ASCEND
towards a sustainable future

KIRON
e-Learning for Future

Python For Data Science

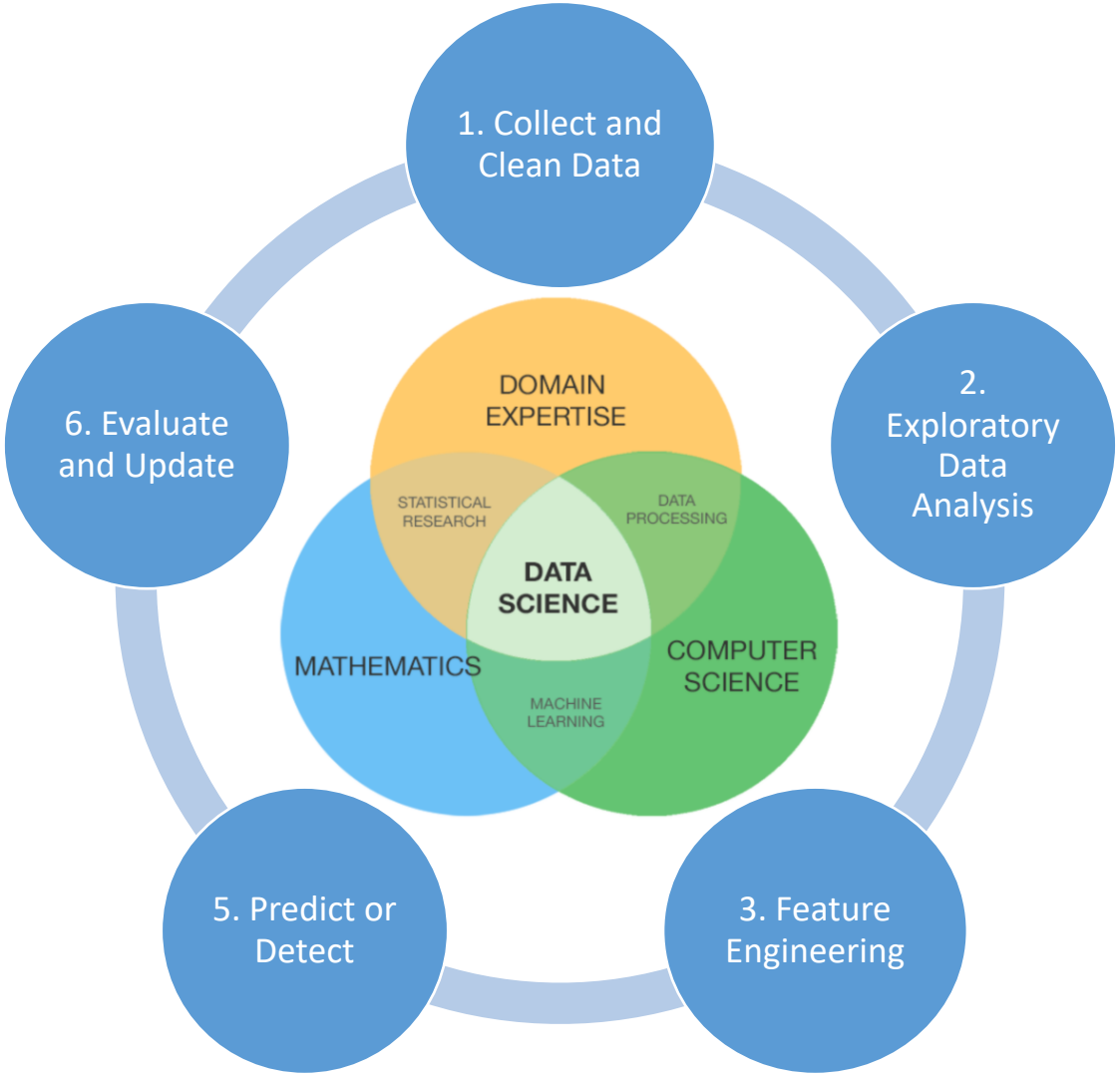
MIST – March 2021

Rabbani Mozahid

What is Data Science?

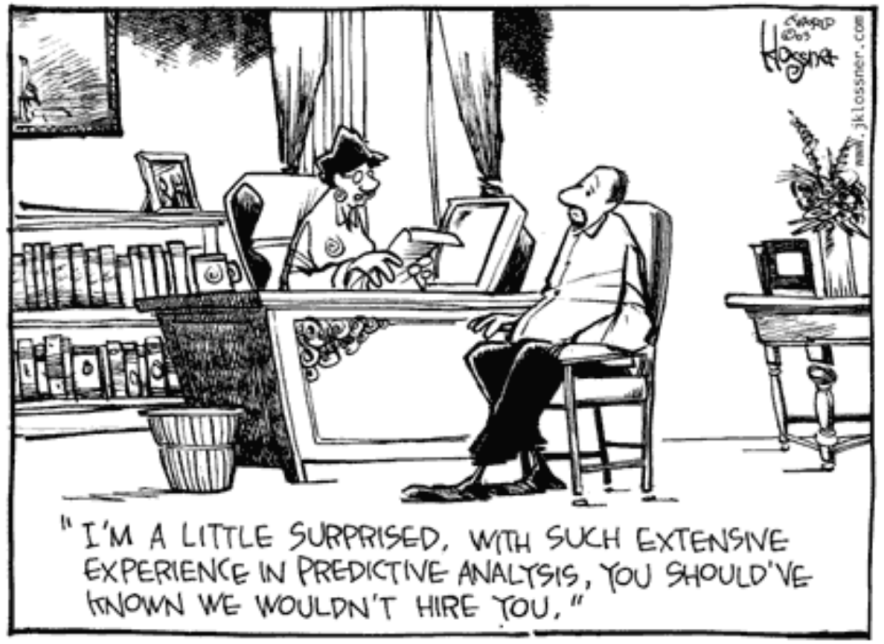
- Data Science is a field of study that combines business understanding with programming skills, math and statistics skills in order to extract meaningful insights from data

- Pattern Discovery



Why You Should Learn Data Science

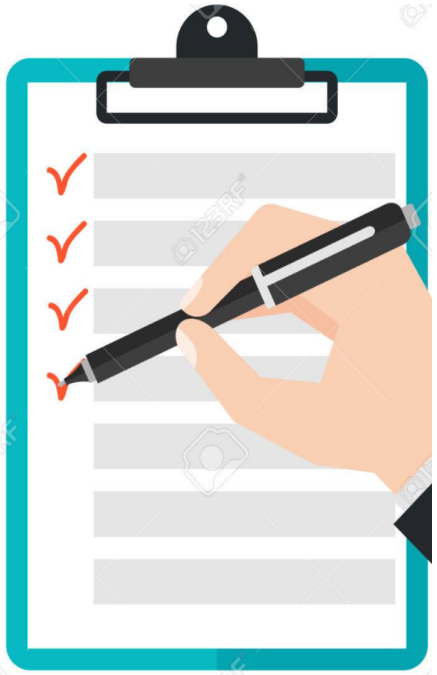
- ❑ High demand in Job Market and low supply of skilled resources
- ❑ A wide range of career options including the following:
  - Data Scientist
  - Data Engineer
  - Data Analyst
  - Machine Learning Engineer
  - Statistician
- ❑ Ability to directly impact business strategy
- ❑ It is an exciting, innovative and creative field



Review Python and Jupyter Notebook

Session-1

Agenda



Lecture -1: Review Python and Jupyter notebook

- ❑ Use Jupyter Notebooks to begin programming in Python
- ❑ Use Python data types including strings, integers, floats, Booleans, lists, tuples and dictionary
- ❑ Learn how to use Python to store and organize your data
- ❑ Using magic functions with Jupyter Notebook

Use Jupyter Notebooks to begin programming in Python

Using JupyterLab

- Open your Jupyter Lab
- If you do not have it installed yet, you can get it here

  https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html

- Let's get familiar with the JupyterLab
  - Open Session1-PythonDataScience.ipynb
  - Type and run some Python commands
  - Explore left pane and menu buttons

Markdown For Documenting Analytics

- A cell can be converted to a markdown cell from menu

Header with single "#"

Subheader with double "##" and so on

- Supports HTML and LaTex
- Bullets can be typed with "-" and *
- LaTeX inside "$ $", for example "\int_0^\infty" inside dollar signs will be $\int_0^\infty$

https://en.support.wordpress.com/markdown-quick-reference/

PYTHON REVIEW

Python Overview

- Currently the most popular programming language
- It is powerful and few lines of code can do a job where most of the other languages can take 3 to 5 times more codes
- It can be a glue language that can be used in-between other language or components
- It can be used almost in any domain and can run everywhere
- Open source and its easy to learn

Quiz

What is the result of running this code?

```
x = 10
y = x
x = 6
print(x,", ",y)
```

a) 10, 6
b) 6, 10
c) 6, 6

Answer 6, 10

Quiz

What is the output of the expression?

```
3*1**3
```

a) 27
b) 9
c) 3
d) 1

Precedence of ** is higher than *, thus first 1**3 will be executed and the result will be multiplied by 3.

Python Syntax

- Python uses indentation and/or whitespace to delimit statement blocks
- Whitespace within lines does not matter
- End-of-Line terminates a statement

```
if 10 > 7:
  print      ("Ten is greater than seven.")
```

Ten is greater than seven.

Python is dynamically typed

```
a = 7 # Instead of static int a = 7
```

Comments are marked by #

```
# This is a comment
print ("Hello Python")
```

Hello Python

Python Types

- **Python types are dynamic typing, so you do not need to declare**
- **Python has a variety of built-in data types by default**

  - Int – Any integers or whole numbers like 1, 2, 89 ;
  - Float – 2.171892 ;
  - Bool – True of False
  - Str, unicode – ‘MyString’, u‘MyString’
  - List – [69, 6.9, ‘mystring’, True]
  - Tuple – (69, 6.9, ‘mystring’, True) immutable
  - Set – set([69, 6.9, ‘str’, True]) –no duplicates & unordered
  - Dictionary – {‘key 1’: 6.9, ‘key2’: False} - group of key-value pairs

Character Types

```
s = 'Hello World'
s[0]
```

Output

H

Python	H	e	l	l	o		W	o	r	l	d
Index value	0	1	2	3	4	5	6	7	8	9	10
Index value R to L	−11	−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

Numeric types

```
x = 1
y = 1.5
z = x * y
```

What is the Python data types for x, y and z?

name	type
x	class 'int'
y	class 'float'
z	class 'float'

Quiz

How do you slice “World’ from the ‘Hello World’?

```
# Hint:
s = 'Hello World'
```

a. s(5:)
b. s(6:)
c. s[6:]
d. s[5:]

Answer
s = 'Hello World'
s[6:]

'World'

Boolean

- Python boolean types are True and False (case sensitive).
- Boolean test of number 0 (print(bool(0)) is always False.
- Test for other numbers are True.

```
# Boolean Type in Python
x = 32.0
y = 32
x == y
```

True

Boolean (contd.)

What is True + False? Explain your answer.

```
[ ]: True + False
```

1

Format

Format specification for floats:

- Display one digit after the decimal using {0:.1f}, two places after the decimal {0:.2f} and so on.
- E. g. "precision: {0:.1f} and {0:.2f}".format(3.14159265) will result in 'precision: 3.1 or 3.14'

```
# Python Format example
"6.33 as a pct. of 150: {0:.2%}".format(6.33/150)
```

OUTPUT:

'6.33 as a pct. of 150: 4.22%'

Datetime - Not a built-in data type

- **Datetime is not a built in data type in Python. It is a module that you can import for manipulating dates and times**

```
from datetime import datetime
now = datetime.now()
print('now: ' , now, '\n' ,
'Year:' , now.strftime("%Y"), '\n' ,
'Month:' , now.strftime("%B")
```

OUTPUT:

now: 2020-02-28 01:41:25.484199

Year: 2020

Month: February

```
date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
```

12/24/2018, 04:59:31

A complete list of strftime() directives is available here at https://strftime.org/

Using List, Tuple and Dictionaries

List

- Flexible and multipurpose data type in Python
- Defined using square bracket

```
list1 = [11,22,33]
list1
```

[11, 22, 33]

```
list1[1]
```

22

```
list1 [0]
```

11

```
#Update
list1[2]=100
list1
```

```
[11, 22, 100]
```

```
list1.append(33)
list1
```

```
[11, 22, 100, 33]
```

List (Contd.)

```
#Looping through a list
for i in list1:
    print(i)
```

```
11
22
100
33
```

```
#deleting an element
list1.remove(100)
list1
```

```
[11, 22, 33]
```

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
for x,y in zip(list1, list2):
    print(x,", ",y)
```

```
1 ,  4
2 ,  5
3 ,  6
```

What is the value of lis1[3]?

Quiz

What will be printed value for y?

```
x = [10,20,30]
y = x
x[1] = 42
print(y)
```

a. [10, 42, 30]
b. [10,20,30]

Answer:
a. [10, 42, 30]

Tuple

- Tuples are useful for parallel computing when you do not want something to be changed by a worker process

```
#Looks same as list
tuple1 = ('FRM',350, 'SF','FL')
tuple1
```

```
('FRM', 350, 'SF', 'FL')
```

- But they are immutable and if the following statement is run, you will get an error message stating - "TypeError: 'tuple' object does not support item assignment"

  tuple1[3]= 'VA'

Dictionaries

- Dictionaries are one of the most widely used data types that makes Python powerful and effective for accessing and manipulating data.

```
#Defining a python dictionary

iphonePrice = {
    'iphone5' : 200,
    'iphone6' : 300,
    'iphone8' : 700,
    'iphonex' : 900   }
```

```
#accessing an element
iphonePrice["iphonex"]
```

```
900
```

```
#adding an element
iphonePrice['iphone7']=500
iphonePrice
```

```
{'iphone5': 200,
 'iphone6': 300,
 'iphone8': 700,
 'iphonex': 900,
 'iphone7': 500}
```

List and Dictionary Comprehension

Comrehension is a way to build list or dictionaries quickly

```
list1 = [i**2 for i in range(1, 11)]
list1
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
import random
list1= [random.randint(0,5) for i in range (1, 10)]
list1
```

```
[3, 0, 4, 4, 4, 2, 2, 1, 2]
```

```
dict= {i : i**3 for i in range (1, 10)}
dict
```

{1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343, 8: 512, 9: 729 }

Quiz

Is X is equal to 1 as shown below?

```
x = [.1] * 10
x == 1
```

a) True
b) False

Answer - False

Explanation: [.1] is a Python list. Value of x is the following.

[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]

Lab – 1

Session1a-PythonDataScience.ipynb

Session1b-PythonDataScience.ipynb

Use Python to Store and Organize Data

Accessing, Manipulating and Storing Data

- Python has reach Data Analysis libraries like the following:
  - Numpy multi-dimentional arrays
  - SciPy libraries
  - Pandas DataFrames

- However, SQL remains as the most dominant way of storing and using data in most of the organization

- So, we need to learn how to use Python with SQL

SQLite

- "SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world."
- We can use this database to store relational data when we develop applications on our local system and then can easily deploy to any larger production databases like Oracle, PostgreSQL etc.
- "There are over 1 trillion (1e12) SQLite databases in active use."
- SQLite is fully open source and is free to everyone to use for any purpose.
- Visit this page to learn more - https://www.sqlite.org/index.html

SQLite (contd.)

[10]:
```
import pandas as pd
import sqlite3
df=pd.read_csv('/Users/adamrob/data/iris/Iris.csv')
df.head(3)
```

[10]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa

[11]:
```
conn = sqlite3.connect('iris.sqlite3')
df.to_sql('flowers', conn, if_exists='replace', index=False)
```

[8]:
```
df_sum = pd.read_sql("select Species, count(*) as total from flowers group by Species", conn)
df_sum
```

[8]:

	Species	total
0	Iris-setosa	50
1	Iris-versicolor	50
2	Iris-virginica	50

Using magic functions with Jupyter Notebook

Magic Commands

- Magic commands are very handy when you are working with Jupyter Notebook or JupyterLab. Although they look like UNIX commands, they are actually implemented in Python.

- There are two categories of magics:
  - Line magics and
  - Cell magics.

- Line magics act on a single line and cell magics apply on entire cells.
  - Line magics start with a percent character %, and cell magics start with double %%.
  - Note that ! is just like a magic function for some Unix shell commands.

- To see the available magic functions in your notbook, you can run %lsmagic.

Magic Commands (contd.)

Writing to a file

[1]:
```
%%writefile quiz.py
from random import randint

#how big a number should we guess?
max_number = 5
first_line = "Guess a number between 1 and %d" % max_number
print(first_line)

number = randint(1, max_number)

not_solved = True

#keep looping unil we guess correctly
while not_solved:
    answer = int(input('?'))
    you_said = "You typed %d" % answer
    print (you_said)
    if answer > number:
        print ("The number is lower")
    elif answer < number:
       print ("The number is higher")
    else:
        print ("You got it right")
        not_solved = False
```

```
Overwriting quiz.py
```

Runing external code

[2]:
```
# Runing external code
%run quiz.py
```

Lab – 2

Session1c-Store-Data-Sqlite.ipynb

Session1d-UsingMagicCommands.ipynb

Thank You