```
#ques 1 Create a 3x3 NumPy array with random integers between 1 and 100. Then, interchange its rows and columns.
import numpy as np
a=np.random.randint(1,100,(3,3))
print("3-d array",a)

print("transpose",a.T)
```

```
3-d array [[49  9 95]
 [ 5 94 33]
 [85 51 82]]
transpose [[49  5 85]
 [ 9 94 51]
 [95 33 82]]
```

```
#ques 2.Generate a 1D NumPy array with 10 elements. Reshape it into a 2x5 array, then into a 5x2 array
a=np.arange(10)
print("1-d array",a)
```

```
1-d array [0 1 2 3 4 5 6 7 8 9]
```

```
#reshaping
b=a.reshape(2,5)
print("2-d array",b)
c=a.reshape(5,2)
print(c)
```

```
2-d array [[0 1 2 3 4]
 [5 6 7 8 9]]
[[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]
```

```
#ques 3. Create a 4x4 NumPy array with random float values. Add a border of zeros around it, resulting in a 6x6 array.
a=np.random.rand(4,4)
print(a)
```

```
[[0.36235989 0.62136732 0.92004971 0.41647902]
 [0.51115288 0.57295385 0.89888009 0.10641482]
 [0.50097118 0.41559967 0.3888847  0.3039621 ]
 [0.12435484 0.42911944 0.21092817 0.31721894]]
```

```
b=np.pad(a,pad_width=1)
print(b)
```

```
[[0.         0.         0.         0.         0.         0.        ]
 [0.         0.36235989 0.62136732 0.92004971 0.41647902 0.        ]
 [0.         0.51115288 0.57295385 0.89888009 0.10641482 0.        ]
 [0.         0.50097118 0.41559967 0.3888847  0.3039621  0.        ]
 [0.         0.12435484 0.42911944 0.21092817 0.31721894 0.        ]
 [0.         0.         0.         0.         0.         0.        ]]
```

```
#ques 4. Using NumPy, create an array of integers from 10 to 60 with a step of 5.
a=np.arange(10,60,5)
print(a)
```

```
[10 15 20 25 30 35 40 45 50 55]
```

```
#ques 5. Create a NumPy array of strings ['python', 'numpy', 'pandas']. Apply different case transformations
 #(uppercase, lowercase, title case, etc.) to each element.
d=np.array(["python","numpy","pandas"])
np.char.upper(d)
```

```
array(['PYTHON', 'NUMPY', 'PANDAS'], dtype='<U6')
```

```
np.char.lower(d)
```

```
array(['python', 'numpy', 'pandas'], dtype='<U6')
```

```
np.char.title(d)
```

```
array(['Python', 'Numpy', 'Pandas'], dtype='<U6')
```

```
#ques 6. Generate a NumPy array of words. Insert a space between each character of every word in the array.
import numpy as np

# Step 1: Create a NumPy array of words
w= np.array(['mansi', 'student', 'at', 'mdu'])

# Step 2: Define a function to insert spaces between characters of a word
def temp(word):
    return ' '.join(word)

# Step 3: Apply the function to each word in the array
spaced = np.vectorize(temp)(w)

print("Original array of words:")
print(w)

print("\nArray with spaces inserted between characters:")
print(spaced)
```

```
Original array of words:
['mansi' 'student' 'at' 'mdu']

Array with spaces inserted between characters:
['m a n s i' 's t u d e n t' 'a t' 'm d u']
```

```
#ques 7.  Create two 2D NumPy arrays and perform element-wise addition, subtraction, multiplication, and division.
arr1=np.random.randint(1,3,(2,2))
arr2=np.random.randint(1,3,(2,2))
print(arr1)
print(arr2)
```

```
[[2 2]
 [2 2]]
[[1 1]
 [1 1]]
```

```
arr1+arr2
```

```
array([[3, 3],
       [3, 3]])
```

```
arr1-arr2
```

```
array([[1, 1],
       [1, 1]])
```

```
arr1*arr2
```

```
array([[2, 2],
       [2, 2]])
```

```
arr1/arr2
```

```
array([[2., 2.],
       [2., 2.]])
```

```
#ques 8  Use NumPy to create a 5x5 identity matrix, then extract its diagonal elements
a=np.eye(5)
b=np.diag(a)
print(a)
print("diagonal",b)
```

```
[[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.]]
diagonal [1. 1. 1. 1. 1.]
```

```
#ques 9. Generate a NumPy array of 100 random integers between 0 and 1000. Find and display all prime numbers in this array.
import numpy as np
```

```
# Step 1: Generate a NumPy array of 100 random integers between 0 and 1000
s = np.random.randint(0, 1001, size=100)

print("Array of random integers:")
print(s)

# Step 2: Define a function to check for prime numbers
def is_prime(n):
    """Return True if n is a prime number, else False."""
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

# Step 3: Apply the function to filter out the prime numbers
prime_numbers = np.array([num for num in s if is_prime(num)])

print("\nPrime numbers in the array:")
print(prime_numbers)
```

```
Array of random integers:
[ 203  324  774  964   47  639  131  972  868  180 1000  846  143  660
  227  954  791  719  909  373  853  560  305  581  169  675  448   95
  197  606  256  881  690  292  930  816  861  387  610  554  973  368
  999  917  201  383  512  906  370  555  954  383   23  699  130  377
   98  574  931  734  123  963  594  942  739  148  209  562  411  782
   41   58  705   36  778   86   43  872   11  770  307   80   32  182
  128  806  275  174  554  371  184  444  488  589  286  280  637  770
  515   94]

Prime numbers in the array:
[ 47 131 227 719 373 853 197 881 383 383  23 739  41  43  11 307]
```

```
#ques 10. Create a NumPy array representing daily temperatures for a month. Calculate and display the weekly averages
# Step 1: Create a NumPy array of daily temperatures for a month (30 days)
s = np.random.randint(0, 35, size=28)  # Random temperatures between 0 and 34

print("Daily temperatures for the month:")
print(s)

# Step 2: Reshape the array into a 2D array with 4 rows (weeks) and 7 columns (days)
r = s.reshape(4, 7)

print("\nWeekly temperatures:")
print(r)

# Step 3: Calculate the weekly averages
weekly_averages = np.mean(r, axis=1)

print("\nWeekly averages:")
print(weekly_averages)
```

```
Daily temperatures for the month:
[19 14 32  1  9 32 31 10 23 11 28 34  0  0  5 17 15  4 31  1  1 11 18 27
  0 14 12 20]

Weekly temperatures:
[[19 14 32  1  9 32 31]
 [10 23 11 28 34  0  0]
 [ 5 17 15  4 31  1  1]
 [11 18 27  0 14 12 20]]

Weekly averages:
[19.71428571 15.14285714 10.57142857 14.57142857]
```