Python Basics

About the Module

- Why Python
- Variables and Operators
- Lists and Dictionaries
- Loops
- Functions
- Files and Libraries

Why Python

- Open Source
- Simple implementation
- Very large and active developer community
- Vast number of packages and libraries

Variables and Operators

- Variables : stores information which can vary, example : current_year = 2019
- Operators : Operations applied to variables
- Types:
  - Mathematical : Addition (+), Subtraction (-), Multiplication (*), Division (/)
  - Logical : and, or, >, <, >=, <=, ==

```
a = 10
b = 20
print (a + b)
>> 30
```

```
a = "data"
b = "science"
print (a + b)
>> "datascience"
```

```
a = 10
b = 20
print (a >= b)
>> False
```

Lists

- One dimensional array which can store multiple elements

I.
```
A = [1, 2, 3, 4, 5]

A.append(6)
print (A)
>> [1, 2, 3, 4, 5, 6]
```

II.
```
A = [1, 2, 3, 4, 5]
B = [6,7,8]
A.extend(B)
print (A)
>> [1, 2, 3, 4, 5, 6, 7, 8]
```

III.
```
A = [“this”, ”is”, "a", “list”]
B = [“of”, “strings”]
C = A + B
print ( C )
>> [“this”, “is”, "a", “list”, “of”, “strings”]
```

IV.
```
D = “ ”.join(C)
print (D)
>> “this is a list of string”

D = “-”.join(C)
D = “,”.join(C)
```

Dictionaries

- Datatype object which contains Keys and Values

I.
```
A = {   “name” : “Adam”,
        “age” : 30,
        “designation” : “doctor”,
        “city” : “New York”}
```

II.
```
# get the value of city
print (A[“city”])
>> “New York”
```

III.
```
# set a new key salary = 5000
A[“Salary”] = “50000”
```

IV.
```
A.keys()
>> "name", "age", "designation".
"city", "salary"
```

V.
```
A.values()
>> [‘Adam,’ 30, ‘doctor’, ‘New York’,
50000]
```

Conditional Statements

- Used to check conditions among the variables or constants

I.
```
A = 20
B = 15
if A > B:
    print ("A is Greater")
>> A is Greater
```

II.
```
A = {“name” : “Adam”, “Age” : 30}

if “salary” in A:
    print ("present")
else:
    print ("not present“)
>> not present
```

III.
```
If A[“Age”] > 65:
    print ("Senior Citizen”)

elif A[“Age”] >= 18 and A["Age"] < 65:
    print (“Adult”)

else:
    print (“Children”)
>>Adult
```

Loops

- Used to iterate in lists or dictionaries

```
A = {“name” : “Adam”,
      “age” : 30,
      “designation” : “doctor”,
      “city” : “New York”}

for key, value in A.items():
     print (key, value)

>> name  Adam
   age  30
   designation doctor
   city New York
```

```
A = [5, 10, 15, 20, 25]

for i, value in enumerate(A):
  print (i, value)

>> 0,5
   1, 10
   2, 15
   3, 20
   4, 25
```

Functions

- Blocks of organized, reusable code that is used to perform a single, related action.
- Provide better modularity for codes and enable code reusing

```
def add(a, b):
    c = a + b
    return c

add(3,5)
>> 8

add("data", "science")
>> "datascience"

add("data", 5)
>> error
```

Files

- Python can read almost any types of file systems : csv, tsv, txt, xlsx etc
- Used to read and load datasets into memory

```
data = open("dataset.csv").read()
lines = data.split("\n")
```

```
import csv
with open("dataset.csv") as csv_file:
        csv_reader = csv.reader(csv_file)
for row in csv_reader:
            print(row)
```

Packages

- Bunch of code bundled together that is written and designed to be reused in different scenarios
- Provides some generic functionality that can be used by specific applications
- In built packages : os, csv, re
- Custom packages : nltk, textblob, scikit learn

```
import math

math.sqrt(49)
>> 7.0
```