Python - Advanced Linked list

We have already seen Linked List in earlier chapter in which it is possible only to travel forward. In this chapter we see another type of linked list in which it is possible to travel both forward and backward. Such a linked list is called Doubly Linked List. Following is the features of doubly linked list.

- Doubly Linked List contains a link element called first and last.
- Each link carries a data field(s) and two link fields called next and prev.
- Each link is linked with its next link using its next link.
- Each link is linked with its previous link using its previous link.
- The last link carries a link as null to mark the end of the list.

Creating Doubly linked list

We create a Doubly Linked list by using the Node class. Now we use the same approach as used in the Singly Linked List but the head and next pointers will be used for proper assignation to create two links in each of the nodes in addition to the data present in the node.

Example

```
class Node:
   def __init__(self, data):
      self.data = data
```

```
        self.next = None
        self.prev = None

class doubly_linked_list:
   def __init__(self):
      self.head = None

# Adding data elements
   def push(self, NewVal):
      NewNode = Node(NewVal)
      NewNode.next = self.head
      if self.head is not None:
         self.head.prev = NewNode
      self.head = NewNode

# Print the Doubly Linked list
   def listprint(self, node):
      while (node is not None):
         print(node.data),
         last = node
         node = node.next

dllist = doubly_linked_list()
dllist.push(12)
dllist.push(8)
dllist.push(62)
dllist.listprint(dllist.head)
```

Output

When the above code is executed, it produces the following result −

```
62 8 12
```

Inserting into Doubly Linked List

Here, we are going to see how to insert a node to the Doubly Link List using the following program. The program uses a method named insert which inserts the new node at the third position from the head of the doubly linked list.

Example

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

# Create the doubly linked list
class doubly_linked_list:
   def __init__(self):
      self.head = None

# Define the push method to add elements
   def push(self, NewVal):
      NewNode = Node(NewVal)
      NewNode.next = self.head
      if self.head is not None:
         self.head.prev = NewNode
      self.head = NewNode

# Define the insert method to insert the element
```

```
      def insert(self, prev_node, NewVal):
         if prev_node is None:
            return
         NewNode = Node(NewVal)
         NewNode.next = prev_node.next
         prev_node.next = NewNode
         NewNode.prev = prev_node
         if NewNode.next is not None:
            NewNode.next.prev = NewNode

   # Define the method to print the linked list
      def listprint(self, node):
         while (node is not None):
            print(node.data),
            last = node
            node = node.next

   dllist = doubly_linked_list()
   dllist.push(12)
   dllist.push(8)
   dllist.push(62)
   dllist.insert(dllist.head.next, 13)
   dllist.listprint(dllist.head)
```

Output

When the above code is executed, it produces the following result −

```
62  8  13  12
```

Appending to a Doubly linked list

Appending to a doubly linked list will add the element at the end.

Example

```
# Create the node class
class Node:
   def __init__(self, data):
      self.data = data
      self.next = None
      self.prev = None
# Create the doubly linked list class
class doubly_linked_list:
   def __init__(self):
      self.head = None

# Define the push method to add elements at the begining
   def push(self, NewVal):
      NewNode = Node(NewVal)
      NewNode.next = self.head
      if self.head is not None:
         self.head.prev = NewNode
      self.head = NewNode

# Define the append method to add elements at the end
   def append(self, NewVal):
      NewNode = Node(NewVal)
      NewNode.next = None
      if self.head is None:
         NewNode.prev = None
         self.head = NewNode
         return
      last = self.head
```

```
        while (last.next is not None):
            last = last.next
        last.next = NewNode
        NewNode.prev = last
        return

    # Define the method to print
    def listprint(self, node):
        while (node is not None):
            print(node.data),
            last = node
            node = node.next

dllist = doubly_linked_list()
dllist.push(12)
dllist.append(9)
dllist.push(8)
dllist.push(62)
dllist.append(45)
dllist.listprint(dllist.head)
```

Output

When the above code is executed, it produces the following result −

```
62 8 12 9 45
```

Please note the position of the elements 9 and 45 for the append operation.

Print Page