



Python Iterators

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

Python Iterators

An iterator is an object that contains a countable number of values.

An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.

Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods `__iter__()` and `__next__()`.

Iterator vs Iterable

Lists, tuples, dictionaries, and sets are all iterable objects. They are iterable *containers* which you can get an iterator from.

All these objects have a `iter()` method which is used to get an iterator:

Example

Return an iterator from a tuple, and print each value:

```
mytuple = ("apple", "banana", "cherry")
myit = iter(mytuple)

print(next(myit))
print(next(myit))
print(next(myit))
```

[Run example »](#)

Even strings are iterable objects, and can return an iterator:

Example

Strings are also iterable objects, containing a sequence of characters:

```
mystr = "banana"
myit = iter(mystr)

print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
```

[Run example »](#)

Looping Through an Iterator

We can also use a `for` loop to iterate through an iterable object:

Example

Iterate the values of a tuple:

```
mytuple = ("apple", "banana", "cherry")

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

[Run example »](#)

Example

Iterate the characters of a string:

```
mystr = "banana"

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

Run example »

The `for` loop actually creates an iterator object and executes the `next()` method for each loop.

Create an Iterator

To create an object/class as an iterator you have to implement the methods `__iter__()` and `__next__()` to your object.

As you have learned in the [Python Classes/Objects](#) chapter, all classes have a function called `__init__()`, which allows you do some initializing when the object is being created.

The `__iter__()` method acts similar, you can do operations (initializing etc.), but must always return the iterator object itself.

The `__next__()` method also allows you to do operations, and must return the next item in the sequence.

Example

Create an iterator that returns numbers, starting with 1, and each sequence will increase by one (returning 1,2,3,4,5 etc.):

```
class MyNumbers:
    def __iter__(self):
        self.a = 1
        return self

    def __next__(self):
        x = self.a
        self.a += 1
```

```
        return x

myclass = MyNumbers()
myiter = iter(myclass)

print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
```

[Run example »](#)

StopIteration

The example above would continue forever if you had enough `next()` statements, or if it was used in a `for` loop.

To prevent the iteration to go on forever, we can use the `StopIteration` statement.

In the `__next__()` method, we can add a terminating condition to raise an error if the iteration is done a specified number of times:

Example

Stop after 20 iterations:

```
class MyNumbers:
    def __iter__(self):
        self.a = 1
        return self

    def __next__(self):
        if self.a <= 20:
            x = self.a
            self.a += 1
            return x
        else:
            raise StopIteration

myclass = MyNumbers()
```

```
myiter = iter(myclass)
```

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

[Run example »](#)[< Previous](#)[Next >](#)

COLOR PICKER



HOW TO

- Tabs
- Dropdowns
- Accordions
- Side Navigation
- Top Navigation
- Modal Boxes
- Progress Bars
- Parallax
- Login Form
- HTML Includes
- Google Maps
- Range Sliders
- Tooltips
- Slideshow
- Filter List
- Sort List

SHARE



CERTIFICATES

HTML
CSS
JavaScript
PHP
jQuery
Bootstrap
XML

[Read More »](#)

[REPORT ERROR](#)

[PRINT PAGE](#)

[FORUM](#)

[ABOUT](#)

Top 10 Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
SQL Tutorial
PHP Tutorial
jQuery Tutorial
Python Tutorial

Top 10 References

HTML Reference
CSS Reference
JavaScript Reference
W3.CSS Reference
Bootstrap Reference

[SQL Reference](#)
[PHP Reference](#)
[HTML Colors](#)
[jQuery Reference](#)
[Python Reference](#)

Top 10 Examples

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

Web Certificates

[HTML Certificate](#)
[CSS Certificate](#)
[JavaScript Certificate](#)
[jQuery Certificate](#)
[PHP Certificate](#)
[Bootstrap Certificate](#)
[XML Certificate](#)

W3Schools is optimized for learning, testing, and training. Examples might be simplified to improve reading and basic understanding. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using this site, you agree to have read and accepted our terms of use, cookie and privacy policy. Copyright 1999-2019 by Refsnes Data. All Rights Reserved.

Powered by W3.CSS.

