**1. How is memory managed in Python?**

Python uses an automatic **memory management system** that handles allocation and deallocation of memory to objects in your program. The key components are:

* **Memory Allocation:**
  + Python uses a private heap to store all its objects and data structures.
  + The memory manager allocates space for Python objects.
  + Some parts of the memory, like integers and strings, are optimized for speed and managed by a mechanism called **object pooling**.
* **Garbage Collection:**
  + Python employs **garbage collection** to free up memory that is no longer in use.
  + It uses **reference counting** as its primary method. If an object’s reference count drops to zero, it is eligible for garbage collection.
  + To handle cyclic references (e.g., objects referring to each other), Python uses a cyclic garbage collector.
* **Memory Modules:**
  + Modules like gc allow manual control over garbage collection, such as enabling, disabling, or running it explicitly.
  + Memory allocation at a lower level is managed by the allocator, designed specifically for Python objects.

**2. What is the purpose of the continue statement in Python?**

The continue statement is used to **skip the rest of the code** inside the current loop iteration and jump to the next iteration. It is commonly used to bypass specific parts of the loop based on a condition.

**Example:**

for i in range(5):

if i == 2:

print(i)

**Output:**

0

1

3

4

**3. What are negative indexes and why are they used?**

Negative indexes in Python allow you to access elements from the **end of a sequence** (e.g., lists, strings, tuples), rather than from the beginning. This is particularly useful for reverse traversal or when you want to access elements relative to the end of a sequence.

**Example:**

my\_list = [10, 20, 30, 40, 50]

print(my\_list[-1])

print(my\_list[-3])

**Output:**

50

30

**Why Use Negative Indexing?**

* It provides a **clean and readable way** to access elements from the end of a sequence without calculating the length of the list.
* Useful for working with dynamic sequences where the length may change.