# Memory management

Memory management is 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.

## Stack memory

When a method is called in Python, a stack frame is allocated. This stack frame will handle all the variables of the method. After the method is returned, the stack frame is automatically destroyed.

### Work of 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.

## Heap memory

- All objects and instance variables are stored in the heap memory. When a variable is created in Python, it is stored in a private heap which will then allow for allocation and deallocation.
- The heap memory enables these variables to be accessed globally by all your program's methods. After the variable is returned, the Python garbage collector gets to work, the workings of which we'll cover later.

### Work of Heap Memory

it is a pile of memory space available to programmers to allocate and deallocate. The variables needed outside of method or function calls or are shared within multiple functions globally are stored in Heap memory.

# Garbage Collection

Garbage collection is a process in which the interpreter frees up the memory when not in use to make it available for other objects.
Assume a case where no reference is pointing to an object in memory i.e. it is not in use so, the virtual machine has a garbage collector that automatically deletes that object from the heap memory.