1.create a 3 * 3 * 3 matrix with random variables

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

In [2]:
```
t=np.random.random((3,3))
t
```

Out[2]:
```
array([[0.08400517, 0.13236976, 0.21540779],
       [0.74069831, 0.91388126, 0.5470316 ],
       [0.92117222, 0.33417923, 0.54742096]])
```

2.Create a 5 * 5 matrix with 1,2,3,4 just below the diagnol

In [4]:
```
c=np.eye(5)
c
```

Out[4]:
```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

In [6]:
```
m=np.zeros((5,5),dtype=int)
for i in range (1,5):
    m[i,i-1]=i

print(m)
```

```
[[0 0 0 0 0]
 [1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]]
```

3.Create a 8 * 8 matrix and fill it with checkerboard pattern

In [10]:
```
m=np.zeros((8,8),dtype=int)
m[1::2,::2]=1
m[::2,1::2]=1
m
```

Out[10]:
```
array([[0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0]])
```

4.Normalize 5 * 5 random matrix

In [11]:
```
import numpy as np

# Create a random 5x5 matrix
random_matrix = np.random.rand(5, 5)

# Calculate the mean and standard deviation of the matrix
mean = np.mean(random_matrix)
std_dev = np.std(random_matrix)

# Normalize the matrix
normalized_matrix = (random_matrix - mean) / std_dev

print("Original Matrix:")
print(random_matrix)
print("\nNormalized Matrix:")
print(normalized_matrix)
```

```
Original Matrix:
[[0.52247308 0.99786667 0.03805944 0.53410605 0.57213547]
 [0.63958346 0.64831299 0.88690027 0.41538648 0.59996696]
 [0.35260653 0.44959303 0.993844   0.20544953 0.18779712]
 [0.79012381 0.46241815 0.1998887  0.49142579 0.29684653]
 [0.2313428  0.2205463  0.69397292 0.04204558 0.69054407]]

Normalized Matrix:
[[ 0.13488575  1.91889526 -1.68297332  0.17854079  0.3212538 ]
 [ 0.57436588  0.60712518  1.50247166 -0.26697814  0.42569702]
 [-0.50257246 -0.13861124  1.90379938 -1.05480854 -1.12105274]
 [ 1.13929862 -0.09048241 -1.07567668  0.01837456 -0.71182301]
 [-0.95763885 -0.99815488  0.77847318 -1.66801453  0.76560573]]
```

5.How to find the commom values between two arrarys

In [16]:
```
# Create two NumPy arrays
a1 = np.array([1, 2, 3, 4, 5])
a2 = np.array([3, 4, 5, 6, 7])

# Find common values using NumPy's intersect1d function
common_vals = np.intersect1d(a1, a2)

print(common_vals)
```

```
[3 4 5]
```

6. How to get dates of yesterday,today and tomorrow

In [17]:
```
import numpy as np
import datetime

# Get today's date
today = datetime.date.today()

# Calculate yesterday's date by subtracting 1 day from today
yesterday = today - datetime.timedelta(days=1)

# Calculate tomorrow's date by adding 1 day to today
tomorrow = today + datetime.timedelta(days=1)

# Convert the dates to NumPy datetime64 format
today_np = np.datetime64(today)
yesterday_np = np.datetime64(yesterday)
tomorrow_np = np.datetime64(tomorrow)

# Print the dates
print("Yesterday (NumPy):", yesterday_np)
print("Today (NumPy):", today_np)
print("Tomorrow (NumPy):", tomorrow_np)
```

```
Yesterday (NumPy): 2023-09-20
Today (NumPy): 2023-09-21
Tomorrow (NumPy): 2023-09-22
```

7.consider two arrays and check them if they are equal

In [6]:
```
import numpy as np
x = np.array([[10,55],[65,95]],dtype=int)
y = np.array([[10,55],[65,95]],dtype=int)
# Check if the arrays are equal using NumPy's array_equal function
are_equal=False
are_equal = np.array_equal(x, y)

print("Array 1:", x)
print("Array 2:", y)
if are_equal:
    print("Yes both are same")
else:
    print("Not same")
```

```
Array 1: [[10 55]
 [65 95]]
Array 2: [[10 55]
 [65 95]]
Yes both are same
```

8. Create a random vector of size 10 and replace the lasrgest value with by 0

In [7]:
```
# Create a random vector of size 10
random_vector = np.random.rand(10)

# Find the index of the maximum value
max_index = np.argmax(random_vector)

# Replace the maximum value with 0
random_vector[max_index] = 0

print("Random Vector with Maximum Replaced:", random_vector)
```

```
Random Vector with Maximum Replaced: [0.47270506 0.19916112 0.07079588 0.7077
5642 0.         0.69919489
 0.28593942 0.09559788 0.46916143 0.59862112]
```

9.How to print all the values of the matrix

In [9]:
```
x = np.array([[10,55],[65,95]],dtype=int)
x
```

Out[9]:
```
array([[10, 55],
       [65, 95]])
```

10.Subtract the mean of each row of a matrix

In [11]:
```
# Create a random 5x5 matrix
random_matrix = np.random.rand(5, 5)

mean = np.mean(random_matrix)



final=random_matrix-mean


print("\nFinal Matrix:")
print(final)
```

```
Normalized Matrix:
[[-0.07069147  0.31091066  0.03078159 -0.55722236 -0.02252854]
 [ 0.36591473 -0.47015734 -0.15781011 -0.27049455  0.43872855]
 [ 0.06004941  0.27873752  0.30904542 -0.11409718  0.38069046]
 [ 0.06598129  0.41650956 -0.4701725   0.31689673 -0.02439094]
 [-0.53134255 -0.10530867  0.00450262  0.36660706 -0.55113938]]
```

11.Consider a given vector. How to add one to each element indexed by a second vector? And be careful with repeated indices

In [12]:
```
# Given vector
given_vector = np.array([1, 2, 3, 4, 5])

# Second vector with indices to increment
indices_to_increment = np.array([1, 3, 3, 4, 1])

# Create a dictionary to keep track of unique indices and their counts
unique_indices = {}
for idx in indices_to_increment:
    if idx in unique_indices:
        unique_indices[idx] += 1
    else:
        unique_indices[idx] = 1

# Add one to the elements at the specified unique indices
for idx, count in unique_indices.items():
    given_vector[idx] += count

print("Given Vector with Incremented Elements:", given_vector)
```

```
Given Vector with Incremented Elements: [1 4 3 6 6]
```

12.How to get diagnol of a dot product

In [13]:
```
# Create two matrices
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])

# Calculate the dot product of the two matrices
dot_product = np.dot(matrix1, matrix2)

# Get the diagonal elements of the dot product matrix
diagonal = np.diag(dot_product)

print("Dot Product Matrix:")
print(dot_product)
print("\nDiagonal of Dot Product:")
print(diagonal)
```

```
Dot Product Matrix:
[[19 22]
 [43 50]]

Diagonal of Dot Product:
[19 50]
```

13.How to find the most frrequent value in an array

In [14]:
```
x=np.random.randint(0,10,50)
x
```

Out[14]:
```
array([7, 4, 7, 6, 6, 8, 6, 1, 6, 1, 2, 6, 2, 4, 7, 7, 6, 4, 3, 5, 0, 5,
       4, 3, 7, 3, 2, 8, 5, 3, 5, 1, 5, 1, 8, 3, 1, 0, 3, 8, 2, 9, 2, 3,
       7, 0, 3, 4, 5, 9])
```

In [15]:
```
print(np.bincount(x).argmax())
```

```
3
```

14.How to get the n largest values from the array

In [16]:
```
import numpy as np

# create numpy 1d-array
arr = np.array([2, 0,  1, 5,
                4, 1, 9])

print("Given array:", arr)

# sort an array in
# ascending order

# np.argsort() return
# array of indices for
# sorted array
sorted_index_array = np.argsort(arr)

# sorted array
sorted_array = arr[sorted_index_array]

print("Sorted array:", sorted_array)

# we want 1 largest value
n = 1

# we are using negative
# indexing concept

# take n largest value
rslt = sorted_array[-n : ]

# show the output
print("{} largest value:".format(n),
      rslt[0])
```

```
Given array: [2 0 1 5 4 1 9]
Sorted array: [0 1 1 2 4 5 9]
1 largest value: 9
```

15.How to create a record array from a regular array

In [19]:
```
arra1 = np.array([("shaik", 88.5, 90),
                  ("Tauheer", 87, 99),
                ("Ahamed", 85.5, 91)])
print("Original arrays:")
print(arra1)
print("\nRecord array;")
result = np.core.records.fromarrays(arra1.T,
                                    names='col1, col2, col3',
                                    formats = 'S80, f8, i8')
print(result)
```

```
Original arrays:
[['shaik' '88.5' '90']
 ['Tauheer' '87' '99']
 ['Ahamed' '85.5' '91']]

Record array;
[(b'shaik', 88.5, 90) (b'Tauheer', 87. , 99) (b'Ahamed', 85.5, 91)]
```

16.How to swap two rows in an array

In [24]:
```
print("Before swapping")
print(arra1)
print("After swapping")
arra1[[0, 1]] = arra1[[1, 0]]
print(arra1)
```

```
Before swapping
[['shaik' '88.5' '90']
 ['Tauheer' '87' '99']
 ['Ahamed' '85.5' '91']]
After swapping
[['Tauheer' '87' '99']
 ['shaik' '88.5' '90']
 ['Ahamed' '85.5' '91']]
```

17.write a python program to reshape to the sumpu array?

In [42]:
```
# importing numpy
import numpy as np

# creating a numpy array
array = np.array([[23, 34, 121],
                  [23, 22, 67],
                  [686, 434, 123]])

# printing array
print(" 2-D Array : ")
print(array)


# reshaping numpy array
reshaped = array.reshape((-1))

# printing reshaped array
print("Reshaped 1-D Array : ")
print(reshaped)
```

```
 2-D Array :
[[ 23  34 121]
 [ 23  22  67]
 [686 434 123]]
Reshaped 1-D Array :
[ 23  34 121  23  22  67 686 434 123]
```

In [ ]: