session7_numpy

January 16, 2022

0.0.1 Creating an Array using Numpy

[ ]:
```
# array creation 1D
import numpy as np
food = np.array(["Pakora","Samosa","Raita"])
food
```

[ ]:
```
array(['Pakora', 'Samosa', 'Raita'], dtype='<U6')
```

[ ]:
```
# array type
price = np.array([5,5,5])
price
type(price)
```

[ ]:
```
numpy.ndarray
```

[ ]:
```
len(price)
len(food)
```

[ ]:
```
3
```

[ ]:
```
# price[3] Index error
price[2]
price[0:2]
z = price[0:]
z
```

[ ]:
```
array([5, 5, 5])
```

[ ]:
```
p= food[1]
p
```

[ ]:
```
'Samosa'
```

[ ]:
```
#Array k functions
price.mean()
```

[ ]:
```
5.0
```

[ ]:
```
#zeros array
np.zeros(6)
#1s array
np.ones(5)
```

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

[ ]:
```
np.empty(7)
```

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

[ ]:
```
# with a spacing of interval
np.arange(2,20,5)
```

[ ]:
```
array([ 2,  7, 12, 17])
```

[ ]:
```
np.arange(10)
# last element is excluded
```

[ ]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[ ]:
```
# line space at specific interval (10 nums at fixed interval)
# isme akhri num ko include kar leta hay
np.linspace(3,20,10)
```

[ ]:
```
array([ 3.        ,  4.88888889,  6.77777778,  8.66666667, 10.55555556,
       12.44444444, 14.33333333, 16.22222222, 18.11111111, 20.        ])
```

[ ]:
```
# specify your data types
np.empty(50, dtype=np.int64)
```

[ ]:
```
array([3706497945030232697, 3328209646291068976, 4123389851770370361,
       3761694506697177401, 2340008602714185781, 4485090493615726966,
       3832057680150884384, 3832617357338806825, 4122818071313266484,
       2531084808905307188, 8241998674912177440, 8319675098974521977,
       4210425200352785012, 7883868074393078282, 8386103967300611952,
       3342349787993173104, 6998721842843253104, 4485033774059364467,
       7935409752961982526, 7021238737122897520, 3900165871320458595,
       4470430867062333484, 8079524940746866238, 2915077370344797292,
       3325662225219202168, 8079506593066003495, 3253604629844359208,
       8079506593066003495, 2318273471217938483, 4485033450774801703,
       7816329705848578110, 2968483698408189289, 2968470478583048202,
       8389203489669922314,   11584967480472366, 1009865545543123192,
                         0,     140723773312096,                3905,
                        -1, 7954884616238688484,                   0,
       8174913433131640140, 8029953815596917109, 7809639147579013484,
       3317475270149629472, 4195777553609138222, 8316292897441849354,
       8079584645411202592, 8367800735126286945], dtype=int64)
```

0.0.2 Array functions

[ ]:
```
a = np.array ([10,12,15,2,4,6,100,320,0,5,10,3])
a.sort()
a
```

[ ]:
```
array([  0,   2,   3,   4,   5,   6,  10,  10,  12,  15, 100, 320])
```

[ ]:
```
b = np.array([10.2,3.4,53.6,91.6,45.5])
c= np.concatenate((a,b))
c
c.sort()
c
```

[ ]:
```
array([  0. ,   2. ,   3. ,   3.4,   4. ,   5. ,   6. ,  10. ,  10. ,
        10.2,  12. ,  15. ,  45.5,  53.6,  91.6, 100. , 320. ])
```

0.1 2D arrays

[ ]:
```
# You have to have same no dimensions to concatenate
a = np.array([[1,2],[5,4]])
b = np.array([[6,7,],[8,9]])

c = np.concatenate((a,b),axis=0)
c
```

[ ]:
```
array([[1, 2],
       [5, 4],
       [6, 7],
       [8, 9]])
```

[ ]:
```
c = np.concatenate((a,b),axis=1)
c
```

[ ]:
```
array([[1, 2, 6, 7],
       [5, 4, 8, 9]])
```

0.2 3D Arrays

[ ]:
```
a = np.array([
     [[0,1,2,3],[4,5,6,7]],              # 1st 2d dim of a 3d
               [[0,1,2,3],[4,5,6,7]],    # 2nd dimension of a 3d
               [[0,1,2,3],[4,5,6,7]]     # 3rd dimension of a 3d
 ])
print(a)
a.ndim
```

```
[[[0 1 2 3]
  [4 5 6 7]]

 [[0 1 2 3]
  [4 5 6 7]]

 [[0 1 2 3]
  [4 5 6 7]]]
```

[ ]:
```
3
```

[ ]:
```
import numpy as np
b = np.array([
    [1,2,3,4],[1,2,3,4],[1,2,3,4]
             ])
print(b)
c=b.ndim
print("The dimension of array b is 2D",c)
d=b.size
print("The size (no of elements) of array b is",d)
e= a.shape
print("The shape  of array a is 3 dimension and 2 row 4 column as above example␣
 ↪that is ",e)

f= b.shape
print("The shape  of array b is 3 dimension and column first row last above␣
 ↪example that is ",f)
```

```
[[1 2 3 4]
 [1 2 3 4]
 [1 2 3 4]]
The dimension of array b is 2D 2
The size (no of elements) of array b is 12
The shape  of array a is 3 dimension and 2 row 4 column as above example that is
(3, 2, 4)
The shape  of array b is 3 dimension and column first row last above example
that is  (3, 4)
```

[ ]:
```
# reshaaping concept (like transpose) and dimension conversion
# a = np.arange(5)    this will generate error cux (3*2=6 me reshape is a␣
 ↪hassle) see line 4
a = np.arange(6)
print(a)
c=a.ndim
print("the dimension of a is",c)
b = a.reshape(3,2)
print(b)
d=b.ndim
```

```
print("The converted dimension of",c,"D into b is 2d=",d)
```

```
[0 1 2 3 4 5]
the dimension of a is 1
[[0 1]
 [2 3]
 [4 5]]
The converted dimension of 1 D into b is 2d= 2
```

[ ]:
```
# More Reshape
import numpy as np
f= np.reshape(b, newshape=(1,6), order='C')
f
```

[ ]:
```
array([[0, 1, 2, 3, 4, 5]])
```

Converting 1D array to 2D array by Axes Method

[ ]:
```
a = np.array([1,2,3,4,5,6,7,8,9])
print(a)
a.shape
print("the shape of 1D array is 9 elements",a.shape)

# row wise 2D conversion
b=a[np.newaxis,:]
print("a is converted to 2D. Nishani is braces",b)
print("the dimension of b is ",b.shape)

# column wise 2D conversion
c=a[:,np.newaxis]
print("a is converted to 2D. Nishani is braces",c)
print("the dimension of b is ",c.shape)
```

```
[1 2 3 4 5 6 7 8 9]
the shape of 1D array is 9 elements (9,)
a is converted to 2D. Nishani is braces [[1 2 3 4 5 6 7 8 9]]
the dimension of b is  (1, 9)
a is converted to 2D. Nishani is braces [[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
the dimension of b is  (9, 1)
```

Adjusting dimensions of converted 2D from above

[ ]:
```
d = c.reshape(3,3)
d[1][1]
```

[ ]: 5