Module – 1 (Fundamentals of python)

Q.4 : How memory is managed in Python?

Ans: 1. Memory Allocation:

Whenever a new object is created, Python dynamically allocates memory for it. Python allocates a specific memory space for the object, depending on the object’s type and size. The process of allocating memory for an object is known as memory allocation.

2. Memory Deallocation:

When an object is no longer required, Python deallocates the memory allocated to it. Python has a built-in garbage collector that manages memory deallocation automatically. The garbage collector frees up memory occupied by objects that are no longer in use and not referenced by any other part of the program.

3. Reference Counting:

Python uses a technique called reference counting to keep track of objects in memory. Every object has a reference count associated with it that keeps track of the number of references to that object. The reference count is incremented whenever a new reference to an object is created and decremented whenever a reference is deleted. When the reference count of an object reaches zero, the object is no longer in use and can be deallocated.

4. Garbage Collection:

Garbage collection is the process of automatically freeing up memory occupied by objects that are no longer in use. Python’s garbage collector runs periodically to check for objects that are no longer referenced and frees up the memory occupied by them.

5. Memory Fragmentation:

Memory fragmentation occurs when there is a shortage of memory space. Python manages memory fragmentation by keeping track of free memory blocks and allocating memory from these free blocks whenever required.

Python uses a technique called reference counting to keep track of the number of references to an object. When an object is created, its reference count is set to 1. Each time the object is assigned to a new variable or passed as an argument to a function, its reference count is incremented. When a variable goes out of scope or is reassigned, the reference count of the object it was pointing to is decremented. When an object’s reference count reaches zero, it is deallocated from memory.

In addition, Python provides several tools for managing memory, such as the gc module for fine-tuning the garbage collector, the sys.getsizeof() function for determining the size of objects in memory, and the ctypes module for working with C libraries that require manual memory management.

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

Ans: **Continue statement** is a loop control statement that forces to execute the next iteration of the loop while skipping the rest of the code inside the loop for the current iteration only, i.e. when the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped for the current iteration and the next iteration of the loop will begin.

Q.14-: What are negative indexes and why are they used?

Ans-: Negative indexes are a feature in many programming languages, including Python, that allow you to access elements from the end of a list or sequence. In Python, this means you can refer to elements relative to the end of a list, tuple, or string. The index -1 refers to the last element, -2 to the second-to-last, and so on.