

Slicing

To select only a part (or subset) of an array is called slicing.

- The slice notation specifies a start and end value **[start:end]** and copies the list from start up to but not including end.
- We can omit the start, in which case the slice start at the beginning of the list using **[:n]**
- We can omit the end, so the slice continues to the end of the list. **[n:]**
- If we omit both the slice created is a copy of the entire list. **[:]**

▼ Slicing a One-dimensional Array

- Slicing a 1D numpy array is almost exactly the same as slicing a list

Example 1:

```
#Using start and end value [start:end]

import numpy as np

array1 = np.arange(9)
print(array1)

print(array1[3:6])
```

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

Index '3' represents the starting element of the slice and it's inclusive.

Index '6' represents the stopping element of the slice and it's exclusive. That's the reason why we did not get the value '6' in the output.

Example 2:

```
#Writing array1[:] is equivalent to writing array1[0:9]

print(array1[:])
```

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

If you do not specify the starting and the stopping index you will get all the values. That's because if the indices are missing, by default, Numpy inserts the starting and stopping indices that select the entire array.

Example 3:

```
#using [n:]  
  
print(array1[4:])  
  
[4 5 6 7 8]
```

- To include only the starting index.

In this case, the slice includes all the elements from the starting index until the end of the array.

Example 4:

```
#using [:n]  
  
print(array1[:7])  
  
[0 1 2 3 4 5 6]
```

- To specify only the stopping index.

Then the slice includes all the elements from the start of the array until the indexed value.

Example 5:

- To modify an item in an array

```
import numpy as np  
  
a1 = np.array([1, 2, 3, 4, 5])  
print(a1)  
  
b = a1[1:4]  
print(b)  
  
b[1] = 10  
print(b)  
  
print(a1)  
  
[1 2 3 4 5]  
[2 3 4]  
[ 2 10  4]  
[ 1  2 10  4  5]
```

Example 6: (Using Step Index)

- To include a step index enables to skip a few elements in the slice operation.

For example:

‘2:6’ indicate the index positions for the slice operation. The value ‘3’ indicates the slice operation to step three elements after every selection.

```
print(array1[2:10:2])
```

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

Here, ‘8’ means slice from ‘0:8’ and the last value ‘2’ indicates a step operation to step two elements after every selection.

```
print(array1[:8:2])
```

```
[0 2 4 6]
```

.

▼ Slicing a 2-dimensional Array

- Slicing a two-dimensional array is very similar to slicing a one-dimensional array.
- We can slice a 2D array in both row and column axes to obtain a rectangular subset of the original array.
- We use a comma to separate the row slice and the column slice.

```
array1 = np.arange(16).reshape(4,4)
print(array1)
print("\n")
```

```
print(array1[0:2, 1:3])
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

```
[[1 2]
 [5 6]]
```

Here is a pictorial representation:

	0	1	2	3	
0	0	1	2	3	Row slice
1	4	5	6	7	
2	8	9	10	11	
3	12	13	14	15	
	Column slice				

```
import numpy as np
```

```
a2 = np.array([[10, 11, 12, 13, 14],
               [15, 16, 17, 18, 19],
               [20, 21, 22, 23, 24],
               [25, 26, 27, 28, 29]])
```

```
print(a2)
print("\n")
```

```
print(a2[1:,2:4])
```

```
[[10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]
 [25 26 27 28 29]]
```

```
[[17 18]
 [22 23]
 [27 28]]
```

This selects rows 1: (1 to the end of bottom of the array) and columns 2:4 (columns 2 and 3), as shown here:

	10	11	12	13	14
1	15	16	17	18	19
2	20	21	22	23	24
3	25	26	27	28	29

▼ Slicing a 3-Dimensional Array

You can slice a 3D array in all 3 axes to obtain a cuboid subset of the original array:

The slicing values are given in the following order:

1. layers - To select a layer of the 3D array
2. rows - To select the row from the previously selected layer.
3. columns - To select the columns from the previously selected rows.

```
import numpy as np

a3 = np.array([[[10, 11, 12], [13, 14, 15], [16, 17, 18]],
               [[20, 21, 22], [23, 24, 25], [26, 27, 28]],
               [[30, 31, 32], [33, 34, 35], [36, 37, 38]]])

print(a3)
print("\n")
print("\n")

print(a3[:2,1:,:2])
print("\n")
```

```
[[[10 11 12]
   [13 14 15]
   [16 17 18]]

  [[20 21 22]
   [23 24 25]
   [26 27 28]]

  [[30 31 32]
   [33 34 35]
   [36 37 38]]]
```

```
[[[13 14]
   [16 17]]

  [[23 24]
   [26 27]]]
```

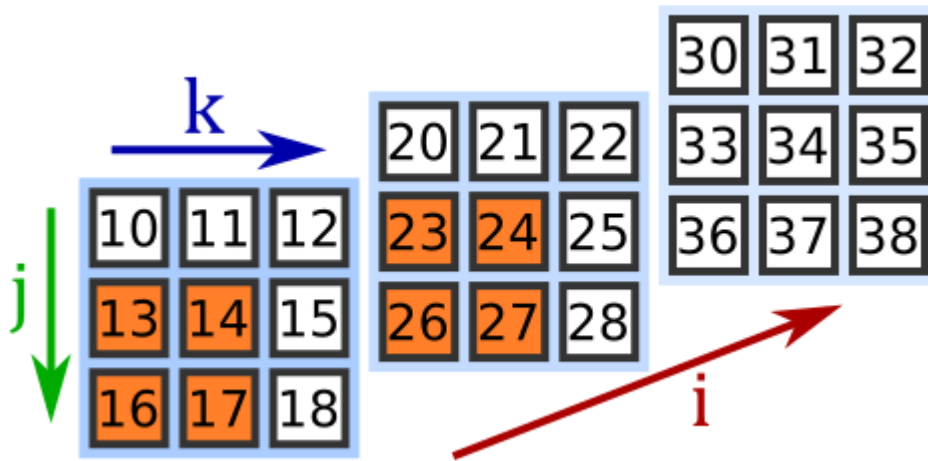
This selects:

layers :2 (the first 2 layers)

rows 1: (the last 2 rows)

columns :2 (the first 2 columns)

As shown here:



```
print(a3[:, :, :])  
print("\n")
```

```
[[[10 11 12]  
  [13 14 15]  
  [16 17 18]]  
  
 [[20 21 22]  
  [23 24 25]  
  [26 27 28]]  
  
 [[30 31 32]  
  [33 34 35]  
  [36 37 38]]]
```

```
print(a3[1:, :2, :]) # is the same as  
print("\n")
```

```
print(a3[1:, :2])  
print("\n")
```

```
[[[20 21 22]  
  [23 24 25]]  
  
 [[30 31 32]  
  [33 34 35]]]  
  
 [[20 21 22]  
  [23 24 25]]  
  
 [[30 31 32]  
  [33 34 35]]]
```

```
print(a3[1:, :, :]) # is the same as
```

```
print("\n")
print("\n")

print(a3[1:,:]) # and is also the same as
print("\n")
print("\n")

print(a3[1:])
print("\n")
print("\n")
```

```
[[[20 21 22]
  [23 24 25]
  [26 27 28]]

 [[30 31 32]
  [33 34 35]
  [36 37 38]]]
```

```
[[[20 21 22]
  [23 24 25]
  [26 27 28]]

 [[30 31 32]
  [33 34 35]
  [36 37 38]]]
```

```
[[[20 21 22]
  [23 24 25]
  [26 27 28]]

 [[30 31 32]
  [33 34 35]
  [36 37 38]]]
```

```
print(a3[1:3,:]) # is the same as
print("\n")
```

```
print(a3[1:3])
```

```
[[[20 21 22]
  [23 24 25]
  [26 27 28]]

 [[30 31 32]
  [33 34 35]
  [36 37 38]]]
```

```
[[[20 21 22]
   [23 24 25]
   [26 27 28]]

 [[30 31 32]
  [33 34 35]
  [36 37 38]]]
```

▼ Additional Points

Slicing using Negative Indexes

- To understand how negative values work, take a look at this picture below:

10	20	30	40	50	60	70	80	90
0	1	2	3	4	5	6	7	8
0	-8	-7	-6	-5	-4	-3	-2	-1



Each element of an array can be referenced with two indices. For example, both '3' and '-6' can be used to retrieve the value '40.' First let's declare an array with similar values:

```
array1 = np.array([10,20,30,40,50,60,70,80,90])
print(array1)

print(array1[-6])
```

```
[10 20 30 40 50 60 70 80 90]
40
```

use a negative value for both the indices.

									
10	20	30	40	50	60	70	80	90	
0	1	2	3	4	5	6	7	8	
0	-8	-7	-6	-5	-4	-3	-2	-1	
									

```
print(array1[-3:-1])
```

```
[70 80]
```

But note that you cannot change the order of the indices. In other words, the slice operation cannot travel backwards. If you try to do that, you will get an empty array as the output.

```
print(array1[-1:-3])
```

```
[]
```