Q1. Write a Python program to multiply two matrices using numpy

In [1]:
```
import numpy as np

# input two matrices
mat1 = ([1, 6, 5],[3 ,4, 8],[2, 12, 3])
mat2 = ([3, 4, 6],[5, 6, 7],[6,56, 7])

# This will return dot product
res = np.dot(mat1,mat2)
print(res)
```

```
[[ 63 320  83]
 [ 77 484 102]
 [ 84 248 117]]
```

Q2. Write a Python program to get the floor, ceiling and truncated values of the elements of an numpy array.

In [5]:
```
import numpy as np

x = np.array([-1.6, -1.5, -0.3, 0.1, 1.4, 1.8, 2.0])
print("orignal array: ",x)

#floor value
print("Floor values of the above array elements: ", np.floor(x))

#ceiling value
print("ceiling values of the array is: ", np.ceil(x))

#truncate
print("Truncated values of the array elements: ", np.trunc(x))
```

```
orignal array:  [-1.6 -1.5 -0.3  0.1  1.4  1.8  2. ]
Floor values of the above array elements:  [-2. -2. -1.  0.  1.  1.  2.]
ceiling values of the array is:  [-1. -1. -0.  1.  2.  2.  2.]
Truncated values of the array elements:  [-1. -1. -0.  0.  1.  1.  2.]
```

Q3. Write a python program to find the inverse of a matrix

In [7]:
```
import numpy as np
array1 = np.array([[2,3],[4,5]])
try:
    inverse = np.linalg.inv(array1)
    print(inverse)
except np.linalg.LinalgError:
    #not invertible. skip this one.
    pass
```

```
[[-2.5  1.5]
 [ 2.  -1. ]]
```

In [8]:
```
import numpy as np
array1 = np.array([[2,3],[4,5]])
inverse = np.linalg.inv(array1)
print(inverse)
```

```
[[-2.5  1.5]
 [ 2.  -1. ]]
```

Q4. Write a Python program to perform addition, subtraction, multiplication and divsion on the given polynomials

In [9]:
```
from numpy.polynomial import polynomial as P
x = (10,20,30)
y = (30,40,50)
print("Addition:")
print(P.polyadd(x,y))
print("Subtraction:")
print(P.polysub(x,y))
print("Multiplication:")
print(P.polymul(x,y))
print("Division:")
print(P.polydiv(x,y))
```

```
Addition:
[40. 60. 80.]
Subtraction:
[-20. -20. -20.]
Multiplication:
[ 300. 1000. 2200. 2200. 1500.]
Division:
(array([0.6]), array([-8., -4.]))
```

Q5. Write a Python program to create a random array with N elements and compute the average,

variance, standard deviation of the array elements

In [75]:
```
import numpy as np
n=int(input("Enter a number: "))
K = np.random.randn(n)

print("Average of the array elements:")
mean = K.mean()

print("\t",mean)

print("Standard deviation of the array elements:")
std = K.std()
print("\t",std)

print("Variance of the array elements:")
var = K.var()
print("\t",var)
```

```
Enter a number: 100
Average of the array elements:
         0.046370962353151704
Standard deviation of the array elements:
         0.975633268555837
Variance of the array elements:
         0.951860274712946
```

Q6. Write a Python program to compute the reciprocal for all elements in a given array.

In [78]:
```
import numpy as np

J = np.array([1.,2.,0.4,.3])
print(J)

r1 = np.reciprocal(J)
print(r1)
```

```
[1.  2.  0.4 0.3]
[1.         0.5        2.5        3.33333333]
```

Q7. Write a Python program to sort the specified number of elements from beginning of a given array

In [82]:
```
import numpy as np
nums = np.random.rand(10)
print(nums)

print("\nSorted first N elements:")

print(nums[np.argpartition(nums,range(n))])
```

```
[0.85108419 0.82893235 0.54456787 0.22406909 0.84405357 0.27511553
 0.64267283 0.7491041  0.21262014 0.99211253]

Sorted first N elements:
[0.21262014 0.22406909 0.27511553 0.82893235 0.84405357 0.54456787
 0.64267283 0.7491041  0.85108419 0.99211253]
```

Q8. write a python program to generate N random numbers from the normal distribution

In [1]:
```
import numpy as np
n = int(input("enter number: "))
x = np.random.normal(size=n)
print(x)
```

```
enter number: 7
[ 0.24834338  1.487423    1.08978733 -0.97848515 -1.9531901  -1.68981199
  1.65115415]
```

Q9. Write a Python program to create random vector of size N and replace the maximum value by N

In [8]:
```
import numpy as np
n=int(input("Enter value of N: "))
x = np.random.random(n)
print("Original array:")
print(x)
x[x.argmax()] = n
print("Maximum value replaced by -1:")
print(x)
```

```
Enter value of N: 4
Original array:
[0.8774987  0.55760855 0.42176148 0.68338819]
Maximum value replaced by -1:
[4.         0.55760855 0.42176148 0.68338819]
```

Q10. Write a Python program to find the most frequent value in an array

In [15]:
```
import numpy as np
x = np.random.randint(0, 10, 40)
print("Original array:")
print(x)
print("Most frequent value in the above array:")
print(np.bincount(x).argmax())
```

```
Original array:
[8 9 0 9 7 0 2 3 0 7 9 6 3 8 5 2 0 7 5 8 1 2 1 5 6 8 3 6 6 1 8 3 9 9 3 7 3
 0 4 5]
Most frequent value in the above array:
3
```

Q11. Write a Python program to convert cartesian coordinates to polar coordinates of a random MxN matrix representing cartesian coordinates

In [22]:
```
import numpy as np
m=int(input("Enter no. of rows: "))
n=int(input("Enter no. of columns: "))
z= np.random.random((m,n))
x,y = z[:,0], z[:,1]
r = np.sqrt(x**2+y**2)
t = np.arctan2(y,x)
print("Cartesian Product: \n",r)
print("Polar Product: \n",t)
```

```
Enter no. of rows: 3
Enter no. of columns: 3
Cartesian Product:
 [1.03284873 0.73454762 0.8492903 ]
Polar Product:
 [1.10414639 0.80932316 1.04965517]
```

Q12.Write a NumPy program to create a 3x3x3 array with random values

In [17]:
```
import numpy as np
x = np.random.random((3,3,3))
print(x)
```

```
[[[0.30090959 0.87888608 0.463959  ]
  [0.89720071 0.68596973 0.23975786]
  [0.04776267 0.29758204 0.94713142]]

 [[0.41005787 0.65493806 0.37058331]
  [0.2837993  0.51455407 0.82395072]
  [0.72247882 0.0391099  0.88927666]]

 [[0.06402469 0.99507112 0.4489893 ]
  [0.55880769 0.26837698 0.33514473]
  [0.98143882 0.52123206 0.43184237]]]
```

Q13. Write a Python program to find point by point distances of a random vector with shape (J,K) representing coordinates

In [ ]:
```
import numpy as np
j=int(input("Enter J: "))
k=int(input("Enter K: "))
a= np.random.random((j,k))
x,y = np.atleast_2d(a[:,0], a[:,1])
d = np.sqrt( (x-x.T)**2 + (y-y.T)**2)
print(d)
```

Q14. Write a NumPy program to check if each element of a given array is composed of digits only, lower case letters only and upper case letters only.

In [18]:
```
import numpy as np
x = np.array(['Python', 'PHP', 'JS', 'Examples', 'html5', '5'], dtype=np.str)
print("\nOriginal Array:")
print(x,"\n")
r1 = np.char.isdigit(x)
r2 = np.char.islower(x)
r3 = np.char.isupper(x)
print("Digits only =", r1)
print("Lower cases only =", r2)
print("Upper cases only =", r3)
```

```
Original Array:
['Python' 'PHP' 'JS' 'Examples' 'html5' '5']

Digits only = [False False False False False  True]
Lower cases only = [False False False False  True False]
Upper cases only = [False  True  True False False False]

<ipython-input-18-3a068d2180f7>:2: DeprecationWarning: `np.str` is a deprecated
alias for the builtin `str`. To silence this warning, use `str` by itself. Doin
g this will not modify any behavior and is safe. If you specifically wanted the
numpy scalar type, use `np.str_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devd
ocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdocs/release/
1.20.0-notes.html#deprecations)
  x = np.array(['Python', 'PHP', 'JS', 'Examples', 'html5', '5'], dtype=np.str)
```

Q15. Write a Python program to Triangulate a location based on co-ordinates.

In [19]:
```
import numpy as np
data = [20.0497520, 31.39864012947, 12.30974023]
print(np.mean(data, axis=0))
```

```
21.25271078649
```

Q16. Write a Python code to determine the rank of the matrix

In [20]:
```
import numpy
A = numpy.matrix([[1,3,7],[2,8,3],[7,8,1]])
numpy.linalg.matrix_rank(A)
```

Out[20]:
```
3
```

Q17. Write a NumPy program to multiply a MxN matrix by a NxA matrix and create a real matrix

product

In [21]:
```
import numpy as np
m=int(input("Enter value M: "))
n=int(input("Enter value N: "))
a=int(input("Enter value A: "))
x = np.random.random((m,n))
print("First array:")
print(x)
y = np.random.random((n,a))
print("Second array:")
print(y)
z = np.dot(x, y)
print("Dot product of two arrays:")
print(z)
```

```
Enter value M: 5
Enter value N: 3
Enter value A: 2
First array:
[[0.70700975 0.25854579 0.67020295]
 [0.38333021 0.77717467 0.98132691]
 [0.70834844 0.66851938 0.13757942]
 [0.93791746 0.5139777  0.60958485]
 [0.54601408 0.85982763 0.98943062]]
Second array:
[[0.21873841 0.49696146]
 [0.21332757 0.04405586]
 [0.71155794 0.66908401]]
Dot product of two arrays:
[[0.68669337 0.81116913]
 [0.94791278 0.88132958]
 [0.39545235 0.47352625]
 [0.74855913 0.89661603]
 [1.00689641 0.9712406 ]]
```