TOPSTechnologies

# Generators And Iterators

**Presented for :**

TOPs Technologies

**Presented by :**

Sumit B Yadav

Que 1

Generators are defined like regular functions but use the yield keyword to return values one at a time.

Each time yield is called, the state of the generator function is "saved" and can be resumed later.

Creating a Generator:
Generators are created by defining a generator function.
Calling the generator function returns a generator object but does not start execution immediately.

Iterating Over a Generator:
You can iterate over a generator using a for loop or functions like next().
Each call to next() resumes the generator from where it left off and runs until the next yield is encountered.

Stopping a Generator:
When the generator function exits, a Stop Iteration exception is raised automatically, signaling that there are no more items to produce.

*Que.2*

*Return Statement:*

*The return statement is used to exit a function and return a value to the caller.*

*When return is executed, the function terminates, and the specified value is returned to the caller.*
*After return is executed, the function's state is not preserved; the function ends.*

*Used when a function needs to output a single value or a result and end its execution.*

*Yield Statement:*
*The yield statement is used to produce a value from a generator function and pause its execution, saving its state for later resumption.*

*When yield is executed, the function's state (including local variables and the current execution point) is saved, and the yielded value is returned to the caller.*
*The function can be resumed later, starting from the point where it was paused, with all its state intact.*

Que.3

*an iterator is an object that allows you to traverse through all the elements of a collection (such as lists, tuples, and dictionaries). Iterators are implemented using two methods:*

- *__iter__(): This method returns the iterator object itself and is called once when the iteration is initialized.*
- *__next__(): This method returns the next value from the collection. When there are no more items to return, it raises the StopIteration exception to signal the end of the iteration.*

*Creating Custom Iterators--*

*To create a custom iterator, you need to define a class that implements both the __iter__() and __next__() methods.*

- *Define the Class:*
  *Define a class for your iterator.*
  *Implement the __init__() method to initialize the state.*
- *Implement __iter__() Method:*
  *This method should return the iterator object itself, usually self.*
- *Implement __next__() Method:*
  *This method should return the next item in the sequence.*
  *Raise StopIteration when there are no more items to return.*