

# DSA Linked Lists Types

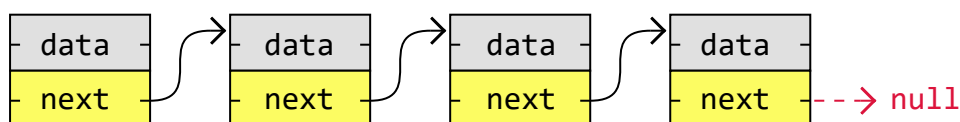
[< Previous](#)[Next >](#)

## Types of Linked Lists

There are three basic forms of linked lists:

1. Singly linked lists
2. Doubly linked lists
3. Circular linked lists

A **singly linked list** is the simplest kind of linked lists. It takes up less space in memory because each node has only one address to the next node, like in the image below.



A **doubly linked list** has nodes with addresses to both the previous and the next node, like in the image below, and therefore takes up more memory. But doubly linked lists are good if you want to be able to move both up and down in the list.

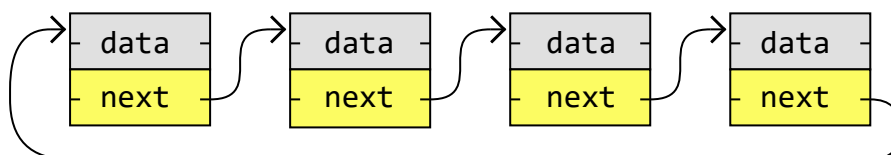


A **circular linked list** is like a singly or doubly linked list with the first node, the "head", and the last node, the "tail", connected.

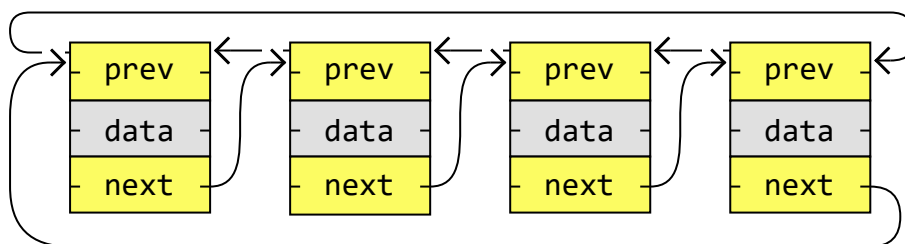
In singly or doubly linked lists, we can find the start and end of a list by just checking if the links are `null`. But for circular linked lists, more complex code is needed to explicitly check for start and end nodes in certain applications.

Circular linked lists are good for lists you need to cycle through continuously.

The image below is an example of a singly circular linked list:



The image below is an example of a doubly circular linked list:



**Note:** What kind of linked list you need depends on the problem you are trying to solve.

## Linked List Implementations

Below are basic implementations of:

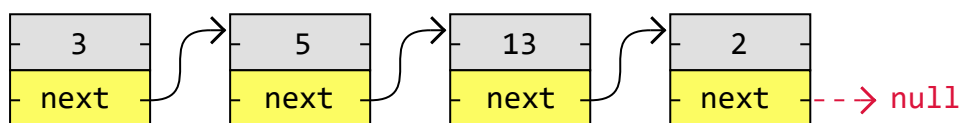
### 1. Singly linked list



The next page will cover different operations that can be done on linked lists.

# 1. Singly Linked List Implementation

Below is an implementation of this singly linked list:



## Example

A basic singly linked list in Python:

(This is the same example as on the bottom of the previous page.)

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

node1 = Node(3)
node2 = Node(5)
node3 = Node(13)
node4 = Node(2)

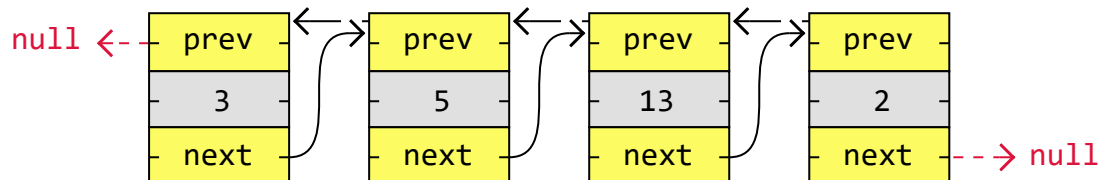
node1.next = node2
node2.next = node3
node3.next = node4

currentNode = node1
while currentNode:
    print(currentNode.data, end=" -> ")
    currentNode = currentNode.next
print("null")
```



## 2. Doubly Linked List Implementation

Below is an implementation of this doubly linked list:



### Example

A basic doubly linked list in Python:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
        self.prev = None

node1 = Node(3)
node2 = Node(5)
node3 = Node(13)
node4 = Node(2)

node1.next = node2

node2.prev = node1
node2.next = node3

node3.prev = node2
node3.next = node4

node4.prev = node3

print("\nTraversing forward:")
currentNode = node1
```



```

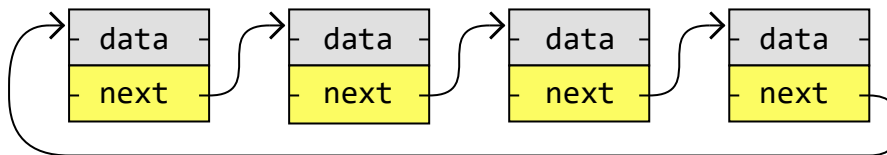
print("\nTraversing backward:")
currentNode = node4
while currentNode:
    print(currentNode.data, end=" -> ")
    currentNode = currentNode.prev
print("null")

```

Try it Yourself »

## 3. Circular Singly Linked List Implementation

Below is an implementation of this circular singly linked list:



### Example

A basic circular singly linked list in Python:

```

1  class Node:
2      def __init__(self, data):
3          self.data = data
4          self.next = None
5
6  node1 = Node(3)
7  node2 = Node(5)
8  node3 = Node(13)
9  node4 = Node(2)
10
11  node1.next = node2

```



```

16 | currentNode = node1

18 | print(currentNode.data, end=" -> ")
19 | currentNode = currentNode.next
20 |
21 | while currentNode != startNode:
22 |     print(currentNode.data, end=" -> ")
23 |     currentNode = currentNode.next
24 |
25 | print("...")

```

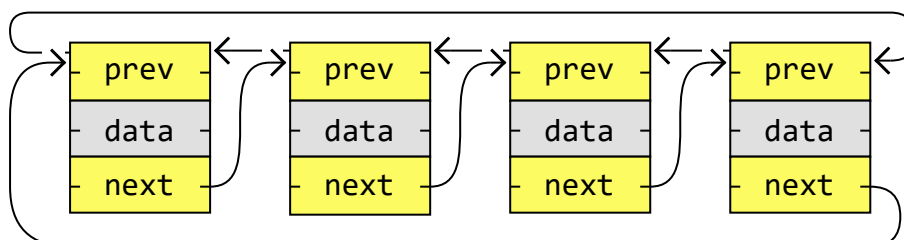
Try it Yourself »

**Line 14:** This makes the singly list circular.

**Line 17:** This is how the program knows when to stop so that it only goes through the list one time.

## 4. Circular Doubly Linked List Implementation

Below is an implementation of this circular doubly linked list:



### Example

A basic circular doubly linked list in Python:





Tutorials ▼

References ▼

Exercises ▼



Sign In

```

4         self.next = None
5         self.prev = None
6
7     node1 = Node(3)
8     node2 = Node(5)
9     node3 = Node(13)
10    node4 = Node(2)
11
12    node1.next = node2
13
14
15    node2.prev = node1
16    node2.next = node3
17
18    node3.prev = node2
19    node3.next = node4
20
21    node4.prev = node3
22
23
24    print("\nTraversing forward:")
25    currentNode = node1
26
27    print(currentNode.data, end=" -> ")
28    currentNode = currentNode.next
29
30    while currentNode != startNode:
31        print(currentNode.data, end=" -> ")
32        currentNode = currentNode.next
33    print("...")
34
35    print("\nTraversing backward:")
36    currentNode = node4
37    startNode = node4
38    print(currentNode.data, end=" -> ")
39    currentNode = currentNode.prev
40
41    while currentNode != startNode:
42        print(currentNode.data, end=" -> ")
43
```





**Lines 13 and 22:** These links makes the doubly linked list circular.

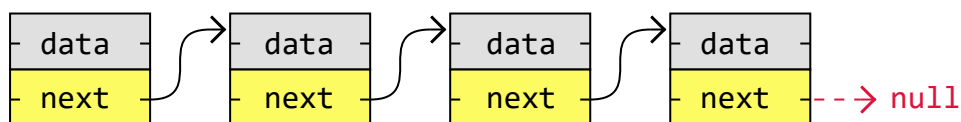
**Lines 26:** This is how the program knows when to stop so that it only goes through the list one time.

## DSA Exercises

### Test Yourself With Exercises

#### Exercise:

Take a look at this singly Linked List:



How can we make this Linked List circular?

The list can be made circular by connecting the next pointer in the last node, to the                      node.

[Submit Answer »](#)



[Tutorials ▼](#)[References ▼](#)[Exercises ▼](#)[Sign In](#)[SS](#)[JAVASCRIPT](#)[SQL](#)[PYTHON](#)[JAVA](#)[PHP](#)[HOW TO](#)[W3.CSS](#)[C](#)[C](#)[< Previous](#)[Sign in to track progress](#)[Next >](#)

**Full access**

All Courses  
All Certificates

**\$499** Save 75%  
~~\$1,995~~



COLOR PICKER





[Tutorials ▼](#)[References ▼](#)[Exercises ▼](#)[Sign In](#)[SS](#)[JAVASCRIPT](#)[SQL](#)[PYTHON](#)[JAVA](#)[PHP](#)[HOW TO](#)[W3.CSS](#)[C](#)[C](#)[GET CERTIFIED](#)[FOR TEACHERS](#)[FOR BUSINESS](#)[CONTACT US](#)

## Top Tutorials

- [HTML Tutorial](#)
- [CSS Tutorial](#)
- [JavaScript Tutorial](#)
- [How To Tutorial](#)
- [SQL Tutorial](#)
- [Python Tutorial](#)
- [W3.CSS Tutorial](#)
- [Bootstrap Tutorial](#)
- [PHP Tutorial](#)
- [Java Tutorial](#)
- [C++ Tutorial](#)
- [jQuery Tutorial](#)

## Top References

- [HTML Reference](#)
- [CSS Reference](#)
- [JavaScript Reference](#)
- [SQL Reference](#)
- [Python Reference](#)
- [W3.CSS Reference](#)
- [Bootstrap Reference](#)
- [PHP Reference](#)
- [HTML Colors](#)
- [Java Reference](#)
- [AngularJS Reference](#)
- [jQuery Reference](#)

## Top Examples

- [HTML Examples](#)
- [CSS Examples](#)
- [JavaScript Examples](#)
- [How To Examples](#)
- [SQL Examples](#)
- [Python Examples](#)
- [W3.CSS Examples](#)
- [Bootstrap Examples](#)
- [PHP Examples](#)
- [Java Examples](#)
- [XML Examples](#)
- [jQuery Examples](#)

## Get Certified

- [HTML Certificate](#)
- [CSS Certificate](#)
- [JavaScript Certificate](#)
- [Front End Certificate](#)
- [SQL Certificate](#)
- [Python Certificate](#)
- [PHP Certificate](#)
- [jQuery Certificate](#)
- [Java Certificate](#)
- [C++ Certificate](#)
- [C# Certificate](#)
- [XML Certificate](#)



[Tutorials ▼](#)[References ▼](#)[Exercises ▼](#)[Sign In](#)[SS](#)[JAVASCRIPT](#)[SQL](#)[PYTHON](#)[JAVA](#)[PHP](#)[HOW TO](#)[W3.CSS](#)[C](#)[C](#)[FORUM](#) [ABOUT](#) [ACADEMY](#)

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning.

Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness

of all content. While using W3Schools, you agree to have read and accepted our [terms of use](#), [cookies](#) and [privacy policy](#).

[Copyright 1999-2026](#) by Refsnes Data. All Rights Reserved. [W3Schools](#) is Powered by [W3.CSS](#).