



Doubly Linked List in Python

Last Updated : 03 Jun, 2024

Doubly Linked List is a type of linked list in which each node contains a data element and two links pointing to the next and previous node in the sequence. This allows for more efficient operations such as traversals, insertions, and deletions because it can be done in both directions.

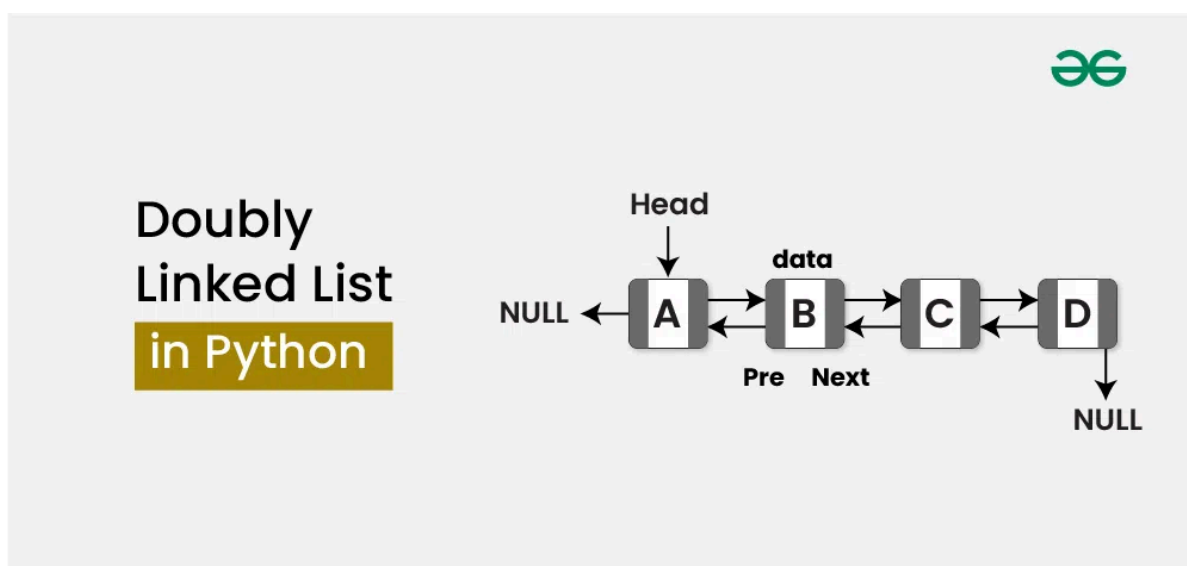


Table of Content

- [What is a Doubly Linked List?](#)
- [Representation of Doubly Linked List in Python](#)
- [Traversal of Doubly Linked List in Python](#)
- [Insertion of Doubly Linked List in Python](#)
- [Deletion of Doubly Linked List in Python](#)
- [Related Practice Problem on Doubly Linked List](#)

What is a Doubly Linked List?

Doubly Linked List (DLL) is a special type of linked list in which each node contains a pointer to the previous node as well as the next node of the linked

Representation of Doubly Linked List in Python:

Here is the representation of the doubly linked list in python:

Python



```
1 # Node of a doubly linked list
2 class Node:
3     def __init__(self, next=None, prev=None, data=None):
4         # reference to next node in DLL
5         self.next = next
6         # reference to previous node in DLL
7         self.prev = prev
8         self.data = data
```

Traversal of Doubly Linked List in Python:

To traverse a doubly linked list in Python, you can simply start from the head of the list and iterate through each node, printing its data.

Below is the implementation of the above idea:

Python



```
1 # Python Program for traversal of a doubly linked list
2 class Node:
3     def __init__(self, data):
4         # Initialize a new node with data, previous, and next
5         # pointers
6         self.data = data
7         self.next = None
8         self.prev = None
9
10 def traverse(head):
11     # Traverse the doubly linked list and print its elements
12     current = head
13     while current:
14         # Print current node's data
15         print(current.data, end=" <-> ")
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
18     print("None")
19
20
21 def insert_at_beginning(head, data):
22     # Insert a new node at the beginning of the doubly linked
23     new_node = Node(data)
24     new_node.next = head
25     if head:
26         head.prev = new_node
27     return new_node
28
29
30 # Driver Code
31 head = None
32 head = insert_at_beginning(head, 4)
33 head = insert_at_beginning(head, 3)
34 head = insert_at_beginning(head, 2)
35 head = insert_at_beginning(head, 1)
36
37 # To traverse and print the nodes:
38 traverse(head)
```

Output

1 <-> 2 <-> 3 <-> 4 <-> None

Insertion of Doubly Linked List in Python:

Inserting a new node in a doubly linked list is very similar to inserting new node in linked list. There is a little extra work required to maintain the link of the previous node. A node can be inserted in a Doubly Linked List in five ways:

- At the front of the DLL.
- After a given node.
- Before a given node.
- At the end of the DLL.

1. Insertion at the Beginning:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

To insert a node at the beginning of a doubly linked list in Python, you need to follow these steps:

- Create a new node with the given data.
- Set the "**next**" pointer of the new node to point to the current head (if any).
- Set the "**previous**" pointer of the new node to None (as it will become the new head).
- If the list is not empty, update the "**previous**" pointer of the current head to point to the new node.
- Update the head of the list to point to the new node.

Below is the implementation of the above idea:

Python

```
1  # Python Program for a doubly linked list at the beginning of
   node
2  class Node:
3      def __init__(self, data):
4          self.data = data
5          self.next = None
6          self.prev = None
7
8  # Function to insert a node at the beginning of a doubly linke
   list
9  def insert_at_beginning(head, data):
10     new_node = Node(data)
11     new_node.next = head
12     if head:
13         head.prev = new_node
14     return new_node
15
16 # Function to display the elements of the doubly linked list
17 def display(head):
18     current = head
19     while current:
20         print(current.data, end=" <-> ")
21         current = current.next
22     print("None")
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
25 # Driver Code
26 head = None
27 head = insert_at_beginning(head, 3)
28 head = insert_at_beginning(head, 2)
29 head = insert_at_beginning(head, 1)
30
31 print("Doubly Linked List after insertion at the beginning:")
32 display(head)
```

Output

Doubly Linked List after insertion at the beginning:

1 <-> 2 <-> 3 <-> None

2. Insertion after a given node:

To insert a node after a given node in a doubly linked list in Python, you can follow these steps:

- Create a new node with the given data.
- Set the "**next**" pointer of the new node to point to the next node of the given node.
- Set the "**previous**" pointer of the new node to point to the given node.
- If the next node of the given node is not None, update the "**previous**" pointer of that node to point to the new node.
- Update the "**next**" pointer of the given node to point to the new node.

Below is the implementation of the above idea:

Python

```
1 # Python Program for Insertion after a given node
2 class Node:
3     def __init__(self, data):
4         self.data = data
5         self.next = None
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
8 # Function to insert a node after a given node in a doubly lin
list
9 def insert_after_node(node, data):
10     if node is None:
11         print("Error: The given node is None")
12         return
13
14     new_node = Node(data)
15     new_node.prev = node
16     new_node.next = node.next
17
18     if node.next:
19         node.next.prev = new_node
20
21     node.next = new_node
22
23 # Function to display the elements of the doubly linked list
24 def display(head):
25     current = head
26     while current:
27         print(current.data, end=" <-> ")
28         current = current.next
29     print("None")
30
31
32 # Driver Code
33 head = Node(1)
34 node2 = Node(2)
35 node3 = Node(3)
36
37 head.next = node2
38 node2.prev = head
39 node2.next = node3
40 node3.prev = node2
41
42 print("Doubly Linked List before insertion:")
43 display(head)
44
45 insert_after_node(node2, 4)
46
47 print("Doubly Linked List after insertion:")
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

Output

Doubly Linked List before insertion:

1 <-> 2 <-> 3 <-> None

Doubly Linked List after insertion:

1 <-> 2 <-> 4 <-> 3 <-> None

3. Insertion before a given node:

To insert a node before a given node in a doubly linked list in Python, you can follow these steps:

- Create a new node with the given data.
- Set the "**next**" pointer of the new node to point to the given node.
- Set the "**previous**" pointer of the new node to point to the previous node of the given node.
- If the previous node of the given node is not None, update the "**next**" pointer of that node to point to the new node.
- Update the "**previous**" pointer of the given node to point to the new node.

Below is the implementation of the above idea:

Python



```
1 # Python Program for Insertion before a given node
```

```
2 class Node:
```

```
4     self.data = data
```

```
5     self.next = None
```

```
6     self.prev = None
```

```
7
```

```
8 # Function to insert a node before a given node in a doubly li  
list
```

```
9 def insert_before_node(node, data):
```

```
10     if node is None:
```

```
11         print("Error: The given node is None")
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

```
15     new_node.prev = node.prev
16     new_node.next = node
17
18     if node.prev:
19         node.prev.next = new_node
20
21     node.prev = new_node
22
23 # Function to display the elements of the doubly linked list
24 def display(head):
25     current = head
26     while current:
27         print(current.data, end=" <-> ")
28         current = current.next
29     print("None")
30
31 # Driver Code
32 head = Node(1)
33 node2 = Node(2)
34 node3 = Node(3)
35
36 head.next = node2
37 node2.prev = head
38 node2.next = node3
39 node3.prev = node2
40
41 print("Doubly Linked List before insertion:")
42 display(head)
43
44 insert_before_node(node2, 4)
45
46 print("Doubly Linked List after insertion:")
47 display(head)
```

Output

Doubly Linked List before insertion:

1 <-> 2 <-> 3 <-> None

Doubly Linked List after insertion:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

4. Insertion at the end:

To insert a node at the end of a doubly linked list in Python, you need to follow these steps:

- Create a new node with the given data.
- If the list is empty (head is None), make the new node the head of the list.
- Otherwise, traverse the list to find the last node.
- Set the "**next**" pointer of the last node to point to the new node.
- Set the "**previous**" pointer of the new node to point to the last node.
- Optionally, update the head of the list to point to the new node if it's the first node in the list.

Below is the implementation of the above idea:

Python

```
1 # Python Program for Insertion at the end
2 class Node:
3     def __init__(self, data):
4         # Initialize a new node with data, previous, and next
          pointers
5         self.data = data
6         self.next = None
7         self.prev = None
8
9
10 def insert_at_end(head, data):
11     # Insert a new node at the end of the doubly linked list
12     new_node = Node(data)
13     if head is None:
14         return new_node
15
16     current = head
17     while current.next:
18         current = current.next
19
20     current.next = new_node
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
24
25 def display(head):
26     # Display the doubly linked list elements
27     current = head
28     while current:
29         # Print current node's data
30         print(current.data, end=" <-> ")
31         # Move to the next node
32         current = current.next
33     print("None")
34
35
36 # Driver Code
37 head = None
38 head = insert_at_end(head, 1)
39 head = insert_at_end(head, 2)
40 head = insert_at_end(head, 3)
41
42 print("Doubly Linked List after insertion at the end:")
43 display(head)
```

Output

Doubly Linked List after insertion at the end:

1 <-> 2 <-> 3 <-> None

Deletion of Doubly Linked List in Python:

Deletion of a node in Doubly Linked List generally involves modifying the next and the previous pointers of nodes. Deletion can be done in 3 ways:

- At the beginning of DLL
- At the end of DLL
- At a given position in DLL

1. Deletion at the beginning:

To delete a node from the beginning of a doubly linked list in Python, you need

- Check if the list is empty (**head is None**). If it is empty, there is nothing to delete.
- If the list has only one node, set the head to None to delete the node.
- Otherwise, update the head to point to the next node.
- Set the "**previous**" pointer of the new head to None.
- Optionally, free the memory allocated to the deleted node.

Below is the implementation of the above idea:

Python

```
1  # Python Program for the deletion at the beginning
2  class Node:
3      def __init__(self, data):
4          # Initialize a new node with data, previous, and next
5          self.data = data
6          self.next = None
7          self.prev = None
8
9  def delete_at_beginning(head):
10     # Delete the first node from the beginning of the doubly l
11     list
12     if head is None:
13         print("Doubly linked list is empty")
14         return None
15
16     if head.next is None:
17         return None
18
19     new_head = head.next
20     new_head.prev = None
21     del head
22     return new_head
23
24 def traverse(head):
25     # Traverse the doubly linked list and print its elements
26     current = head
27     while current:
28         # Print current node's data
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```

30         current = current.next
31     print("None")
32
33     def insert_at_beginning(head, data):
34         # Insert a new node at the beginning of the doubly linked
35         new_node = Node(data)
36         new_node.next = head
37         if head:
38             head.prev = new_node
39         return new_node
40
41     # Driver Code
42     head = None
43     head = insert_at_beginning(head, 4)
44     head = insert_at_beginning(head, 3)
45     head = insert_at_beginning(head, 2)
46     head = insert_at_beginning(head, 1)
47
48     # Delete the first node (node with data 1) from the beginning:
49     head = delete_at_beginning(head)
50
51     # Traverse and print the nodes after deletion:
52     traverse(head)

```

Output

2 <-> 3 <-> 4 <-> None

2. Deletion at a given position:

To delete a node at a given position in a doubly linked list in Python, you need to follow these steps:

- Check if the list is empty (head is None). If it is empty, there is nothing to delete.
- If the position is less than 0, print an error message as it's an invalid position.

• If the position is 0, delete the node at the beginning of the list.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- Update the "**next**" pointer of the previous node to skip the node to be deleted.
- Update the "**previous**" pointer of the next node to point to the previous node of the node to be deleted.
- Optionally, free the memory allocated to the deleted node.

Below is the implementation of the above idea:

Python

```
1 # Python Program for Deletion of a given node
2 class Node:
3     def __init__(self, data):
4         # Initialize a new node with data, previous, and next
          pointers
5         self.data = data
6         self.next = None
7         self.prev = None
8
9
10 def delete_at_position(head, position):
11     # Delete the node at a given position from the doubly link
12     if head is None:
13         print("Doubly linked list is empty")
14         return None
15
16     if position < 0:
17         print("Invalid position")
18         return head
19
20     if position == 0:
21         if head.next:
22             head.next.prev = None
23         return head.next
24
25     current = head
26     count = 0
27     while current and count < position:
28         current = current.next
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

```
31     if current is None:
32         print("Position out of range")
33         return head
34
35     if current.next:
36         current.next.prev = current.prev
37     if current.prev:
38         current.prev.next = current.next
39
40     del current
41     return head
42
43
44 def traverse(head):
45     # Traverse the doubly linked list and print its elements
46     current = head
47     while current:
48         # Print current node's data
49         print(current.data, end=" <-> ")
50         # Move to the next node
51         current = current.next
52     print("None")
53
54
55 def insert_at_beginning(head, data):
56     # Insert a new node at the beginning of the doubly linked
57     new_node = Node(data)
58     new_node.next = head
59     if head:
60         head.prev = new_node
61     return new_node
62
63
64 # Driver Code
65 head = None
66 head = insert_at_beginning(head, 4)
67 head = insert_at_beginning(head, 3)
68 head = insert_at_beginning(head, 2)
69 head = insert_at_beginning(head, 1)
70
71 # Delete the node at position 2 (node with data 3):
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

```
74 # Traverse and print the nodes after deletion:
75 traverse(head)
```

Output

1 <-> 2 <-> 4 <-> None

3. Deletion at the end:

To delete a node at the end of a doubly linked list in Python, you need to follow these steps:

- Check if the list is empty (**head is None**). If it is empty, there is nothing to delete.
- If the list has only one node, set the head to None to delete the node.
- Traverse the list to find the last node.
- Set the "**next**" pointer of the second-to-last node to None.
- Optionally, free the memory allocated to the deleted node.

Below is the implementation of the above idea:

Python

```
1 # Python Program Deletion at the end
2 class Node:
3     def __init__(self, data):
4         # Initialize a new node with data, previous, and next
          pointers
5         self.data = data
6         self.next = None
7         self.prev = None
8
9 def delete_at_end(head):
10     # Delete the last node from the end of the doubly linked l
11     if head is None:
12         print("Doubly linked list is empty")
```

```
16         return None
17
18     current = head
19     while current.next.next:
20         current = current.next
21
22     del_node = current.next
23     current.next = None
24     del del_node
25     return head
26
27 def traverse(head):
28     # Traverse the doubly linked list and print its elements
29     current = head
30     while current:
31         # Print current node's data
32         print(current.data, end=" <-> ")
33         # Move to the next node
34         current = current.next
35     print("None")
36
37 def insert_at_beginning(head, data):
38     # Insert a new node at the beginning of the doubly linked
39     new_node = Node(data)
40     new_node.next = head
41     if head:
42         head.prev = new_node
43     return new_node
44
45 # Driver Code
46 head = None
47 head = insert_at_beginning(head, 4)
48 head = insert_at_beginning(head, 3)
49 head = insert_at_beginning(head, 2)
50 head = insert_at_beginning(head, 1)
51
52 # Delete the last node (node with data 4) from the end:
53 head = delete_at_end(head)
54
55 # Traverse and print the nodes after deletion:
56 traverse(head)
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

Output

1 <-> 2 <-> 3 <-> None

Related Practice Problem on Doubly Linked List:

Article	Link
Design a data structure for LRU Cache	Link
Reverse a Doubly Linked List	Link
Delete a node in a Doubly Linked List	Link
Insertion in a Doubly Linked List	Link
Insert value in sorted way in a sorted doubly linked list	Link
Find pairs with given sum in doubly linked list	Link
Merge Sort for Doubly Linked List	Link
QuickSort on Doubly Linked List	Link
XOR Linked List – A Memory Efficient Doubly Linked List Set 2	Link

Join [GfG 160](#), a 160-day journey of coding challenges aimed at sharpening your skills. Each day, solve a handpicked problem, dive into detailed solutions through articles and videos, and enhance your preparation for any interview—all for free! Plus, win exciting GfG goodies along the way! - [Explore Now](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

[Comment](#)[More info](#)[Next Article](#)[Circular Linked List in Python](#)

Similar Reads

Doubly Linked List in Python

Doubly Linked List is a type of linked list in which each node contains a data element and two links pointing to the next and previous node in the sequence. This allows for more efficient operations such as traversals,...

13 min read

Circular Linked List in Python

A Circular Linked List is a variation of the standard linked list. In a standard linked list, the last element points to null, indicating the end of the list. However, in a circular linked list, the last element points back to the first...

13 min read

Doubly Linked List in C

A doubly linked list is a type of linked list in which each node contains 3 parts, a data part and two addresses, one points to the previous node and one for the next node. It differs from the singly linked list as it has an...

14 min read

Doubly Linked List in C++

A Doubly Linked List (DLL) is a two-way list in which each node has two pointers, the next and previous that have reference to both the next node and previous node respectively. Unlike a singly linked list where each...

13 min read

How to copy linked list in Python?

Linked Lists: Linked Lists are a type of 'Abstract Data Types' in Python. These data structures are linear in nature. Linked Lists are different from Lists as Linked Lists are stored in non-contiguous (non-continuous)...

13 min read

Python Linked List

In this article, we will learn about the implementation of a linked list in Python. To implement the linked list in Python, we will use classes in Python. Now, we know that a linked list consists of nodes and nodes have tw...

13 min read

Traversal in Doubly Linked List

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

15+ min read

Python | Stack using Doubly Linked List

A stack is a collection of objects that are inserted and removed using Last in First out Principle(LIFO). User can insert elements into the stack, and can only access or remove the recently inserted object on top of the stack....

4 min read

Python | Queue using Doubly Linked List

A Queue is a collection of objects that are inserted and removed using First in First out Principle(FIFO). Insertion is done at the back(Rear) of the Queue and elements are accessed and deleted from first(Front)...

3 min read

Python Library for Linked List

Linked list is a simple data structure in programming, which obviously is used to store data and retrieve it accordingly. To make it easier to imagine, it is more like a dynamic array in which data elements are linked via...

5 min read

Doubly Linked List Tutorial

A doubly linked list is a more complex data structure than a singly linked list, but it offers several advantages. The main advantage of a doubly linked list is that it allows for efficient traversal of the list in both directions....

15+ min read

Singly Linked List in Python

A Singly Linked List is a type of data structure that is made up of nodes that are created using self-referential structures. Each node contains a data element and a reference (link) to the next node in the sequence. This...

10 min read

Doubly Linked List meaning in DSA

A doubly linked list is a special type of linked list in which each node contains a pointer to the previous node as well as the next node in the structure. Characteristics of the Doubly Linked List: The characteristics of a...

3 min read

Memory efficient doubly linked list

We need to implement a doubly linked list with the use of a single pointer in each node. For that we are given a stream of data of size n for the linked list, your task is to make the function insert() and getList(). The insert...

9 min read

Why use a Doubly Linked List?

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

Program to find size of Doubly Linked List

Given a doubly linked list, The task is to find the number of nodes in the given doubly linked list. Example:

Input : 1<->2<->3<->4output : 4 Input : 1<->2output : 2 By Traversing the Doubly linked list – O(n) Time and...

5 min read

Print Doubly Linked list in Reverse Order

Given a doubly-linked list of positive integers. The task is to print the given doubly linked list data in reverse order. Examples: Input: List = 1 <=> 2 <=> 3 <=> 4 <=> 5 Output: 5 4 3 2 1 Input: 10 <=> 20 <=> 30 <=> 40...

7 min read

Deletion in a Doubly Linked List

Deleting a node in a doubly linked list is very similar to deleting a node in a singly linked list. However, there is a little extra work required to maintain the links of both the previous and next nodes. In this article, we will...

15+ min read

Insertion in a Doubly Linked List

Inserting a new node in a doubly linked list is very similar to inserting new node in linked list. There is a little extra work required to maintain the link of the previous node. In this article, we will learn about different way...

15+ min read

Pretty print Linked List in Python

Creating custom data types can be tricky, especially when you want to use it like any other data type. Linked List can be thought of as an example of a custom data type. In other languages, if you want to print the linke...

3 min read

Article Tags : [DSA](#) [Linked List](#) [doubly linked list](#) [Python-DSA](#)

Practice Tags : [Linked List](#)



Corporate & Communications Address:-

A-142, 7th Floor, Sovereign Corporate

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305



Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science

Operating Systems
Computer Network
Database Management System

DevOps

Git
Linux
AWS

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

[Software Development](#)[Software Testing](#)[GCP](#)[DevOps Roadmap](#)

System Design

[High Level Design](#)[Low Level Design](#)[UML Diagrams](#)[Interview Guide](#)[Design Patterns](#)[OOAD](#)[System Design Bootcamp](#)[Interview Questions](#)

Interview Preparation

[Competitive Programming](#)[Top DS or Algo for CP](#)[Company-Wise Recruitment Process](#)[Company-Wise Preparation](#)[Aptitude Preparation](#)[Puzzles](#)

School Subjects

[Mathematics](#)[Physics](#)[Chemistry](#)[Biology](#)[Social Science](#)[English Grammar](#)[Commerce](#)[World GK](#)

GeeksforGeeks Videos

[DSA](#)[Python](#)[Java](#)[C++](#)[Web Development](#)[Data Science](#)[CS Subjects](#)

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved