Python

Python Environment

jupyter

spyder

ANACONDA®

Types Of Variables

- Integer
- Float / double
- String
- Logical / Boolean

Operators

- **Comparison opr.**

  < > <= >= != <> ==

- **Logical Operator**

  and or not

- **Arithmetic opr.**

  + - / ( ) %

While Loop

No { } brackets

Indentation is important

```
while condition:
    executable code1
    executable code2
    executable code3
executable code4
```

```
while condition:
    executable code1
    executable code2
executable code3
executable code4
```

For Loop

```
for i in range(5):
    print('Hello ')
```

```
for j in range(1,10):
    print('Hello :', j)
```

range(begin,end,step)

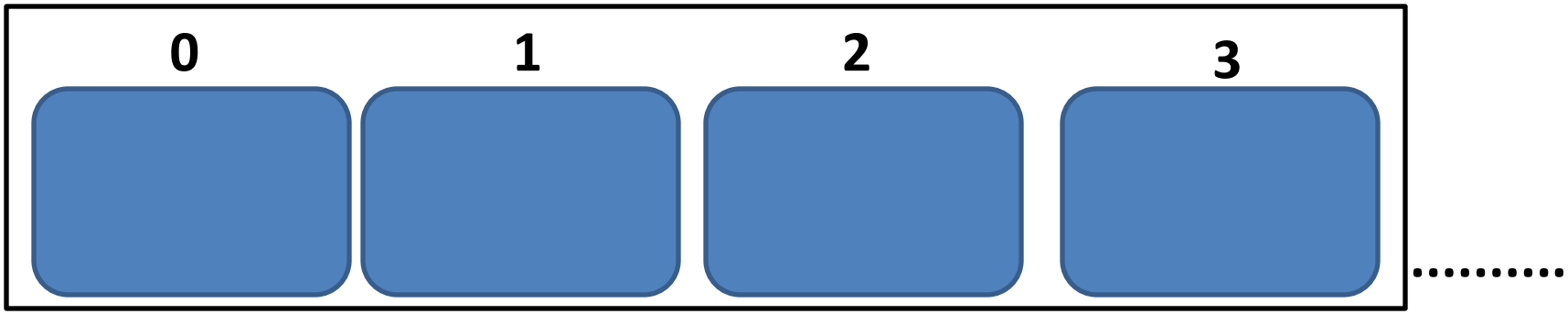
```
for k in range(10,100,5):
    print( k )
```

If stmt

```
if condition1:
        executable code
elif condition2:
        executable code
else:
        executable code
```

List

- Like Arrays
- Ordered Sequence of values
- Enumerated starting with zero
- Can be of mixed datatype



List

- list1 = [1,2,3,4,5,6]
- list2 = [‘a’, 55.5, ‘b’,2000]
- list3 = [‘123’,’how are you?’, list2]

list1.append(55)

list1[2] =55

- range(15)
- myList = list(range(10))

list1.sort()

list1.reverse()

Slicing

- **Subset the list**

Slicing

[____ : ____]

Start Stop

Advance Slicing

[____ : ____ : ____]

Start Stop Step

Slicing

letters

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	J
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

letters[:]

letters[: 7]

letters[2 :]

letters[2 : 7]

letters[2: 9 : 2]

letters[-8 : 7]

letters[: : 3]

letters[: : -1]

Tuples

- **Immutable list of values**
  - myTuple = (123, 456, 343)
  - myTuple[:]
  - type(myTuple)
  - len(myTuple)
  - myTuple[1] = 777 --error

Assignment

FINANCIAL STATEMENT ANALYSIS

Packages & Modules

- **Modules** in Python are simply **Python files** with a .py extension.
- The name of the module will be the name of the file.
- A Python **module** can have a set of **functions**, **classes** or **variables** defined and implemented.

e.g. Module color (color.py)

Function red()

Function blue()

Function green()

```
import color
        color.red
        color.green
OR
from color import red

from color import *
```

Packages & Modules

- **Packages** are **namespaces** which contain **multiple packages** and **modules** themselves. They are simply **directories**.

- We create a directory **drawing**

  Include modules in it:

  color, line, rectangle, square, circle

- To use line module from drawing package

```
import drawing.line
from drawing import circle
```

```
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt2
```

Packages & Modules

Install a New Package

conda install packg_name OR **pip install** packg_name

Anaconda Prompt

```
'chcp' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users>conda install scrapy
Solving environment: done


## Package Plan ##

  environment location: C:\Users\Bibhu\Anaconda3

  added / updated specs:
    - scrapy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    hyperlink-18.0.0           |            py36_0          62 KB
    automat-0.6.0              |   py36hc6d8c19_0           67 KB
    parsel-1.4.0               |            py36_0          27 KB
    pydispatcher-2.0.5         |            py36_0          18 KB
    queuelib-1.5.0             |            py36_0          21 KB
    constantly-15.1.0          |            py36_0          13 KB
    zope-1.0                   |            py36_0           3 KB
    w3lib-1.19.0               |            py36_0          31 KB
    pytest-runner-4.2          |            py36_0          12 KB
    twisted-17.5.0             |            py36_0         4.4 MB
    service_identity-17.0.0    |            py36_0          18 KB
    pyasn1-0.4.2               |   py36h22e697c_0         101 KB
    pyasn1-modules-0.2.1       |   py36hd1453cb_0          86 KB
    incremental-17.5.0         |   py36he5b1da3_0          25 KB
```

Numpy Arrays

- Can hold Same Datatype values only
- Contains very powerful and versatile set of methods

e.g.
```
import numpy as np
a = np.array([1,2,3,4,5,6])
a.min()
        a.mean()
len(a)
        np.append(a, 55)
```

Slicing Numpy Arrays

- When we slice a list it creates new list
- When we slice a Numpy Array it doesnt create a new array, saving memory

e.g

a = numpy.array([1,2,3,4,5])

b = a[2:]

⇒ **b** is like a **view** pointing to **original array**

⇒ changes to **b** reflect in **a** and **vice versa**

c = a.copy() => creates a new array c

Dictionaries

- A dictionary is an associative array
- Any key of the dictionary is associated (or mapped) to a value.
- The values of a dictionary can be any Python data type
- Dictionaries are unordered key-value-pairs.
- Dictionaries can easily be changed, can be shrunk and grown at run time

Operators on Dictionaries

Operator	Explanation
len(d)	returns the number of stored entries, i.e. the number of (key,value) pairs.
del d[k]	deletes the key k together with his value
k in d	True, if a key k exists in the dictionary d
k not in d	True, if a key k doesn't exist in the dictionary d

Dictionaries

d1 = {‘key1’ : ‘val1’ , ‘key2’ : ‘val2’, ‘key3’ : ‘val3’ }

d1[‘key1’]

dishes = ["pizza", “pretzel", "

countries = ["Italy", "Germany", "Sp

country_specialities = zip(countries

Two lists get combined like a zipper

convert the zipped list to dictionary

country_specialities_dict = dict(country_specialities)

Matrices

A lot of data used for processing is stored in tabular format and **Matrices** is one solution in Python to manage such type of data

A[0,:] **A[:,4]** **A[2,3]** **A[row,col]**

A =

	0	1	2	3	4
0	21	31	41	51	61
1	22	32	42	52	62
2	23	33	43	53	63

Matrix Operations

- **matrix1 + matrix2**
- **matrix1 - matrix2**
- **matrix1 * matrix2**
- **matrix1 / matrix2**
- **np.matrix.round(matrix1 / matrix2)**
- **np.nan_to_num(myMatrix)**
- **for index, item in enumerate(myMatrix)**