Module – 2 (Fundamentals of python)

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, pre-allocation 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 adaptedto 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 trade-offs. 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.

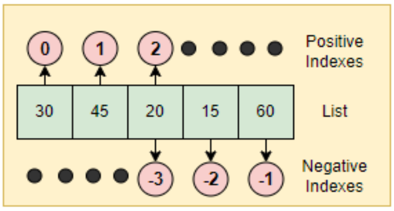
What is the purpose continue statement in python?

The continue keyword is used to end the current iteration in a - for loop (or a while loop), and continues to the next iteration.

Python 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 currentiteration 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.

What are negative indexes and why are they used?

Negative Indexing is used to in Python to begin slicing from the end of the string i.e. the last. Slicing in Python gets a sub-string from a string. The slicing range is set as parameters i.e.start, stop and step.



Negative indexing is used in Python to manipulate sequence objects such as lists, arrays, strings, etc. Negative indexing retrieves elements from the end by providing negative numbers as sequence indexes.