1) How memory managed in python?

Understanding Memory allocation is important to any software developer as writing efficient code means writing a memory-efficient code. Memory allocation can be defined as allocating a block of space in the computer memory to a program. In Python memory allocation and deallocation method is automatic as the Python developers created a **garbage collector** for Python so that the user does not have to do manual garbage collection.

Python Memory Allocation

Memory allocation is an essential part of the memory management for a developer. This process basically allots free space in the computer's virtual memory, and there are two types of virtual memory works while executing programs.

- Static Memory Allocation
- Dynamic Memory Allocation

Python follows dynamic memory allocation for object instances. This is also known as run-time memory allocation as the memory is allocated to the object while the program is running. This is done using a heap data structure.

Static memory is used for method declarations and function call stack. This is implemented using the Stack data structure. This memory is allocated while the program is being interpreted and will not change during the running of the program, which is why it is called static memory.

When a variable is declared in python, mainly two processes take place:

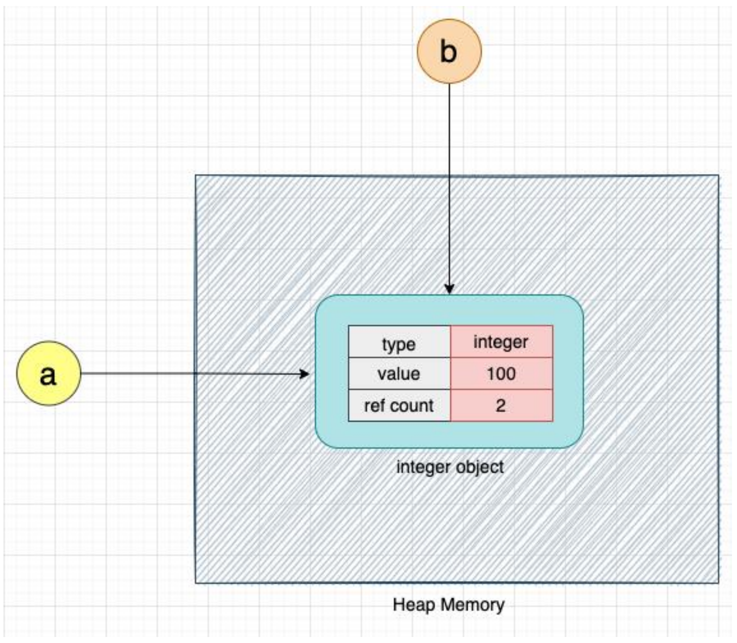
1. An object (or instance) of the respective class is created in the heap memory or runtime memory. It stores the data of the variable.

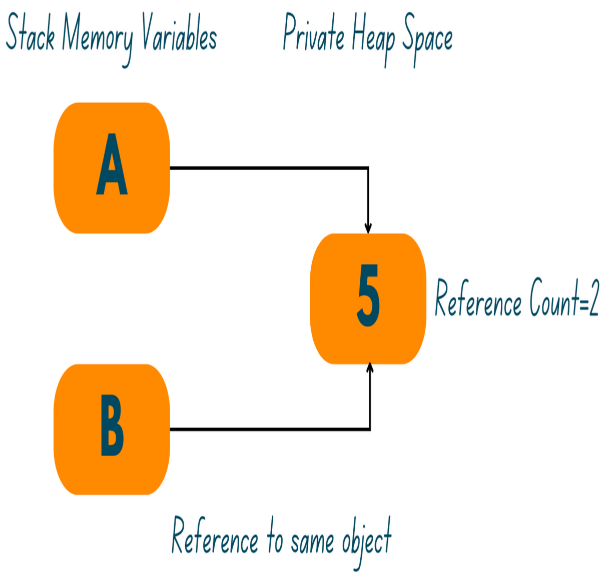
2. A reference to this object is created in the static memory which holds the address of the instance in heap memory.

This process is important to understand if we want to understand the deletion of variables in Python.

Python does not require much manual intervention when it comes to memory management.

All of the memory management is handled by **Python Manager**, which includes allocating memory to all variables and functions as well as maintaining the function call stack.


b
a
type
integer
value
100
ref count
2
integer object
Heap Memory

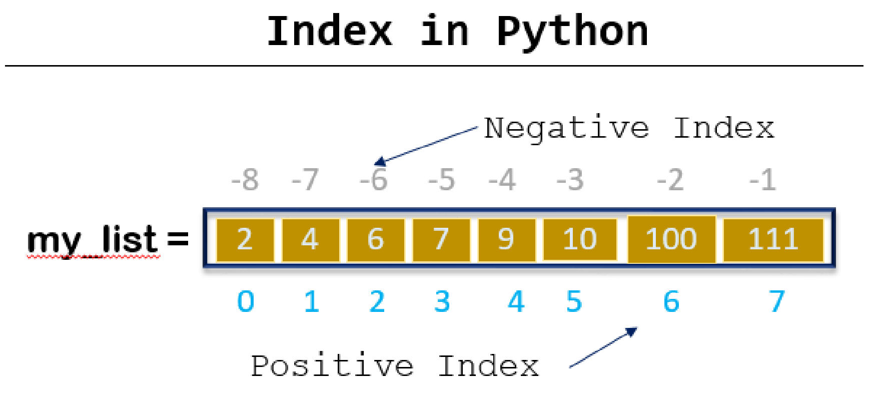

Stack Memory Variables
Private Heap Space
A
5
Reference Count=2
B
Reference to same object

2) What are negative index and why they are use ?

The index() in Python returns the position of the element in the specified list or the characters in the string. It follows the same syntax whether you use it on a list or a string. The reason being that a string in Python is considered as a list of characters starting from the index 0.

indexes are used in arrays in all the programming languages. We can access the elements of an array by going through their indexes. But no programming language allows us to use a negative index value such as -4. Python programming language supports negative indexing of arrays, something which is not available in arrays in most other programming languages. This means that the index value of -1 gives the last element, and -2 gives the second last element of an array. The negative indexing starts from where the array ends. This means that the last element of the array is the first element in the negative indexing which is -1.

```
Example:                                          output:
arr = [10, 20, 30, 40, 50]                        50
print (arr[-1])                                   40
print (arr[-2])
```



The Python index() method helps you find the index position of an element or an item in a string of characters or a list of items.

3) What is the purpose of 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.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.

```
for letter in 'Python':     # First Example
   if letter == 'h':
      continue
   print 'Current Letter :', letter
```

Output:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
```

Loops in Python automate and repeat the tasks efficiently. But sometimes, there may arise a condition where you want to exit the loop completely, skip an iteration or ignore that condition. These can be done by loop control statements. Continue is a type of loop control statement that can alter the flow of the loop.