***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 Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching the process of efficiently allocating, de-allocating, and coordinating memory So that all the different processes run smoothly and can optimally access different system resources

**Garbage Collectionp;** Garbage collection is a process in which the interpreter frees up the memory when not in use to make it available for other objects. A new object starts its life in the first generation of the garbage collector. If Python executes a garbage collection process on a generation and an object survives, it moves up into a second, older generation the process by which Java programs perform automatic memory management. Java programs compile to bytecode that can be run on a Java Virtual Machine, or JVM for short. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program

**Reference Counting;** Reference counting works by counting the number of times an object is referenced by other objects in the system. When references to an object are removed, the reference count for an object is decremented. When the reference count becomes zero, the object is deallocated

There are 2 ways to get the reference count of the object

1,Using getrefcount from sys module. In Python, by default, variables are passed by reference. Hence, when we run sys. ...

2,Using c\_long.from\_address from ctypes module. In this method, we

pass the memory address of the variable. So, ctypes.

**Memory Allocation in Python;** Memory management in Python **i**nvolves the management of a private heap. A private heap is a portion of memory that is exclusive to the Python process. All Python objects and data structures are stored in the private heap. The operating system cannot allocate this piece of memory to another process In Python, the process of memory allocation and deallocation is handled automatically as there is a garbage collector which is created in python so that users don't have to handle the process of garbage collection

**Stack Memory;** The allocation happens on contiguous blocks of memory. We call it stack memory allocation because the allocation happens in the function call stack. The size of memory to be allocated is known to the compiler and whenever a function is called, its variables get memory allocated on the stack.It is the memory that is only needed inside a particular function or method call. When a function is called, it is added onto the program’s call stack. Any local memory assignments such as variable initializations inside the particular functions are stored temporarily on the function call stack, where it is deleted once the function returns, and the call stack moves on to the next task. This allocation onto a contiguous block of memory is handled by the compiler using predefined routines, and developers do not need to worry about it

**Heap Memory**The memory is allocated during the execution of instructions written by programmers. Note that the name heap has nothing to do with the heap data structure. It is called heap because it is a pile of memory space available to programmers to allocated and de-allocate. The variables are needed outside of method or function calls or are shared within multiple functions globally are stored in Heap memory