1. Create a 3x3x3 array with random values

```
import numpy as np

random_array = np.random.rand(3, 3, 3)

print(random_array)
```

```
[[[0.45198155 0.07678681 0.67606888]
  [0.40704164 0.49020921 0.84470826]
  [0.80758708 0.29154294 0.60037452]]

 [[0.7322369  0.35830193 0.85503119]
  [0.26754801 0.24897212 0.21689195]
  [0.06542521 0.81721318 0.55235928]]

 [[0.86486811 0.60197519 0.58109906]
  [0.62265963 0.49347626 0.91160679]
  [0.25876599 0.68218588 0.64477402]]]
```

2.Create a 5x5 matrix with values 1,2,3,4 just below the diagonal

```
import numpy as np

matrix = np.zeros((5, 5))

np.fill_diagonal(matrix[1:], [1, 2, 3, 4])

print(matrix)
```

```
[[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 8x8 matrix and fill it with a checkerboard pattern

```
import numpy as np

checkerboard = np.zeros((8, 8), dtype=int)

checkerboard[1::2, ::2] = 1
checkerboard[::2, 1::2] = 1

print(checkerboard)
```

```
[[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 a 5x5 random matrix

```
import numpy as np

random_matrix = np.random.rand(5, 5)

mean = random_matrix.mean()
std_dev = random_matrix.std()

normalized_matrix = (random_matrix - mean) / std_dev

print("Original random matrix:")
print(random_matrix)
print("\nNormalized matrix:")
print(normalized_matrix)
```

```
Original random matrix:
[[0.744028   0.79839416 0.48948235 0.83856293 0.08674592]
 [0.65292975 0.92569961 0.37573943 0.97945137 0.61019141]
 [0.0361548  0.32155153 0.00831282 0.03318112 0.96893106]
 [0.54970563 0.15787144 0.84740816 0.59452967 0.49010516]
 [0.15955113 0.6789319  0.26845783 0.16952908 0.93620191]]

Normalized matrix:
[[ 0.73330247  0.90283169 -0.06044352  1.02808937 -1.31629068]
 [ 0.44923213  1.29980643 -0.41512643  1.46741974  0.31596179]
 [-1.47404826 -0.58409978 -1.56086747 -1.48332102  1.43461442]
 [ 0.12734984 -1.094501    1.05567134  0.26712401 -0.05850142]
 [-1.08926325  0.53031425 -0.74966109 -1.05814917  1.33255558]]
```

5. How to find common values between two arrays?

```
import numpy as np

array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([3, 4, 5, 6, 7])

common_values = np.intersect1d(array1, array2)
print(common_values)
```

```
[3 4 5]
```

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

```
import numpy as np
import datetime

today = np.datetime64('today')

yesterday = today - np.timedelta64(1, 'D')
tomorrow = today + np.timedelta64(1, 'D')

today = today.astype(datetime.datetime)
yesterday = yesterday.astype(datetime.datetime)
tomorrow = tomorrow.astype(datetime.datetime)

print("Yesterday:", yesterday)
print("Today:", today)
print("Tomorrow:", tomorrow)
```

```
Yesterday: 2023-09-20
Today: 2023-09-21
Tomorrow: 2023-09-22
```

7. Consider two random array A and B, check if they are equal

```
import numpy as np
```

```
import numpy as np

A = np.random.rand(5)
B = np.random.rand(5)

are_equal = np.array_equal(A, B)

if are_equal:
    print("Arrays A and B are equal.")
else:
    print("Arrays A and B are not equal.")
```

```
Arrays A and B are not equal.
```

8.Create random vector of size 10 and replace the maximum value by 0

```
import numpy as np

random_vector = np.random.rand(10)

max_index = np.argmax(random_vector)

random_vector[max_index] = 0

print("Random vector with the maximum value replaced:")
print(random_vector)
```

```
Random vector with the maximum value replaced:
[0.52591835 0.62564628 0.52246204 0.2648957  0.41301604 0.00178379
 0.         0.14435567 0.15005658 0.4024856 ]
```

9. How to print all the values of an array?

```
import numpy as np

my_array = np.array([1, 2, 3, 4, 5])
print(my_array)
```

```
[1 2 3 4 5]
```

10.Subtract the mean of each row of a matrix

```
import numpy as np

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

row_means = np.mean(matrix, axis=1, keepdims=True)

normalized_matrix = matrix - row_means

print("Original matrix:")
print(matrix)
print("\nMatrix with row-wise means subtracted:")
print(normalized_matrix)
```

```
Original matrix:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Matrix with row-wise means subtracted:
[[-1.  0.  1.]
 [-1.  0.  1.]
 [-1.  0.  1.]]
```

11.Consider a given vector, how to add 1 to each element indexed by a second vector (be careful with repeated indices)?

```
import numpy as np

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

index_vector = np.array([1, 3, 3, 4])

unique_indices, counts = np.unique(index_vector, return_counts=True)

given_vector[unique_indices] += 1

print("Given vector with 1 added at unique indices:")
```

```
print(given_vector)
```

```
Given vector with 1 added at unique indices:
[1 3 3 5 6]
```

12.How to get the diagonal of a dot product?

```
import numpy as np

A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

dot_product = np.dot(A, B)

diagonal = np.diag(dot_product)

print("Matrix A:")
print(A)
print("\nMatrix B:")
print(B)
print("\nDot product of A and B:")
print(dot_product)
print("\nDiagonal of the dot product:")
print(diagonal)
```

```
Matrix A:
[[1 2]
 [3 4]]

Matrix B:
[[5 6]
 [7 8]]

Dot product of A and B:
[[19 22]
 [43 50]]

Diagonal of the dot product:
[19 50]
```

13.How to find the most frequent value in an array?

```
import numpy as np
```

```
arr = np.array([1, 2, 2, 3, 4, 4, 4, 5, 5, 5, 5])

unique_values, counts = np.unique(arr, return_counts=True)

mode_index = np.argmax(counts)

most_frequent_value = unique_values[mode_index]

print("Array:", arr)
print("Most frequent value(s):", most_frequent_value)
```

```
Array: [1 2 2 3 4 4 4 5 5 5 5]
Most frequent value(s): 5
```

14.How to get the n largest values of an array

```
import numpy as np

arr = np.array([3, 1, 4, 1, 5, 9, 2, 6, 5, 3])

n = 3
largest_values = np.partition(arr, -n)[-n:]

print("Array:", arr)
print(f"{n} Largest Values:", largest_values)
```

```
Array: [3 1 4 1 5 9 2 6 5 3]
3 Largest Values: [5 6 9]
```

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

```
import numpy as np

data = [(1, 'Alice', 25),
        (2, 'Bob', 30),
        (3, 'Charlie', 35)]

dtype = [('ID', int), ('Name', 'U10'), ('Age', int)]
structured_array = np.array(data, dtype=dtype)

record_array = structured_array.view(np.recarray)

print("Record Array:")
```

```
print(record_array)
print("\nAccessing elements:")
print("ID:", record_array.ID)
print("Name:", record_array.Name)
print("Age:", record_array.Age)
```

```
Record Array:
[(1, 'Alice', 25) (2, 'Bob', 30) (3, 'Charlie', 35)]

Accessing elements:
ID: [1 2 3]
Name: ['Alice' 'Bob' 'Charlie']
Age: [25 30 35]
```

16. How to swap two rows of an array?

```
import numpy as np

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

row1_index = 0
row2_index = 1

array[row1_index], array[row2_index] = array[row2_index].copy(), array[row1_index].copy()

print("Original array:")
print(array)
```

```
Original array:
[[4 5 6]
 [1 2 3]
 [7 8 9]]
```

17. Write python code to reshape to the next dimension of numpy array?

```
import numpy as np

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

```
reshaped_array = array.reshape(-1, 2)

print("Original array:")
print(array)
print("\nReshaped array:")
print(reshaped_array)
```

```
Original array:
[1 2 3 4 5 6]

Reshaped array:
[[1 2]
 [3 4]
 [5 6]]
```