# Module -1 (Fundamentals of Python)

## How memory is managed in Python?

All Python objects and data structures are stored in a private heap, which is used for memory management. Internally, the Python memory manager is responsible for managing this private heap. A variety of dynamic storage management issues, including sharing, segmentation, preallocation, and caching, are handled by various components of the Python memory manager.

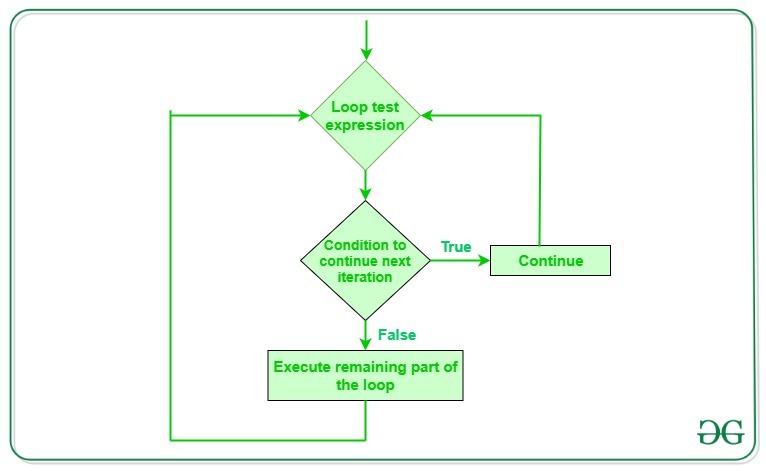
At the most fundamental level, a raw memory allocator works with the operating system's memory management to make sure there is enough space in the private heap to store all Python-related data. In addition to the raw memory allocator, a number of object-specific allocators work on the same heap and apply various memory management strategies according to the unique characteristics of each type of object.

For example, integer objects are managed differently within the heap than strings, tuples or dictionaries because integers imply different storage requirements and speed/space tradeoffs. The Python memory manager thus delegates some of the work to the object-specific allocators, but ensures that the latter operate within the bounds of the private heap. Even though users frequently change object references to memory blocks inside that heap, it is crucial to realize that the Python heap is managed by the interpreter itself and that the user has no influence over it. The Python memory manager uses the Python/C API functions specified in this article to allocate heap space on demand for Python objects and other internal buffers.

Writers of extension code should never attempt to use the malloc(), calloc(), realloc(), and free() functions supplied by the C library to interact with Python objects in order to prevent memory corruption. Due to the fact that they use distinct heaps and implement different algorithms, this will lead to mixed calls between the Python memory management and the C allocator, which will have deadly repercussions. However, using the C library allocator, one can securely allocate and release memory blocks for personal use.

## What is the purpose continue statement in python?

Python Continue Statement skips the execution of the program block after the continue statement and forces the control to start the next iteration. When the continue statement is executed in the loop, the code inside the loop that comes after the continue statement will be skipped for the current iteration and the next iteration of the loop will start. This 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.



## What are negative indexes and why are they used?

The act of indexing from the end of the list with indexing starting at -1, i.e., -1 gives the last element of the list, -2 gives the second last member of the list, and so on, is known as negative indexing.

Negative indexing can be used to show or use data from the list's end, as well as to reverse a number or text without the need for additional operations.