NUMPY PART - 2 BY VIRAT TIWARI

October 17, 2023

ALL ARRAY MANIPULATION OPERATIONS IN NUMPY -

[31]:
```
import numpy as np
```

[32]:
```
# np.random.randint ( ) function is used for getting the random data

arr=np.random.randint(1,10,(3,4))
```

[33]:
```
arr
```

[33]:
```
array([[8, 7, 3, 8],
       [9, 4, 6, 2],
       [1, 3, 7, 3]])
```

[34]:
```
# .reshape ( ) function is used for changing the shape of array

arr.reshape(6,2)
```

[34]:
```
array([[8, 7],
       [3, 8],
       [9, 4],
       [6, 2],
       [1, 3],
       [7, 3]])
```

[35]:
```
# T stands for the transpose that rotate the array

arr.T
```

[35]:
```
array([[8, 9, 1],
       [7, 4, 3],
       [3, 6, 7],
       [8, 2, 3]])
```

[36]:
```
# .flatten ( ) function is used for changing the data into the 1 - dimentional

arr.flatten()
```

```
[36]: array([8, 7, 3, 8, 9, 4, 6, 2, 1, 3, 7, 3])
```

```
[37]: arr1=np.array([1,2,3,4])
```

```
[38]: # ndim - no of dimentions shows that the total dimentions carried by a array

      arr1.ndim
```

```
[38]: 1
```

```
[39]: # .expand_dims ( ) function is used for expanding the dimention of array

      np.expand_dims(arr1,axis=1)
```

```
[39]: array([[1],
             [2],
             [3],
             [4]])
```

```
[40]: np.expand_dims(arr1,axis=0)
```

```
[40]: array([[1, 2, 3, 4]])
```

```
[41]: np.squeeze(arr)
```

```
[41]: array([[8, 7, 3, 8],
             [9, 4, 6, 2],
             [1, 3, 7, 3]])
```

```
[42]: data=np.array([[1],[2],[3]])
```

```
[43]: data
```

```
[43]: array([[1],
             [2],
             [3]])
```

```
[44]: # squeeze ( ) function gives the single dimentional array

      np.squeeze(data)
```

```
[44]: array([1, 2, 3])
```

```
[45]: arr1
```

```
[45]: array([1, 2, 3, 4])
```

[46]:

```
# .repeat ( ) function is used when we want to repeat the numbers of array , we␣
↪simply pass the value that how much time we want to repeat the numbers of␣
↪array

# Here we pass " 3 " so all values of array will repeat 3 times

np.repeat(arr1,3)
```

[46]:

```
array([1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4])
```

[48]:

```
# .roll ( ) function is rotate the no of array

np.roll(arr1,2)
```

[48]:

```
array([3, 4, 1, 2])
```

[50]:

```
np.diag(arr1)
```

[50]:

```
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
```

ARRAY BINARY OPERATIONS IN NUMPY -

[51]:

```
# This is how we generate the random array

arr1=np.random.randint(1,10,(3,4))
```

[52]:

```
arr2=np.random.randint(1,10,(3,4))
```

[53]:

```
arr1
```

[53]:

```
array([[5, 8, 9, 6],
       [7, 3, 5, 6],
       [1, 2, 5, 6]])
```

[54]:

```
arr2
```

[54]:

```
array([[2, 6, 8, 4],
       [5, 9, 6, 1],
       [5, 1, 4, 2]])
```

NOTE - BINARY OPERATIONS IS ALL ABOUT PLUS MINUS MULTIPLICATION AND DIVISION

[55]:

```
# Addition Operation -1
```

```
arr1+arr2
```

```
[55]: array([[ 7, 14, 17, 10],
             [12, 12, 11,  7],
             [ 6,  3,  9,  8]])
```

```
[56]: # Subtraction Operation - 2

      arr1-arr2
```

```
[56]: array([[ 3,  2,  1,  2],
             [ 2, -6, -1,  5],
             [-4,  1,  1,  4]])
```

```
[57]: # Multiplication Operation - 3

      arr1*arr2
```

```
[57]: array([[10, 48, 72, 24],
             [35, 27, 30,  6],
             [ 5,  2, 20, 12]])
```

```
[58]: # Division Operation - 4

      arr1/arr2
```

```
[58]: array([[2.5       , 1.33333333, 1.125     , 1.5       ],
             [1.4       , 0.33333333, 0.83333333, 6.        ],
             [0.2       , 2.        , 1.25      , 3.        ]])
```

```
[59]: # Remainder Operation - 5

      arr1%arr2
```

```
[59]: array([[1, 2, 1, 2],
             [2, 3, 5, 0],
             [1, 0, 1, 0]])
```

```
[60]: # Power Operation - 6

      arr1**arr2
```

```
[60]: array([[      25,   262144, 43046721,     1296],
             [   16807,    19683,    15625,        6],
             [       1,        2,      625,       36]])
```

[61]:
```
# And ( & ) Operation - 7

arr1 & arr2
```

[61]:
```
array([[0, 0, 8, 4],
       [5, 1, 4, 0],
       [1, 0, 4, 2]])
```

[62]:
```
# Or ( | ) Operation - 8

arr1|arr2
```

[62]:
```
array([[ 7, 14,  9,  6],
       [ 7, 11,  7,  7],
       [ 5,  3,  5,  6]])
```

[64]:
```
# Negate Operation - 9

~arr1
```

[64]:
```
array([[ -6,  -9, -10,  -7],
       [ -8,  -4,  -6,  -7],
       [ -2,  -3,  -6,  -7]])
```

[65]:
```
# Negate Operation - 10

~arr1
```

[65]:
```
array([[ -6,  -9, -10,  -7],
       [ -8,  -4,  -6,  -7],
       [ -2,  -3,  -6,  -7]])
```

[66]:
```
# Greater ( > )  Operation - 11

arr1>arr2
```

[66]:
```
array([[ True,  True,  True,  True],
       [ True, False, False,  True],
       [False,  True,  True,  True]])
```

[67]:
```
# Smaller ( < )  Operation - 11

arr1<arr2
```

[67]:
```
array([[False, False, False, False],
       [False,  True,  True, False],
       [ True, False, False, False]])
```

STRING FUNCTIONS IN NUMPY -

[69]:
```python
# WE made array with the help of .array ( ) function

# steps -

# 1 - np.array ( )
# 2 - pass list [ ] inside the array
# 3 - put the the variables whatever we want put inside the list of array

arr=np.array(["Virat","Tiwari"])
```

[70]:
```python
arr
```

[70]:
```
array(['Virat', 'Tiwari'], dtype='<U6')
```

[73]:
```python
# .char.upper ( ) function is used for converting the string or characters into the UPPER CASE of array

np.char.upper(arr)
```

[73]:
```
array(['VIRAT', 'TIWARI'], dtype='<U6')
```

[74]:
```python
# .char.title ( ) function is used for converting string into the title

np.char.title(arr)
```

[74]:
```
array(['Virat', 'Tiwari'], dtype='<U6')
```

[75]:
```python
# .char.capitalize ( ) function is used for capitalize the entire text of array

np.char.capitalize(arr)
```

[75]:
```
array(['Virat', 'Tiwari'], dtype='<U6')
```

MATHEMATICAL FUNCTIONS IN NUMPY -

[77]:
```python
arr1
```

[77]:
```
array([[5, 8, 9, 6],
       [7, 3, 5, 6],
       [1, 2, 5, 6]])
```

[78]:
```python
# TRIGONOMETRIC OPERATIONS

# This is how we find SIN , COS , TAN , TANH and many more trigometry functions
```

```
np.sin(arr1)
```

```
[78]: array([[-0.95892427,  0.98935825,  0.41211849, -0.2794155 ],
              [ 0.6569866 ,  0.14112001, -0.95892427, -0.2794155 ],
              [ 0.84147098,  0.90929743, -0.95892427, -0.2794155 ]])
```

```
[79]: # TRIGONOMETRIC OPERATIONS

      # This is how we find SIN , COS , TAN , TANH and many more trigometry functions␣
        ↪



      np.cos(arr1)
```

```
[79]: array([[ 0.28366219, -0.14550003, -0.91113026,  0.96017029],
              [ 0.75390225, -0.9899925 ,  0.28366219,  0.96017029],
              [ 0.54030231, -0.41614684,  0.28366219,  0.96017029]])
```

```
[80]: # TRIGONOMETRIC OPERATIONS

      # This is how we find SIN , COS , TAN , TANH and many more trigometry functions␣
        ↪




      np.tan(arr1)
```

```
[80]: array([[-3.38051501, -6.79971146, -0.45231566, -0.29100619],
              [ 0.87144798, -0.14254654, -3.38051501, -0.29100619],
              [ 1.55740772, -2.18503986, -3.38051501, -0.29100619]])
```

```
[81]: # TRIGONOMETRIC OPERATIONS

      # This is how we find SIN , COS , TAN , TANH and many more trigometry functions␣
        ↪




      np.tanh(arr1)
```

```
[81]: array([[0.9999092 , 0.99999977, 0.99999997, 0.99998771],
              [0.99999834, 0.99505475, 0.9999092 , 0.99998771],
              [0.76159416, 0.96402758, 0.9999092 , 0.99998771]])
```

```
[82]: # Used foe getting the logarithm of array

      np.log10(arr1)
```

```
[82]: array([[0.69897   , 0.90308999, 0.95424251, 0.77815125],
             [0.84509804, 0.47712125, 0.69897   , 0.77815125],
             [0.        , 0.30103   , 0.69897   , 0.77815125]])
```

[83]:

```
# Used for getting the exponential of array

np.exp(arr1)
```

```
[83]: array([[1.48413159e+02, 2.98095799e+03, 8.10308393e+03, 4.03428793e+02],
             [1.09663316e+03, 2.00855369e+01, 1.48413159e+02, 4.03428793e+02],
             [2.71828183e+00, 7.38905610e+00, 1.48413159e+02, 4.03428793e+02]])
```

[85]:

```
# Used foe getting the square root of array

#sqrt = square root

np.sqrt(arr1)
```

```
[85]: array([[2.23606798, 2.82842712, 3.        , 2.44948974],
             [2.64575131, 1.73205081, 2.23606798, 2.44948974],
             [1.        , 1.41421356, 2.23606798, 2.44948974]])
```

[87]:

```
# Used for getting power of array by passing the like power of 2 ,3 etc

np.power(arr1,2)
```

```
[87]: array([[25, 64, 81, 36],
             [49,  9, 25, 36],
             [ 1,  4, 25, 36]])
```

[88]:

```
# used foe getting mean or average of array

np.mean(arr1)
```

```
[88]: 5.25
```

[90]:

```
# Used foe getting median of array

np.median(arr1)
```

```
[90]: 5.5
```

[91]:

```
# Used for getting standard deviation of array

np.std(arr1)
```

```
[91]: 2.2407216099581255
```

[92]:

```
# Used foe getting variance of array

np.var(arr1)
```

[92]:

```
5.020833333333333
```

[93]:

```
# Used foe getting min value

np.min(arr1)
```

[93]:

```
1
```

[94]:

```
# Used For getting max value

np.max(arr1)
```

[94]:

```
9
```

NOTE - NUMPY CAN SOLVE WIDELY USED MATHEMATICAL PROBLEMS VERY EASILY

THANK YOU SO MUCH !!

YOURS VIRAT TIWARI :)