NumPy provides a number of efficient sorting functions that make it very easy to sort an array. The first function for sorting is sort(), which takes in an array and returns a sorted array.

In [1]:

```
import numpy as np
ages = np.array([34,12,37,5,13])
sorted_ages = np.sort(ages)
print("The sorted ages are:", sorted_ages)
print("The ages are:", ages)
```

```
The sorted ages are: [ 5 12 13 34 37]
The ages are: [34 12 37  5 13]
```

As we can see from the output, the sort() function does not modify the original array. Instead it returns a sorted array. If we want to sort the original array, call the sort() function on the array itself.

In [2]:

```
import numpy as np
ages = np.array([34,12,37,5,13])
print("The ages are:", ages)
ages.sort()
print("The sorted ages are:", ages)
```

```
The ages are: [34 12 37  5 13]
The sorted ages are: [ 5 12 13 34 37]
```

There is another function used for sorting; that’s argsort(). The argsort() function returns the indices that will sort an array.

In [3]:

```
import numpy as np
ages = np.array([34,12,37,5,13])
print("The index of the sorted ages are:", ages.argsort())
```

```
The index of the sorted ages are: [3 1 4 0 2]
```

To print the sorted ages array, use the result of argsort() as the index to the **ages** array.

In [4]:

```
import numpy as np
ages = np.array([34,12,37,5,13])
print("The sorted ages are:", ages[ages.argsort()])
```

```
The sorted ages are: [ 5 12 13 34 37]
```

What is the real use of argsort()? Imagine that we have three arrays representing a list of people, along with their ages and heights. Suppose that we want to sort this group of people by age. If we simply sort the ages array by itself, the other two arrays would not be sorted correctly based on age. This is where argsort() comes in really handy. Once the sort indices are obtained, simply feed them into the three arrays

In [5]:

```
import numpy as np
persons = np.array(['Johnny','Mary','Peter','Will','Joe'])
ages = np.array([34,12,37,5,13])
heights = np.array([1.76,1.2,1.68,0.5,1.25])
sort_indices = np.argsort(ages)
print("The sorted names with respect to ages are:", persons[sort_indices])
print("The sorted ages are:", ages[sort_indices])
print("The sorted heights with respect to ages are:", heights[sort_indices])
```

```
The sorted names with respect to ages are: ['Will' 'Mary' 'Joe' 'Johnny' 'Pe
ter']
The sorted ages are: [ 5 12 13 34 37]
The sorted heights with respect to ages are: [0.5  1.2  1.25 1.76 1.68]
```

They would now be sorted based on age. As we can see, Will is the youngest, followed by Mary, and so on. The corresponding height for each person would also be in the correct order. If we wish to sort based on name, then simply use argsort() on the persons array and feed the resulting indices into the three arrays.

In [6]:

```
import numpy as np
persons = np.array(['Johnny','Mary','Peter','Will','Joe'])
ages = np.array([34,12,37,5,13])
heights = np.array([1.76,1.2,1.68,0.5,1.25])
sort_indices = np.argsort(persons)
print("The sorted names are:", persons[sort_indices])
print("The sorted ages with respect to names are:", ages[sort_indices])
print("The sorted heights with respect to names are:", heights[sort_indices])
```

```
The sorted names are: ['Joe' 'Johnny' 'Mary' 'Peter' 'Will']
The sorted ages with respect to names are: [13 34 12 37  5]
The sorted heights with respect to names are: [1.25 1.76 1.2  1.68 0.5 ]
```

To reverse the order of the names and display them in descending order, use the Python[::-1] notation.

In [7]:

```
import numpy as np
persons = np.array(['Johnny','Mary','Peter','Will','Joe'])
ages = np.array([34,12,37,5,13])
heights = np.array([1.76,1.2,1.68,0.5,1.25])
reverse_sort_indices = np.argsort(persons)[::-1]
print("The reverse sorted names are:", persons[reverse_sort_indices])
print("The revere sorted ages with respect to names are:", ages[reverse_sort_indices])
print("The reverse sorted heights with respect to names are:", heights[reverse_sort_indices
```

```
The reverse sorted names are: ['Will' 'Peter' 'Mary' 'Johnny' 'Joe']
The revere sorted ages with respect to names are: [ 5 37 12 34 13]
The reverse sorted heights with respect to names are: [0.5  1.68 1.2  1.76
1.25]
```

In [ ]:

```

```