Full stack web development using python

Iterator and generator

Saurabh Shukla (MySirG)

Agenda

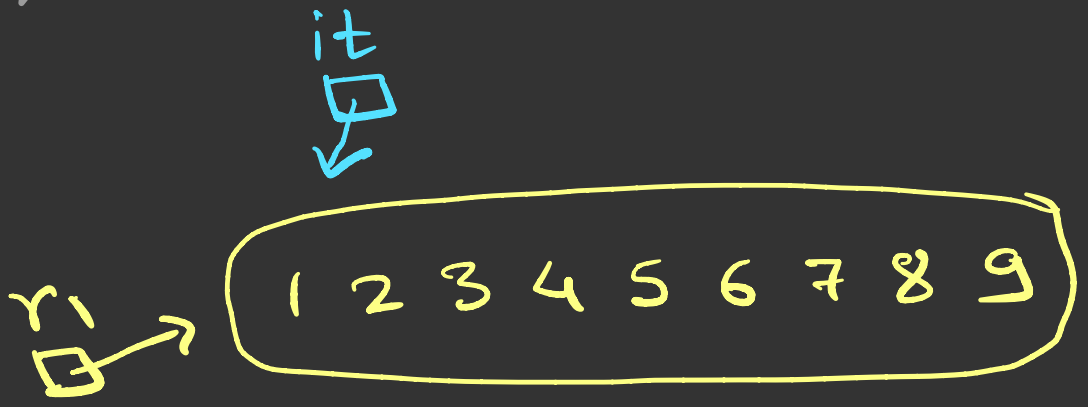
1. Iterator
2. Generator
3. yield
4. StopIteration

Iterator

- An iterator can be seen as a pointer to a container.
- The iterator is an abstraction, which enables the programmer to access all the elements of a container (a set, a list, so on) without any deeper knowledge of the data structure of this container object.
- In Python iterator is implicitly available in for loop.

iter() and next()

```
r1 = range(1, 10, 1)
it = iter(r1)
```




```
print( next(it))
```

Generator

- A generator is called like a function
- It is defined like a normal function, but it generates value through yield keyword.
- In simple word, if the body of def contains yield, the function is generator function.

Defining Generator

```
def f1():
    yield 10
    yield 20
    yield 30

g = f1()
a = next(g)
b = next(g)
c = next(g)
```

← function returns generator object

next method is used to get generated values.

Accessing Generator Object via for loop

```
g = f1()
for e in g:
    print(e)
```

Generator object is always iterable.

Define a generator to yield first n natural numbers.

```
def f1(n):
    i=1
    while i<=n:
        yield i
        i+=1
```

return vs yield

return and yield both can return value, the difference is 'return' ends the execution of function and yield pauses the execution

StopIteration

If we call an iterator after all the elements have been iterated, then StopIteration is raised