NumPy - Array From Numerical Ranges

In this chapter, we will see how to create an array from numerical ranges.

numpy.arange

This function returns an **ndarray** object containing evenly spaced values within a given range. The format of the function is as follows –

```
numpy.arange(start, stop, step, dtype)
```

The constructor takes the following parameters.

Sr.No.	Parameter & Description
1	**start** The start of an interval. If omitted, defaults to 0
2	**stop** The end of an interval (not including this number)
3	**step** Spacing between values, default is 1
4	**dtype** Data type of resulting ndarray. If not given, data type of input is used

The following examples show how you can use this function.

Example 1

Live Demo

```
import numpy as np

x = np.arange(5)

print x
```

Its output would be as follows −

```
[0  1  2  3  4]
```

Example 2

Live Demo

```
import numpy as np

# dtype set

x = np.arange(5, dtype = float)

print x
```

Here, the output would be −

```
[0.  1.  2.  3.  4.]
```

Example 3

Live Demo

```
# start and stop parameters set

import numpy as np

x = np.arange(10,20,2)

print x
```

Its output is as follows −

```
[10  12  14  16  18]
```

numpy.linspace

This function is similar to **arange()** function. In this function, instead of step size, the number of evenly spaced values between the interval is specified. The usage of this function is as follows −

```
numpy.linspace(start, stop, num, endpoint, retstep, dtype)
```

The constructor takes the following parameters.

Sr.No.	Parameter & Description

1	**start** The starting value of the sequence
2	**stop** The end value of the sequence, included in the sequence if endpoint set to true
3	**num** The number of evenly spaced samples to be generated. Default is 50
4	**endpoint** True by default, hence the stop value is included in the sequence. If false, it is not included
5	**retstep** If true, returns samples and step between the consecutive numbers
6	**dtype** Data type of output **ndarray**

The following examples demonstrate the use **linspace** function.

Example 1

Live Demo

```
import numpy as np

x = np.linspace(10,20,5)

print x
```

Its output would be −

```
[10.   12.5   15.   17.5  20.]
```

Example 2

Live Demo

```
# endpoint set to false

import numpy as np

x = np.linspace(10,20, 5, endpoint = False)

print x
```

The output would be −

```
[10.   12.   14.   16.   18.]
```

Example 3

Live Demo

```
# find retstep value

import numpy as np


x = np.linspace(1,2,5, retstep = True)

print x

# retstep here is 0.25
```

Now, the output would be −

```
(array([ 1.  ,  1.25,  1.5 ,  1.75,  2.  ]), 0.25)
```

numpy.logspace

This function returns an **ndarray** object that contains the numbers that are evenly spaced on a log scale. Start and stop endpoints of the scale are indices of the base, usually 10.

```
numpy.logspace(start, stop, num, endpoint, base, dtype)
```

Following parameters determine the output of **logspace** function.

Sr.No.	Parameter & Description

1	**start** The starting point of the sequence is $base^{start}$
2	**stop** The final value of sequence is $base^{stop}$
3	**num** The number of values between the range. Default is 50
4	**endpoint** If true, stop is the last value in the range
5	**base** Base of log space, default is 10
6	**dtype** Data type of output array. If not given, it depends upon other input arguments

The following examples will help you understand the **logspace** function.

Example 1

Live Demo

```
import numpy as np

# default base is 10

a = np.logspace(1.0, 2.0, num = 10)

print a
```

Its output would be as follows –

```
[ 10.          12.91549665     16.68100537      21.5443469  27.82559402
  35.93813664   46.41588834     59.94842503      77.42636827    100.    ]
```

Example 2

Live Demo

```
# set base of log space to 2

import numpy as np

a = np.logspace(1,10,num = 10, base = 2)

print a
```

Now, the output would be −

```
[ 2.     4.     8.    16.    32.    64.   128.   256.    512.   1024.]
```