

LAB – 3

PYTHON BASIC PRACTICE – III

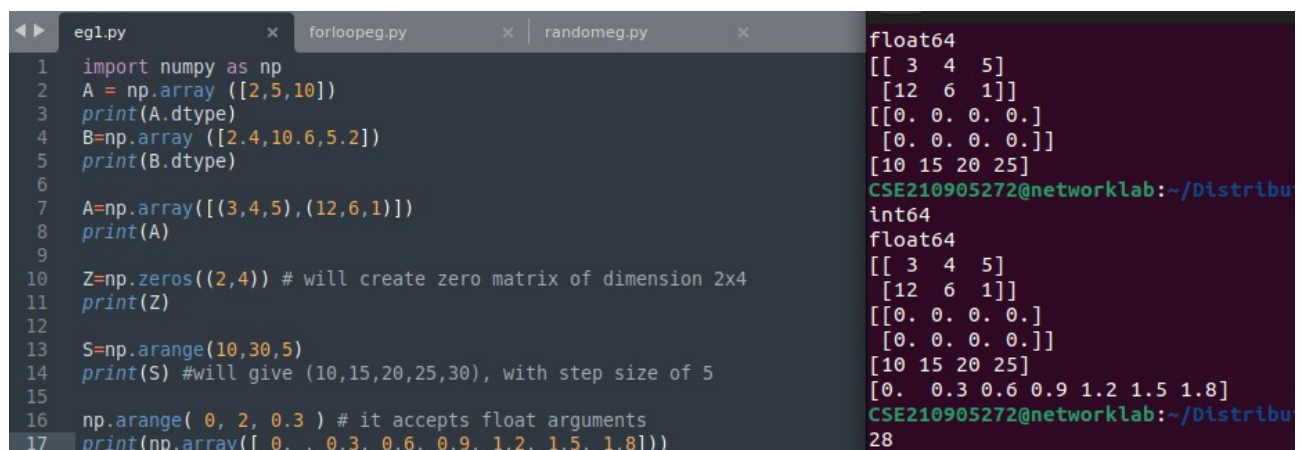
```
import numpy as np
```

```
A = np.array ([2,5,10])  
print(A.dtype)
```

```
B=np.array ([2.4,10.6,5.2])  
print(B.dtype)
```

```
A=np.array([(3,4,5),(12,6,1)])  
print(A)
```

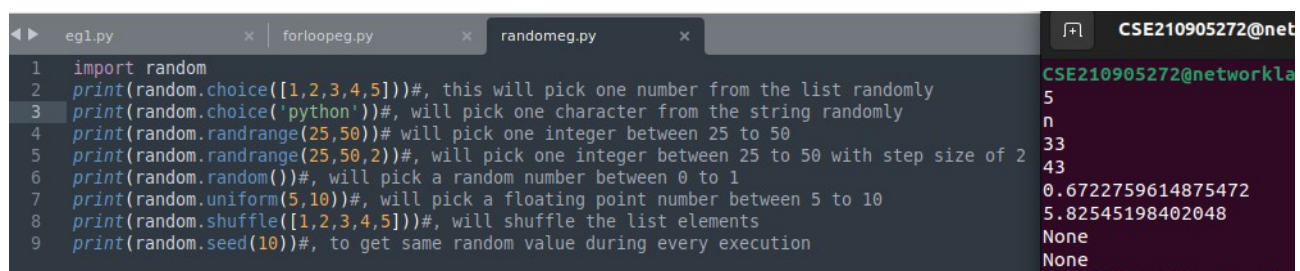
```
Z=np.zeros((2,4)) # will create zero matrix of dimension 2x4  
print(Z)
```



```
1 import numpy as np  
2 A = np.array ([2,5,10])  
3 print(A.dtype)  
4 B=np.array ([2.4,10.6,5.2])  
5 print(B.dtype)  
6  
7 A=np.array([(3,4,5),(12,6,1)])  
8 print(A)  
9  
10 Z=np.zeros((2,4)) # will create zero matrix of dimension 2x4  
11 print(Z)  
12  
13 S=np.arange(10,30,5)  
14 print(S) #will give (10,15,20,25,30), with step size of 5  
15  
16 np.arange( 0, 2, 0.3 ) # it accepts float arguments  
17 print(np.array([ 0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8]))
```

float64
[[3 4 5]
 [12 6 1]]
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [10 15 20 25]]
CSE210905272@networklab:~/Distribu
int64
float64
[[3 4 5]
 [12 6 1]]
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [10 15 20 25]
 [0. 0.3 0.6 0.9 1.2 1.5 1.8]]
CSE210905272@networklab:~/Distribu
28

```
import random  
print(random.choice([1,2,3,4,5]))#, this will pick one number from the list randomly  
print(random.choice('python'))#, will pick one character from the string randomly  
print(random.randrange(25,50))# will pick one integer between 25 to 50  
print(random.randrange(25,50,2))#, will pick one integer between 25 to 50 with step size of 2  
print(random.random())#, will pick a random number between 0 to 1  
print(random.uniform(5,10))#, will pick a floating point number between 5 to 10  
print(random.shuffle([1,2,3,4,5]))#, will shuffle the list elements  
print(random.seed(10))#, to get same random value during every execution
```



```
1 import random  
2 print(random.choice([1,2,3,4,5]))#, this will pick one number from the list randomly  
3 print(random.choice('python'))#, will pick one character from the string randomly  
4 print(random.randrange(25,50))# will pick one integer between 25 to 50  
5 print(random.randrange(25,50,2))#, will pick one integer between 25 to 50 with step size of 2  
6 print(random.random())#, will pick a random number between 0 to 1  
7 print(random.uniform(5,10))#, will pick a floating point number between 5 to 10  
8 print(random.shuffle([1,2,3,4,5]))#, will shuffle the list elements  
9 print(random.seed(10))#, to get same random value during every execution
```

CSE210905272@net
5
n
33
43
0.6722759614875472
5.82545198402048
None
None

```
# Calculate sum of all the elements in a 2D Numpy Array (iterate over range)
```

```
import numpy as np
```

```
a=np.array([(3,2,9),(1,6,7)])
```

```
s=0
```

```
for i in range(a.shape[0]):  
    for j in range(a.shape[1]):  
        s+=a[i,j]
```

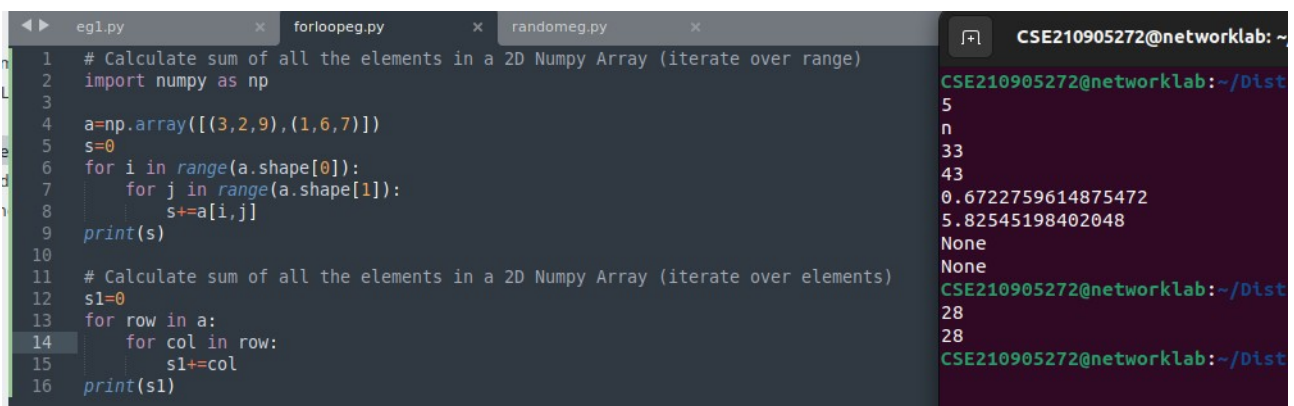
```
print(s)
```

```
# Calculate sum of all the elements in a 2D Numpy Array (iterate over elements)
```

```
s1=0
```

```
for row in a:  
    for col in row:  
        s1+=col
```

```
print(s1)
```



```
1 # Calculate sum of all the elements in a 2D Numpy Array (iterate over range)
2 import numpy as np
3
4 a=np.array([(3,2,9),(1,6,7)])
5 s=0
6 for i in range(a.shape[0]):
7     for j in range(a.shape[1]):
8         s+=a[i,j]
9 print(s)
10
11 # Calculate sum of all the elements in a 2D Numpy Array (iterate over elements)
12 s1=0
13 for row in a:
14     for col in row:
15         s1+=col
16 print(s1)
```

CSE210905272@networklab: ~/Dist
5
n
33
43
0.6722759614875472
5.82545198402048
None
None
CSE210905272@networklab: ~/Dist
28
28
CSE210905272@networklab: ~/Dist

```
import numpy as np
```

```
a = np.arange(15).reshape(3, 5)
```

```
np.array([[ 0, 1, 2, 3, 4],
```

```
[ 5, 6, 7, 8, 9],
```

```
[10, 11, 12, 13, 14]])
```

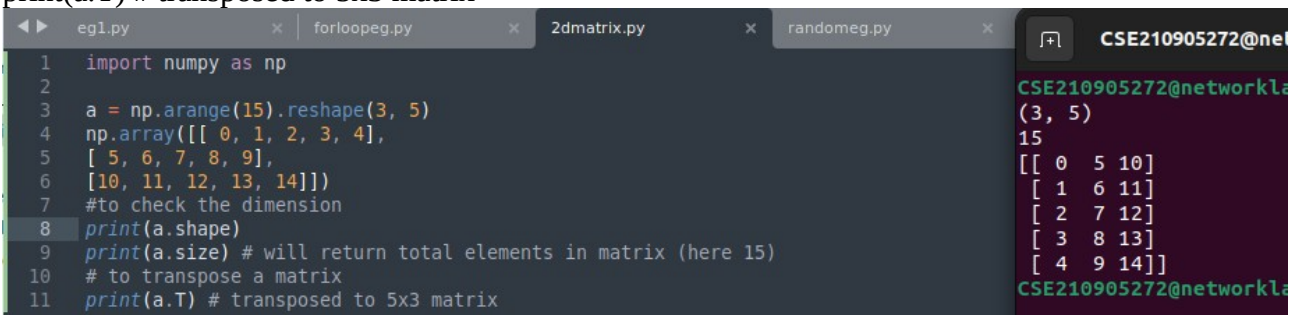
```
#to check the dimension
```

```
print(a.shape)
```

```
print(a.size) # will return total elements in matrix (here 15)
```

```
# to transpose a matrix
```

```
print(a.T) # transposed to 5x3 matrix
```



```
1 import numpy as np
2
3 a = np.arange(15).reshape(3, 5)
4 np.array([[ 0, 1, 2, 3, 4],
5 [ 5, 6, 7, 8, 9],
6 [10, 11, 12, 13, 14]])
7 #to check the dimension
8 print(a.shape)
9 print(a.size) # will return total elements in matrix (here 15)
10 # to transpose a matrix
11 print(a.T) # transposed to 5x3 matrix
```

CSE210905272@networkla
(3, 5)
15
[[0 5 10]
 [1 6 11]
 [2 7 12]
 [3 8 13]
 [4 9 14]]
CSE210905272@networkla

```
import numpy as np
```

```
c = np.arange(24).reshape(2,3,4) # 1st value indicates (no of planes) (3,4) is the dimension
print(c)
```

```
print(c.shape)
```

The screenshot shows a Jupyter Notebook with several tabs: 'egl.py', 'forloopeg.py', '2dmatrix.py', '3dmatrix.py', and 'randomeg.py'. The active tab is '3dmatrix.py', which contains the following code:

```
1 import numpy as np
2
3 c = np.arange(24).reshape(2,3,4) # 1st value indicates (no of planes) (3,4) is the dimension
4 print(c)
5
6 print(c.shape)
```

The output of the code is displayed on the right side of the notebook, showing the 3D array 'c' and its shape:

```
[[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
 [[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]]
(2, 3, 4)
```

```
import numpy as np
a = np.array( [20,30,40,50] )
b = np.arange( 4 )
print(b)
# array([0, 1, 2, 3])
c = a-b
print(c)
# array([20, 29, 38, 47])
```

```
print(b**2)
array([0, 1, 4, 9])
10*np.sin(a)
array([ 9.12945251, -9.88031624, 7.4511316 , -2.62374854])
a<35
array([ True,  True, False, False], dtype=bool)
```

```
import numpy as np
a = np.array([4.,2.])
b = np.array([3.,8.])
print(np.column_stack((a,b))) # returns a 2D array
print(np.array([[ 4., 3.],
 [ 2., 8.])))
```

```
np.hstack((a,b))
# the result is different
np.array([ 4., 2., 3., 8.])
print(np.hstack((a[0],b[0]))) # the result is the same
np.array([[ 4., 3.],
 [ 2., 8.]])
```

```
import numpy as np
a = np.arange(10)**3
print(a)
np.array([ 0, 1, 8, 27, 64, 125, 216, 343, 512, 729])
```

```

print(a[2:5])
np.array([ 8, 27, 64])
print(a[0:6:2])
np.array([0,8,64,216])

```

The screenshot shows a code editor with four tabs: q1.py, q2.py, forlooppeg.py, and slicingeg.py. The active tab is slicingeg.py, which contains the following code:

```

1
2
3 import numpy as np
4 a = np.arange(10)**3
5 print(a)
6 np.array([ 0, 1, 8, 27, 64, 125, 216, 343, 512, 729])
7 print(a[2:5])
8 np.array([ 8, 27, 64])
9 print(a[0:6:2])
10 np.array([0,8,64,216])

```

The terminal output on the right shows the execution of the script:

```

CSE210905272@networklab:~/DistributedSystems/Lab3_NumPy/sample$
eg.py
[[4. 3.]
 [2. 8.]]
[[4. 3.]
 [2. 8.]]
[4. 3.]
CSE210905272@networklab:~/DistributedSystems/Lab3_NumPy/sample$
g.py
[ 0  1  8 27 64 125 216 343 512 729]
[ 8 27 64]
[ 0 8 64]
CSE210905272@networklab:~/DistributedSystems/Lab3_NumPy/sample$

```

```
import numpy as np
```

```

A = np.array( [[1,1],[0,1]] )
B = np.array( [[2,0],[3,4]] )
print(A*B)
# elementwise product
np.array([[2, 0],
[0, 4]])
A.dot(B)
# matrix product
np.array([[5, 4],
[3, 4]])
# (OR)
np.dot(A, B)
np.array([[5, 4],
[3, 4]])
# another matrix product
b = np.arange(12).reshape(3,4)
print(b)
np.array([[ 0, 1, 2, 3],
[ 4, 5, 6, 7],
[ 8, 9, 10, 11]])
b.sum(axis=0)
np.array([12, 15, 18, 21])# sum of each column
b.sum(axis=1)
np.array([6, 22, 38])

```

```

1 import numpy as np
2
3 A = np.array( [[1,1],[0,1]] )
4 B = np.array( [[2,0],[3,4]] )
5 print(A*B)
6 # elementwise product
7 np.array([[2, 0],
8 [0, 4]])
9 A.dot(B)
10 # matrix product
11 np.array([[5, 4],
12 [3, 4]])
13 # (OR)
14 np.dot(A, B)
15 np.array([[5, 4],
16 [3, 4]])
17 # another matrix product
18 b = np.arange(12).reshape(3,4)
19 print(b)
20 np.array([[ 0, 1, 2, 3],
21 [ 4, 5, 6, 7],
22 [ 8, 9, 10, 11]])
23 b.sum(axis=0)
24 np.array([12, 15, 18, 21])# sum of each column
25 b.sum(axis=1)
26 np.array([6, 22, 38])

```

```

CSE210905272@networklab:~/D
seg.py
[[2 0]
 [0 4]]
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
CSE210905272@networklab:~/D

```

```

1 import numpy as np
2 b=np.array([[ 0, 1, 2, 3],
3 [10, 11, 12, 13],
4 [20, 21, 22, 23],
5 [30, 31, 32, 33],
6 [40, 41, 42, 43]])
7 b[2,3] #will fetch 23
8 b[0:5,1] #or b[:5,1] or b[:,1] #will fetch [1,11,21,31,41]
9 b[-1,:] # will fetch last row
10 b[:, -1] # will fetch last col
11 for row in b:
12     print (row) # will print every rowfor element in b.flat:
13 # print (element) # will show all elements of b in 1-D array
14 # Changing the shape of a matrix
15 b.ravel() # returns the array flattened to (1x 20)
16 # Later, we can convert 5

```

```

CSE210905272@networklab: ~/Distrib
CSE210905272@networklab:~/Distributed
[0 1 2 3]
[10 11 12 13]
[20 21 22 23]
[30 31 32 33]
[40 41 42 43]
CSE210905272@networklab:~/Distributed

```

LAB EXERCISES:

Q1)Write a program to find the factors of a given number (get input from user) using for loop.

->

```
import numpy as np
import pandas as pd
```

```
n=int(input("Enter number: "))
print("The factors are: \n")
for i in range(1,n+1):
    if (n%i==0):
        print(i)
```



```
q1.py x
1 # Write a program to find the factors of a given number (get input from user)
2
3 import numpy as np
4 import pandas as pd
5 from icecream import ic
6
7 n=int(input("Enter number: "))
8 print("The factors are: \n")
9 for i in range(1,n+1):
10     if (n%i==0):
11         # ic(i)
12         print(i)
```

```
CSE210905272@networklab: ~/DistributedSys
CSE210905272@networklab:~/DistributedSys
Enter number: 12
The factors are:
1
2
3
4
6
12
CSE210905272@networklab:~/DistributedSys
```

Q2)Find the sum of columns and rows using axis.

->

```
import numpy as np
import pandas as pd
from icecream import ic
```

```
a=np.array([[ 0, 1, 2, 3, 4],
[ 5, 6, 7, 8, 9],
[10, 11, 12, 13, 14]])
ic("ROW sum:")
print(a.sum(axis=1))
ic("COLUMN sum:")
print(a.sum(axis=0))
```

```
q1.py x q2.py x q3.py x
1 # Find the sum of columns and rows using axis
2
3 import numpy as np
4 import pandas as pd
5 from icecream import ic
6
7 a=np.array([[ 0, 1, 2, 3, 4],
8 [ 5, 6, 7, 8, 9],
9 [10, 11, 12, 13, 14]])
10 ic("ROW sum:")
11 print(a.sum(axis=1))
12 ic("COLUMN sum:")
13 print(a.sum(axis=0))
```

```
CSE210905272@networklab:~/D
ic| 'ROW sum:'
[10 35 60]
ic| 'COLUMN sum:'
[15 18 21 24 27]
CSE210905272@networklab:~/D
```

Q3)Operations on Arrays (use numpy wherever required):

- Create array from list with type float
- Create array from tuple
- Creating a 3X4 array with all zeros
- Create a sequence of integers from 0 to 20 with steps of 5
- Reshape 3X4 array to 2X2X3 array

f) Find maximum and minimum element of array, Row wise max and min, column wise max and min and sum of elements. (Use functions max(), min(), sum())

->

```
import numpy as np
import pandas as pd
from icecream import ic
```

```
#a
list1=[1,2,3,4]
arr1=np.array(list1, dtype="float")
print(arr1)
```

```
#b
tuple1=(7,6,5,4)
arr2=np.array(tuple1, dtype="float")
print(arr2)
```

```
#c
print("Zeroes matrix:")
zeroes1=np.zeros((3,4))
print(zeroes1_
```

```
#d
print(np.arange(0,20,5))
```

```
#e
e=zeros1.reshape(2,2,3)
print(e)
```

```
#f
f=np.array([[1,11,2],[3,55,6]])
print("The array: ")
print(f)
print(f.max(axis=0))
print(f.min(axis=0))
print(f.max(axis=1))
print(f.min(axis=1))
```

```
q1.py x q2.py x q3.py x q4.py x q5.py
1 # Operations on Arrays (use numpy wherever required):
2 # a) Create array from list with type float
3 # b) Create array from tuple
4 # c) Creating a 3X4 array with all zeros
5 # d) Create a sequence of integers from 0 to 20 with steps of 5
6 # e) Reshape 3X4 array to 2X2X3 array
7 # f) Find maximum and minimum element of array, Row wise max and min, column wise max
8 # and min and sum of elements. (Use functions max(), min(), sum())
9
10 import numpy as np
11 import pandas as pd
12 from icecream import ic
13 #a
14 list1=[1,2,3,4]
15 arr1=np.array(list1, dtype="float")
16 print(arr1)
17 #b
18 tuple1=(7,6,5,4)
19 arr2=np.array(tuple1, dtype="float")
20 print(arr2)
21 #c
22 print("Zeroes matrix:")
23 zeroes1=np.zeros((3,4))
24 print(zeroes1)
25 #d
26 print(np.arange(0,20,5))
27
28 #e
29 e=zeros1.reshape(2,2,3)
30 print(e)
31
32 #f
33
34 f=np.array([[1,11,2],[3,55,6]])
35 print("The array: ")
36 print(f)
37 print(f.max(axis=0))
38 print(f.min(axis=0))
39 print(f.max(axis=1))
40 print(f.min(axis=1))
```

```
CSE210905272@netwo
CSE210905272@networklab:
[1. 2. 3. 4.]
[7. 6. 5. 4.]
Zeroes matrix:
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[ 0  5 10 15]
[[[0. 0. 0.]
  [0. 0. 0.]]
 [0. 0. 0.]]

[[[0. 0. 0.]
  [0. 0. 0.]]
 [0. 0. 0.]]
The array:
[[ 1 11  2]
 [ 3 55  6]]
[ 3 55  6]
[ 1 11  2]
[11 55]
[1 3]
```

Q4)Write a program to transpose a given matrix.

```
import numpy as np
import pandas as pd
from icecream import ic
```

```
A=np.array([[1,2,3],[4,5,6],[7,8,9],[11,12,13]])
```

```
print(A)
print(A.shape)
```

```
#transpose
A=A.T
print(A)
print(A.shape)
```



```
q1.py x q2.py x q3.py x q4.py
1 # Write a program to transpose a given matrix.
2
3 import numpy as np
4 import pandas as pd
5 from icecream import ic
6
7 A=np.array([[1,2,3],[4,5,6],[7,8,9],[11,12,13]])
8
9 print(A)
10
11 print(A.shape)
12 #transpose
13 A=A.T
14 print(A)
15 print(A.shape)
```

```
CSE210905272@networklab:~/Distr
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [11 12 13]]
(4, 3)
[[ 1  4  7 11]
 [ 2  5  8 12]
 [ 3  6  9 13]]
(3, 4)
CSE210905272@networklab:~/Distr
```

Q5)Write a program to add two matrices.

->

```
import numpy as np
import pandas as pd
from icecream import ic
```

```
X=np.array([[12,7,3],
            [4 ,5,6],
            [7 ,8,9]])
```

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

```
Z=X+Y
print(Z)
```

```
q1.py x q2.py x q3.py x q4.py x q5.py
1 # Write a program to add two matrices.
2
3 import numpy as np
4 import pandas as pd
5 from icecream import ic
6
7 X=np.array([[12,7,3],
8             [4 ,5,6],
9             [7 ,8,9]])
10 Y=np.array([[5,8,1],
11             [6,7,3],
12             [4,5,9]])
13
14 Z=X+Y
15 print(Z)
16
17 X = [[12,7,3],
18       [4 ,5,6],
19       [7 ,8,9]]
20
21 Y = [[5,8,1],
22       [6,7,3],
23       [4,5,9]]
24
25 result = [[0,0,0],
26            [0,0,0],
27            [0,0,0]]
28
29 for i in range(len(X)):
30     for j in range(len(X[0])):
31         result[i][j] = X[i][j] + Y[i][j]
32
33 for r in result:
34     print(r)
35
```

```
CSE210905272@networkla
CSE210905272@networkLab:~/D
[[17 15  4]
 [10 12  9]
 [11 13 18]]
[17, 15, 4]
[10, 12, 9]
[11, 13, 18]
CSE210905272@networkLab:~/D
```

Q6)Write a program to find element wise product between two matrices.

->

```
import numpy as np
import pandas as pd
from icecream import ic
```

```
X=np.array([[12,7,3],
            [4 ,5,6],
            [7 ,8,9]])
Y=np.array([[5,8,1],
            [6,7,3],
            [4,5,9]])
```

```
Z=X*Y
print(Z)
```

```
q1.py x q2.py x q3.py x q4.py x q5.py q6.py
1 # Write a program to find element wise product between two matrices.
2
3 import numpy as np
4 import pandas as pd
5 from icecream import ic
6
7 X=np.array([[12,7,3],
8            [4 ,5,6],
9            [7 ,8,9]])
10 Y=np.array([[5,8,1],
11            [6,7,3],
12            [4,5,9]])
13
14 Z=X*Y
15 print(Z)
16
17 X = [[12,7,3],
18      [4 ,5,6],
19      [7 ,8,9]]
20
21 Y = [[5,8,1],
22      [6,7,3],
23      [4,5,9]]
24
25 Z = [[0,0,0],
26      [0,0,0],
27      [0,0,0]]
28
29 for i in range(len(X)):
30     for j in range(len(X[0])):
31         Z[i][j] = X[i][j] * Y[i][j]
32
33 for r in Z:
34     print(r)
```

```
CSE210905272@networklab: ~/DistributedSystems/Lab
CSE210905272@networklab:~/DistributedSystems/Lab3_
[[60 56  3]
 [24 35 18]
 [28 40 81]]
[60, 56,  3]
[24, 35, 18]
[28, 40, 81]
CSE210905272@networklab:~/DistributedSystems/Lab3_
```