\ ASSIGNMENT:3 LOKESH SARODE ROLL=158 PRN=202201040005

June 9, 2023

[17]:
```
import numpy as np
array1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
array1
```

[17]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

[18]:
```
array2=np.array([[11,12,13],[14,15,16],[17,18,19]])
array2
```

[18]:
```
array([[11, 12, 13], [14,
       15, 16],
      [17, 18, 19]])
```

[19]:
```
resultarray=array1+array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.add(array1,array2)
print("\nUsing Numpy Function:\n",resultarray)
```

```
Using Operator:
[[12 14 16]
[18 20 22]
 [24 26 28]]

Using Numpy Function:
 [[12 14 16]
 [18 20 22]
```

```
 [24 26 28]]
```

[20]:
```
resultarray=array1-array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.subtract(array1,array2)
print("\nUsing  Numpy  Function:\n",resultarray)
```

```
Using Operator:


 [[-10 -10 -10]
 [-10 -10 -10]
 [-10 -10 -10]]

Using Numpy Function:
 [[-10 -10 -10]
 [-10 -10 -10]
 [-10 -10 -10]]
```

[21]:
```
resultarray=array1*array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.multiply(array1,array2)
print("\nUsing  Numpy  Function:\n",resultarray)
```

```
Using Operator:
  [[ 11  24  39]
  [ 56  75  96]
 [119 144 171]]

Using Numpy Function:
  [[ 11  24  39]
 [ 56  75  96]
 [119 144 171]]
```

[22]:
```
resultarray=array1/array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.divide(array1,array2)
print("\nUsing  Numpy  Function:\n",resultarray)
```

```
Using Operator:
 [[0.09090909 0.16666667 0.23076923]
 [0.28571429 0.33333333 0.375      ]
 [0.41176471 0.44444444 0.47368421]]

Using Numpy Function:
 [[0.09090909 0.16666667 0.23076923]
 [0.28571429 0.33333333 0.375      ]
 [0.41176471 0.44444444 0.47368421]]
```

[23]:
```
resultarray=array1%array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.mod(array1,array2)
print("\nUsing  Numpy  Function:\n",resultarray)
```

```
Using Operator:
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Using Numpy Function:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

[26]:
```
resultarray=np.dot(array1,array2)
print("",resultarray)
```

```
 [[ 90  96 102]
 [216 231 246]
 [342 366 390]]
```

[27]:
```
resultarray=np.transpose(array1)
print(resultarray)
#Or
resultarray=array1.transpose()
print(resultarray)
```

```
[[1 4 7]
 [2 5 8]
 [3 6 9]]
[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

[28]:
```
resultarray=np.hstack((array1,array2))
resultarray
```

[28]:
```
array([[ 1,  2,  3, 11, 12, 13],
       [ 4,  5,  6, 14, 15, 16],
       [ 7,  8,  9, 17, 18, 19]])
```

[29]:
```
resultarray=np.vstack((array1,array2))
resultarray
```

[29]:
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
```

```
[ 7,  8,  9],  [11,  12,  13],
               [14,  15,  16],
               [17,  18,  19]])
```

[30]:
```
import numpy as np
nparray=np.arange(0,12,1).reshape(3,4)
nparray
```

[30]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9,  10,  11]])
```

```
nparray=np.linspace(start=0,stop=24,num=12).reshape(3,4)
nparray
```

[31]:

[31]:
```
array([[ 0.        ,  2.18181818,  4.36363636,  6.54545455],
       [ 8.72727273,  10.90909091,  13.09090909,  15.27272727],
       [17.45454545,  19.63636364,  21.81818182,  24.        ]])
```

```
nparray=np.empty((3,3),int)
nparray
```

[32]:

[32]:
```
array([[ 90,  96,  102],
       [216,  231,  246],
       [342,  366,  390]])
```

[33]:
```
nparray=np.empty_like(array1)
nparray
```

[33]:
```
array([[ 90,  96,  102],
       [216,  231,  246],
       [342,  366,  390]])
```

[34]:
```
nparray=np.identity(3)
nparray
```

[34]:
```
array([[1., 0., 0.], [0.,
       1., 0.],
      [0., 0., 1.]])
```

[35]:
```
array1=np.array([1,2,3,4,5])
array2=np.array([11,12,13,14,15])
print(array1)
print(array2)
```

```
[1 2 3 4 5]
[11 12 13 14 15]
```

[36]:
```
# Addition
print(np.add(array1,array2))
# Subtraction
print(np.subtract(array1,array2))
# Multiplication
print(np.multiply(array1,array2))
# Division
print(np.divide(array1,array2))
```

```
[12 14 16 18 20]
[-10 -10 -10 -10 -10]
[11[0.09090909 24 39 56  0.1666666775]  0.23076923 0.28571429 0.33333333]
```

[37]:
```
array1=np.array([1,2,3,4,5,9,6,7,8,9,9])
# Standard Deviation print(np.std(array1))
#Minimum
print(np.min(array1)) #Summation
print(np.sum(array1))
#Median
print(np.median(array1))
#Mean
print(np.mean(array1))
#Mode
from scipy import stats print("Most Frequent element=",stats.mode(array1)[0]) print("Number of Occarances=",stats.mode(array1)[1])
# Variance
print(np.var(array1))
```

```
2.7990553306073913
1
63
6.0
5.7272727272727275
Most Frequent element= [9]
Number of Occarances= [3]
7.834710743801653
```

```
<ipython-input-37-e89f83956b1b>:14: FutureWarning: Unlike other reduction
functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically
preserves the axis it acts along. In SciPy 1.11.0, this behavior will change:
the default value of `keepdims` will become False, the `axis` over which the
statistic is taken will be eliminated, and the value None will no longer be
accepted. Set `keepdims` to True or False to avoid this warning.
  print("Most Frequent element=",stats.mode(array1)[0])
<ipython-input-37-e89f83956b1b>:15: FutureWarning: Unlike other reduction
functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically
preserves the axis it acts along. In SciPy 1.11.0, this behavior will change:
the default value of `keepdims` will become False, the `axis` over which the
statistic is taken will be eliminated, and the value None will no longer
beaccepted. Set `keepdims` to True or False to avoid this warning.
   print("Number of Occarances=",stats.mode(array1)[1])
```

[38]:

```
array1=np.array([1,2,3],dtype=np.uint8) array2=np.array([4,5,6])
# AND
resultarray=np.bitwise_and(array1,array2) print(resultarray)
# OR
resultarray=np.bitwise_or(array1,array2) print(resultarray)
#LeftShift
resultarray=np.left_shift(array1,2)
print(resultarray)
#RightShift
resultarray=np.right_shift(array1,2) print(resultarray)
```

```
[0 0 2]
[5 7 7]
 [ 4 8 12]
```

```
[0 0 0]
```

[39]:
```
### You can get Binary Representation of Number ######
print(np.binary_repr(10,8))
resultarray=np.left_shift(10,2)
print(resultarray)
print(np.binary_repr(np.left_shift(10,2),8))
```

```
00001010
40
00101000
```

[40]:
```
array1=np.arange(1,10) print(array1)
newarray=array1.copy()
print(newarray)
##modification in Original Array
array1[0]=100 print(array1)
print(newarray)
```

```
[1 2 3 4 5 6 7 8 9]
[1 2 3 4 5 6 7 8 9]
                  [100 2 3 4 5 6 7 8 9]
[1 2 3 4 5 6 7 8 9]
```

[41]:
```
array1=np.arange(1,10) print(array1)
newarray=array1.view()
print(newarray)
##modification in Original Array
array1[0]=100 print(array1)
print(newarray)
```

```
[1 2 3 4 5 6 7 8 9]
[1 2 3 4 5 6 7 8 9]
[100 2 3 4 5           6   7   8   9]
[100 2 3 4 5           6   7   8   9]
```

```
array1=np.array([[1,2,3,12,5,7],[94,5,6,7,89,44],[7,8,9,11,13,14]])
print(array1)
```

[42]:

```
[[ 1  2  3 12  5  7]
 [94  5  6  7 89 44]
 [ 7  8  9 11 13 14]]
```

[43]:
```
np.sort(array1,axis=0)#Horizontally Sort
```

[43]:
```
array([[ 1,  2,  3,  7,  5,  7],
       [ 7,  5,  6, 11, 13, 14],
       [94,  8,  9, 12, 89, 44]])
```

[44]:
```
np.sort(array1,axis=1)# Vertically Sort
```

[44]:
```
array([[ 1,  2,  3,  5,  7, 12],
       [ 5,  6,  7, 44, 89, 94],
       [ 7,  8,  9, 11, 13, 14]])
```

```
array1=np.array([1,2,3,12,5,7])
np.searchsorted(array1,7,side="left")#Perform Search After sorting
```

[45]:

[45]: 3

[46]:
```
array1=np.array([1,2,3,12,5,7,0])
print(np.count_nonzero(array1))#Return total Non Zero element
print(np.nonzero(array1))#Return Index
print(array1.size)#Total Element
```

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

[47]:
```
array1=np.array(np.arange(1,5).reshape(2,2))
print(array1)
array2=np.array(np.arange(11,15).reshape(2,2))
print(array2)
```

```
[[1 2]
 [3 4]]
[[11 12]
 [13 14]]
```

[48]:
```
newarray=np.stack([array1,array2],axis=0)
print(newarray)
```

```
[[[ 1 2]
[ 3 4]]

 [[11 12]
  [13 14]]]
```

[49]:
```
newarray=np.stack([array1,array2],axis=1)
print(newarray)
```

```
[[[ 1 2]
  [11 12]]
```

```
 [[ 3  4]
  [13  14]]]
```

[50]:
```
array1=np.arange(1,10).reshape(3,3)
print(array1)
array2=np.arange(21,30).reshape(3,3)
print(array2)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[21 22 23]
 [24 25 26]
 [27 28 29]]
```

[51]:
```
np.append(array1,array2,axis=0)
```

[51]:
```
array([[ 1,  2,  3],  [ 4,
        5,  6],
       [ 7,  8,  9],
       [21,  22,  23],

        [24,  25,  26],
       [27,  28,  29]])
```

[52]:
```
np.append(array1,array2,axis=1)
```

[52]: array([[1, 2, 3, 21, 22, 23], [4,
5, 6, 24, 25, 26],
[7, 8, 9, 27, 28, 29]])

[53]:
```
array1=np.arange(1,10).reshape(3,3)
print(array1)
array2=np.arange(21,30).reshape(3,3)
print(array2)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[21 22 23]
 [24 25 26]
 [27 28 29]]
```

[54]:
```
np.concatenate((array1,array2),axis=0)
```

[54]: array([[1, 2, 3], [4,
5, 6],
[7, 8, 9],
[21, 22, 23],
[24, 25, 26],
[27, 28, 29]])

[55]:
```
np.concatenate((array1,array2),axis=1)
```

[55]: array([[1, 2, 3, 21, 22, 23], [4,
5, 6, 24, 25, 26],

```
       [ 7,  8,  9, 27, 28, 29]])
```

[60]:
```
import numpy as np

# using  loadtxt()
arr = np.loadtxt("testmarks1.csv",delimiter=",",skiprows=1)
print(type(arr))
arr.shape
```

```
<class 'numpy.ndarray'>
```

[60]: (10, 5)

```
EDS=arr[:,1]
print(EDS)
```

[64]:

```
[43.05 43.47 42.24 39.24 40.9 39.47 41.68 42.19 44.75 46.95]
```

```
SON=arr[:,2]
print(SON)
```

[63]:

```
[27.79 28.52 28.16 26.16 26.03 26.31 25.63 27.61 28.35 28.88]
```