Python Programming

PYTHON GENERATORS

PYTHON GENERATORS

- A Generator in Python is a function that returns an iterator using the Yield keyword.
- A generator function in Python is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. If the body of a def contains yield, the function automatically becomes a Python generator function.
- Python generators are a type of iterable, like lists or dictionaries, but unlike lists, they don't allow indexing with arbitrary indices.
- They generate values one at a time and only when requested.
- This provides memory efficiency, as they don't store the entire sequence in memory at once.

PYTHON GENERATORS

Create a Generator in Python

In Python, we can create a generator function by simply using the def keyword and the yield keyword. The generator has the following syntax in Python:

```
def function_name:
        yield statement
```

When called, a generator function doesn't run the code immediately; instead, it returns a generator object that can be iterated over.

PYTHON GENERATORS

Here's an example of a simple generator function that generates squares of numbers:

```
def square_generator(n):
    for i in range(n):
        yield i ** 2
```

You can use this generator like so:

```
gen = square_generator(5)

for value in gen:
    print(value)
```

PYTHON DECORATORS

This will give the output as:

```
0
1
4
9
16
```

PYTHON GENERATORS

The key difference between a function that uses ‘yield’ and a regular function is that the state of the function is saved. When you call a generator function, it doesn't start from the beginning; it resumes from where it last yielded a value.

This makes generators particularly useful for working with very large or even infinite sequences of data. Since they don't store the entire sequence in memory, they can be more memory efficient than creating and storing a list or other collection.

Some built-in functions in Python, like ‘range()’, ‘map()’, ‘filter()’ also return iterators (which are a specific type of generator) rather than lists.

In Python 3.3 and later versions, you can also use a syntax called a "generator expression" which is similar to a list comprehension but produces values one at a time, thus behaving like a generator.

Thank You