Data Structure in Python

10-08-2024/Saturday

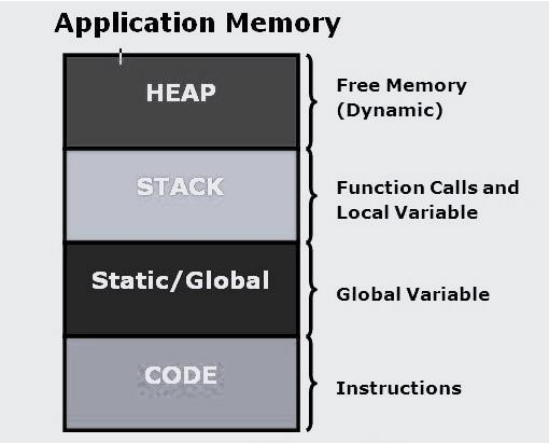
- ## Memory Allocation in Python

Memory allocation in Python is managed automatically, and the process involves a combination of stack memory, heap memory, and specialized memory management mechanisms.

Dynamic memory allocation in Python refers to the process of allocating memory during the runtime of a program, as opposed to static memory allocation, which is determined at compile-time. Python's dynamic nature allows it to handle varying data sizes and structures, making it very flexible for a wide range of programming tasks.

- ## Memory Allocation for Different Types

  - **Immutable Types** (e.g., integers, strings, tuples):
    - These types are stored in the heap memory.
    - Python often optimizes the allocation of small and frequently used immutable objects. For example, small integers and some string literals are cached and reused.
  - **Mutable Types** (e.g., lists, dictionaries, sets):
    - These types are also stored in heap memory.
    - Since they can change in size, Python allocates them dynamically. The memory manager may over-allocate space to minimize the need for resizing operations, which are computationally expensive.



- ## Heap Memory for Dynamic Allocation

  - In Python, dynamic memory allocation occurs in the heap memory. This is where objects like lists, dictionaries, and instances of custom classes are stored.
  - When you create a new object, Python allocates memory for that object from the heap. The size of this memory is determined by the object's type and the data it holds.

- ## Garbage Collection:

Python uses a garbage collector to automatically manage memory. The garbage collector reclaims memory that is no longer in use, preventing memory leaks.