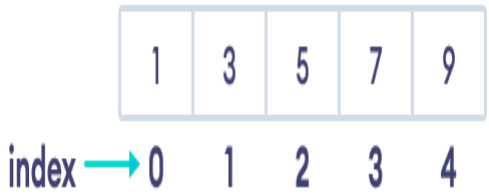
Numpy Array Indexing

In NumPy, each element in an array is associated with a number. The number is known as an **array index**.

Let's see an example to demonstrate NumPy array indexing.



Array Indexing in NumPy

In the above array, 5 is the 3rd element. However, its index is 2.

This is because the array indexing starts from 0, that is, the first element of the array has index 0, the second element has index 1, and so on.

Now, we'll see how we can access individual items from the array using the index number.

Access Array Elements Using Index

We can use indices to access individual elements of a NumPy array.

Suppose we have a NumPy array:

```
array1 = np.array([1, 3, 5, 7, 9])
```

Now, we can use the index number to access array elements as:

- `array1[0]` - to access the first element, i.e. **1**
- `array1[2]` - to access the third element, i.e. **5**
- `array1[4]` - to access the fifth element, i.e. **9**

Example: Access Array Elements Using Index

```
import numpy as np

array1 = np.array([1, 3, 5, 7, 9])

# access numpy elements using index
print(array1[0])    # prints 1
print(array1[2])    # prints 5
print(array1[4])    # prints 9
```

Note: Since the last element of `array1` is at index **4**, if we try to access the element beyond that, say index **5**, we will get an index error: `IndexError: index 5 is out of bounds for axis 0 with size 5`

Modify Array Elements Using Index

We can use indices to change the value of an element in a NumPy array. For example,

```
import numpy as np

# create a numpy array
numbers = np.array([2, 4, 6, 8, 10])

# change the value of the first element
numbers[0] = 12
print("After modifying first element:",numbers)    # prints [12 4 6 8 10]

# change the value of the third element
numbers[2] = 14
print("After modifying third element:",numbers)
# prints [12 4 14 8 10]
```

Output

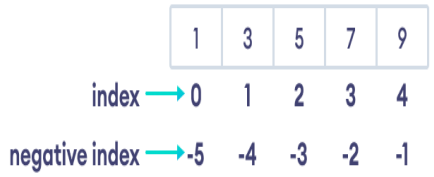
```
After modifying first element: [12  4  6  8 10]
After modifying third element: [12  4 14  8 10]
```

In the above example, we have modified elements of the `numbers` array using array indexing.

- `numbers[0] = 12` - modifies the first element of `numbers` and sets its value to **12**
- `numbers[2] = 14` - modifies the third element of `numbers` and sets its value to **14**

NumPy Negative Array Indexing

NumPy allows negative indexing for its array. The index of **-1** refers to the last item, **-2** to the second last item and so on.



NumPy Array Negative Indexing

Let's see an example.

```
import numpy as np

# create a numpy array
numbers = np.array([1, 3, 5, 7, 9])

# access the last element
print(numbers[-1])    # prints 9

# access the second-to-last element
print(numbers[-2])    # prints 7
```

Output

```
9
7
```

Modify Array Elements Using Negative Indexing

Similar to regular indexing, we can also modify array elements using negative indexing. For example,

```
import numpy as np

# create a numpy array
```

```
numbers = np.array([2, 3, 5, 7, 11])

# modify the last element
numbers[-1] = 13
print(numbers)    # Output: [2 3 5 7 13]

# modify the second-to-last element
numbers[-2] = 17
print(numbers)    # Output: [2 3 5 17 13]
```

Here, `numbers[-1] = 13` modifies the last element to **13** and `numbers[-2] = 17` modifies the second-to-last element to **17**.

Note: Unlike regular indexing, negative indexing starts from **-1** (not **0**) and it starts counting from the end of the array.

2-D NumPy Array Indexing

Array indexing in NumPy allows us to access and manipulate elements in a 2-D array.

To access an element of `array1`, we need to specify the row index and column index of the element.

Suppose we have following 2-D array,

```
array1 = np.array([[1, 3, 5],
                   [7, 9, 2],
                   [4, 6, 8]])
```

Now, say we want to access the element in the third row and second column we specify the index as:

```
array1[2, 1] # returns 6
```

Since we know indexing starts from **0**. So to access the element in the third row and second column, we need to use index **2** for the third row and index **1** for the second column respectively.

Example: 2-D NumPy Array Indexing

```
import numpy as np

# create a 2D array
array1 = np.array([[1, 3, 5, 7],
                   [9, 11, 13, 15],
                   [2, 4, 6, 8]])


# access the element at the second row and fourth column
element1 = array1[1, 3]  # returns 15
print("4th Element at 2nd Row:",element1)

# access the element at the first row and second column
element2 = array1[0, 1]  # returns 3
print("2nd Element at First Row:",element2)
```

Output

```
4th Element at 2nd Row: 15
2nd Element at First Row: 3
```

Access Row or Column of 2D Array Using Indexing

In NumPy, we can access specific rows or columns of a 2-D array using array indexing.

Let's see an example.

```
import numpy as np

# create a 2D array
array1 = np.array([[1, 3, 5],
                   [7, 9, 2],
                   [4, 6, 8]])

# access the second row of the array
second_row = array1[1, :]
print("Second Row:", second_row)  # Output: [7 9 2]

# access the third column of the array
third_col = array1[:, 2]
print("Third Column:", third_col)  # Output: [5 2 8]
```

Output

```
Second Row: [7 9 2]
Third Column: [5 2 8]
```

Here,

- `array1[1, :]` - access the second row of `array1`
- `array1[:, 2]` - access the third column of `array1`

3-D NumPy Array Indexing

We learned how to access elements in a 2D array. We can also access elements in higher dimensional arrays.

To access an element of a 3D array, we use **three indices** separated by commas.

- The **first index** refers to the slice
- The **second index** refers to the row
- The **third index** refers to the column.

Note: In 3D arrays, slice is a 2D array that is obtained by taking a subset of the elements in one of the dimensions.

Let's see an example.

```
import numpy as np

# create a 3D array with shape (2, 3, 4)
array1 = np.array([[[1, 2, 3, 4],
                    [5, 6, 7, 8],
                    [9, 10, 11, 12]],

                   [[13, 14, 15, 16],
                    [17, 18, 19, 20],
                    [21, 22, 23, 24]]])

# access a specific element of the array
element = array1[1, 2, 1]
```

```
# print the value of the element
print(element)

# Output: 22
```

Here, we created a 3D array called `array1` with shape **(2, 3, 4)**. This array contains **2** 2D arrays, each with **3** rows and **4** columns.
Then, we used indexing to access a specific element of `array1`. Notice the code,

```
array1[1, 2, 1]
```

Here,

- `array1[1, , ,]` - access the second 2D array, i.e.

```
[13, 14, 15, 16],
[17, 18, 19, 20],
[21, 22, 23, 24]
```

- `array1[ ,2, ]` - access the third row of the 2D array, i.e.

```
[21, 22, 23, 24]
```

- `array1[ , ,1]` - access the second element of the third row, i.e.

```
[22]
```