PROGRAMMING WITH PYTHON

NumPy and Pandas

TABLE OF CONTENT

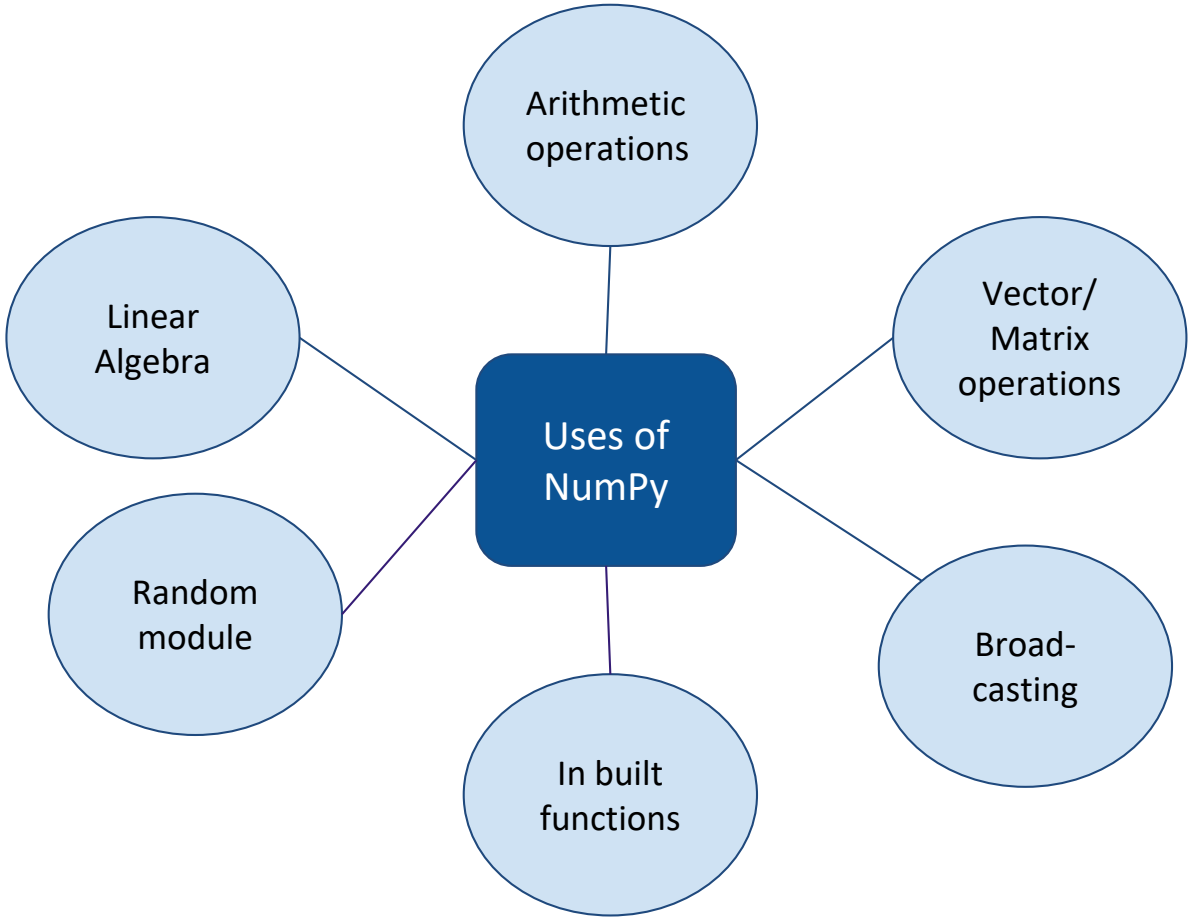
NumPy

PYTHON NumPy

- NumPy – Numerical Python
- Created by Travis Oliphant in 2005
- Open source
- Python Library for scientific computation
- Supports multi dimensional arrays and matrices
- Provides mathematical functions to operate on arrays

PYTHON NumPy

- Installation – conda install numpy
- Usually imported with alias name np



NumPy ARRAYS

- NumPy arrays are faster than Lists
- Stored at one continuous location in memory
- Values of the same type
- Indexing by a tuple
- Number of dimension = Rank of array
- Can create ndarray (array object in numpy) using array( ) function

```
import numpy as np
#1-D array
new_array = np.array([20,40,60,80])
print(f" \nArray : {new_array} ")
print(f" Type of array : {type(new_array)} ")
print(f" Array dimension : {new_array.ndim}")

#2-D array
new_array = np.array([[20,40], [60,80]])
print(f" \nArray : {new_array} ")
print(f" Type of array : {type(new_array)} ")
print(f" Array dimension : {new_array.ndim}")
```

```
Array : [20 40 60 80]
 Type of array : <class 'numpy.ndarray'>
 Array dimension : 1

Array : [[20 40]
 [60 80]]
 Type of array : <class 'numpy.ndarray'>
 Array dimension : 2
```

FUNCTIONS TO CREATE SPECIFIC ARRAYS

- zeros()
- ones()
- empty()
- eye()
- identity()
- empty_like()
- zeros_like()
- onces_like()
- asarray()
- full()
- full_like()

Functions for creating numpy arrays

```
zeros_array = np.zeros(4, dtype=int)
zeros_array
```

```
array([0, 0, 0, 0])
```

```
zeros_array = np.zeros((4,2), dtype=int)
zeros_array
```

```
array([[0, 0],
       [0, 0],
       [0, 0],
       [0, 0]])
```

```
ones_array = np.ones((2,4), dtype=int)
ones_array
```

```
array([[1, 1, 1, 1],
       [1, 1, 1, 1]])
```

NumPy MATRIX

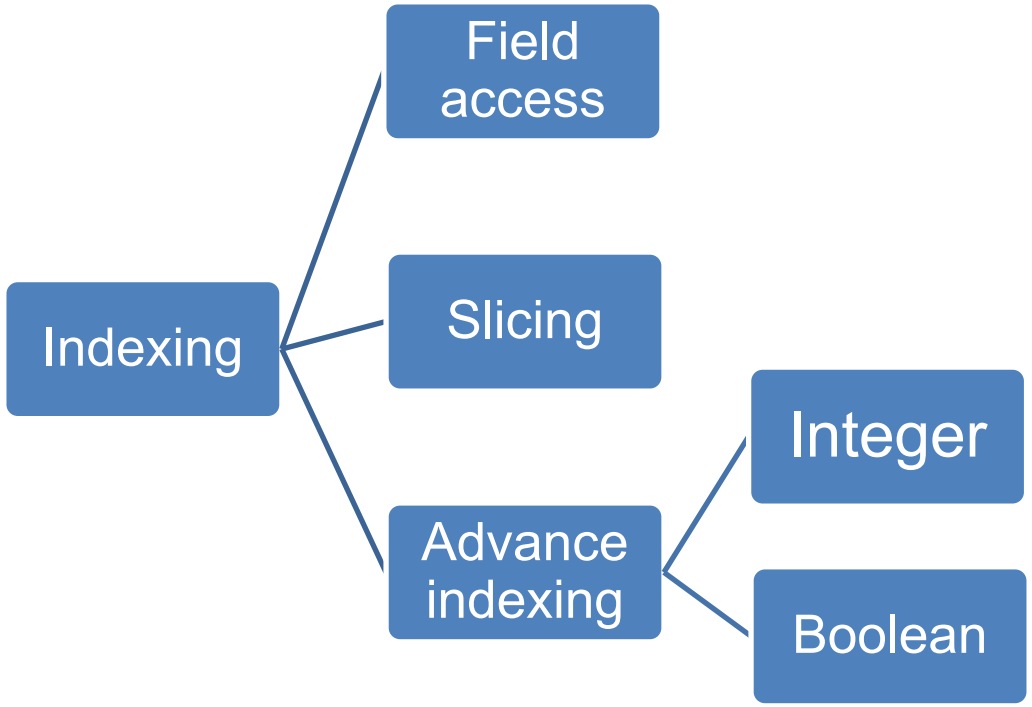
- Designated 2 dimensional array
- Has matrix multiplication and matrix squared operators
- Returns a matrix of same dimension (2-D)

DIFFERENCE BETWEEN NumPy ARRAY AND NumPy Matrix

Difference	NumPy N-D Array Object	NumPy Matrix Object
Dimensionality	Multi-dimensional	2 Dimensional
Multiplication with * operator	Element by element multiplication	Dot product
Multiplication with * * operator	Element wise squared operation	Matrix multiplication
Class	Base class	Derived class (Inherits from N-d array)
Inverse method	Do not have inverse method (.I)	Have inverse (.I) method
Usage/ Demand	Highly used	No longer recommended to use

NOTE: Matrix class is deprecated for future. Instead use Numpy arrays

INDEXING AND SLICING




```
new_array = np.array([1,10,20,2,30,3,40,4,50,5])
new_array[1:6:2]
```

```
array([10,  2,  3])
```

Field access

```
new_array[-3:3:-1]
```

```
array([ 4, 40,  3, 30])
```

Boolean Indexing

```
new_array[new_array > 5]
```

```
array([10,  6])
```

```
new_array = np.array([np.nan, np.nan, 2 ,4, 6 ,10, np.nan])
new_array[~np.isnan(new_array)]
```

```
array([ 2.,  4.,  6., 10.])
```

NumPy SELECTION TECHNIQUES

- where()
- choose()
- select()
- random.choice()
- invert()

```
#create an array
arr = np.arange(20)
```

```
#numpy.select()
np.select(condlist= [arr<5, arr>7], choicelist= [arr, arr**2])
```

```
array([  0,   1,   2,   3,   4,   0,   0,   0,  64,  81, 100, 121, 144,
       169, 196, 225, 256, 289, 324, 361])
```

```
#numpy.where()
np.where(arr < 6, arr, 2*arr)
```

```
array([ 0,  1,  2,  3,  4,  5, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32,
       34, 36, 38])
```

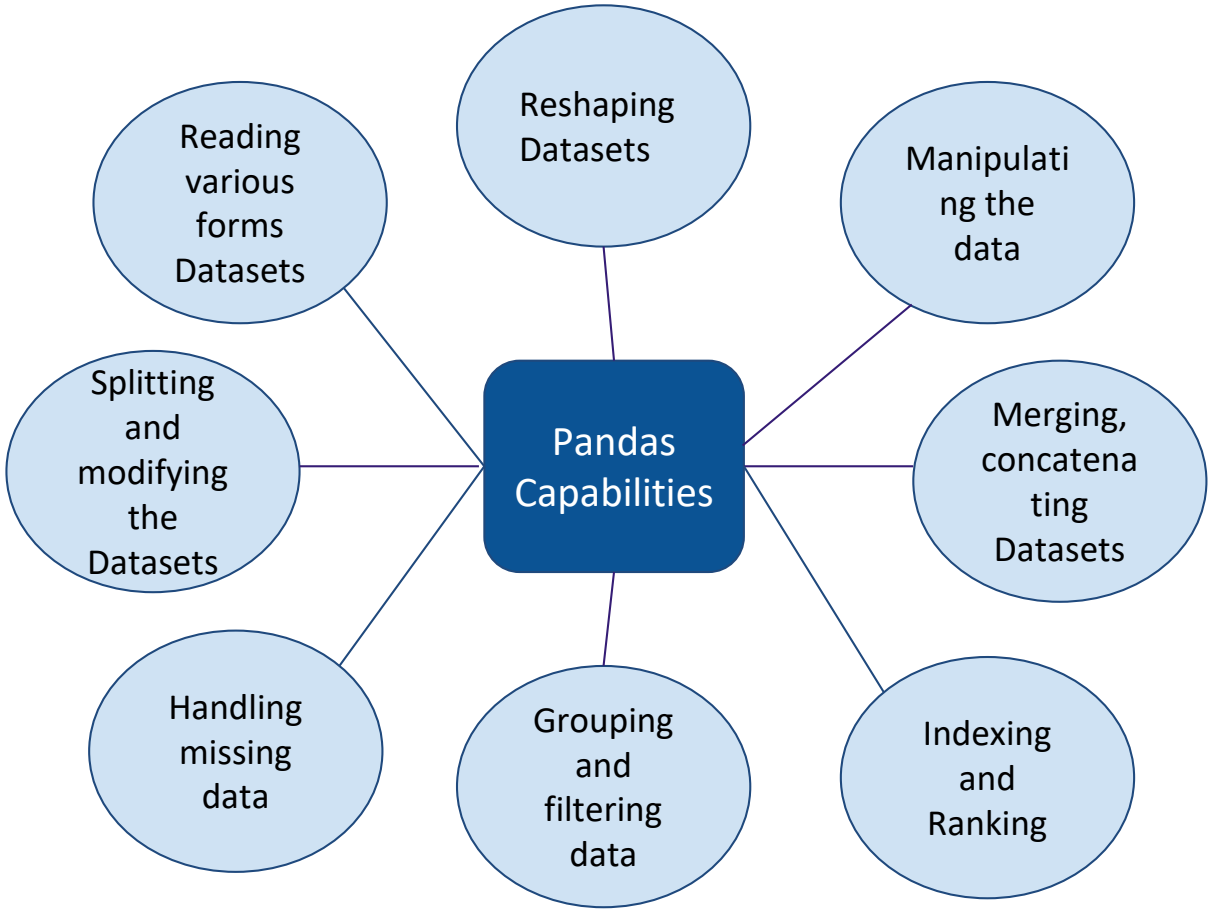
```
#numpy.choose()
choices = [[0, 1, 10, 9], [10, 20, 30, 40],
  [50, 51, 52, 53], [60, 61, 62, 63]]
np.choose([2,3,1,0], choices)
```

```
array([50, 61, 30,  9])
```

Pandas

Pandas IN PYTHON

- Build on top of NumPy
- Supports data manipulation
- Open source
- Stable release – 1.3.0 / 2
- Offer powerful data processing capabilities
- Usually imported with alias name pd



Pandas SERIES

- One dimensional array which can hold any data type
- Supports various methods:
- append()
- abs()
- add()
- agg()
- describe()
- all()
- apply()
- copy()
- count()

```
# creating series from dictionary
dictionary = {'w': 21, 'x': 400, 'y': 39, 'z': 10}
# keys and Series index should match, if not nan values are added
series = pd.Series(data=dictionary, index=['w', 'x', 'y', 'z'])
series
```

```
w     21
x    400
y     39
z     10
dtype: int64
```

Series methods

```
# describe() method
series = pd.Series([2,4,6,8,10,12])
series.describe()
```

```
count     6.000000
mean      7.000000
std       3.741657
min       2.000000
25%       4.500000
50%       7.000000
75%       9.500000
max      12.000000
dtype: float64
```

Pandas DATAFRAME

- 2 D tabular data structure
- Similar to SQL table
- Most widely used Pandas object
- Like Series, input can be anything (ndarray, dictionary, list, series, etc.)

```
# creating dataframe(df) from dictionary
dictionary = {'col1': [10.0, 20.0], 'col2': [30.0, np.nan]}
df = pd.DataFrame(data=dictionary)
df
```

	col1	col2
0	10.0	30.0
1	20.0	NaN

```
# creating df from nd array
df1 = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]),
                   columns=['col1', 'col2', 'col3'])
df1
```

	col1	col2	col3
0	1	2	3
1	4	5	6
2	7	8	9

DATAFRAME METHODS

- append()
- abs()
- add()
- agg()
- describe()
- all()
- apply()
- copy()
- count()
- align()
- corr()

```
# append() method
df.append(df1)
```

	col1	col2	col3
0	10.0	30.0	NaN
1	20.0	NaN	NaN
0	1.0	2.0	3.0
1	4.0	5.0	6.0
2	7.0	8.0	9.0

```
# apply method
df1.apply(lambda x: x**2)
```

	col1	col2	col3
0	1	4	9
1	16	25	36
2	49	64	81

Pandas INDEXING

- Indexing can be done through .iloc[ ] , .loc[ ]
- .loc[ ] uses label to select data
- .iloc[ ] uses position (integer location) to select data

```
df4.loc[['pune', 'nagpur']]
```

	max_speed(kmph)	distance(km)
pune	100	2500
nagpur	500	1500

iloc based indexing

```
df4.iloc[[0,2]]
```

	max_speed(kmph)	distance(km)
pune	100	2500
nagpur	500	1500

LOADING AND SAVING DATAFRAME

- **Pickle** : used for saving and loading Dataframe
- df.to_pickle(filename)
- pd.read_pickle(filename)
- **csv file** : can save Dataframe using csv file
- df.to_csv(filename)
- pd.read_csv(filename)

```
import pickle
df.to_pickle("Week3_Profile_Data.pkl")
```

```
df1 = pd.read_pickle("Week3_Profile_Data.pkl")
df1
```

	Name	Age	Sex	City	Country
0	Alex	30	Male	NY	USA
1	Anita	23	Female	Pune	India
2	Bliss	60	Female	Mumbai	India
3	Max	40	Male	Chicago	USA

SUMMARY

- Discussed Numpy arrays
- Functions to create numpy arrays
- Difference between numpy array and numpy matrix
- Indexing and slicing
- Numpy selection methods
- Pandas in Python
- Series and methods associated with it
- Dataframe and methods associated with it
- Indexing in Dataframe
- Dataframe loading and saving

Hands On

THANK YOU

Happy Learning ☺