Memory Leaks – C++ Vs Python

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Agenda

- What is memory leak?
- Difference in memory in w.r.to C++ and Python
- Code walk through of C++ implementation of Linked list with and without memory leak and its similar implementation in Python
- Insights into python list data structure

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Memory leak in C++

- Memory leaks happen when memory is allocated dynamically in heap and not deallocated in software programs.
- **malloc** or **calloc** are the functions used in C programs to allocate new memory it is de-allocated by using the **free** function.
- In C++, **new** operator is usually used to allocate, and deallocated by the **delete** or the **delete []** operator.
- For normal variables like "**int a**", of "**char c[50]**", etc., the memory is allotted in the **stack** within a function and they are automatically deallocated when the scope of the function ends.
- The memory is allotted in the heap during dynamic allocation. For ex.: "**int *i = new int[10]**"
- The new and new[] will raise an the **std::bad_alloc exception**, when it fails to allocate the memory

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Drawbacks of explicit memory management in C++

- When the memory leaks get accumulated over time and, if left unchecked, may crash a program or cripple it.
- While developing programs like servers and daemons, which by definition never terminate, memory leaks are very serious issues.
- If a memory is prematurely de-allotted and if the control tries to access that memory later, then the program will crash.
- Also if memory is explicitly allotted and deleted frequently, over aperiod of time, might result in memory fragmentation. This will also result in de-graded performance of the application.

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Ways of avoiding memory leaks in C++

- Using smart pointers instead of managing memory manually.
- By using std string library instead of using char*. The string library handles memory leaks internally
- Avoiding the usage of raw pointers as much as possible
- By allocating the memory using **new** keyword and then promptly releasing them by using matching **delete** keyword between start and end of functions

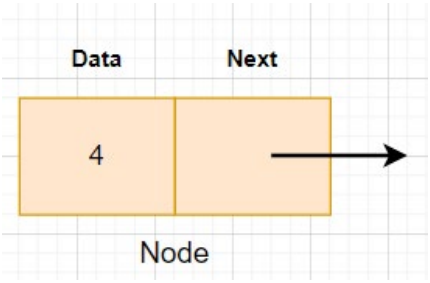
Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Memory leak in Python

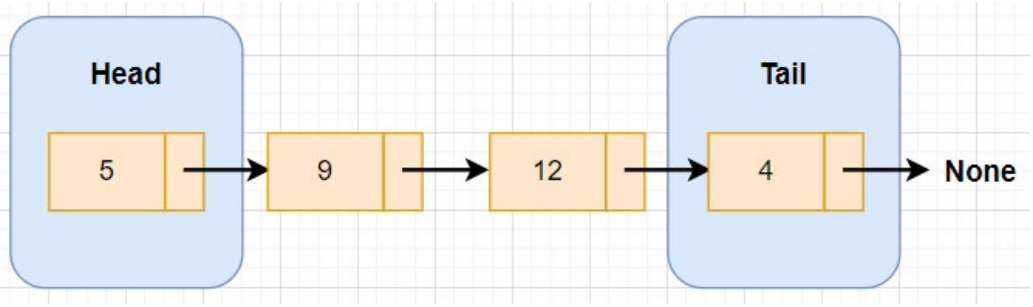
- In python, memory management uses reference counting. when the references of the object reaches zero, the object is garbage collected automatically.
- This essentially means the Python program need not worry about allocation and deallocation of memory explicitly unlike C or C++ program.
- In python everything, be it a basic type or complex type, are represented as objects.
  - Methods and variables are created in **Stack** memory.. These memory are destroyed automatically whenever methods are returned.
  - Objects and instance variables are created in **Heap** memory. When variables and functions returns, the dead objects are garbage collected.

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Linked list



- Linked list is an ordered collection of objects.
- It stores references as part of their own elements.
- An element of a linked list is a node, and every node has two fields:
  - Data
  - A reference to the next node on the list.
- The first node is the head and is the starting point for any iteration in the list.
- The last node will have its next reference pointing to NULL.



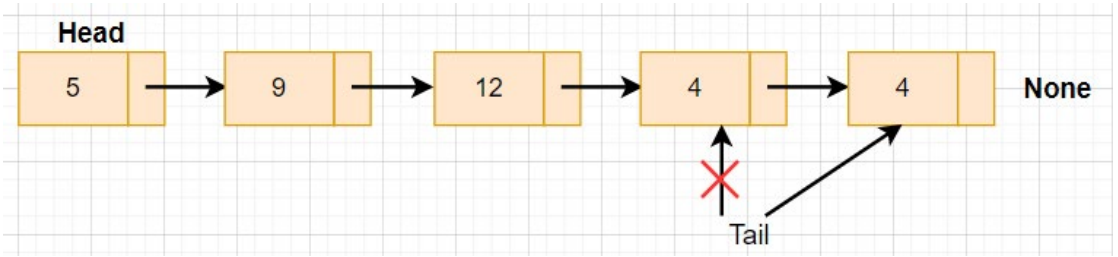
Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

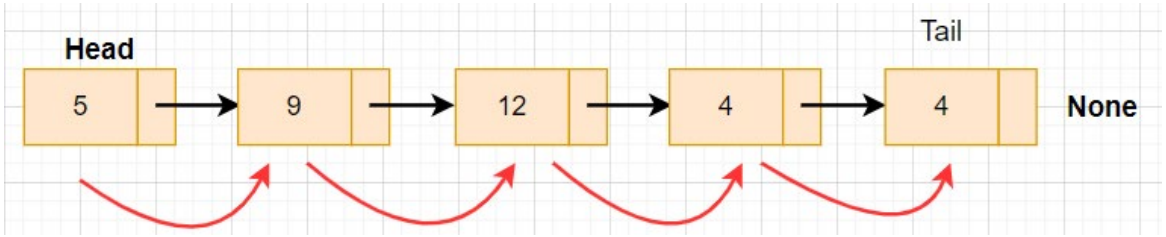
Linked list implementation - Code walkthrough

A structure which defines a single node. It has one variable for data and a pointer for the next node.

Following are the functions needed to handle the nodes

- **Creating a Node**: Allocates memory for a new node, and links it with the tail node and makes the tail pointer to point to the next node
- **Display of all Nodes**: Loops through all the nodes and displays the data, until the next node is NULL.

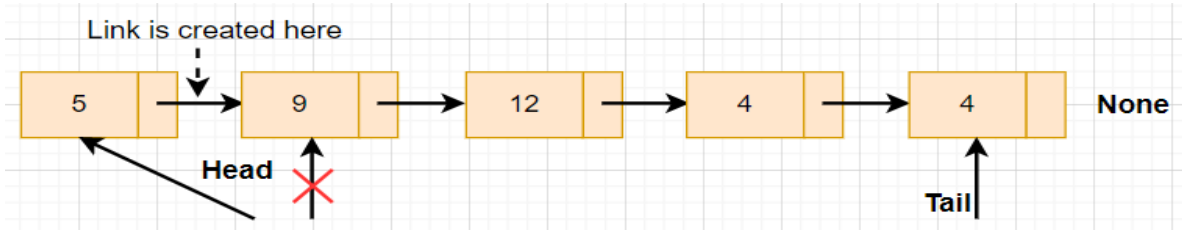




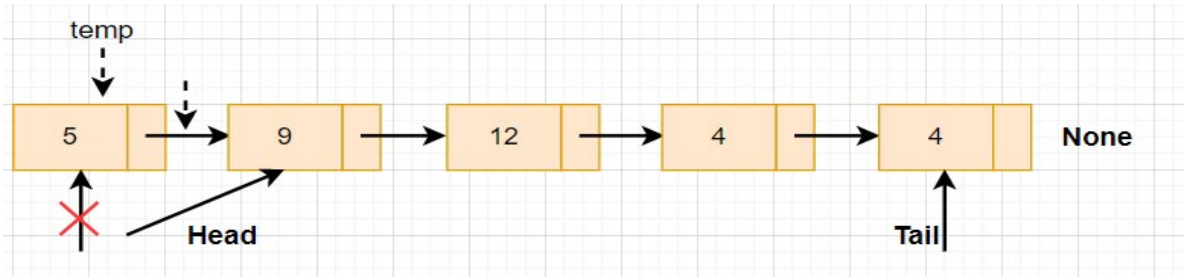
Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Linked list implementation - Code walkthrough

- **Inserting a Node at Start**: New node is connected to first node (Head) by putting the head in the next field of new node.



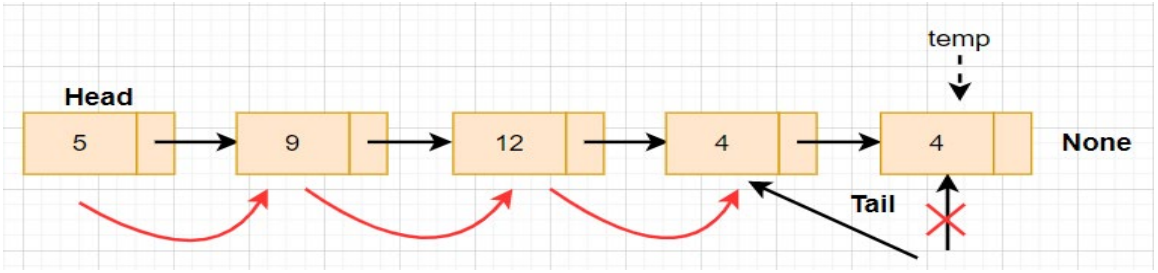
- **Deleting a Node at Start**: The process of deletion includes:
  - Declaring a temporary pointer and pass the head address to this pointer (which is the first node)
  - Declare the second node as head, because this will be the first node of linked list after deletion.
  - Delete the temporary node.



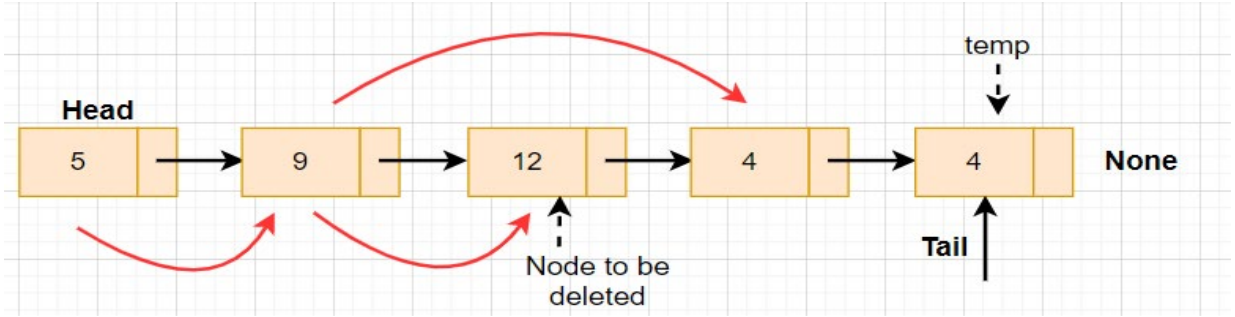
Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Linked list implementation - Code walkthrough

- **Deleting a Node at End** : Need to traverse to the end of the list and move the tail to the last but one node, and the last node should be removed.



- **Deleting a Node at a specified position**: Need to traverse to the position of the list, delete the node and pass the address to the previous pointer.



Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Linked list function in C++ and Python

S.No	Functionality	Function name in C++	Function name in Python
1	Creating a Node	CreateNode	insert_at_first
2	Display of all Nodes	DisplayNodes	print_nodes
3	Inserting a Node at Start	InsertNodeAtStart	insert_at_first
4	Inserting a Node at specified position	InsertNodeAtPosition	insert_at_position
5	Deleting a Node at Start	DeleteNodeAtFirst	insert_at_position
6	Deleting a Node at End	DeleteNodeAtLast	insert_at_last
7	Deleting a Node at a specified position	DeleteNodeAtPosition	remove_at_position
8	Deleting the entire linked list	Destroy	-

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Python List data structure - Insights

- List is a data structure, which just uses a single variable to store many items and they are mutable.
- The collection of items are placed in square braces as shown.
  - L1 = [“Henry", 102, "USA"]
  - L2 = [6, 5, 4, 3, 2, 1]
- A python list could store either variables of single type or multiple types. However with strongly typed programs such as C++, the list can store only single type.

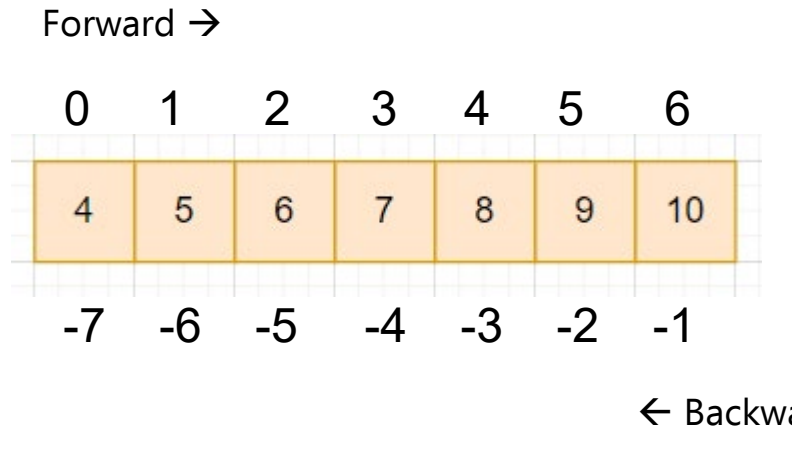
Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Python List – Indexing and Splitting

- The element of the list can be accessed by index using slice operator [ ] and can be split using "**:**" operator for sub-listing as shown below
- Python supports negative indexing also.

  - **print[List[2])** = 6
  - **print(List[0:6])** = [4, 5, 6, 7, 8, 9]
  - **print(List[:])** = [4, 5, 6, 7, 8, 9, 10]
  - **print(List[2:5])** = [6, 7, 8]
  - **print(List[1:6:2])** = [5, 7, 9]
  - **print(List[-1]) =** 10
  - **print(List[-3:])** = [8, 9, 10]
  - **print(List[-3:-1])** = [8, 9]

List = [4, 5, 6, 7, 8, 9, 10]



Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Python List Operations

Consider List1 = [14, 15, 16, 17, 18, 19, 20] and List2 = [111, 112, 113, 114, 15, 116, 117]

Operation	Description	Example
Repetition	Lists elements will be repeated	print(List1*2) = [14, 15, 16, 17, 18, 19, 20, 14, 15, 16, 17, 18, 19, 20]
Concatenation	Concatenates 2 lists	print(List1+List2) = [14, 15, 16, 17, 18, 19, 20, 111, 112, 113, 114, 115, 116, 117]
Membership	Returns true if an item exists, else false	print(15 in List1) = True
Iteration	For is used to iterate items in list	for i in List1: print(i) 14 15 . . 20

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Python List Built-in functions

Consider List1 = [14, 15, 16, 17, 18, 19, 10]

Function	Description	Example
len(list)	Returns the length of list	print(len(List1)) = 7
max(list)	Returns maximum element in list	print(max(List1)) = 19
min(list)	Returns minimum element in list	print(min(List1)) = 14

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Thank you!!

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited