```
def simple_gen_fun():
    for i in range(3):
        yield i

iter_obj = simple_gen_fun()
```

Works Like a Class

Create 1 object of this type

```
print(type(iter_obj))
```

```
<class 'generator'>
```

Whenever I call next() method again it is going to continue from the previous state of the object

next (iter_obj)

simple_gen_fun() ——— 0

for i in range(3):

yield i (it returns the value 0, it stores the state of function where next is called)

```
def simple_gen_fun():
    yield 'I AM ON 1ST LINE'
    yield 'I AM ON SECOND LINE'
    yield 'I AM ON THIRD LINE'
    yield 'I AM ON LAST LINE'

iter_obj = simple_gen_fun()
iter_obj2 = simple_gen_fun()
```

First obj

Another obj

They have their own separate existence like object in OOP.

next(iter_obj) 'I AM on 1st LINE', (2)

next(iter_obj) 'I AM ON SECOND LINE', (3)

next(iter_obj) 'I AM ON THIRD LINE' (4)