Full stack web development using python

Iterator and generator

Saurabh Shukla (MySirG)

Agenda

1. Iterator
2. iter and next
3. Generator
4. Endless iterator

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, without any deeper knowledge of the data structure of this container object.
- iterator is implicitly available in for loop.

iter and next

```
it = iter(iterable)
```

↑ iterator object

```
e = next(it)
```

↑ obtaining next element

- next method returns the element pointed by specified iterator object
- next method advances iterator object so that it can point to the next element of the container object.

StopIteration

when next method is called for an iterator object which surpasses the last element of the container object, it produces StopIteration exception

iterator object can only access elements in a sequence from first element to the last element

next method is used to advances the iterator object

there is no previous method to traverse iterator object on the element in reverse direction

Similarly, you cannot randomly access the elements using iterator.

```
l1 = [10, 50, 12, 18, 24]
it = iter(l1)
while True:
    print(next(it))
```

Iterator object can be obtained in two ways -

1. using iter() method
2. generator - function

Generator

- Generators are special kind of functions
- A generator-function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return.
- if the body of a def contains yield, the function automatically becomes a generator function.

Example

```
def f1():
    yield 5
    yield 10
    yield 15
```

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

or

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

Example

Create a generator to produce first n even natural numbers

```
def evenProducer(n):
    a = 2
    while n:
        yield a
        a += 2
        n -= 1
```

Endless iterator

We can create a generator object, which is capable of producing endless iterator.

```
def fib():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a+b
```

Python allows the use of return in generator

The return statement in generator is equivalent to raise StopIteration