Singly Linked list

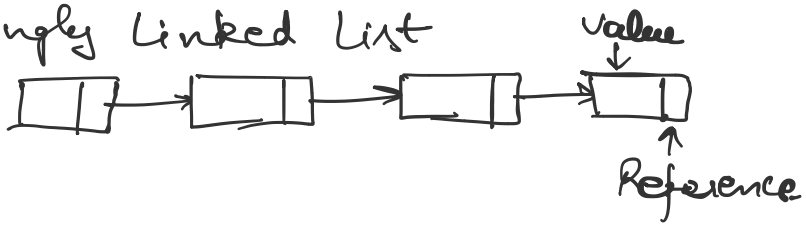
Tuesday, 2 April 2024 8:14 PM

- First component is head.
- Node consists of 2 parts, values & reference to the next node
- Tail's reference should be null but its not necessary.
-

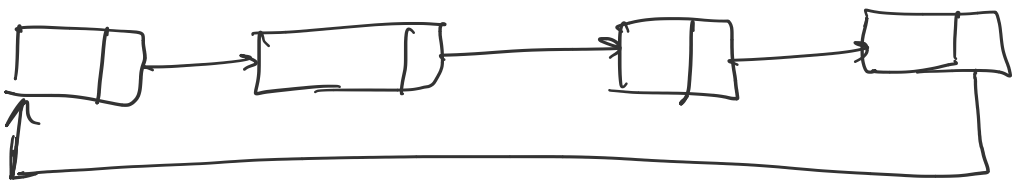
Python List	Linked List
1. Has index	Has references
2. Element can't concurrently	No need to have concurrency.

Types of Linked List.

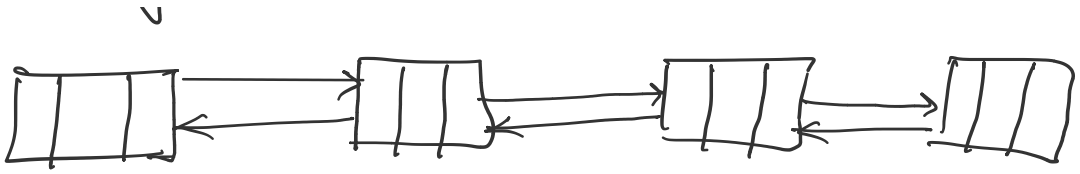
1. Singly Linked List



2. Circular singly linked list

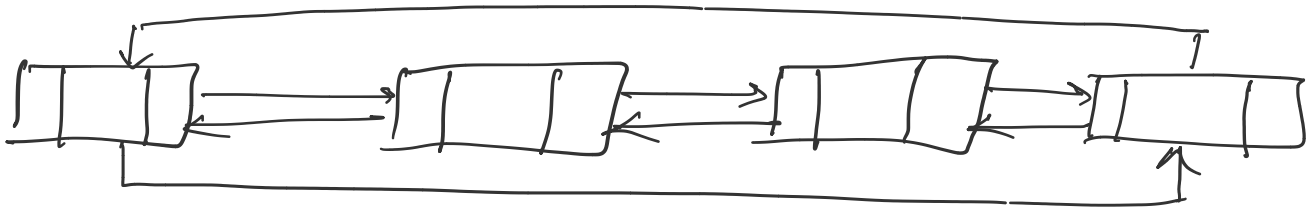


3. Doubly liked list

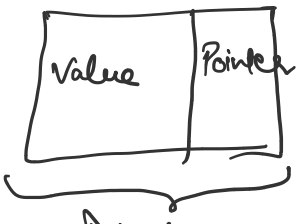


Enables traversal in both directions.

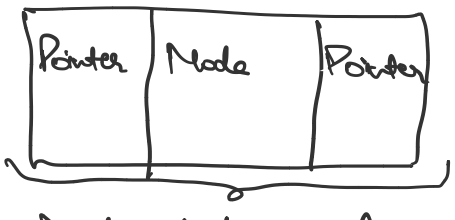
4. Circular double linked list



- All elements are always stored randomly in memory. Hence to find any element traversal of all previous elements is required.
- Python by default does not have a linked list datatype by default. Hence it needs to be implemented via classes in Python.
- A node is a combination of an object that contains both value & the pointer to the next node (in case of doubly linked list, it will also include pointer to the previous node).



Node.



Node of doubly linked list

How to define a node:

How to define a node:

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

new_node = Node(10)
```

How to define LinkedList in Python:

```
class LinkedList:
    def __init__(self, value):
        new_node = Node(value)
        self.head = new_node
        self.tail = new_node
        self.length = 1   # This attribute is not
mandatory, but makes traversal easier through the LL
    def insert(self, value):
        new_node = Node(value)
        if self.head == None:
            self.head = new_node
            self.tail = new_node
        else:
            self.tail.next = new_node
            self.tail = new_node
        self.length = 1
```

How to print LinkedList:

```
def __str__(self):
        temp_node = self.head
        result = ""
        while temp_node != None:
            result += str(temp_node.value)
            if temp_node.next is not None:
                result += " -> "
            temp_node = temp_node.next
        return result
```

How to insert a node at the end:

```
def insert(self, value):
    new_node = Node(value)
    if self.head == None:
        self.head = new_node
        self.tail = new_node
    else:
        self.tail.next = new_node
        self.tail = new_node
    self.length = 1
```

How to insert a node at the start/head:

```
def prepend(self, value):

    new_node = Node(value)
    if self.head is not None:
        new_node.next = self.head
        self.head = new_node
    else:
        self.head = new_node
        self.tail = new_node
    self.length += 1
```