```python
import numpy as np

print(np.__version__)
```

```
1.26.2
```

```python
arr = np.array((1, 2, 3, 4))
print(arr, arr.ndim)
print(arr.dtype)
```

```
[1 2 3 4] 1
int32
```

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

print(arr[1, 2], arr[0, -2])
```

```
7 3
```

```python
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
print(arr[1::2])
# good for printing elements with gaps
```

```
[2 4 6 8]
```

```python
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

i = 0
while i < 2 :
    print(arr[i, 1:4])
    i+=1
```

```
[2 3 4]
[7 8 9]
```

```python
# data types in numpy

# i - integer
# b - boolean
# u - unsigned integer
# f - float
# c - complex float
# m - timedelta
# M - datetime
# O - object
# S - string
# U - unicode string
# V - fixed chunk of memory for other type ( void )

arr = np.array([1, 2, 3, 4], dtype = 'S')
print(arr)
```

```python
arr1 = np.array([1, 2, 3, 4], dtype = 'i8')
print(arr1, arr1.dtype)
```

```
[b'1' b'2' b'3' b'4']
[1 2 3 4] int64
```

```python
A = np.array([1.1, 2.5, 45.7879])
B = A.astype(int)
print(B)
```

```
[ 1  2 45]
```

```python
A = np.array([1, 2, 3, 4, 5])
B = A.view() # its the original array any change to it will change the original array
C = A.copy() # its a copy does not change the original array

C[0] = 500
print(C)
print(A)

B[0] = 50
print(B)
print(A)
```

```
[500   2   3   4   5]
[1 2 3 4 5]
[50  2  3  4  5]
[50  2  3  4  5]
```

```python
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
newarr = arr.reshape(-1)
print(newarr)
```

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

```python
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
# method - 1
for i in arr:
    for j in i:
        print(j)

# method - 2
for i in np.nditer(arr):
    print(i)
```

```
1
2
3
4
5
```

```
6
7
8
9
10
1
2
3
4
5
6
7
8
9
10
```

```
import numpy as np

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

arr2 = np.array([[5, 6], [7, 8]])

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

print(arr)
```

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

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
arr1 = np.array_split(arr, 5)
print(arr1[2], arr1[4])
```

```
[5 6] [8]
```

```
arr1 = np.array([[1, 2], [3, 1]])
ans = np.where(arr1 == 1)
print(ans)
```

```
(array([0, 1], dtype=int64), array([0, 1], dtype=int64))
```

```
# Search Sorted
# There is a method called searchsorted() which performs a binary
search in the array, and returns the index where the specified value
would be inserted to maintain the search order.

A = np.array([3, 4, 6, 7, 34, 878, 999])
x = np.searchsorted(A, 20)
print(x)
```

```
4
```

```
#sorting 2-d array

arr = np.array([[3, 2, 4], [5, 0, 1]])
print(np.sort(arr)) # will sort every row
```

```
[[2 3 4]
 [0 1 5]]
```

```
x = [1, 2, 3, 4]
y = [4, 5, 6, 7]
z = []
z1 = np.add(x, y)

for i in range(4):
    z.append(x[i] + y[i])

print(z, z1)
```

```
[5, 7, 9, 11] [ 5  7  9 11]
```

```
# Arithmetic operation on two arrays

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([20, 21, 22, 23, 24, 25])

arr3 = np.subtract(arr1, arr2)
print(arr3)

# add, multiply, divide, power, abs, mod, floor, ceil, truncate,
roundoff, around, fix and others also
```

```
[-10  -1   8  17  26  35]
```

```
X = np.arange(1, 101) #awesome thing🎂🎂🎂🎂🎂
print(X)
```

```
[  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  28  29  30  31  32  33  34  35
36
  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
72
  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
90
  91  92  93  94  95  96  97  98  99 100]
```

```
#cummulative sum -> good for prefix sum and suffix sum
arr = np.array([1, 2, 3])
```

```
newarr = np.cumsum(arr)
print(newarr)
```

```
[1 3 6]
```

```
arr = np.array([20, 8, 32, 36, 16])

x = np.gcd.reduce(arr) # for lcm also
y = np.gcd(arr[1], arr[4]) # reduce will find for whole array
print(x, y)
```

```
4 8
```

```
arr = np.array([1, 1, 1, 2, 3, 4, 5, 5, 6, 7]) # similar to set in c++
x = np.unique(arr)
```