**Python Module – 2**

• How memory is managed in Python?

Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the Python memory manager. The essential points about memory management in Python are:

1. **Python Memory Manager**: This manager handles the allocation of memory for Python objects. It uses various strategies to allocate memory blocks.
2. **Garbage Collection**: Python uses automatic garbage collection to manage memory. This means that Python automatically frees up memory that is no longer in use. It uses reference counting and a cyclic garbage collector to track and collect garbage.
3. **Reference Counting**: Each object in Python has a reference count. When this count drops to zero, the memory occupied by the object is released.
4. **Cyclic Garbage Collector**: In addition to reference counting, Python has a cyclic garbage collector that looks for and cleans up reference cycles (groups of objects that reference each other but are not reachable from the outside).
5. **Memory Pools**: Python's memory manager allocates memory in blocks called "pools" to improve efficiency and reduce fragmentation.

• What is the purpose continue statement in python?

The continue statement in Python is used to skip the rest of the code inside a loop for the current iteration only. The loop does not terminate but continues with the next iteration.

for i in range(10):

if i % 2 == 0:

continue

print(i)

• What are negative indexes and why are they used?

Negative indexes in Python are used to access elements from the end of a sequence (like a list, tuple, or string). They allow for a more convenient way to refer to elements relative to the end of the sequence without needing to calculate the positive index.