

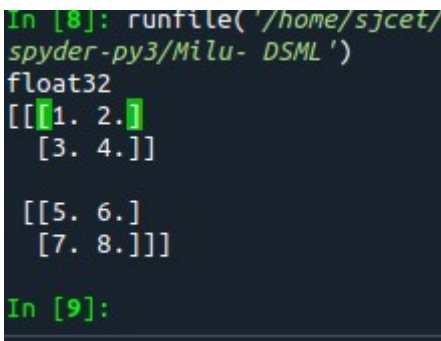
CYCLE 2

1. Create a three dimensional array specifying float data type and print it.

CODE

```
import numpy as np
array_3D = np.array([[[1,2],[3,4]],[[5,6],[7,8]]], dtype='f')
print(array_3D.dtype)
print(array_3D)
```

OUTPUT



```
In [8]: runfile('/home/sjcet/
spyder-py3/Milu- DSML')
float32
[[[1. 2.]
  [3. 4.]

  [5. 6.]
  [7. 8.]]]
In [9]:
```

2. Create a 2 dimensional array (2X3) with elements belonging to complex data type and print it. Also display

- the no: of rows and columns
- dimension of an array
- reshape the same array to 3X2

CODE

```
import numpy as np

array_2d=np.array([[complex(1,2),complex(2,3),complex(3,4)],
[complex(4,5),complex(5,6),complex(6,7)]])
print(array_2d)

print("the no. of rows and columns :",array_2d.shape)
print("dimension of an array",array_2d.ndim)

print("reshape the same array to 3x2",array_2d.reshape(3,2))
```

OUTPUT

```
In [1]: runfile('/home/sjcet/Downloads/Cycle 2/p2',
sjcet/Downloads/Cycle 2')
[[1.+2.j 2.+3.j 3.+4.j]
 [4.+5.j 5.+6.j 6.+7.j]]
the no. of rows and columns : (2, 3)
dimension of an array 2
reshape the same array to 3x2 [[1.+2.j 2.+3.j]
 [3.+4.j 4.+5.j]
 [5.+6.j 6.+7.j]]
```

3. Familiarize with the functions to create

- a) an uninitialized array
- b) array with all elements as 1,
- c) all elements as 0

CODE

```
import numpy as np

print("an uninitialized array :")
print(np.empty([2,2]))
print("array with all elements as 1 :")
print(np.full([2,2],1))
print("all elements as 0 :")
print(np.zeros([2,2]))
```

OUTPUT

```
In [2]: runfile('/home/sjcet/Download
an uninitialized array :
[[4.63600470e-310 4.63592420e-310]
 [3.16202013e-322 6.61175426e+265]]
array with all elements as 1 :
[[1 1]
 [1 1]]
all elements as 0 :
[[0. 0.]
 [0. 0.]]
In [3]:
```

4. Create an one dimensional array using arange function containing 10 elements.

Display

- First 4 elements
- Last 6 elements
- Elements from index 2 to 7

CODE

```
import numpy as np
array_1d=np.array([1,2,3,4,5,6,7,8,9,10])
print("First 4 elements",array_1d[:4])
print("Last 4 elements", array_1d[4:])
print("Elements from index 2 to 7",array_1d[2:7])
```

OUTPUT

```
In [3]: runfile('/home/sjctet/Downloads/Cycle 2/p4', wdir=
sjctet/Downloads/Cycle 2')
First 4 elements [1 2 3 4]
Last 4 elements [ 5  6  7  8  9 10]
Elements from index 2 to 7 [3 4 5 6 7]
```

5. Create an 1D array with arange containing first 15 even numbers as elements

- Elements from index 2 to 8 with step 2(also demonstrate the same using slice function)
- Last 3 elements of the array using negative index
- Alternate elements of the array
- Display the last 3 alternate elements

CODE

```
import numpy as np
array_1d=np.array([0,2,4,6,8,10,12,14,16,18,20,22,24,26,28])
print("Elements from index 2 to 8 with step 2",array_1d[2:8:2])
print("Last 3 elements of the array using Negative number",array_1d[-3:-1])
print("Alternate elements of the array",array_1d[::2])
print("Display the last 3 alternate elements",array_1d[-3:-1:2])
```

OUTPUT

```
In [4]: runfile('/home/sjcet/Downloads/Cycle 2/p5', wdir='/home/
sjcet/Downloads/Cycle 2')
Elements from index 2 to 8 with step 2 [ 4  8 12]
Last 3 elements of the array using Negative number [24 26]
Alternate elements of the array [ 0  4  8 12 16 20 24 28]
Display the last 3 alternate elements [24]
```

6. Create a 2 Dimensional array with 4 rows and 4 columns.
 - a. Display all elements excluding the first row
 - b. Display all elements excluding the last column
 - c. Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row
 - d. Display the elements of 2 nd and 3 rd column
 - e. Display 2 nd and 3 rd element of 1 st row
 - f. Display the elements from indices 4 to 10 in descending order(use -values)

CODE

```
import numpy as np
array_2d=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]])
print(array_2d)
print("Display all elements excluding the first row")
print(array_2d[1:4,:])
print("Display all elements excluding the last column")
print(array_2d[:,0:3])
print("Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row")
print(array_2d[1:3,1:3])
print("Display the elements of 2 nd and 3 rd column")
print(array_2d[:,1:3])
print("Display 2 nd and 3 rd element of 1 st row")
print(array_2d[0,1:3])
```

OUTPUT

```
In [5]: runfile('/home/sjcet/Downloads/Cycle 2/p6', wdir='/home/sjcet/Do
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
Display all elements excluding the first row
[[ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
Display all elements excluding the last column
[[ 1  2  3]
 [ 5  6  7]
 [ 9 10 11]
 [13 14 15]]
Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row
Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row
[[ 6  7]
 [10 11]]
Display the elements of 2 nd and 3 rd column
[[ 2  3]
 [ 6  7]
 [10 11]
 [14 15]]
Display 2 nd and 3 rd element of 1 st row
[2 3]
```

7. Create two 2D arrays using array object and
 - a. Add the 2 matrices and print it
 - b. Subtract 2 matrices
 - c. Multiply the individual elements of matrix
 - d. Divide the elements of the matrices
 - e. Perform matrix multiplication
 - f. Display transpose of the matrix
 - g. Sum of diagonal elements of a matrix

CODE

```
import numpy as np
M1 = np.array([[3, 6], [14, 21]])
M2 = np.array([[9, 27], [11, 22]])
M3 = M1 + M2
print("Matrix addition")
print(M3)
M3 = M1 - M2
print("Matrix Substract")
```

```

print(M3)
M3 = M1 / M2
print("Divide the elements of the matrices")
print(M3)
M3 = M1 * M2
print("Multiply the individual elements of matrix")
print(M3)
M1 = np.array([[3, 6], [5, -10]])
M2 = np.array([[9, -18], [11, 22]])
M3 = M1.dot(M2)
print("matrix multiplication")
print(M3)
M1 = np.array([[3, 6, 9], [5, -10, 15], [4,8,12]])
M2 = M1.transpose()
print("Transpose of the matrix")
print(M2)
M1 = np.array([[3, 6, 9], [5, -10, 15], [4,8,12]])
print("Sum of diagonal elements of a matrix")
print(np.trace(M1))

```

OUTPUT

```

In [6]: runfile('/home/sjcet/Downloads/Cycle 2/p
Matrix addition
[[12 33]
 [25 43]]
Matrix Substract
[[ -6 -21]
 [  3  -1]]
Divide the elements of the matrices
[[0.33333333 0.22222222]
 [1.27272727 0.95454545]]
Multiply the individual elements of matrix
[[ 27 162]
 [154 462]]

```

```

matrix multiplication
[[  93  78]
 [-65 -310]]
Transpose of the matrix
[[  3  5  4]
 [  6 -10  8]
 [  9 15 12]]
Sum of diagonal elements of a matrix
5

```

8. Demonstrate the use of insert() function in 1D and 2D array

CODE

```
import numpy as np
arr1 = np.arange(10, 16)
print("1D ARRAY ")
print("The array is: ", arr1)
obj = 2
value = 40
arr = np.insert(arr1, obj, value, axis=None)
print("After inserting the new array is: ")
print(arr)
print("Shape of the new array is : ", np.shape(arr))
print("2D ARRAY ")
arr1 = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9), (50, 51, 52)])
print("The array is: ")
print(arr1)
print("The shape of the array is: ", np.shape(arr1))
a = np.insert(arr1, 1, [[50], [100], ], axis=0)
print("New array is : ")
print(a)
print("Shape of the array is: ", np.shape(a))
```

OUTPUT

```
In [7]: runfile('/home/sjcet/Downloads/Cycle 2/p8', wdir='/home/sjcet/Downloads/Cycle 2/p8')
1D ARRAY
The array is: [10 11 12 13 14 15]
After inserting the new array is:
[10 11 40 12 13 14 15]
Shape of the new array is : (7,)
2D ARRAY
The array is:
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [50 51 52]]
The shape of the array is: (4, 3)
New array is :
[[ 1  2  3]
 [50 50 50]
 [100 100 100]
 [ 4  5  6]
 [ 7  8  9]
 [50 51 52]]
Shape of the array is: (6, 3)
```

9. Demonstrate the use of diag() function in 1D and 2D array.

CODE

```
import numpy as np

array_1d=np.arange(1,6).reshape((1,5))
print("1D array\n",array_1d)
print("use of diag() function in 1D array.")
x=np.diag(array_1d)
print(x,"\n")

array_2d=np.arange(9).reshape((3,3))
print("2D array\n",array_2d)
print("use of diag() function in 2D array.")
y=np.diag(array_2d)
print(y)
```

OUTPUT

```
In [8]: runfile('/home/sjcet/Downloads/Cycle 2/p9', w
1D array
[[1 2 3 4 5]]
use of diag() function in 1D array.
[1]

2D array
[[0 1 2]
 [3 4 5]
 [6 7 8]]
use of diag() function in 2D array.
[0 4 8]
```

10. Demonstrate the use of append() function in 1D and 2D array.

CODE

```
import numpy as np

array_1d=np.array([1,2,3,4,5])
x=np.append(array_1d,6)
print("1D array",array_1d)
print("the use of append() function in 1D array",x)

array_2d=np.array([[1,2,3],[4,5,6]])
y=np.append(array_2d,np.array([[7,8,9]]),axis=0)
```



```
print("\n2D array")
print(array_2d)
print("the use of append() function in 2D array")
print(y)
```

OUTPUT

```
1D array [1 2 3 4 5]
the use of append() function in 1D array [1 2 3 4 5 6]

2D array
[[1 2 3]
 [4 5 6]]
the use of append() function in 2D array
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

11. Demonstrate the use of sum() function in 1D and 2D array.

CODE

```
import numpy as np

array_1d=np.array([1,2,3,4,5])
x=np.sum(array_1d)
print(array_1d)
print("the use of sum() function in 1D array",x)

array_2d=np.array([[1,2,3],[4,5,6],[7,8,9]])
y=np.sum(array_2d)
print("\n",array_2d)
print("the use of sum() function in 1D array",y)
```

OUTPUT

```
In [10]: runfile('/home/sjcet/Downloads/Cycle 2/p11', wdir='/
[1 2 3 4 5]
the use of sum() function in 1D array 15

[[1 2 3]
 [4 5 6]
 [7 8 9]]
the use of sum() function in 1D array 45
```