NumPy Array Creation

An array allows us to store a collection of multiple values in a single data structure.

The NumPy array is similar to a list, but with added benefits such as being faster and more memory efficient.

NumPy library provides various methods to work with data. To leverage all those features, we first need to create numpy arrays.

There are multiple techniques to generate arrays in NumPy, and we will explore each of them below.

Create Array Using Python List

We can create a NumPy array using a Python List. For example,

```
import numpy as np


# create a list named list1

list1 = [2, 4, 6, 8]


# create numpy array using list1

array1 = np.array(list1)


print(array1)


# Output: [2 4 6 8]
```

In the above example, we first imported the numpy library as np and created a list named list1. Notice the code

```
array1 = np.array(list1)
```

Here, we have created an array by passing list1 as an argument to the np.array() function.

Instead of creating a list and using the list variable with the np.array() function, we can directly pass list elements as an argument. For example,

```
import numpy as np


# create numpy array using a list
```

```
array1 = np.array([2, 4, 6, 8])
print(array1)


# Output: [2  4  6 8]
```

Run Code

This code gives the same output as the previous code.

Note: Unlike lists, arrays can only store data of a similar type.

Create an Array Using np.zeros()

The np.zeros() function allows us to create an array filled with all zeros. For example,

```
import numpy as np


# create an array with 4 elements filled with zeros
array1 = np.zeros(4)


print(array1)


# Output: [0. 0. 0. 0.]
```

Here, we have created an array named array1 with **4** elements all initialized to **0** using the np.zeros(4) function.

Note: Similarly we can use np.ones() to create an array filled with values **1.**

Create an Array With np.arange()

The np.arange() function returns an array with values within a specified interval. For example,

```
import numpy as np


# create an array with values from 0 to 4
array1 = np.arange(5)
```

```
print("Using np.arange(5):", array1)


# create an array with values from 1 to 8 with a step of 2
array2 = np.arange(1, 9, 2)


print("Using np.arange(1, 9, 2):",array2)
```

Output

```
Using np.arange(5): [0 1 2 3 4]
Using np.arange(1, 9, 2): [1 3 5 7]
```

In the above example, we have created arrays using the np.arange() function.

- np.arange(5) - create an array with 5 elements, where the values range from **0** to **4**
- np.arange(1, 9, 2) - create an array with 5 elements, where the values range from **1** to **8** with a step of **2**.

Create an Array With np.random.rand()

The np.random.rand() function is used to create an array of random numbers.

Let's see an example to create an array of **5** random numbers,

```
import numpy as np


# generate an array of 5 random numbers
array1 = np.random.rand(5)


print(array1)
```

Output

```
[0.08455648 0.56379034 0.66463204 0.97608605 0.30700052]
```

In the above example, we have used the np.random.rand() function to create an array array1 with **5** random numbers.

This code generates a different output each time we run it.

Create an Empty NumPy Array

To create an empty NumPy array, we use the np.empty() function. For example,

```
import numpy as np


# create an empty array of length 4

array1 = np.empty(4)


print(array1)
```

Output

```
[0.0.0.0]
```

Here, we have created an empty array of length **4** using the np.empty() function.

If we look into the output of the code, we can see the empty array is actually not empty, it has some values in it.

It is because although we are creating an empty array, NumPy will try to add some value to it. The values stored in the array are arbitrary and have no significance.