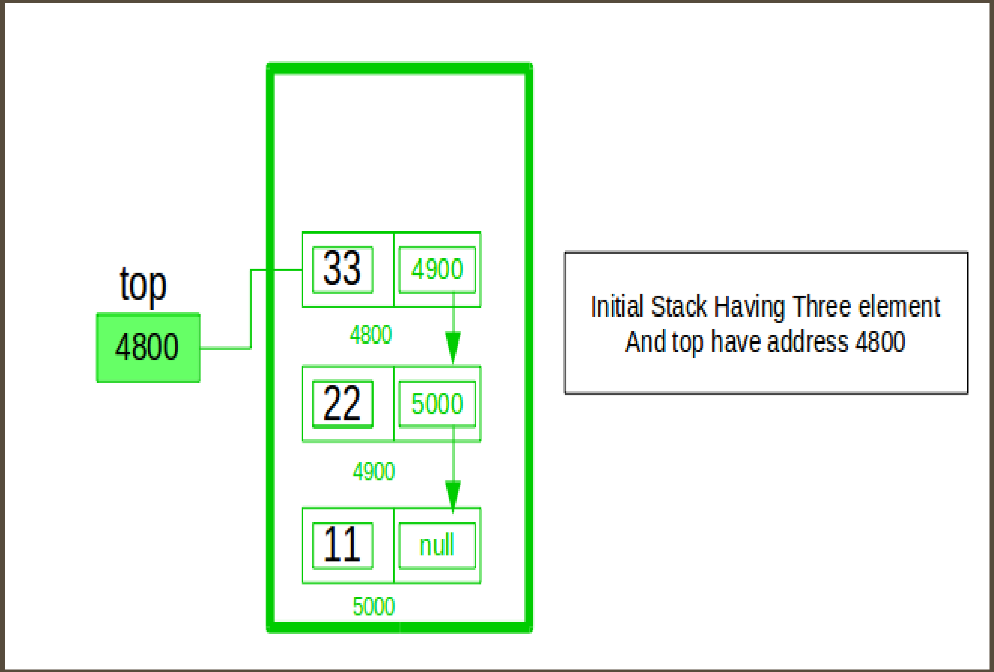
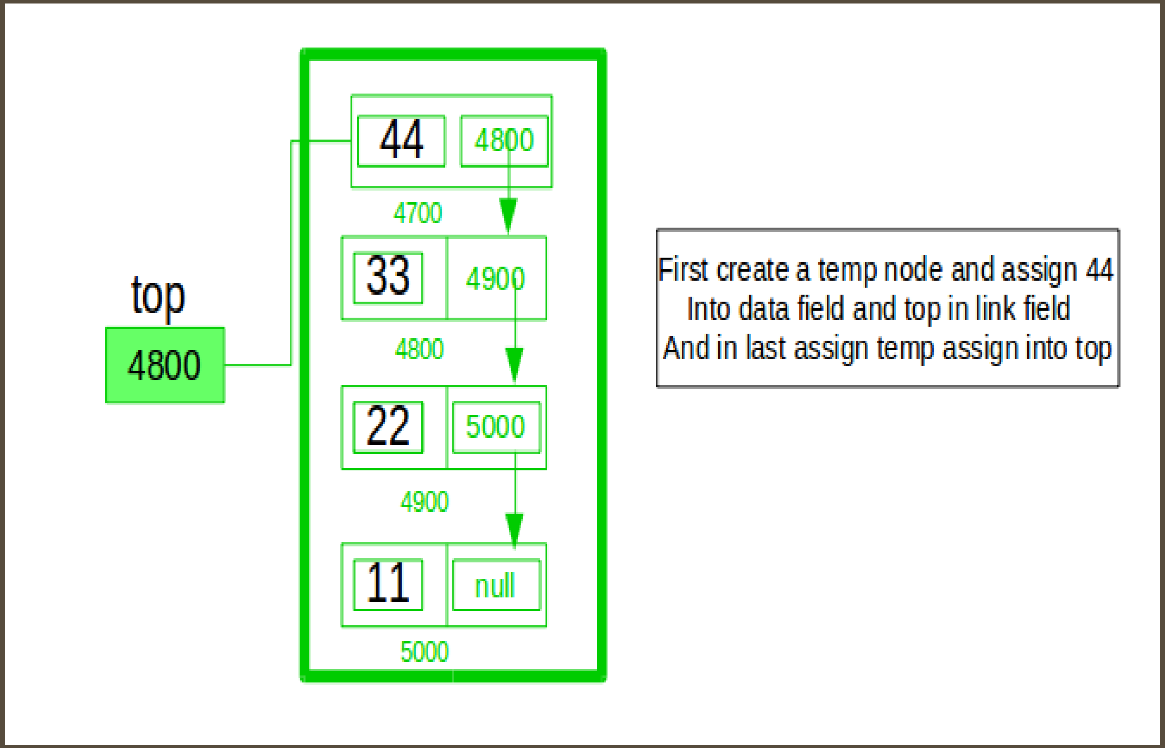
ADDITIONAL NOTES

STACK LINKED LISTS

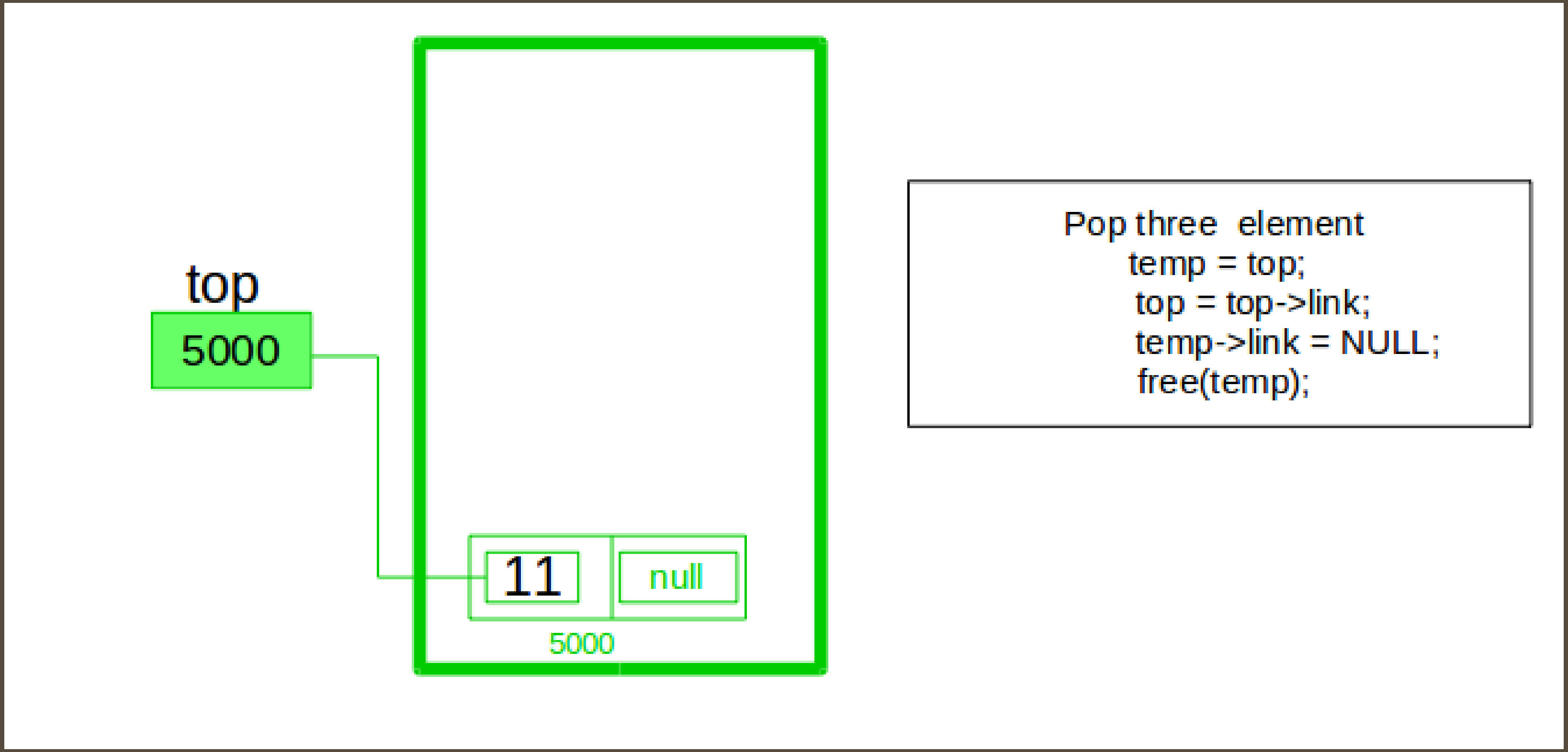


Initial Stack Having Three element
And top have address 4800



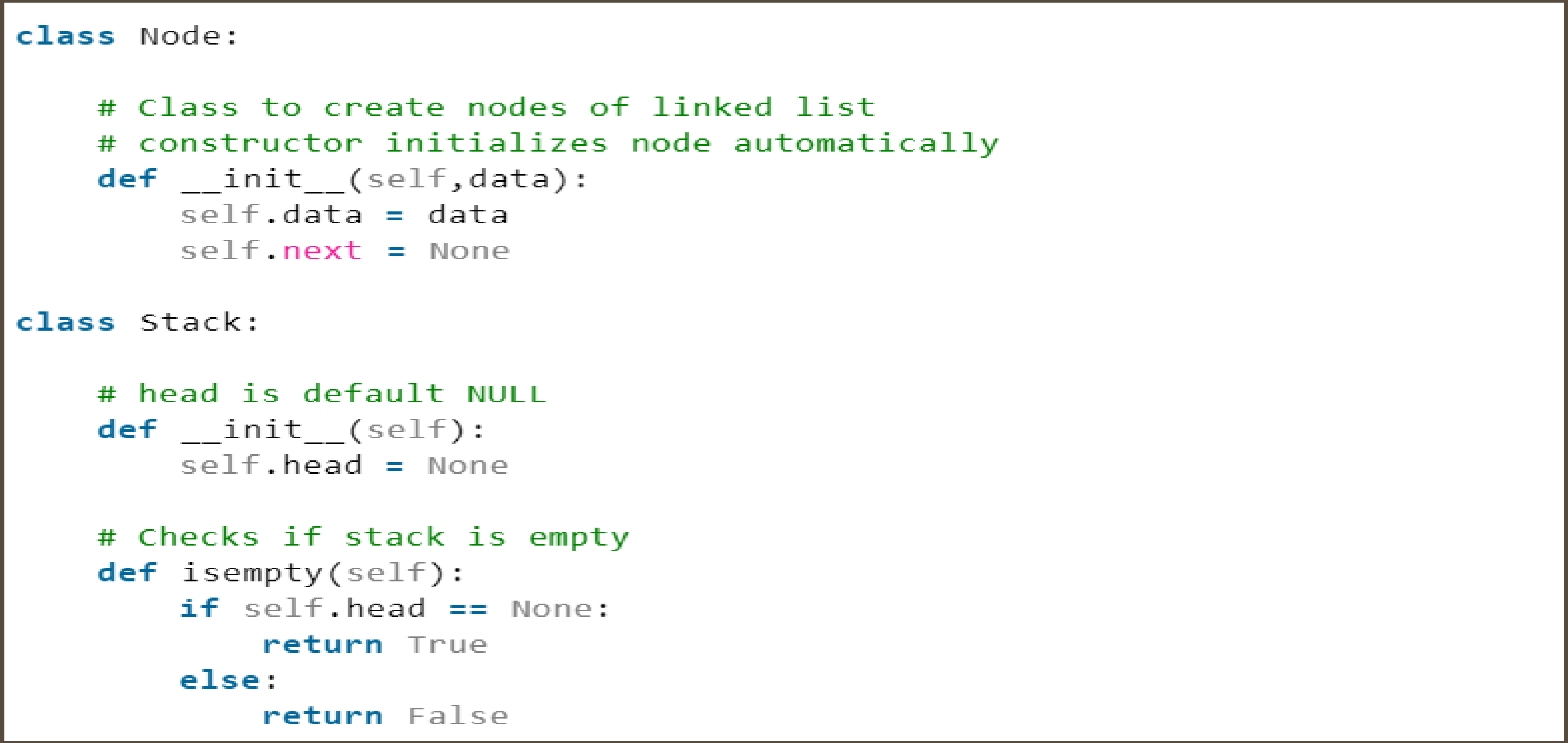
First create a temp node and assign 44
Into data field and top in link field
And in last assign temp assign into top

STACK LINKED LISTS



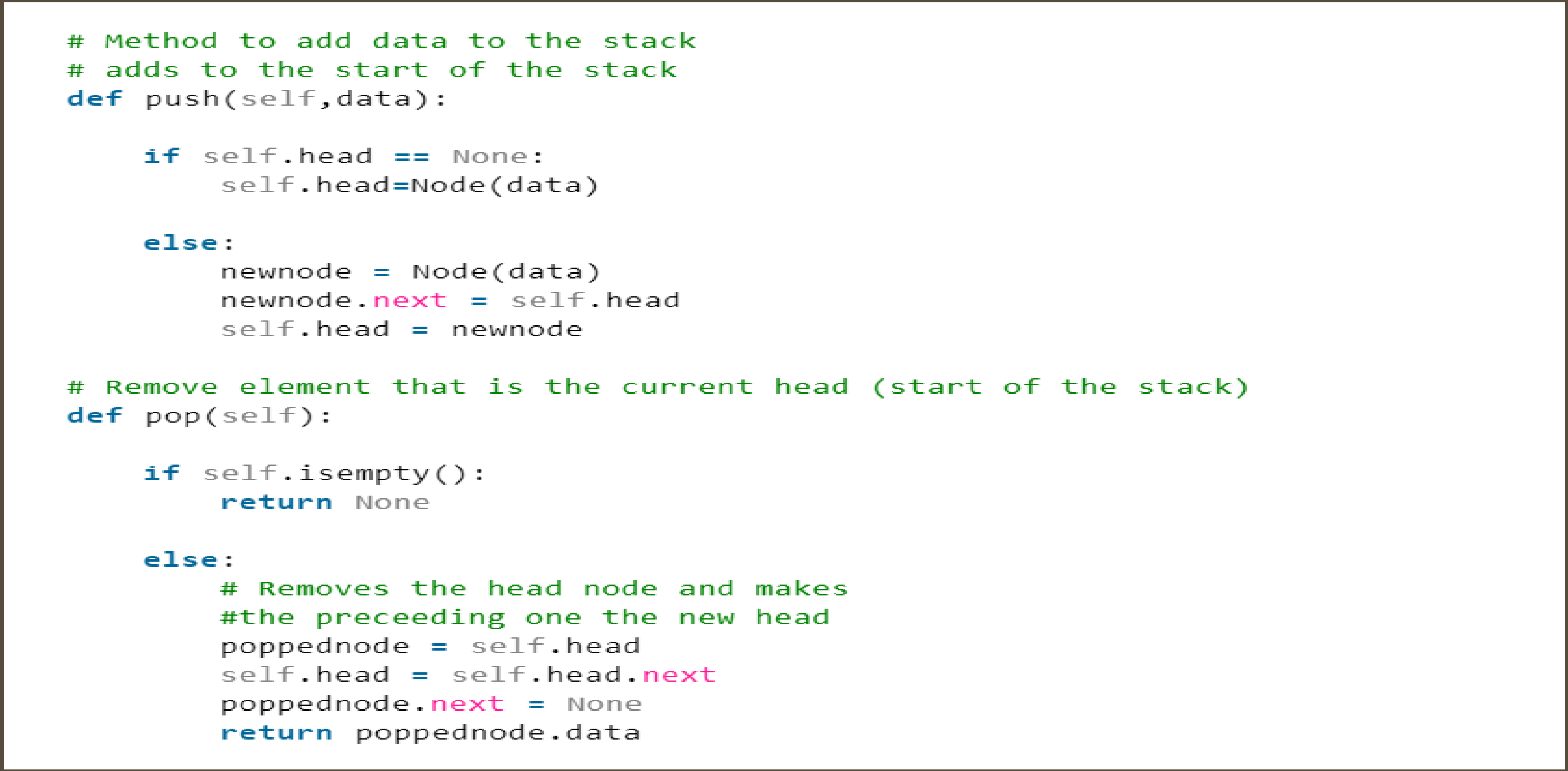
```
Pop three  element
      temp = top;
       top = top->link;
       temp->link = NULL;
        free(temp);
```

EXAMPLE



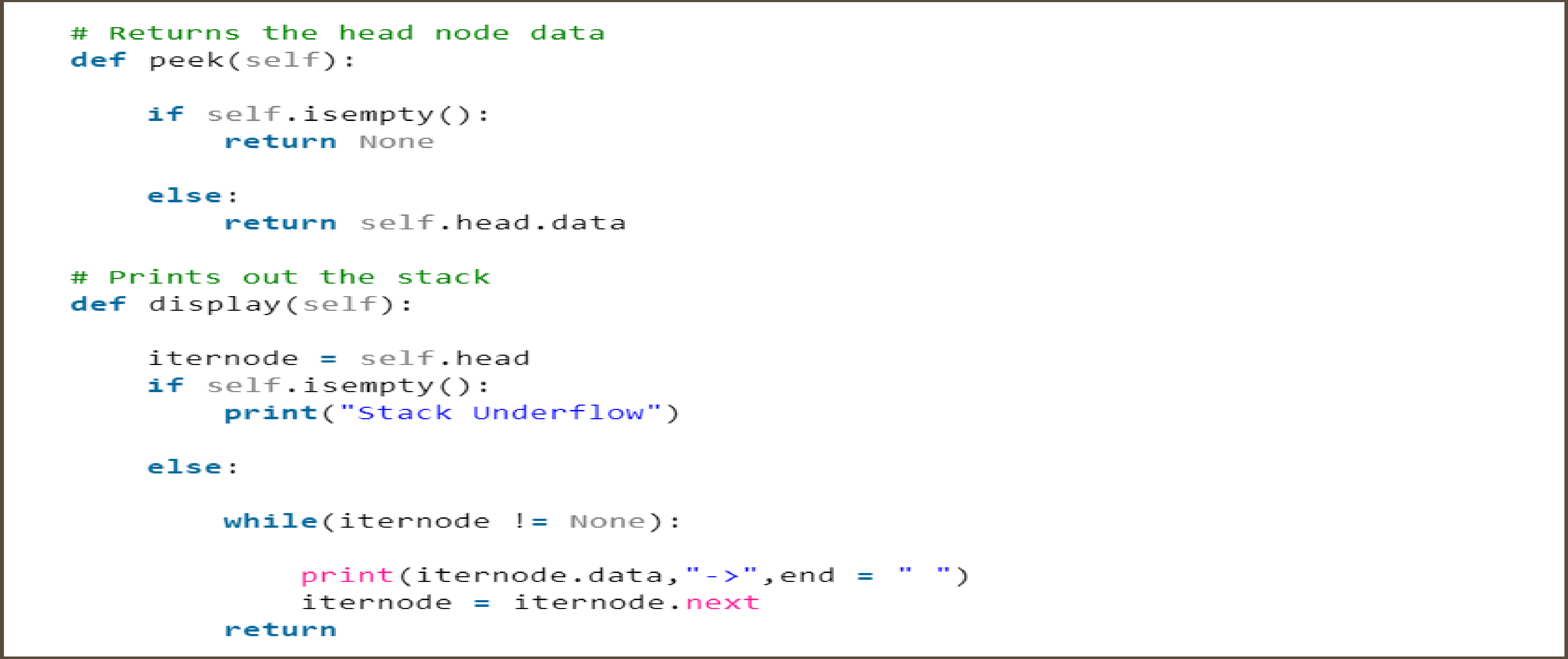
```python
class Node:

    # Class to create nodes of linked list
    # constructor initializes node automatically
    def __init__(self,data):
        self.data = data
        self.next = None

class Stack:

    # head is default NULL
    def __init__(self):
        self.head = None

    # Checks if stack is empty
    def isempty(self):
        if self.head == None:
            return True
        else:
            return False
```

EXAMPLE



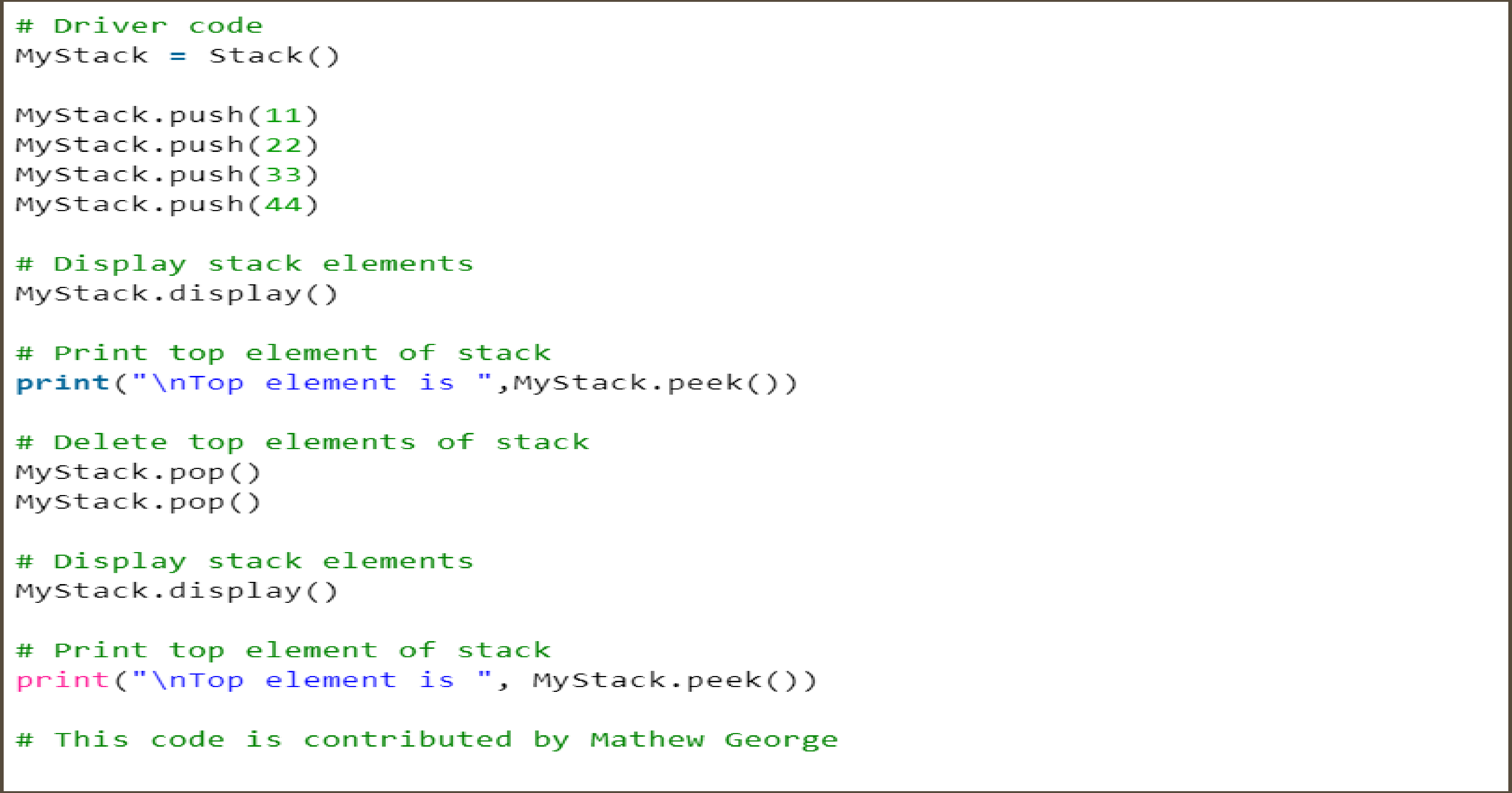
```python
# Method to add data to the stack
# adds to the start of the stack
def push(self,data):

    if self.head == None:
        self.head=Node(data)

    else:
        newnode = Node(data)
        newnode.next = self.head
        self.head = newnode

# Remove element that is the current head (start of the stack)
def pop(self):

    if self.isempty():
        return None

    else:
        # Removes the head node and makes
        #the preceeding one the new head
        poppednode = self.head
        self.head = self.head.next
        poppednode.next = None
        return poppednode.data
```

EXAMPLE



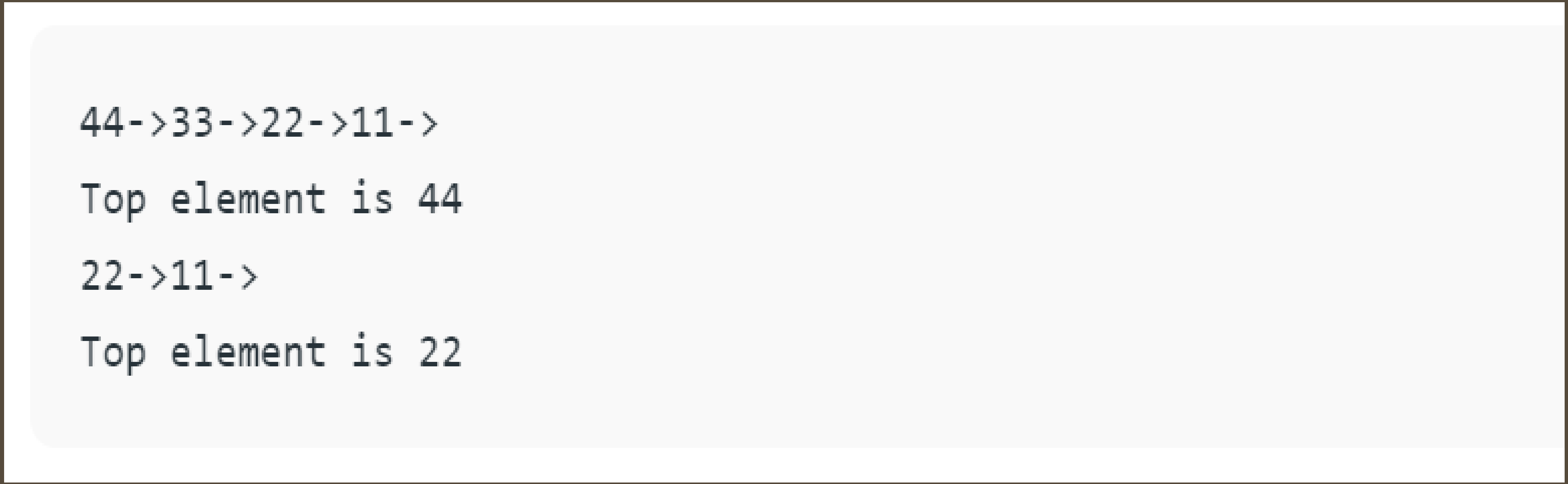
```python
# Returns the head node data
def peek(self):

    if self.isempty():
        return None

    else:
        return self.head.data

# Prints out the stack
def display(self):

    iternode = self.head
    if self.isempty():
        print("Stack Underflow")

    else:

        while(iternode != None):

            print(iternode.data,"->",end = " ")
            iternode = iternode.next
        return
```

EXAMPLE



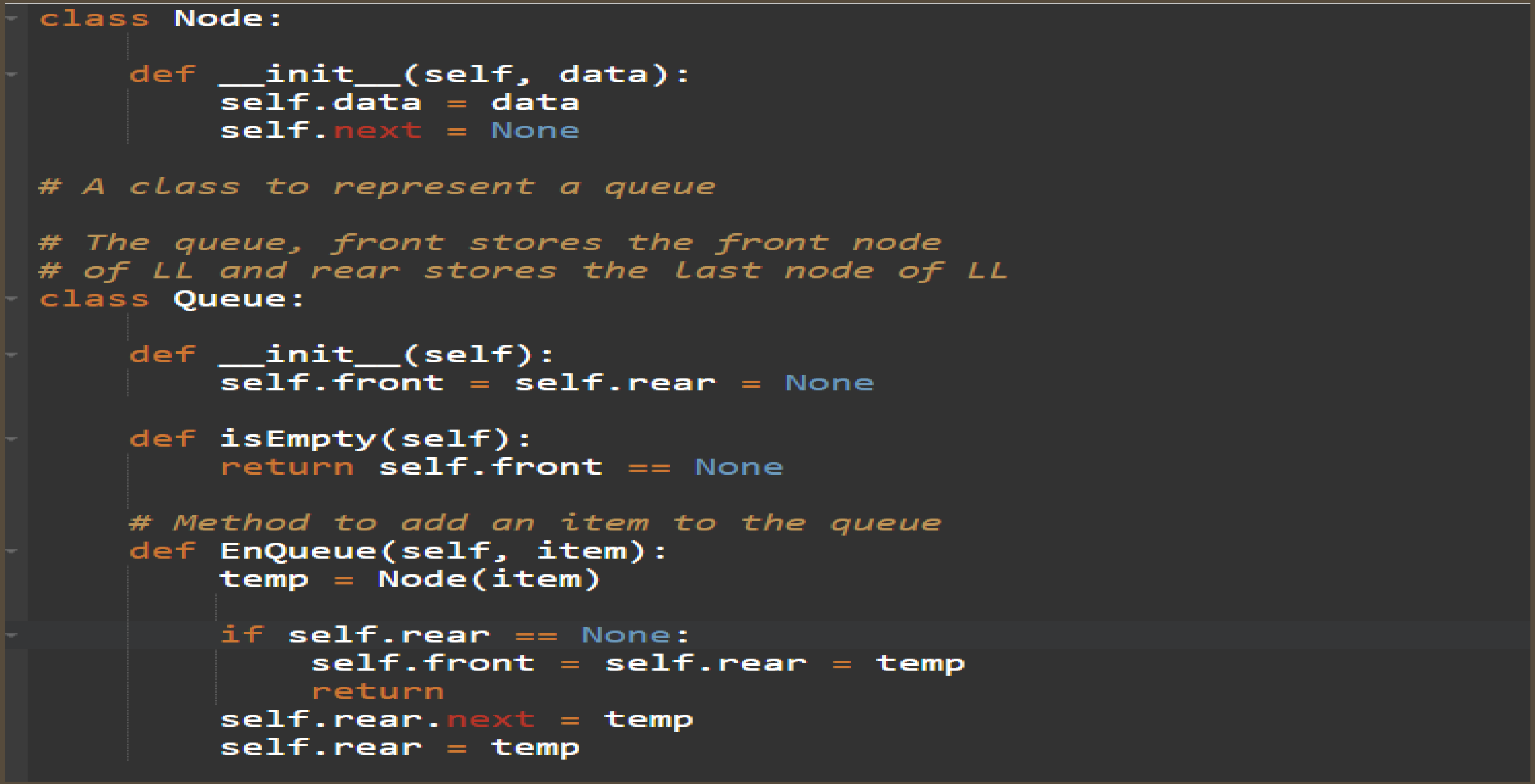
```
# Driver code
MyStack = Stack()

MyStack.push(11)
MyStack.push(22)
MyStack.push(33)
MyStack.push(44)

# Display stack elements
MyStack.display()

# Print top element of stack
print("\nTop element is ",MyStack.peek())

# Delete top elements of stack
MyStack.pop()
MyStack.pop()

# Display stack elements
MyStack.display()

# Print top element of stack
print("\nTop element is ", MyStack.peek())

# This code is contributed by Mathew George
```

EXAMPLE (OUTPUT)



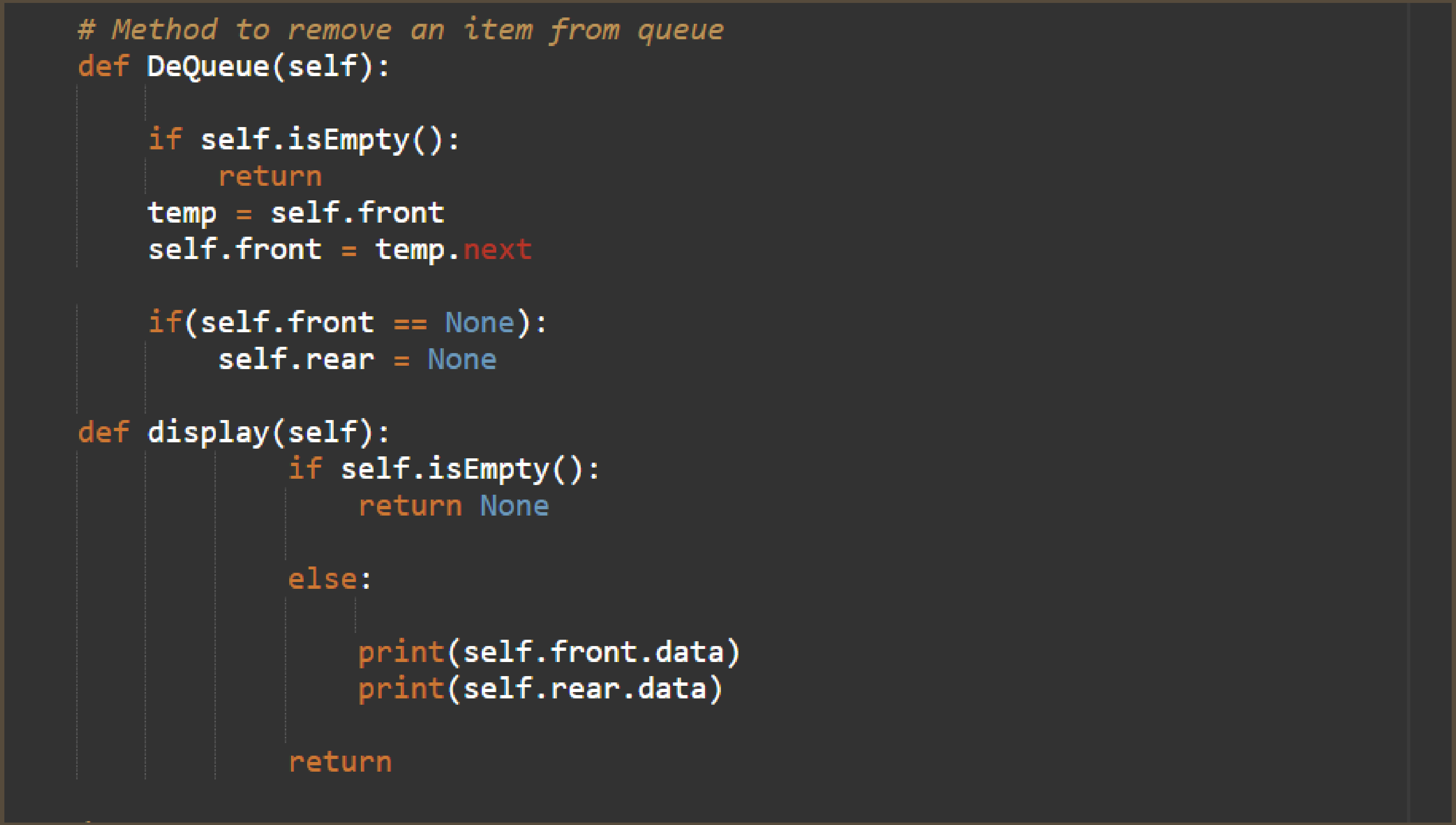
```
44->33->22->11->

Top element is 44

22->11->

Top element is 22
```

QUEUE LINKED LISTS



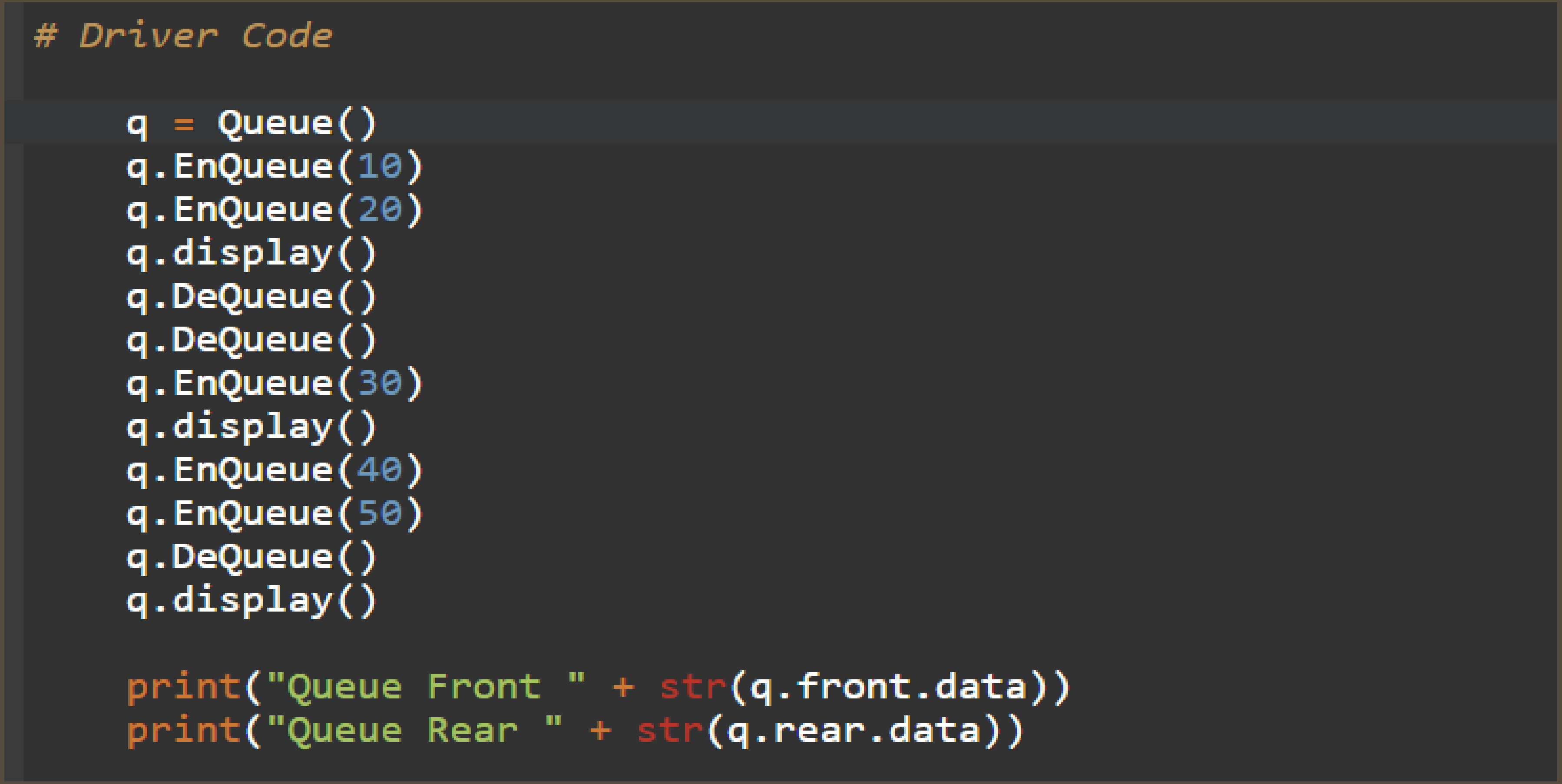
```
class Node:

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

# A class to represent a queue

# The queue, front stores the front node
# of LL and rear stores the last node of LL
class Queue:

    def __init__(self):
        self.front = self.rear = None

    def isEmpty(self):
        return self.front == None

    # Method to add an item to the queue
    def EnQueue(self, item):
        temp = Node(item)

        if self.rear == None:
            self.front = self.rear = temp
            return
        self.rear.next = temp
        self.rear = temp
```

QUEUE LINKED LISTS



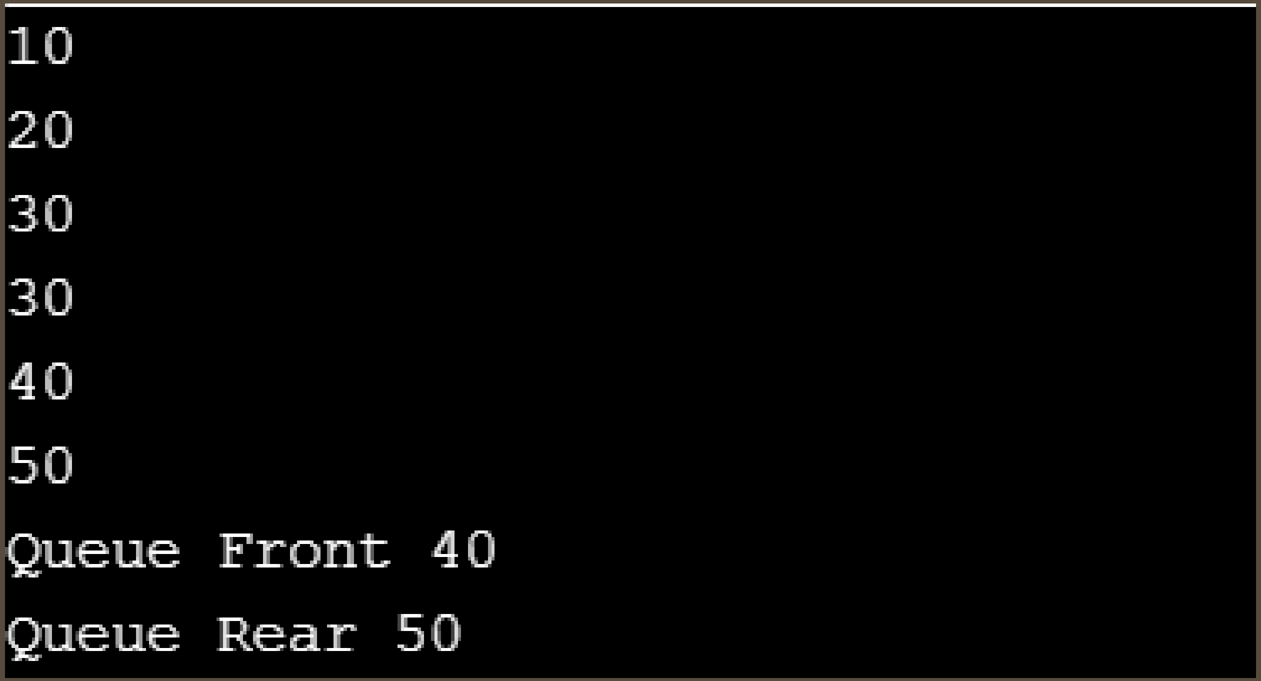
```python
# Method to remove an item from queue
def DeQueue(self):

    if self.isEmpty():
        return
    temp = self.front
    self.front = temp.next

    if(self.front == None):
        self.rear = None

def display(self):
            if self.isEmpty():
                return None

            else:

                print(self.front.data)
                print(self.rear.data)

            return
```

QUEUE LINKED LISTS



```
# Driver Code

    q = Queue()
    q.EnQueue(10)
    q.EnQueue(20)
    q.display()
    q.DeQueue()
    q.DeQueue()
    q.EnQueue(30)
    q.display()
    q.EnQueue(40)
    q.EnQueue(50)
    q.DeQueue()
    q.display()

    print("Queue Front " + str(q.front.data))
    print("Queue Rear " + str(q.rear.data))
```

OUTPUT



```
10
20
30
30
40
50
Queue Front 40
Queue Rear 50
```