ASSIGNMENT-2

1. How memory is managed in python?

- Automatic Memory Allocation and Deallocation:
  Python uses a dynamic memory management system that automatically allocates memory when objects are created and deallocates memory when objects are no longer in use.
- Reference Counting:
  Each object in python has an associated reference count, which keeps track of how many references or pointers point to that object.
- Garbage Collection:
  In addition to reference counting, python uses a cyclic garbage collector to detect and collect objects that are no longer reachable even if they have cyclic references. This helps prevent memory leaks caused by circular references.

ASSIGNMENT-2

- Memory Pools:

  Python’s memory manager employs a memory pool mechanism to allocate small memory blocks efficiently. The memory manager maintains a pool of pre-allocated memory blocks of various sizes.
- Memory Fragmentation:

  Python’s memory manager also tries to mitigate memory fragmentation issues by reusing and compacting memory blocks when possible. This helps ensures that memory fragmentation does not cause excessive memory consumption over time .
- Memory Profiling:

  Python provides tools and libraries for memory profiling and analysis. Libraries like ‘tracemaloc’ and third party tools like ‘memory_profiler’ can help developers identity memory usage patterns and optimize their code to reduce memory consumption.

ASSIGNMENT-2

2. What is the purpose of continue statement in python?

- In Python, the `continue` statement is used inside loops (such as `for` loops and `while` loops) to control the flow of the loop. When a `continue` statement is encountered within a loop, it causes the loop to skip the current iteration and immediately move to the next iteration. In other words, it allows you to bypass the remaining code within the current iteration and proceed with the next iteration of the loop.
- The primary purpose of the `continue` statement is to provide a way to selectively skip certain iterations of a loop based on a condition. It is often used when you want to exclude specific elements or cases from processing within a loop.

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

- Negative indexes in Python are used to access elements in a sequence (like a string, list, or tuple)

ASSIGNMENT-2

from the end of the sequence, rather than from the beginning. In Python, the last element of a sequence has an index of -1, the second-to-last element has an index of -2, and so on. Negative indexes provide a convenient way to access elements in reverse order or to refer to elements near the end of a sequence without needing to know its length explicitly.