Name: Modi Nirmeet

Sub: MLP

Roll no:B-26

// print hello world

Input:

```
print('hello world')
```

output:

```
hello world
```

// use NumPy

Input:

```
import numpy as np
```

// Create Array , Save in variable, print Array and print Array size, shape, data type, itemsize

Input:

```
a = np.array([1,2,3])
print(a)
print('array a:{}'.format(a))
print('ndim:',a.ndim)
print('size of array:',a.size)
print('shape of array:',a.shape)
print('data type:',a.dtype)
print('itemsize:',a.itemsize)
```

output:

```
[1 2 3]
array a:[1 2 3]
ndim: 1
size of array: 3
shape of array: (3,)
data type: int64
itemsize: 8
```

// create another array (2-dimensional NumPy array)

Input:

```
b = np.array([[1,2,3],[1,2,3]])
print(b)
print('ndim:',b.ndim)
print('size of array:',b.size)
print('shape of array:',b.shape)
print('data type:',b.dtype)
print('itemsize:',b.itemsize)
```

output:

```
[[1 2 3] [1 2 3]]
ndim: 2
size of array: 6
shape of array: (2, 3)
data type: int64
```

itemsize: 8

// create 2 array and apply addition, subtract, divide, power, multiply, mod, remainder, absolute

Input:

```
c = np.array([5,7,6])
print('add', np.add(a,c))
print('sub', np.subtract(a,c))
print('div',np.divide(a,c))
print('pow',np.power(a,c))
print('multiple',np.multiply(a,c))
print('mod',np.mod(a,c))
print('remainder',np.remainder(a,c))
print('absolute',np.absolute(a,c))
```

output:

```
add [6 9 9]
sub [-4 -5 -3]
div [0.2        0.28571429 0.5       ]
pow [  1 128 729]
multiple [ 5 14 18]
mod [1 2 3]
remainder [1 2 3]
absolute [1 2 3]
```

// Create Array , Save in variable, print Array and print Array size, shape, data type, itemsize (with string data)

Input:

```
a = np.array(['sandip','naman','digu'])
print(a)
print('ndim:',a.ndim)
print('size of array:',a.size)
print('shape of array:',a.shape)
print('data type:',a.dtype)
print('itemsize:',a.itemsize)
```

output:

```
['sandip' 'naman' 'digu']
ndim: 1
size of array: 3
shape of array: (3,)
data type: <U6
itemsize: 24
```

// print 1 to 5 with interval of 0.5

Input:

```
ar = np.arange(1,5,0.5)
print(ar)
```

output:

```
[1.  1.5 2.  2.5 3.  3.5 4.  4.5]
```

// create array with 2 row and 2 column (random data)

Input:

```
a = np.random.random((2,2))
print(a))
```

output:

```
[[0.90707307 0.25997148]
[0.0225364  0.7046009 ]]
```

// create Array with default value zeros, ones, empty and random (between 1 to 2 with 7 data)

Input:

```
z=np.zeros((4,4))
o=np.ones((3,3))
e=np.empty((1,1))
ls=np.linspace(1,2,7)
```

output:

```
[[0. 0. 0. 0.][0. 0. 0. 0.][0. 0. 0. 0.][0. 0. 0. 0.]]
[[1. 1. 1.][1. 1. 1.][1. 1. 1.]]
[[5.e-324]]
[1.      1.16666667  1.33333333  1.5       1.66666667  1.83333333  2. ]
```

// create array (2-dimensional NumPy array)

Input:

```
ar=np.array([[10,20,30],[40,50,60]])
print(ar)
```

output:

```
[[10 20 30] [40 50 60]]
```

// array from the first row onwards and from the second column onwards

Input:

```
sliced=ar[:1:,1:]
print(sliced)
```

output:

```
[[20 30]]
```

// create array from ar with ‘1 row 2 column’ and ‘2 row 2 column’

Input:

```
indexed=ar[[0,1],[1,1]]
print(indexed)
```

output:

```
[20 50]
```

// sum

Input:

```
print(ar.sum())
```

output:

```
210
```

// sqrt

Input:

```
print(np.sqrt(ar))
```

output:

```
[[3.16227766 4.47213595 5.47722558]
 [6.32455532 7.07106781 7.74596669]]
```

// add elemenmts in aray

Input:

```
print(ar+10)
```

output:

```
[[20 30 40] [50 60 70]]
```

// Compute the dot product of the two matrices

Input:

```
matrix1=np.array([[1,2,3],[4,5,6],[7,8,9]])
matrix2=np.array([[4,5,6],[1,2,3],[7,8,9]])

matrix3=np.dot(matrix1,matrix2)

print(matrix3)
```

output:

```
[[ 27  33  39][ 63  78  93][ 99 123 147]]
```

// transpose

Input:

```
matrix1=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(format(matrix1.T))
```

output:

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

// boolean

Input:

```
bol=np.array([[True,True],[False,False]])
print(np.all(bol)) #all array (false present = false)
print(np.all(bol,axis=0)) #column (false present = false)
print(np.all(bol,axis=1)) #row (false absent = true, else false)
print(np.any(bol))          (True present = true)
print(np.any(bol,axis=0)) #column (True present = true)
print(np.any(bol,axis=1)) #row (True absent = false, else true)
```

output:

```
False
[False False]
[ True False]
True
[ True  True]
[ True False]
```

// arange

Input:

```
ar=np.arange(8)
print(ar)
```

output:

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

// reshape

Input:

```
reshaped=ar.reshape(2,4)
print(reshaped)
```

output:

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

// merge verticxaly vstack

Input:

```
matrix1=np.array([[1,2,3],[4,5,6],[7,8,9]])
matrix2=np.array([[4,5,6],[1,2,3],[7,8,9]])
matrix3=np.array([[ 27  33  39][ 63  78  93][ 99 123 147]])

vstackAr=np.vstack((matrix1,matrix2,matrix3))
print(vstackAr)
```

output:

```
[[  1   2   3] [  4   5   6][  7   8   9]
 [  4   5   6][  1   2   3][  7   8   9]
 [ 27  33  39][ 63  78  93][ 99 123 147]]
```

// merge horizon hstack

Input:

```
matrix1=np.array([[1,2,3],[4,5,6],[7,8,9]])
matrix2=np.array([[4,5,6],[1,2,3],[7,8,9]])
matrix3=np.array([[ 27  33  39][ 63  78  93][ 99 123 147]])

hstackAr=np.hstack((matrix1,matrix2,matrix3))
print(hstackAr)
```

output:

```
[[  1   2   3   4   5   6  27  33  39]
 [  4   5   6   1   2   3  63  78  93]
 [  7   8   9   7   8   9  99 123 147]]
```

// hsplit

Input:

```
hplited=np.hsplit(hstackAr, 3)
print(hplited)
```

output:

```
[array([[1, 2, 3], [4, 5, 6],[7, 8, 9]]),
 array([[4, 5, 6], [1, 2, 3],[7, 8, 9]]),
 array([[ 27,  33,  39], [ 63,  78,  93], [ 99, 123, 147]])]
```

// vsplit

Input:

```
vplited=np.vsplit(vstackAr, 3)
print(vplited)
```

output:

```
[array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]),
 array([[4, 5, 6], [1, 2, 3], [7, 8, 9]]),
 array([[ 27,  33,  39], [ 63,  78,  93], [ 99, 123, 147]])]
```

// mathamatical functions

Input:

```
ar=np.array([1,2,3])
print("sqrt",np.sqrt(ar)) # Calculate the square root of each element
print("exp",np.exp(ar))  # Calculate the exponential of each element (e^x)
print("sin",np.sin(ar))
print("cos",np.cos(ar))
print("log",np.log(ar))
print("sum",np.sum(ar))
print("std",np.std(ar))  # Calculate the standard deviation of the elements in the array
```

output:

```
sqrt [1.         1.41421356 1.73205081]
exp [ 2.71828183  7.3890561  20.08553692]
sin [0.84147098 0.90929743 0.14112001]
cos [ 0.54030231 -0.41614684 -0.9899925 ]
log [0.         0.69314718 1.09861229]
sum 6
std 0.816496580927726
```

// create 20 rendom number between 0 to 1

Input:

```
print(np.random.random(20))
```

output:

```
[0.12345678 0.45678901 0.98765432 0.23456789 0.3456789  0.67890123
 0.89012345 0.56789012 0.09876543 0.7654321  0.43210987 0.21098765
 0.54321098 0.87654321 0.78901234 0.90123457 0.654321   0.32109876
0.10987654 0.87654321]
```

// rand (Generate a 3x4 array of random numbers from a uniform distribution over [0, 1))

Input:

```
print(np.random.rand(3,4))
```

output:

```
[[0.12345678 0.45678901 0.98765432 0.23456789]
 [0.3456789  0.67890123 0.89012345 0.56789012]
 [0.09876543 0.7654321  0.43210987 0.21098765]]
```

// randint (Generate an array of 20 random integers between 0 and 100)

Input:

```
print(np.random.randint(0,100,20))
```

output:

```
[87 28 12 45 71 52 35 96 59 19 88 63  3 71 53 87 26 94 64 39]
```

// permutation (Generate a random permutation of integers from 0 to 19)

Input:

```
print(np.random.permutation(np.arange(20)))
```

output:

```
[18  6  4 16 13 11 14 19  7  3  0  5  2 12  8 10  1 15 17  9]
```