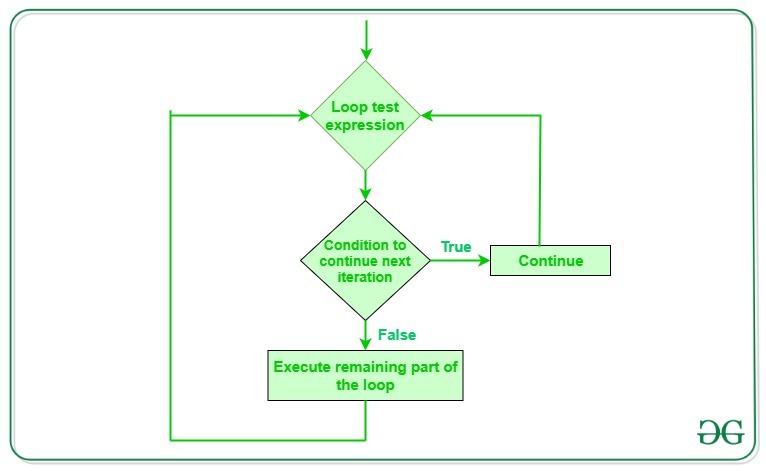
• How memory is managed in Python?

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 Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching.

At the lowest level, a raw memory allocator ensures that there is enough room in the private heap for storing all Python-related data by interacting with the memory manager of the operating system. On top of the raw memory allocator, several object-specific allocators operate on the same heap and implement distinct memory management policies adapted to the peculiarities of every object type. 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.

• What is the purpose continue statement in python?

The continue statement in Python is used to skip the remaining code inside a loop for the current iteration only. Pass: The pass statement in Python is used when a statement or a condition is required to be present in the program, but we don't want any command or code to execute.

**Flowchart of Continue Statement**

• What are negative indexes and why are they used?

Negative indexing allows you to access elements of a sequence from the end, using negative numbers as indexes. This can be useful for getting the last few elements of a sequence, reversing a sequence, or performing other operations that require accessing elements from the end.

