# Module Python -Fundamentals of Python Language

Vrinda Mavadhiya 10th June Python

**Understanding how generators work in Python.**

** Difference between yield and return.**

** Understanding iterators and creating custom iterators**

What is a Generator?

A generator is a special type of function that produces values one at a time, using the yield keyword, instead of returning them all at once with return.

It remembers its state between each call, making it memory efficient.

✅ Example of a Generator:

def count\_up\_to(n):

count = 1

while count <= n:

yield count

count += 1

Using the generator:

for num in count\_up\_to(5):

print(num)

Output:

1

2

3

4

5

🔹 Generator vs Normal Function

✅ Normal Function with return:

def get\_numbers():

return [1, 2, 3]

* Returns all values at once.
* Consumes more memory if the list is large.

✅ Generator Function with yield:

def get\_numbers():

yield 1

yield 2

yield 3

* Yields one value at a time.
* Saves memory by producing values lazily (on demand).

🔹 Key Difference: yield vs return

| Feature | return | yield |
| --- | --- | --- |
| Behavior | Ends the function and returns a value | Pauses the function and saves its state |
| Memory usage | Loads everything into memory | Lazy (on-demand) loading |
| Use case | Simple results | Large data or continuous streams |

🔹 What is an Iterator?

An iterator is an object that can be iterated (looped) over using next().

To be an iterator, an object must have:

* \_\_iter\_\_() method → returns the iterator object itself
* \_\_next\_\_() method → returns the next value or raises StopIteration

✅ Built-in iterators:

my\_list = [10, 20, 30]

it = iter(my\_list)

print(next(it)) # 10

print(next(it)) # 20

print(next(it)) # 30

🔹 Creating a Custom Iterator

✅ Example: Custom Counter Iterator

class Counter:

def \_\_init\_\_(self, start, end):

self.current = start

self.end = end

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

if self.current <= self.end:

val = self.current

self.current += 1

return val

else:

raise StopIteration

Using the iterator:

for num in Counter(1, 5):

print(num)

Output:

CopyEdit

1

2

3

4

5

🔹 Summary:

| Concept | Description |
| --- | --- |
| Generator | Function that uses yield, returns values one-by-one |
| yield | Pauses function & returns value without ending it |
| return | Ends function and returns the final result |
| Iterator | Object with \_\_iter\_\_() and \_\_next\_\_() methods |
| Custom Iterator | You can create your own iterator class |