```
# Frist we import the libary

import numpy as np
import random
import numpy.matlib as nm


#Question no 1 :- Create a NumPy array 'arr' of integers from 0 to 5 and print its data type.

#Answer:-

arr = np.random.randint(0,5,(3,3))


arr
```

```
array([[4, 4, 1],
       [0, 3, 1],
       [2, 3, 2]])
```

```
type(arr)
```

```
numpy.ndarray
```

```
#Question no 2 :-  Given a NumPy array 'arr', check if its data type is float64.arr = np.arr([1.5,2.6,3.7])

#Answer :-
arr = np.array([1.5, 2.6, 3.7])

if arr.dtype == np.float64:
    print("The data type of arr is float64.")
else:
    print("The data type of arr is not float64.")
```

```
The data type of arr is float64.
```

```
#Question no 3:- Create a NumPy array 'arr' with a data type of complex128 containing three complex numbers.

#Answer :-

arr = np.array([1+2j, 3-4j, 5+6j], dtype=np.complex128)

print(arr)
print("Data type of arr:", arr.dtype)
```

```
[1.+2.j 3.-4.j 5.+6.j]
Data type of arr: complex128
```

```
#Question no 4:- Convert an existing NumPy array 'arr' of integers to float32 data type.

#Answer :-

arr = np.array([1,2,3,4,56])
arr1 = np.float32(arr)
print("After conversion the given array is :",arr1)
```

```
After conversio the given array is : [ 1.  2.  3.  4. 56.]
```

```
#Question no 5:- Given a NumPy array 'arr' with float64 data type, convert it to float32 to reduce decimal precision.

#Answer :-

arr = np.array([1.267,3.892,4.563,8.965])

if arr.dtype == np.float64:
  arr1 = np.float32(arr)
print(f"After conversion the given array is {arr1} and its data type is {arr1.dtype}")
```

```
After conversion the given array is [1.267 3.892 4.563 8.965] and its data type is float32
```

```
#Question no 6:- Write a function array_attributes that takes a NumPy array as input and returns its shape, size, and data type.

#Answer :-

def fun(arr):
    shape_info = f"The shape of the given array is {arr.shape}"
    size_info = f"The size of the given array is {arr.size}"
    dtype_info = f"The data type of the given array is {arr.dtype}"

    return shape_info, size_info, dtype_info

arr = np.array([1, 3, 4, 9, 10, -9])
result = fun(arr)

for info in result:
    print(info)
```

```
The shape of the given array is (6,)
The size of the given array is 6
The data type of the given array is int64
```

```
#Question no 7:- Create a function array_dimension that takes a NumPy array as input and returns its dimensionality.

##Anser :-

def dim(arr):
  return arr.ndim

arr = np.array([[1, 3, 4, 9, 10, -9],[1,2,3,4,5,6]])
result = dim(arr)

print(result)
```

```
2
```

```
#Question no 8:- Design a function item_size_info that takes a NumPy array as input and returns the item size and the total size in bytes.

#Answer:-

import sys

def item_size_info(arr):
  item_size = f"The item size of the the array is {arr.size}"
  siz_bytes = f"The total byte size of the array is {sys.getsizeof(arr)}"
  return [item_size,siz_bytes]

arr = np.array([1,5,-9,6,3])
result = item_size_info(arr)

for info in result:
  print(info)
```

```
The item size of the the array is 5
The total byte size of the array is 152
```

```
#Question no 9 :- Create a function array_strides that takes a NumPy array as input and returns the strides of the array.

#Answer :-

def array_stride(arr):
  strid = f"The strids of the given array is {arr.strides}"
  return strid

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

print(result)
```

```
The strids of the given array is (48, 8)
```

```
#Question no 10:- Design a function shape_stride_relationship that takes a NumPy array as input and returns the shape and strides of the arr

#Answer :-

def shape_stride_relationshi(arr):
  shape_info = f"The shape of the given array is {arr.shape}"
  stride_info = f"The stride of the given array is {arr.strides}"
  return [shape_info,stride_info]

arr = np.random.randint(1,100,(3,3))
result = shape_stride_relationshi(arr)

for info in result:
  print(info)
```

```
The shape of the given array is (3, 3)
The stride of the given array is (24, 8)
```

```
#Question no 11:- Create a function `create_zeros_array` that takes an integer `n` as input and returns a NumPy array of zeros with `n` elem

#Answer :-

def create_zeros_arra(n):
  return np.zeros(n)

n = int(input("Enter a no which is the size of the array : "))
result = create_zeros_arra(n)
print(result)
```

```
Enter a no which is the size of the array : 5
[0. 0. 0. 0. 0.]
```

```
#Question no 12:- Write a function `create_ones_matrix` that takes integers `rows` and `cols` as inputs and generates a 2D NumPy array fille

#Answer :-

def create_ones_matri(row,col):
  return np.ones((row,col),dtype = int)

row = int(input("Enter the size of the row : "))
col = int(input("Enter the size of the column : "))

result = create_ones_matri(row,col)

print(result)
```

```
Enter the size of the row : 3
Enter the size of the column : 3
[[1 1 1]
 [1 1 1]
 [1 1 1]]
```

```
#Question no 13 :-  Write a function `generate_range_array` that takes three integers start, stop, and step as arguments and creates a NumPy

#Amswer :-

def generate_range_array(start,stop,step):
  return np.arange(start,stop,step)

start = int(input("Enter the starting point of the range of the array : "))
stop = int(input("Enter the stoping point of the range of the array : "))
step = int(input("Enput the step size of the array : "))

result = generate_range_array(start,stop,step)

print(f"The resultant array is {result}")
```

```
Enter the starting point of the range of the array : 1
Enter the stoping point of the range of the array : 20
Enput the step size of the array : 2
The resultant array is [ 1  3  5  7  9 11 13 15 17 19]
```

```
#Question no 14 :- Design a function `generate_linear_space` that takes two floats `start`, `stop`, and an integer `num` as
#arguments and generates a NumPy array with num equally spaced values between `start` and `stop` (inclusive).

#Answer :-

def generate_linear_space(start,stop,num):
  return np.linspace(start,stop,num)

start = float(input("Enter the starting point of the range of the array : "))
stop = float(input("Enter the stoping point of the range of the array : "))
num = int(input("Enput the number present in the range : "))

result = generate_linear_space(start,stop,num)

print(result)
```

```
Enter the starting point of the range of the array : 0
Enter the stoping point of the range of the array : 20
Enput the number present in the range : 30
[ 0.          0.68965517  1.37931034  2.06896552  2.75862069  3.44827586
  4.13793103  4.82758621  5.51724138  6.20689655  6.89655172  7.5862069
  8.27586207  8.96551724  9.65517241 10.34482759 11.03448276 11.72413793
 12.4137931  13.10344828 13.79310345 14.48275862 15.17241379 15.86206897
 16.55172414 17.24137931 17.93103448 18.62068966 19.31034483 20.        ]
```

```
#Question no 15:- Create a function `create_identity_matrix` that takes an integer `n` as input and generates a square
#identity matrix of size `n x n` using `numpy.eye`.

#Answer :-

def create_identity_matrix(n):
  return np.eye(n)

n = int(input("Enter the size of the matrix : "))
result = create_identity_matrix(n)

print(result)
```

```
Enter the size of the matrix : 5
[[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.]]
```

```
#Question no 16 :- Write a function that takes a Python list and converts it into a NumPy array.

#Answer:-

def create_array_from_list(lis):
  return np.array(lis)

l = input("Enter numbers with comma seperated : ")
l1 = l.split(",")

lis = []

for i in l1:
  lis.append(int(i))


result = create_array_from_list(lis)
print(result)
```

```
Enter numbers with comma seperated : 1,5,9,6,3,5,7,4,1
[1 5 9 6 3 5 7 4 1]
```

```
#Question no 17 :- Create a NumPy array and demonstrate the use of `numpy.view` to create a new array object with the same data.

#Answer:-

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

view_arr = arr.view()

view_arr[0][2] = 50

print(f"The view array after modification is {view_arr}")
```

```
The view array after modification is [[ 1  2 50  4]
 [ 2  9  3  4]]
```

```
#Question no 18 :- Write a function that takes two NumPy arrays and concatenates them along a specified axis.

#Answer:-

arr1 = np.array([[1,2,3],[4,5,6]])
arr2 = np.array([[7,8,9],[10,11,12]])

print(arr1)
print(arr2)

result = np.concatenate((arr1,arr2),axis = 1)

print(result)
```

```
[[1 2 3]
 [4 5 6]]
[[ 7  8  9]
 [10 11 12]]
[[ 1  2  3  7  8  9]
 [ 4  5  6 10 11 12]]
```

```
#Question no 19:-  Create two NumPy arrays with different shapes and concatenate them horizontally using `numpy.concatenate`.

#Answer:-

arr1 = np.array([[1,2,3],[4,5,6]])
arr2 = np.array([[7,8,9],[10,11,12]])

print(arr1)
print(arr2)

result = np.concatenate((arr1,arr2),axis = 0)

print(result)
```

```
[[1 2 3]
 [4 5 6]]
[[ 7  8  9]
 [10 11 12]]
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

```
#Question no 20:- Write a function that vertically stacks multiple NumPy arrays given as a list.

#Answer:-

def stack_arrays_vertically(array_list):
    return np.vstack(array_list)

array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
array3 = np.array([7, 8, 9])

arrays = [array1, array2, array3]


stacked_array = stack_arrays_vertically(arrays)
print(stacked_array)
```

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

```
#Question no 21:- Write a Python function using NumPy to create an array of integers within a specified range (inclusive)
#with a given step size.

#Answer:-

def int_array(start,stop,step):
  return np.arange(start,stop + 1,step)

start = int(input("Enter a no : "))
stop = int(input("Enter a no : "))
step = int(input("Enter a no : "))
print(int_array(start,stop,step))
```

```
Enter a no : 1
Enter a no : 10
Enter a no : 2
[1 3 5 7 9]
```

```
#Question no 22:-  Write a Python function using NumPy to generate an array of 10 equally spaced values between 0 and 1 (inclusive).

#Answer:-

def generate_arr():
  return np.arange(0,1,0.1)

arr1 = generate_arr()

print(arr1)
```

```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
```

```
#Question no 23:- Write a Python function using NumPy to create an array of 5 logarithmically spaced values between 1 and 1000 (inclusive).

#Answer:-

def log_arr():
  return np.logspace(1,1000,5)

arr1 = log_arr()

print(arr1)
```

```
[1.00000000e+001 5.62341325e+250             inf             inf
             inf]
/usr/local/lib/python3.10/dist-packages/numpy/core/function_base.py:298: RuntimeWarning: overflow encountered in power
  return _nx.power(base, y)
```

```
#Question no 24:- Create a Pandas DataFrame using a NumPy array that contains 5 rows and 3 columns, where the values are random integers bet

#Answer:-

import pandas as pd
arr = np.random.randint(1,100,size = (5,3))

print(arr)

df = pd.DataFrame(arr)
print(df)
```

```
[[75 31 46]
 [59 27 74]
 [52 52  6]
 [86 29 23]
 [39 23 17]]
    0   1   2
0  75  31  46
1  59  27  74
2  52  52   6
3  86  29  23
4  39  23  17
```

```
#Question no 25:-  Write a function that takes a Pandas DataFrame and replaces all negative values in a specific column
#with zeros. Use NumPy operations within the Pandas DataFrame.

#Answer:-
import pandas as pd

def replece_negative(df,column_name):
  df[column_name] = df[column_name].apply(lambda x : 0 if x < 0 else x)
  return df

data = {
    'A': [1, -2, 3, -4, 5],
    'B': [6, 7, -8, 9, -10]
}
df = pd.DataFrame(data)
print("original Data frame :")
print(df)

df1 = replece_negative(df,"A")

print("After replaceing the negative number the data frame is :")
print(df1)
```

```
original Data frame :
   A   B
0  1   6
1 -2   7
2  3  -8
3 -4   9
4  5 -10
After replaceing the negative number the data frame is :
   A   B
0  1   6
1  0   7
2  3  -8
3  0   9
4  5 -10
```

```
#Question no 26:- Access the 3rd element from the given NumPy array
                                  # arr = np.array([10, 20, 30, 40, 50])

#Answer:-

arr = np.array([10, 20, 30, 40, 50])

print(arr[2])
```

```
30
```

```
#Question no 27:- Retrieve the element at index (1, 2) from the 2D NumPy array.
                                  # arr_2d = np.array([[1, 2, 3],
                                                     #[4, 5, 6],
                                                     #[7, 8, 9]])


#Answer:-

arr_2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])

element = arr_2d[1,2]
print(element)
```

```
6
```

```
#Question no 28:- Using boolean indexing, extract elements greater than 5 from the given NumPy array.
                                               # arr = np.array([3, 8, 2, 10, 5, 7])

#Answer:-

arr = np.array([3, 8, 2, 10, 5, 7])
arr[arr > 5]
```

```
array([ 8, 10,  7])
```

```
#Question no 29 :- Perform basic slicing to extract elements from index 2 to 5 (inclusive) from the given NumPy array
                                                # arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])

#Answer:-

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

```
array([3, 4, 5, 6])
```

```
#Question no 30:-  Slice the 2D NumPy array to extract the sub-array `[[2, 3], [5, 6]]` from the given array.
                                                # arr_2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])

#Answer:-

arr_2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
arr_2d[0:2,[1,2]]
```

```
array([[2, 3],
       [5, 6]])
```

```
#Question no 31:- Write a NumPy function to extract elements in specific order from a given 2D array based on indices
#provided in another array.

#Answer:-

def extract_element(arr,row_wise_order,column_wise_order):
  return arr[row_wise_order,column_wise_order]

arr = np.random.randint(1,10,size = (3,3))
print(arr)
row_wise_order = int(input("Enter the row wise order : "))
column_wise_order = int(input("Enter the column wise order : "))

element = extract_element(arr,row_wise_order,column_wise_order)

print(element)
```

```
[[1 5 6]
 [3 6 3]
 [6 4 8]]
Enter the row wise order : 2
Enter the column wise order : 1
4
```

```
#Question no 32:- . Create a NumPy function that filters elements greater than a threshold from a given 1D array using
#boolean indexing.

#Answer:-

def filter(arr,threshold_no):
  return arr[arr > threshold_no]


arr = np.array([1,5,9,7,53,6])
threshold_no = int(input("Enter the no : "))

f = filter(arr,threshold_no)

print(f)
```

```
Enter the no : 6
[ 9  7 53]
```

```
#Question no 33:- Develop a NumPy function that extracts specific elements from a 3D array using indices provided in three
#separate arrays for each dimension.

#Answer:-
import numpy as np

def extract_elements(arr, idx_x, idx_y, idx_z):
    return arr[idx_x, idx_y, idx_z]

arr = np.array([
    [[1, 2, 3], [4, 5, 6], [7, 8, 9]],
    [[10, 11, 12], [13, 14, 15], [16, 17, 18]],
    [[19, 20, 21], [22, 23, 24], [25, 26, 27]]
])


idx_x = np.array([0, 1, 2])
idx_y = np.array([1, 2, 0])
idx_z = np.array([2, 0, 1])


result = extract_elements(arr, idx_x, idx_y, idx_z)
print(result)
```

```
[ 6 16 20]
```

```
#Question no 34:- Write a NumPy function that returns elements from an array where both two conditions are satisfied
#using boolean indexing.

#Answer:-


def filter_elements(arr, condition1, condition2):
    combined_condition = np.logical_and(condition1, condition2)
    return arr[combined_condition]


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


condition1 = arr > 3
condition2 = arr % 2 == 0

filtered_elements = filter_elements(arr, condition1, condition2)

print(filtered_elements)
```

```
[4 6 8]
```

```
#Question no 35:- Create a NumPy function that extracts elements from a 2D array using row and column indices provided
#in separate arrays.

#Answer:-

def extract_elements(arr, row_indices, col_indices):
    return arr[row_indices, col_indices]


arr = np.array([[10, 20, 30], [40, 50, 60], [70, 80, 90]])


row_indices = np.array([0, 1, 2])
col_indices = np.array([2, 1, 0])


extracted_elements = extract_elements(arr, row_indices, col_indices)

print(extracted_elements)
```

```
[30 50 70]
```

```
#Question no 36:- Given an array arr of shape (3, 3), add a scalar value of 5 to each element using NumPy broadcasting.
```

```
#Answer:-

arr = np.random.randint(1,10,(3,3))
print(arr)

arr1 = arr + 5

print(arr1)
```

```
[[2 1 1]
 [4 3 9]
 [7 6 3]]
[[ 7  6  6]
 [ 9  8 14]
 [12 11  8]]
```

```
#Question no 37:-  Consider two arrays arr1 of shape (1, 3) and arr2 of shape (3, 4). Multiply each row of arr2 by the
#corresponding element in arr1 using NumPy broadcasting.

#Answer:-

arr1 = np.random.randint(1,10,(1,3))
print(arr1)

arr2 = np.random.randint(1,10,(3,4))
print(arr2)

arr3 = arr1 @ arr2
print(arr3)
```

```
[[7 7 3]]
[[7 2 5 2]
 [4 1 9 7]
 [5 2 4 6]]
[[ 92  27 110  81]]
```

```
#Question no 38:- Given a 1D array arr1 of shape (1, 4) and a 2D array arr2 of shape (4, 3), add arr1 to each row of arr2 using
#NumPy broadcasting.

#Answer:-

arr1 = np.random.randint(1,10,(4,3))
print(arr1)

arr2 = np.random.randint(1,10,(4,3))
print(arr2)

arr3 = arr1 + arr2
print(arr3)
```

```
[[4 5 8]
 [4 3 3]
 [6 5 9]
 [7 1 1]]
[[1 2 3]
 [2 3 3]
 [9 6 1]
 [1 2 2]]
[[ 5  7 11]
 [ 6  6  6]
 [15 11 10]
 [ 8  3  3]]
```

```
#Question no 39:- Consider two arrays arr1 of shape (3, 1) and arr2 of shape (1, 3). Add these arrays using NumPy
#broadcasting.

#Answer:-

arr1 = np.array([[1], [2], [3]])
arr2 = np.array([[4, 5, 6]])


result = arr1 + arr2

print(result)
```

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

```
#Question no 40:- Given arrays arr1 of shape (2, 3) and arr2 of shape (2, 2), perform multiplication using NumPy
#broadcasting. Handle the shape incompatibility.

#Answer:-


arr1 = np.array([[1, 2, 3], [4, 5, 6]])
arr2 = np.array([[1, 2], [3, 4]])


arr2_reshaped = np.concatenate([arr2, arr2], axis=1)


result = arr1 * arr2_reshaped[:, :3]

print(result)
```

```
[[ 1  4  3]
 [12 20 18]]
```

```
#Question no 41:-  Calculate column wise mean for the given array:
                           # arr = np.array([[1, 2, 3], [4, 5, 6]])

#Answer:-

arr = np.array([[1, 2, 3], [4, 5, 6]])
m = np.mean(arr,axis = 1)
print(m)
```

```
[2. 5.]
```

```
#Question no 42:- Find maximum value in each row of the given array:
                          # arr = np.array([[1, 2, 3], [4, 5, 6]])

#Answer:-

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

```
[3 6]
```

```
#Question no 43:-  For the given array, find indices of maximum value in each column.
                         # arr = np.array([[1, 2, 3], [4, 5, 6]])

#Answer:-

arr = np.array([[1, 2, 3], [4, 5, 6]])
max_indices = np.argmax(arr,axis = 0)
print(max_indices)
```

```
[1 1 1]
```

```
#Question no 44:- For the given array, apply custom function to calculate moving sum along rows.
                        # arr = np.array([[1, 2, 3], [4, 5, 6]])

#Answer:-

def moving_sum(arr, window_size):
    if window_size > arr.shape[1]:
        raise ValueError("Window size must be less than or equal to the number of columns")

    result = np.zeros((arr.shape[0], arr.shape[1] - window_size + 1))


    for i in range(result.shape[1]):
        result[:, i] = np.sum(arr[:, i:i + window_size], axis=1)

    return result


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


window_size = 2
result = moving_sum(arr, window_size)

print(result)
```

```
[[ 3.  5.]
 [ 9. 11.]]
```

```
#Question no 45:-  In the given array, check if all elements in each column are even.
                       # arr = np.array([[2, 4, 6], [3, 5, 7]])

#Answer:-

arr = np.array([[2, 4, 6], [3, 5, 7]])

is_even = arr % 2 == 0

print(is_even)
```

```
[[ True  True  True]
 [False False False]]
```

```
#Question no 46:-  Given a NumPy array arr, reshape it into a matrix of dimensions `m` rows and `n` columns. Return the
#reshaped matrix.                                    # original_array = np.array([1, 2, 3, 4, 5, 6])

#Answer:-
original_array = np.array([1, 2, 3, 4, 5, 6])
reshaped_matrix = original_array.reshape(3,2)

print(reshaped_matrix)
```

```
[[1 2]
 [3 4]
 [5 6]]
```

```
#Question no 47:-  Create a function that takes a matrix as input and returns the flattened array.
                                    # input_matrix = np.array([[1, 2, 3], [4, 5, 6]])

#Answer:-
input_matrix = np.array([[1, 2, 3], [4, 5, 6]])
print(input_matrix.flatten())
```

```
[1 2 3 4 5 6]
```

```
#Question no 48:- Write a function that concatenates two given arrays along a specified axis.
                                  # array1 = np.array([[1, 2], [3, 4]])
                                  # array2 = np.array([[5, 6], [7, 8]])

#Answer:-


def concate_array(arr1,arr2,axis = 0):
  return np.concatenate((arr1, arr2), axis=axis)

arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])

result = concate_array(arr1,arr2)

print(result)
```

```
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

```
#Question no 49:- Create a function that splits an array into multiple sub-arrays along a specified axis.
                                 # original_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

#Answer:-
def array_splits(original_array,num_splits,axis):
  return np.array_split(original_array, num_splits, axis=axis)

original_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
num_splits = int(input("Enter the no you want to split :"))

result = array_splits(original_array,num_splits,axis = 0)
print(result)
```

```
Enter the no you want to split :3
[array([[1, 2, 3]]), array([[4, 5, 6]]), array([[7, 8, 9]])]
```

```
#Question no 50:-  Write a function that inserts and then deletes elements from a given array at specified indices.
                                 # original_array = np.array([1, 2, 3, 4, 5])
                                 #indices_to_insert = [2, 4]
                                 #values_to_insert = [10, 11]
                                 #indices_to_delete = [1, 3]

#Answer:-

def insert_and_delete_elements(original_array, indices_to_insert, values_to_insert, indices_to_delete):
    modified_array = np.insert(original_array, indices_to_insert, values_to_insert)
    modified_array = np.delete(modified_array, sorted(indices_to_delete, reverse=True))
    return modified_array

original_array = np.array([1, 2, 3, 4, 5])
indices_to_insert = [2, 4]
values_to_insert = [10, 11]
indices_to_delete = [1, 3]

result = insert_and_delete_elements(original_array, indices_to_insert, values_to_insert, indices_to_delete)
print(result)
```

```
[ 1 10  4 11  5]
```

```
#Question no 51:- Create a NumPy array `arr1` with random integers and another array `arr2` with integers from 1 to 10.
#Perform element-wise addition between `arr1` and `arr2`.

#Answer:-
arr1 = np.random.randint(1,5,size = 9)
print(arr1)

arr2 = np.arange(1,10,1)
print(arr2)

print("Element wise addition is :")
print(arr1 + arr2)
```

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

```
Element wise addition is :
[ 5  5  7  7  6  8 11 12 12]
```

```
#Question no 52:-  Generate a NumPy array `arr1` with sequential integers from 10 to 1 and another array `arr2` with integers
#from 1 to 10. Subtract `arr2` from `arr1` element-wise.

#Answer:-

arr1 = np.arange(10,1,-1)
print(arr1)
arr2 = np.arange(1,10,1)
print(arr2)

print(arr2 - arr1)
```

```
[10  9  8  7  6  5  4  3  2]
[1 2 3 4 5 6 7 8 9]
[-9 -7 -5 -3 -1  1  3  5  7]
```

```
#Question no 53:- Create a NumPy array `arr1` with random integers and another array `arr2` with integers from 1 to 5.
#Perform element-wise multiplication between `arr1` and `arr2`.

#Answer:-

arr1 = np.random.randint(1,10,4)
print(arr1)

arr2 = np.arange(1,5,1)
print(arr2)

arr1 * arr2
```

```
[7 6 3 9]
[1 2 3 4]
array([ 7, 12,  9, 36])
```

```
#Question no 54:-  Generate a NumPy array `arr1` with even integers from 2 to 10 and another array `arr2` with integers from 1
#to 5. Perform element-wise division of `arr1` by `arr2`.

#Answer:-

arr1 = np.arange(2,10,2)
arr2 = np.arange(1,5,1)
print(arr1)
print(arr2)

arr1/arr2
```

```
[2 4 6 8]
[1 2 3 4]
array([2., 2., 2., 2.])
```

```
#Question no 55:- Create a NumPy array `arr1` with integers from 1 to 5 and another array `arr2` with the same numbers
#reversed. Calculate the exponentiation of `arr1` raised to the power of `arr2` element-wise.

#Answer:-

arr1 = [1,2,3,4,5]
arr2 = [5,4,3,2,1]

np.power(arr1,arr2)
```

```
array([ 1, 16, 27, 16,  5])
```

```
#Question no 56:- Write a function that counts the occurrences of a specific substring within a NumPy array of strings.
                          # arr = np.array(['hello', 'world', 'hello', 'numpy', 'hello'])

#Answer:-
def str_count(arr,s):
  l = list(arr)
  return l.count(s)

arr = np.array(['hello', 'world', 'hello', 'numpy', 'hello'])
s = input("Enter the occurance of the str :")

result = str_count(arr,s)
print(result)
```

```
Enter the occurance of the str :hello
3
```

```
#Question no 57:- Write a function that extracts uppercase characters from a NumPy array of strings.
                                    #arr = np.array(['Hello', 'World', 'OpenAI', 'GPT'])

#Answer:-
def uppercase_let(arr):
  u = []
  for i in arr:
    if i == i.upper():
       u.append(i)
  return u

arr = np.array(['Hello', 'World', 'OpenAI', 'GPT'])
result = uppercase_let(arr)
print(result)
```

```
['GPT']
```

```
#Question no 58:-  Write a function that replaces occurrences of a substring in a NumPy array of strings with a new string.
                                    # arr = np.array(['apple', 'banana', 'grape', 'pineapple'])
#Answer:-

def replace_substring(arr, old_substring, new_substring):
    return np.char.replace(arr, old_substring, new_substring)

arr = np.array(['apple', 'banana', 'grape', 'pineapple'])
old_substring = 'apple'
new_substring = 'orange'

result = replace_substring(arr, old_substring, new_substring)
print(result)
```

```
['orange' 'banana' 'grape' 'pineorange']
```

```
#Question no 59:- Write a function that concatenates strings in a NumPy array element-wise.
                                    # arr1 = np.array(['Hello', 'World'])
                                    # arr2 = np.array(['Open', 'AI'])

#Answer:-

arr1 = np.array(['Hello', 'World'])
arr2 = np.array(['Open', 'AI'])

print("Array 1 : ")
print(arr1)

print("\nArray 2 : ")
print(arr2)

def str_concatenate(arr1, arr2):
    arr3 = []
    for i in range(len(arr1)):
        arr3.append(arr1[i] + arr2[i])
    return np.array(arr3)

result = str_concatenate(arr1,arr2)
print("After concatenates the array is : ")
print(result)
```

```
Array 1 :
['Hello' 'World']

Array 2 :
['Open' 'AI']
After concatenates the array is :
['HelloOpen' 'WorldAI']
```

```
#Question no 60:- Write a function that finds the length of the longest string in a NumPy array.
                                  # arr = np.array(['apple', 'banana', 'grape', 'pineapple'])

#Answer:-

arr = np.array(['apple', 'banana', 'grape', 'pineapple'])

print("The given array is \n",arr)

def str_len(arr):
  length = np.vectorize(len)(arr) # It gives the length of each of the string in the array
  return max(length)

result = str_len(arr)
print("The length of the longest string is")
print(result)
```

```
The given array is
 ['apple' 'banana' 'grape' 'pineapple']
The length of the longest string is
9
```

```
#Question no 61:- Create a dataset of 100 random integers between 1 and 1000. Compute the mean, median, variance, and
#standard deviation of the dataset using NumPy's functions.

#Answer:-

arr = np.random.randint(1,1000, size = 100)

print("Array is\n")
print(arr)

me = np.mean(arr)
med = np.median(arr)
var = np.var(arr)
st = np.std(arr)

print("Mean is ",me)
print("Median is ",med)
print("Variance is",var)
print("Standerd deviation is ",st)
```

```
Array is

[211 387 852 564 601 572 770 907 914 991 176 877 495 401 724 482 170 738
 103 154 180 624 562 623 537 984 268 301 556 621 285 523 398 718 107  70
 700  40 355 336 219 686 448 205 669 206 539 304 443 178 561 696 686 480
 774 920 287  74 313 364 793 172 619 668 582 886 641 239 108 441 961 729
 994 667 893 679 284 772 170 435  21 390 829 359 292 170 346 840 287 787
 171 518 333 339 724 513 261 559 488 728]
Mean is  496.17
Median is  504.0
Variance is 66488.9211
Standerd deviation is  257.8544572040592
```

```
#Question no 62 :- Generate an array of 50 random numbers between 1 and 100. Find the 25th and 75th percentiles of the dataset.

#Answer:-

arr = np.random.randint(1,100, size = 50)

# 25th percentiles of the array

percentile_25 = np.percentile(arr,25)
print("The 25th percentile of the given array is ",percentile_25)


# 75th percentiles of the array

percentile_75 = np.percentile(arr,75)
print("The 25th percentile of the given array is ",percentile_75)
```

```
The 25th percentile of the given array is  28.5
The 25th percentile of the given array is  74.25
```

```
#Question no 63 :-  Create two arrays representing two sets of variables. Compute the correlation coefficient between these
# arrays using NumPy's `corrcoef` function.

#Answer:-

arr1 = np.random.randint(1,10,(3,3))
arr2 = np.random.randint(10,20,(3,3))

coerelation = np.corrcoef(arr1,arr2)

print(f"The coerelation coefficent of the two array \n{arr1} and \n{arr2} is \n{coerelation}")
```

```
The coerelation coefficent of the two array
[[8 4 6]
 [9 5 6]
 [1 7 3]] and
[[10 10 15]
 [19 10 10]
 [13 12 10]] is
[[ 1.          0.96076892 -0.98198051  0.          0.8660254   0.32732684]
 [ 0.96076892  1.         -0.89104211 -0.2773501   0.97072534  0.57655666]
 [-0.98198051 -0.89104211  1.         -0.18898224 -0.75592895 -0.14285714]
 [ 0.         -0.2773501  -0.18898224  1.         -0.5        -0.94491118]
 [ 0.8660254   0.97072534 -0.75592895 -0.5         1.          0.75592895]
 [ 0.32732684  0.57655666 -0.14285714 -0.94491118  0.75592895  1.        ]]
```

```
#Question no 64:-  Create two matrices and perform matrix multiplication using NumPy's `dot` function.

#Answer:-

arr1 = np.random.randint(1,10,(3,3))
arr2 = np.random.randint(10,20,(3,3))

mul = np.dot(arr1,arr2)

print(f"The dot miltiplication of the given two array \n{arr1} and \n{arr2} is : \n")
print(mul)
```

```
The dot miltiplication of the given two array
[[4 7 6]
 [1 7 5]
 [8 7 7]] and
[[14 13 15]
 [16 16 17]
 [19 13 15]] is :

[[282 242 269]
 [221 190 209]
 [357 307 344]]
```

```
#Question no 65:- Create an array of 50 integers between 10 and 1000. Calculate the 10th, 50th (median), and 90th
#percentiles along with the first and third quartiles.

#Answer:-

arr = np.random.randint(10, 1001, size=50)

percentiles = {
    '10th percentile': np.percentile(arr, 10),
    '25th percentile (1st quartile)': np.percentile(arr, 25),
    '50th percentile (median)': np.percentile(arr, 50),
    '75th percentile (3rd quartile)': np.percentile(arr, 75),
    '90th percentile': np.percentile(arr, 90)
}

print("Array:", arr)
print("\nPercentiles and Quartiles:")
for key, value in percentiles.items():
    print(f"{key}: {value:.2f}")
```

```
Array: [195 198 812 584 281 501 952 635 217 110 331  66  97 180 477 216 373 785
 952 619 358  41 132 378 458  20 411 242 338 521 253 726  11 380 802 814
 562 938 400 440 128  19 800 176 929 313 195 114 637 515]

Percentiles and Quartiles:
10th percentile: 93.90
25th percentile (1st quartile): 195.00
50th percentile (median): 375.50
75th percentile (3rd quartile): 610.25
90th percentile: 812.20
```

```
#Question no 66:- Create a NumPy array of integers and find the index of a specific element.

#Answer:-

arr = np.array([[1,0,9],[7,11,3],[4,13,6]])

index = np.where(arr == 11)
print(f"The index of the specified element is : {index}")
```

```
The index of the specified element is : (array([1]), array([1]))
```

```
#Question no 67 :- Generate a random NumPy array and sort it in ascending order.

#Answer:-

arr = np.random.randint(1,100,(5,5))
print(f"The array is \n{arr}")

arr1 = np.sort(arr)

print("\nThe sorted array is\n")
print(arr1)
```

```
The array is
[[86 31 33 44  5]
 [94 25 66 68 95]
 [65 67 77 38  6]
 [ 9 95 88 95 54]
 [94 46 84 65 57]]

The sorted array is

[[ 5 31 33 44 86]
 [25 66 68 94 95]
 [ 6 38 65 67 77]
 [ 9 54 88 95 95]
 [46 57 65 84 94]]
```

```
#Question no 68 :- Filter elements >20  in the given NumPy array.
                # arr = np.array([12, 25, 6, 42, 8, 30])

#Answer:-

arr = np.array([12, 25, 6, 42, 8, 30])

element = arr > 20
arr1 = arr[element]

print("After filtering the array:\n")
print(arr1)
```

```
After filtering the array:

[25 42 30]
```

```
#Question no 69 :-Filter elements which are divisible by 3 from a given NumPy array.
                                # arr = np.array([1, 5, 8, 12, 15])

#Answer:-

arr = np.array([1, 5, 8, 12, 15])

element = arr % 3 == 0

arr1 = arr[element]

print("After filtering the array:\n")
print(arr1)
```

```
After filtering the array:

[12 15]
```

```
#Question no 70 :- Filter elements which are ≥ 20 and ≤ 40 from a given NumPy array.
                                          # arr = np.array([10, 20, 30, 40, 50])

#Answer:-

arr = np.array([10, 20, 30, 40, 50])

arr1 = arr[(arr >= 20) & (arr <= 40)]

print("After filtering the array:\n")
print(arr1)
```

```
After filtering the array:

[20 30 40]
```

```
#Question no 71:- For the given NumPy array, check its byte order using the `dtype` attribute byteorder.
                                 # arr = np.array([1, 2, 3])

#Answer:-

arr = np.array([1, 2, 3])
byte_order = arr.dtype.byteorder

print("Byte order of the array:", byte_order)
```

```
Byte order of the array: =
```

```
#Question no 72:- For the given NumPy array, perform byte swapping in place using `byteswap()`.
                                # arr = np.array([1, 2, 3], dtype=np.int32)

#Answer:-

arr = np.array([1, 2, 3], dtype=np.int32)
arr_swap = arr.byteswap()

print(arr)
print(arr_swap)
```

```
[1 2 3]
[16777216 33554432 50331648]
```

```
#Question no 73:-  For the given NumPy array, swap its byte order without modifying the original array using `newbyteorder()`.
                                # arr = np.array([1, 2, 3], dtype=np.int32)
#Answer:-

arr = np.array([1, 2, 3], dtype=np.int32)
arr_swapped = arr.newbyteorder()

print(arr)
print(arr_swapped)
```

```
[1 2 3]
[16777216 33554432 50331648]
```

```
#Question no 74:- For the given NumPy array and swap its byte order conditionally based on system endianness using `newbyteorder()`.
                                            # arr = np.array([1, 2, 3], dtype=np.int32)

#Answer:-

arr = np.array([1, 2, 3], dtype=np.int32)



#Question no 75:- For the given NumPy array, check if byte swapping is necessary for the current system using `dtype` attribute `byteorder`.
                                             # arr = np.array([1, 2, 3], dtype=np.int32)

#Answer:-

arr = np.array([1, 2, 3], dtype=np.int32)

byte_order = arr.dtype.byteorder

if byte_order == '=':
    print("Byte swapping is not necessary")
else:
    print(f"Byte swapping is necessary and byte order is {byte_order})")
```

```
Byte swapping is not necessary
```

```
#Question no 76:- Create a NumPy array `arr1` with values from 1 to 10. Create a copy of `arr1` named `copy_arr` and modify
#an element in `copy_arr`. Check if modifying `copy_arr` affects `arr1`.

#Answer:-

arr1 = np.arange(1,10).reshape(3,3)
print("Original array\n")
print(arr1)

copy_arr = arr1.copy()
print("\ncopied array")
print(copy_arr)

copy_arr[1][2] = 325

if np.any(arr1 > 10):
  print(f"\nModifying the copy_arr affects the original array\n")
  print(arr1)
else:
  print(f"\nModifying the copy_arr does not affects the original array\n")
  print(arr1)
```

```
Original array

    [[1 2 3]
     [4 5 6]
     [7 8 9]]

    copied array
    [[1 2 3]
     [4 5 6]
     [7 8 9]]

    Modifying the copy_arr does not affects the original array

    [[1 2 3]
```

```
       [4 5 6]
       [7 8 9]]
```

```
#Question no 77:- Create a 2D NumPy array `matrix` of shape (3, 3) with random integers. Extract a slice `view_slice` from
#the matrix. Modify an element in `view_slice` and observe if it changes the original `matrix`.

#Answer:-

import numpy.matlib as nm

matrix = nm.random.randint(1,10,(3,3))
print("Matrix is \n")
print(matrix)

view_slice = matrix[0:1,[0,2]]
print("\nslice view \n")
print(view_slice)

view_slice[0][1] = 500

if nm.any(matrix > 10):
  print(f"\nModify an element in `view_slice` changes the original `matrix` and matrix is \n{matrix}")
else:
  print(f"\nModify an element in `view_slice` does not changes the original `matrix` and matrix is \n{matrix}")
```

```
Matrix is

    [[6 6 1]
     [1 8 7]
     [8 4 8]]

    slice view

    [[6 1]]

    Modify an element in `view_slice` does not changes the original `matrix` and matrix is
    [[6 6 1]
     [1 8 7]
     [8 4 8]]
```

```
#Question no 78:- Create a NumPy array `array_a` of shape (4, 3) with sequential integers from 1 to 12. Extract a slice
#`view_b` from `array_a` and broadcast the addition of 5 to view_b. Check if it alters the original `array_a`.

#Answer:-

array_a = np.arange(1, 13).reshape(4, 3)

print("array_a:")
print(array_a)

view_b = array_a[0:3, [0, 2]]
print("\nview_b:")
print(view_b)

view_c = view_b + 5
print("\nview_c:")
print(view_c)

if np.any(array_a > 12):
    print(f"\nAddition of 5 affects the original array_a, and the original array is \n{array_a}")
else:
    print(f"\nAddition of 5 does not affect the original array_a, and the original array is \n{array_a}")
```

```
array_a:
    [[ 1  2  3]
     [ 4  5  6]
     [ 7  8  9]
     [10 11 12]]

    view_b:
    [[1 3]
     [4 6]
     [7 9]]

    view_c:
    [[ 6  8]
     [ 9 11]
```

```
 [12 14]]

Addition of 5 does not affect the original array_a, and the original array is
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

```
#Question no 79:- Create a NumPy array `orig_array` of shape (2, 4) with values from 1 to 8. Create a reshaped view
#`reshaped_view` of shape (4, 2) from orig_array. Modify an element in `reshaped_view` and check if it
#reflects changes in the original `orig_array

#Answer:-

orig_array = np.random.randint(1,8,(2,4))
print(f"The original array is \n{orig_array}")
# reshaping the original array

reshaped_view = orig_array.reshape(4,2)
print(f"After reshape the reshaped view is \n{reshaped_view}")

# Modify the elements of the reshaped_view

reshaped_view[0][0] = 100

if np.any(reshaped_view > 8):
  print(f"After modification the original array affects and the array is \n{orig_array}")
else:
  print(f"After modification the original array is not effects and the array is \n{orig_array}")
```

```
The original array is
[[4 5 1 2]
 [6 6 7 2]]
After reshape the reshaped view is
[[4 5]
 [1 2]
 [6 6]
 [7 2]]
After modification the original array affects and the array is
[[100   5   1   2]
 [  6   6   7   2]]
```

```
#Question no 80:- Create a NumPy array `data` of shape (3, 4) with random integers. Extract a copy `data_copy` of
#elements greater than 5. Modify an element in `data_copy` and verify if it affects the original `data`.

#Answer :-

#creating an random numpy array

data = np.random.randint(1,10,(3,4))

print(data)

#Extract the elements of the array from data which satisfies the condition > 5.

data_copy = data[data > 5]
data_copy[0] = 50

print(data_copy)


if np.any(data > 10):
  print("Modifying the data copy affects the original data")
  print(data)
else:
  print("Modifying the data does not affects the original data")
  print(data)
```

```
[[1 5 4 3]
 [4 4 9 5]
 [8 5 4 4]]
[50  8]
Modifying the data does not affects the original data
[[1 5 4 3]
 [4 4 9 5]
 [8 5 4 4]]
```

```
#Question no 81:- Create two matrices A and B of identical shape containing integers and perform addition and subtraction operations between

#Answer :-

A = np.random.randint(1,10,(3,3))
B = np.random.randint(10,20,(3,3))

sum = A + B
subtraction = A - B

print(f"The addition of the matrix \n{A} \nand \n{B} \nis \n{sum}")
print(f"The subtraction of the matrix \n{A} \nand \n{B} \nis \n{subtraction}")
```

```
The addition of the matrix
[[6 3 9]
 [4 7 5]
 [6 5 9]]
and
[[13 13 14]
 [17 11 17]
 [18 14 16]]
is
[[19 16 23]
 [21 18 22]
 [24 19 25]]
The subtraction of the matrix
[[6 3 9]
 [4 7 5]
 [6 5 9]]
and
[[13 13 14]
 [17 11 17]
 [18 14 16]]
is
[[ -7 -10  -5]
 [-13  -4 -12]
 [-12  -9  -7]]
```

```
#Question no 82:- Generate two matrices `C` (3x2) and `D` (2x4) and perform matrix multiplication.

#Answer: -

#Input frist matrix from user.

row1 = int(input("Enter the no of row of the matrix : "))
col1 = int(input("Entert the column of the matrix : "))

print("Please enter the element in row wise")

matrix = []

for i in range(row1):
  for j in range(col1):
    element = int(input(f"Enter the [{i+1}][{j+1}] element of the matrix : "))
    matrix.append(element)

given_matrix = np.array(matrix)
C = given_matrix.reshape(row1,-1)

print(C) #Printing the entered matrix

#Input second matrix from user.

row2 = int(input("Enter the no of row of the matrix : "))
col2 = int(input("Entert the column of the matrix : "))

print("Please enter the element in row wise")

matrix = []

for i in range(row2):
  for j in range(col2):
    element = int(input(f"Enter the [{i+1}][{j+1}] element of the matrix : "))
    matrix.append(element)

given_matrix = np.array(matrix)
D = given_matrix.reshape(row2,-1)

print(D) #Printing the entered matrix

#Multiply the two matrix

if col1 == row2:
  mul = C @ D
  print(f"After multiplication the matrix is {mul}")
else:
  print("Multiplication is not possible for the given matrix")
```

```
Enter the no of row of the matrix : 3
Entert the column of the matrix : 2
Please enter the element in row wise
Enter the [1][1] element of the matrix : 1
Enter the [1][2] element of the matrix : 2
Enter the [2][1] element of the matrix : 3
Enter the [2][2] element of the matrix : 4
Enter the [3][1] element of the matrix : 5
Enter the [3][2] element of the matrix : 6
[[1 2]
 [3 4]
 [5 6]]
Enter the no of row of the matrix : 2
Entert the column of the matrix : 4
Please enter the element in row wise
Enter the [1][1] element of the matrix : 9
Enter the [1][2] element of the matrix : 8
Enter the [1][3] element of the matrix : 7
Enter the [1][4] element of the matrix : 6
Enter the [2][1] element of the matrix : 5
Enter the [2][2] element of the matrix : 4
Enter the [2][3] element of the matrix : 2
Enter the [2][4] element of the matrix : 1
[[9 8 7 6]
 [5 4 2 1]]
After multiplication the matrix is [[19 16 11  8]
 [47 40 29 22]
 [75 64 47 36]]
```

```
#Question no 83:-  Create a matrix `E` and find its transpose.

#Answer :-
```