Part-2

Python Libraries

RANKA
DATE / /
PAGE

Numpy Fundamental :-

What is numpy?

- Numpy is fundamental package for scientific computing.
- It is Laibrary provide multi dim array, object. (such as matrix) and assortment for fast operation on arrays including mathematics.

Numpy array —

At the core of Numpy Package, is the ndarray object. The encapsulate n-dim array of homogenous data type.

Numpy Arrays V/s Python Sequence :-

Numpy	Python List
• fixed array.	• dynamic array.
• if change size create new array delete previous one.	• Manipulate old array
• Convenience in Mathematical concept	
• Matplotlib, Scikit learn originate from it	

Numpy से ही इसके अन्दर जो main चीज है उसको Numpy array कहते हैं इस array में हम number को store कर सकते हैं और उन पर computation कर सकते हैं

Numpy is a God father.

Importing Laibrary -

impost Numpy as np.

RANKA
DATE / /
PAGE

Creating Numpy Array

1) np.array

```
a = np.array([1,2,3])
print(a)
```

2) 2D & 3D

```
a = np.array([[1,2,3],[4,5,6]])
```

Vector – 1D
Matrix – 2D
Tensor – 3D

3) dtype

```
np.array([1,2,3], dtype = bool)
```
float.
Complex

4) np.arange – like a range – in list

```
np.arange(1,11,2)
```

5) ~~np.~~ with reshape.

```
np.arange(1,13).reshape(5,2)
np.arange(16).reshape(2,2,2,2)
```

6) np.ones((3,4)) | np.zeros((3,1))

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

Note: default value in most of case is float

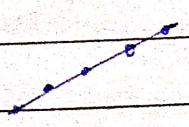
7) np.random.random((3,4))

random number array

8) np.linspace

Linearly seperable points generate

```
np.linspace(1, 10, 10)
```



9) np.identity → to draw Identity matrix.

```
np.identity(3)
```

32 → 4
64 → 8

DATE / /
PAGE

Array Attributes

a1 = np.arange(10)
a2 = np.arange(12, dtype = float).reshape(3,4)
a3 = np.arrange(8).reshape(2,2,2)

→ ndim

No. of dimension.

1-D
2-D
3-D

a1.ndim = 1
a2.ndim = 2
a3.ndim = 3

→ Shape → कि dimension में कितने rows & Column exist करते हैं

print(a3.shape) → (2,2,2) array (कि

→ Size → No. of items कितने हैं

a1.size = 10
a2.size = 12
a3.size = 8

→ item size → कितनी size occupy करते हैं

→ dtype

print(a2.dtype) → float 64
(a1.dtype) → int 32

Changing Data type

- astype

किसी भी array का datatype change करना

a3.astype(np.int32)

Array Operation :

Ex.

a1 = np.arange(12).reshape(3,4)

a2 = np.arange(12,24).reshape(3,4)

A Scalar Operation

1) Arithmetic Operation → Multiply each entry by scalar

a1 ** 2

2) Relational → a2 == 15 → Compare each element

Vector Operation :

1) Arithmetic →

a1 ** a2

→ Array function :

a1 = np.random.random((3,3))

np.max(a1)

sum

"

1) max, min, sum, prod

np. 0 → col 1 → row

np.max(a1, axis = 1)

2) mean / median / std / var

np.mean(a1)

np.mead(a1, axis=0)

RANKA
DATE / /
PAGE

Trigonometric function

np.sin(a,)

dot product

Criteria → m x n n x m

a2 = np.arange(12).reshape(3,4)

a3 = np.arange(12,24).reshape(4,3)

np.dot(a2,a3)

$$\begin{bmatrix} 2 & 4 & 5 \\ 5 & 6 & 8 \end{bmatrix} \cdot \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 2\times1+4\times2+5\times3 & 0 \\ 0 & 0 \end{bmatrix}$$

→ Logarithm & Exponent –

Logarithm

np.log(a,)

Exponent

np.exp

→ round / floor / ceil

np.ceil(

a2 [row] [column

Indexing and Slicing

a1 = np.arange(10)

a

Indexing → a2 [row, column]

a2 [2, 3]

Slicing → a1 = [1, 2, 3, 4, 5, 6, 7]

a1 [1 : 3]

$$a_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

1st row

a1 [0, :]

~~2nd row~~

3rd column

a1 [:, 2]

(ii) [[0, 1, 2, 3]

a2 = [4, 5, 6, 7]

[8, 9, 10, 11]]

(i) $\begin{bmatrix} 0 & 3 \\ 8 & 11 \end{bmatrix}$, a2 [: : 2, : : 3]

ii) $\begin{bmatrix} 1 & 3 \\ 9 & 11 \end{bmatrix}$, a2 [: : 2, 1 : : 2]

iii) [4, 7] , a2 [1 , 0 : : 3]

KANKA

Iterating :-

1) a1 = [1, 2, 3, 4]

```
for i in a1:
    print(i)
```

2) $a_2 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

```
for i in a1:
    print(i)
```

Every element :-

```
for i in np.nditer(a):
    print(i)
```

Reshape :-

1) reshape

2) Transpose

```
np.transpose(a2)
a2.T
```

3) ravel : convert any n dim array in 1D

```
a2.ravel()
```

Stacking :-

horizontal stacking

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$

Horizontal

Vertical

$\begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{bmatrix}$

np.hstack(a1, a2)

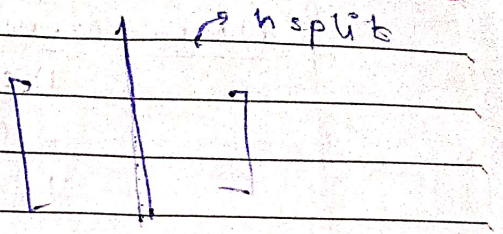
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix}$

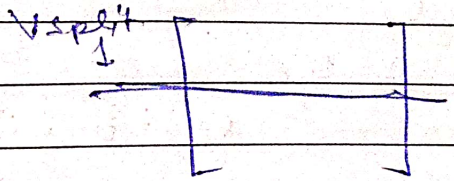
np.vstack(a1, a2)

RANKA
DATE / /
PAGE

Splitting: opposite of stacking

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}$$





List can store non homogenous data

RANKA
DATE / /
PAGE

Numpy array vs Python List

1) Speed
2) Memory
3) Convenience.

Speed

1) List

```
a = [i for i in range(10**7)]
b = [i for i in range(
c = []
for i in range(len(a)):
    c.append [a[i] + b[i]]
```

2) arrays

```
import numpy as np
a1 = np.arange(10**7)
a2 = np.arange(10**7, 2*10**7)
c = a1+a2
```

Memory

int-8 (256)
int-16 (32000)
int-32 ''

```
a = [i for i in range(10000000)]
import sys
sys.getsizeof(a)
```

Advanced Indexing

1) Fancy Indexing.

$$a_1 = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \end{bmatrix}$$

$a_1[[0,2,3]]$

$a_1[:, [0,2,3]]$

2) Boolean Indexing.

$$a = \begin{bmatrix} 50 & 65 \\ 72 & 24 \end{bmatrix}$$

1) Find all number greater than 50
2) Find all number which are even

i) `a[a>50]`

ii) `a[a%2==0]`

iii) `a[(a>50) & (a%2==0)]`

Brodcasting :

The term brodcasting describes how Numpy treats arrays with different shapes during arithmetic operation

The smaller array is "brodcast" across the larger array so that they have compatible shapes

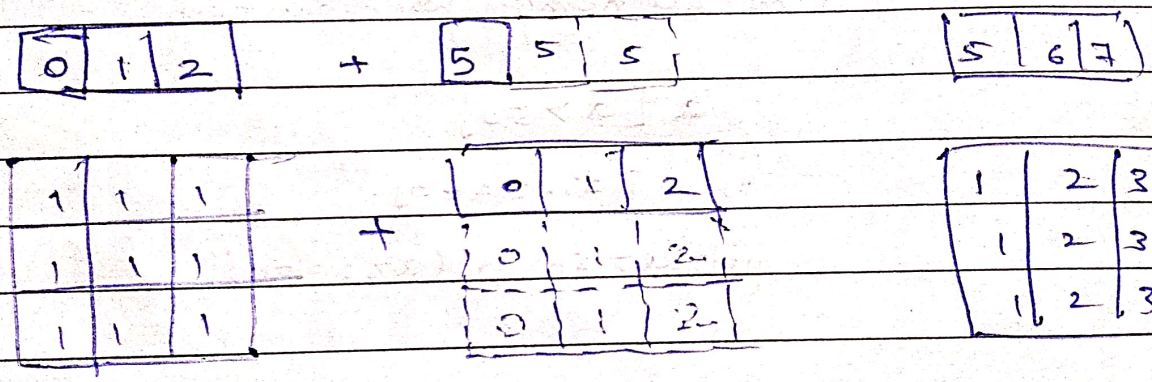
RANKA
DATE / /
PAGE

Same Shape

```
a = np.arange(6).reshape(2,3)
b = np.arange(6,12).reshape(2,3)
print(a)
print(b)
print(a+b)
```

different shape

```
a = np.arange(6).reshape(2,3)
b = np.arange(6,9).reshape(1,3)
print(a)
print(b)
print(a+b)
```



(3,2) (3,) → (1,3)

2D 1D

(3,3,3) (3,) → (1,1,3)

RANKA
DATE / /
PAGE

Working with Mathematical formulaes :

1) Sigmoid 2) Mean Square Error 3) Categorical Cross entropy.

$$\frac{1}{1+e^{-x}}$$

1) # sigmoid

```
def sigmoid (array):
    return 1/(1+ np.exp(-(array)))

a = np.arrange(10)
sigmoid (a)
```

2) Mean Square Error

$$= \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2$$

RANKA
DATE / /
PAGE

meshgrids

Lec Numpy Trick

1) np.sort

```
a = np.random.randint(1,100,25)
a
np.sort(a)
np.sort(a)[::-1]
```

2d-

```
b = np.random.randint(1,100,24).reshape(6,4)
b
np.sort(b)   ← row wise
np.sort(b,axis=1)   ← column wise
```

2) np.append

```
np.append(9,200)
np.append(b, np.ones(b.shape[0],1)), axis=1)
```

3) np. concatenate

```
c = np.arange(6).reshape(2,3)
d = np.arange(6,12).reshape(2,3)
print(c)
print(d)
np.concatenate((c,d),axis=1)
```

4) np. unique — provide all the unique vals

```
e = np.array([1,2,3,4,5,2,4,1,1,2,3,4])
np.unique(e)
```

5) np. expand_dims — convert 1d - 2d - 3d

```
np.expand_dims(a, axis=0).shape
np.expand_dims(
```

6) np. where → This function returns the indices of elements in an input array where the given condition is satisfied

* Find all value greater than 50

```
np.where(a>50)
```

* replace all values >50 with 0

```
np.where(condition, true, false)
np.where(a>50, 0, a)
```

7) np. argmax → It returns the maximum element of the array in a particular axis

```
np.argmax(a)
```

np. argmin

↳ Index position

axis → 0 Column
1 row.

np.cumsum : (cumulative sum)
Virat Kohli ने अपने career में कितने runs बनाए

```
a
np.cumsum(a)
b
np.cumsum(b)  |  np.cumsum(b, axis=1)
```

np.comprod :

```
np.comprod(a)
```

percentile : used to compute nth percentile of given data along specified axis
np.percentile.

```
np.percentile(a, 100)
np.percentile(a, 50)
```

$$P = \frac{n}{N} \times 100$$

n → No. of values below 'x'
N → Total no. of population

np.histogram :

5 15 9 17 25 23 29 → histogram

```
a
np.histogram(a, bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
```

RANKA
DATE / /
PAGE

np.corrcoef :-

Return Pearson product-moment correlation coefficients.

```
Salary = np.array([20000, 40000, 25000, 35000, 60,000])
experience = np.array([1,3,2,4,2])

np.corrcoef(salary, experience)
```

np.isin :- We can see that one array having values are checked in different

किसी Array में Multiple Item को search करके बताता है कि वो Item given है या नहीं

```
a
items = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
a[np.isin(a, items)]
```

np.flip :- Array को flip कर देता है

```
np.flip(a)
np.flip(b, axis = 1)
```

np.put :- change the

```
a
np.put(a, [0,1], [110, 530])
```

np.delete :- delete item on given indices

```
np.delete(a, [0, 1])
```

qwerty 123 @.

RANKA
DATE / /
PAGE

Set functions

1) np.union1d 2) np.intersect1d 3) np.setdiff1d
4) np.setxor1d 5) np.in1d (membership)

```
m = np.array([1,2,3,4,5])
n = np.array([3,4,5,6,7])
np.union1d(m,n)
np.intersect1d(m,n)
np.setdiff1d(m,n)
np.setxor1d(m,n)
```

np.clip : used to clip (limit) the values in an array.

```
a
np.clip(a, min
np.clip(a, a_min = 25, a_max = 75)
```

75, 75, 28, 50, 38