There is a lot of overhead in building an iterator in Python; we have to implement a class with __iter__() and __next__() method, keep track of internal states, raise StopIterationwhen there was no values to be returned etc.

This is both lengthy and counter intuitive. Generator comes into rescue in such situations.

Python generators are a simple way of creating iterators. All the overhead we mentioned above are automatically handled by generators in Python.

How to create a generator in Python?

It is fairly simple to create a generator in Python. It is as easy as defining a normal function with yield statement instead of a return statement.

If a function contains at least one yield statement (it may contain other yield or returnstatements), it becomes a generator function. Both yield and return will return some value from a function.

The difference is that, while a return statement terminates a function entirely, yieldstatement pauses the function saving all its states and later continues from there on successive calls.

Example:

```
def my_gen():
    n = 1
    print('This is printed first')
    # Generator function contains yield statements
    yield n

    n += 1
    print('This is printed second')
    yield n

    n += 1
    print('This is printed at last')
    yield n
```

Python generators using Loop:

```
def rev_str(my_str):
    length = len(my_str)
    for i in range(length - 1,-1,-1):
        yield my_str[i]

for char in rev_str("hello"):
     print(char)
```

Benefits of Generators:

Memory Efficient

A normal function to return a sequence will create the entire sequence in memory before returning the result. This is an overkill if the number of items in the sequence is very large.

Generator implementation of such sequence is memory friendly and is preferred since it only produces one item at a time.

Represent Infinite Stream

Generators are excellent medium to represent an infinite stream of data. Infinite streams cannot be stored in memory and since generators produce only one item at a time, it can represent infinite stream of data.

Easy to Implement