Fundamental of Data Analytics

"If you torture the data long enough, it will confess."
Ronald Coase, Economics"

Hans Rosling's 200 Countries, 200 Years

Data

Characteristics / Information

Data Analytics

Data Analytics is the systematic computational analysis of data or statistics.

It is used for the discovery, interpretation, and communication of meaningful patterns in data. It also entails applying data patterns towards effective decision making.

Example 1

X	Y
1	2
4	8
3	6

Discover something from the above data…….

Example 2: Annual Result of class

Passed,
Failed,
Failed,
Passed,
Passed,
Passed,
Passed,
Failed,
Passed,
Passed,

Discover pattern from the above data……

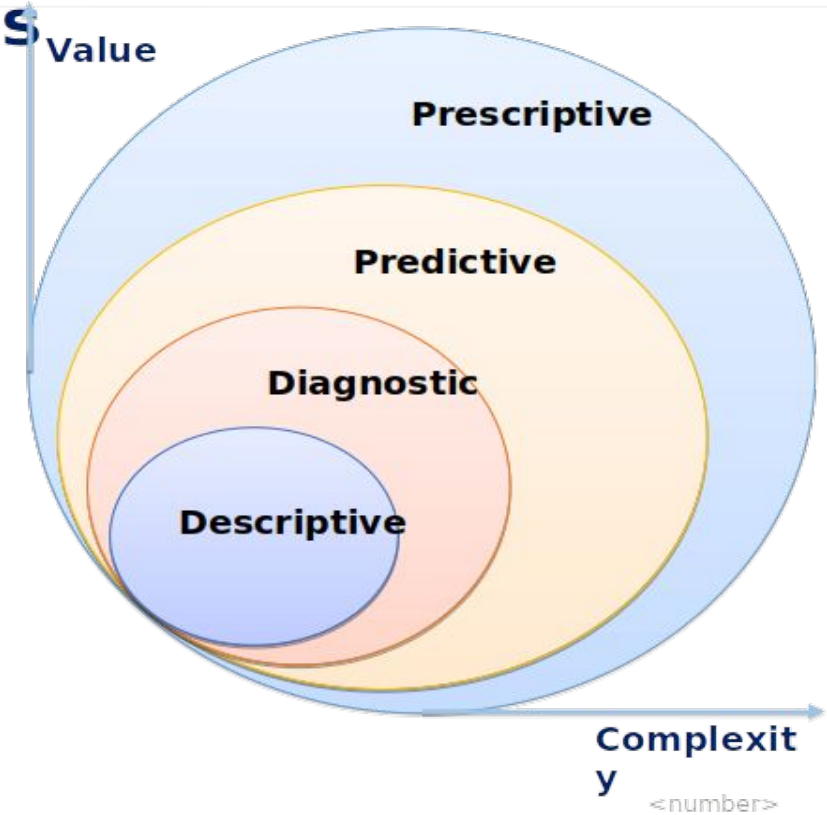
Example 3

positivity	integrity	Hardwork	self-regulation.	Result
30%	20%	45%	34%	**Failed**
50%	50%	50%	50%	**Succeed**
70%	70%	70%	70%	?
15%	10%	70%	60%	?

Can you predict/forecast the Result for last two rows?

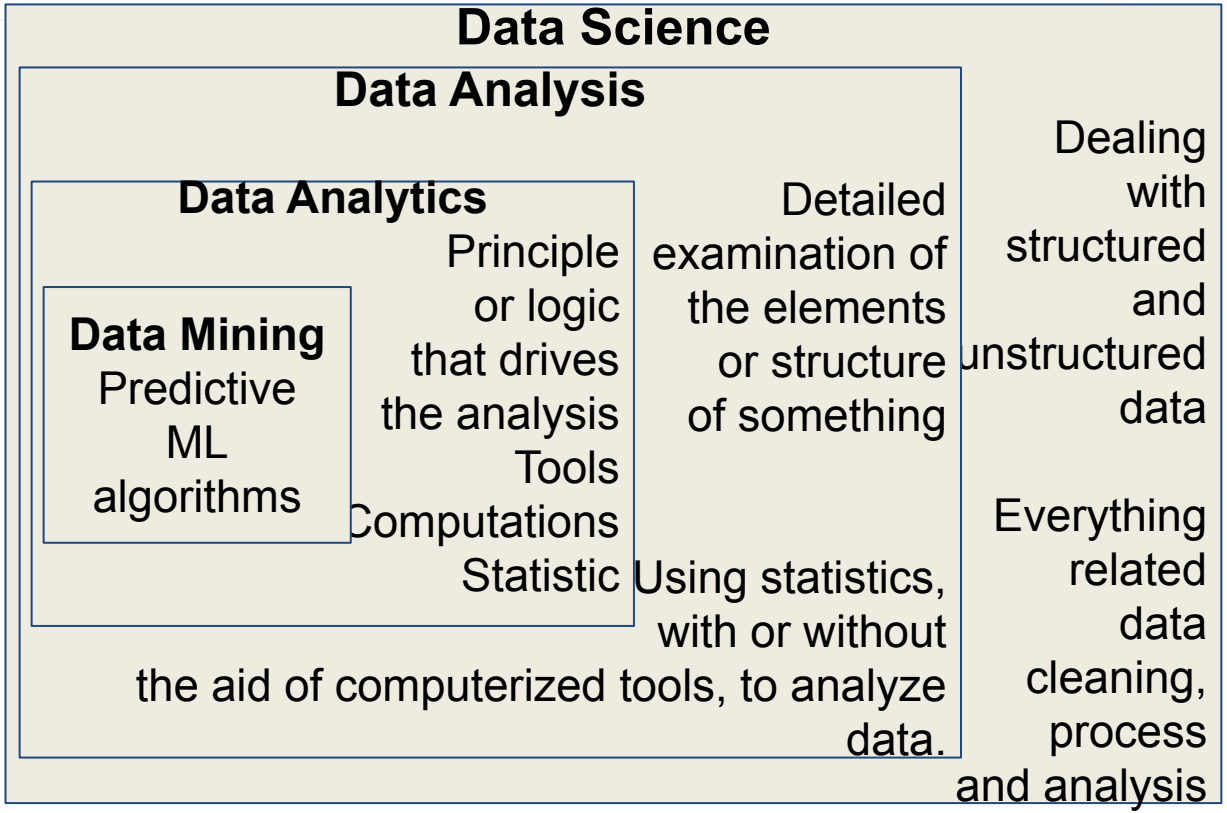
Types of Data Analytics

1. **Descriptive Analytics:** *What is happening?*
   - Comprehensive, accurate and live data
   - Effective visualization
2. **Diagnostic Analytics:** *Why is it happening?*
   - Drill down the root cause
   - Isolate the confounding information
3. **Predictive Analytics:** *What is likely to happen?*
   - Likelihood of an event happening in future
   - Estimating a point in time at which something might happen
4. **Prescriptive Analytics:** *What do I need to do?*
   - Recommended actions and strategies based upon the challenges
   - Applying advanced analytical technique
   - Google maps suggesting best route home



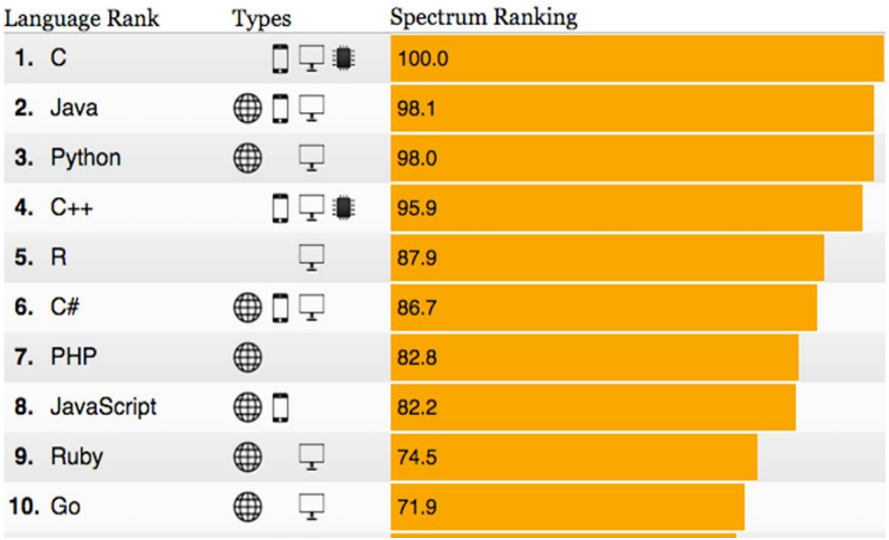
Definition

Data Science
Data Analysis
Data Analytics
Data Mining



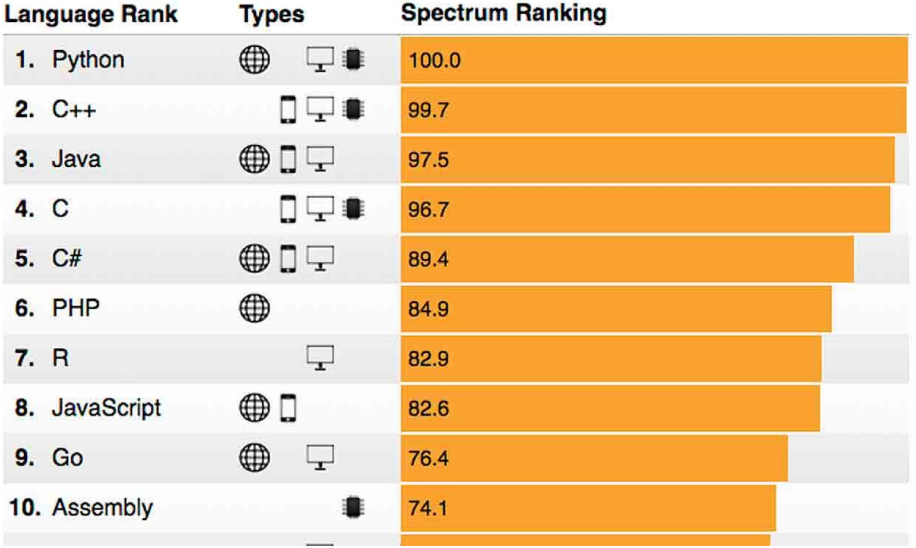
Introduction to Python

IEEE Spectrum 2016 Ranking



Language Rank	Types	Spectrum Ranking
1. C		100.0
2. Java		98.1
3. Python		98.0
4. C++		95.9
5. R		87.9
6. C#		86.7
7. PHP		82.8
8. JavaScript		82.2
9. Ruby		74.5
10. Go		71.9

IEEE Spectrum 2018 Ranking



Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		99.7
3. Java		97.5
4. C		96.7
5. C#		89.4
6. PHP		84.9
7. R		82.9
8. JavaScript		82.6
9. Go		76.4
10. Assembly		74.1

Vodafone India Foundation

_VOIS

Connecting Dreams

IEEE Spectrum 2020 Ranking

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.3
3	C		94.6
4	C++		87.0
5	JavaScript		79.5
6	R		78.6
7	Arduino		73.2
8	Go		73.1
9	Swift		70.5
10	Matlab		68.4

Choosing Python for Data Analytics

Available of Significant openSource Python Libraries

Numpy : Support for large multi-dimensional arrays and matrices

Pandas : Fast, powerful, flexible, and ease to use data analysis and manipulation

Matplotlib : Static, animated and interactive visualization

Scikit-learn : Machine learning and data preprocessing library

Tensorflow : Deep Learning

....

....

Python

Interpreted, High-Level, General-purpose Programming Language

What can Python do?

- System utilities (system admin tools, command line programs)
- Web Development
- Graphical User Interfaces (Tkinter, gtk, Qt).
- Internet scripting
- Embedded scripting
- Database access and programming
- Game programming
- Rapid prototyping and development
- Distributed programming

Organizations using python

- It is a very popular language in multiple domains like Automation, Big Data, AI etc.
  You will also be impressed as it used by the vast multitude of companies around the globe.

Python Syntax compared to other programming languages

C++ Program :

```cpp
#include <iostream>
int main()
  {
    std::cout << "Hello World" << std::endl;
    return 0;
  }
```

C Program :

```c
#include <stdio.h>
int main(int argc, char ** argv)
  {
    printf("Hello, World!\n");
  }
```

Java Program :

```java
public class Hello
 {
   public static void main(String argv[])
     {
       System.out.println("Hello, World!");
     }
 }
```

Python Program:

```python
print ( "Hello World")
```

Anaconda

- **Anaconda** is a free and open-source[6] distribution of the Python for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

- **Anaconda distribution** comes with over 250 packages automatically installed, and over 7,500 additional open-source packages.

- The following applications are available by default in Navigator
  - JupyterLab
  - Jupyter Notebook
  - Spyder
  - RStudio
  - Visual Studio Code

Jupyter notebook

- The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

conda package manager

To install

conda install -c conda-forge notebook

Pip package manager

To install

pip install notebook

To Run

jupyter notebook

Vodafone India Foundation

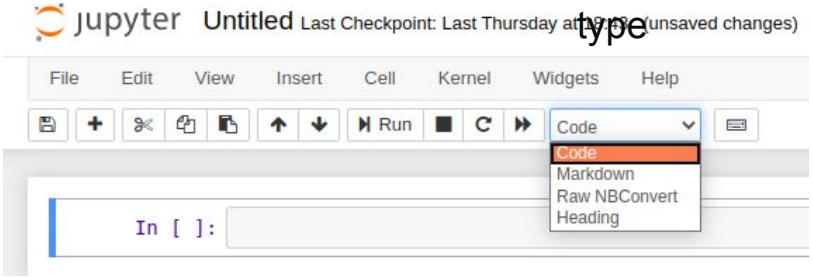
_VOIS

Connecting Dreams

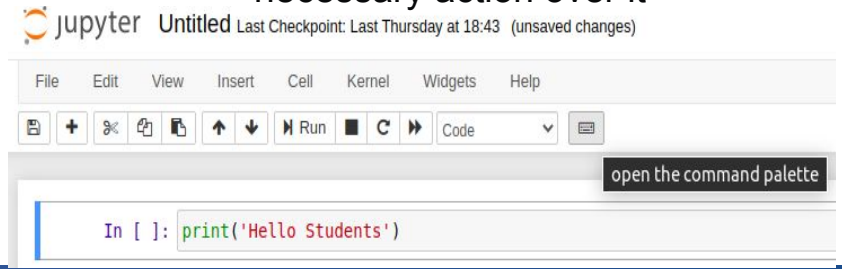
Working on Jupyter notebook

Create a new notebook on python

Select 'Code' as cell type



Write code in the cell, use command to take necessary action over it



Few commands

Python Variable Name Rules

- Must begin with a letter (a - z, A - B) or underscore (_)
- Other characters can be letters, numbers or _
- Case Sensitive
- Can be any (reasonable) length
- There are some reserved words which you cannot use as a variable name because Python uses them for other things.

Python Reserved words:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	el	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Function

The keyword def introduces a function definition.

It must be followed by the function name and the parenthesized list of formal parameters.

The statements that form the body of the function start at the next line, and must be indented.

The return statement returns with a value from a function.

return without an expression argument returns None.

```
def function_name(parameters):
        """docstring"""
        statement(s)
```

```
def fun(a,b):
    s = a+b
    return s
result=fun(5,3)
print(result)
```

```
8
```

Lambda expression

Small anonymous functions can be created with the lambda keyword.

```
x = lambda a,b: a*b
```

```
x(3,4)
```

```
12
```

This function returns the product of its two arguments(a,b).

Lambda functions can be used wherever function objects are required. They are syntactically restricted to a single expression.

Python Data Type

- Everything in Python is an object and every object has an identity, a type, and a value
- Type represents the kind of value and determines how the value can be used
- **Python has three distinct numeric types:** integers, floating point numbers and complex numbers
- Booleans are a subtype of plain integers. **Boolean** values behave like the values 0 and 1

```
x = 212
y = 12.6
z = complex(1,2)
name = ("Hello World")
var_bol= True
```

```
print (type(x))
print (type(y))
print (type(z))
print (type(name))
print(type(var_bol))
```

Strings

In Python, a string type object is a sequence (left-to- right order) of characters. Strings start and end with single, double or tripple quotes.

```
mystr1 = 'Python'
mystr2 = "Python's Usage"
mystr3 = '''He said,"He is alright"'''
print (mystr1)
print (mystr2)
print (mystr3)
```

String indices and accessing string elements

Strings are arrays of characters and elements of an array can be accessed using indexing. Indices start with 0 from left side and -1 when starting from right side.

character	P	Y	T	H	O	N		R	O	C	K	S
Index from left	0	1	2	3	4	5	6	7	8	9	10	11
Index from right	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
myStr = "PYTHON ROCKS"
print (myStr)
print(len(myStr))
print(myStr[0])
print(myStr[-11])
```

Strings are immutable

Strings are immutable character sets. Once a string is generated, you can not change any character within the string.

```
myStr1 = "PYTHON ROCKS"
print(myStr1[0])

myStr1[0]="M"      # Will not work
```

Some methods associated with string

capitalize()
find()
isupper()
replace()
split()
swapcase()
upper()

String Slicing

To cut a substring from a string is called string slicing.

Here two indices are used separated by a colon (:). A slice 3:7 means indices characters of 3rd, 4th, 5th and 6th positions.

```
my_str = "PYTHON ROCKS"
print(len(my_str))
print(my_str[0:7])
print(my_str[-12:-1])
print(my_str[-12:])
print(my_str[-12:-6])
```

Lists

- Collection of values
- Contain any data type
- Contain different data types and lists also within the list

As opposed to int, bool etc, a list is a compound data type: you can group values together:

```
myList = ["HI", 3, 5.3, True]
```

Lists are mutable

- Unlike String, lists are mutable

```
myList = ["HI", 3, 5.3, True]
myList[2]='Hello'
print(myList)
```

Some methods associated with list

append()
extend()
insert()
remove()
reverse()

Membership operator ('in' and ‘not in’)

The 'in' operator is used to check whether a character or a substring is present in a string or not.
The 'not in' operator is used to check whether a character or a substring is not present in a string or present (reverse of ‘in’).
The expression returns a Boolean value.

```
myStr2 = "PYTHON ROCKS"
print(myStr2[1])
print("Y" in myStr2)
print("A" not in myStr2)
print("ROCKS" in myStr2)
```

range() function

If you do need to iterate over a sequence of numbers, the built-in function range() comes in handy. It generates arithmetic progressions

```
range(stop) -> range object
range(start, stop) -> range object
range(start, stop, step) -> range object
Return an object that produces a sequence of
integers.
If you want to print the sequence, it is needed
to convert a suitable printable sequence like
list
```

```
In [1]: range(5)
Out[1]: range(0, 5)

In [2]: list(range(5))
Out[2]: [0, 1, 2, 3, 4]

In [3]: list(range(2,5))
Out[3]: [2, 3, 4]

In [4]: list(range(1,5,2))
Out[4]: [1, 3]
```

filter() function

```
print(filter.__doc__)
```

```
filter(function or None, iterable) --> filter object

Return an iterator yielding those items of iterable for which function(item)
is true. If function is None, return the items that are true.
```

```
def f(n):
    if n%5==0:
        return True
```

```
y=filter(f,[3,5,9,15,7,0])
print(list(y))
```

```
[5, 15, 0]
```

```
y=filter(None,[3,5,9,15,7,0])
print(list(y))
```

```
[3, 5, 9, 15, 7]
```

map() function

The map() function executes a specified function for each item in a iterable.
The item is sent to the function as a parameter.

```
def f(n):
    return len(n)
```

```
x = map(f, ('Vodafone', 'India'))
print(list(x))
```

```
[8, 5]
```

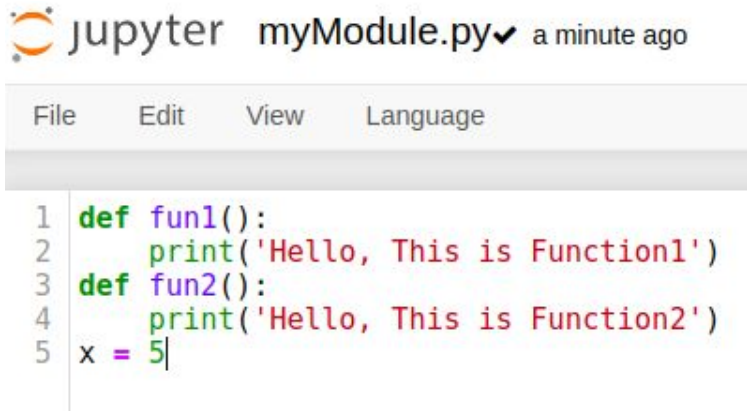
Modules:

A module is a file containing Python definitions and statements.

The file name is the module name with the suffix .py appended.

Within a module, the module's name (as a string) is available as the value of the global variable __name__.

For instance, use your favorite text editor to create a file called **myModule.py** in the current directory with the following contents:



Modules:

Import the '**myModule.py'** as alias '**mm**'.

(donot use .py, only name is required to import module)

```
import myModule as mm
```

```
mm.fun1()
```

```
Hello, This is Function1
```

```
mm.fun2()
```

```
Hello, This is Function2
```

Only import the fun1 from '**myModule.py'**.

(Selective import)

```
from myModule import fun1
```

```
fun1()
```

```
Hello, This is Function1
```

```
fun2()
```

```
---------------------------------------------
NameError                                 T
<ipython-input-4-a7e4d4b4ceec> in <module>
----> 1 fun2()

NameError: name 'fun2' is not defined
```

Local and global variables in Python

Variables that are only referenced inside a function are implicitly global.
If a variable is assigned a value anywhere within the function's body, it's assumed to be a local unless explicitly declared as global.

```
x = 5
def func1():
        print(x)
def func2():
        x = 7
        print(x)
func1()                 5
func2()                 7
print(x)                5
```

```
def func3():
        global x
      x = 8
        print(x)
func3()                 8
print(x)                8
```

Thank You