



NumPy Array Iterating

[< Previous](#)[Next >](#)

Iterating Arrays

Iterating means going through elements one by one.

As we deal with multi-dimensional arrays in numpy, we can do this using basic **for** loop of python.

If we iterate on a 1-D array it will go through each element one by one.

Example

Iterate on the elements of the following 1-D array:

```
import numpy as np

arr = np.array([1, 2, 3])

for x in arr:
    print(x)
```

[Try it Yourself »](#)

Iterating 2-D Arrays

In a 2-D array it will go through all the rows.

Example

Iterate on the elements of the following 2-D array:

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

for x in arr:
    print(x)
```

[Try it Yourself »](#)

If we iterate on a n -D array it will go through $n-1$ th dimension one by one.

To return the actual values, the scalars, we have to iterate the arrays in each dimension.

Example

Iterate on each scalar element of the 2-D array:

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

for x in arr:
    for y in x:
        print(y)
```

[Try it Yourself »](#)

Iterating 3-D Arrays

In a 3-D array it will go through all the 2-D arrays.

Example

Iterate on the elements of the following 3-D array:

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

for x in arr:
    print(x)
```

[Try it Yourself »](#)

To return the actual values, the scalars, we have to iterate the arrays in each dimension.

Example

Iterate down to the scalars:

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

for x in arr:
    for y in x:
        for z in y:
            print(z)
```

[Try it Yourself »](#)

Iterating Arrays Using `nditer()`

The function `nditer()` is a helping function that can be used from very basic to very advanced iterations. It solves some basic issues which we face in iteration, let's go through it with examples.

Iterating on Each Scalar Element

In basic `for` loops, iterating through each scalar of an array we need to use `n for` loops which can be difficult to write for arrays with very high dimensionality.

Example

Iterate through the following 3-D array:

```
import numpy as np

arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

for x in np.nditer(arr):
    print(x)
```

[Try it Yourself »](#)

Iterating Array With Different Data Types

We can use `op_dtypes` argument and pass it the expected datatype to change the datatype of elements while iterating.

NumPy does not change the data type of the element in-place (where the element is in array) so it needs some other space to perform this action, that extra space is called buffer, and in order to enable it in `nditer()` we pass `flags=['buffered']`.

Example

Iterate through the array as a string:

```
import numpy as np

arr = np.array([1, 2, 3])
```

```
for x in np.nditer(arr, flags=['buffered'], op_dtypes=['S']):  
    print(x)
```

[Try it Yourself »](#)

Iterating With Different Step Size

We can use filtering and followed by iteration.

Example

Iterate through every scalar element of the 2D array skipping 1 element:

```
import numpy as np  
  
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])  
  
for x in np.nditer(arr[:, ::2]):  
    print(x)
```

[Try it Yourself »](#)

Enumerated Iteration Using ndenumerate()

Enumeration means mentioning sequence number of somethings one by one.

Sometimes we require corresponding index of the element while iterating, the `ndenumerate()` method can be used for those usecases.

Example

Enumerate on following 1D arrays elements:

```
import numpy as np  
  
arr = np.array([1, 2, 3])
```

```
for idx, x in np.ndenumerate(arr):  
    print(idx, x)
```

[Try it Yourself »](#)

Example

Enumerate on following 2D array's elements:

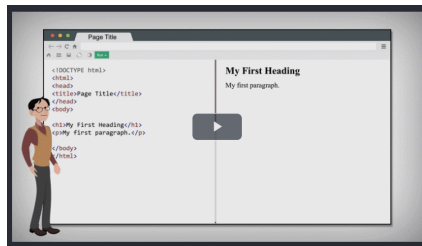
```
import numpy as np  
  
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])  
  
for idx, x in np.ndenumerate(arr):  
    print(idx, x)
```

[Try it Yourself »](#)

[< Previous](#)[Next >](#)

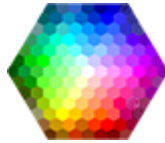
NEW

We just launched
W3Schools videos



Explore now

COLOR PICKER



Get certified
by completing
a Python
course today!



Get started

CODE GAME



[Play Game](#)



[Report Error](#)

[Spaces](#)

[Pro](#)

[Get Certified](#)

Top Tutorials

- [HTML Tutorial](#)
- [CSS Tutorial](#)
- [JavaScript Tutorial](#)
- [How To Tutorial](#)
- [SQL Tutorial](#)
- [Python Tutorial](#)
- [W3.CSS Tutorial](#)
- [Bootstrap Tutorial](#)
- [PHP Tutorial](#)
- [Java Tutorial](#)
- [C++ Tutorial](#)
- [jQuery Tutorial](#)

Top References

- [HTML Reference](#)
- [CSS Reference](#)
- [JavaScript Reference](#)

[SQL Reference](#)
[Python Reference](#)
[W3.CSS Reference](#)
[Bootstrap Reference](#)
[PHP Reference](#)
[HTML Colors](#)
[Java Reference](#)
[Angular Reference](#)
[jQuery Reference](#)

Top Examples

[HTML Examples](#)
[CSS Examples](#)
[JavaScript Examples](#)
[How To Examples](#)
[SQL Examples](#)
[Python Examples](#)
[W3.CSS Examples](#)
[Bootstrap Examples](#)
[PHP Examples](#)
[Java Examples](#)
[XML Examples](#)
[jQuery Examples](#)

Get Certified

[HTML Certificate](#)
[CSS Certificate](#)
[JavaScript Certificate](#)
[Front End Certificate](#)
[SQL Certificate](#)
[Python Certificate](#)
[PHP Certificate](#)
[jQuery Certificate](#)
[Java Certificate](#)
[C++ Certificate](#)
[C# Certificate](#)
[XML Certificate](#)

[FORUM](#) | [ABOUT](#)

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using W3Schools, you agree to have read and accepted our [terms of use](#), [cookie](#) and [privacy policy](#).

Copyright 1999-2022 by Refsnes Data. All Rights Reserved.
W3Schools is Powered by W3.CSS.

