NAME: SAMYAK SONAWANE

ROLL NO:-763

BATCH :-G3

PRN:-202201090097

ASSIGNMENT 3

Import numpy as np dl= np.genfromtxt("/content/sample_data/testmarks1.csv",delimiter=',') print(dl)

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]]
```

EDS=dl[1:,1] print(EDS) print(type(EDS)) print(max(EDS))

OUTPUT:

```
[43.05 43.47 42.24 39.24 40.9 39.47 41.68 42.19 44.75 46.95]
<class 'numpy.ndarray'>
46.95
```

Import numpy as np d2= np.genfromtxt("/content/sample_data/testmarks2.csv",delimiter=',') 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]]

[ ]

Print(dl) print(d2) result=dl-d2

Print("\nUsing Operator:\n",resultarray) result=np.subtract(dl,d2)

Print("\nUsing Numpy Function:\n",result)

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]]

[[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]]

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.]

[0. 0. 0. 0. 0.]]

Using Numpy Function:

[[nan nan nan nan nan]

[0. 14.57 -6.39 -1.86 5.56]

[0. 15.37 -5.2 -1.7 5.07]

[0. 16.08 -3.23 -0.04 3.1]

[0. 13.08 -5.23 -2.62 5.23]

```
[ 0.          14.8 -5.29 -0.95 4.83]
[ 0.          14.02 -4.23 -1.42 4.16]
[ 0.          15.52 -5.76 -0.22 4.95]
[ 0.          14.75 -5.32 -0.7 4.13]
[ 0.          16.12 -6. -1.2 5.53]
[ 0.          16.6 -7.54 -0.08 5.43]]
```

```
Resultarray=dl+d2 print("\nUsing Numpy Function:\n",resultarray) resultarray=np.add(dl,d2)
print("\nUsingOperator:\n",resultarray)
```

OUTPUT:

Using Numpy Function:

```
[[ nan    nan    nan    nan    nan]
[1602.    71.53 61.97 59.26 50.02]
[1604.    71.57 62.24 59.66 50.71]
[1606.    68.4 59.55 56.36 48.16]
[1608.    65.4 57.55 54.94 47.09]
[1610.    67.    57.35 55.49 46.47]
```

```
[1612.    64.92 56.85 54.04 46.26]

[1614.    67.84 57.02 55.8 45.97]

[1616.    69.63 60.54 56.96 48.29]

[1618.    73.38 62.7 60.86 50.89]

[1620.    77.3 65.3 62.68 51.63]]
```

Using Operator:

```
[[ nan    nan    nan    nan    nan]

[1602.    71.53 61.97 59.26 50.02]

[1604.    71.57 62.24 59.66 50.71]

[1606.    68.4 59.55 56.36 48.16]

[1608.    65.4 57.55 54.94 47.09]

[1610.    67.    57.35 55.49 46.47]

[1612.    64.92 56.85 54.04 46.26]

[1614.    67.84 57.02 55.8 45.97]

[1616.    69.63 60.54 56.96 48.29]
```

[1618.   73.38 62.7 60.86 50.89]

[1620.   77.3 65.3 62.68 51.63]]

Resultarray=dl%d2

Print("\nUsing Operator:\n",resultarray)

Resultarray=np.mod(dl,d2)

Print("\nUsing Numpy Function:\n",resultarray)

OUTPUT:

Using Operator:

[[nan nan nan nan nan]

[0.   14.57 27.79 28.7 5.56]

[0.   15.37 28.52 28.98 5.07]

[0.   16.08 28.16 28.16 3.1]

[0.   13.08 26.16 26.16 5.23]

[0.   14.8 26.03 27.27 4.83]

[0.   14.02 26.31 26.31 4.16]

[0. 15.52 25.63 27.79 4.95]

[0. 14.75 27.61 28.13 4.13]

[0. 16.12 28.35 29.83 5.53]

[0. 16.6 28.88 31.3 5.43]]

Using Numpy Function:

[[nan nan nan nan nan]

[0. 14.57 27.79 28.7 5.56]

[0. 15.37 28.52 28.98 5.07]

[0. 16.08 28.16 28.16 3.1]

[0. 13.08 26.16 26.16 5.23]

[0. 14.8 26.03 27.27 4.83]

[0. 14.02 26.31 26.31 4.16]

[0. 15.52 25.63 27.79 4.95]

[0. 14.75 27.61 28.13 4.13]

[0. 16.12 28.35 29.83 5.53]

[0.   16.6 28.88 31.3 5.43]]

Resultarray=dl*d2

Print(“\nUsing Operator:\n”,resultarray) resultarray=np.multiply(dl,d2)

Print(“\nUsing Numpy Function:\n”,resultarray)

OUTPUT:

Using Operator:

[[nan nan nan nan nan]

[6.4160100e+05 1.2260640e+03 9.4986220e+02 8.7707200e+02 6.1777170e+02]

[6.4320400e+05 1.2215070e+03 9.6169440e+02 8.8910640e+02 6.3644980e+02]

[6.4480900e+05 1.1049984e+03 8.8394240e+02 7.9411200e+02 5.7744390e+02]

[6.4641600e+05 1.0265184e+03 8.2116240e+02 7.5288480e+02 5.4752880e+02]

[6.4802500e+05 1.0674900e+03 8.1525960e+02 7.6955940e+02 5.3403300e+02]

[6.4963600e+05 1.0045115e+03 8.0350740e+02 7.2957630e+02 5.3067050e+02]

[6.5124900e+05 1.0903488e+03 8.0452570e+02 7.7839790e+02 5.2218460e+02]

[6.5286400e+05 1.1576936e+03 9.0919730e+02 8.1098790e+02 5.7871680e+02] [6.5448100e+05 1.2811925e+03 9.7382250e+02 9.2562490e+02 6.3980280e+02]

[6.5610000e+05 1.4249325e+03 1.0518096e+03 9.8219400e+02 6.5904300e+02]]

Using Numpy Function:

[[nan nan nan nan nan]

[6.4160100e+05 1.2260640e+03 9.4986220e+02 8.7707200e+02 6.1777170e+02]

[6.4320400e+05 1.2215070e+03 9.6169440e+02 8.8910640e+02 6.3644980e+02]

[6.4480900e+05 1.1049984e+03 8.8394240e+02 7.9411200e+02 5.7744390e+02]

[6.4641600e+05 1.0265184e+03 8.2116240e+02 7.5288480e+02 5.4752880e+02]

[6.4802500e+05 1.0674900e+03 8.1525960e+02 7.6955940e+02 5.3403300e+02]

[6.4963600e+05 1.0045115e+03 8.0350740e+02 7.2957630e+02 5.3067050e+02]

[6.5124900e+05 1.0903488e+03 8.0452570e+02 7.7839790e+02 5.2218460e+02]

[6.5286400e+05 1.1576936e+03 9.0919730e+02 8.1098790e+02 5.7871680e+02]

[6.5448100e+05 1.2811925e+03 9.7382250e+02 9.2562490e+02 6.3980280e+02]

[6.5610000e+05 1.4249325e+03 1.0518096e+03 9.8219400e+02 6.5904300e+02]]

Resultarray=dl/d2

Print("\nUsing Operator:\n",resultarray) resultarray=np.divide(dl,d2) print("\nUsing Numpy Function:\n",resultarray)

OUTPUT:

Using Operator:

```
[[        nan        nan        nan        nan        nan]
 [1.         1.51158708 0.81304857 0.93913613 1.25011246]
 [1.         1.54697509 0.84578885 0.94458931 1.22217353]
 [1.         1.6146789  0.89710099 0.99858156 1.13759432]
 [1.         1.5        0.83338643 0.90896456 1.24988055]
 [1.         1.56704981 0.83109834 0.96633593 1.23198847] [1.         1.55088409 0.86149312 0.94879192 1.1976247 ] [1.         1.59327217 0.81650207 0.99214566 1.24134569] [1.         1.53753644 0.83844519 0.97571974 1.1870471 ] [1.         1.56304576 0.82532751 0.96132775 1.24382716]
 [1.         1.54695222 0.7929709  0.99745061 1.23506494]]
```

Using Numpy Function:

```
[[        nan        nan        nan        nan        nan]
 [1.         1.51158708 0.81304857 0.93913613 1.25011246]
```

[1.   1.54697509 0.84578885 0.94458931 1.22217353]

[1.   1.6146789 0.89710099 0.99858156 1.13759432]

[1.   1.5   0.83338643 0.90896456 1.24988055]

[1.   1.56704981 0.83109834 0.96633593 1.23198847] [1.   1.55088409 0.86149312 0.94879192 1.1976247] [1.   1.59327217 0.81650207 0.99214566 1.24134569] [1.   1.53753644 0.83844519 0.97571974 1.1870471] [1.   1.56304576 0.82532751 0.96132775 1.24382716]

[1.   1.54695222 0.7929709 0.99745061 1.23506494]]

HORIZONTAL STACKING

Resultarray=np.hstack((dl,d2))   resultarray

OUTPUT:

Array([[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan], [801. , 43.05, 27.79, 28.7 , 27.79, 801. , 28.48, 34.18, 30.56, 22.23], [802. , 43.47, 28.52, 28.98, 27.89, 802. , 28.1 , 33.72, 30.68, 22.82], [803. , 42.24,

28.16, 28.16, 25.63, 803. , 26.16, 31.39, 28.2 , 22.53], [804. , 39.24,

26.16, 26.16, 26.16, 804. , 26.16, 31.39, 28.78, 20.93], [805. , 40.9 ,

26.03, 27.27, 25.65, 805. , 26.1 , 31.32, 28.22, 20.82], [806. , 39.47, 26.31, 26.31, 25.21, 806. , 25.45, 30.54, 27.73, 21.05], [807. , 41.68, 25.63, 27.79, 25.46, 807. , 26.16, 31.39, 28.01, 20.51], [808. , 42.19,

27.61, 28.13, 26.21, 808. , 27.44, 32.93, 28.83, 22.08], [809. , 44.75, 28.35, 29.83, 28.21, 809. , 28.63, 34.35, 31.03, 22.68], [810. , 46.95,

28.88, 31.3 , 28.53, 810. , 30.35, 36.42, 31.38, 23.1]])

VERTICAL STACKING

```
Resultarray=np.vstack((dl,d2))
```

Resultarray

OUTPUT:

Array([[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], [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]])

RANGE

[ ]

Arr1=np.arange(800,810,1)

Print(arr1)

OUTPUT:

[800 801 802 803 804 805 806 807 808 809]

EMPTY LIKE SOME OTHER ARRAY

[ ]

Nparray=np.empty_like(dl)

Nparray

OUTPUT:

Array([[nan, nan, nan, nan, nan], [1. , 1.51158708, 0.81304857,

,

, [1. ,

0.93913613, 1.25011246], [1. , 1.54697509, 0.84578885, 0.94458931

1.22217353], [1. , 1.6146789 , 0.89710099, 0.99858156, 1.13759432]

1.5 , 0.83338643, 0.90896456, 1.24988055], [1. , 1.56704981, 0.83109834,

0.96633593, 1.23198847], [1. , 1.55088409, 0.86149312, 0.94879192,

1.1976247], [1. , 1.59327217, 0.81650207, 0.99214566, 1.24134569], [1. ,

1.53753644, 0.83844519, 0.97571974, 1.1870471], [1. , 1.56304576,

0.82532751, 0.96132775, 1.24382716], [1. , 1.54695222, 0.7929709 ,

0.99745061, 1.23506494]])

ARITHMETIC OPERATIONS

Addition print(np.add(dl,d2)) # Subtraction print(np.subtract(dl,d2))

Multiplication print(np.multiply(dl,d2))

Division print(np.divide(dl,d2))

OUTPUT:

```
[[ nan    nan    nan    nan    nan]

[1602.   71.53  61.97  59.26  50.02]

[1604.   71.57  62.24  59.66  50.71]

[1606.   68.4   59.55  56.36  48.16]

[1608.   65.4   57.55  54.94  47.09]

[1610.   67.    57.35  55.49  46.47]

[1612.   64.92  56.85  54.04  46.26]

[1614.   67.84  57.02  55.8   45.97]

[1616.   69.63  60.54  56.96  48.29]

[1618.   73.38  62.7   60.86  50.89]

[1620.   77.3   65.3   62.68  51.63]]

[[ nan   nan nan nan nan]

[ 0.     14.57 -6.39 -1.86 5.56]

[ 0.     15.37 -5.2 -1.7 5.07]

[ 0.     16.08 -3.23 -0.04 3.1 ]
```

[0.   13.08 -5.23 -2.62 5.23]

[0.   14.8 -5.29 -0.95 4.83]

[0.   14.02 -4.23 -1.42 4.16]

[0.   15.52 -5.76 -0.22 4.95]

[0.   14.75 -5.32 -0.7 4.13]

[0.   16.12 -6. -1.2 5.53]

[0.   16.6 -7.54 -0.08 5.43]]

[[nan   nan   nan   nan   nan]

[6.4160100e+05 1.2260640e+03 9.4986220e+02 8.7707200e+02 6.1777170e+02]

[6.4320400e+05 1.2215070e+03 9.6169440e+02 8.8910640e+02 6.3644980e+02]

[6.4480900e+05 1.1049984e+03 8.8394240e+02 7.9411200e+02 5.7744390e+02]

[6.4641600e+05 1.0265184e+03 8.2116240e+02 7.5288480e+02 5.4752880e+02]

[6.4802500e+05 1.0674900e+03 8.1525960e+02 7.6955940e+02 5.3403300e+02]

[6.4963600e+05 1.0045115e+03 8.0350740e+02 7.2957630e+02 5.3067050e+02]

[6.5124900e+05 1.0903488e+03 8.0452570e+02 7.7839790e+02 5.2218460e+02]

[6.5286400e+05 1.1576936e+03 9.0919730e+02 8.1098790e+02 5.7871680e+02]

[6.5448100e+05 1.2811925e+03 9.7382250e+02 9.2562490e+02 6.3980280e+02]

```
[6.5610000e+05 1.4249325e+03 1.0518096e+03 9.8219400e+02 6.5904300e+02]]

[[        nan        nan        nan        nan        nan]

 [1.         1.51158708 0.81304857 0.93913613 1.25011246]

 [1.         1.54697509 0.84578885 0.94458931 1.22217353]

 [1.         1.6146789 0.89710099 0.99858156 1.13759432]

 [1.         1.5        0.83338643 0.90896456 1.24988055]

[1.         1.56704981 0.83109834 0.96633593 1.23198847] [1.         1.55088409 0.86149312 0.94879192 1.1976247 ] [1.
 1.59327217 0.81650207 0.99214566 1.24134569] [1.         1.53753644 0.83844519 0.97571974 1.1870471 ] [1.
 1.56304576 0.82532751 0.96132775 1.24382716]

 [1.         1.54695222 0.7929709 0.99745061 1.23506494]]
```

STATISTICAL OPERATIONS

```
# Standard Deviation print(np.std(dl))

#Minimum print(np.min(dl)) #Summation print(np.sum(dl))

#Median print(np.median(dl))

#Mean
```

Print(np.mean(dl))

#Mode from scipy import stats

Print("Most Frequent element=",stats.mode(dl)[0]) print("Number of Occarances=",stats.mode(dl)[1])

Variance

Print(np.var(dl))

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]] nan

<ipython-input-56-da9861487e77>:13: 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(dl)[0])

<ipython-input-56-da9861487e77>: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("Number of Occarances=",stats.mode(dl)[1])