Q-4: How is memory managed in Python?

here’s a simpler explanation of how memory is managed in Python:

1. Reference Counting

Every time you create an object in Python, it keeps track of how many places are using that object. This is called reference counting.

If you create a list and assign it to a variable, its reference count goes up. If you then assign this list to another variable, the count goes up again.

When you delete a variable or it goes out of scope, the reference count goes down. Once the reference count hits zero, Python knows it can safely delete that object from memory.

2. Garbage Collection

Sometimes, objects reference each other, creating a cycle that reference counting alone can’t handle. For example, two objects might each have a reference to the other.

Python has a garbage collector that looks for these cycles and cleans them up. It runs in the background and makes sure that even these tricky objects get deleted when they’re no longer needed.

3. Memory Pools

Python uses a system of memory pools to manage small objects more efficiently. Instead of always asking the operating system for memory (which is slow), Python grabs larger chunks of memory and then hands out small pieces from these chunks as needed.

This system helps reduce fragmentation (when memory gets broken up into lots of little pieces) and speeds up how quickly Python can get and release memory.

4. The ‘gc’ Module

Python provides a module called gc that lets you interact with the garbage collector.

You can use this module to enable or disable the garbage collector, force it to run, or check how many objects it’s tracking.

5. Memory Allocation Functions

Under the hood, Python uses various functions to handle memory allocation for objects. These functions are like the behind-the-scenes workers that make sure memory gets allocated and freed as efficiently as possible.

Q-5: What is the purpose of the continue statement in python?

The continue statement in Python is used within loops to skip the rest of the code inside the current iteration and move on to the next iteration of the loop. It's particularly useful when you want to skip certain conditions and continue with the next loop cycle.

Purpose of the continue Statement

The main purpose of the continue statement is to provide a way to skip the remaining code inside the current iteration and proceed with the next iteration. This can help make your code cleaner and avoid deeply nested if statements.