Chapter - 12

Numpy Basic Array :

* **Introduction** - Numpy की full form (Numerical python) है। इसे Travis Oliphant के द्वारा 2005 में develop किया गया था। यह एक open source Project है जिसे आप free में use कर सकते हैं। यह एक Python library है। लेकिन इसे ज्यादातर function C or C++ language से लिये गये हैं। Numpy Python में Scientific computing करने के लिए fundamental Package ही है। यह Python library ही है जो multi dimentional Array, object Array Provide करती है।

* **Why use Numpy** - सबसे ज्यादा Numpy का use Array के साथ work करने के लिए करते हैं। Array Python में list और tuple के समान है। Numpy का main purpose array object provide करना है। जो traditional Python से 50 गुणा ज्यादा fast है। Array object की Numpy में ndarray कहते हैं। (nd. ndimentional) यह कई सारे supporty function Provide करता है। जिससे Array के साथ easily work किया जा सकता है।

* **Numpy Installation** Numpy install करने के लिए आपके system में PIP Install होना चाहिए। (Package installer for Python) लेकिन Python में Pip inbuilt होता है लेकिन अगर यह नही है तो यह सबसे पहले Pip install करते हैं। Pip install होने के बाद निम्न command को run करते हैं। Pip install Numpy

* Import Numpy - Successfully import होने के बाद अब आप इसे बाकी Package या module की तरह import कर सकते है।

Import numpy as np

* Creating Numpy Array - Numpy का प्रयोग करके हम array को निम्न तीन तरीको से बना सकते है।

(i) Array function
(ii) ARRange function
(iii) Link space

(i) Array function - Array function का प्रयोग करके array बनाया जा सकता है। यह एक argument input करता है। argument list या tuple हो सकता है।

Syntax - <array name> = <numpy-obj>.array(<list, tuple name>

Example - import numpy as np
a = [23, 45, 56, 75, 78]
print (type (a))
h = np.array(a)
print (type (h))

Question = एक Program 10 person की age की list बनाता है इसे array में change करके calculate करे की इसमे कितने major और minor है।

```
import numpy as np
m = n = 0
a = [17, 15, 20, 25, 40, 55]
print (type (a))
b = np.array (a)
  print (type (b))
for i in b:
      if (i >= 18):
            m = m+1
      else:
            n = n+1
print (m)
print (n)
```

(ii) ARange Function - यह function value का एक array बनाता है। value argument के रूप में input की जाती है। यह तीन argument input करता है। start, stop, step

Syntax:

<array name> = <numpy-obj>.array ([start], [stop], [step])

Example -

```
import numpy as np
h = np.arrange (1,10,2)
print (h)
```

```
import numpy as m
h = m.arrange (20, 3, 2)
print (h)
```

(iii) linspace - यह function numpy array start तथा stop के बीच element की संख्या को समान दूरी पर return करता है।

syntax -

```
<array name> = <numpy-obj>.linspace (start, stop,
num = <number>, endpoint = True, restep = False)
```

- start - यह शुरुआती value को define करता है।
- Stop - यह series की last value return करता है।
- Num - यह element की संख्या को define करता है। कि array में कितने number होने चाहिए।
- restep - Restep को False set किया जाता है। यह function उसी array को वापिस return करता है जिसे एक variable में store किया जा सकता है। जब restep को True set किया जाता है तो यह function दो मान return करता है।

(i) n-d array
(ii) Step value

और दोनों को store करने के लिए अलग अलग variable देता है।

- endpoint - जब end point parameter false होता है। तो यह series को stop value से एक पहले समाप्त कर देता है।
लेकिन जब end point True होता है तो यह series stop value तक ही run करेगी।

Example - import numpy as np
a = np.linspace (1, 50, 10)
print (a)

default value of num = 50

Example import numpy as np
a = np.linspace (1, 5, 40, endpoint = True, retstep = True)
print (a)

* Dimensions of array -

Numpy में multi dimensional array बनाये जा सकते है। ndim function का use करके array की dimension का पता लगाया जा सकता है।

Syntax :-
<numpy-obj>.ndim (<array name>)

Example - import numpy as np
a = np.linspace (1, 5, 40)
print (np.ndim (a))

(1) zero dimensional array - Single element के द्वारा बनाया गया array zero dimensional array कहलाता है।

(2) One dimensional array - इस प्रकार के array में दो या अधिक dimensi. number हो सकते हैं।

Example - import numpy as np
a = np.array([23, 45, 78])
print(np.ndim(a))

(3) Multi dimensional array - Multi dimensional array बनाने के लिए nested list का use किया जाता है।

Syntax -
a = np.array([<list>]) # 1-D array
a = np.array([[<list>], [<list>]]) # 2-D array
a = np.array([[[<list>], [<list>]], [[<list>], [<list>]], [[<list>], [<list>]]]) # 3d array

Example - import numpy as np
a = np.array([2, 5]) 1-D array
a = np.array([[4, 6], [7, 8]]) 2D array
a = np.array([[[4, 7], [9, 6]], 3D array
[[5, 9], [9, 10]],
[[3, 7], [8, 3]]])
print(a)
print("dim", np.ndim(a))

* Shape of Array - Array की shape integers का tuple होती है। इसमें दो integers होती है। ये array की dimensions की लम्बाई को दर्शाते है।

Syntax -
<array obj> shape (<array name>)

Shape function का use करके किसी भी array की shape का पता लगाया जा सकता है।

Example - import numpy as np
a = np.array([[4,7,6],[7,8,4]])
print (a)
print (np.shape(a))

syntax 2 - <array name>.shape

Example - print (a.shape)

• Shape function का use करके array की shape को change किया जा सकता है।

Syntax 3 - <array name>.shape = (<shape tuple>)

Example - a.shape = (1,6)
print (a)

* Array index - Array में किसी element को Access करने तथा के लिए index number का use किया जाता है।

Syntax :
<array name>[<index>]

Example - a = np.array ([4,7,6]) → 1-d-array
print (a[1])

• 2d - array -

Syntax - <array name> [row no] [col no]

Example - a = np.array ([[4,7,6], [7,8,4]])
print (a[1][1]) = 8
print (a[0][2]) = 6

OR

Syntax - <array name> [row, col]

Example - a = np.array ([[4,7,6],[7,8,4],[8,4,2]])
print (a[2,2])

• 3d - array -

Syntax - <array name> [row] [col] [element no]

Example = a = np.array [[[2,3],[3,6]],[[4,6],[7,3]],
[[4,7],[2,9]]]

```
print (a)
print (a[1][0][0]
print (a[2][1][1]
```

* Array Slicing -

1. Slicing for 1D-Array - index number का use करके slicing की जा सकती है। इसमें 3 argument का use किया जाता है।
start, stop, step

Syntax - <array name>[start : stop : step]

Example - a = np.array([3,5,7,8,4,6,2])
print (a[3:5]) = 8,4
print (a[0:7:2]) = 3742
print (a[::-1]) = Reverse के लिए
print (a[:]) = सारी Print होगी
print (a[3:]) = 7 के बाद सारी

2. Slicing for 2D-Array - 2D Array slicing के लिए 6 argument की requirement होती है - क्योंकि row तथा column के लिए start, stop तथा step की value input की जाती है।

syntax -

<array name>[<start>:<stop>:<step>, <start>:<stop>:<step>]

Example -

```
import numpy as np
a = np.array([[2,3,5,6],[3,4,6,8],[5,7,8,9]])
print(a)
print(a[:2, :2])
print(a[1:3, 2:4])
print(a[0:3, 1:3])
```

~~pru~~

3. Slicing for 3-D Array - 3-D Array slicing के लिए 3 का start; stop, step की value enter की जाती है।

syntax -

<array name>[<start>:<stop>:<step>, <start>:<stop>:<step>, <start>:<stop>:<step>]

Example -

```
import numpy as np
a = np.array([[[3,4,5],[5,6,7]],[[4,5,8],
               [7,8,9]],[[6,7,8],[4,3,2]]])
print(a)
print(a[1:2, 0:2, 1:3])
```

* Reshape - इस function का use array की shape change करने के लिए किया जाता है।

Syntax -
<array-name>.reshape (<r>, <c>)

Example - import numpy as np
a = np.array [[3,4,5,6], [7,8,9,9], [2,8,9,2]]
print (a)
b = a.reshape (6,2)
print (b)

* Ones - यह function ones के साथ array को प्रारम्भिक मान देने के लिए किया जाता है।
यह function दो argument लेता है - जो-अव
(i) पहला tuple - जो array के size को show करता है।
(ii) D-type - Float या int के लिए

syntax :
<numpy-obj>.ones ((size/shape), dtype = int/Float)

Example - import numpy as np
a = np.ones ((5,6), dtype = int)
print (a)

* zeroes - इस function का use zero के साथ array को प्रारम्भिक मान देने के लिए किया जाता है। यह 2 argument -

(i) size
(iii) d-type input के रूप मे लेता है।

syntax -
<numpy-obj>.zeroes.((<size/shape>), dtype = int/float)

Example - import numpy as np
a = np.zeroes ((5,6), dtype = int)
print (a)

* ones-like - इस function का use New array बनाने के लिए तथा उस array के size को लेकर ones के साथ प्रारम्भिक मान देने के लिए।
यह 2 argument लेता है -

(i) array
(ii) d-type

syntax -
<numpy-obj>.ones-like.((<array name>, dtype = int/float)

Example -
import numpy as np
b = np.array ([2, 4, 5, 6, 7])
a = np.ones-like (b, dtype = int)
print (a)

* zeros-like - same as ones-like

* empty - array size तथा d-type argument के रूप में प्राप्त करके empty array बनाने के लिए इसका use किया जाता है।

syntax :
<numpy obj · empty ((shape/size), dtype = int/float)

Example - import numpy as np
a = np.empty ((4,4), dtype = int)
print (a)

* copy - इस function का use करके एक array की copy करके नया array बनाने में किया जाता है।

syntax -
<array 2> = <array 1> · copy ()

Example - import numpy as np
a = np.array ([3,4,6])
b = a.copy ()
print (a)
print (b)

* Identity - Identity array का type n*n square matrix होता है तथा मुख्य diagonal 1 तथा बाकि सभी 0 होते है। यह function array का size और argument को type के रूप में लेता है तथा एक identity matrix return करता है।

syntax -
<numpy obj>.identity ((shape/size), dtype = int/float)

Example - import numpy as np
a = np.identity (4) [(4, int) = • डिफॉल्ट]
print (a)

* eye - इस function के द्वारा 4 peremeter input करके एक identity matrix बनाया जाता है।

Syntax -
<numpy obj>. eye (<R>, <C>, <pod>, dtype = int/float)

Where - R = row
C = column
pod - position of diagonal

Example - import numpy as np
a = np.eye (4, 5, -2, int)
print (a)