**Decorator: (@)**

**Iterator and Generator:( yield):**

|  |  |
| --- | --- |
|  | **Iteration** is a general term for taking each item of something, one after another. Any time you use a loop, explicit or implicit, to go over a group of items, that is iteration.  In Python, **iterable** and **iterator** have specific meanings.  An **iterable** is an object that has an \_\_iter\_\_ method which returns an **iterator**, or which defines a \_\_getitem\_\_ method that can take sequential indexes starting from zero (and raises an IndexErrorwhen the indexes are no longer valid). So an **iterable** is an object that you can get an **iterator** from.  An **iterator** is an object with a next (Python 2) or \_\_next\_\_ (Python 3) method.  Whenever you use a for loop, or map, or a list comprehension, etc. in Python, the next method is called automatically to get each item from the **iterator**, thus going through the process of **iteration**. |

iterator is an object which implements the iterator protocol. The iterator protocol consists of two methods. The \_\_iter\_\_() method, which must return the iterator object and the next() method, which returns the next element from a sequence.

Python has several built-in objects, which implement the iterator protocol.

**The Iteration Protocol**

The built-in function iter takes an iterable object and returns an iterator.

>>> x **=** iter([1, 2, 3])

>>> x

<listiterator object at 0x1004ca850>

>>> x**.**next()

1

>>> x**.**next()

2

>>> x**.**next()

3

>>> x**.**next()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

StopIteration

Each time we call the next method on the iterator gives us the next element. If there are no more elements, it raises a *StopIteration*.

Iterators are implemented as classes. Here is an iterator that works like built-in xrange function.

**class** **yrange**:

**def** **\_\_init\_\_**(self, n):

self**.**i **=** 0

self**.**n **=** n

**def** **\_\_iter\_\_**(self):

**return** self

**def** **next**(self):

**if** self**.**i **<** self**.**n:

i **=** self**.**i

self**.**i **+=** 1

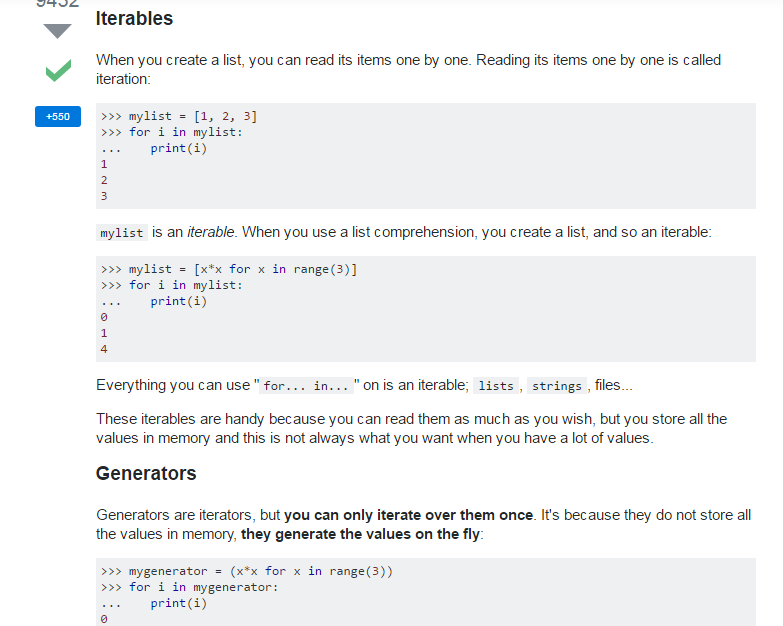
**return** i

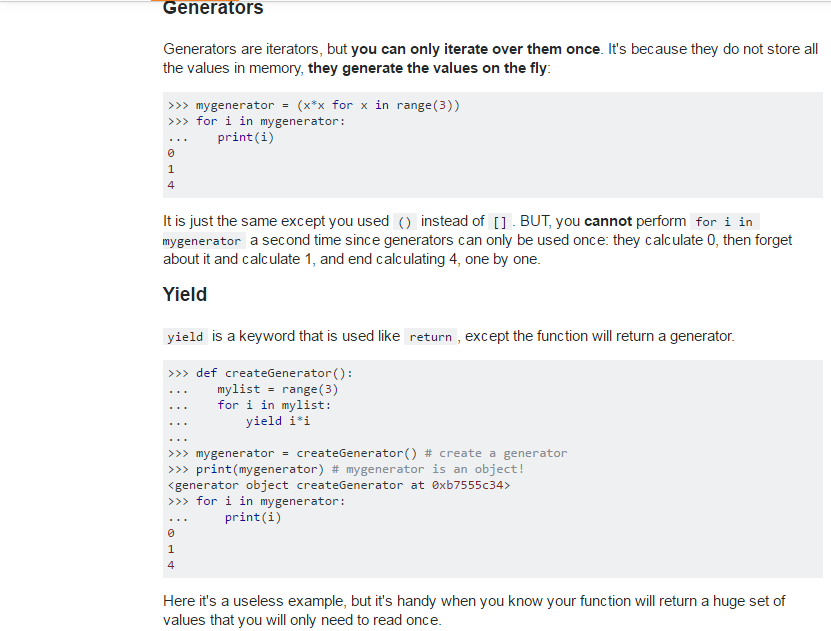
**else**:

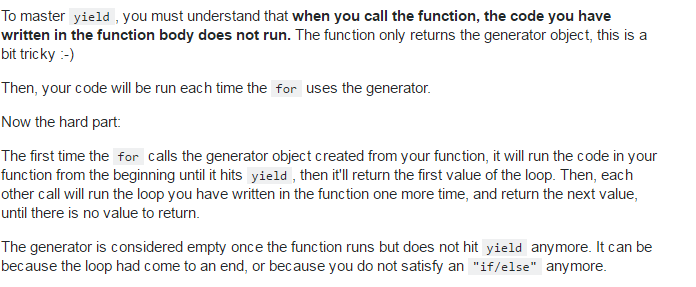
**raise** **StopIteration**()

The \_\_iter\_\_ method is what makes an object iterable. Behind the scenes, the *iter* function calls \_\_iter\_\_ method on the given object.

The return value of \_\_iter\_\_ is an iterator. It should have a next method and raise StopIterationwhen there are no more elements.







References:

http://stackoverflow.com/questions/231767/what-does-the-yield-keyword-do-in-python

**Regular expression:**

A **regular expression** is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern.**Regular expressions** are widely used in UNIX world. The module re provides full support for Perl-like **regular expressions in Python**