add the element apple to the end of the list. Also, + can be used
- To find the length of the list or the number of element in the list use len(list_name)
- To find the ranges of the list use list(range(12)) ,where 12 is whatever is the maximum value you are looking for to get a range.
- pop([i]) - Remove the item at the given position in the list, and return it
- Try getting the min() and max() of a list:

```
min(list_a) and max(list_a)
```

1.7 Class Exercise

1. places = ["St Lucia", "Greece", "Antigua", "New York", "Japan"]

   print(places) #to print the list

2. print(len(places)) #to count the number of places
3. list(reversed(places)) #to reverse the list of places
4. list(sorted(places)) #To sort the list of places in Alphabetical order
5. To remove the 4th element use del places[3]

1.8 Dictionaries in Python

Just like lists, dictionaries are another data structure. Dictionaries are used for storing pairs of values (key-value pairs that are known as items).

Dictionary - A built-in Python data type composed of arbitrary keys and values; sometimes called a "hash" or a "hash map" in other languages. Examples:

```
d = {'A':65, 'B':66}
d['C'] = 67
```

- A dictionary has a name. for example:

```
elements = {'Hg':'Mercury', 'Ag': 'Silver'},
```

In the line above, 'elements' is the name of the python dictionary.

- Python Dictionaries are always contained within curly braces {key1:value1, key2:value2}
- To create an empty dictionary, simply do the following:

```
d3 = {} # d3 is the name of the dictionary (currently, it is empty)
```

- Dictionaries are unordered, unlike a List.
- A key and value pair is called an 'item'
- To add any item, simply say:

```
d3['newkeyname'] = 1234
```

- They cannot contain two identical keys
- If you try to add the same key twice, it will simply overwrite the value.

```
d3['newkeyname'] = 1235 #1234 is gone, overwritten
```

Order of the dictionaries have no meaning (this is different to lists)
Dictionaries are made up of a key and a value, this is a Key:Value pair or item.

1.9 Questions from Chapter 1 Material:

1.9.1 Day 1: Commonly Asked Questions by students

1. How do we get to the Home screen from any Jupyter Notebook?

2. What is a module? Where does it come from and what the process and schematic?
3. What is the difference between a string and a float?
4. Can we access Jupyter outside our laptops?
5. Where are our Notebooks Stored?
6. What happens if I multiply a string by a number? String by string?
7. How to navigate back to the home screen?

   Answer: Click the Jupyter Logo at the top left corner

   - Why use Jupyter and not go straight to Python/Anaconda? Jupyter excellent for teaching and learning plus it supports other languages.

2 Chapter 2 Getting to Know Python Better

Concepts covered here include: Variables, Control of Flow (Conditional if, elif and for & while loops). We are also introduce Python Dictionaries. Finally, we look into writing our own Python functions.

2.1 Variables

Variables

- **A variables is a placeholder used to refer to values**. Therefore, variables can change their values and name.
- x can be a variable but 3 is a constant
- Can be any length: But only made up of letters, numbers or underscore (_)
- The First character of a variable name cannot be a number
- Python is case-sensitive. X and x are two different variable names.
- **You cannot use any Python keyword as a variable name, eg print**

A temporary placeholder where values are stored and are changed
Assigning variables: in these two examples the user determines the values
x=3
name=Ali

Note: case sensitive, first character cannot be a number, cannot use any Python keyword as a variable name, can be any of length (only made of letters, numbers, underscore)

- Variable types = string, integer, float, and Boolean
  - To view the data type use "type(x)"
    - E.g. x=3 (type:int) x="3" (type:str) x=3.8 (type:float)

- **Rules of giving Variable Names**
-
- A variable is a placeholder to refer to values (e.g. x=3 or Name=Ali)
  - Is an assignment statement
  - Can have letters, numbers, and _ (cannot start with a number)
  - A Python keyword should not be used as a variable name (Python keywords turn green)
  - Index variable "for i in range (1,9)" – the variable is i (can be any letter)

- Can be any length: But only made up of letters, numbers or underscore (_)
- The First character of a variable name cannot be a number
- Python is case-sensitive. **X** and **x** are two *different* variable names. (Be careful!)

- You cannot use any Python *keyword* as a variable name

- A variable is a placeholder in where we store values in Python. Variables can change their values (hence the name). Variables cannot start with a number and are case sensitive. Python keywords cannot be used as a variable.
- Assignment statement is assigning a value to a variable (*x = 4*)
- You can assign multiple variables in one line

  x, y, z = 1, 2, 3
  print(x,y,z)

2.2 Control of Flow

- Flow of control using if statements and loops (jumping to different parts of the code)
  - If, elif (if not), and else (otherwise)
  - 2 types of loops = for loops (used 90% of the time) and while loops
  - Cannot have a for loop without anything in the block – indentation error
- Boolean Logic
  - True and false values
  - Keywords and, or, and not
  - +, <, >, <=, >=, ==, !=
- Indentation is required for statement blocks
  - Anything that is not indented will not be considered as part of the block
  - Python indentation is 4 spaces (or Tab – Jupyter will automatically give 4 spaces)

Flow control uses Boolean operations
== is equal to
!= is not equal to

"If statements" use indentation between if and else, there is no requirements to close the parenthesis. Indentations will cause an error if not used correctly.

Yes, we use if statements inside other if statements?
Will python identify indentation errors within your code? Yes

Loops – *For* loops and *While* Loops are the 2 types of loops
i is part of the loop index

2.2 Writing Functions in Python

Functions are objects, defined with the 'def' statement followed by argument list. The function body is defined using indentation like other python statements.

Some functions will return useful values, others may return nothing.

It is important to remember that keywords must come at the end when using both non-keyword and keyword arguments.

When creating a function you use the prefix:

```
Def name_of_function:
```

When building a function the end of the function is defined by *return.* The function will not look at the code past the return line.

Note: keyboards are named arguments; Non keyboards are un named arguments

return() (close the function).

2.2.1 Reference for Functions

This document on writing functions in Python is very good.

http://devarea.com/python-writing-functions-complete-guide/#.Wo0SxyXwa00

2.3 Starting out with Python

- 5 layers of architecture
  - Application
  - Visualization
  - Analytics
  - Data
  - Server

2.4 Questions asked by students

1. Why shouldn't we use both the variables x and X?
2. How do end the if statement?

3. What happens if we indent the last print within an if statement
4. Is python going to tell us we have an indentation error?
5. How did we define the variable i here: **for i in range(8)**
6. Assuming we do not use \tab, what's the problem with that?
7. How can I view and use someone else's function? Source code?

3 Chapter: Using the Pandas Library

3.1 What is Pandas?

pandas is a Python package providing fast and flexible data structures designed to make working with data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

pandas is a software **library** written in Python. It is mainly intended for data manipulation and analysis. In particular, it offers two things that are useful for us:

1. data structures and operations for **manipulating numerical tables**
2. Ability to work with and manipulate **time series**.

'pandas' has functionality that is similar to (and often faster) than Excel. It can also handle "big data." Pandas is the toolset of choice for data scientists and professionals in the Data Analytics world that know Python.

3.2 Pandas – Absolute Basics

- Pandas module – Pandas is a Python package that provides practical, real-world data analysis
  - There are two panda structures = series (1-D) and dataframe (2-D or more)
- df.head() will print by default the top 5 rows of the imported dataframe
  - This is good to check if the dataframe has been imported correctly.
  - df.head(10) will display the top 10 rows
  - df.tail() will print by default the bottom 5 rows of the imported dataframe
  - df on its own will display all rows and columns
- df.shape = (number of rows, number of columns) e.g. (58788, 24)
- df['column name'] will display a specific column
  - Within a column we can investigate the mean or median of our data by using: df['column name'].mean() or df['column name'].median()
  - Having a low median means that the data is highly skewed
- Using conditions:
  - Condition=df['year']==2000
    df[condition]['comedy] – these two conditions will show us comedies in the year 2000
  - df.[condition]['comedy].sum() will give the sum
- To find the max value within a column use df.columnname.max()
- df.describe() will give information about the dataframe

- To create a histogram df.hist("columnname", bins=10)
  - Bins refer to the number of bars to display the data (also known as buckets)
- To create a line plot df.plot(x="time", y="columnname") – call the plot function then specify x and y
-
- Library is also known as Package
- Two important starting points – probability and linear regression
- To view what files are in a directory "import directoryname" then "directoryname.listdir()"
- To create a dataframe: df.pd.dataframe ({"list name":[a,b,c]})
  - For multiple columns: df.pd.dataframe ({"col1":x, "col2":y, "col3":z}) – key:value pairs
  - The value must be a list (x y and z in above examples are lists)
- To save and select specific columns: useful_cols=["col1","col2"] then can use df.[useful_cols]

Pandas has two fundamental data structures.

1. A Pandas Series
2. A Pandas DataFrame

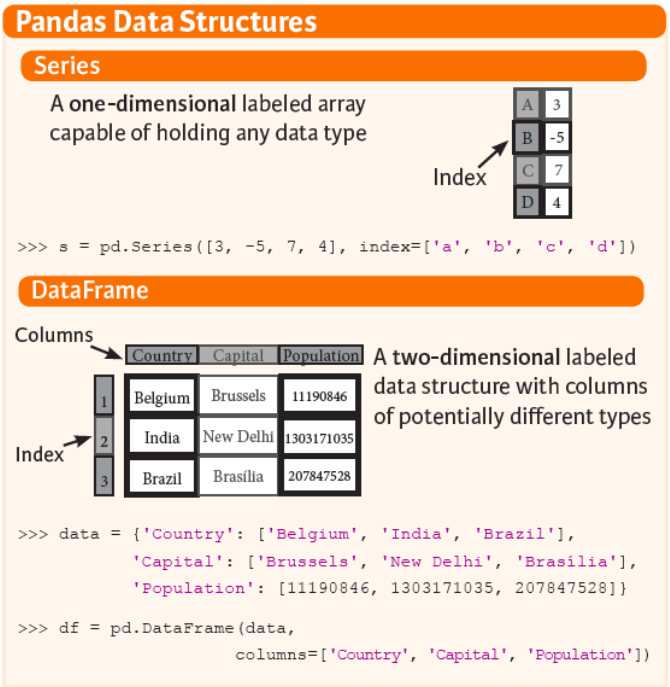
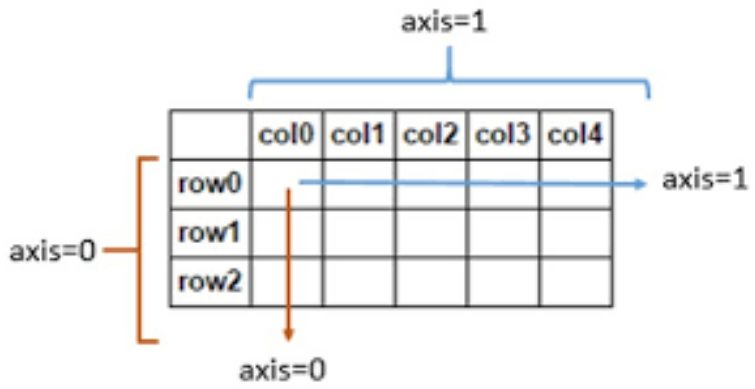


Figure 2 The two fundamental data structures of Pandas - Series and DataFrame



For simplicity, we can think of
INDEX = ROW NAMES (ROW LABELS)
COLUMNS = HEADER (COLUMN NAMES)

A data frame has the shape property: shape() is just the number of rows and the number of columns in a data frame.

```
In [37]: df1
```

Out[37]:

	GEOID	State	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	04000US01	Alabama	37150	37952	42212	44476	39980	40933	42590	43464	41381
1	04000US02	Alaska	55891	56418	62993	63989	61604	57848	57431	63648	61137
2	04000US04	Arizona	45245	46657	47215	46914	45739	46896	48621	47044	50602
3	04000US05	Arkansas	36658	37057	40795	39586	36538	38587	41302	39018	39919
4	04000US06	California	51755	55319	55734	57014	56134	54283	53367	57020	57528

Figure 3 A sample data frame showing the "index" and "columns"

3.3 "Interrogating" Data using Pandas

3.3.1 Things that can be done with a Series (or a Data Frame column)

- **What can you do with Columns?**
- # of Unique values
- Counts (Frequency of each unique value)
- Sort By Column (Ascending, Descending)

- Count Number of Missing Values
- Calculate Means for each column
- Add two columns
- Advanced: Apply any function to any column

3.4 Operations that can be performed with a Data Frame

df = data frame

1. # Rows and columns
2. Examine (Head and Tail) of the df
3. Combine two data frames
4. Merge two data frames
5. Describe() each of the columns
    a. Statistics about the column
6. Statistics [mean, count, max, min, median]
7. Filter, Sort and GroupBy

Note: You don't have to remember all these commands. I use them a lot, but I don't remember the commands! I just look them up. The goal is to understand the *concepts*, don't worry too much about the *syntax*.

3.5 Loading and Viewing Data Frames

3.5.1 Selecting Data Using Boolean Masks

- SELECT only rows that follow a particular condition
    - Very powerful!
    - This is where we end up using Boolean masks.

4 Chapter: Plotting – Visualization Using Python

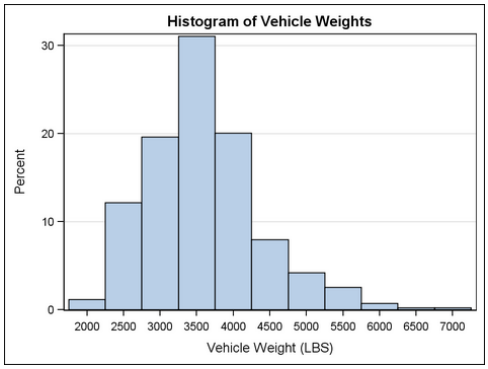
4.1 Types of Plots

Visualization is a vast topic, with several good tools available these days. For our purposes, we'll be focusing on 4 distinct types of plots. These are also the most commonly used plots in business, especially for day-to-day use.

They are:

1. Histograms
2. Bar Plots
3. Scatter Plots (sometimes called XY Plots)
4. Line plots (useful for plotting Time Series data)

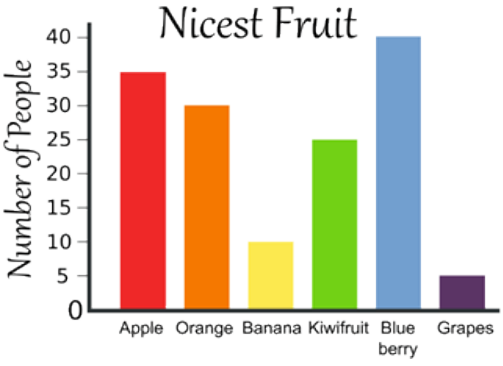
4.1.1 Histogram



A Histogram is a plot that lets you discover, and show, the underlying frequency distribution (shape) of a set of continuous data. A graphical display where the data is grouped into ranges (such as "40 to 49", "50 to 59", etc), and then plotted as bars. (Not to be confused with Bar Plots, which are for Categories.)

One important aspect of histograms (which many people seem to miss!) is that histograms are for only one variable at a time.

4.1.2 Bar Plots



A bar chart or bar plot is a graph that presents categorical data with rectangular bars with heights or proportional to the values that they represent. It shows the relationship between a numerical variable and a categorical variable. For example, you can display the height of several individuals using a bar chart.

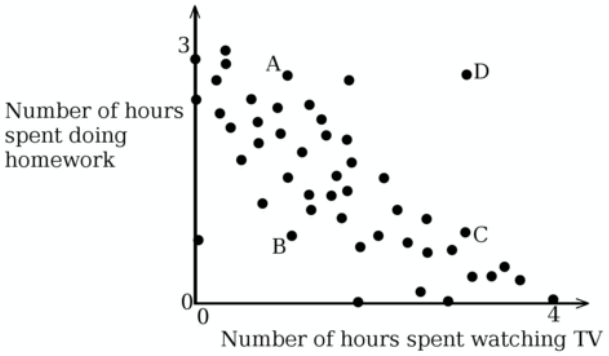
4.1.3 Comparing Histograms and Bar Plots

- Histogram vs. bar charts

Histogram	Bar chart
y-axis almost always frequency	y-axis is a number
1 column/variable	2 columns/variables
Variable type is continuous	Variable type is categorical for x and number for y
Does not contain gaps	Contains gaps
No colours	Has one colour for each category
x-axis contains a number range	x-axis contains categories

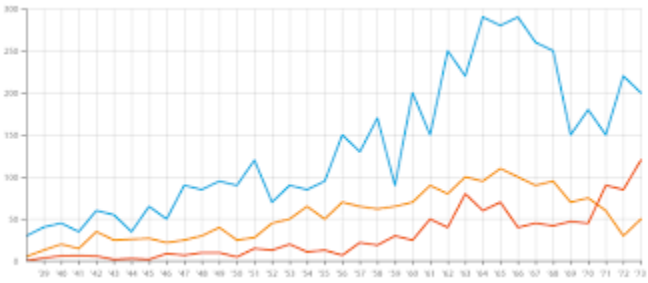
4.1.4 Scatter Plots

A scatter plot is a graph in which the values of two variables are plotted along two axes. These are sometimes referred to as X-Y plots. They are very common and especially suited when you have two continuous variables and are interested in seeing their relationship for each observation.



4.1.5 Line Plots

Also known as line charts, these are graphs in which the data points are placed on their XY position, and connected by straight lines. In many line plots, the x-axis is usually time (calendar), and thus these are also referred to as *time series plots.* One main purpose of time series charts is to help visualize trends over time.



In the next section, we will look at how to plot all these types of plots, using Python.

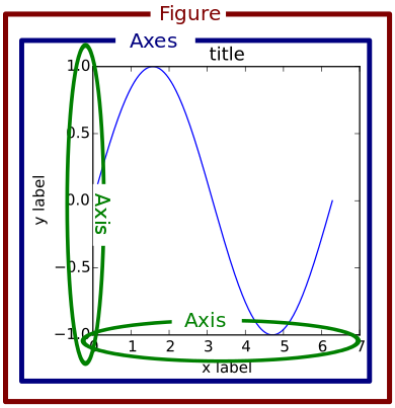
4.2 Plotting Using Python

Now that we know a little about these 4 types of plots, let's see how to make them using Python. For this, we will be using 3 different plotting modules.

1. Matplotlib
2. Seaborn
3. Pandas

Note: Depending on how your data is laid out, and the plot that you are after, Pandas or Seaborn or Matplotlib might be the easiest. It is best to look at the Web for examples and get ideas for what you are looking for. (I use all three libraries for my work to create the plots I need.)

4.2.1 Plotting using Matplotlib



Matplotlib has careful definitions for Figure, Axes and Axis. Once you understand what these are, a lot of plotting using Matplotlib is easier. One Figure can contain multiple Axes. Each Axes has its own 'handle.' And each Axes has the 2 axis (x and y). As one stackoverflow answer explains it nicely, "**Axis** is the axis of the plot, the thing that gets ticks and tick labels. The **axes** is the area your plot appears in."

```
fig, ax = plt.subplots()
```

`plt.subplots()` is a function that returns a tuple containing a figure and axes object(s). Thus when using `fig, ax = plt.subplots()` you unpack this tuple into the variables `fig` and `ax`. Having `fig` is useful if you want to change figure-level attributes or save the figure as an image file later (e.g. with `fig.savefig('yourfilename.png')`

The easiest way to use these is to get working code from the Web, and modify them for your needs.

Histograms Using Matplotlib

```
x = [1,3,3,4,1,4,1,2,5,5,7,7,4,2,5,6,4,3,3,6,8,9]
plt.hist(x);
```

Barplot Using Matplotlib

For a barplot we need two columns (lists). One should be numerical (the y-axis) and the x-axis would be discrete (categorical).

```
fruits = ['Apple', 'Banana', 'Orange', 'Kiwi'] #Data
x_pos = [0,1,2,3]
count = [4,8,6,4]
plt.bar(x_pos, count, align='center', alpha=0.5)
```

Creating Scatterplots using Matplotlib

Matplotlib has a built-in function, called 'scatter()'. It plots the X an Y values for each data point.

```
x = np.random.rand(100)
y = np.random.rand(100)

# Plot
plt.scatter(x, y, alpha=0.5)
```

You can also have your data grouped, and use a different color for each group. For that, you have to get the plot's *axes* and plot in the same plot again and again for each group.

See https://pythonspot.com/matplotlib-scatterplot/ (Second example shows groups)

Line charts using Matplotlib

Again, as a reminder, A line chart or line graph is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments. This code from this article is very clear and easy to understand.

```
year = [1960, 1970, 1980, 1990, 2000, 2010]
pop_pakistan = [44.91, 58.09, 78.07, 107.7, 138.5, 170.6]
pop_india = [449.48, 553.57, 696.783, 870.133, 1000.
4, 1309.1]

plt.plot(year, pop_pakistan, color='g')
plt.plot(year, pop_india, color='orange')
plt.xlabel('Countries')
plt.ylabel('Population in million')
plt.title('Pakistan India Population till 2010')
plt.show()
```

This produces the following plot:

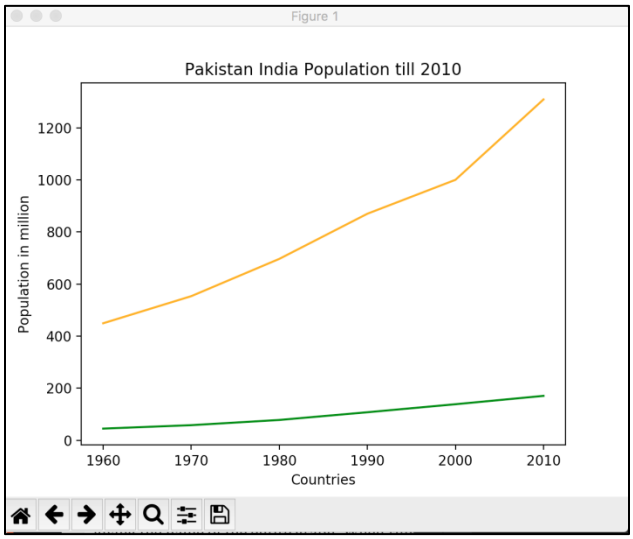


Figure 4 A Line chart with 2 Series using Matplotlib

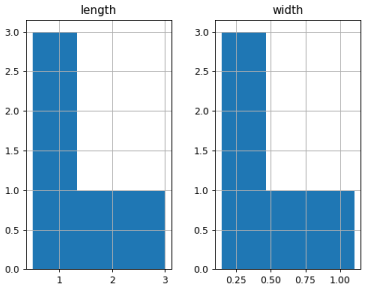
Matplotlib References

- Basic Beginners' Introduction to plotting in Python (Sarah Blyth)
  - http://www.ast.uct.ac.za/~sarblyth/pythonGuide/PythonPlottingBeginnersGuide.pdf
- This is a good tutorial for Matplotlib, if you are planning to use it seriously
  - https://www.labri.fr/perso/nrougier/teaching/matplotlib/

4.2.2 Plotting using Pandas

Since we use Pandas quite heavily for data analysis, it makes sense to learn how to do plotting with it as well. The good news is that Matplotlib and Pandas are very nicely integrated (in certain cases.) Note that we will need a data frame in all the following cases. When plotting with Pandas, we are typically plotting one or more columns in the data frame.

Creating Histograms using Pandas



This is very easy and direct. Just chain a method called 'hist()' to your data frame.

```
hist = df.hist(bins=3)
```

Creating Barplots using Pandas

Again, we invoke the power of Matplotlib.

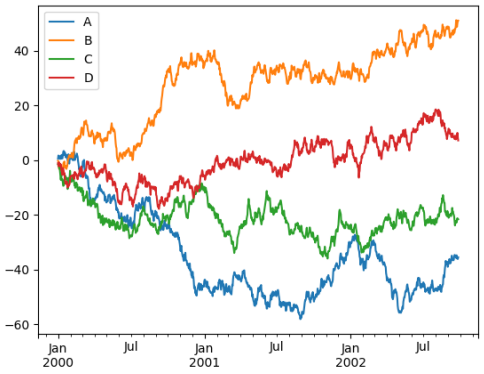
```
df['column1'].plot(kind='bar');
```

Creating Scatterplots using Pandas

Keep in mind that a scatter plot requires both axes (x and y) to be numerical.

```
df.plot.scatter(x='a', y='b');
```

Line charts using Pandas



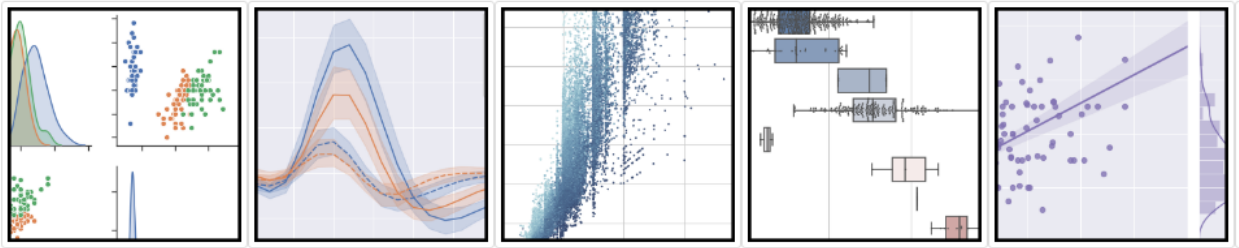
Line plots are the default plots for Pandas. So you can simply call the plot() function. Optionally, you can give it one or more columns as inputs.

```
df.plot()

df3.plot(x='A', y='B')

df[['Temp', 'Pressure_psi']].plot()
#plot multiple columns
```

Pandas Plotting References

- The official Pandas Visualization documentation is quite good: https://pandas.pydata.org/pandas-docs/stable/visualization.html

4.2.3 Plotting using Seaborn



Seaborn is a Python data visualization library based on matplotlib. (https://seaborn.pydata.org/) You can do a few things with it much easier than using the base Matplotlib. For this reason, many students find it easier to learn Seaborn first, and to use Matplotlib for finishing touches to their plots. Seaborn provides a high-level interface for drawing attractive and informative statistical graphics.

The nice thing is that you can first create a Seaborn plot, and add many Matplotlib commands on top of that, to enhance the aesthetic of your graph.

Finally, another nice thing about Seaborn is that it works wonderfully well with Pandas data frames. So you can simply pass it one or two columns, and it will plot it.

Creating Histograms using Seaborn

In Seaborn terminology, histograms are "Distribution Plots" or displots.

```
sns.distplot(data['x'])
```

Creating Barplots using Seaborn

There is a simple and convenient function called 'barplot'. Simply call it and pass the data to it.

```
sns.barplot(x=x, y=y3, palette="deep")
```

Creating Scatterplots using Seaborn

```
sns.lmplot(data=df, x='Column1', y='Column2', size=4, aspect=5)
```

Line charts using Seaborn

This can be done, by directly calling the lineplot() function of Seaborn.

```
sns.set(style='darkgrid')
sns.lineplot(x='Column1', y='Column2', data=df)
```

Seaborn Plotting References

- An excellent tutorial by DataCamp https://www.datacamp.com/community/tutorials/seaborn-python-tutorial
- Kaggle's Seaborn tutorial is quite easy to follow: https://www.kaggle.com/residentmario/plotting-with-seaborn

4.2.4 Plotting Resources

Python Plot for Exploratory Data Analysis (EDA): Nicely laid out set of graphs (with code) for exploring data. A good place to start, if you get stuck. (http://pythonplot.com/)

The Python Graph Gallery: This website displays hundreds of charts, always providing the reproducible python code! It aims to showcase the awesome data visualization possibilities of python and to help you benefit it. (https://python-graph-gallery.com/)

5 Time Series Based Analysis

5.1 How to create Time based Index?

This is quite easy to do. We just identify the time column and assign it as the index!

```
df.index = df['time_column']
del df['time_column']
```

What are the advantages?

Or we could say:

```
df.set_index('time_column', inplace=True)
```

Now, the time_column gets removed and it becomes the index (row numbers) for the entire data frame.

5.2 How to "Resample" data?

```
If the index is a DatetimeIndex(), then we can do:

df.resample('D') #'D' for daily, 'H' for Hourly
df.resample('D')[[col1, col2]]
df.resample('D')[[col1, col2]].mean()
df.resample('D')[[col1, col2]].max()
```

For further details, here is the help on resample:

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.resample.html

5.2.1 Questions Asked by Students

The following are questions that were asked by past students. These questions are a *very* good way of testing our understanding. I have answered each of them in the Appendix, but I urge you to first think about the answer yourself, before reading the answer provided. That is one very good way to learn.

1. When cutting of quartiles on a list, does it take up last 3 or 4?
2. How many bins should you decide on in a histogram?
3. What is Kde?
4. Can we just use a box plot rather than going through all percentile functions?
5. Why does mid diff for box plot? Why not use all values?
6. sns is a library? Where is it from?

6 Chapter: Machine Learning Using Big Data

Chapter's Outline

- An Intro to Probability
- What is ML?
- The main techniques within Machine Learning

- What is Linear Regression?
- How can we measure the "goodness" of a Linear Regression fit?

- What is Classification?
- Dividing data into Train and Test datasets
- How do we measure whether a classifier is working fine?
- What is a Confusion Matrix?

Oil & Gas

- ML in the Oil & Gas Industry
- Where do we use classification in O&G?
- Where do we use clustering in O&G?

6.1 What is Machine Learning?

Definitions:

1. Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed
2. Within the field of data analytics, machine learning is a method used to devise complex models and algorithms for prediction; sometimes referred to as predictive analytics. These analytical models allow researchers to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.
3. Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

    a. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

6.2 The main techniques within Machine Learning

Machine learning algorithms are often categorized as “supervised” or “unsupervised.”

- **Supervised machine learning algorithms** apply what has been learned in the past to new data using *labeled examples* to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces a mathematical function to make predictions about the output values. The system is able to provide targets (labels) for any new input after sufficient training.
    - We can compare the output of the learning algorithm with the correct, intended output and find errors in order to modify the model accordingly.
- In contrast, **unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function *to describe a hidden structure from unlabeled data*. The system explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

6.3 What is Linear Regression?

In Statistics, Linear regression refers to a model that can show relationship between two variables and how one can impact the other. In essence, it involves showing how the variation in the “dependent variable” can be captured by change in the “independent variables”.

Linear regression consists of finding the best-fitting straight line through the points. The best-fitting line is called a regression line. In essence, we are looking for a linear relationship among several variables to predict a “target” variable.

Target Variable is a function of (Predicting variables)

```
Y ~ X1, X2, X3…Xn
Y is the called the “target,” Xi are called the “predictors.”
```

In a SIMPLE LINEAR REGRESSION, we use one variable to predict another.

Simple Linear Regression model: Simple linear regression is a statistical method that enables users to summarize and study relationships between two continuous (quantitative) variables. Linear regression is a linear model wherein a model that assumes a linear relationship between the input variables (x) and the single output variable (y). Here the y can be calculated from a linear combination of the input variables (x).

- When there is a single input variable (x), the method is called a **simple linear regression**.
- When there are multiple input variables, the procedure is referred as **multiple linear regression.**

Linear Regression Using Python

Intent:

- To quantify the relationship between variable and then make predictions .
- To understand the rate of change
- ~ : a function of
- Through Multiple linear regression the response/target/dependent variable (say y) is a function of multiple predictors/independent variables (x1,x2, x3)
- Finding the best fit/line.

- Residual: predicted-actual

- Linear regression
  - 1. Quantify the relationship between variables
  - 2. Make predictions
  - Linear(straight line) equation y=mx+c where m is the slope and c is the intercept/constant
    - y is a function of x (y=fx)
  - Target variable (y) and factors (x)
  - Simple linear model has one y and one x
  - Multiple linear regression has one y and multiple x's (as many slopes as there are x variables)
- It is important that the model is statistically significant, with significant coefficients, and fits the data well
- Residuals (n) is the difference between the predicted and the actual values (y predicted – y actual)

6.3.1 How can we measure the "goodness" of a Linear Regression "fit"?

Intuition: Look at the 'residuals.' By comparing the difference between actual y-value and model's predicted y-value, we can get an idea of the accuracy of the fit.

Idea called "Cross Validation:" Using (Cross) Validation is one way to measure the accuracy of such kinds of predictions. The idea is as follows: Randomly select one or more of your data points which you set

aside and not use to fit the parameters of the model. Then, build your model and given the x-value of the data point(s) set aside, predict its y-value using the model. You can then calculate the prediction error and compare different models. Usually, you calculate the mean of the squared error. Cross Validation is an extension of this idea where you compute several models, say, 5 models, leaving out different 20% of the data points to build each model.

R-squared is the percentage of the dependent variable variation that a linear model explains.

$$R^2 = \frac{Variance\ explained\ by\ the\ model}{Total\ Variance}$$

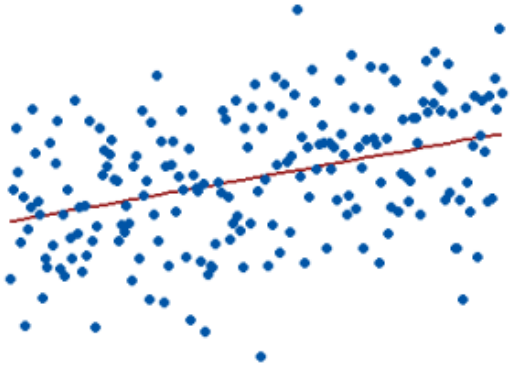


Figure 5 R-square of 0.15 (15%)

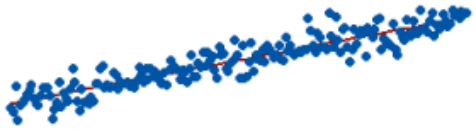


Figure 6 R-squared of 0.85 (85%)

RSME: A measure of the errors in our Linear Model

$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

Y-hat are the predicted values. (Predicted – Actual) are the residuals.
Take the sum of the squares of the residuals, then take square root. Thus, we have one number (RMSE) for the points that we are trying to predict for.

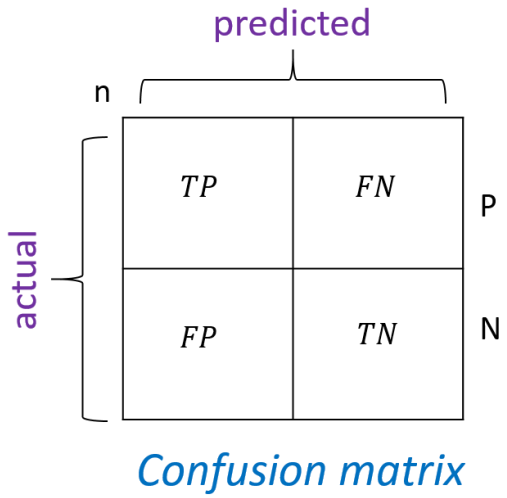
- Check two things
  - Homoscedasticity – constant variance of residuals (errors are uniform – good)
  - Normality of errors – errors are normally distributed
- Module statsmodels (smf) – statistical model function

6.4 What is Classification?

- Dividing data into Train and Test datasets
- How do we measure whether a classifier is working fine?
- What is a Confusion Matrix?

- Machine learning:
  - Giving machines the ability to learn without explicably programming
  - Computer generates rules based on raw data given
  - Inducing patterns, regularities, and rules from past experiences
- A computer "learns" through repetition, you have to "train" the program

6.4.1 What is the "Confusion Matrix?" Why is it useful?



True Positive and True Negatives are correct predictions. A False Positive is where the model (classifier) thinks something is Postive (True) but the actual is False. (Think of this as a "False Alarm")

A False Negative is where the model falsely predicts a True case to be Negative.

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- A confusion matrix is a special kind of table.
- That table helps us evaluate the performance of a Classification Model (classifier)
- We let the classifier classify test data (data it has not seen before.)
- True values (Ground Truth) is known.
- So we can see how well our classifier is performing. Elements are on the Top Left to Bottom Right diagonal are indications of good performance
- Off-diagonal values are “misclassifications.”

6.5 ML in the Oil & Gas Industry

Now that we have seen some general concepts and techniques in ML, let’s shift our focus to the O&G Industry.

6.5.1 Where do we use classification in O&G?

6.5.2 Where do we use clustering in O&G?

6.5.3 Where should we use Linear Regression in O&G?

Example 1: To find the properties of Crude blends, from tables that show the properties at discrete boiling points.

Crude assays nearly always include a true boiling point (TBP) distillation, which consists of crude properties given at discrete boiling points. However, an engineer is often interested in blends with different boiling ranges than those reported.

Using regression analysis, an engineer can get the information that crude assays do not provide.

Example 2: **Prediction of Output in Oilfield Using Multiple Linear Regression**

http://www.ijastnet.com/journals/Vol_1_No4_July_2011/14.pdf

In oil production, there are two major factors affecting oilfield production which are geological factors and human factors. Therefore, these two factors are being considered to predict the output of oilfield. Considering the geological factors, the oil wells are the utmost important element in predicting oilfield's output directly determines the yield of oilfield.

To predict future output of an oilfield, the influencing factors combined with actual production are selected and analyzed. The Eight factors are selected as follows:

1. The total numbers of wells
2. The startup number of wells
3. The number of new adding wells
4. The injected water volume last year
5. The oil moisture content of previous year
6. The oil production rate of previous year
7. The recovery percent of previous year
8. The oil output of previous year

Oilfield Output = Function of (X1, X2, …, X7, X8)

X1 = The total numbers of wells

X2 = The start up number of wells

X3 = The number of new adding wells

X4 = The injected water volume last year

X5 = The oil moisture content of previous year

X6 = The oil production rate of previous year

X7 = The recovery percent of previous year

X8 = The oil output of previous year

Y = The oil output

After fitting a Linear Model, we can test how accurate it is. For that we make predictions and compare the predictions to Actuals. The RMSE (root mean square error) is one indication of how good or bad our model is.

6.5.4 Questions Asked by Students

The following are questions that were asked by past students. These questions are a *very* good way of testing our understanding. I have answered each of them in the Appendix, but I urge you to first think about the answer yourself, before reading the answer provided. That is one very good way to learn.

7. What is the difference between machine learning and AI?
   Short Answer: AI is a broad field of science, which attempts to get "machines to think." ML is a very small subset of AI, in which we use lots of data (Big Data) to find patterns and use those patterns for predictions. ML is strongly based on statistics.
8. Why do we call it "linear model"?
9. How do you know if a relationship is linear or not?
10. How can we make other "fits" besides just linear ones?
11. Can you please explain how different homo-skedastic and hetro-skedastic plots are?
12. Can you please explain what R^2 means and what does it indicate?
13. For linear regression, do we always have to plot a scatter?
14. How to plot relationships with 3 variables, how does this effect R^2?

7 Introduction to Cloud Computing

7.1 Cloud Services

In order to evaluate the feasibility of the application of the cloud services, is important to analyze the current situation of the opportunity area considering the following steps:

Define the premises of the analysis.

- Iaas .-the components required to analyze certain process.

- Paas .- the result of the analysis of the components of certain process, in this case can be bases on the data of the own company or the external sources.

- Saas.- based on the analysis result, is possible to identify the feasibility of the application on the Cloud technology

7.2 Intro to Azure ML

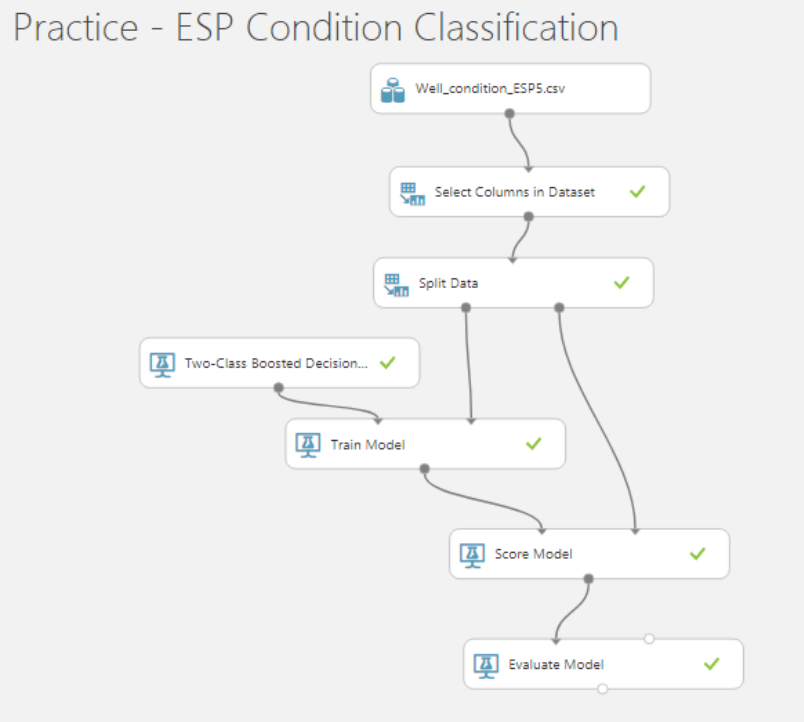
Log in to Azure ML Studio.

Understand how to build a Classification model.

Look at the Income Prediction example.

Upload the ESP_Wells Dataset. (5)

Follow the Steps given.



How to Upload a Data Set to Azure?

7.2.1 Questions Asked by Students

The following are questions that were asked by past students. These questions are a *very* good way of testing our understanding. I have answered each of them in the Appendix, but I urge you to first think about the answer yourself, before reading the answer provided. That is one very good way to learn.

1. What tech breakthrough made cloud feasible?
2. How reliable is cloud?
3. How do hardware manufacturers within the cloud industry take a stance in the market?
4. What O.S do cloud companies use?
5. How could people get paid on YouTube? Is anything free?
6. What is AI?

8 The Internet of Things

1. We started by looking at the O&G Digital Framework.
2. We brushed up on Internet of things
    a. We compared the ioT to the conventional human to human internet scheme. We discussed the technologies that enable the ioT such as ipV6, cheap bandwidth, cheaper processing, smartphones and wireless coverage.
    b. We touch upon on PAN, LAN, WAN and MAN,
    c. We talked a bit about the ioT disadvantages as a class

Digital Thread

- Engineering (Design)> Procurement (Source) > advance manufacturing(Create) >Services(Provide)

9 Some Concepts from Probability and Statistics

9.1 Basics of probability

Probability is a fraction

$$\frac{\text{Number of probable cases}}{\text{Number of all cases possible}}$$

np module for probability: np.random.binomial (n,p,s)

- n=number p=probability s=size/replications
- 0=fail 1+=success
- Replications is the number of times that we run the experiment
- Success of a project is highly dependent on the base probability assumption

There are 3 main important variables to define the probability of a Binomial event:

- Size: number of trials
- n : quantity of thing to analyze
- p = probability of each trial of success (this variable is the most important, this is based on the characteristics or historical behavior of the variable to analyze), in order to define the probability with less uncertainty is necessary to run several cases in order to evaluate all the possible cases.

Concepts: Mean (average), Median, Histogram and Sampling

Definitions

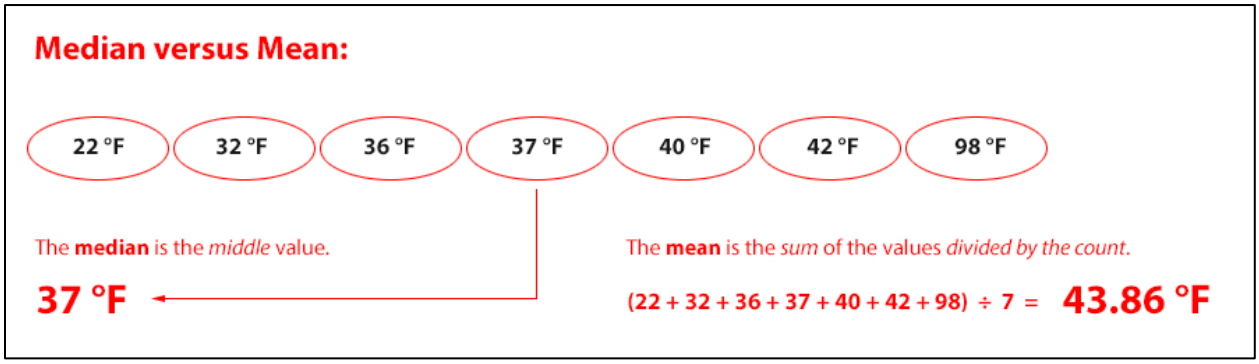
Mean

"Average" is technically called "the arithmetic mean": adding up the values and then dividing by the number of values.

Median

In statistics, the middle value of a set of numbers or data points; half the figures will fall below the median and half above. Quantity lying at the midpoint of observed values or quantities, such that there is an equal probability of falling above or below it.

Some Ideas for pictures: (taken from the Web)



10 Statistics Detour: Introduction to Sampling

- What is Sampling? Why is it needed?

- How to take Unbiased Samples? Avoiding Biases
- Resampling

Let's start with some quick terminology:

Population: the entire set of entities
Sample: A subset of the population
Sampling: the process of selecting a sample.

Sampling

Definition: Statistical method of obtaining representative data or observations from a group (lot, batch, population, or universe).

Sampling is the technique of selecting a representative part of a population, in order to determine characteristics of the whole population.

What is the purpose of sampling? To draw conclusions about populations, we must use techniques which enables us to determine a population`s characteristics by directly observing only a small portion (or sample) of the population

10.1.1 Why is Sampling Needed?

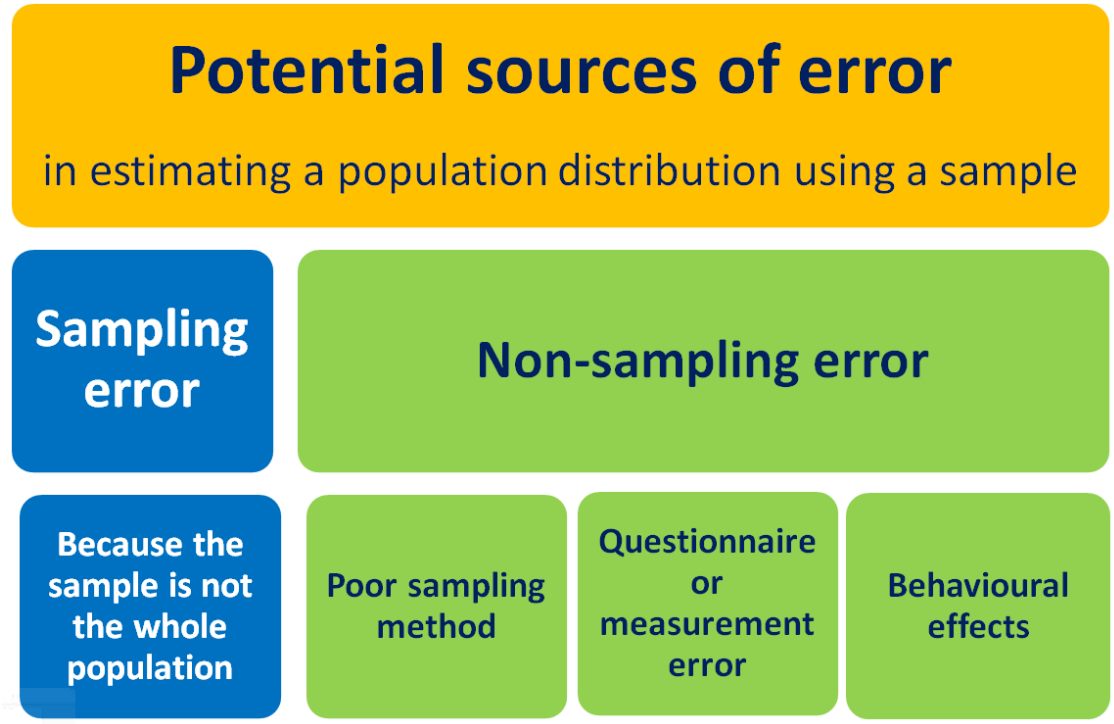
- Sometimes, it is very expensive and time-consuming to measure every single element. So we resort to measuring or auditing samples. (Saves time and money)
- If the underlying data is really large (Big Data), we can take samples and come to a very good understanding of the overall population.
- Sometimes measurement can be destructive.
- The results can be obtained quicker

10.1.2 Staying Unbiased

The three requirements for a sample to be unbiased

1. Each member of the population should be equally likely to be chosen
2. The sample is representative of the population as a whole
3. The sample is large enough that the results can be generalized

The sample should be a representation of the entire population. When taking a sample from a larger population, it is important to consider how the sample is chosen. To get a representative sample, the sample must be drawn randomly and encompass the whole population.



Source: https://learnandteachstatistics.wordpress.com/2014/09/04/sampling-and-non-sampling-error/

"Non-sampling error is the error that arises in a data collection process as a result of factors other than taking a sample. Non-sampling errors have the potential to cause bias in polls, surveys or samples.

By contrast, a *biased* sample is one where some members of the population are more likely to be chosen in the sample than others.

If a dataset has many many rows sampling can be excellent.

10.1.3 Resampling Techniques

Sometimes, we want the *distribution* of a population parameter. (What is the *distribution* of the heights of Italian women?) There are also cases when obtaining samples is very difficult (rare species) or very expensive. In those cases, resampling can be useful.

- Ideally, we would have access to the entire population or a lot of representative data from it.
- This is usually not the case, and the limited data available has to be re-used in clever ways in order to be able to estimate the error of our classifiers as reliably as possible, i.e., to be re-used in clever ways in order to obtain sufficiently large numbers of samples.

In resampling statistics, statistical estimates are formed by taking random samples directly from the data at hand. In other words, you randomly sample your random sample!

Bootstrapping Technique

Bootstrapping Example

- Suppose that we want to calculate the 95% confidence interval for population mean.
- We have got the sample of size 10.
- The following are steps we can perform for bootstrapping:
  1. We draw a resample from the sample
  2. For the resample, we compute the mean
  3. We repeat steps 1 and 2 for about 1000 (say) times
  4. Hence , now we have 1000 different means calculated
  5. We take 2.5^{th} Percentile as the lower confidence limit and 97.5^{th} percentile as upper confidence limit

- **Important**: Bootstrapping does not give "better" estimates. It gives a better range for the estimate
- For Bootstrapping to work, original sample must be representative and no crazy outliers

Jack-knifing Technique

Jackknifing

- Used to estimate the bias in sample statistics, the influence of particular observations, to estimate variances
- If you have some data X_1, X_2, ..., X_n and you are computing a statistic θ (e.g. the mean)
- Systematically leave out one observation at a time and recompute the statistic
  - mean(X_2, ..., X_n); mean(X_1,X_3, ..., X_n); mean(X_1,X_2,X_4, ..., X_n)

Comparing Bootstrap vs. Jackknifing

- Bootstrap: Shuffle a huge deck where the 10 El Nino years are replicated many, many times and average the first 10 (equivalent to random draw of 10 from 10 with replacement). Repeat huge (1000?) number of times.

- Jackknife: Delete one of 10 El Nino years from the sample and average the rest. Repeat for each of the 10 years. Produce 10 9-year means.

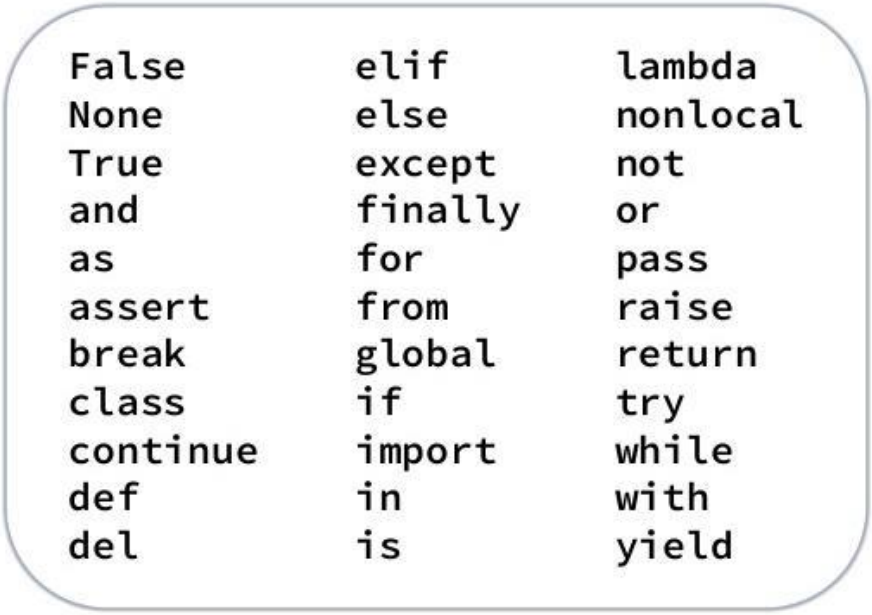
11 References

Python:

12 Appendix

Instructions for Downloading and Installing Anaconda (and Python)

12.1 All the keywords in Python

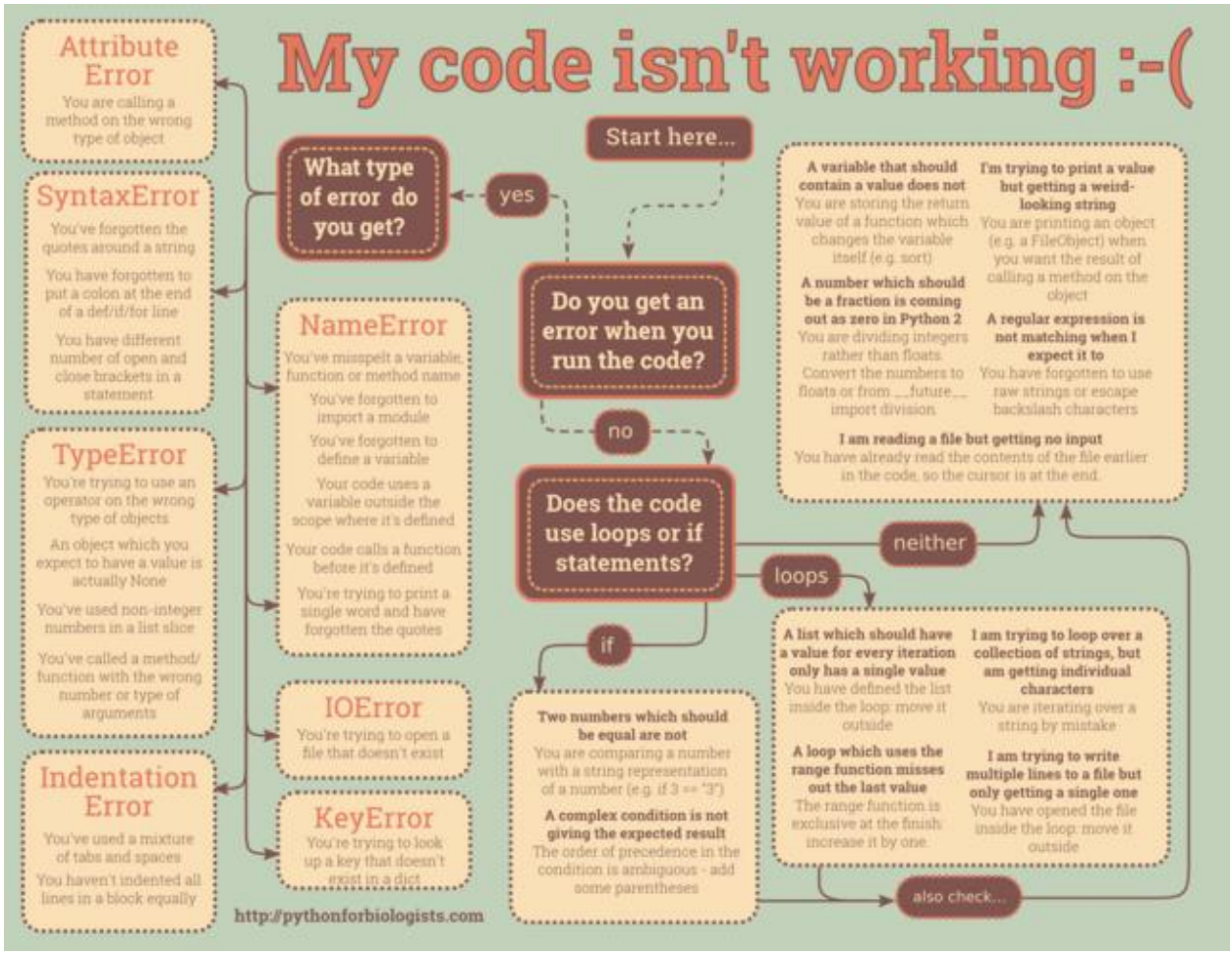


To see all the python built-in commands, try typing in your Jupyter notebook:

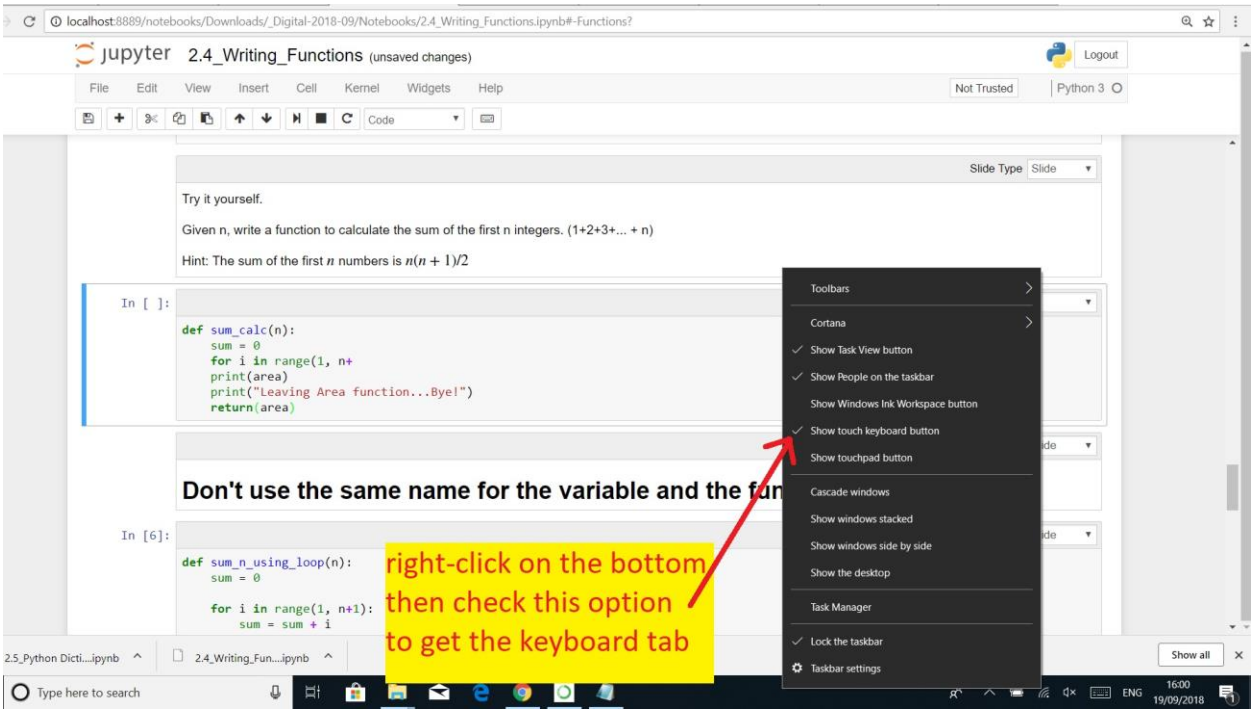
```
> dir(__builtins__)
```

12.2 Common Python Errors

Exception	Description
IOError	If the file cannot be opened
ImportError	If python cannot find the module
ValueError	Raised when a built-in operation or function receives an argument that has the right type but an inappropriate value
KeyError	Raised when a mapping (dictionary) key is not found in the set of existing keys
IndentationError	raised due to incorrect indentation
SyntaxError	Raised when the parser encounters a syntax error


My code isn't working :-(
Start here...
What type of error do you get?
yes
Do you get an error when you run the code?
no
Does the code use loops or if statements?
neither
loops
if
Attribute Error
You are calling a method on the wrong type of object
SyntaxError
You've forgotten the quotes around a string
You have forgotten to put a colon at the end of a def/if/for line
You have different number of open and close brackets in a statement
TypeError
You're trying to use an operator on the wrong type of objects
An object which you expect to have a value is actually None
You've used non-integer numbers in a list slice
You've called a method/ function with the wrong number or type of arguments
Indentation Error
You've used a mixture of tabs and spaces
You haven't indented all lines in a block equally
NameError
You've misspelt a variable, function or method name
You've forgotten to import a module
You've forgotten to define a variable
Your code uses a variable outside the scope where it's defined
Your code calls a function before it's defined
You're trying to print a single word and have forgotten the quotes
IOError
You're trying to open a file that doesn't exist
KeyError
You're trying to look up a key that doesn't exist in a dict
http://pythonforbiologists.com
A variable that should contain a value does not
You are storing the return value of a function which changes the variable itself (e.g. sort)
A number which should be a fraction is coming out as zero in Python 2
You are dividing integers rather than floats. Convert the numbers to floats or from __future__ import division
I am reading a file but getting no input
You have already read the contents of the file earlier in the code, so the cursor is at the end.
I'm trying to print a value but getting a weird-looking string
You are printing an object (e.g. a FileObject) when you want the result of calling a method on the object
A regular expression is not matching when I expect it to
You have forgotten to use raw strings or escape backslash characters
A list which should have a value for every iteration only has a single value
You have defined the list inside the loop: move it outside
A loop which uses the range function misses out the last value
The range function is exclusive at the finish: increase it by one
I am trying to loop over a collection of strings, but am getting individual characters
You are iterating over a string by mistake
I am trying to write multiple lines to a file but only getting a single one
You have opened the file inside the loop: move it outside
Two numbers which should be equal are not
You are comparing a number with a string representation of a number (e.g. if 3 == "3")
A complex condition is not giving the expected result
The order of precedence in the condition is ambiguous - add some parentheses
also check...

12.3 Dealing with the International Keyboard



(Thanks, Jamal!)

13 Acknowledgements

These class notes are possible, thanks to the work of many students who willingly contributed. In Alphabetical order, they include: Pablo Ahorsey, Tariq Al-Hajri, Ali Alshehhi, Laras D., Lachlan Hoqq, Ram Narasimhan, Manayer Salmeen, Lauro Vargas, Marissa Williams.

To Obtain the Latest version: This handbook is created and maintained by Ram Narasimhan. It gets updated whenever I use it to teach a class. For the latest version, simply send an email to ramnarasimhan@gmail.com