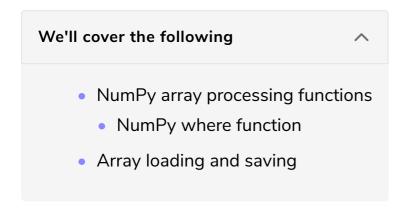
Processing NumPy Arrays

In this lesson, you will explore some more NumPy array processing techniques.



So far, we have discussed the basics of NumPy. Now, these techniques will be applied to data along with some advanced functions to see what more can be achieved.

NumPy array processing functions

Now, let's discuss some NumPy functions that help in array processing.

NumPy where function

Let's look at an example of the where function and try to understand it.

```
import numpy as np
# Declare 2 arrays
arr1 = np.array([10, 20, 30, 40])
arr2 = np.array([500, 600, 700, 800])

cond = np.array([False, True, False, True]) # Create an array with bool operators

res = np.where(cond, arr1, arr2) # apply the where condition
print(res)
```

What exactly happened here? The NumPy where function took 3 values cond, arr1, and arr2. This function first checks the value for cond. If the value of cond is True, then it returns the value of arr1 for the respective index, and if it is False, then it returns the value of arr2 for its respective index. In this case, arr1 is the first parameter after cond, and arr2 is the last parameter of the where

function. These arrays can be changed and replaced with any desired value.

This function prevents us from writing complex nested if statements and provides an easy alternative.

Some more NumPy functions for array processing are given below.

Function	Description
np.random	Generates an array with random numbers
np.mean	Returns mean of all elements in the array
np.var	Returns the variance of all elements in the array
np.std	Returns the standard deviation of all elements in the array
np.sum	Returns the sum of all elements in the array
np.sort	Returns a sorted copy of an array
np.unique	Returns array with all unique elements
np.any	Returns True if any element of boolean array is True
np.all	Returns True if all elements of boolean array are True

The following example makes use of all these functions.

```
import numpy as np
# Declare 2 arrays of different types
arr1 = np.random.rand(7)
arr2 = np.array(['Dog', 'Cat', 'Lion', 'Dog', 'Eagle', 'Turtle', 'Lion'])
print("The original array")
print(arr1)
print("\nMean of all elements of array")
print(arr1.mean())
print("\nVariance of all elements of array")
print(arr1.var())
print("\nStandard Deviation of all elements of array")
print(arr1.std())
print("\nSum of all elements of array")
print(arr1.sum())
print("\nSorted array")
arr1.sort()
print(arr1)
print("\n0riginal array")
print(arr2)
print("\nUnique array")
print(np.unique(arr2))
```

More functions like these can be found here. The link points to the scipy package, which contains universal functions for n-dimensional arrays.

Array loading and saving

Sometimes we might need to save our arrays for later use. In these cases, the NumPy functions of array saving and loading come into play. The functions np.save and np.load are used to save and load arrays.

```
import numpy as np

arr1 = np.random.randn(7)

print("The original array:")
print(arr1)

np.save('saved_arr',arr1) # save the array

arr2 = np.load('saved_arr.npy') # load the saved array from file

print("\nThe array loaded from file:")
```







[]

The array is first saved using the <code>np.save</code> function. The first parameter to this is the name of the file we want to save it in, and the second parameter is the array we want to save.

The array is then loaded back into a new variable arra, using the np.load function. This function takes the name of the file the array is saved in and returns the saved array.

Note: Do add .npy at the end of the file name in the np.load function.

That's it for Python's NumPy package. Next, a challenge awaits to test your newly acquired NumPy skills.