Create a list and convert it into array

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

In [2]:
```
my_list = [1, 2, 3, 4, 5]
my_arr = np.array(my_list)
my_arr
```

Out[2]:
```
array([1, 2, 3, 4, 5])
```

A single dimension array is created, single beacuse open and close by one 3rd bracket []

Now lets see types of my_arr

In [3]:
```
type(my_arr)
```

Out[3]:
```
numpy.ndarray
```

"nd" stands for n-dimensional array (1D, 2D, 3D, etc.)

In [4]:
```
my_arr.shape
```

Out[4]:
```
(5,)
```

this is 1D array thats why there is a single number (5)

In [5]:
```
list_1 = [1,2,3,4,5]
list_2 = [4,5,6,7,8]
list_3 = [7,8,9,3,1]

my_arr_1 = np.array([list_1,list_2,list_3])
my_arr_1
```

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

A 2d array is created, 2d beacuse open and close by two 3rd bracket []

In [6]:
```
my_arr_1.shape
```

Out[6]:
```
(3, 5)
```

this is 2d array thats why there is a double number (3,3)

Now we learn indexing

In [7]:
```
my_arr
```

Out[7]:
```
array([1, 2, 3, 4, 5])
```

In [8]:
```
my_arr[2]
```

Out[8]:
```
3
```

Now we change the element

In [9]:
```
my_arr[4]=9
my_arr
```

Out[9]:
```
array([1, 2, 3, 4, 9])
```

Another technique of indexing

In [10]:
```
my_arr[2:]
```

Out[10]:
```
array([3, 4, 9])
```

We can also fix the range, we have already learned these concepts

In [11]:
```
my_arr[2:4]
```

Out[11]:
```
array([3, 4])
```

we can also retrieve the last element

In [12]:
```
my_arr[-1]
```

Out[12]:
```
9
```

we can also retrieve whole array except last element

In [13]:
```
my_arr[:-1]
```

Out[13]:
```
array([1, 2, 3, 4])
```

here : is like not(!)

Now we use the double colon

In [14]:
```
my_arr[::-1]
```

Out[14]:
```
array([9, 4, 3, 2, 1])
```

My array is reversed

Now what ahppend if I use -2

In [15]:
```
my_arr[::-2]
```

Out[15]:
```
array([9, 3, 1])
```

It is reversed and jump over 2 numbers

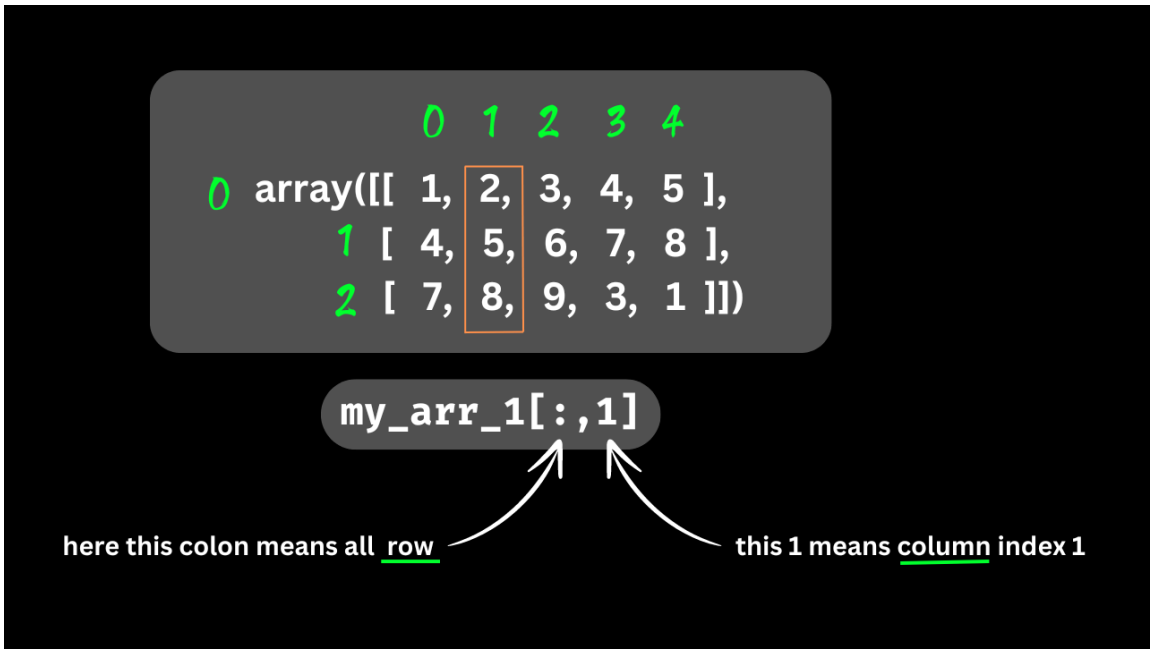
Now we do multi-dimensional indexing

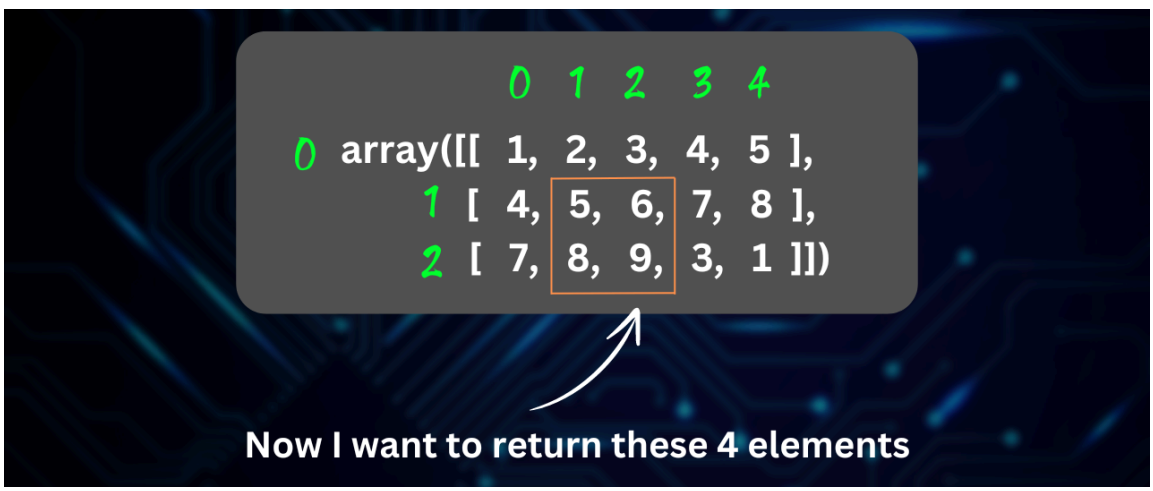
In [16]:
```
my_arr_1
```

Out[16]:
```
array([[1, 2, 3, 4, 5],
       [4, 5, 6, 7, 8],
       [7, 8, 9, 3, 1]])
```

In [17]:
```
my_arr_1[:,1]
```

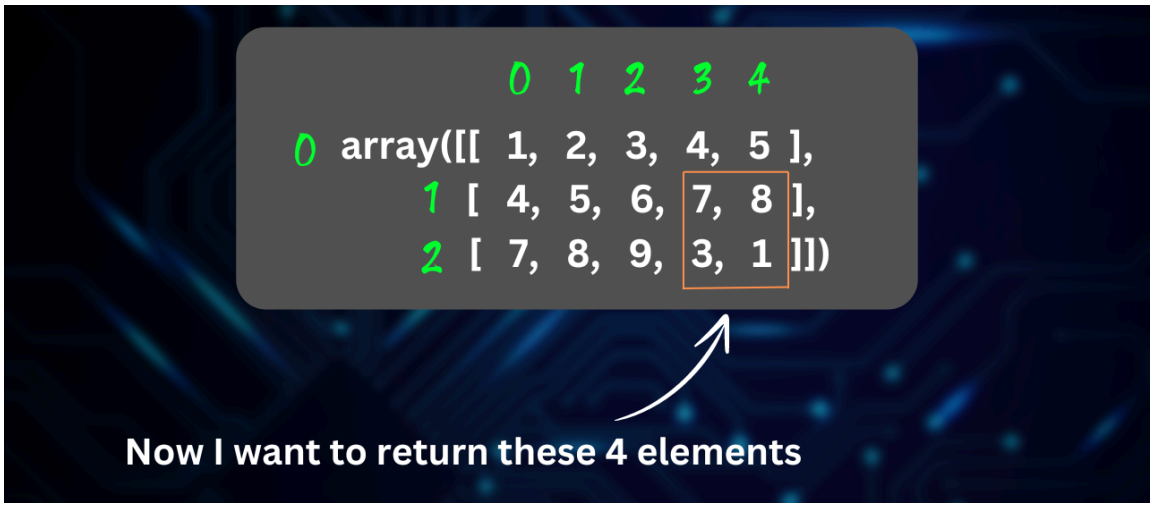
Out[17]:
```
array([2, 5, 8])
```





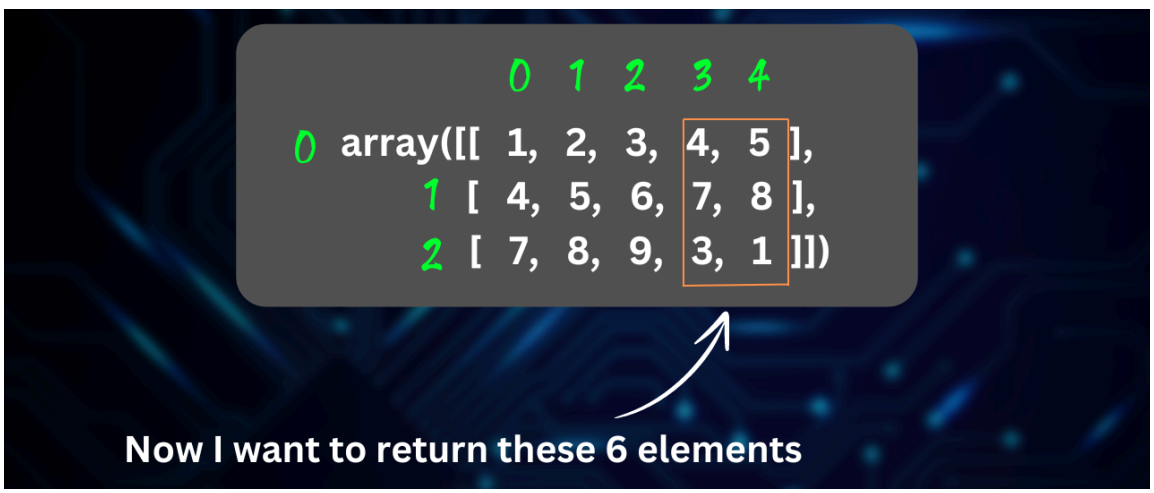
In [18]:
```
my_arr_1[1:,1:3]
```

Out[18]:
```
array([[5, 6],
       [8, 9]])
```



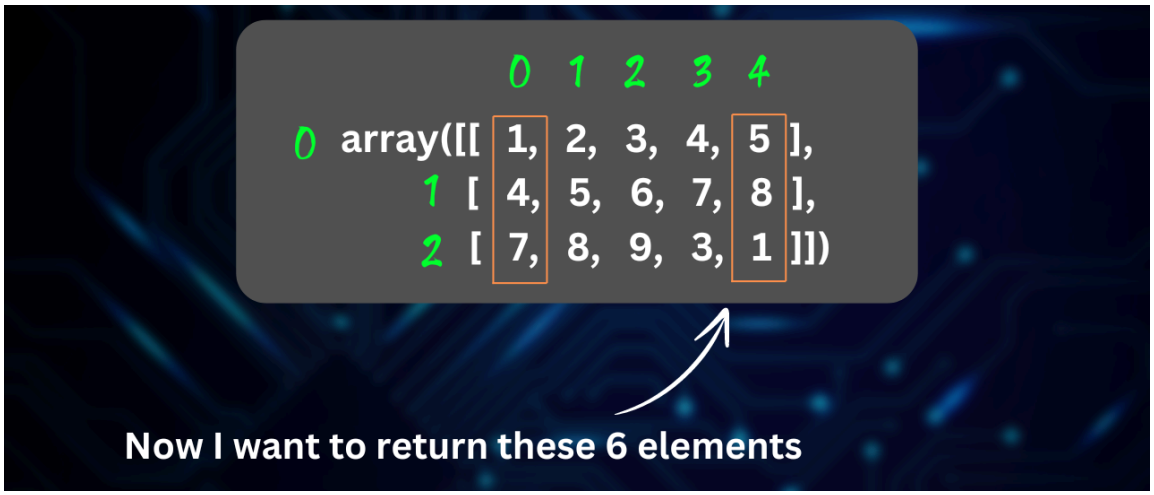
In [19]:
```
my_arr_1[1:,3:]
```

Out[19]:
```
array([[7, 8],
       [3, 1]])
```



In [20]:
```
my_arr_1[:,3:]
```

Out[20]:
```
array([[4, 5],
       [7, 8],
       [3, 1]])
```



Method 1

In [21]:
```
my_arr_1[0:,[0,4]]
```

Out[21]:
```
array([[1, 5],
       [4, 8],
       [7, 1]])
```

Method 2

In [22]:
```
my_arr_1[0:,[0,-1]]
```

Out[22]:
```
array([[1, 5],
       [4, 8],
       [7, 1]])
```

Exploratory Data Analysis (EDA) point of view

In [23]:
```
my_arr
```

Out[23]:
```
array([1, 2, 3, 4, 9])
```

In [24]:
```
my_arr<3
```

Out[24]:
```
array([ True,  True, False, False, False])
```

we can use this for indexing

In [25]:
```
my_arr[my_arr<3]
```

Out[25]:
```
array([1, 2])
```

We dont have to use for loop here, thats why numpy is so helpful & easy

We can also reshape the array

In [26]:
```
my_arr_1
```

Out[26]:
```
array([[1, 2, 3, 4, 5],
       [4, 5, 6, 7, 8],
       [7, 8, 9, 3, 1]])
```

In [27]:
```
my_arr_1.shape
```

Out[27]:
```
(3, 5)
```

In [28]:
```
my_arr_1.reshape(5,3)
```

Out[28]:
```
array([[1, 2, 3],
       [4, 5, 4],
       [5, 6, 7],
       [8, 7, 8],
       [9, 3, 1]])
```

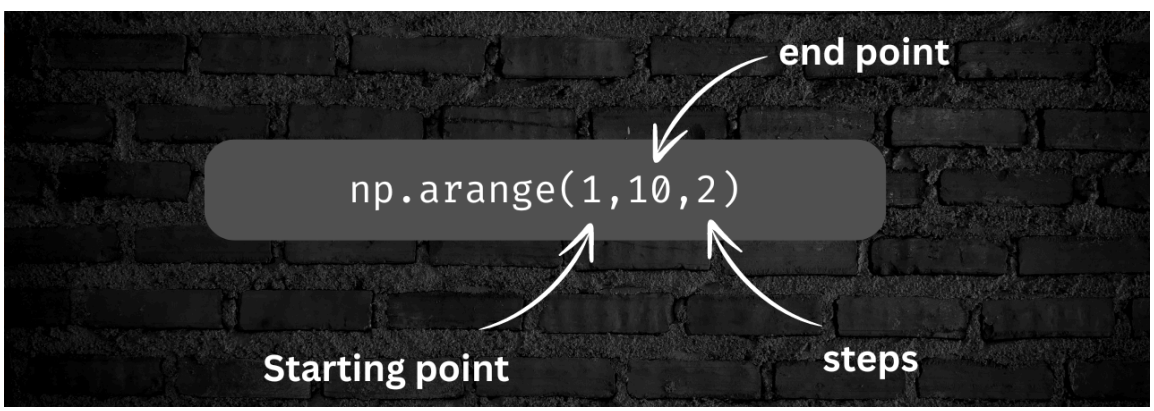
But we have to reshape where total number of element are same(15)

here it has 4 ways - (5,3) (3,5) (15,1) (1,15)

Mechanism to create an array

In [29]:
```
np.arange(1,10,2)
```

Out[29]:
```
array([1, 3, 5, 7, 9])
```



Now I reshape the new created array

In [30]:
```
np.arange(1,20,2).reshape(2,5)
```

Out[30]:
```
array([[ 1,  3,  5,  7,  9],
       [11, 13, 15, 17, 19]])
```

or we can do this:

In [31]:
```
np.arange(1,20,2).reshape(1,10)
```

Out[31]:
```
array([[ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19]])
```

Now we do some operations

In [32]:
```
my_arr_1
```

Out[32]:
```
array([[1, 2, 3, 4, 5],
       [4, 5, 6, 7, 8],
       [7, 8, 9, 3, 1]])
```

In [33]:
```
my_arr_1 * my_arr_1
```

Out[33]:
```
array([[ 1,  4,  9, 16, 25],
       [16, 25, 36, 49, 64],
       [49, 64, 81,  9,  1]])
```

In [34]:
```
np.ones((5,3))
```

Out[34]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

In [35]:
```
np.zeros((2,5))
```

Out[35]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

How to generate random numbers?

In [36]:
```
np.random.randint(10,99)
```

Out[36]:
```
74
```

What if we want to generate more than one number?

In [37]:
```
np.random.randint(10,99,4)
```

Out[37]: array([73, 29, 81, 11])

How to reshape it to a multidimensional array?

In [38]:
```
np.random.randint(10,99,4).reshape(2,2)
```

Out[38]:
```
array([[80, 12],
       [63, 71]])
```