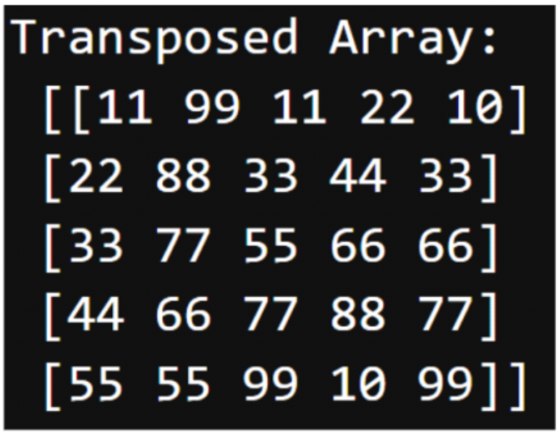
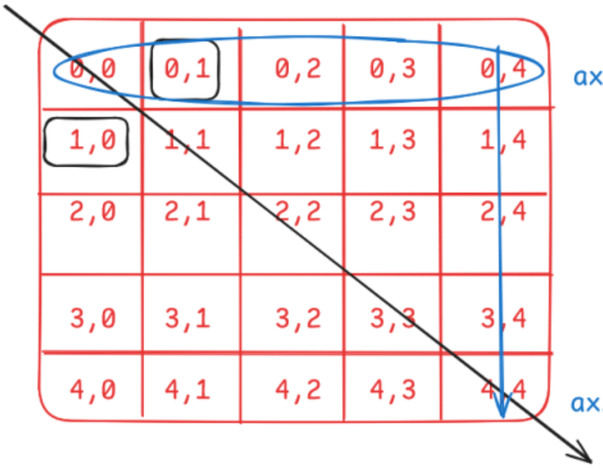
NumPy-II

🎯 Session Objectives:

- ☑ Apply indexing and slicing on arrays
- ☑ Understand how to perform common array operations using NumPy.
- ☑ Perform sorting, reshaping, and concatenating arrays.
- ☑ Distinguish between lists and arrays.
- ☑ Learn statistical and transformation operations on arrays.



```
Transposed Array:
 [[11 99 11 22 10]
 [22 88 33 44 33]
 [33 77 55 66 66]
 [44 66 77 88 77]
 [55 55 99 10 99]]
```



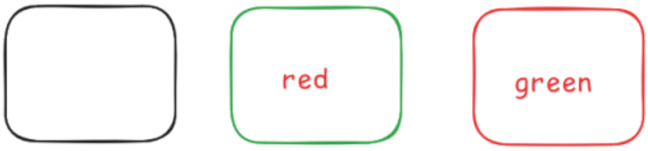
0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

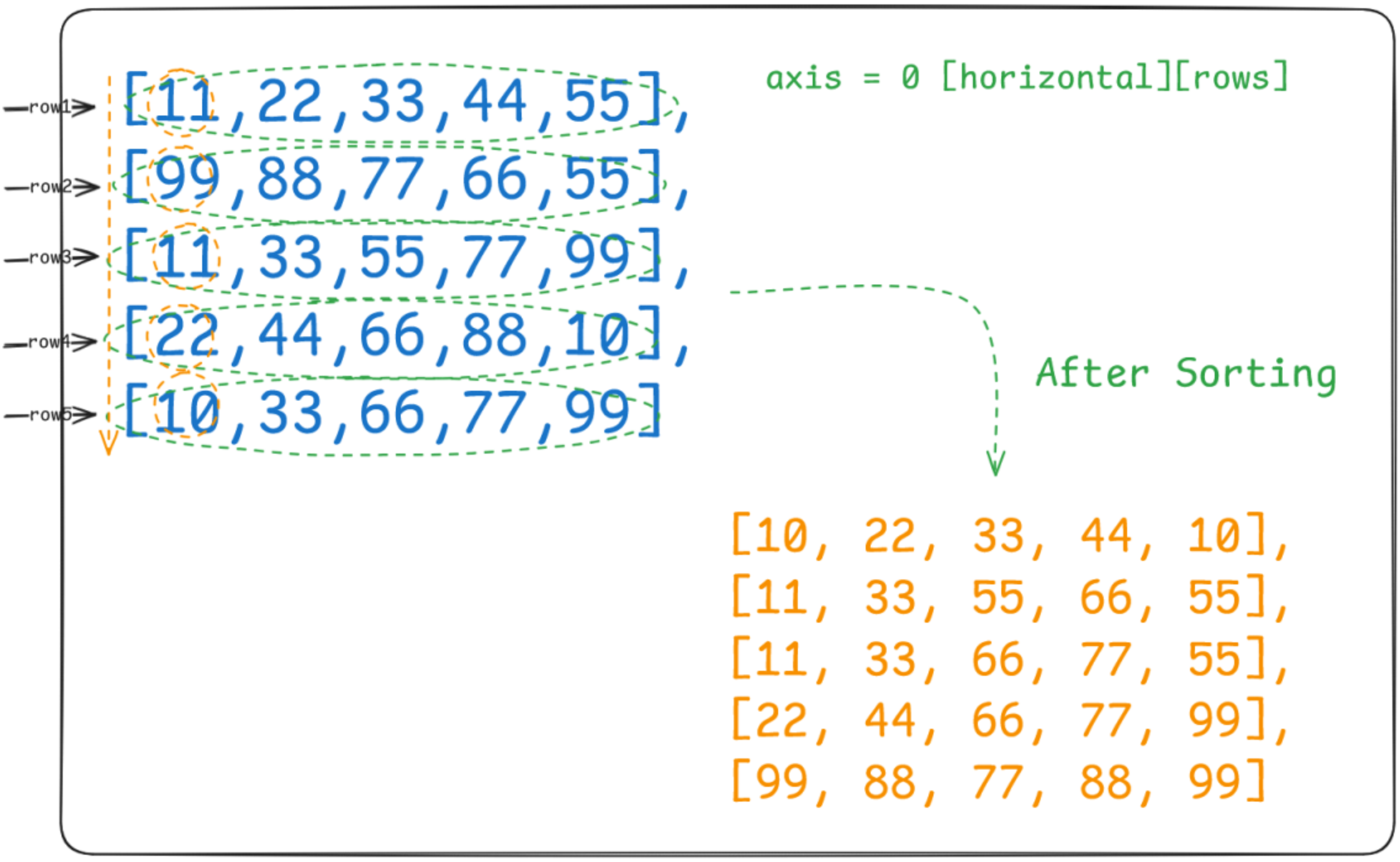
axis = 0 [Horizontal]

```
temp = arr[i][j]
arr[i][j] = arr[j][i]
arr[j][i] = temp
```

axis = 1 [Vertical]

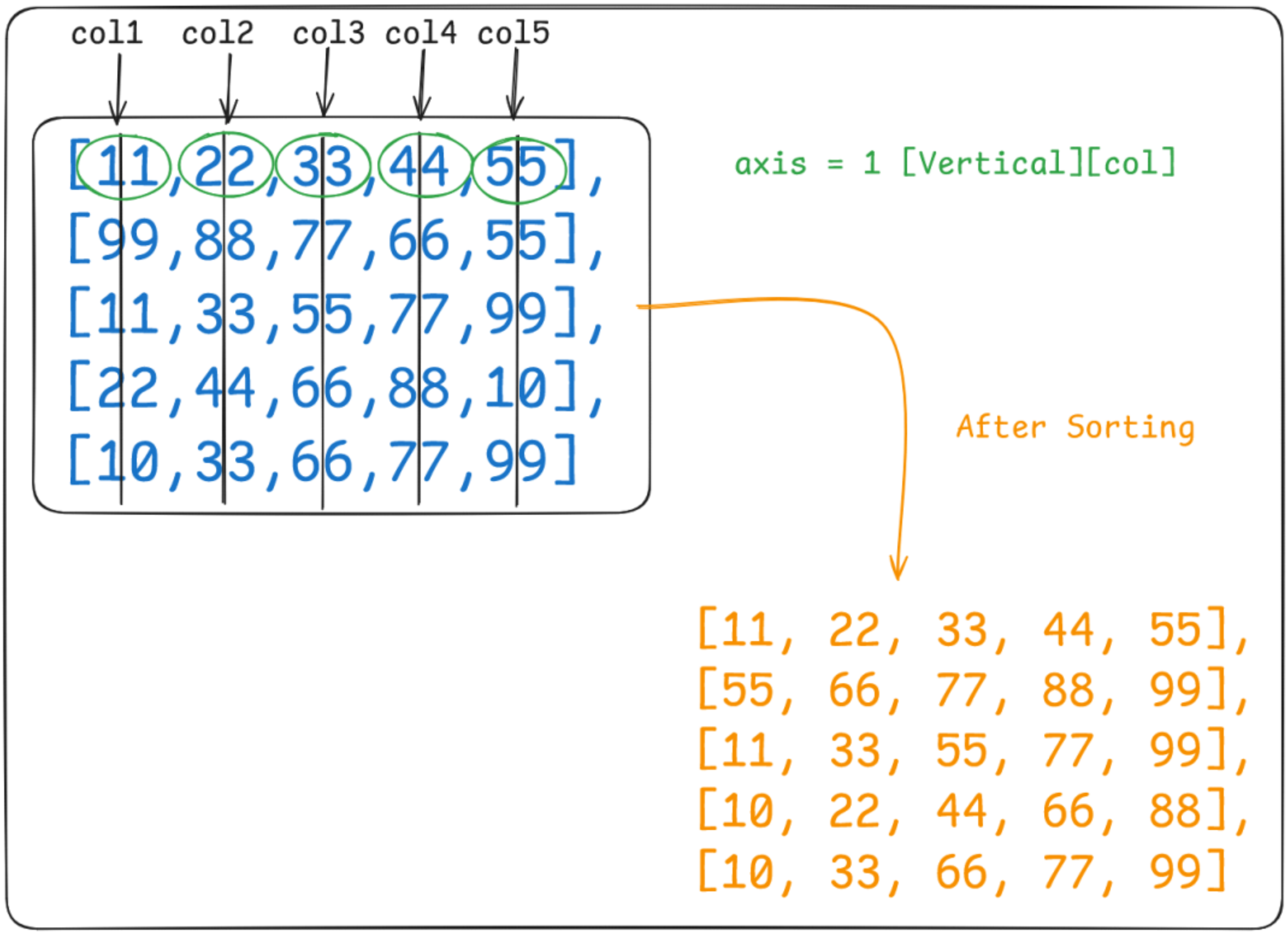
i == j





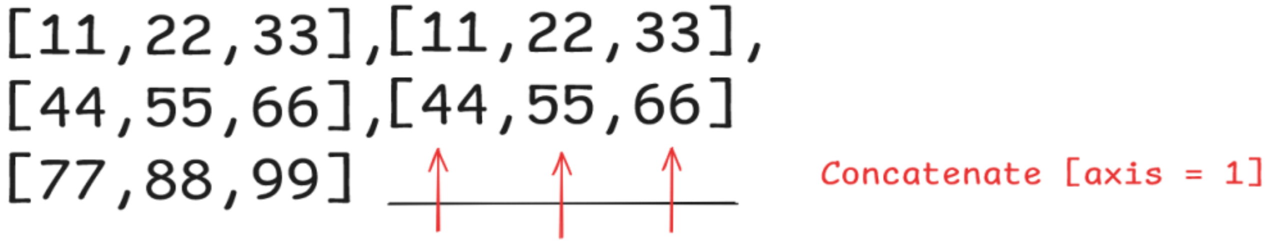
```
[11,22,33,44,55],
[99,88,77,66,55],
[11,33,55,77,99],
[22,44,66,88,10],
[10,33,66,77,99]
```

```
[10, 22, 33, 44, 10],
[11, 33, 55, 66, 55],
[11, 33, 66, 77, 55],
[22, 44, 66, 77, 99],
[99, 88, 77, 88, 99]
```



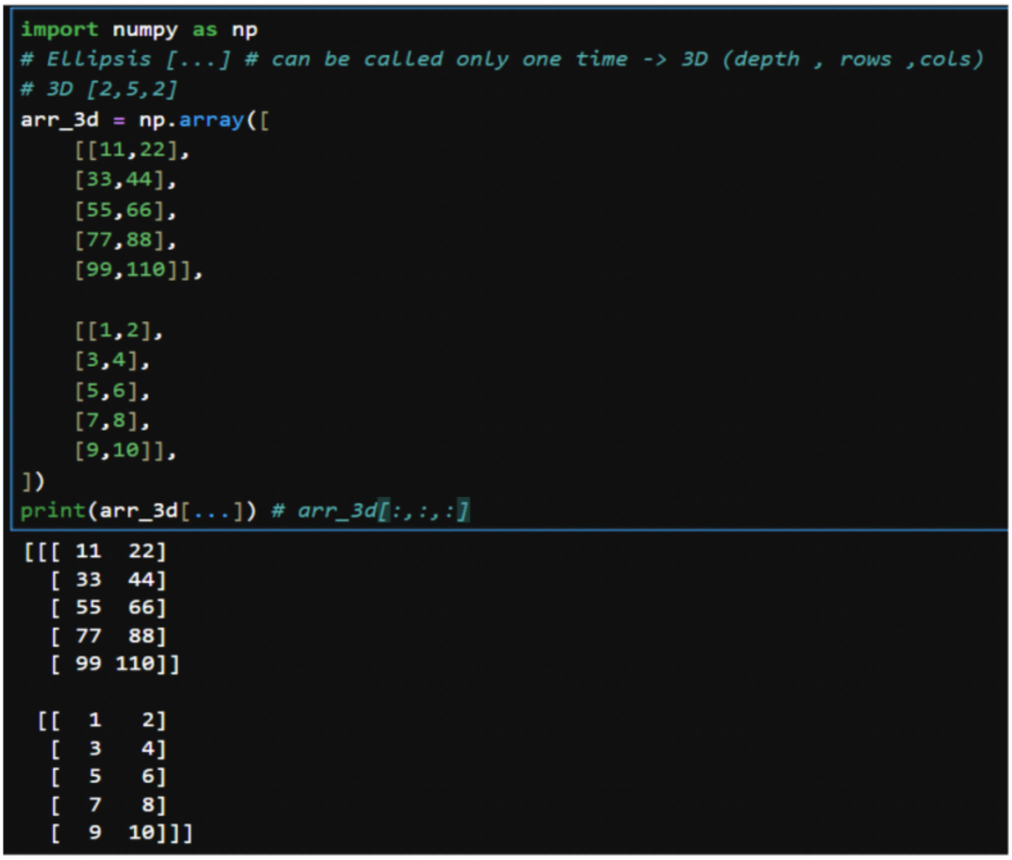
```
[11,22,33,44,55],
[99,88,77,66,55],
[11,33,55,77,99],
[22,44,66,88,10],
[10,33,66,77,99]
```

```
[11, 22, 33, 44, 55],
[55, 66, 77, 88, 99],
[11, 33, 55, 77, 99],
[10, 22, 44, 66, 88],
[10, 33, 66, 77, 99]
```



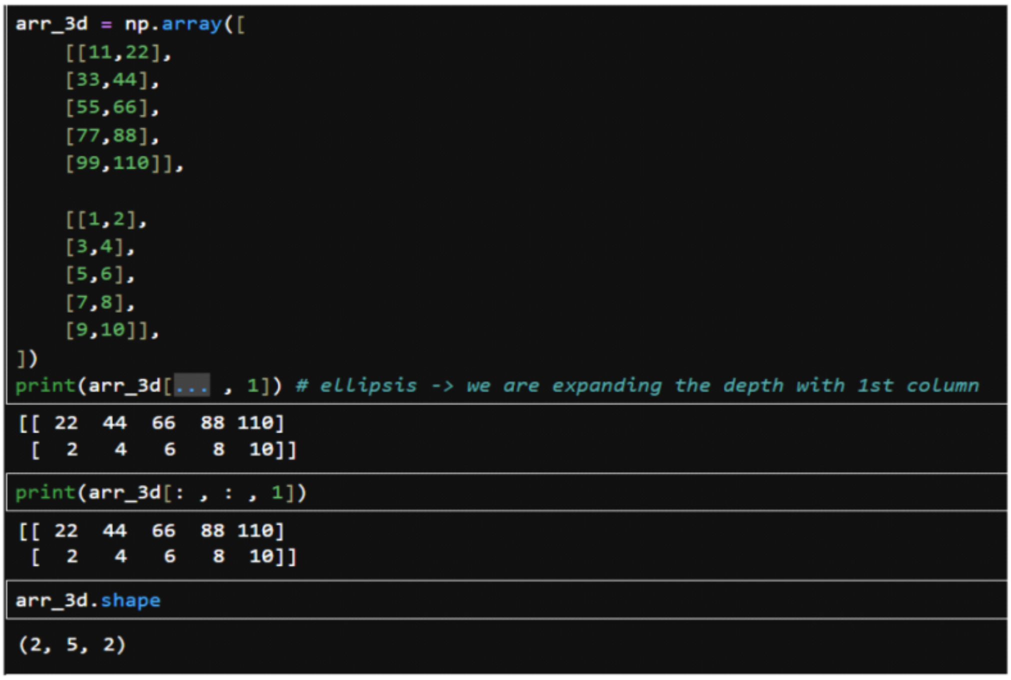
[11,22,33],[11,22,33],
[44,55,66],[44,55,66]
[77,88,99]

Concatenate [axis = 1]



```
import numpy as np
# Ellipsis [...] # can be called only one time -> 3D (depth , rows ,cols)
# 3D [2,5,2]
arr_3d = np.array([
    [[11,22],
    [33,44],
    [55,66],
    [77,88],
    [99,110]],

    [[1,2],
    [3,4],
    [5,6],
    [7,8],
    [9,10]],
])
print(arr_3d[...]) # arr_3d[:,:,:]
```

```
[[[ 11  22]
  [ 33  44]
  [ 55  66]
  [ 77  88]
  [ 99 110]]

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



```
arr_3d = np.array([
    [[11,22],
    [33,44],
    [55,66],
    [77,88],
    [99,110]],

    [[1,2],
    [3,4],
    [5,6],
    [7,8],
    [9,10]],
])
print(arr_3d[... , 1]) # ellipsis -> we are expanding the depth with 1st column
```

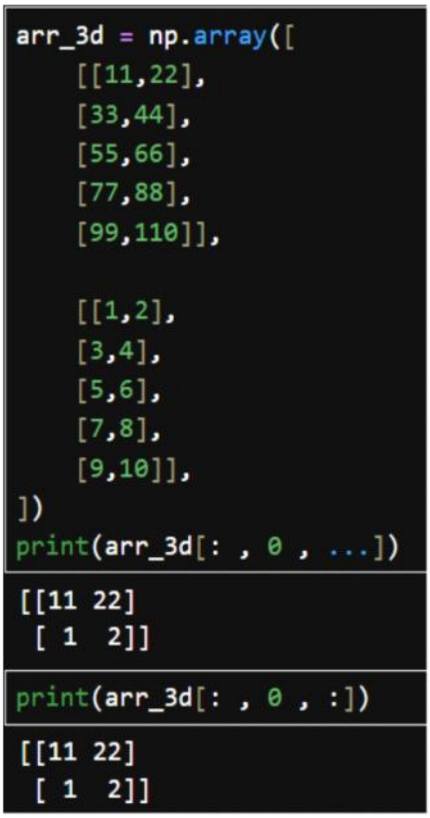
```
[[ 22  44  66  88 110]
 [  2   4   6   8  10]]
```

```
print(arr_3d[: , : , 1])
```

```
[[ 22  44  66  88 110]
 [  2   4   6   8  10]]
```

```
arr_3d.shape
```

```
(2, 5, 2)
```

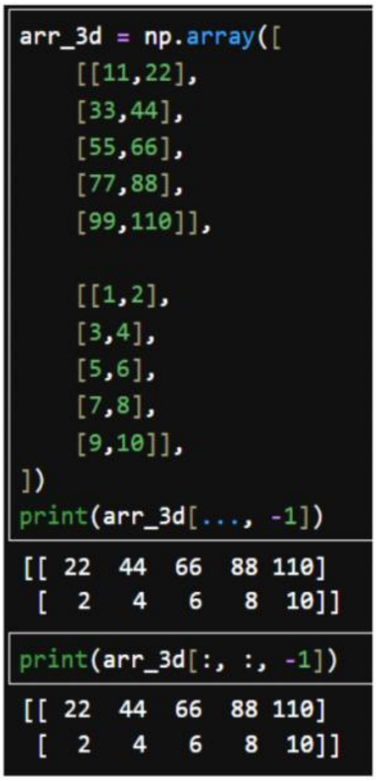


```
arr_3d = np.array([
    [[11,22],
    [33,44],
    [55,66],
    [77,88],
    [99,110]],

    [[1,2],
    [3,4],
    [5,6],
    [7,8],
    [9,10]],
])
print(arr_3d[: , 0 , ...])
```

```
[[11 22]
 [ 1  2]]
```

```
print(arr_3d[: , 0 , :])
```

```
[[11 22]
 [ 1  2]]
```

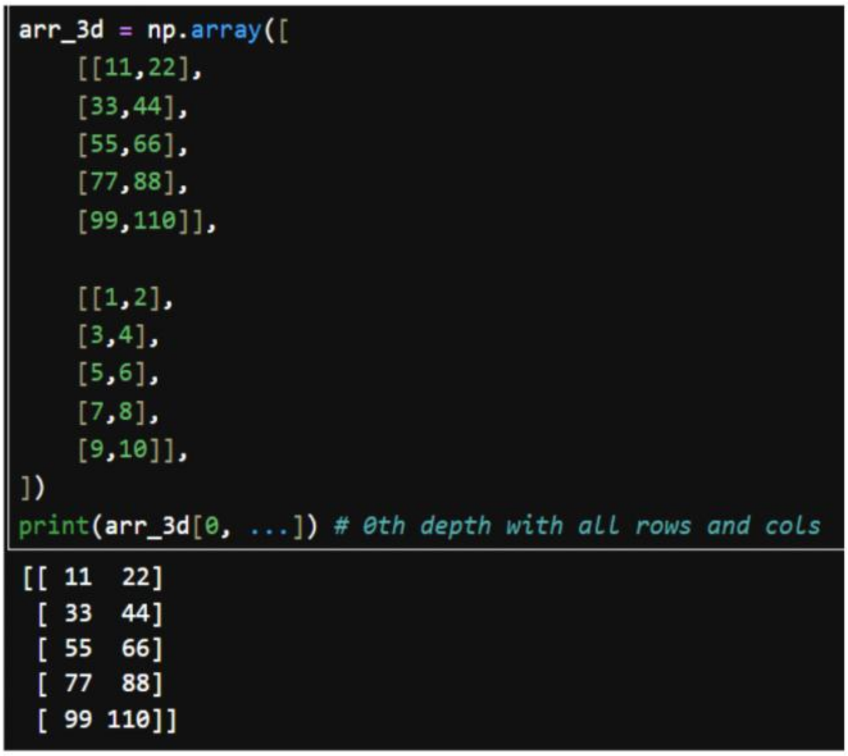


```
arr_3d = np.array([
    [[11,22],
    [33,44],
    [55,66],
    [77,88],
    [99,110]],

    [[1,2],
    [3,4],
    [5,6],
    [7,8],
    [9,10]],
])
print(arr_3d[..., -1])
```

```
[[ 22  44  66  88 110]
 [  2   4   6   8  10]]
```

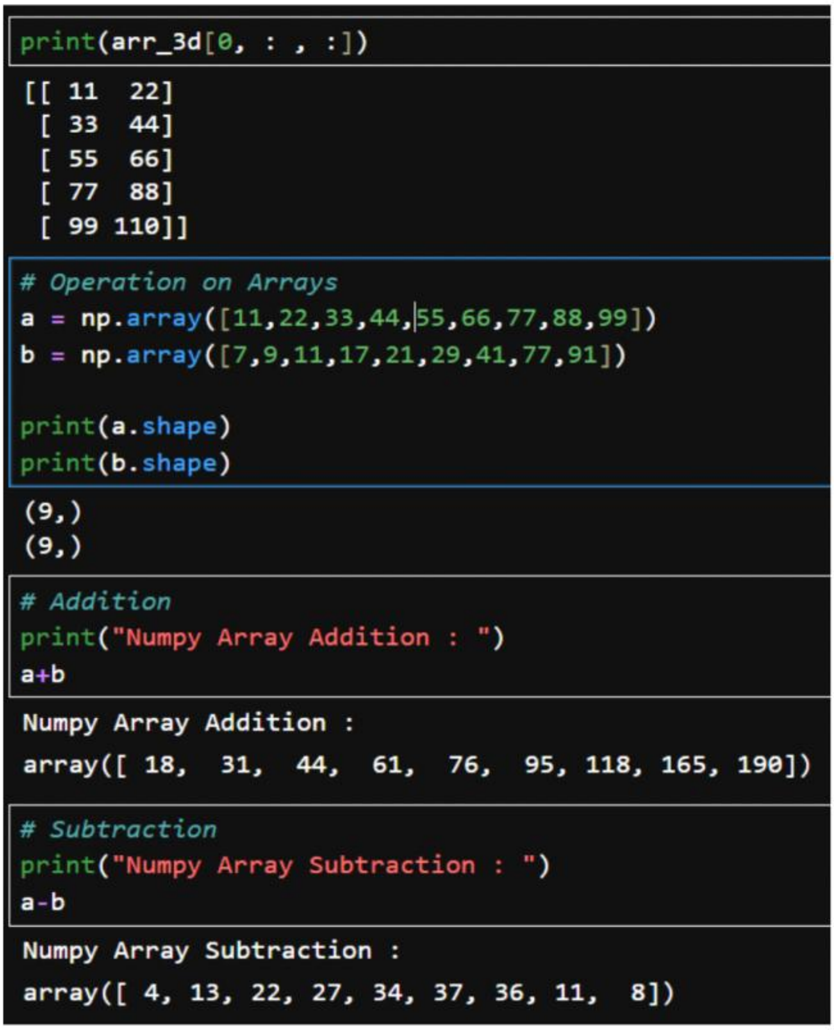
```
print(arr_3d[:, :, -1])
```

```
[[ 22  44  66  88 110]
 [  2   4   6   8  10]]
```



```
arr_3d = np.array([
    [[11,22],
    [33,44],
    [55,66],
    [77,88],
    [99,110]],

    [[1,2],
    [3,4],
    [5,6],
    [7,8],
    [9,10]],
])
print(arr_3d[0, ...]) # 0th depth with all rows and cols
```

```
[[ 11  22]
 [ 33  44]
 [ 55  66]
 [ 77  88]
 [ 99 110]]
```



```
print(arr_3d[0, : , :])
```

```
[[ 11  22]
 [ 33  44]
 [ 55  66]
 [ 77  88]
 [ 99 110]]
```

```
# Operation on Arrays
a = np.array([11,22,33,44,55,66,77,88,99])
b = np.array([7,9,11,17,21,29,41,77,91])

print(a.shape)
print(b.shape)
```

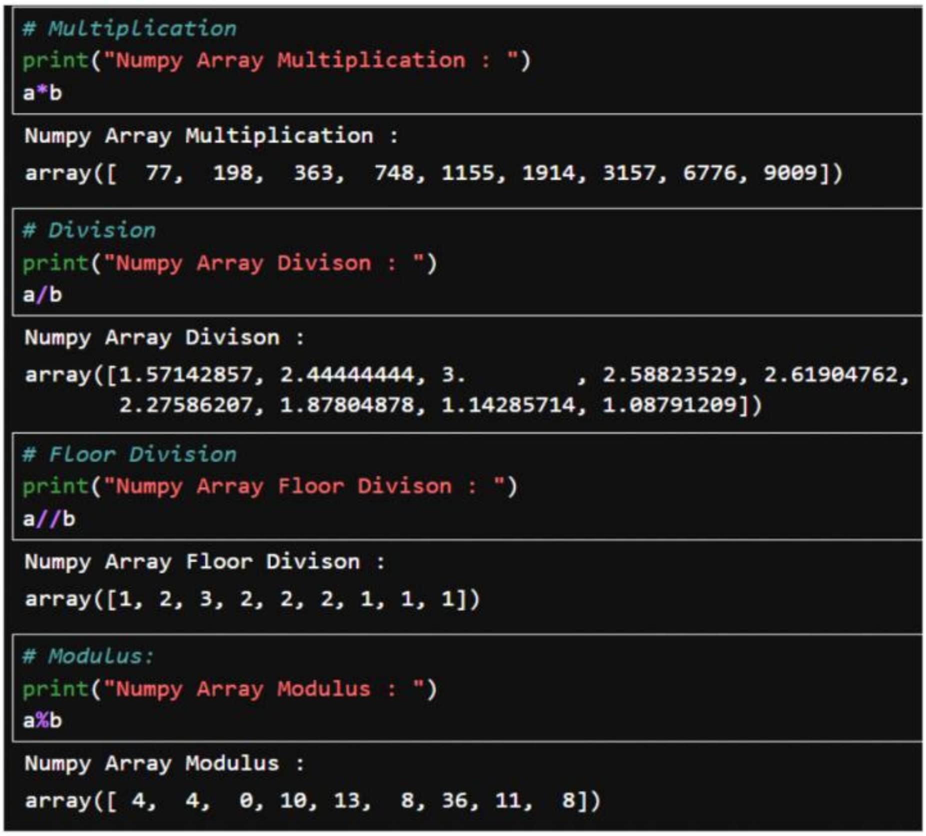
```
(9,)
(9,)
```

```
# Addition
print("Numpy Array Addition : ")
a+b
```

```
Numpy Array Addition :
array([ 18,  31,  44,  61,  76,  95, 118, 165, 190])
```

```
# Subtraction
print("Numpy Array Subtraction : ")
a-b
```

```
Numpy Array Subtraction :
array([ 4, 13, 22, 27, 34, 37, 36, 11,  8])
```



```
# Multiplication
print("Numpy Array Multiplication : ")
a*b
```

```
Numpy Array Multiplication :
array([  77,  198,  363,  748, 1155, 1914, 3157, 6776, 9009])
```

```
# Division
print("Numpy Array Divison : ")
a/b
```

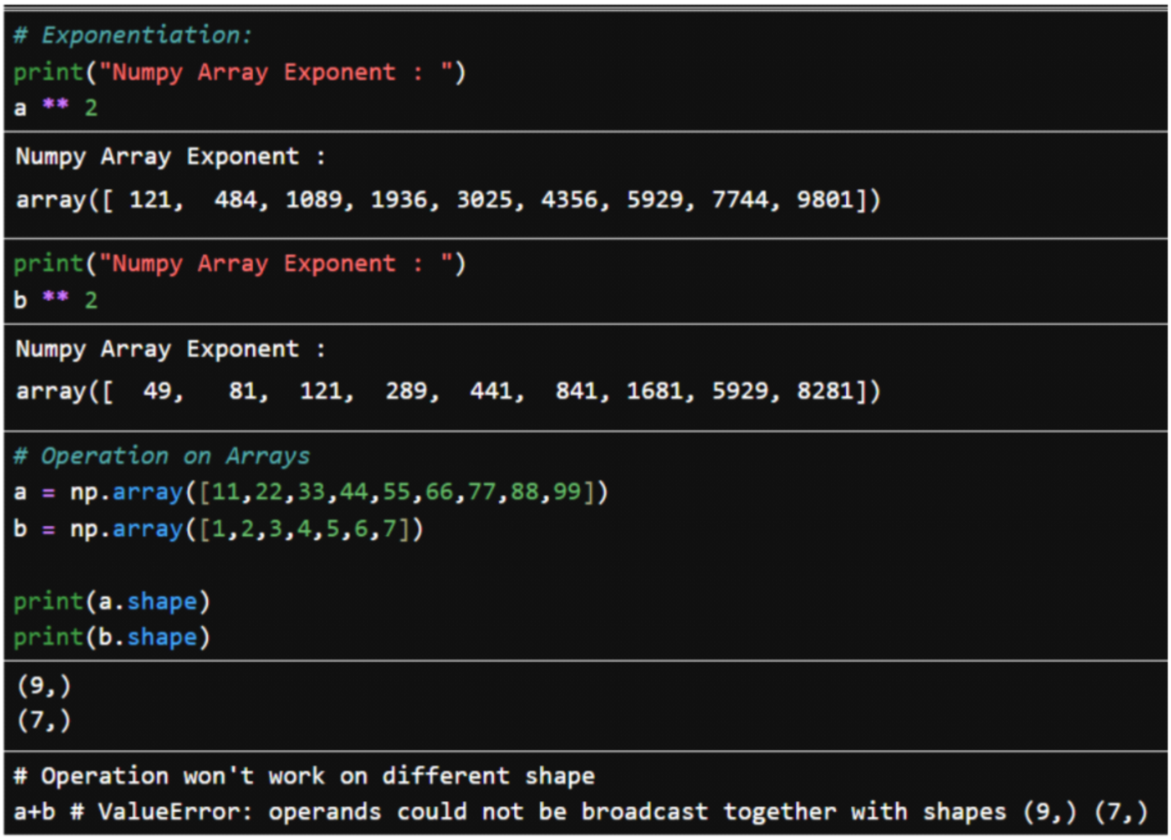
```
Numpy Array Divison :
array([1.57142857, 2.44444444, 3.        , 2.58823529, 2.61904762,
       2.27586207, 1.87804878, 1.14285714, 1.08791209])
```

```
# Floor Division
print("Numpy Array Floor Divison : ")
a//b
```

```
Numpy Array Floor Divison :
array([1, 2, 3, 2, 2, 2, 1, 1, 1])
```

```
# Modulus:
print("Numpy Array Modulus : ")
a%b
```

```
Numpy Array Modulus :
array([ 4,  4,  0, 10, 13,  8, 36, 11,  8])
```



```
# Exponentiation:
print("Numpy Array Exponent : ")
a ** 2
```

```
Numpy Array Exponent :
array([ 121,  484, 1089, 1936, 3025, 4356, 5929, 7744, 9801])
```

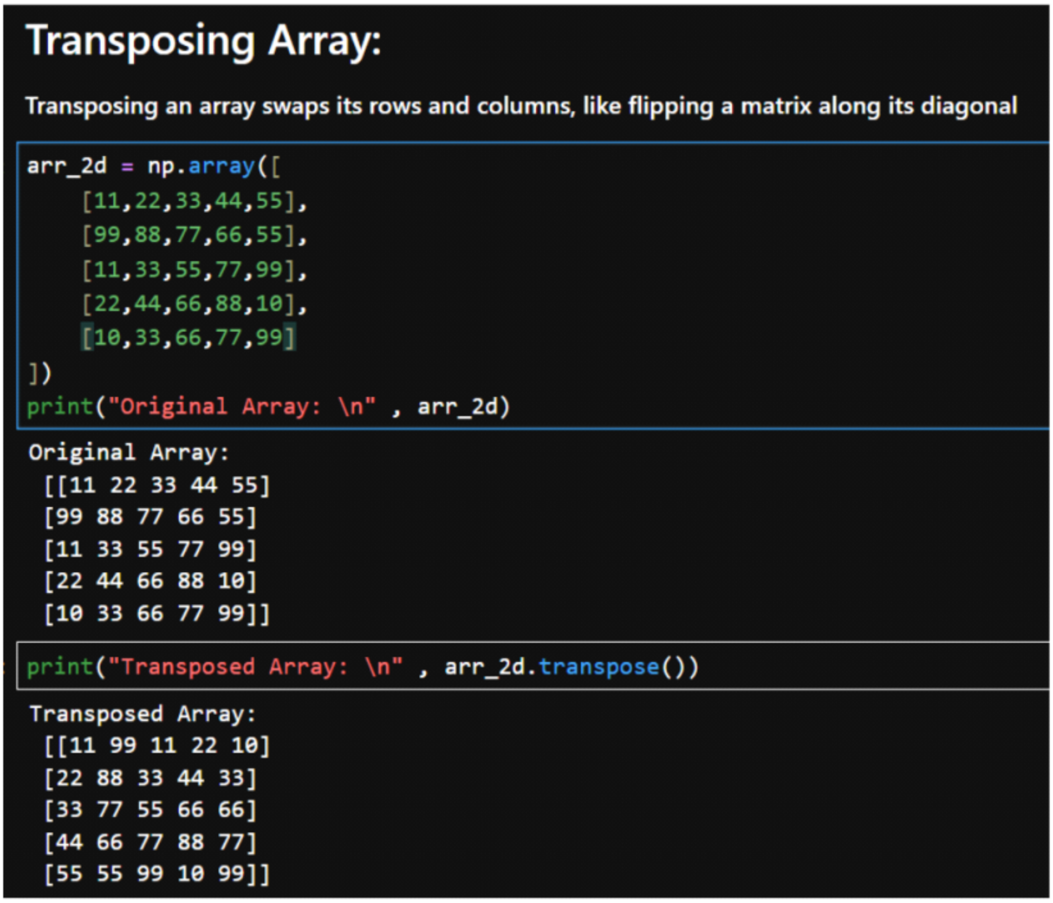
```
print("Numpy Array Exponent : ")
b ** 2
```

```
Numpy Array Exponent :
array([  49,   81,  121,  289,  441,  841, 1681, 5929, 8281])
```

```
# Operation on Arrays
a = np.array([11,22,33,44,55,66,77,88,99])
b = np.array([1,2,3,4,5,6,7])

print(a.shape)
print(b.shape)
```

```
(9,)
(7,)
```

```
# Operation won't work on different shape
a+b # ValueError: operands could not be broadcast together with shapes (9,) (7,)
```



Transposing Array:

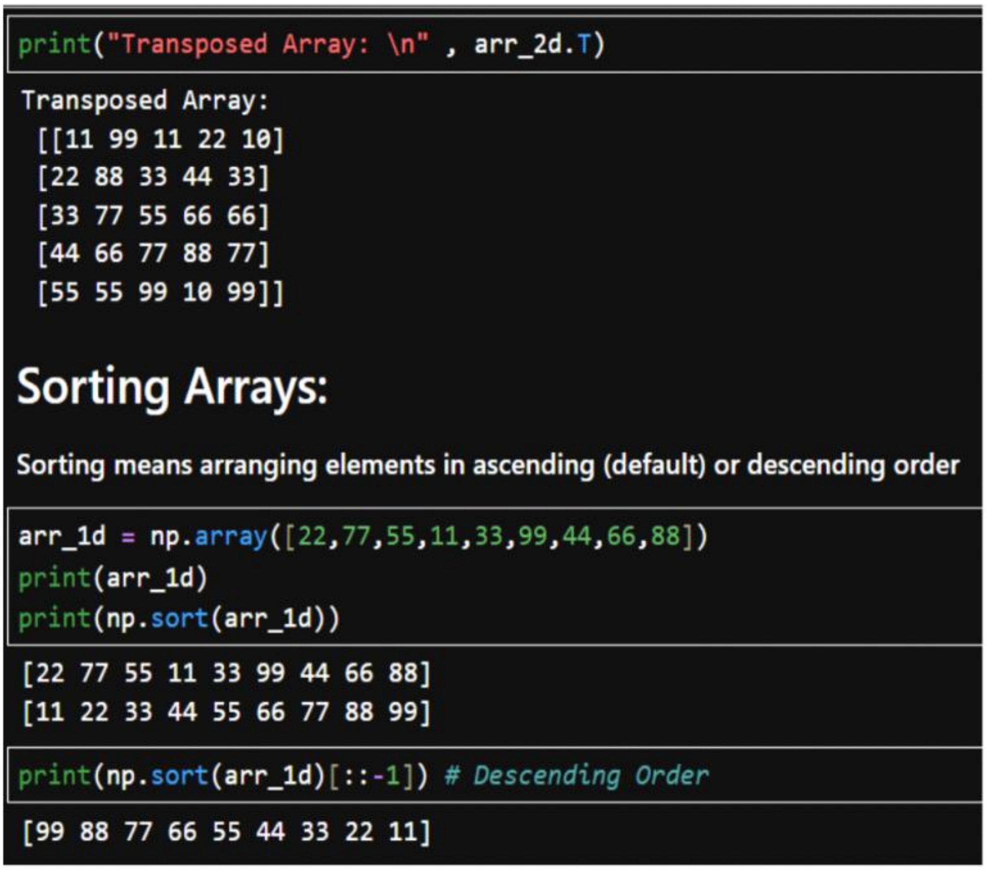
Transposing an array swaps its rows and columns, like flipping a matrix along its diagonal

```
arr_2d = np.array([
    [11,22,33,44,55],
    [99,88,77,66,55],
    [11,33,55,77,99],
    [22,44,66,88,10],
    [10,33,66,77,99]
])
print("Original Array: \n" , arr_2d)
```

```
Original Array:
 [[11 22 33 44 55]
 [99 88 77 66 55]
 [11 33 55 77 99]
 [22 44 66 88 10]
 [10 33 66 77 99]]
```

```
print("Transposed Array: \n" , arr_2d.transpose())
```

```
Transposed Array:
 [[11 99 11 22 10]
 [22 88 33 44 33]
 [33 77 55 66 66]
 [44 66 77 88 77]
 [55 55 99 10 99]]
```



```
print("Transposed Array: \n" , arr_2d.T)
```

```
Transposed Array:
 [[11 99 11 22 10]
 [22 88 33 44 33]
 [33 77 55 66 66]
 [44 66 77 88 77]
 [55 55 99 10 99]]
```

Sorting Arrays:

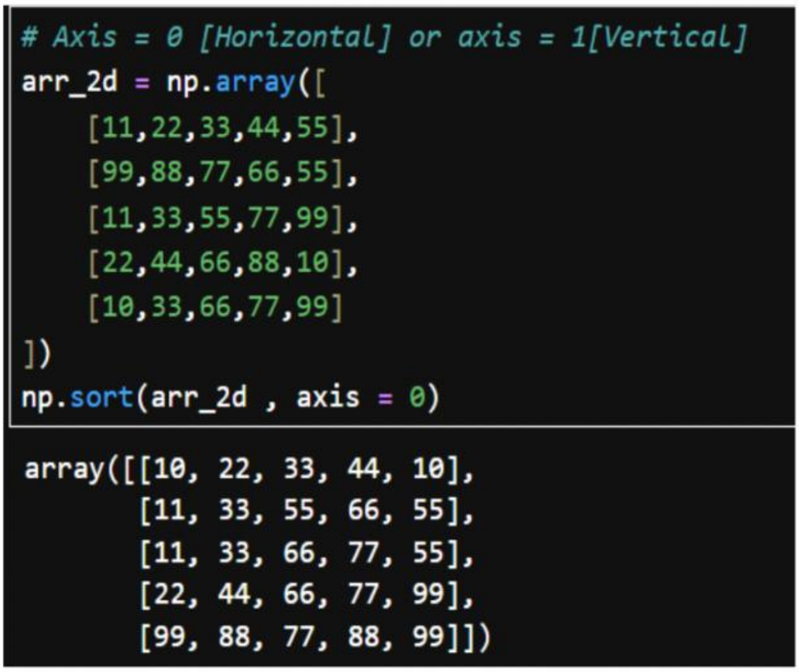
Sorting means arranging elements in ascending (default) or descending order

```
arr_1d = np.array([22,77,55,11,33,99,44,66,88])
print(arr_1d)
print(np.sort(arr_1d))
```

```
[22 77 55 11 33 99 44 66 88]
[11 22 33 44 55 66 77 88 99]
```

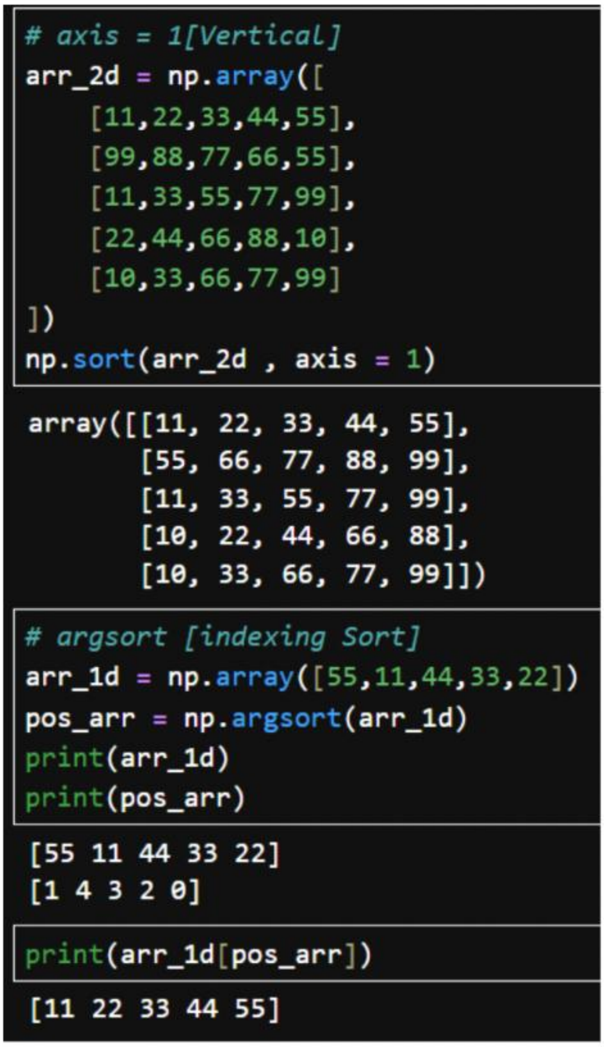
```
print(np.sort(arr_1d)[::-1]) # Descending Order
```

```
[99 88 77 66 55 44 33 22 11]
```



```
# Axis = 0 [Horizontal] or axis = 1[Vertical]
arr_2d = np.array([
    [11,22,33,44,55],
    [99,88,77,66,55],
    [11,33,55,77,99],
    [22,44,66,88,10],
    [10,33,66,77,99]
])
np.sort(arr_2d , axis = 0)
```

```
array([[10, 22, 33, 44, 10],
       [11, 33, 55, 66, 55],
       [11, 33, 66, 77, 55],
       [22, 44, 66, 77, 99],
       [99, 88, 77, 88, 99]])
```



```
# axis = 1[Vertical]
arr_2d = np.array([
    [11,22,33,44,55],
    [99,88,77,66,55],
    [11,33,55,77,99],
    [22,44,66,88,10],
    [10,33,66,77,99]
])
np.sort(arr_2d , axis = 1)
```

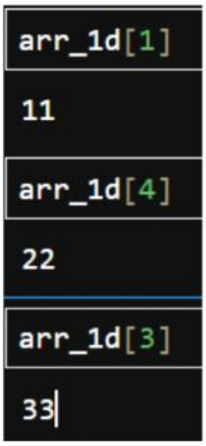
```
array([[11, 22, 33, 44, 55],
       [55, 66, 77, 88, 99],
       [11, 33, 55, 77, 99],
       [10, 22, 44, 66, 88],
       [10, 33, 66, 77, 99]])
```

```
# argsort [indexing Sort]
arr_1d = np.array([55,11,44,33,22])
pos_arr = np.argsort(arr_1d)
print(arr_1d)
print(pos_arr)
```

```
[55 11 44 33 22]
[1 4 3 2 0]
```

```
print(arr_1d[pos_arr])
```

```
[11 22 33 44 55]
```



```
arr_1d[1]
```

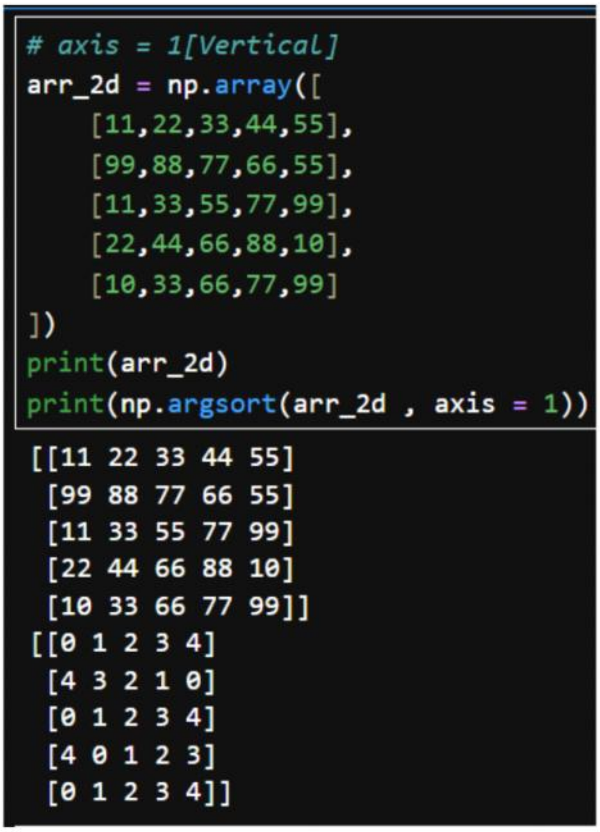
```
11
```

```
arr_1d[4]
```

```
22
```

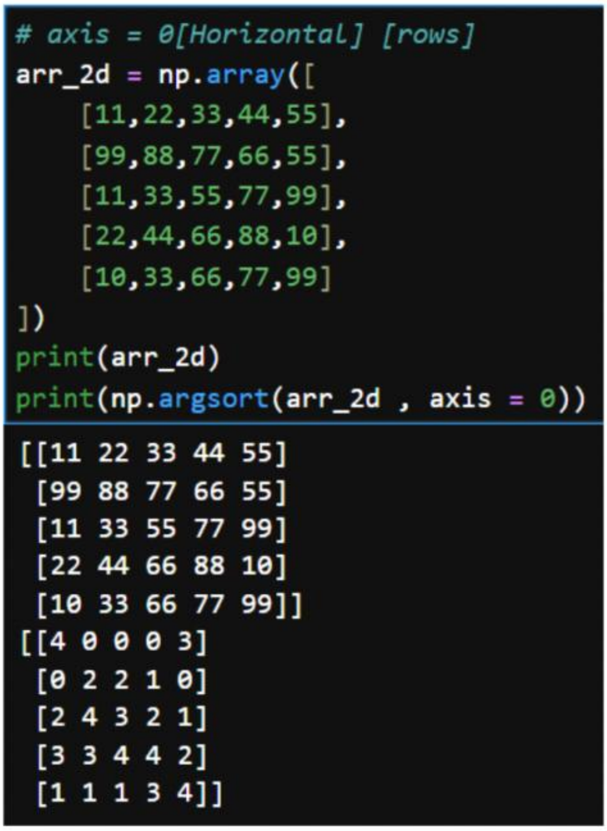
```
arr_1d[3]
```

```
33
```



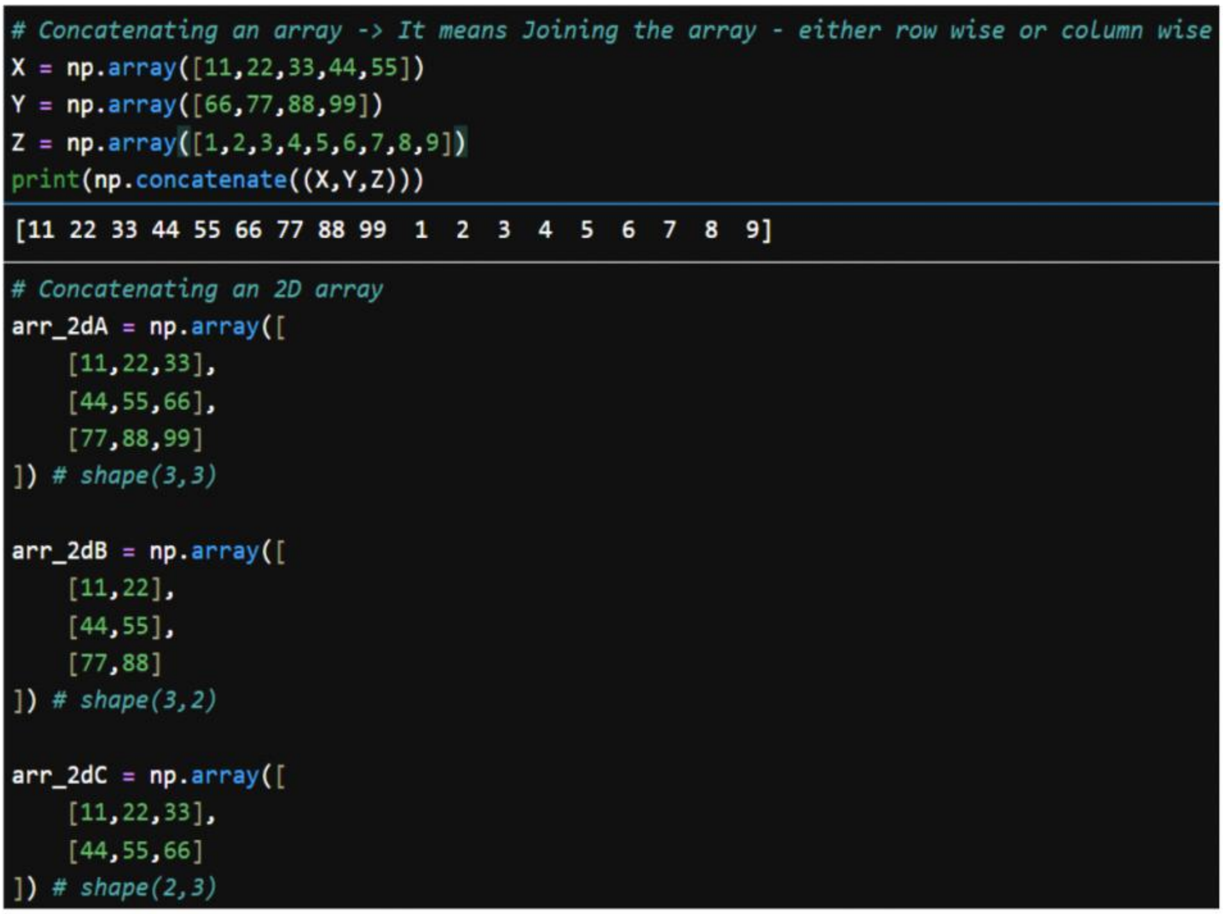
```
# axis = 1[Vertical]
arr_2d = np.array([
    [11,22,33,44,55],
    [99,88,77,66,55],
    [11,33,55,77,99],
    [22,44,66,88,10],
    [10,33,66,77,99]
])
print(arr_2d)
print(np.argsort(arr_2d , axis = 1))
```

```
[[11 22 33 44 55]
 [99 88 77 66 55]
 [11 33 55 77 99]
 [22 44 66 88 10]
 [10 33 66 77 99]]
[[0 1 2 3 4]
 [4 3 2 1 0]
 [0 1 2 3 4]
 [4 0 1 2 3]
 [0 1 2 3 4]]
```



```
# axis = 0[Horizontal] [rows]
arr_2d = np.array([
    [11,22,33,44,55],
    [99,88,77,66,55],
    [11,33,55,77,99],
    [22,44,66,88,10],
    [10,33,66,77,99]
])
print(arr_2d)
print(np.argsort(arr_2d , axis = 0))
```

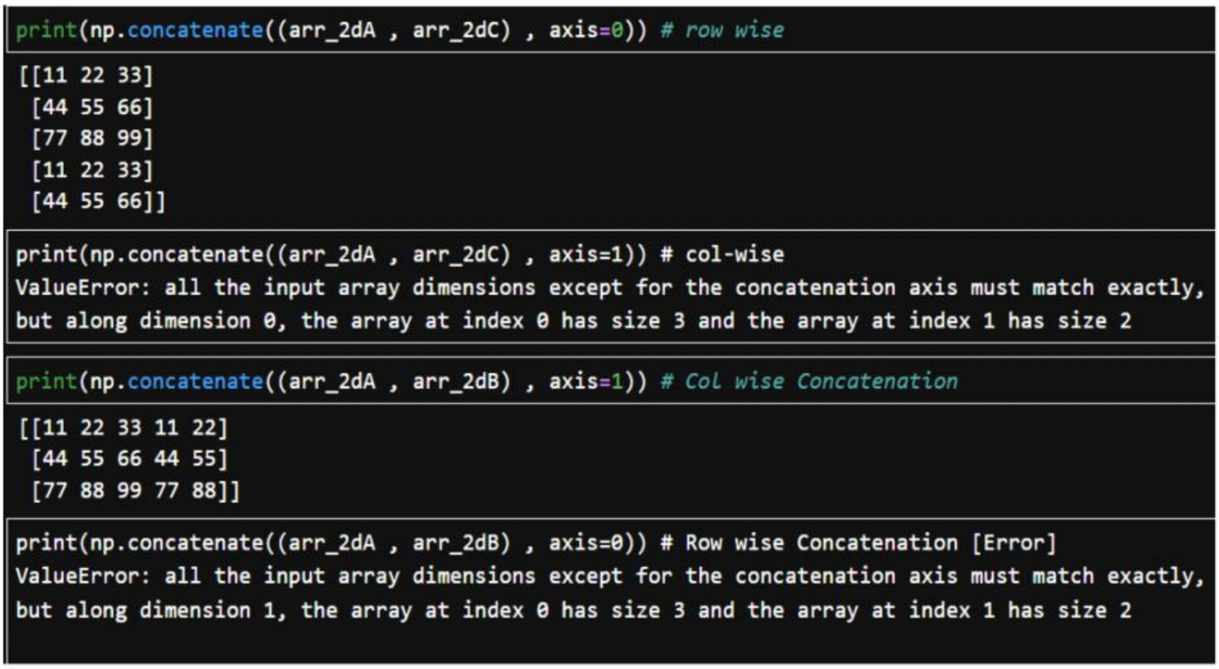
```
[[11 22 33 44 55]
 [99 88 77 66 55]
 [11 33 55 77 99]
 [22 44 66 88 10]
 [10 33 66 77 99]]
[[4 0 0 0 3]
 [0 2 2 1 0]
 [2 4 3 2 1]
 [3 3 4 4 2]
 [1 1 1 3 4]]
```



```
# Concatenating an array -> It means Joining the array - either row wise or column wise
X = np.array([11,22,33,44,55])
Y = np.array([66,77,88,99])
Z = np.array([1,2,3,4,5,6,7,8,9])
print(np.concatenate((X,Y,Z)))
```

```
[11 22 33 44 55 66 77 88 99  1  2  3  4  5  6  7  8  9]
```

```
# Concatenating an 2D array
arr_2dA = np.array([
    [11,22,33],
    [44,55,66],
    [77,88,99]
]) # shape(3,3)

arr_2dB = np.array([
    [11,22],
    [44,55],
    [77,88]
]) # shape(3,2)

arr_2dC = np.array([
    [11,22,33],
    [44,55,66]
]) # shape(2,3)
```



```
print(np.concatenate((arr_2dA , arr_2dC) , axis=0)) # row wise
```

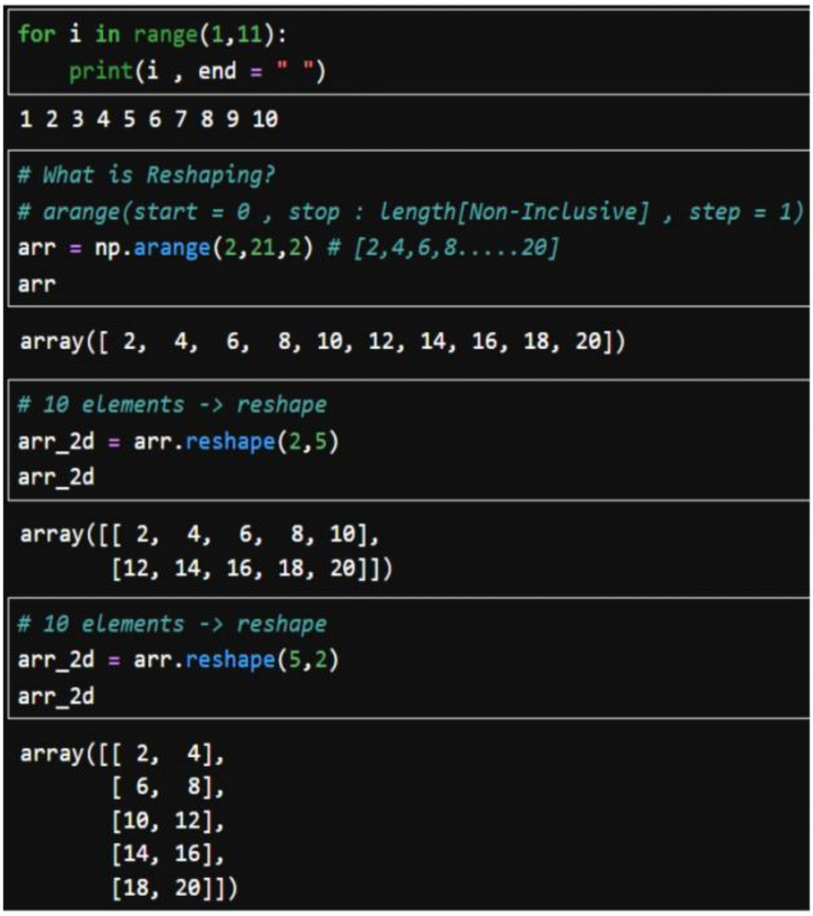
```
[[11 22 33]
 [44 55 66]
 [77 88 99]
 [11 22 33]
 [44 55 66]]
```

```
print(np.concatenate((arr_2dA , arr_2dC) , axis=1)) # col-wise
ValueError: all the input array dimensions except for the concatenation axis must match exactly,
but along dimension 0, the array at index 0 has size 3 and the array at index 1 has size 2
```

```
print(np.concatenate((arr_2dA , arr_2dB) , axis=1)) # Col wise Concatenation
```

```
[[11 22 33 11 22]
 [44 55 66 44 55]
 [77 88 99 77 88]]
```

```
print(np.concatenate((arr_2dA , arr_2dB) , axis=0)) # Row wise Concatenation [Error]
ValueError: all the input array dimensions except for the concatenation axis must match exactly,
but along dimension 1, the array at index 0 has size 3 and the array at index 1 has size 2
```



```
for i in range(1,11):
    print(i , end = " ")
```

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

```
# What is Reshaping?
# arange(start = 0 , stop : length[Non-Inclusive] , step = 1)
arr = np.arange(2,21,2) # [2,4,6,8.....20]
arr
```

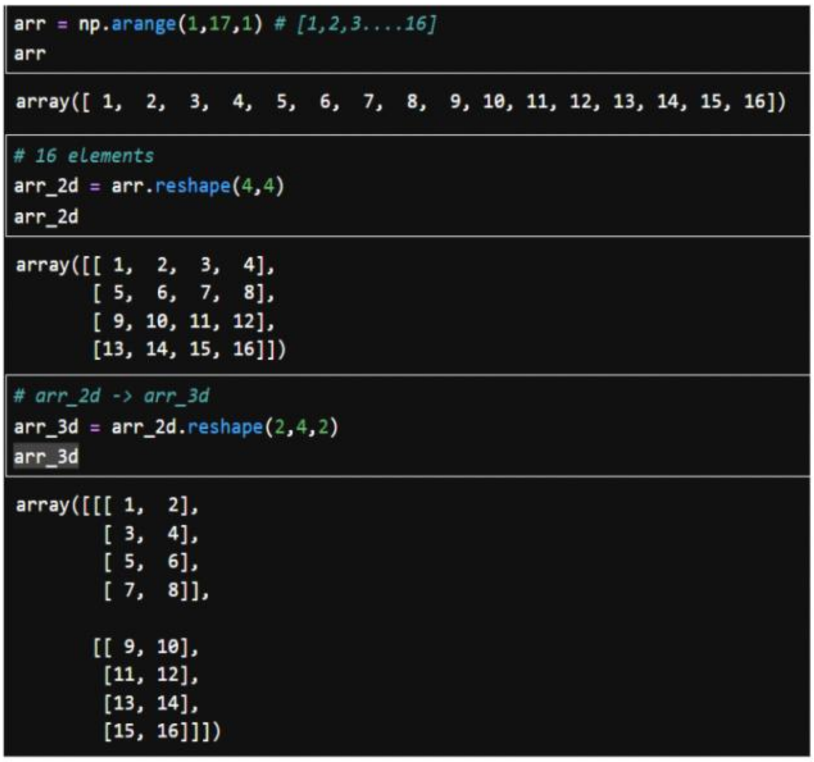
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

```
# 10 elements -> reshape
arr_2d = arr.reshape(2,5)
arr_2d
```

```
array([[ 2,  4,  6,  8, 10],
       [12, 14, 16, 18, 20]])
```

```
# 10 elements -> reshape
arr_2d = arr.reshape(5,2)
arr_2d
```

```
array([[ 2,  4],
       [ 6,  8],
       [10, 12],
       [14, 16],
       [18, 20]])
```



```
arr = np.arange(1,17,1) # [1,2,3....16]
arr
```

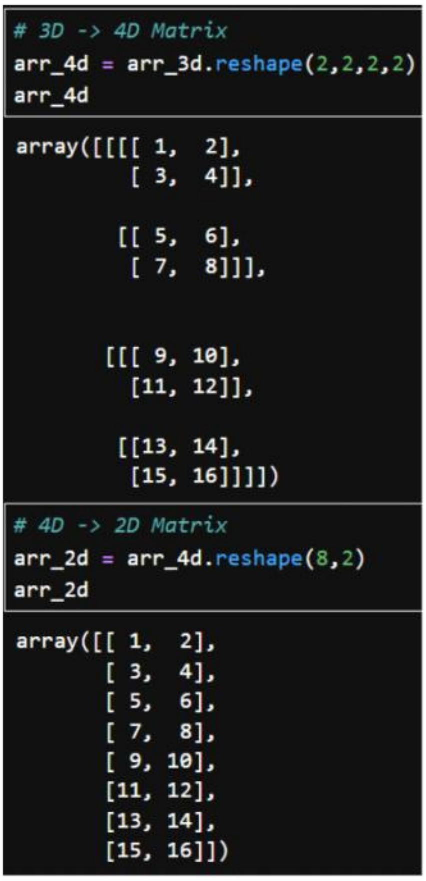
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

```
# 16 elements
arr_2d = arr.reshape(4,4)
arr_2d
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

```
# arr_2d -> arr_3d
arr_3d = arr_2d.reshape(2,4,2)
arr_3d
```

```
array([[[ 1,  2],
        [ 3,  4],
        [ 5,  6],
        [ 7,  8]],

       [[ 9, 10],
        [11, 12],
        [13, 14],
        [15, 16]]])
```

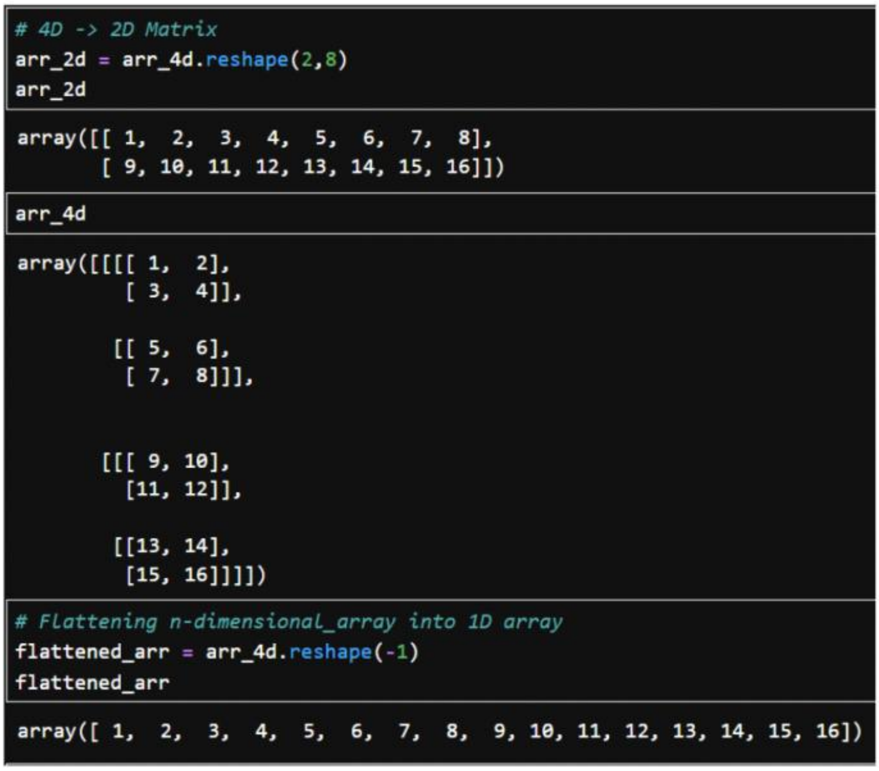


```
# 3D -> 4D Matrix
arr_4d = arr_3d.reshape(2,2,2,2)
arr_4d
```

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

        [[ 5,  6],
         [ 7,  8]]],


       [[[ 9, 10],
         [11, 12]],

        [[13, 14],
         [15, 16]]]])
```

```
# 4D -> 2D Matrix
arr_2d = arr_4d.reshape(8,2)
arr_2d
```

```
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10],
       [11, 12],
       [13, 14],
       [15, 16]])
```



```
# 4D -> 2D Matrix
arr_2d = arr_4d.reshape(2,8)
arr_2d
```

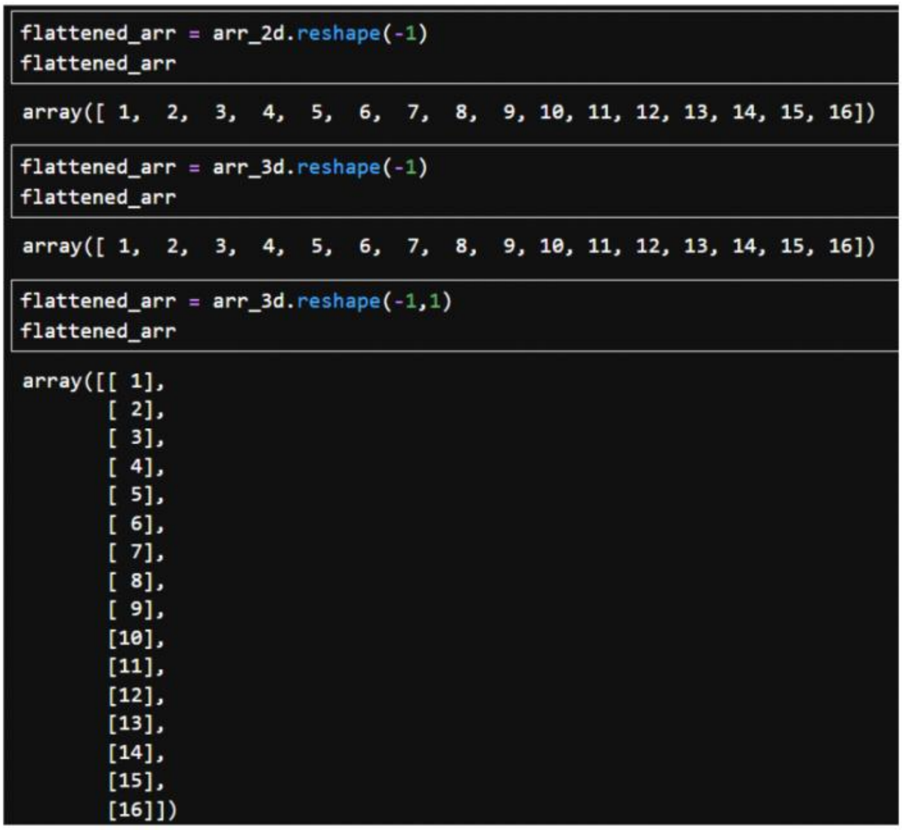
```
array([[ 1,  2,  3,  4,  5,  6,  7,  8],
       [ 9, 10, 11, 12, 13, 14, 15, 16]])
```

```
arr_4d
```

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

        [[ 5,  6],
         [ 7,  8]]],


       [[[ 9, 10],
         [11, 12]],

        [[13, 14],
         [15, 16]]]])
```

```
# Flattening n-dimensional_array into 1D array
flattened_arr = arr_4d.reshape(-1)
flattened_arr
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```



```
flattened_arr = arr_2d.reshape(-1)
flattened_arr
```

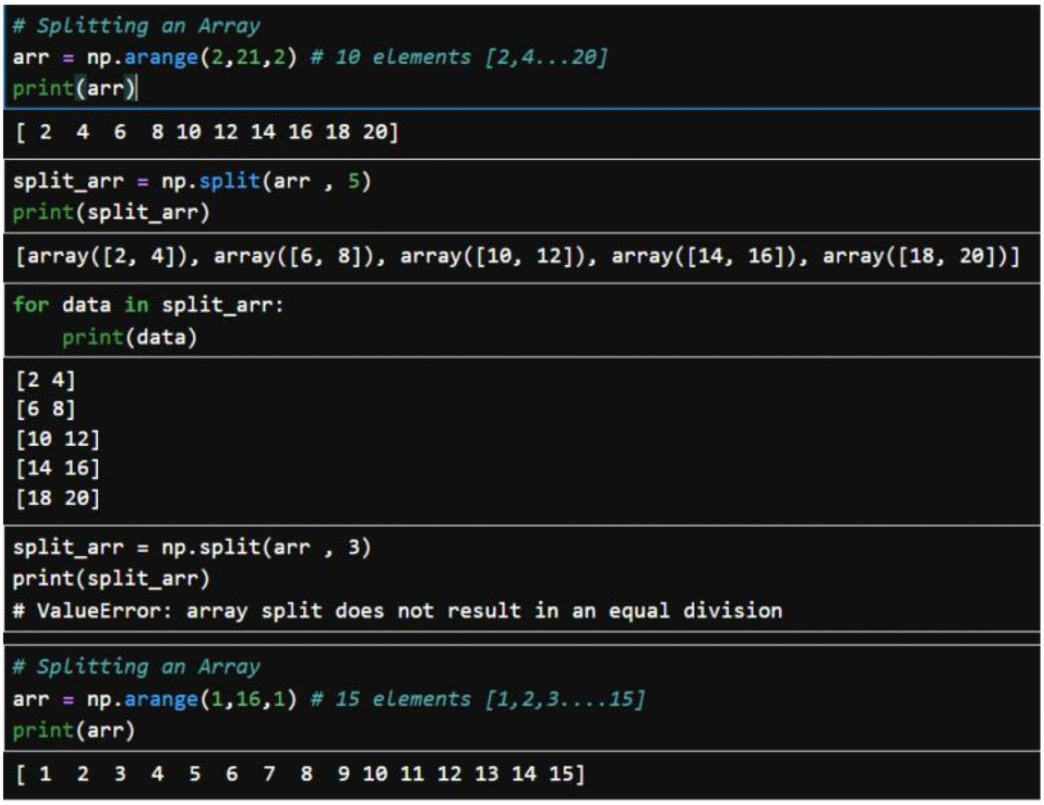
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

```
flattened_arr = arr_3d.reshape(-1)
flattened_arr
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

```
flattened_arr = arr_3d.reshape(-1,1)
flattened_arr
```

```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10],
       [11],
       [12],
       [13],
       [14],
       [15],
       [16]])
```



```
# Splitting an Array
arr = np.arange(2,21,2) # 10 elements [2,4...20]
print(arr)
```

```
[ 2  4  6  8 10 12 14 16 18 20]
```

```
split_arr = np.split(arr , 5)
print(split_arr)
```

```
[array([2, 4]), array([6, 8]), array([10, 12]), array([14, 16]), array([18, 20])]
```

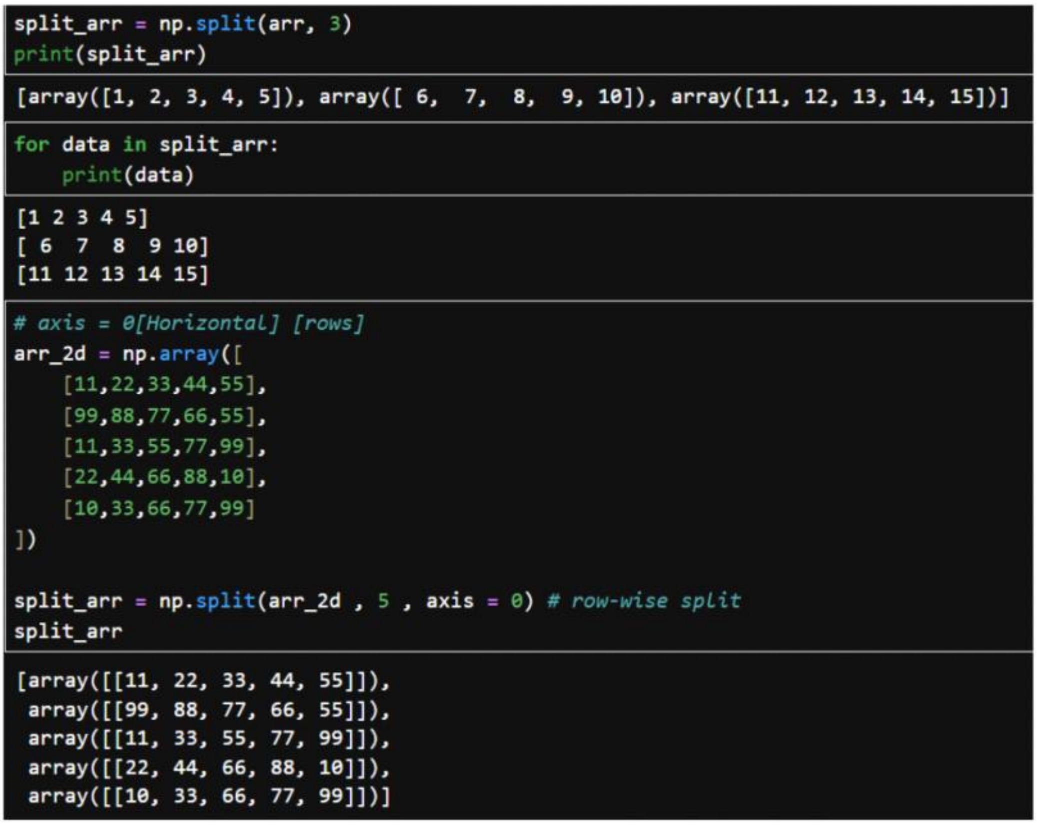
```
for data in split_arr:
    print(data)
```

```
[2 4]
[6 8]
[10 12]
[14 16]
[18 20]
```

```
split_arr = np.split(arr , 3)
print(split_arr)
# ValueError: array split does not result in an equal division
```

```
# Splitting an Array
arr = np.arange(1,16,1) # 15 elements [1,2,3....15]
print(arr)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
```



```
split_arr = np.split(arr, 3)
print(split_arr)
```

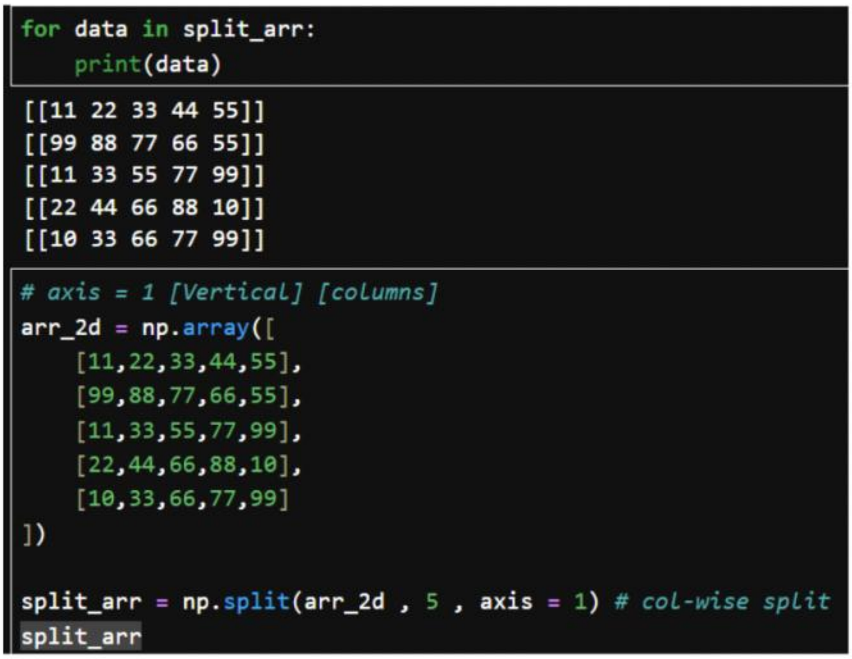
```
[array([1, 2, 3, 4, 5]), array([ 6,  7,  8,  9, 10]), array([11, 12, 13, 14, 15])]
```

```
for data in split_arr:
    print(data)
```

```
[1 2 3 4 5]
[ 6  7  8  9 10]
[11 12 13 14 15]
```

```
# axis = 0[Horizontal] [rows]
arr_2d = np.array([
    [11,22,33,44,55],
    [99,88,77,66,55],
    [11,33,55,77,99],
    [22,44,66,88,10],
    [10,33,66,77,99]
])

split_arr = np.split(arr_2d , 5 , axis = 0) # row-wise split
split_arr
```

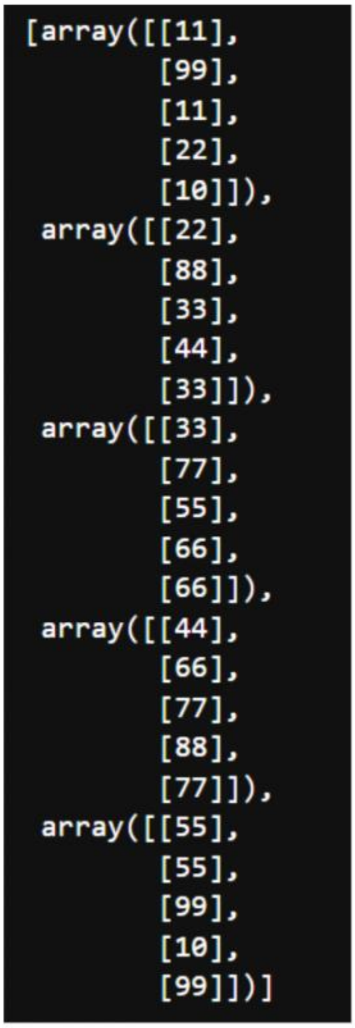
```
[array([[11, 22, 33, 44, 55]]),
 array([[99, 88, 77, 66, 55]]),
 array([[11, 33, 55, 77, 99]]),
 array([[22, 44, 66, 88, 10]]),
 array([[10, 33, 66, 77, 99]])]
```



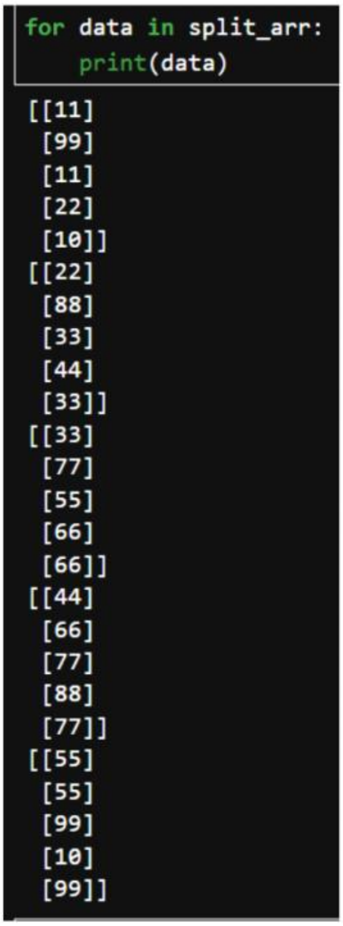
```
for data in split_arr:
    print(data)
```

```
[[11 22 33 44 55]]
[[99 88 77 66 55]]
[[11 33 55 77 99]]
[[22 44 66 88 10]]
[[10 33 66 77 99]]
```

```
# axis = 1 [Vertical] [columns]
arr_2d = np.array([
    [11,22,33,44,55],
    [99,88,77,66,55],
    [11,33,55,77,99],
    [22,44,66,88,10],
    [10,33,66,77,99]
])

split_arr = np.split(arr_2d , 5 , axis = 1) # col-wise split
split_arr
```



```
[array([[11],
        [99],
        [11],
        [22],
        [10]]),
 array([[22],
        [88],
        [33],
        [44],
        [33]]),
 array([[33],
        [77],
        [55],
        [66],
        [66]]),
 array([[44],
        [66],
        [77],
        [88],
        [77]]),
 array([[55],
        [55],
        [99],
        [10],
        [99]])]
```



```
for data in split_arr:
    print(data)
```

```
[[11]
 [99]
 [11]
 [22]
 [10]]
[[22]
 [88]
 [33]
 [44]
 [33]]
[[33]
 [77]
 [55]
 [66]
 [66]]
[[44]
 [66]
 [77]
 [88]
 [77]]
[[55]
 [55]
 [99]
 [10]
 [99]]
```