D**ATA SCIENCE & MACHINE LEARNING**

**LAB CYCLE 2**

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

**Program**

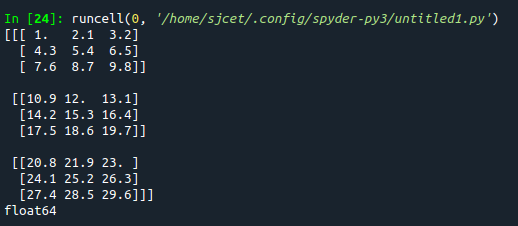
import numpy as np

array = np.arange(1,30,1.1).reshape(3,3,3)

print(array)

print(array.dtype)

**Output**



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

**a. the no: of rows and columns**

**b. dimension of an array**

**c. reshape the same array to 3X2**

**Program**

import numpy as np

array = np.arange(6).astype(complex).reshape(2,3)

print(array)

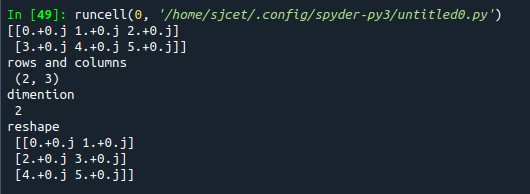
print("rows and columns\n", array.shape)

print("dimention\n", array.ndim)

arr2 = array.reshape(3,2)

print("reshape\n", arr2)

**Output**



**3. Familiarize with the functions to create**

**a) an uninitialized array**

**b) array with all elements as 1,**

**c) all elements as 0**

**Program**

import numpy as np

a = []

print(a)

#1

print(np.empty(2))

#2

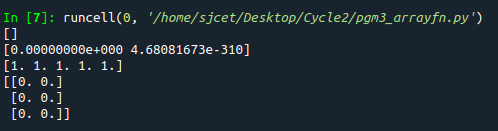
print(np.ones(5))

#3

a=(3,2)

print(np.zeros(a))

**Output**



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

**Display**

**a. First 4 elements**

**b. Last 6 elements**

**c. Elements from index 2 to 7**

**Program**

import numpy as np

print(np.arange(1, 11, 1))

# 1

print(np.arange(1, 5, 1))

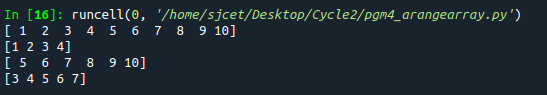
# 2

print(np.arange(5, 11, 1))

# 3

print(np.arange(3, 8, 1))

**Output**



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

**a. Elements from index 2 to 8 with step 2(also demonstrate the same**

**using slice function)**

**b. Last 3 elements of the array using negative index**

**c. Alternate elements of the array**

**d. Display the last 3 alternate elements**

**Program**

import numpy as np

array=np.arange(0,15,2)

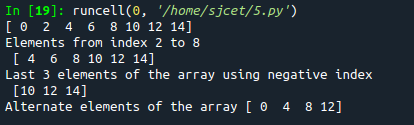
print(array)

print("Elements from index 2 to 8\n",array[2:8])

print("Last 3 elements of the array using negative index\n",array[-3:])

print("Alternate elements of the array",array[::2])

**Output**



**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)**

**Program**

import numpy as np

x = np.array([[2, 4, 6,1], [6, 8, 10,1],[1, 2, 1,1], [1, 1, 1,1]])

print(x)

#1

print("excluding first row\n",x[1:])

#2

print("excluding last column\n",x[:,:3])

#3

print("Display the elements of 1st and 2nd column in 2nd and 3rd row")

print(x[1:3,0:2])

#4

print("dispaly 2 and 3 element",x[:1,1:3])

#5

print("display 2nd and 3rd element of 1st row")

print(x[0:1,1:3])

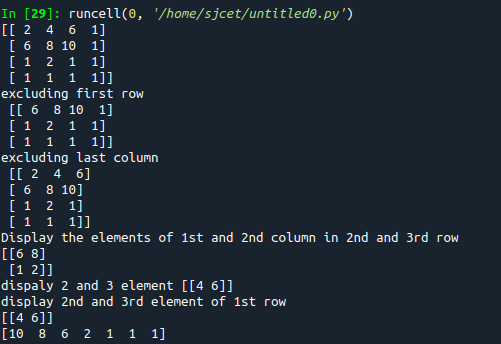
#6

arr=np.array([1,6,8,10,1,1,2])

sarr=np.sort(arr)[::-1]

print(sarr)

**Output**



**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**

**Program**

import numpy as np

array = np.arange(1,5,1).reshape(2,2)

array2 = np.arange(6,10,1).reshape(2,2)

print ("\n", array)

print ("\n", array2)

print ("\nsum of two 2darrays is: \n", array + array2)

print ("\n2darrays subtracted: \n", array - array2)

print ("\nproduct of individual elements: \n", array \* array2)

print ("\n2darrays divided: \n", array / array2)

matrixprod = np.matmul(array, array2)

print("\nproduct of two matrices\n", matrixprod)

transpose = np.transpose(array)

print("\ntranspose of 1st array\n", transpose)

transpose2 = np.transpose(array2)

print("\ntranspose of 2nd array\n", transpose2)

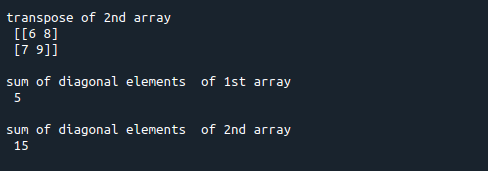
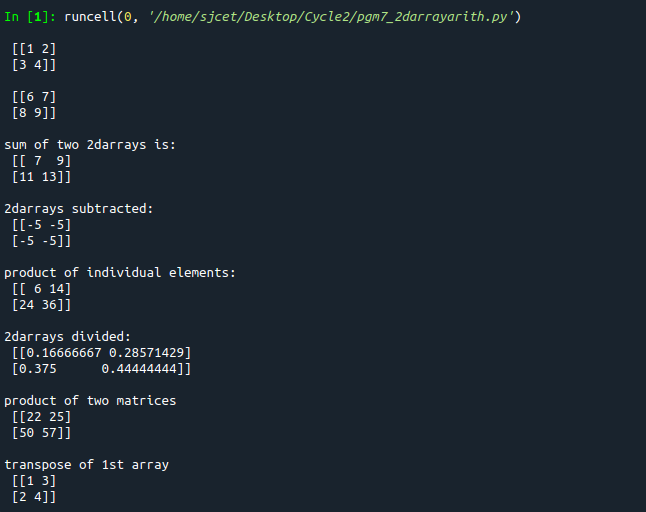
sumofd = np.trace(array)

print("\nsum of diagonal elements of 1st array\n", sumofd)

sumofd2 = np.trace(array2)

print("\nsum of diagonal elements of 2nd array\n", sumofd2)

**Output**



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

**Program**

import numpy as np

arr1 = np.arange(10, 16)

print("1D ARRAY ")

print("\nThe array is: ", arr1)

obj = 2

value = 40

arr = np.insert(arr1, obj, value, axis=None)

print("\nAfter inserting the new array is: ")

print(arr)

print("\nShape of the new array is : ", np.shape(arr))

print("\n2D ARRAY ")

arr1 = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9), (50, 51, 52)])

print("\nThe array is: ")

print(arr1)

print("\nThe shape of the array is: ", np.shape(arr1))

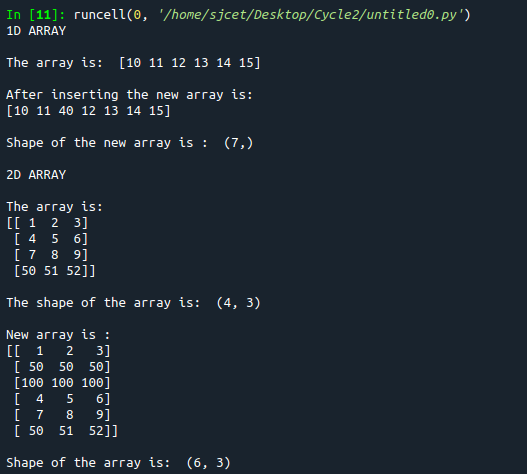
a = np.insert(arr1, 1, [[50], [100]], axis=0)

print("\nNew array is : ")

print(a)

print("\nShape of the array is: ", np.shape(a))

**Output**



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

**Program**

**Program**

import numpy as np

a= np.array([[3, 6,7,8]])

b=np.array([[3, 6,8,7], [4, 2,1,0],[3,1,3,3],[1,1,2,2]])

print("\n",a)

print("\n",b)

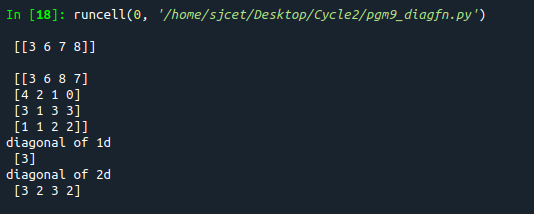
x=np.diag(a)

y=np.diag(b)

print("diagonal of 1d\n",x)

print("diagonal of 2d\n",y)

**Output**



**10. Demonstarte the use of append() function in 1D and 2D**

**array.**

**Program**

import numpy as np

a = np.array([(1, 2, 3), (4, 5, 6)])

b = np.array([1, 2, 3])

print("First array:")

print(a)

print("Second array")

print(b)

print("\n")

print("Append elements to array:")

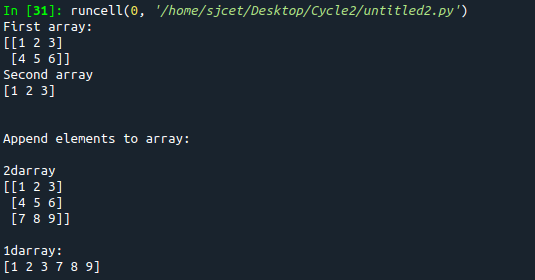
print("\n2darray")

print(np.append(a, [7, 8, 9]).reshape(3, 3))

print("\n1darray:")

print(np.append(b, [7, 8, 9]))

**Output**



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

**Program**

import numpy as np

a=np.array([0.4,0.5])

b=np.sum(a)

print ("\nsum:", b)

**Output**



**12.Create a 1 Dimensional array .Display the elements from indices 4 to 10 in descending order(use–values)**

**Program**

import numpy as np

a = np.array([1,2,8,9,3,4,5,6,7,10])

print(a)

array\_copy = np.sort(a)[::-1]

print("sorted array desc",array\_copy)

print("sliced 4:10:",array\_copy[4:10])

**Output**

