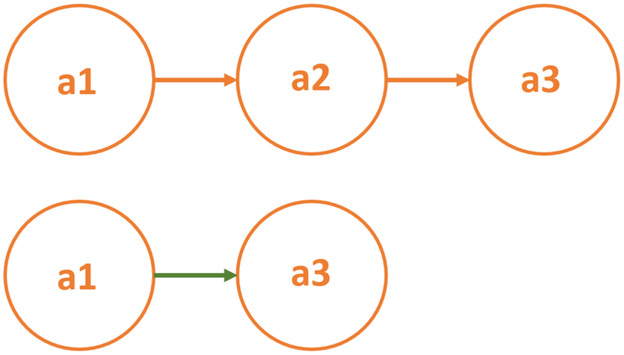
Linked Lists

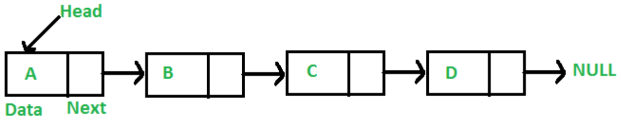
Removing a node from the middle of a linked list

When removing a node from the middle of a linked list, it is necessary to adjust the link on the previous node so that it points to the following node. In the given illustration, the node `a1` must point to the node `a3` if the node `a2` is removed from the linked list.



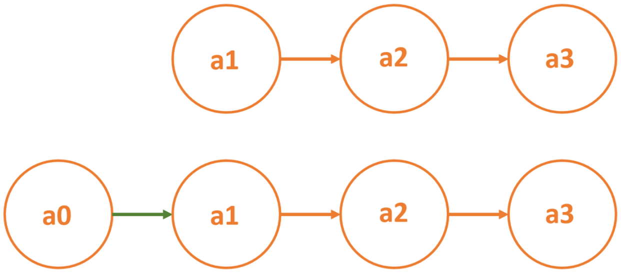
Linked List data structure

A **linked list** is a linear data structure where elements are not stored at contiguous location. Instead the elements are linked using pointers.
In a linked list data is stored in nodes and each node is linked to the next and, optionally, to the previous. Each node in a list consists of the following parts:
1) data 2) A pointer (Or reference) to the next node 3) Optionally, a pointer to the previous node



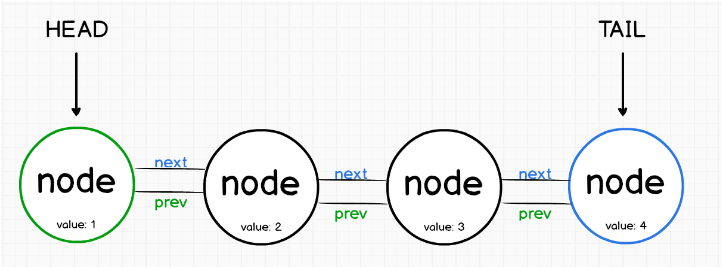
Adding a new head node in a linked list

When adding a new node to the start of a linked list, it is necessary to maintain the list by giving the new head node a link to the current head node. For instance, to add a new node `a0` to the begining of the linked list, `a0` should point to `a1`.



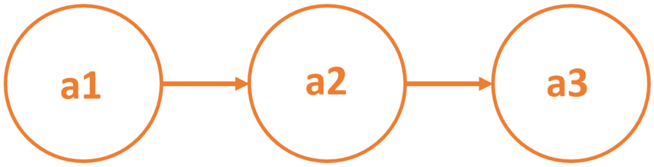
The Head Node in Linked Lists

The first node in a linked list is called the head node. If the linked list is empty, then the value of the head node is **NULL**.



Implementing a linked list

A linked list exposes the ability to traverse the list from one node to another node. The starting node is considered the head node from where the list can be traversed.



Linked List Data Structure

A `linked list` is a data structure that consists of a list of `nodes`. Each node contains data and a link to the next node. As shown below, you can implement a `LinkedList` class in Python, utilizing a Python implementation of the `Node` class.

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

  def get_head_node(self):
    return self.head_node

  def insert_beginning(self, new_value):
    new_node = Node(new_value)
    new_node.set_next_node(self.head_node)
  self.head_node = new_node

 def stringify_list(self):
  string_list = ""
  current_node = self.get_head_node()
  while current_node:
   if current_node.get_value() != None:
    string_list +=
str(current_node.get_value()) + "\n"
   current_node =
current_node.get_next_node()
  return string_list

 def remove_node(self, value_to_remove):
  current_node = self.get_head_node()
  if current_node.get_value() ==
value_to_remove:
   self.head_node =
current_node.get_next_node()
  else:
   while current_node:
    next_node =
current_node.get_next_node()
    if next_node.get_value() ==
value_to_remove:

current_node.set_next_node(next_node.get_n
ext_node())
     current_node = None
    else:
     current_node = next_node
```

Print Share ▾