How do you shift elements in a NumPy array?

Authored by

stats writer

July 1, 2024

NumPy is a popular library for scientific computing in Python that allows for efficient manipulation and analysis of multi-dimensional arrays. One common operation in NumPy is shifting elements within an array, which involves rearranging the order of elements in the array along a given axis. This can be achieved using the np.roll() function, which takes in the array and the number of positions to shift the elements by. The elements at the end of the array are then moved to the beginning or vice versa, depending on the direction of the shift. By specifying the axis parameter, the shifting can be done along a specific axis of the array. This operation is useful for various data processing and analysis tasks, such as reordering data or creating sliding windows for time series data.

Shift Elements in NumPy Array (With Examples)

You can use one of the following methods to shift the elements in a NumPy array:

Method 1: Shift Elements (Keep All Original Elements)

#shift each element two positions to the right
data_new = np.roll(data, 2)

Method 2: Shift Elements (Allow Elements to Be Replaced)

#define shifting functiondefshift_elements(arr, num, fill_value):
result = np.empty_like(arr)
if num > 0:
result = fill_value

```
result = arr
elif num < 0:
result = fill_value
result = arr
else:
result = arr
return result

#shift each element two positions to the right (replace shifted elements with zero)
data_new = shift_elements(data, 2, 0)
```

The following examples show how to use each method in practice.

Method 1: Shift Elements (Keep All Original Elements)

The following code shows how to use the function to shift each element in a NumPy array two positions to the right:

```
import numpy as np

#create NumPy array
data = np.array()
```

```
#shift each element two positions to the right
data_new = np.roll(data, 2)

#view new NumPy array
data_new

array()
```

Notice that each element was shifted two positions to the right and elements at the end of the array simply got moved to the front.

We could also use a negative number in the np.roll() function to shift elements to the left:

```
import numpy as np

#create NumPy array
data = np.array()

#shift each element three positions to the left
data_new = np.roll(data, -3)

#view new NumPy array
data_new
```

array()

Method 2: Shift Elements (Allow Elements to Be Replaced)

We can also define a custom function to shift the elements in a NumPy array and allow elements that are shifted to be replaced by a certain value.

For example, we can define the following function to shift elements and replace any shifted elements with the value 0:

```
import numpy as np

#create NumPy array
data = np.array()

#define custom function to shift elements
defshift_elements(arr, num, fill_value):
result = np.empty_like(arr)
if num > 0:
result = fill_value
result = arr
elif num < 0:
result = fill_value
result = arr
```

```
else:
result = arr
return result

#shift each element two positions to the right and replace shifted values with zero
data_new = shift_elements(data, 2, 0)

#view new NumPy array
data_new

array()
```

We can also use a negative number in the function to shift the elements to the left:

```
import numpy as np

#create NumPy array
data = np.array()

#define custom function to shift elements
defshift_elements(arr, num, fill_value):
result = np.empty_like(arr)
if num > 0:
result = fill_value
```

```
result = arr
elif num < 0:
result = fill_value
result = arr
else:
result = arr
return result

#shift each element three positions to the left and replace shifted values with 50
data_new = shift_elements(data, -3, 50)

#view new NumPy array
data_new

array()
```

Additional Resources

The following tutorials explain how to perform other common operations in NumPy: