Yield in Python Tutorial: Generator & Yield vs Return Example

While working with a large amount of data the programmer sometimes needs to control the function’s start and stop state. The Yield gives the programmer the power to execute the program the way he/she wants rather than computing them at once. Let’s take a look at an example. Suppose you have a function that prints “Hello World” 5 times and functions that Yields “Hello World” five times. Let’s take a look at the difference between Yield and simple return.

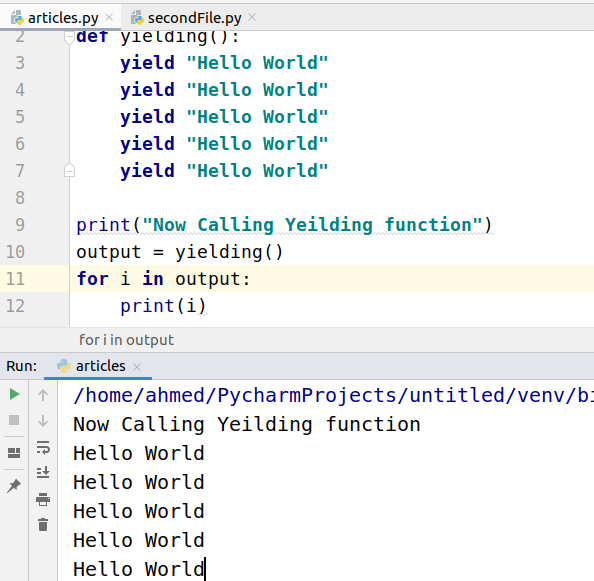
|  |
| --- |
| def normal():  print("Hello World")  print("Hello World")  print("Hello World")  print("Hello World")  print("Hello World") def yielding():  yield "Hello World"  yield "Hello World"  yield "Hello World"  yield "Hello World"  yield "Hello World"  normal() print("Now Calling Yeilding function") print(yielding()) |



Take a look at the above output. The normal function gives us back 5 print statements but the yielding function gives a generator.

Now we need to loop over the generator to get the output.

|  |
| --- |
| def yielding():  yield "Hello World"  yield "Hello World"  yield "Hello World"  yield "Hello World"  yield "Hello World"  print("Now Calling Yeilding function") output = yielding() for i in output:  print(i) |



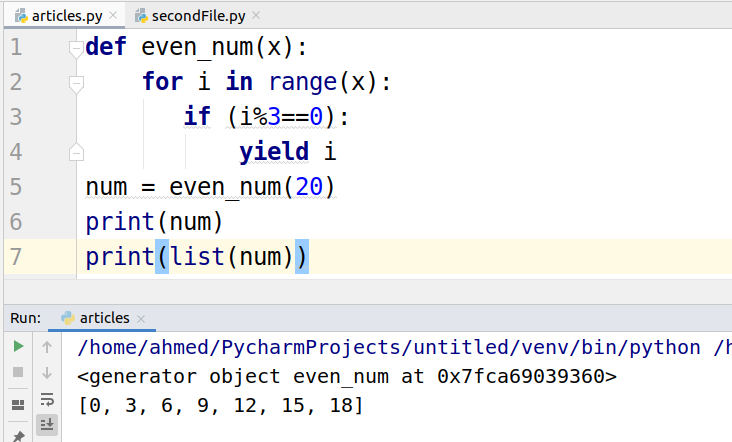
## How to read the values from the generator?

Let’s take a look at different ways to read values from generators.

### Using : list()

We can use the list function to generate all the output and store it in a list.

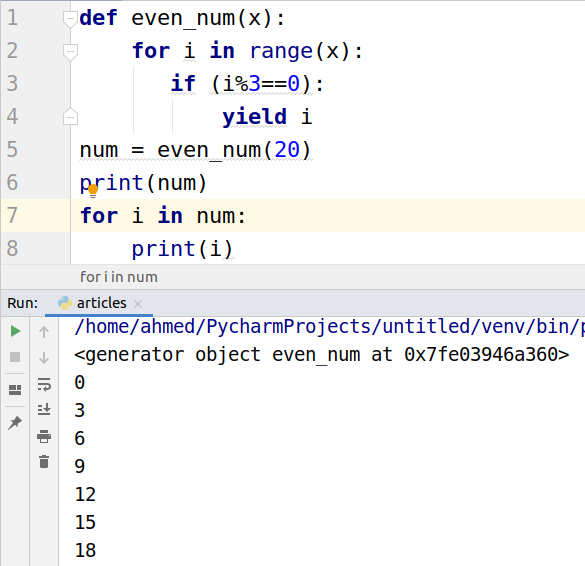
|  |
| --- |
| def even\_num(x):  for i in range(x):  if (i%3==0):  yield i num = even\_num(20) print(num) print(list(num)) |



### Using : for-loop

We used for loop to iterate over the generator in the first example. Let’s take another example.

|  |
| --- |
| def even\_num(x):  for i in range(x):  if (i%3==0):  yield i num = even\_num(20) print(num) for i in num:  print(i) |



### **Using next()**

There is a built-in function next which is used to generate the next number from generator. After generating the last item it will give an error.

|  |
| --- |
| def even\_num(x):  for i in range(x):  if (i%3==0):  yield i num = even\_num(10) print(num) print(next(num)) print(next(num)) print(next(num)) print(next(num)) print(next(num)) |

