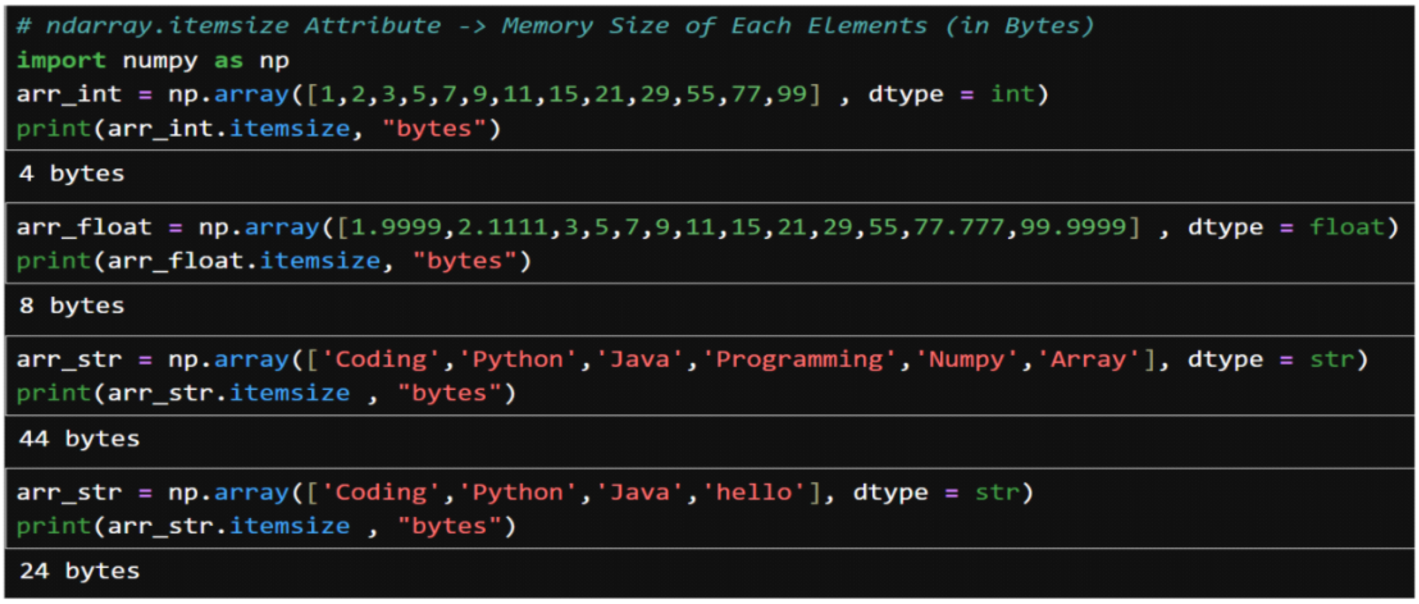
NumPy-II

🎯 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.



```
# ndarray.itemsize Attribute -> Memory Size of Each Elements (in Bytes)
import numpy as np
arr_int = np.array([1,2,3,5,7,9,11,15,21,29,55,77,99] , dtype = int)
print(arr_int.itemsize, "bytes")
```

```
4 bytes
```

```
arr_float = np.array([1.9999,2.1111,3,5,7,9,11,15,21,29,55,77.777,99.9999] , dtype = float)
print(arr_float.itemsize, "bytes")
```

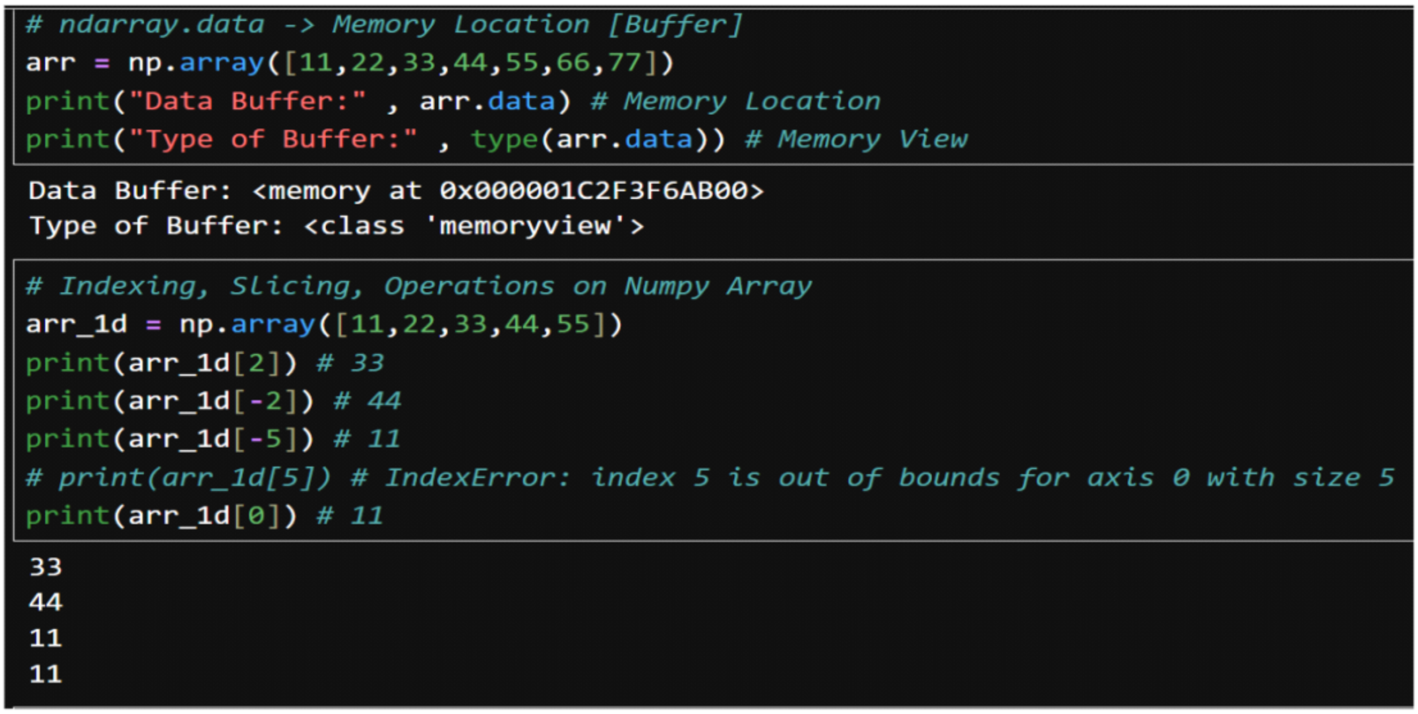
```
8 bytes
```

```
arr_str = np.array(['Coding','Python','Java','Programming','Numpy','Array'], dtype = str)
print(arr_str.itemsize , "bytes")
```

```
44 bytes
```

```
arr_str = np.array(['Coding','Python','Java','hello'], dtype = str)
print(arr_str.itemsize , "bytes")
```

```
24 bytes
```

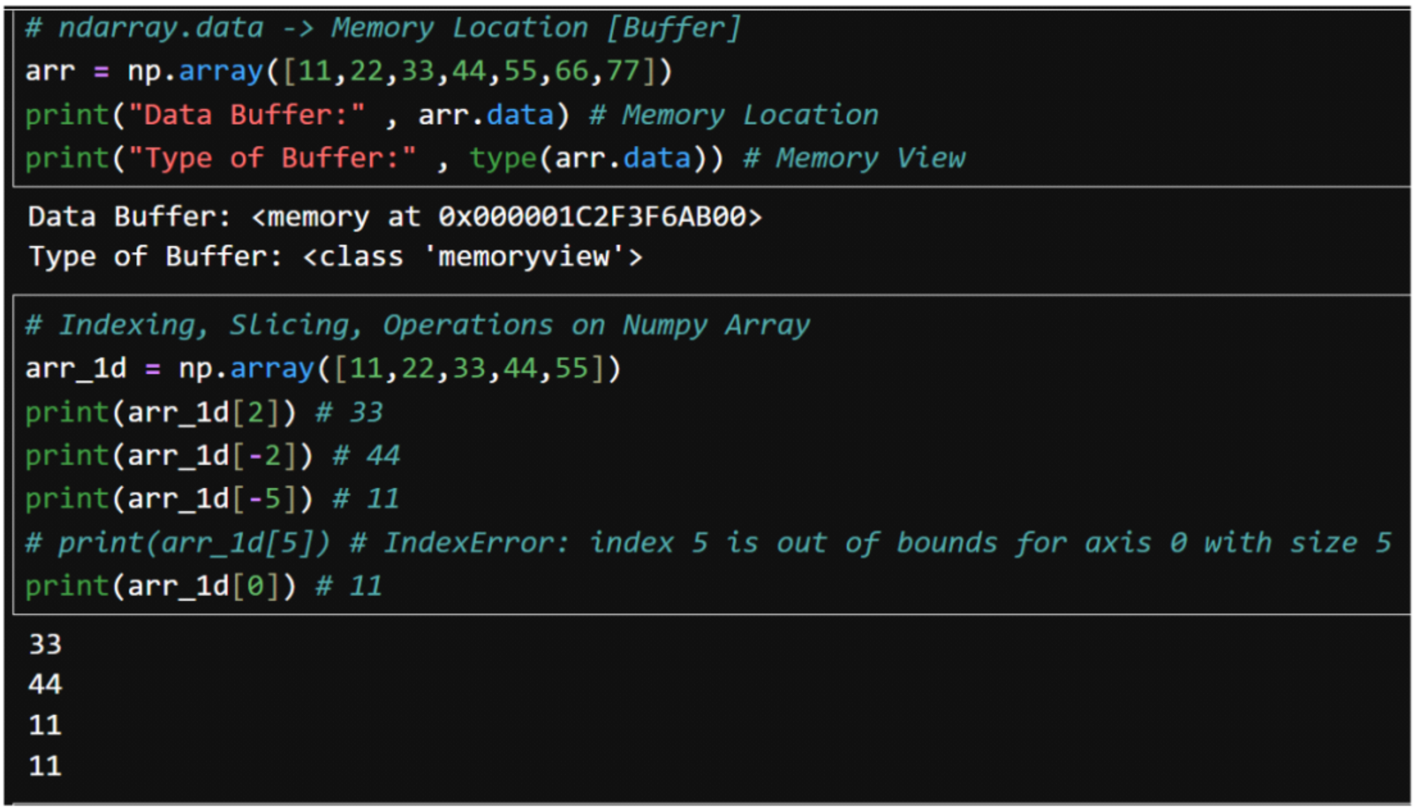


```
# ndarray.data -> Memory Location [Buffer]
arr = np.array([11,22,33,44,55,66,77])
print("Data Buffer:" , arr.data) # Memory Location
print("Type of Buffer:" , type(arr.data)) # Memory View
```

```
Data Buffer: <memory at 0x000001C2F3F6AB00>
Type of Buffer: <class 'memoryview'>
```

```
# Indexing, Slicing, Operations on Numpy Array
arr_1d = np.array([11,22,33,44,55])
print(arr_1d[2]) # 33
print(arr_1d[-2]) # 44
print(arr_1d[-5]) # 11
# print(arr_1d[5]) # IndexError: index 5 is out of bounds for axis 0 with size 5
print(arr_1d[0]) # 11
```

```
33
44
11
11
```

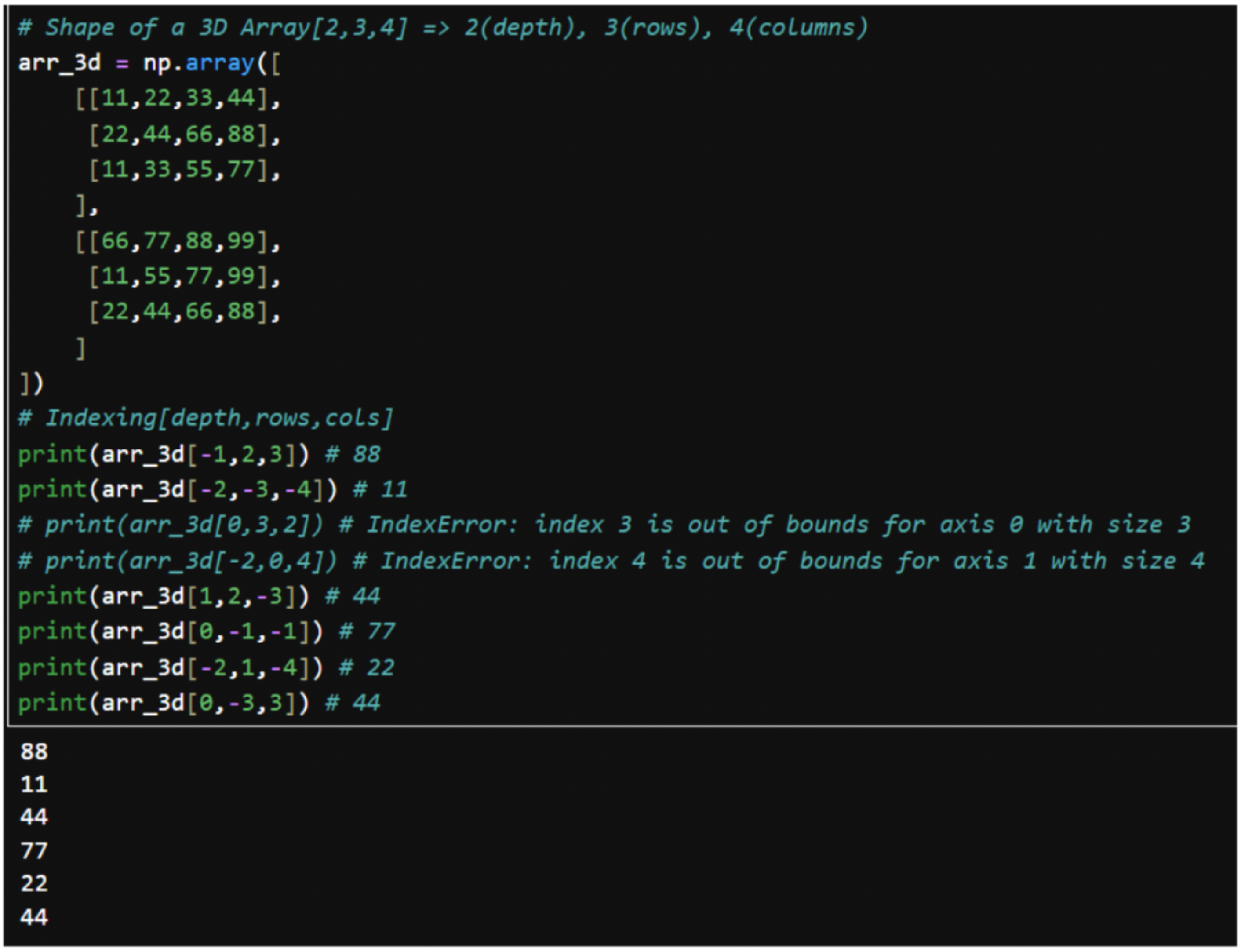


```python
# ndarray.data -> Memory Location [Buffer]
arr = np.array([11,22,33,44,55,66,77])
print("Data Buffer:" , arr.data) # Memory Location
print("Type of Buffer:" , type(arr.data)) # Memory View
```

```
Data Buffer: <memory at 0x000001C2F3F6AB00>
Type of Buffer: <class 'memoryview'>
```

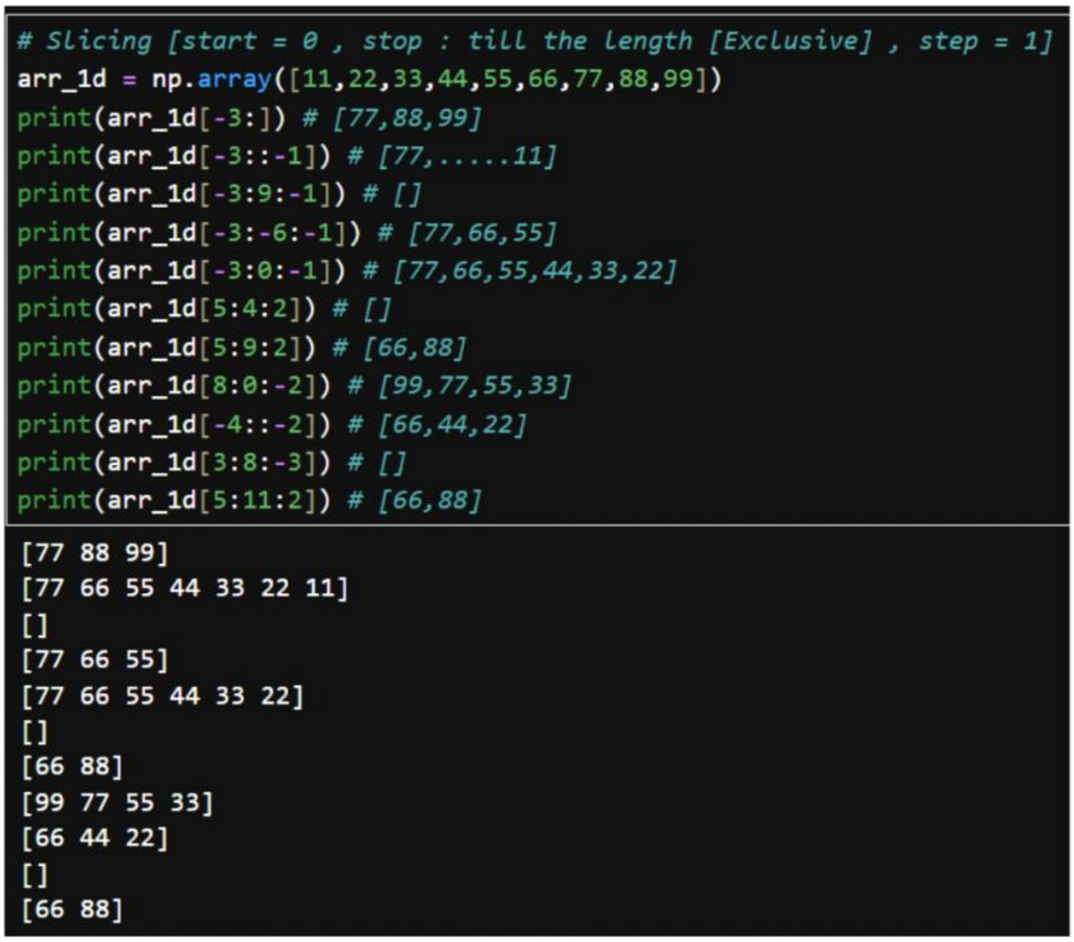
```python
# Indexing, Slicing, Operations on Numpy Array
arr_1d = np.array([11,22,33,44,55])
print(arr_1d[2]) # 33
print(arr_1d[-2]) # 44
print(arr_1d[-5]) # 11
# print(arr_1d[5]) # IndexError: index 5 is out of bounds for axis 0 with size 5
print(arr_1d[0]) # 11
```

```
33
44
11
11
```



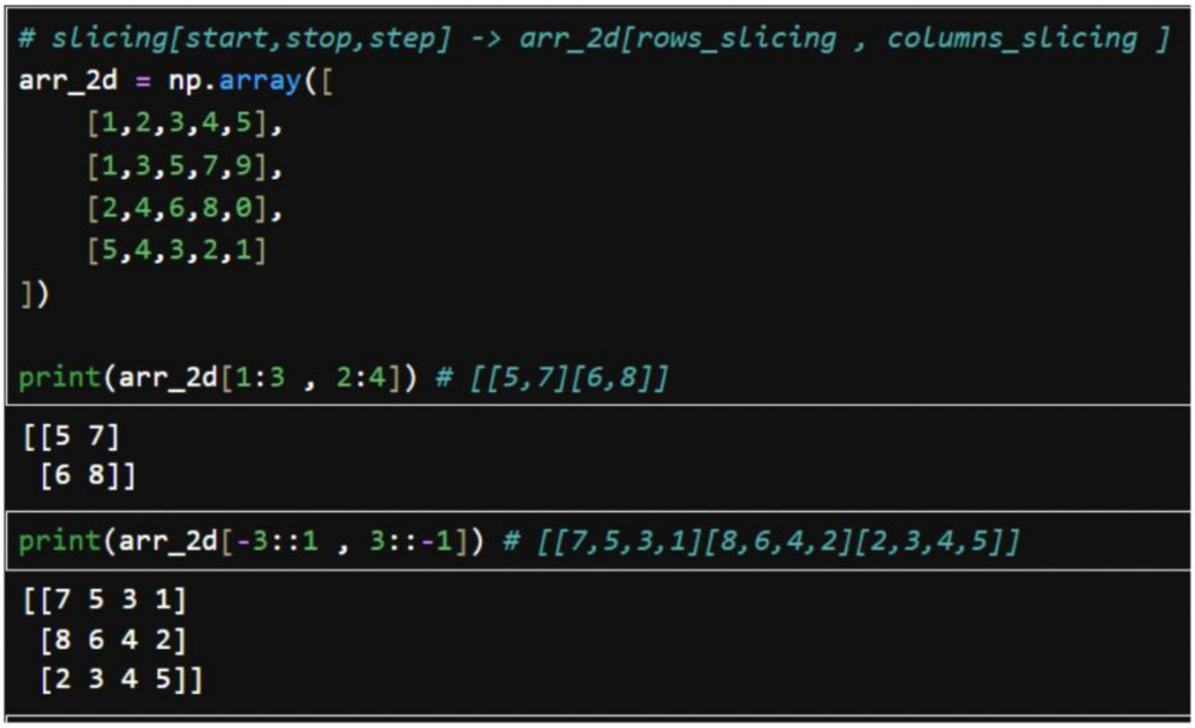
```python
# Shape of a 3D Array[2,3,4] => 2(depth), 3(rows), 4(columns)
arr_3d = np.array([
    [[11,22,33,44],
     [22,44,66,88],
     [11,33,55,77],
    ],
    [[66,77,88,99],
     [11,55,77,99],
     [22,44,66,88],
    ]
])
# Indexing[depth,rows,cols]
print(arr_3d[-1,2,3]) # 88
print(arr_3d[-2,-3,-4]) # 11
# print(arr_3d[0,3,2]) # IndexError: index 3 is out of bounds for axis 0 with size 3
# print(arr_3d[-2,0,4]) # IndexError: index 4 is out of bounds for axis 1 with size 4
print(arr_3d[1,2,-3]) # 44
print(arr_3d[0,-1,-1]) # 77
print(arr_3d[-2,1,-4]) # 22
print(arr_3d[0,-3,3]) # 44
```

```
88
11
44
77
22
44
```



```
# Slicing [start = 0 , stop : till the Length [Exclusive] , step = 1]
arr_1d = np.array([11,22,33,44,55,66,77,88,99])
print(arr_1d[-3:]) # [77,88,99]
print(arr_1d[-3::-1]) # [77,.....11]
print(arr_1d[-3:9:-1]) # []
print(arr_1d[-3:-6:-1]) # [77,66,55]
print(arr_1d[-3:0:-1]) # [77,66,55,44,33,22]
print(arr_1d[5:4:2]) # []
print(arr_1d[5:9:2]) # [66,88]
print(arr_1d[8:0:-2]) # [99,77,55,33]
print(arr_1d[-4::-2]) # [66,44,22]
print(arr_1d[3:8:-3]) # []
print(arr_1d[5:11:2]) # [66,88]
```

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

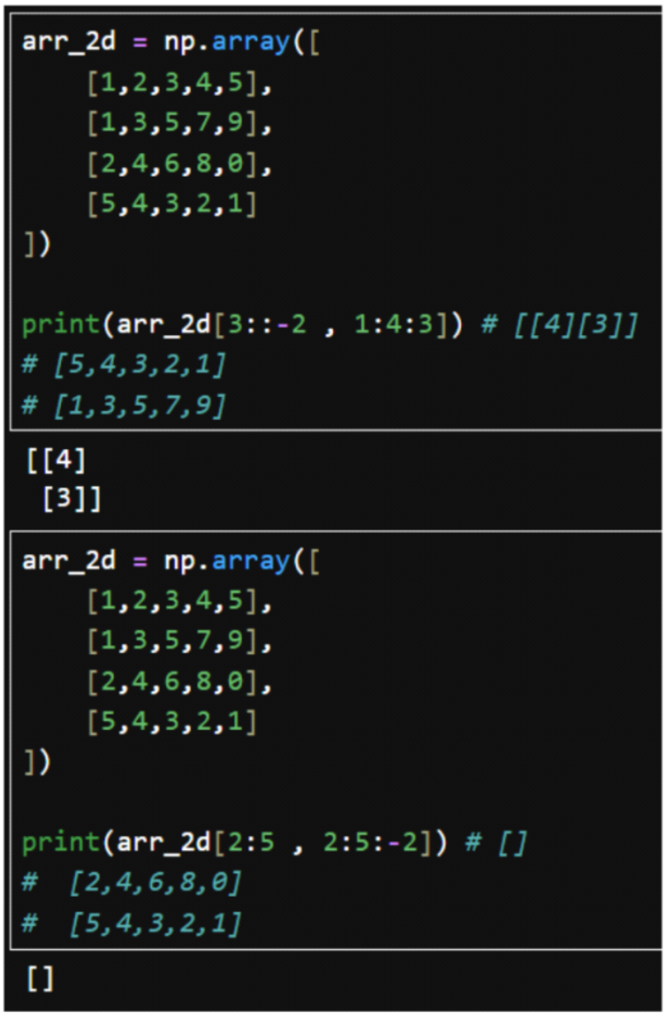


```
# slicing[start,stop,step] -> arr_2d[rows_slicing , columns_slicing ]
arr_2d = np.array([
    [1,2,3,4,5],
    [1,3,5,7,9],
    [2,4,6,8,0],
    [5,4,3,2,1]
])

print(arr_2d[1:3 , 2:4]) # [[5,7][6,8]]
```

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

```
print(arr_2d[-3::1 , 3::-1]) # [[7,5,3,1][8,6,4,2][2,3,4,5]]
```

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

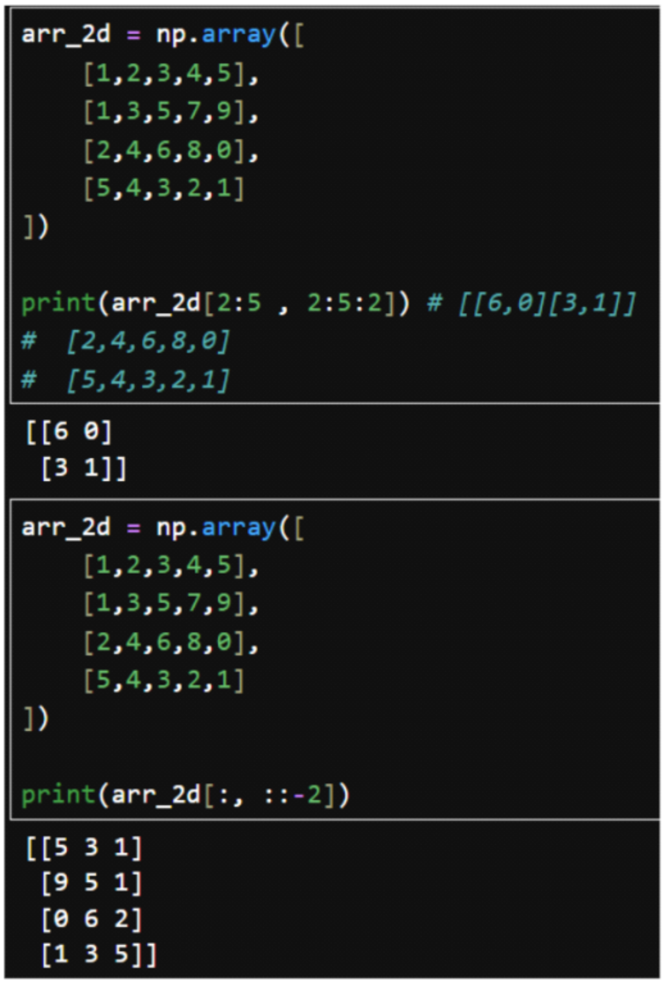


```
arr_2d = np.array([
    [1,2,3,4,5],
    [1,3,5,7,9],
    [2,4,6,8,0],
    [5,4,3,2,1]
])

print(arr_2d[3::-2 , 1:4:3]) # [[4][3]]
# [5,4,3,2,1]
# [1,3,5,7,9]
```

```
[[4]
 [3]]
```

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

print(arr_2d[2:5 , 2:5:-2]) # []
#  [2,4,6,8,0]
#  [5,4,3,2,1]
```

```
[]
```

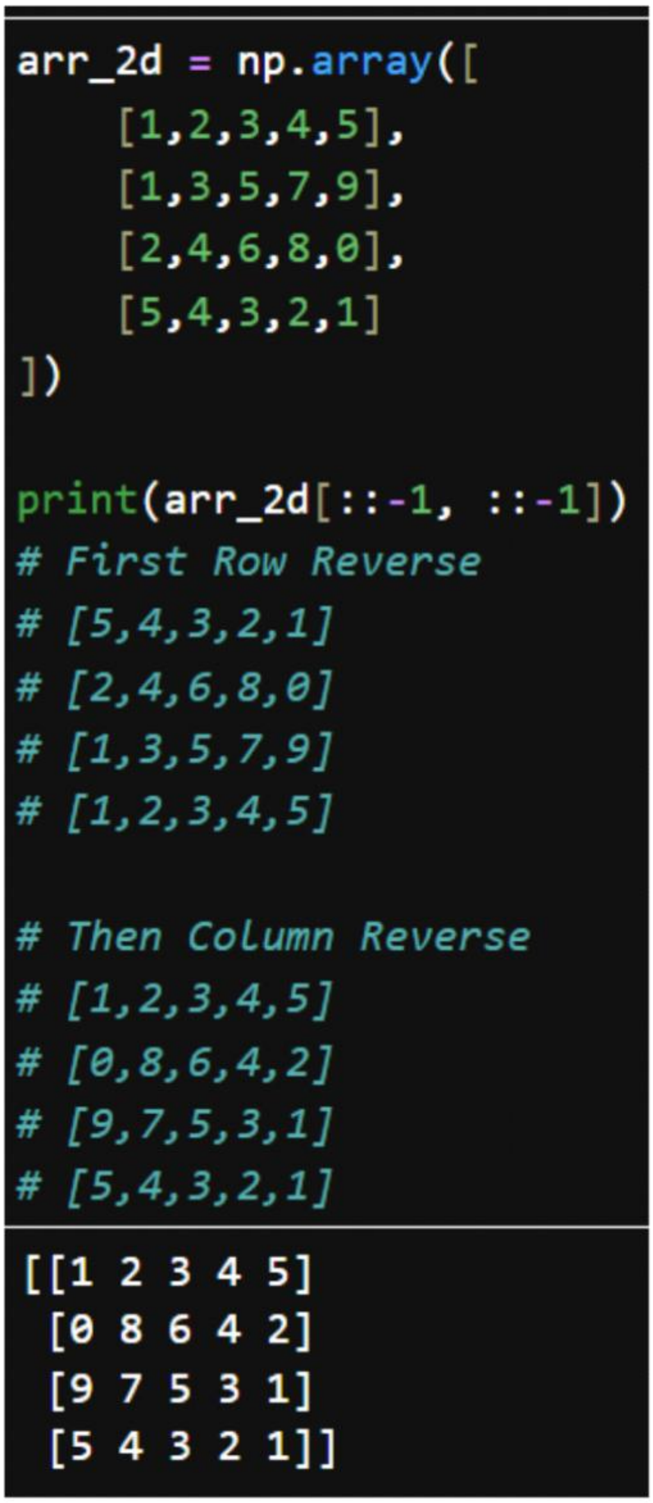


```
arr_2d = np.array([
    [1,2,3,4,5],
    [1,3,5,7,9],
    [2,4,6,8,0],
    [5,4,3,2,1]
])

print(arr_2d[2:5 , 2:5:2]) # [[6,0][3,1]]
#  [2,4,6,8,0]
#  [5,4,3,2,1]
```

```
[[6 0]
 [3 1]]
```

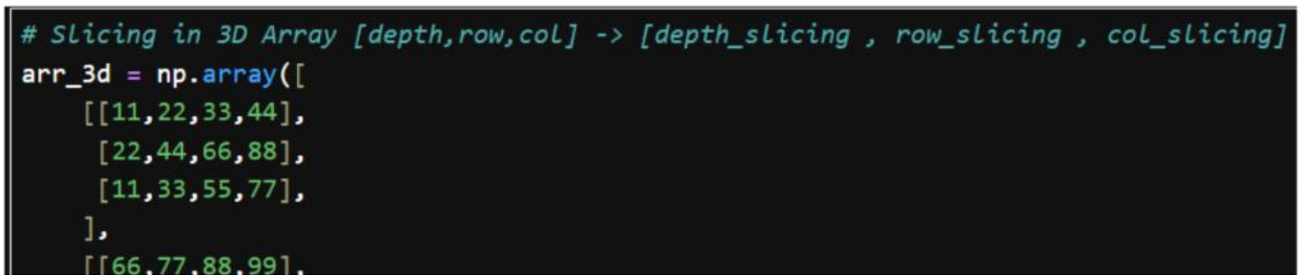
```
arr_2d = np.array([
    [1,2,3,4,5],
    [1,3,5,7,9],
    [2,4,6,8,0],
    [5,4,3,2,1]
])

print(arr_2d[:, ::-2])
```

```
[[5 3 1]
 [9 5 1]
 [0 6 2]
 [1 3 5]]
```

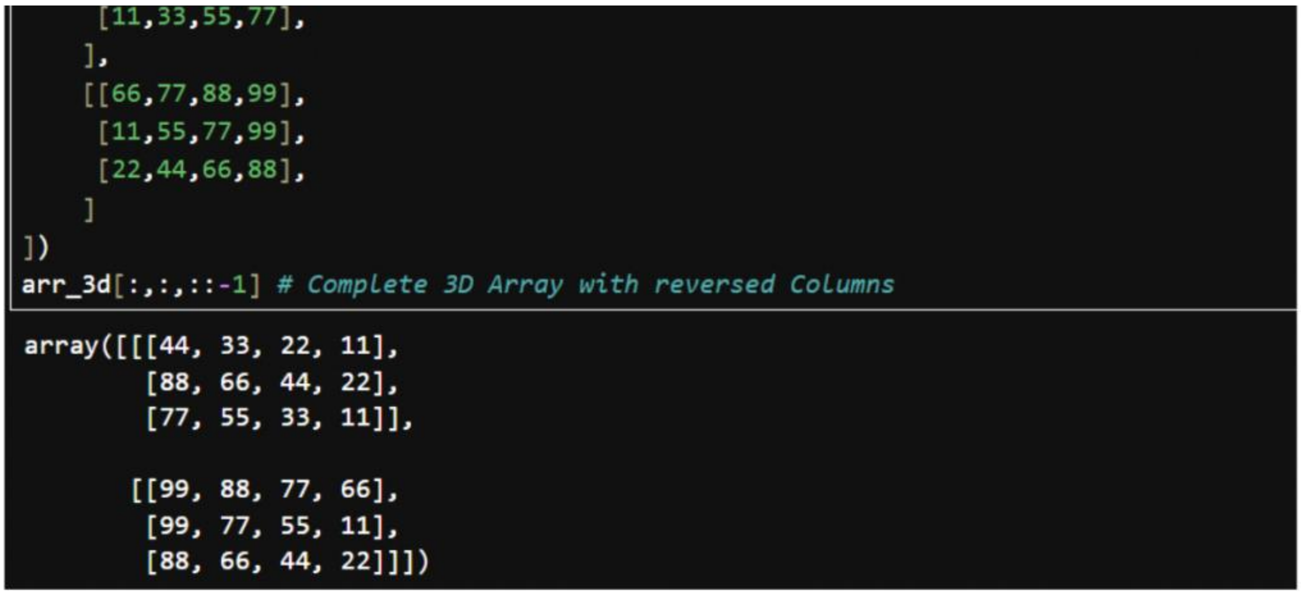


```
arr_2d = np.array([
    [1,2,3,4,5],
    [1,3,5,7,9],
    [2,4,6,8,0],
    [5,4,3,2,1]
])

print(arr_2d[::-1, ::-1])
# First Row Reverse
# [5,4,3,2,1]
# [2,4,6,8,0]
# [1,3,5,7,9]
# [1,2,3,4,5]

# Then Column Reverse
# [1,2,3,4,5]
# [0,8,6,4,2]
# [9,7,5,3,1]
# [5,4,3,2,1]
```

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

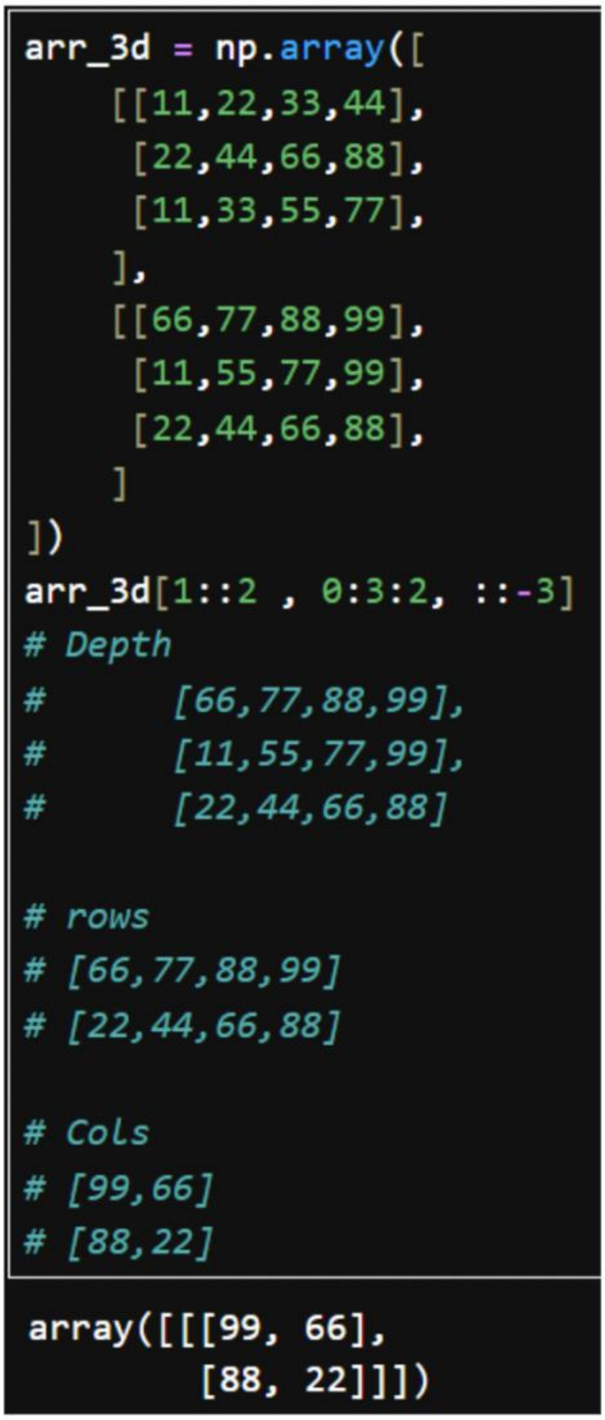


```
# Slicing in 3D Array [depth,row,col] -> [depth_slicing , row_slicing , col_slicing]
arr_3d = np.array([
    [[11,22,33,44],
     [22,44,66,88],
     [11,33,55,77],
    ],
    [[66,77,88,99],
```



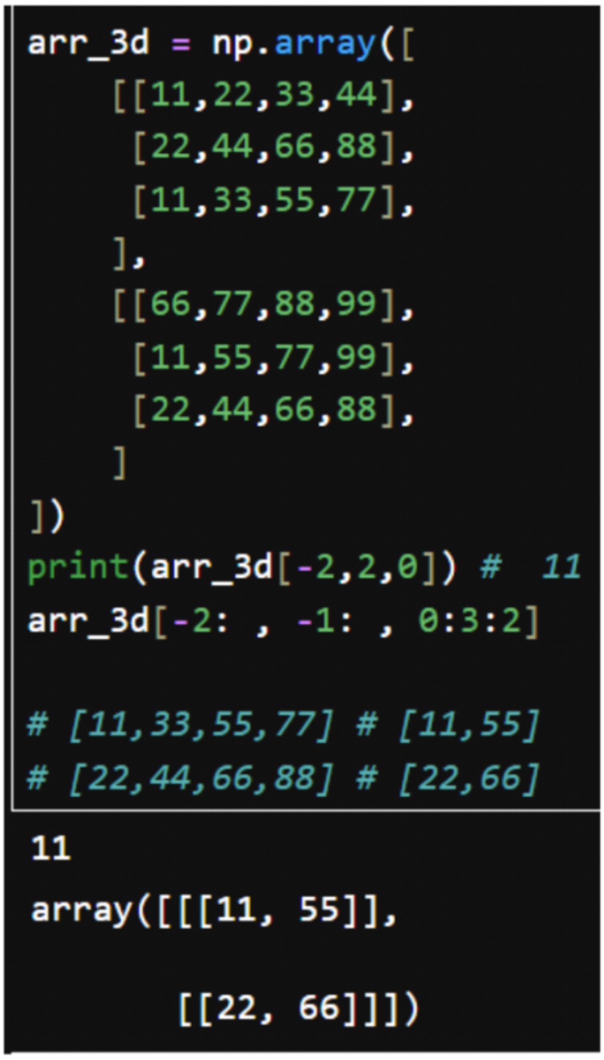
```
     [11,33,55,77],
    ],
    [[66,77,88,99],
     [11,55,77,99],
     [22,44,66,88],
    ]
])
arr_3d[:,:,::-1] # Complete 3D Array with reversed Columns
```

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

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



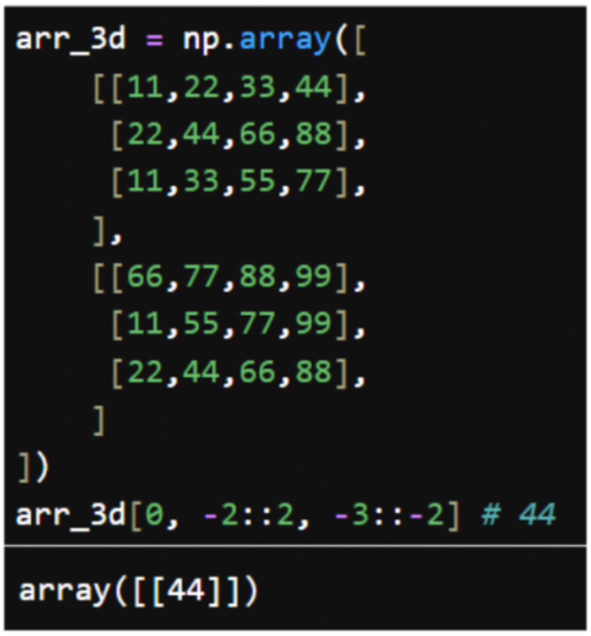
```
arr_3d = np.array([
    [[11,22,33,44],
     [22,44,66,88],
     [11,33,55,77],
    ],
    [[66,77,88,99],
     [11,55,77,99],
     [22,44,66,88],
    ]
])
arr_3d[1::2 , 0:3:2, ::-3]
# Depth
#       [66,77,88,99],
#       [11,55,77,99],
#       [22,44,66,88]

# rows
# [66,77,88,99]
# [22,44,66,88]

# Cols
# [99,66]
# [88,22]
```

```
array([[[99, 66],
        [88, 22]]])
```



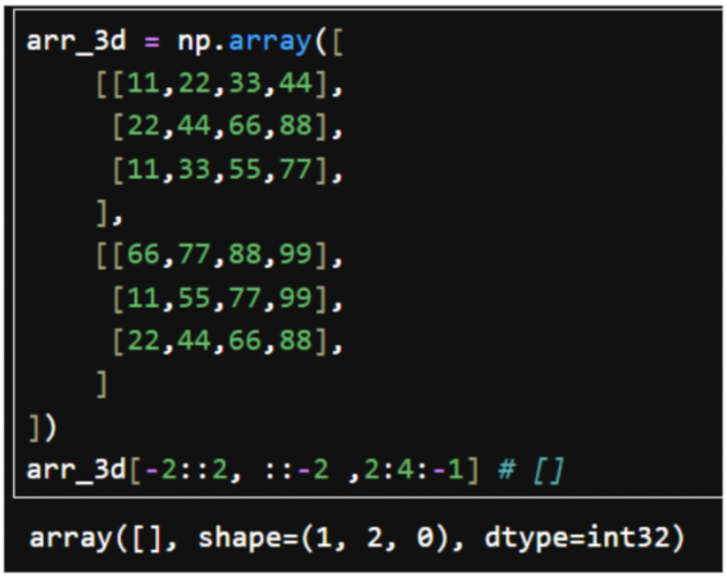
```
arr_3d = np.array([
    [[11,22,33,44],
     [22,44,66,88],
     [11,33,55,77],
    ],
    [[66,77,88,99],
     [11,55,77,99],
     [22,44,66,88],
    ]
])
print(arr_3d[-2,2,0]) #  11
arr_3d[-2: , -1: , 0:3:2]

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

```
11
array([[[11, 55]],

       [[22, 66]]])
```



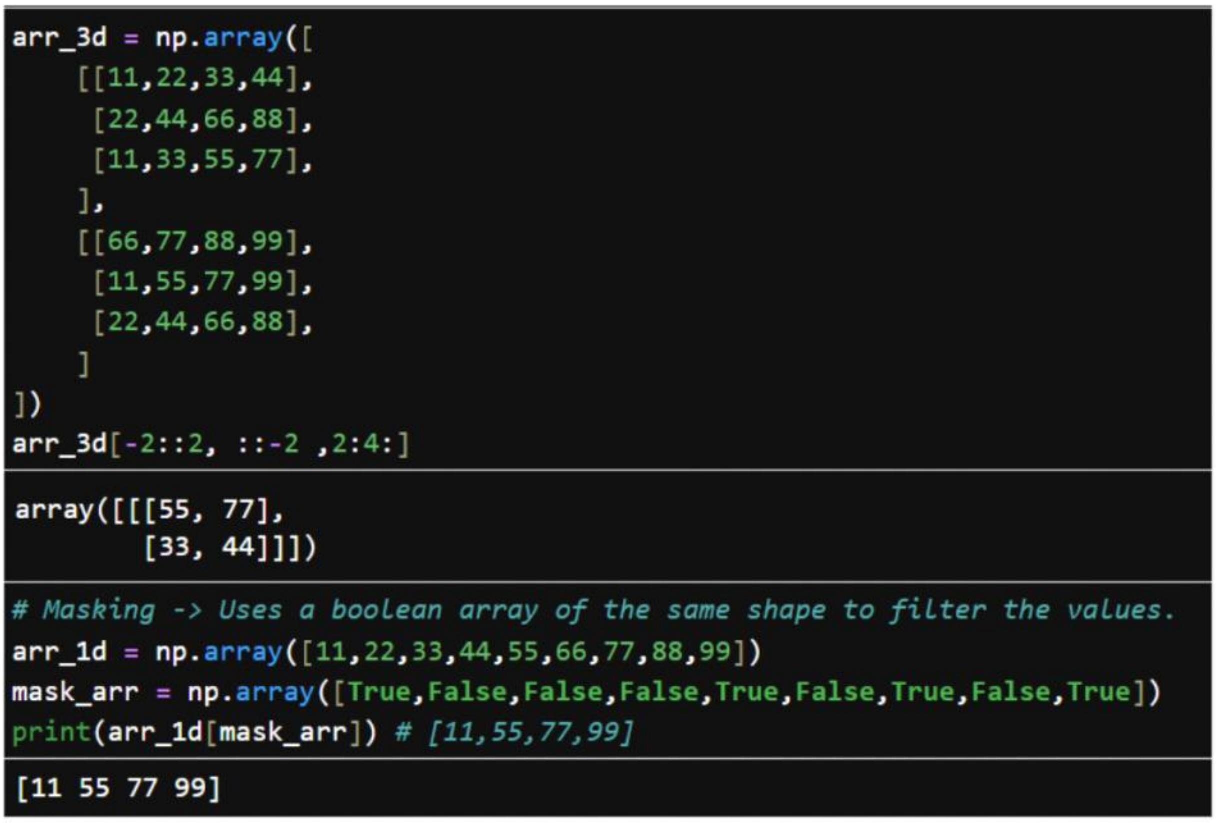
```
arr_3d = np.array([
    [[11,22,33,44],
     [22,44,66,88],
     [11,33,55,77],
    ],
    [[66,77,88,99],
     [11,55,77,99],
     [22,44,66,88],
    ]
])
arr_3d[0, -2::2, -3::-2] # 44
```

```
array([[44]])
```



```
arr_3d = np.array([
    [[11,22,33,44],
     [22,44,66,88],
     [11,33,55,77],
    ],
    [[66,77,88,99],
     [11,55,77,99],
     [22,44,66,88],
    ]
])
arr_3d[-2::2, ::-2 ,2:4:-1] # []
```

```
array([], shape=(1, 2, 0), dtype=int32)
```

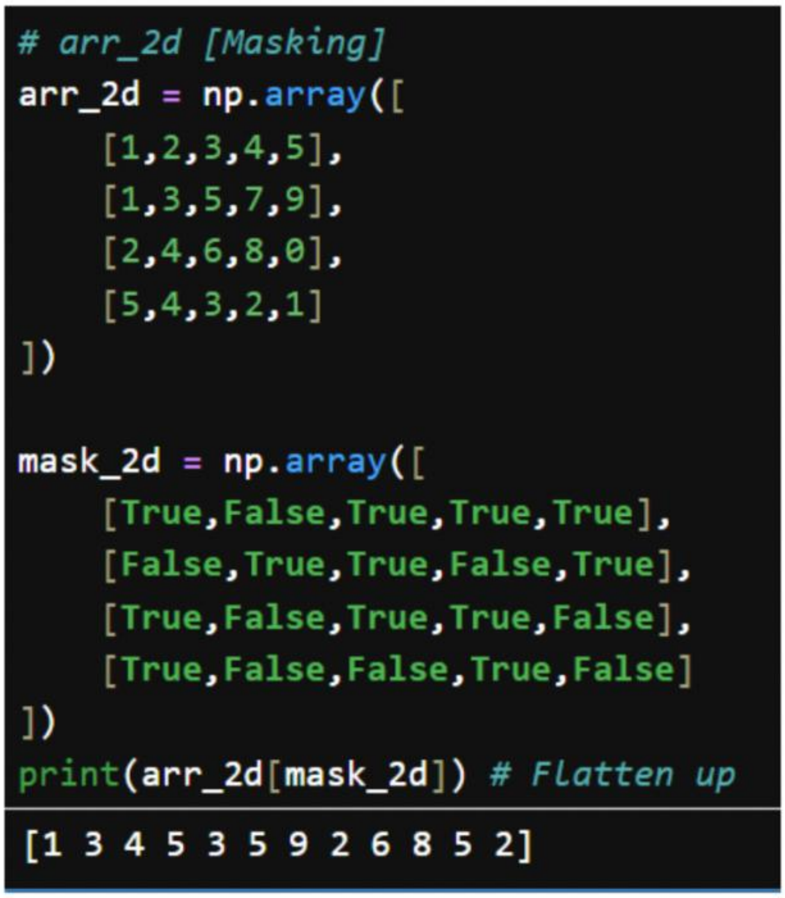


```
arr_3d = np.array([
    [[11,22,33,44],
     [22,44,66,88],
     [11,33,55,77],
    ],
    [[66,77,88,99],
     [11,55,77,99],
     [22,44,66,88],
    ]
])
arr_3d[-2::2, ::-2 ,2:4:]
```

```
array([[[55, 77],
        [33, 44]]])
```

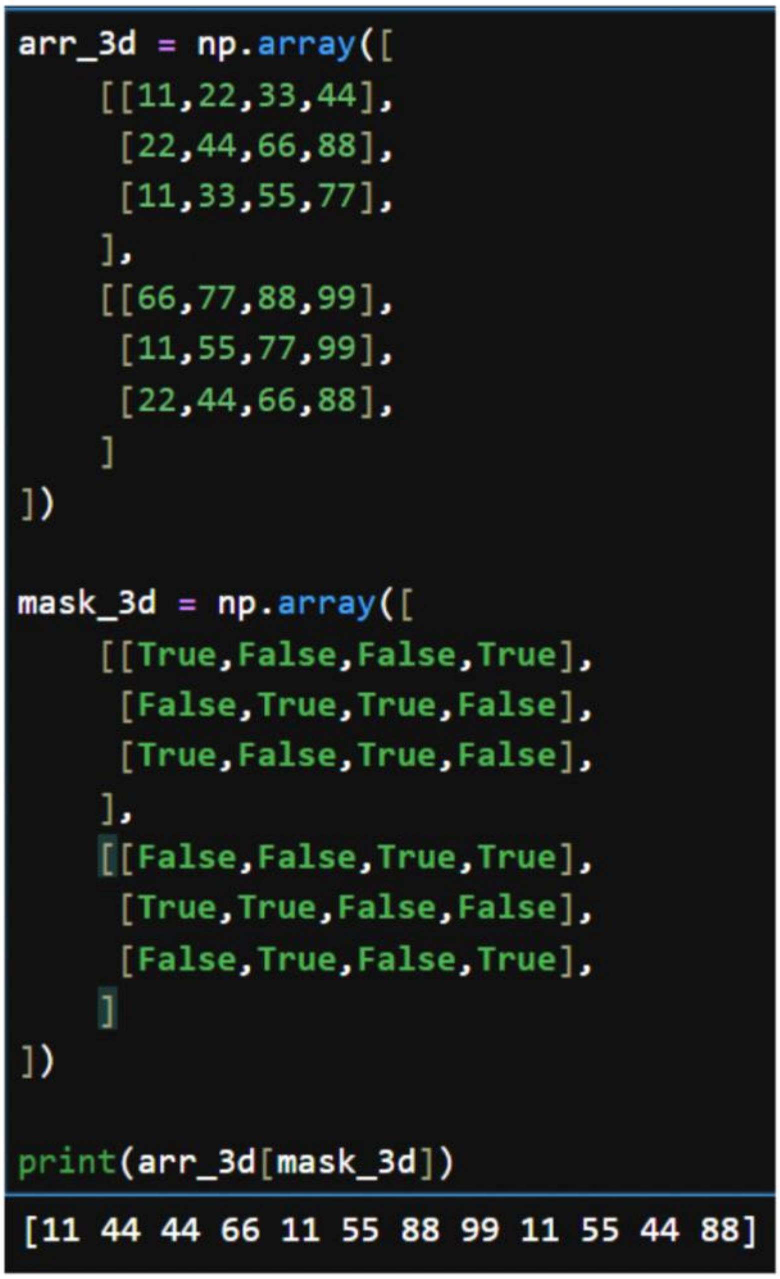
```
# Masking -> Uses a boolean array of the same shape to filter the values.
arr_1d = np.array([11,22,33,44,55,66,77,88,99])
mask_arr = np.array([True,False,False,False,True,False,True,False,True])
print(arr_1d[mask_arr]) # [11,55,77,99]
```

```
[11 55 77 99]
```



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

mask_2d = np.array([
    [True,False,True,True,True],
    [False,True,True,False,True],
    [True,False,True,True,False],
    [True,False,False,True,False]
])
print(arr_2d[mask_2d]) # Flatten up
```

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



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

mask_3d = np.array([
    [[True,False,False,True],
     [False,True,True,False],
     [True,False,True,False],
    ],
    [[False,False,True,True],
     [True,True,False,False],
     [False,True,False,True],
    ]
])

print(arr_3d[mask_3d])
```

```
[11 44 44 66 11 55 88 99 11 55 44 88]
```