1.What are Generators?

⁕ Are functions that return traversable objects

⁕ Produce items one at a time and only when required

⁕ Are run along with 'for' loops

2.Advantages of using Generators

⁕ Easy to Implement.

⁕ Better memory management and utilization.

⁕ Can be used to produce infinite items.

⁕ Can also be used to pipeline a number of operations.

3.Normal functions vs Generators

|  |  |
| --- | --- |
| **Generator functions** | **Normal Functions** |
| Make use of ‘yield’ keyword | Make use of ‘return’ keyword |
| Run when ‘next()’ method is called | Run when name of the method is called |
| Produce items one at a time and only when required | Produce all the items at once |

4.Writing Generators in Python

⁕ Generators created using the ‘def’ keyword

⁕ Make use of the yield keyword instead of return

5.Generators with Loops

⁕ To execute the generator function at once. You can ‘for’ loop. This loop iterates over all the objects and after all implementations, it executes StopIteration.

6.Generator Expressions

⁕ Resemble list comprehensions and like lambda functions, generator expressions create anonymous generator functions.

7.Use Cases

⁕ Fibonacci Series: A series of numbers where in each number also called as the Fibonacci number is the sum of the two preceding numbers.

⁕ Number Stream: Generating a stream of numbers.

⁕ Sinewave: Generating sine waves using Seaborn.