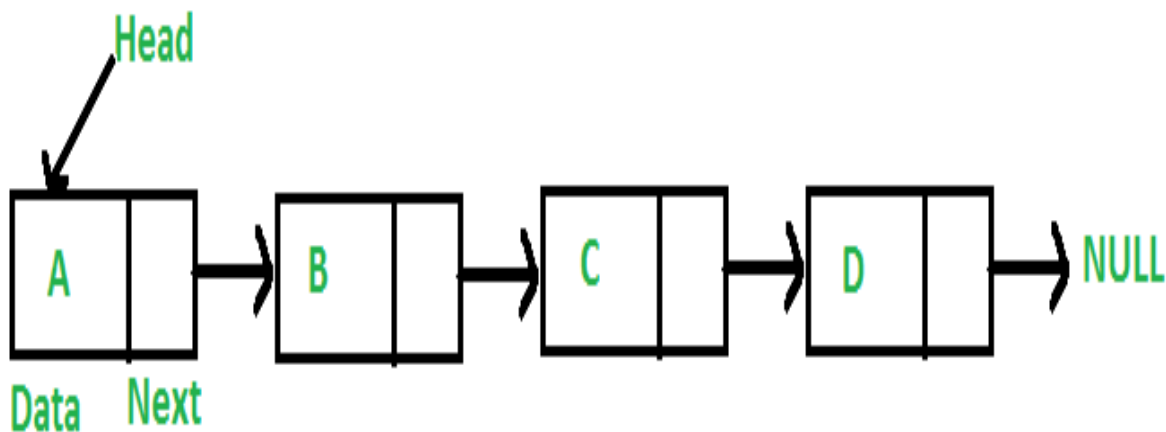




# Linked List | Set 1 (Introduction)

Difficulty Level : Easy • Last Updated : 18 May, 2022

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers.



## Why Linked List?

Arrays can be used to store linear data of similar types, but arrays have the following limitations.

**1)** The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit

**Start Your Coding Journey Now!**

Login

Register

shifted but in Linked list if we have the head node then we can traverse to any node through it and insert new node at the required position.

For example, in a system, if we maintain a sorted list of IDs in an array `id[]`.

`id[] = [1000, 1010, 1050, 2000, 2040]`.


And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).

Deletion is also expensive with arrays until unless some special techniques are used.

For example, to delete 1010 in `id[]`, everything after 1010 has to be moved due to this so much work is being done which affects the efficiency of the code.

### Advantages over arrays

- 1) Dynamic size
- 2) Ease of insertion/deletion



### DS Using Python Programming

By Sandeep Jain

Beginner to Advance Level ★★★★★

Master Data Structures using the Python language. Best suited for working professionals and students who are well versed in Python. Enrol Now!

[Explore Now](#)

### Drawbacks:

- 1) Random access is not allowed. We have to access elements sequentially starting from the first node(head node). So we cannot do binary search with linked lists efficiently with its default implementation. Read about it [here](#).
- 2) Extra memory space for a pointer is required with each element of the list.
- 3) Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

### Representation:

A linked list is represented by a pointer to the first node of the linked list. The first node is called the head. If the linked list is empty, then the value of the head points to NULL.

Each node in a list consists of at least two parts:

- 1) data (we can store integer, strings or any type of data).

**Start Your Coding Journey Now!**

Login

Register

The LinkedList class contains a reference of Node class type.

---

```
// A linked list node
struct Node {
    int data;
    struct Node* next;
};
```

## C++

```
class Node {
public:
    int data;
    Node* next;
};
```

## Java

```
class LinkedList {
    Node head; // head of the list

    /* Linked list Node*/
    class Node {
        int data;
        Node next;

        // Constructor to create a new node
        // Next is by default initialized
        // as null
        Node(int d) { data = d; }
    }
}
```

**Start Your Coding Journey Now!**

Login

Register

```
# Function to initialize the node object
def __init__(self, data):
    self.data = data # Assign data
    self.next = None # Initialize
                      # next as null

# Linked List class
class LinkedList:

    # Function to initialize the Linked
    # List object
    def __init__(self):
        self.head = None
```

## C#

```
class LinkedList {
    // The first node(head) of the linked list
    // Will be an object of type Node (null by default)
    Node head;

    class Node {
        int data;
        Node next;

        // Constructor to create a new node
        Node(int d) { data = d; }
    }
}
```

## Javascript

```
<script>
var head; // head of the list

/* Linked list Node*/
class Node
{

```

```
// Constructor to create a new node
```

**Start Your Coding Journey Now!**

Login

Register

```
}  
}
```

```
// This code is contributed by gauravrajput1  
</script>
```

**First Simple Linked List in C** Let us create a simple linked list with 3 nodes.

```
// A simple CPP program to introduce  
// a linked list  
#include <bits/stdc++.h>  
using namespace std;  
  
class Node {  
public:  
    int data;  
    Node* next;  
};  
  
// Program to create a simple linked  
// list with 3 nodes  
int main()  
,
```



◀ Array Matrix Strings Hashing **Linked List** Stack Queue Binary Tree Binary Search Tree ▶

```
head = new Node();  
second = new Node();  
third = new Node();
```

```
/* Three blocks have been allocated dynamically.  
We have pointers to these three blocks as head,  
second and third
```

```
head      second      third  
|          |          |
```

**Start Your Coding Journey Now!**

Login

Register

Data is random because we haven't assigned anything yet \*/

```
head->data = 1; // assign data in first node
head->next = second; // Link first node with
// the second node
```

/\* data has been assigned to the data part of first block (block pointed by the head). And next pointer of the first block points to second. So they both are linked.

```

head          second          third
  |            |              |
  |            |              |
+---+---+    +---+---+    +---+---+
| 1 | o----->| # | # |    | # | # |
+---+---+    +---+---+    +---+---+
*/
```

```
// assign data to second node
second->data = 2;
```

```
// Link second node with the third node
second->next = third;
```

/\* data has been assigned to the data part of the second block (block pointed by second). And next pointer of the second block points to the third block. So all three blocks are linked.

```

head          second          third
  |            |              |
  |            |              |
+---+---+    +---+---+    +---+---+
| 1 | o----->| 2 | o----->| # | # |
+---+---+    +---+---+    +---+---+    */
```

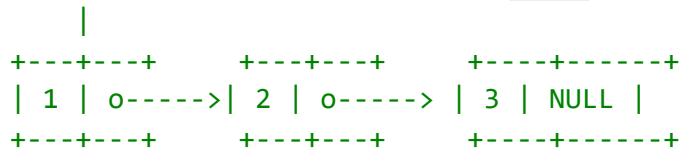
```
third->data = 3; // assign data to third node
third->next = NULL;
```

/\* data has been assigned to the data part of the third block (block pointed by third). And next pointer of the third block is made NULL to indicate

**Start Your Coding Journey Now!**

Login

Register



Note that only the head is sufficient to represent the whole list. We can traverse the complete list by following the next pointers. \*/

```

return 0;
}

// This code is contributed by rathbhupendra

```

## C

```

// A simple C program to introduce
// a linked list
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

// Program to create a simple linked
// list with 3 nodes
int main()
{
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;

    // allocate 3 nodes in the heap
    head = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));

    /* Three blocks have been allocated dynamically.
    We have pointers to these three blocks as head.

```

Start Your Coding Journey Now!

Login

Register

```

| # | # |      | # | # |      | # | # |
+---+---+      +---+---+      +---+---+

```

# represents any random value.

Data is random because we haven't assigned anything yet \*/

```

head->data = 1; // assign data in first node
head->next = second; // Link first node with
// the second node

```

```

/* data has been assigned to the data part of the first
   block (block pointed by the head). And next
   pointer of first block points to second.
   So they both are linked.

```

```

      head          second          third
      |             |             |
      |             |             |
+---+---+      +---+---+      +---+---+
| 1 | o----->| # | # |      | # | # |
+---+---+      +---+---+      +---+---+
*/

```

```

// assign data to second node
second->data = 2;

```

```

// Link second node with the third node
second->next = third;

```

```

/* data has been assigned to the data part of the second
   block (block pointed by second). And next
   pointer of the second block points to the third
   block. So all three blocks are linked.

```

```

      head          second          third
      |             |             |
      |             |             |
+---+---+      +---+---+      +---+---+
| 1 | o----->| 2 | o----->| # | # |
+---+---+      +---+---+      +---+---+      */

```

```

third->data = 3; // assign data to third node
third->next = NULL;

```

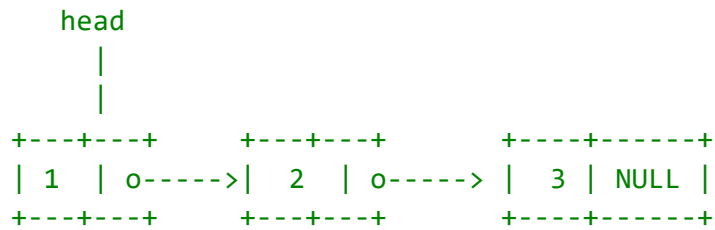
**Start Your Coding Journey Now!**

Login

Register



We have the linked list ready.



Note that only head is sufficient to represent the whole list. We can traverse the complete list by following next pointers. \*/

```

return 0;
}

```

## Java

```

// A simple Java program to introduce a linked list
class LinkedList {
    Node head; // head of list

    /* Linked list Node. This inner class is made static so that
    main() can access it */
    static class Node {
        int data;
        Node next;
        Node(int d)
        {
            data = d;
            next = null;
        } // Constructor
    }

    /* method to create a simple linked list with 3 nodes*/
    public static void main(String[] args)
    {
        /* Start with the empty list. */
        LinkedList llist = new LinkedList();

        llist.head = new Node(1);

```

Start Your Coding Journey Now!

Login

Register

second and third

```

l1list.head      second      third
  |              |           |
  |              |           |
+---+---+---+   +---+---+---+   +---+---+---+
| 1 | null |    | 2 | null |    | 3 | null |
+---+---+---+   +---+---+---+   +---+---+---+ */

```

```
l1list.head.next = second; // Link first node with the second node
```

```
/* Now next of the first Node refers to the second. So they
both are linked.
```

```

l1list.head      second      third
  |              |           |
  |              |           |
+---+---+---+   +---+---+---+   +---+---+---+
| 1 | o----->| 2 | null |    | 3 | null |
+---+---+---+   +---+---+---+   +---+---+---+ */

```

```
second.next = third; // Link second node with the third node
```

```
/* Now next of the second Node refers to third. So all three
nodes are linked.
```

```

l1list.head      second      third
  |              |           |
  |              |           |
+---+---+---+   +---+---+---+   +---+---+---+
| 1 | o----->| 2 | o----->| 3 | null |
+---+---+---+   +---+---+---+   +---+---+---+ */

```

```
}
```

```
}
```

## Python

```
# A simple Python program to introduce a linked list
```

```
# Node class
```

```
class Node:
```

**Start Your Coding Journey Now!**

Login

Register

# Linked List class contains a Node object

```
class LinkedList:
```

```
    # Function to initialize head
```

```
    def __init__(self):
        self.head = None
```

# Code execution starts here

```
if __name__ == '__main__':
```

```
    # Start with the empty list
```

```
    llist = LinkedList()
```

```
    llist.head = Node(1)
```

```
    second = Node(2)
```

```
    third = Node(3)
```

```
    ...
```

Three nodes have been created.

We have references to these three blocks as head, second and third

llist.head	second	third
+---+---+	+---+---+	+---+---+
1   None	2   None	3   None
+---+---+	+---+---+	+---+---+
...		

```
llist.head.next = second; # Link first node with second
```

```
...
```

Now next of first Node refers to second. So they both are linked.

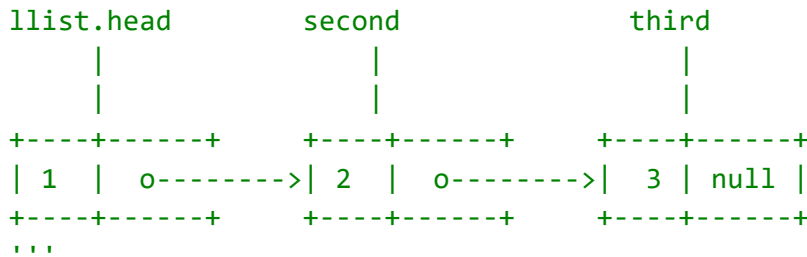
llist.head	second	third
+---+---+	+---+---+	+---+---+
1   o----->	2   null	3   null
+---+---+	+---+---+	+---+---+

**Start Your Coding Journey Now!**

Login

Register

nodes are linked.



## C#

```
// A simple C# program to introduce a linked list
using System;

public class LinkedList {
    Node head; // head of list

    /* Linked list Node. This inner class is made static so that
    main() can access it */
    public class Node {
        public int data;
        public Node next;
        public Node(int d)
        {
            data = d;
            next = null;
        } // Constructor
    }

    /* method to create a simple linked list with 3 nodes*/
    public static void Main(String[] args)
    {
        /* Start with the empty list. */
        LinkedList llist = new LinkedList();

        llist.head = new Node(1);
        Node second = new Node(2);
        Node third = new Node(3);

        /* Three nodes have been allocated dynamically.
        We have references to these three blocks as head.

```

Start Your Coding Journey Now!

Login

Register

```

+---+-----+      +---+-----+      +---+-----+
| 1 | null |      | 2 | null |      | 3 | null |
+---+-----+      +---+-----+      +---+-----+ */

```

```
l1list.head.next = second; // Link first node with the second node
```

```
/* Now next of first Node refers to second. So they
   both are linked.
```

```

l1list.head      second      third
  |              |           |
  |              |           |
+---+-----+    +---+-----+    +---+-----+
| 1 | o----->| 2 | null |      | 3 | null |
+---+-----+    +---+-----+    +---+-----+ */

```

```
second.next = third; // Link second node with the third node
```

```
/* Now next of the second Node refers to third. So all three
   nodes are linked.
```

```

l1list.head      second      third
  |              |           |
  |              |           |
+---+-----+    +---+-----+    +---+-----+
| 1 | o----->| 2 | o----->| 3 | null |
+---+-----+    +---+-----+    +---+-----+ */

```

```
}
```

```
}
```

```
// This code has been contributed by 29AjayKumar
```

## Javascript

```
<script>
```

```
// A simple javascript program to introduce a linked list
```

```
var head; // head of list
```

```
/* Linked list Node. This inner class is made so that
   main() can access it */
```

```
class Node {
```

**Start Your Coding Journey Now!**

Login

Register

```
}
```

```
/* method to create a simple linked list with 3 nodes*/
```

```
var head = new Node(1);
var second = new Node(2);
var third = new Node(3);
```

```
/* Three nodes have been allocated dynamically.
   We have references to these three blocks as head,
   second and third
```

```

l1list.head      second      third
  |              |           |
  |              |           |
+---+---+---+   +---+---+---+   +---+---+---+
| 1 | null |    | 2 | null |    | 3 | null |
+---+---+---+   +---+---+---+   +---+---+---+ */
```

```
head.next = second; // Link first node with the second node
```

```
/* Now next of the first Node refers to the second. So they
   both are linked.
```

```

l1list.head      second      third
  |              |           |
  |              |           |
+---+---+---+   +---+---+---+   +---+---+---+
| 1 | o----->| 2 | null |    | 3 | null |
+---+---+---+   +---+---+---+   +---+---+---+ */
```

```
second.next = third; // Link second node with the third node
```

```
/* Now next of the second Node refers to third. So all three
   nodes are linked.
```

```

l1list.head      second      third
  |              |           |
  |              |           |
+---+---+---+   +---+---+---+   +---+---+---+
| 1 | o----->| 2 | o----->| 3 | null |
+---+---+---+   +---+---+---+   +---+---+---+ */
```

## Start Your Coding Journey Now!

[Login](#)
[Register](#)

## Linked List Traversal

In the previous program, we have created a simple linked list with three nodes. Let us traverse the created list and print the data of each node. For traversal, let us write a general-purpose function `printList()` that prints any given list.

[We strongly recommend that you click here and practice it, before moving on to the solution.](#)

---

```
// A simple C++ program for traversal of a linked list
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int data;
    Node* next;
};

// This function prints contents of linked list
// starting from the given node
void printList(Node* n)
{
    while (n != NULL) {
        cout << n->data << " ";
        n = n->next;
    }
}

// Driver code
int main()
{
    Node* head = NULL;
    Node* second = NULL;
    Node* third = NULL;
```

**Start Your Coding Journey Now!**

Login

Register

```
head->data = 1; // assign data in first node
head->next = second; // Link first node with second

second->data = 2; // assign data to second node
second->next = third;

third->data = 3; // assign data to third node
third->next = NULL;

printList(head);

return 0;
}

// This code is contributed by rathbhupendra
```

## C

```
// A simple C program for traversal of a linked list
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

// This function prints contents of linked list starting from
// the given node
void printList(struct Node* n)
{
    while (n != NULL) {
        printf(" %d ", n->data);
        n = n->next;
    }
}

int main()
{
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;
```

Start Your Coding Journey Now!

Login

Register



```
head->data = 1; // assign data in first node
head->next = second; // Link first node with second

second->data = 2; // assign data to second node
second->next = third;

third->data = 3; // assign data to third node
third->next = NULL;

printList(head);

return 0;
}
```

## Java

```
// A simple Java program for traversal of a linked list
class LinkedList {
    Node head; // head of list

    /* Linked list Node. This inner class is made static so that
       main() can access it */
    static class Node {
        int data;
        Node next;
        Node(int d)
        {
            this.data = d;
            next = null;
        } // Constructor
    }

    /* This function prints contents of linked list starting from head */
    public void printList()
    {
        Node n = head;
        while (n != null) {
            System.out.print(n.data + " ");
            n = n.next;
        }
    }
}
```

**Start Your Coding Journey Now!**

Login

Register

```
LinkedList llist = new LinkedList();

llist.head = new Node(1);
Node second = new Node(2);
Node third = new Node(3);

llist.head.next = second; // Link first node with the second node
second.next = third; // Link second node with the third node

llist.printList();
}
}
```

## Python3

# A simple Python program for traversal of a linked list

# Node class

**class** Node:

# Function to initialise the node object

**def** \_\_init\_\_(self, data):

self.data = data # Assign data

self.next = None # Initialize next as null

# Linked List class contains a Node object

**class** LinkedList:

# Function to initialize head

**def** \_\_init\_\_(self):

self.head = None

# This function prints contents of linked list

# starting from head

**def** printList(self):

temp = self.head

**while** (temp):

**print** (temp.data)

temp = temp.next

**Start Your Coding Journey Now!**

Login

Register

```
l1list.head = Node(1)
second = Node(2)
third = Node(3)

l1list.head.next = second; # Link first node with second
second.next = third; # Link second node with the third node

l1list.printList()
```

## C#

```
// A simple C# program for traversal of a linked list
using System;
```

```
public class LinkedList {
    Node head; // head of list

    /* Linked list Node. This inner
    class is made static so that
    main() can access it */
    public class Node {
        public int data;
        public Node next;
        public Node(int d)
        {
            data = d;
            next = null;
        }
    } // Constructor
}

/* This function prints contents of
linked list starting from head */
public void printList()
{
    Node n = head;
    while (n != null) {
        Console.Write(n.data + " ");
        n = n.next;
    }
}
```

Start Your Coding Journey Now!

Login

Register

```
LinkedList llist = new LinkedList();

l1list.head = new Node(1);
Node second = new Node(2);
Node third = new Node(3);

l1list.head.next = second; // Link first node with the second node
second.next = third; // Link second node with the third node

l1list.printList();
}
}

/* This code contributed by PrinciRaj1992 */
```

## Javascript

```
<script>
// A simple javascript program for traversal of a linked list
var head; // head of list

/* Linked list Node. This inner class is made so that
main() can access it */
class Node {
    constructor(val) {
        this.data = val;
        this.next = null;
    }
}

/* This function prints contents of linked list starting from head */
function printList()
{
    var n = head;
    while (n != null) {
        document.write(n.data + " ");
        n = n.next;
    }
}

/* method to create a simple linked list with 3 nodes*/
```

**Start Your Coding Journey Now!**

Login

Register

```
var second = new Node(2);  
var third = new Node(3);  
  
head.next = second; // Link first node with the second node  
second.next = third; // Link second node with the third node  
  
printList();
```

```
// This code contributed by gauravrajput1  
</script>
```

### Output:

1 2 3

### Important Links :

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

**Curated by experts.**  
**Trusted by 1 Lac+ students.**  
Data Structures & Algorithms Self-Paced Course  
[Enrol Now](#)



**Start Your Coding Journey Now!**

[Login](#)

[Register](#)

ADVERTISEMENT BY ADRECOVER

**Elevate your job search.**

Dice delivers industry-leading tools and resources to help you land the right job in tech.

Dice

## RECOMMENDED ARTICLES

Page : 1 2 3

**01** XOR Linked List - A Memory Efficient Doubly Linked List | Set 1  
23, May 11

**05** Convert singly linked list into circular linked list  
22, Sep 17

**02** XOR Linked List - A Memory Efficient Doubly Linked List | Set 2  
14, Jun 12

**06** Difference between Singly linked list and Doubly linked list  
07, Jan 19

**03** Merge a linked list into another linked list at alternate positions  
20, Aug 13

**07** Convert Singly Linked List to XOR Linked List  
18, Jan 19

**04** Check if a linked list is Circular Linked List

**08** Create new linked list from two given linked list with greater element at each node

**Start Your Coding Journey Now!**[Login](#)[Register](#)

## Article Contributed By :



GeeksforGeeks

## Vote for difficulty

Current difficulty : [Easy](#)

[Easy](#)[Normal](#)[Medium](#)[Hard](#)[Expert](#)

**Improved By :** [ashwani khemani](#), [tobamaestro](#), [princiraj1992](#), [rathbhupendra](#), [29AjayKumar](#), [NikhilKoyikkamannil](#), [Royalcoder](#), [rakshitparashar1](#), [prakashprakhhar2021](#), [sahilkansal09](#), [GauravRajput1](#), [adityarajawasthi1602](#), [umadevi9616](#)

**Article Tags :** [Python-Data-Structures](#), [Linked List](#)

**Practice Tags :** [Linked List](#)

[Improve Article](#)[Report Issue](#)

Writing code in comment? Please use [ide.geeksforgeeks.org](https://ide.geeksforgeeks.org), generate link and share the link here.


[Load Comments](#)


ADVERTISEMENT BY ADRECOVER



## Start Your Coding Journey Now!

[Login](#)[Register](#)

 5th Floor, A-118,  
Sector-136, Noida, Uttar Pradesh - 201305

 [feedback@geeksforgeeks.org](mailto:feedback@geeksforgeeks.org)



## Company

[About Us](#)  
[Careers](#)  
[In Media](#)  
[Contact Us](#)  
[Privacy Policy](#)  
[Copyright Policy](#)

## Learn

[Algorithms](#)  
[Data Structures](#)  
[SDE Cheat Sheet](#)  
[Machine learning](#)  
[CS Subjects](#)  
[Video Tutorials](#)

## News

[Top News](#)  
[Technology](#)  
[Work & Career](#)  
[Business](#)  
[Finance](#)  
[Lifestyle](#)

## Languages

[Python](#)  
[Java](#)  
[CPP](#)  
[Golang](#)  
[C#](#)  
[SQL](#)

## Web Development

[Web Tutorials](#)  
[Django Tutorial](#)  
[HTML](#)  
[CSS](#)

## Contribute

[Write an Article](#)  
[Improve an Article](#)  
[Pick Topics to Write](#)  
[Write Interview Experience](#)

**Start Your Coding Journey Now!**

[Login](#)

[Register](#)