MODULE -1 FUNDAMENTALS OF PYTHON

[1] How memory is managed in Python?

- 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.

- 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.

Memory Allocation in Python

There are two parts of memory:

- stack memory
- heap memory

The methods/method calls and the references are stored in stack memory and all the values objects are stored in a private heap.

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. When a function is called, it is added onto the program's call stack. Any local memory assignments such as variable initializations inside the particular functions are stored temporarily on the function call stack, where it is deleted once the function returns, and the call stack moves on to the next task. This allocation onto a contiguous block of memory is handled by the compiler using predefined routines, and developers do not need to worry about it.

Example:

```
Def func():
# All these variables get memory
# Allocated on stack
a = 20
b = []
c = ""
```

Work of Heap Memory

- The memory is allocated during the execution of instructions written by programmers. Note that the name heap has nothing to do with the heap data structure. It is called heap because it is a pile of memory space available to programmers to allocated and de-allocate. The variables are needed outside of method or function calls or are shared within multiple functions globally are stored in Heap memory.

Example:

```
# This memory for 10 integers
# is allocated on heap.
a = [0]*10
```

[2] What is the purpose continue statement in python?

Continue: The continue statement in Python is used to skip the remaining code inside a loop for the current iteration only.

The continue statement is used to skip the remaining code inside a loop for the current iteration only.

Syntax:

Continue

Example:

```
for char in “python’:
if (char == ‘y’):
continue
print(“current char: “, char)
```

Output:

```
current char: p
current char: t
current char: h
current char: o
current char: n
```

- In the above example during the second iteration if the condition evaluates to true, then it will execute the continue statement. So whatever statements are present below, for loop will be skipped, hence letter ‘y’ is not printed.

- Therefore, the continue statement works opposite to the break statement. Instead of terminating the loop, it forces it to execute the next iteration of the loop.

[3] What are negative indexes and why are they used?

- 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:

```
arr = [10, 20, 30, 40, 50]

print (arr[-1])

print (arr[-4:-1])

print (arr[-2])
```

Output:

```
50

20,30,40

40
```