NUMPY - Module / Library

Numpy is a powerfull python library used for numerical computations. It's especially useful for:

- Working with arrays & matrices
- Performing mathmatical & logical operations
- Efficient computation

syntax to import

```
import numpy as np. # where 'np' is a alias name
```

Syntax to check system-version.

```
import sys
sys.version.
```

creating a List.

```
my-list = [0,1,2,3,4]
my-list
```

converting list to array

```
arr = np.array(my_list)
arr
```

o/p:- array([0,1,2,3,4])

* arange() function

arange() function in the Numpy library that returns an array with evenly spaced values with in a given range.

Code

```
np.arange(10)
```

o/p:- array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

Code:

```
np.arange(10,20)
```

o/p:- array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])

P.T.O

* zeros() function.

- zero() function in the Numpy library that creates an array filled with all zeros.

NOTE*-

Parameter Tuning: System uses its own parameters.

-Hyperparameter Tuning: User changes the system parameters.

code:-

```
np.zeros(5)
```

o/p:- array([0., 0., 0., 0., 0.])

By default it prints in float values.
(it's an eg of parameter tuning, which is sys uses its own paras).

code:-

```
np.zeros(10, dtype = 'int')
```

o/p:- array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

It prints in "int" values, beoz This is called ~~Hyperpreter~~ Hyperparamete tuning which is user changed.

* ones() function.

- ones() is a function in Numpy library that creates an array filled with all ones.

code :-

```
np.ones(5)
```

o/p :- array([1, 1, 1, 1, 1]).

code

```
np.ones((3,3), dtype = int)
```

o/p :- array([[1, 1, 1
1, 1, 1
1, 1, 1]])

* random() function.

- random() is a function in the Numpy library (under np.random) used to generate random numbers.

- np.random.randint() is a Numpy function used to generate random integers in a specified range.

Code:- np.random.rand(3)

O/P:- array([0.123, 0.426, 0.786])

Code:- np.random.rand(3,9)

O/P:- a 3. # returns random integer. b/w 3-9; and always returns less than 2nd argument.

* Slicing in Matrix.

- Slicing is used to extract specific sub-array or parts of a matrix (2D array). In Numpy, we can use slicing to access specific rows, colums, or sub-matrices.

Code:- b= np.random.randint(10,20,(5,4))

O/P:- array([[13, 14, 16, 18
[14, 18, 18, 19
[17, 19, 19, 11
[17, 19, 17, 10
[13, 16, 13, 17]])

code: b[1:3]

o/p:- array([14, 18, 18, 19
17, 19, 19, 11])

* Advanced slicing

Advanced slicing allows more control over the selection of matrix. (we can specify the step size).

* Max - finds the largest number
* min - finds the smallest number
* Mean - finds the average of all numbers
* Median - finds the middle value when the numbers are sorted

Code arr.max() arr.min() arr.mean()

arr.median(). # for more eg's go through practice Jupyter NB.

* reshape() function.

- reshape() is a function in Numpy that allows you to change the shape of an array without changing it's data.

code :- arr = np.array([0,1,2,3,4,5])

arr

o/p = array([0, 1, 2, 3, 4, 5])

code

arr.reshape(2,3)

o/p :- array([[0, 1, 2],
[3, 4, 5]])

Order in ~~Recha~~ Reshaping.

- 'C' : C-style - major order (default)
- 'A' : A - Fortran style column - major order
- 'F' : Fortran - style order

code:-

```
arr.reshape (3,2, order = 'c')
```

o/p :- array ([[0,1],
[2,3],
[4,5]])

code:-

```
arr.reshape (3,2, order = 'A')
```

o/p :- array ([[0, 1],
[2, 2],
[4, 5]])

mostly 'c' & 'A' will gives you similar output.

code:-

```
arr.reshape (3,2, order = 'F')
```

o/p :- array ([[0, 3],
[1, 4],
[2, 5]]).

P.T.O

* Indexing in Numpy

- Indexing in Numpy allows you to access and modify specific elements or slices of an array

Code
```
mat = np.arange(0,100).reshape(10,10)
mat.
```

o/p:-
```
array([[ 0, 1, 2 ....
        ...
        ...97,98,99]])
```

It returns 10 * 10 matrix from 0-100 values, where values always less than 2nd argument i.e 100.

* Masking / filtering in Numpy

- Masking is a technique used in Numpy to filter or select elements from an array based on specific conditions.

code

```
mat = np.arange(0,100).reshape(10,10)
mat
```

o/p:- array([[0, 1, 2, ...
.....
.....
... 97, 98, 99]])

code

```
mat < 50
```

o/p:- array([[[Tru, Tru, Tru ....
.....
..... Fals, Fals
.....]]).

It returns conditions of values in matrix based and the code, ~~ice~~ is

code :-

```
mat > 50      # same as above code
mat == 50
```

P.T.O

Code :- mat [mat == 50]

o/p: array ([50]) # returns the value in a matrix.

Code :- a1 = mat [mat < 50]

o/p :- # returns all values which are less than "50".

Code a2 = mat [mat > 50] # returns values which are more than 50.

a3 = mat [mat >= 50] # return value which are equal & more than 50.