Iterators in Python:-

- **An iterator in Python is an object that is used to iterate over iterable objects like lists, tuples, dicts, and sets. The Python iterators object is initialized using the iter() method. It uses the next() method for iteration.**

1. **__iter__(): The iter() method is called for the initialization of an iterator. This returns an iterator object**
2. **__next__(): The next method returns the next value for the iterable. When we use a for loop to traverse any iterable object, internally it uses the iter() method to get an iterator object, which further uses the next() method to iterate over. This method raises a StopIteration to signal the end of the iteration.**

Example of Iterators :-

```
# A simple Python program to demonstrate

# working of iterators using an example type

# that iterates from 10 to given value

# An iterable user defined type

class Test:

    # Constructor

    def __init__(self, limit):

        self.limit = limit

    # Creates iterator object

    # Called when iteration is initialized

    def __iter__(self):

        self.x = 10

        return self

    # To move to next element. In Python 3,

    # we should replace next with __next__
```

```
    def __next__(self):
        # Store current value ofx
        x = self.x
        # Stop iteration if limit is reached
        if x > self.limit:
            raise StopIteration
        # Else increment and return old value
        self.x = x + 1;
        return x
# Prints numbers from 10 to 15
for i in Test(15):
    print(i)
for i in Test(5):
    print(i)
```

Output:-

```
10
11
12
13
14
15
```

Generators in Python :-

- A Generator in Python is a function that returns an iterator using the Yield keyword. In this article, we will discuss how the generator function works in

Python.

Example for Generators :-

```
# A generator function that yields 1 for first time,
# 2 second time and 3 third time
def simpleGeneratorFun():
    yield 1
    yield 2
    yield 3

# Driver code to check above generator function
for value in simpleGeneratorFun():
    print(value)
```

Output :-

```
1
2
3
```