What is the difference between np.linspace and np.arange in NumPy?

Authored by

stats writer

June 27, 2024

Np.linspace and np.arange are both functions in NumPy that are used to generate arrays of evenly spaced numbers. However, the main difference between the two is that np.linspace generates an array with a specified number of elements, while np.arange generates an array with a specified step size.

In other words, np.linspace takes in a start and end point, and a number of elements, and evenly divides the interval between the two points to create an array with the specified number of elements. On the other hand, np.arange takes in a start and end point, and a step size, and creates an array by incrementing the start point by the given step size until the end point is reached.

Additionally, np.linspace can handle non-integer values for the number of elements, while np.arange can only use integer values for the step size. This makes np.linspace more precise for creating evenly spaced arrays with specific number of elements.

In summary, while both functions are used for creating arrays with evenly spaced values, the main difference lies in the way they determine the values in the array - np.linspace uses the number of elements, while np.arange uses the step size.

NumPy: The Difference Between np.linspace and np.arange

When it comes to creating a sequence of values, linspace and arange are two commonly used NumPy functions.

Here is the subtle difference between the two functions:

linspace allows you to specify the *number* of stepsarange allows you to specify the *size* of the steps

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

Example 1: How to Use np.linspace

The np.linspace() function uses the following basic syntax:

np.linspace(start, stop, num, ...)

where:

start: The starting value of the sequencestop: The end value of the sequencenum: the number of values to generate

The following code shows how to use np.linspace() to create 11 values evenly spaced between 0 and 20:

```
import numpy as np

#create sequence of 11 evenly spaced values between 0 and 20
np.linspace(0, 20, 11)

array()
```

The result is an array of 11 values that are evenly spaced between 0 and 20.

Using this method, np.linspace() automatically determines how far apart to space the values.

Example 2: How to Use np.arange

The np.arange() function uses the following basic syntax:

np.arange(start, stop, step, ...)

where:

start: The starting value of the sequencestop: The end value of the sequencestep: The spacing between values

The following code shows how to use np.arange() to create a sequence of values between 0 and 20 where the spacing between each value is 2:

```
import numpy as np

#create sequence of values between 0 and 20 where spacing is 2
np.arange(0, 20, 2)

array()
```

The result is a sequence of values between 0 and 20 where the spacing between each value is 2.

Using this method, np.arange() automatically determines how many values to generate.

If we use a different step size (like 4) then np.arange() will automatically adjust the total number of values generated:

```
import numpy as np

#create sequence of values between 0 and 20 where spacing is 4
np.arange(0, 20, 4)

array()
```

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