Name: Payal Prashant Misal

Roll No: 641

PRN: 202201030024

Division: F(F2)

Assignment 3.1

Input:

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

Output:

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

Input:

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

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

#1 Matrix Operation

Input:

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

Output:

```
Using Operator:
 [[12 14]
 [16 18]]

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

Input:

```
#Subtraction
resultarray=array1-array2
print("\nUsing Operator:\n",resultarray)
```

```
resultarray=np.subtract(array1,array2)
print("\nUsing Numpy Function:\n",resultarray)
```

Output:

```
Using Operator:
 [[-10 -10]
 [-10 -10]]

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

Input:

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

Output:

```
Using Operator:
 [[11 24]
 [39 56]]

Using Numpy Function:
 [[11 24]
 [39 56]]
```

Input:

```
#Division
resultarry=array1/array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.divide(array1,array2)
print("\nUsing Numpy Function:\n",resultarray)
```

Output:

```
Using Operator:
 [[11 24]
 [39 56]]

Using Numpy Function:
 [[0.09090909 0.16666667]
 [0.23076923 0.28571429]]
```

Input:

```
#MOD
resultarry=array1%array2
print("\nUsing Operator:\n",resultarray)
```

```
resultarray=np.mod(array1,array2)
print("\nUsing Numpy Function:\n",resultarray)
```

Output:

Using Operator: [[1 2 3] [4 5 6] [7 8 9]]

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

Input:

```
#Dot Product
resultarray=np.dot(array1,array2)
print("",resultarray)
```

Output:

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

Input:

```
#Transpose
resultarray=np.transpose(array1)
print(resultarray)
```

Output:

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

#2 Horizontal and vertical stacking of Numpy Arrays

Input:

```
#2.1 Horizontal Stacking
resultarray=np.hstack((array1,array2))
resultarray
```

Output:

resultarray=np.hstack((array1,array2))

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

Input:

```
#2.2 Vertical Stacking
resultarray=np.vstack((array1,array2))
reaultarray
```

Output:

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

#3 Custom sequence generation

Input:

```
#3.1 Range
nparray=np.arange(0,12,1).reshape(3,4)
nparray
```

Output:

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

Input:

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

Output:

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

Input:

```
#3.3 Empty Array
nparray=np.empty((3,3),int)
nparray
```

Output:

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

Input:

```
#3.4 Emply like some other array
nparray=np.empty_like(array1)
nparray
```

Output:

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

Input:

```
#3.5 Index Matrix
nparray=np.identity(3)
nparray
```

Output:

array([[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]])

#4 Arithmetic and statistical operations,Mathmatical operations,bitwise operators

Input:

```
#4.1 Arithmatic operation
array1=np.array([1,2,3,4,5])
array2=np.array([11,12,13,14,15])
print(array1)
print(array2)
```

Output:

[1 2 3 4 5]

[11 12 13 14 15]

Input:

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

Output:

[12 14 16 18 20]

[-10 -10 -10 -10 -10] [11 24 39 56 75]

[0.09090909 0.16666667 0.23076923 0.28571429 0.33333333

Input:

```
#4.2 Statistical and mathmatical operation
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))
```

Output:

2.7990553306073913

1

63 6.0

5.7272727272727275

Most Frequent element= [9]

Number of Occarances= [3]

Input:

```
#4.3 Bitwise Operator
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)
```

Output:

[0 0 2]

[5 7 7]

[4 8 12] [0 0 0]

Input:

```
print(np.binary_repr(10,8))
resultarray=np.left_shift(10,2)
```

```
print(resultarray)
print(np.binary_repr(np.left_shift(10,2),8))
```

Output:

00001010

40

00101000

#5 Copying and viewing array

Input:

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

Output:

[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]

Input:

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

Output:

[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]

#6 Serching

Input:

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

Output:

[[1 2 3 12 5 7]

[94 5 6 7 89 44]

[7 8 9 11 13 14]]

Input:

```
np.sort(array1,axis=0)#Horizontally Sort
```

Output:

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

Input:

```
np.sort(array1,axis=1)# Vertically Sort
```

Output:

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

#7 Searching

Input:

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

Output:

3

#8 Counting

Input:

```
#8 Counting
```

```
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
```

Output:

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

#9 Data Stacking

Input:

```
#9 Data Stacking
array1=np.array(np.arange(1,5).reshape(2,2))
print(array1)
array2=np.array(np.arange(11,15).reshape(2,2))
print(array2)
```

Output:

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

Input:

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

Ouput:

[[[1 2] [3 4]]

[[11 12]

[13 14]]]

Input:

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

Output:

[[[1 2]

[11 12]]

[[3 4]

[13 14]]]

#10 Append

Input:

```
#10 Append
array1=np.arange(1,10).reshape(3,3)
print(array1)
array2=np.arange(21,30).reshape(3,3)
print(array2)
```

Output:

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

Input:

```
np.append(array1,array2,axis=0)
```

Output:

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

Input:

```
np.append(array1,array2,axis=1)
```

Output:

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

#11 Concat

Input:

```
#11 Concat
array1=np.arange(1,10).reshape(3,3)
print(array1)
array2=np.arange(21,30).reshape(3,3)
print(array2)
```

Output:

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

Assignment 3.2

Input:

```
import numpy as np
d1=np.genfromtxt("/content/testmarks1.csv",delimiter=",")
print(d1)
EDS=d1[:,1]
print(type(EDS))
print(max(EDS))
```

Output:

```
[[   nan    nan    nan    nan    nan]
 [801.    43.05  27.79  28.7   27.79]
 [802.    43.47  28.52  28.98  27.89]
 [803.    42.24  28.16  28.16  25.63]
 [804.    39.24  26.16  26.16  26.16]
 [805.    40.9   26.03  27.27  25.65]
 [806.    39.47  26.31  26.31  25.21]
 [807.    41.68  25.63  27.79  25.46]
 [808.    42.19  27.61  28.13  26.21]
 [809.    44.75  28.35  29.83  28.21]
 [810.    46.95  28.88  31.3   28.53]]
<class 'numpy.ndarray'>
Nan
```

Input:

```
import numpy as np
d2=np.genfromtxt("/content/testmarks2.csv",delimiter=",")
print(d2)
EDS=d1[:,1]
print(type(EDS))
print(max(EDS))
```

Output:

```
[[   nan    nan    nan    nan    nan]
 [801.    28.48  34.18  30.56  22.23]
 [802.    28.1   33.72  30.68  22.82]
 [803.    26.16  31.39  28.2   22.53]
 [804.    26.16  31.39  28.78  20.93]
 [805.    26.1   31.32  28.22  20.82]
 [806.    25.45  30.54  27.73  21.05]
 [807.    26.16  31.39  28.01  20.51]
 [808.    27.44  32.93  28.83  22.08]
 [809.    28.63  34.35  31.03  22.68]
 [810.    30.35  36.42  31.38  23.1 ]]
<class 'numpy.ndarray'>
Nan
```

input:

```
import numpy as np
d1=np.genfromtxt("/content/testmarks1.csv",delimiter=",")
print(d1)
EDS=d1[1:,1]
print(type(EDS))
print(max(EDS))
```

Output:

```
[[   nan    nan    nan    nan    nan]
 [801.    43.05  27.79  28.7   27.79]
```

```
 [802.    43.47  28.52  28.98  27.89]
 [803.    42.24  28.16  28.16  25.63]
 [804.    39.24  26.16  26.16  26.16]
 [805.    40.9   26.03  27.27  25.65]
 [806.    39.47  26.31  26.31  25.21]
 [807.    41.68  25.63  27.79  25.46]
 [808.    42.19  27.61  28.13  26.21]
 [809.    44.75  28.35  29.83  28.21]
 [810.    46.95  28.88  31.3   28.53]]
<class 'numpy.ndarray'>
46.95
```

Input:

```
import numpy as np
d1=np.genfromtxt("/content/testmarks2.csv",delimiter=",")
print(d2)
EDS=d1[1:,1]
print(type(EDS))
print(max(EDS))
```

Output:

```
[[   nan    nan    nan    nan    nan]
 [801.    28.48  34.18  30.56  22.23]
 [802.    28.1   33.72  30.68  22.82]
 [803.    26.16  31.39  28.2   22.53]
 [804.    26.16  31.39  28.78  20.93]
 [805.    26.1   31.32  28.22  20.82]
 [806.    25.45  30.54  27.73  21.05]
 [807.    26.16  31.39  28.01  20.51]
 [808.    27.44  32.93  28.83  22.08]
 [809.    28.63  34.35  31.03  22.68]
 [810.    30.35  36.42  31.38  23.1 ]]
<class 'numpy.ndarray'>
30.35
```

Input:

```
d1=np.genfromtxt("/content/testmarks1.csv",delimiter=",")
print(d1)
EDS=d1[1:,1]
print(type(EDS))
print(max(EDS))
np.count_nonzero(EDS>40)
```

Output:

```
[[ nan  nan  nan  nan  nan]
 [801. 43.05 27.79 28.7  27.79]
 [802.  43.47 28.52 28.98 27.89]
 [803.  42.24 28.16 28.16 25.63]
 [804.  39.24 26.16 26.16 26.16]
 [805. 40.9  26.03 27.27 25.65]
 [806.  39.47 26.31 26.31 25.21]
 [807.  41.68 25.63 27.79 25.46]
```

```
 [808.   42.19  27.61  28.13 26.21]
 [809.   44.75  28.35  29.83 28.21]
 [810.   46.95  28.88  31.3  28.53]]
<class 'numpy.ndarray'>
46.95
8
```

Input:

```
d1=np.genfromtxt("/content/testmarks2.csv",delimiter=",")
print(d2)
EDS=d1[1:,1]
print(type(EDS))
print(max(EDS))
np.count_nonzero(EDS>40)
```

Output:

```
[[  nan   nan   nan   nan   nan]
 [801.   28.48  34.18  30.56 22.23]
 [802.   28.1  33.72  30.68  22.82]
 [803.   26.16  31.39  28.2  22.53]
 [804.   26.16  31.39  28.78 20.93]
 [805.   26.1  31.32  28.22  20.82]
 [806.   25.45  30.54  27.73 21.05]
 [807.   26.16  31.39  28.01 20.51]
 [808.   27.44  32.93  28.83 22.08]
 [809.   28.63  34.35  31.03 22.68]
 [810.   30.35  36.42  31.38 23.1 ]]
<class 'numpy.ndarray'>
30.35
0
```

Input:

```
#Addition
result=d1+d2
print("\nUsing Operator:\n",result)
resultarray=np.add(d1,d2)
print("\nUsing Numpy Function:\n",result)
```

Output:

```
Using Operator:
 [[    nan     nan     nan     nan     nan]
 [1602.    56.96   68.36   61.12   44.46]
 [1604.    56.2    67.44   61.36   45.64]
 [1606.    52.32   62.78   56.4    45.06]
 [1608.    52.32   62.78   57.56   41.86]
 [1610.    52.2    62.64   56.44   41.64]
 [1612.    50.9    61.08   55.46   42.1 ]
 [1614.    52.32   62.78   56.02   41.02]
 [1616.    54.88   65.86   57.66   44.16]
 [1618.    57.26   68.7    62.06   45.36]
```

```
 [1620.      60.7     72.84    62.76    46.2 ]]
```

```
Using Numpy Function:
 [[     nan      nan      nan      nan      nan]
 [1602.      56.96    68.36    61.12    44.46]
 [1604.      56.2     67.44    61.36    45.64]
 [1606.      52.32    62.78    56.4     45.06]
 [1608.      52.32    62.78    57.56    41.86]
 [1610.      52.2     62.64    56.44    41.64]
 [1612.      50.9     61.08    55.46    42.1 ]
 [1614.      52.32    62.78    56.02    41.02]
 [1616.      54.88    65.86    57.66    44.16]
 [1618.      57.26    68.7     62.06    45.36]
 [1620.      60.7     72.84    62.76    46.2 ]]
```

Input:

```
#Subtraction
resul=d1-d2
print("\nUsing Operator:\n",resul)
resultarray=np.subtract(d1,d2)
print("\nUsing Numpy Function:\n",result)
```

Output:

```
Using Operator:
 [[nan nan nan nan nan]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]]
```

```
Using Numpy Function:
 [[     nan      nan      nan      nan      nan]
 [1602.      56.96    68.36    61.12    44.46]
 [1604.      56.2     67.44    61.36    45.64]
 [1606.      52.32    62.78    56.4     45.06]
 [1608.      52.32    62.78    57.56    41.86]
 [1610.      52.2     62.64    56.44    41.64]
 [1612.      50.9     61.08    55.46    42.1 ]
 [1614.      52.32    62.78    56.02    41.02]
 [1616.      54.88    65.86    57.66    44.16]
 [1618.      57.26    68.7     62.06    45.36]
 [1620.      60.7     72.84    62.76    46.2 ]]
```

Input:

```
#Multiplication
resultarray=d1*d2
print("\nUsing Operator:\n",result)
resultarray=np.multiply(d1,d2)
```

```
print("\nUsing Numpy Function:\n",result)
```

Output:

```
Using Operator:
 [[    nan     nan     nan     nan     nan]
 [1602.    56.96   68.36   61.12   44.46]
 [1604.    56.2    67.44   61.36   45.64]
 [1606.    52.32   62.78   56.4    45.06]
 [1608.    52.32   62.78   57.56   41.86]
 [1610.    52.2    62.64   56.44   41.64]
 [1612.    50.9    61.08   55.46   42.1 ]
 [1614.    52.32   62.78   56.02   41.02]
 [1616.    54.88   65.86   57.66   44.16]
 [1618.    57.26   68.7    62.06   45.36]
 [1620.    60.7    72.84   62.76   46.2 ]]

Using Numpy Function:
 [[    nan     nan     nan     nan     nan]
 [1602.    56.96   68.36   61.12   44.46]
 [1604.    56.2    67.44   61.36   45.64]
 [1606.    52.32   62.78   56.4    45.06]
 [1608.    52.32   62.78   57.56   41.86]
 [1610.    52.2    62.64   56.44   41.64]
 [1612.    50.9    61.08   55.46   42.1 ]
 [1614.    52.32   62.78   56.02   41.02]
 [1616.    54.88   65.86   57.66   44.16]
 [1618.    57.26   68.7    62.06   45.36]
 [1620.    60.7    72.84   62.76   46.2 ]]
```

Input:

```
#Division
resultarry=d1/d2
print("\nUsing Operator:\n",result)
resultarray=np.divide(d1,d2)
print("\nUsing Numpy Function:\n",result)
```

Output:

```
Using Operator:
 [[    nan     nan     nan     nan     nan]
 [1602.    56.96   68.36   61.12   44.46]
 [1604.    56.2    67.44   61.36   45.64]
 [1606.    52.32   62.78   56.4    45.06]
 [1608.    52.32   62.78   57.56   41.86]
 [1610.    52.2    62.64   56.44   41.64]
 [1612.    50.9    61.08   55.46   42.1 ]
 [1614.    52.32   62.78   56.02   41.02]
 [1616.    54.88   65.86   57.66   44.16]
 [1618.    57.26   68.7    62.06   45.36]
 [1620.    60.7    72.84   62.76   46.2 ]]

Using Numpy Function:
 [[    nan     nan     nan     nan     nan]
 [1602.    56.96   68.36   61.12   44.46]
```

```
 [1604.      56.2     67.44    61.36    45.64]
 [1606.      52.32    62.78    56.4     45.06]
 [1608.      52.32    62.78    57.56    41.86]
 [1610.      52.2     62.64    56.44    41.64]
 [1612.      50.9     61.08    55.46    42.1 ]
 [1614.      52.32    62.78    56.02    41.02]
 [1616.      54.88    65.86    57.66    44.16]
 [1618.      57.26    68.7     62.06    45.36]
 [1620.      60.7     72.84    62.76    46.2 ]]
```

Input:

```
#MOD
resultarry=d1%d2
print("\nUsing Operator:\n",result)
resultarray=np.mod(d1,d2)
print("\nUsing Numpy Function:\n",result)
```

Output:

```
Using Operator:
 [[    nan      nan      nan      nan      nan]
 [1602.      56.96    68.36    61.12    44.46]
 [1604.      56.2     67.44    61.36    45.64]
 [1606.      52.32    62.78    56.4     45.06]
 [1608.      52.32    62.78    57.56    41.86]
 [1610.      52.2     62.64    56.44    41.64]
 [1612.      50.9     61.08    55.46    42.1 ]
 [1614.      52.32    62.78    56.02    41.02]
 [1616.      54.88    65.86    57.66    44.16]
 [1618.      57.26    68.7     62.06    45.36]
 [1620.      60.7     72.84    62.76    46.2 ]]
```

```
Using Numpy Function:
 [[    nan      nan      nan      nan      nan]
 [1602.      56.96    68.36    61.12    44.46]
 [1604.      56.2     67.44    61.36    45.64]
 [1606.      52.32    62.78    56.4     45.06]
 [1608.      52.32    62.78    57.56    41.86]
 [1610.      52.2     62.64    56.44    41.64]
 [1612.      50.9     61.08    55.46    42.1 ]
 [1614.      52.32    62.78    56.02    41.02]
 [1616.      54.88    65.86    57.66    44.16]
 [1618.      57.26    68.7     62.06    45.36]
 [1620.      60.7     72.84    62.76    46.2 ]]
```

Input:

```
#Dot Product
resultarray=np.dot(d1,d2)
print("",resultarray)
```

Output:

Input:

```
#Transpose
```

```
resultarray=np.transpose(d1)
print(resultarray)
```

Output:

```
[[   nan 801.   802.   803.   804.   805.   806.   807.   808.   809.
  810.  ]
 [   nan  28.48  28.1   26.16  26.16  26.1   25.45  26.16  27.44  28.63
   30.35]
 [   nan  34.18  33.72  31.39  31.39  31.32  30.54  31.39  32.93  34.35
   36.42]
 [   nan  30.56  30.68  28.2   28.78  28.22  27.73  28.01  28.83  31.03
   31.38]
 [   nan  22.23  22.82  22.53  20.93  20.82  21.05  20.51  22.08  22.68
   23.1 ]]
```

Input:

```
#Mean
resultd=d1+d2/2

print("\nUsing Operator:\n",resultd)
resultd=np.add(d1,d2)
print("\nUsing NumpyFunction:\n",resultd)
```

Output:

```
Using Operator:
 [[     nan      nan      nan      nan      nan]
 [1201.5    42.72    51.27    45.84    33.345]
 [1203.     42.15    50.58    46.02    34.23 ]
 [1204.5    39.24    47.085   42.3     33.795]
 [1206.     39.24    47.085   43.17    31.395]
 [1207.5    39.15    46.98    42.33    31.23 ]
 [1209.     38.175   45.81    41.595   31.575]
 [1210.5    39.24    47.085   42.015   30.765]
 [1212.     41.16    49.395   43.245   33.12 ]
 [1213.5    42.945   51.525   46.545   34.02 ]
 [1215.     45.525   54.63    47.07    34.65 ]]
```

```
Using NumpyFunction:
 [[    nan     nan     nan     nan     nan]
 [1602.    56.96   68.36   61.12   44.46]
 [1604.    56.2    67.44   61.36   45.64]
 [1606.    52.32   62.78   56.4    45.06]
 [1608.    52.32   62.78   57.56   41.86]
 [1610.    52.2    62.64   56.44   41.64]
 [1612.    50.9    61.08   55.46   42.1 ]
 [1614.    52.32   62.78   56.02   41.02]
 [1616.    54.88   65.86   57.66   44.16]
 [1618.    57.26   68.7    62.06   45.36]
 [1620.    60.7    72.84   62.76   46.2 ]]
```

Input:

```
#Horizontal Stacking
```

```
resultarray=np.hstack((d1,d2))
resultarray
```

Output:

```
array([[ nan,  nan,  nan,  nan,  nan,  nan,  nan,  nan,  nan,  nan], [801. ,
28.48, 34.18, 30.56, 22.23, 801. , 28.48, 34.18, 30.56, 22.23], [802. ,
28.1 , 33.72, 30.68, 22.82, 802. , 28.1 , 33.72, 30.68, 22.82], [803. ,
26.16, 31.39, 28.2 , 22.53, 803. , 26.16, 31.39, 28.2 , 22.53], [804. ,
26.16, 31.39, 28.78, 20.93, 804. , 26.16, 31.39, 28.78, 20.93], [805. ,
26.1 , 31.32, 28.22, 20.82, 805. , 26.1 , 31.32, 28.22, 20.82], [806. ,
25.45, 30.54, 27.73, 21.05, 806. , 25.45, 30.54, 27.73, 21.05], [807. ,
26.16, 31.39, 28.01, 20.51, 807. , 26.16, 31.39, 28.01, 20.51], [808. ,
27.44, 32.93, 28.83, 22.08, 808. , 27.44, 32.93, 28.83, 22.08], [809. ,
28.63, 34.35, 31.03, 22.68, 809. , 28.63, 34.35, 31.03, 22.68], [810. ,
30.35, 36.42, 31.38, 23.1 , 810. , 30.35, 36.42, 31.38, 23.1 ]])
```

Input:

```
#2.2 Vertical Stacking
resultarray=np.vstack((d1,d2))
resultarray
```

Output:

```
array([[ nan,  nan,  nan,  nan,  nan], [801. , 28.48, 34.18, 30.56, 22.23],
[802. , 28.1 , 33.72, 30.68, 22.82], [803. , 26.16, 31.39, 28.2 ,
22.53], [804. , 26.16, 31.39, 28.78, 20.93], [805. , 26.1 , 31.32,
28.22, 20.82], [806. , 25.45, 30.54, 27.73, 21.05], [807. , 26.16,
31.39, 28.01, 20.51], [808. , 27.44, 32.93, 28.83, 22.08], [809. ,
28.63, 34.35, 31.03, 22.68], [810. , 30.35, 36.42, 31.38, 23.1 ], [
nan,  nan,  nan,  nan,  nan], [801. , 28.48, 34.18, 30.56, 22.23], [802. ,
28.1 , 33.72, 30.68, 22.82], [803. , 26.16, 31.39, 28.2 , 22.53], [804.
, 26.16, 31.39, 28.78, 20.93], [805. , 26.1 , 31.32, 28.22, 20.82],
[806. , 25.45, 30.54, 27.73, 21.05], [807. , 26.16, 31.39, 28.01,
20.51], [808. , 27.44, 32.93, 28.83, 22.08], [809. , 28.63, 34.35,
31.03, 22.68], [810. , 30.35, 36.42, 31.38, 23.1 ]])
```

Input:

```
#3.1 Range
nparray=np.arange(0,12,1).reshape(3,4)
nparray
```

Output:

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

Input:

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

Output:

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

Input:

```
#3.3 Empty Array
nparray=np.empty((3,3),int)
nparray
```

Input:

```
#3.4 Emply like some other array
nparray=np.empty_like(d1)
nparray
```

Output:

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

Input:

```
#3.5 Index Matrix
nparray=np.identity(3)
nparray
```

Output:

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

Input:

```
#4.1 Arithmatic operation
array1=np.array([1,2,3,4,5])
array2=np.array([11,12,13,14,15])
print(d1)
print(d2)
```

Output:

```
[[   nan    nan    nan    nan    nan]
 [801.    28.48  34.18  30.56  22.23]]
```

```
 [802.    28.1   33.72  30.68  22.82]
 [803.    26.16  31.39  28.2   22.53]
 [804.    26.16  31.39  28.78  20.93]
 [805.    26.1   31.32  28.22  20.82]
 [806.    25.45  30.54  27.73  21.05]
 [807.    26.16  31.39  28.01  20.51]
 [808.    27.44  32.93  28.83  22.08]
 [809.    28.63  34.35  31.03  22.68]
 [810.    30.35  36.42  31.38  23.1 ]]
[[   nan    nan    nan    nan    nan]
 [801.    28.48  34.18  30.56  22.23]
 [802.    28.1   33.72  30.68  22.82]
 [803.    26.16  31.39  28.2   22.53]
 [804.    26.16  31.39  28.78  20.93]
 [805.    26.1   31.32  28.22  20.82]
 [806.    25.45  30.54  27.73  21.05]
 [807.    26.16  31.39  28.01  20.51]
 [808.    27.44  32.93  28.83  22.08]
 [809.    28.63  34.35  31.03  22.68]
 [810.    30.35  36.42  31.38  23.1 ]]
```

Input:

```
# Addition
print(np.add(d1,d2))
# Subtraction
print(np.subtract(d1,d2))
# Multiplication
print(np.multiply(d1,d2))
# Division
print(np.divide(d1,d2))
```

Output:

```
[[    nan     nan     nan     nan     nan]
 [1602.    56.96   68.36   61.12   44.46]
 [1604.    56.2    67.44   61.36   45.64]
 [1606.    52.32   62.78   56.4    45.06]
 [1608.    52.32   62.78   57.56   41.86]
 [1610.    52.2    62.64   56.44   41.64]
 [1612.    50.9    61.08   55.46   42.1 ]
 [1614.    52.32   62.78   56.02   41.02]
 [1616.    54.88   65.86   57.66   44.16]
 [1618.    57.26   68.7    62.06   45.36]
 [1620.    60.7    72.84   62.76   46.2 ]]
[[nan nan nan nan nan]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]]
```

```
[[           nan            nan            nan            nan
nan]
 [6.4160100e+05 8.1111040e+02 1.1682724e+03 9.3391360e+02
4.9417290e+02]
 [6.4320400e+05 7.8961000e+02 1.1370384e+03 9.4126240e+02
5.2075240e+02]
 [6.4480900e+05 6.8434560e+02 9.8533210e+02 7.9524000e+02
5.0760090e+02]
 [6.4641600e+05 6.8434560e+02 9.8533210e+02 8.2828840e+02
4.3806490e+02]
 [6.4802500e+05 6.8121000e+02 9.8094240e+02 7.9636840e+02
4.3347240e+02]
 [6.4963600e+05 6.4770250e+02 9.3269160e+02 7.6895290e+02
4.4310250e+02]
 [6.5124900e+05 6.8434560e+02 9.8533210e+02 7.8456010e+02
4.2066010e+02]
 [6.5286400e+05 7.5295360e+02 1.0843849e+03 8.3116890e+02
4.8752640e+02]
 [6.5448100e+05 8.1967690e+02 1.1799225e+03 9.6286090e+02
5.1438240e+02]
 [6.5610000e+05 9.2112250e+02 1.3264164e+03 9.8470440e+02
5.3361000e+02]]
[[nan nan nan nan nan]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]]
```

Input:

```
# Standard Deviation
print(np.std(d1))
#Minimum
print(np.min(d1))
#Summation
print(np.sum(d1))
#Median
print(np.median(d1))
#Mean
print(np.mean(d1))
#Mode
from scipy import stats
print("Most Frequent element=",stats.mode(d1)[0])
print("Number of Occarances=",stats.mode(d1)[1])
# Variance
print(np.var(d1))
```

Output:

nan nan nan nan nan Most Frequent element= [[801. 39.24 25.63 26.16 25.21]] Number of Occarances= [[1 1 1 1 1]]