Generators

This lesson introduces generators and how to use them in Python.

WE'LL COVER THE FOLLOWING

- Generators

Generators

If you read the previous chapter, you know that iterators are objects that are regularly used with `for` loops. In other words, iterators are objects that implement the iteration protocol. A Python generator is a convenient way to implement an iterator. Instead of a class, a generator is a function which returns a value each time the `yield` keyword is used. Here's an example of a generator to count the values between two numbers:

```
def myrange(a, b):
  while a < b:
    yield a
    a += 1
a = myrange(2, 4) # call the generator function which returns an object
print (next(a)) # iterate through items using next
print (next(a))
```

Like iterators, generators can be used with the `for` loop:

```
def myrange(a, b):
    while a < b:
      yield a
      a += 1
for value in myrange(1, 4):
  print(value)
```

Under the hood, generators behave similarly to iterators. As can be seen in the example below, uncommenting the **line 9** should give an error:

> **Why error?**
>
> Since on **line 6**, the range is defined from 1 to 3, so generating the next number will give an error.

```
def myrange(a, b):
    while a < b:
      yield a
      a += 1

seq = myrange(1,3)
print(next(seq))
print(next(seq))
##next(seq)
```

The interesting thing about generators is the `yield` keyword. The `yield` keyword works much like the `return` keyword, but—unlike `return`—it allows the function to eventually resume its execution. In other words, each time the next value of a generator is needed, Python wakes up the function and resumes its execution from the `yield` line as if the function had never exited.

Generator functions can use other functions inside. For instance, it is very common to use the `range` function to iterate over a sequence of numbers:

```
def squares(n):
  for value in range(n):
    yield value * value

sqr = squares(8)
print(next(sqr))
print(next(sqr))
print(next(sqr))
```

Now, let's solve some exercises to practice your concept of generators.