Data Structures

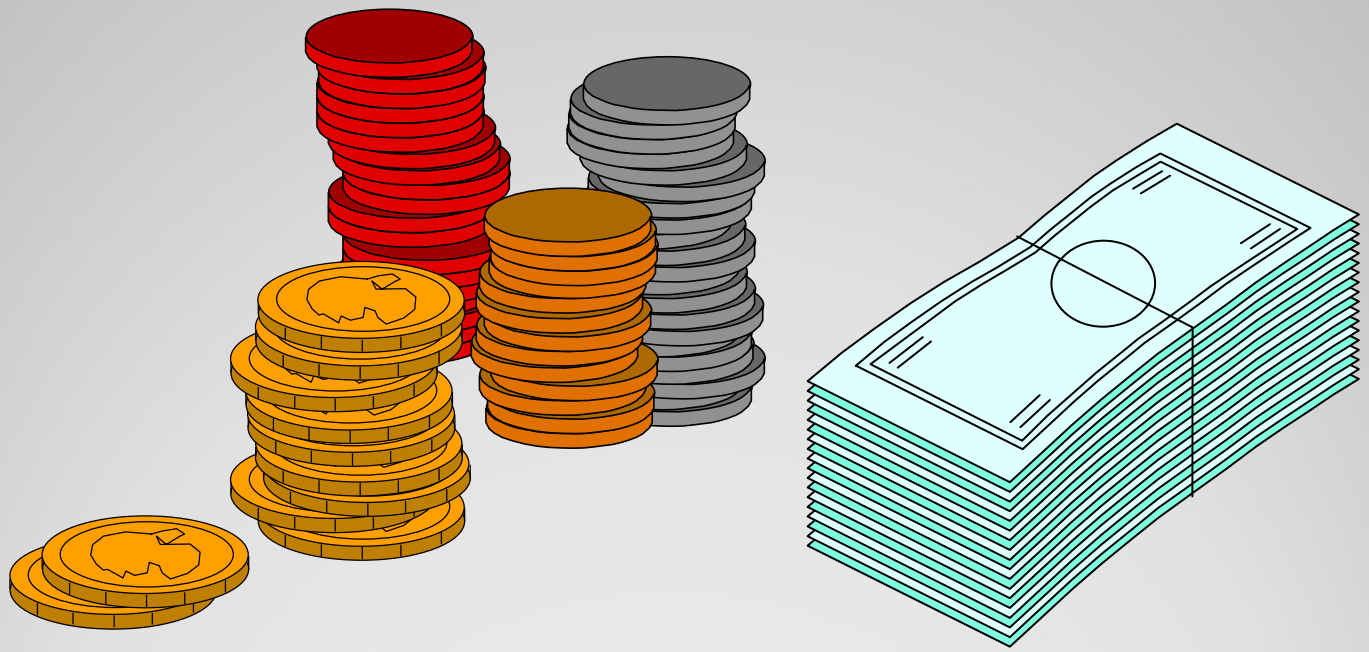
Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

- Stack (Linked)

Today's Lecture

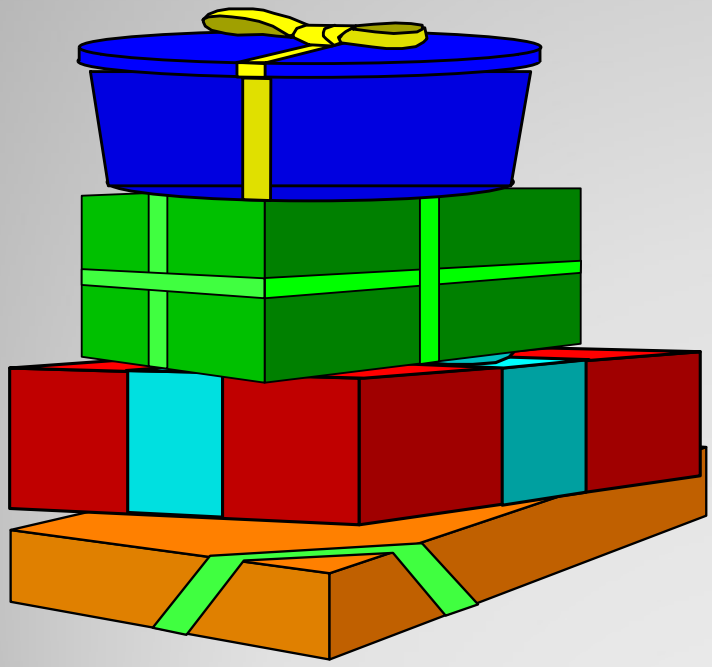
Goals

- Describe a stack and its operations at a logical level
- Demonstrate the effect of stack operations using a particular implementation of a stack
- Implement the Stack ADT, in both an array-based implementation and a linked implementation

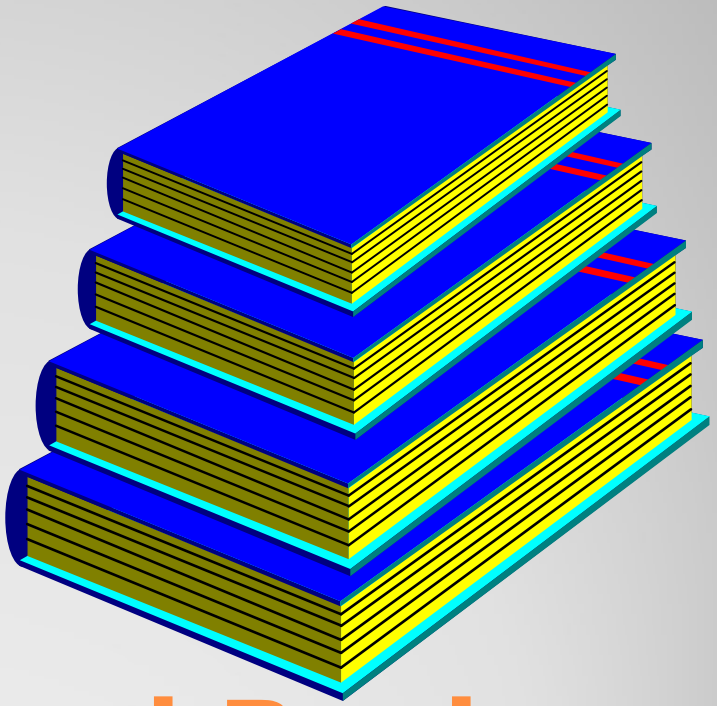


Stacks of Coins and Bills

TOP OF THE STACK



TOP OF THE STACK



Stacks of Boxes and Books

- *What do these composite objects all have in common?*

Stack

Stack

An abstract data type in which elements are added and removed from only one end (LIFO)

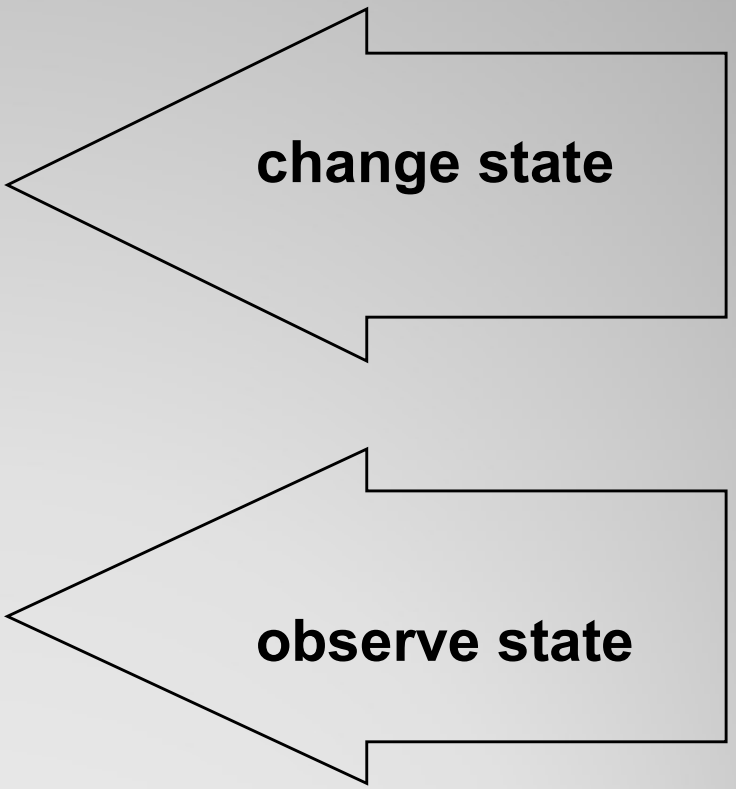
Stack

Stack

- *What operations would be appropriate for a stack?*

Stacks

- **Transformers**
  - Push
  - Pop
- **Observers**
  - ***IsEmpty***
  - ***IsFull***
  - ***Top***



Stack

Here is the interface for the Stack ADT:

```
public interface Stack {
    void push(int item) throws Exception;
    void pop() throws Exception;
    int top() throws Exception;
    void makeEmpty();

    boolean isEmpty();
    boolean isFull();
}
```

Same methods as for the array-based implementation

- What does a stack look like if we insert the following elements (in the given order):
  11, 14, 32

Stack

- *Insert: 11, 14, 32*
- *Which stack was created from the above insertions?*

Stack

- *Insert: 11, 14, 32*
- *Answer: Stack A occurs with the above insertions.*

Stack

- Now we will move on to the linked implementation of the stack.

Stack (Linked)

StackLinked Class

- We will write an StackLinked class that implements our Stack interface.

```
public class StackLinked implements Stack
{
   // Implementation code goes here…
}
```

Node

- The linked stack data structure requires that we keep more information at EACH place inside of it.
- Each item in the stack will be a "Node" (not just the data).
- A node stores the data and a reference to the next node
- It should be defined as an inner class within the StackLinked class.

```
class Node {
   Declare int data
   Declare Node next
}
```

Data for this node (change data type as necessary to store other types of data)

Points to next node in list

- Link-based **private** members

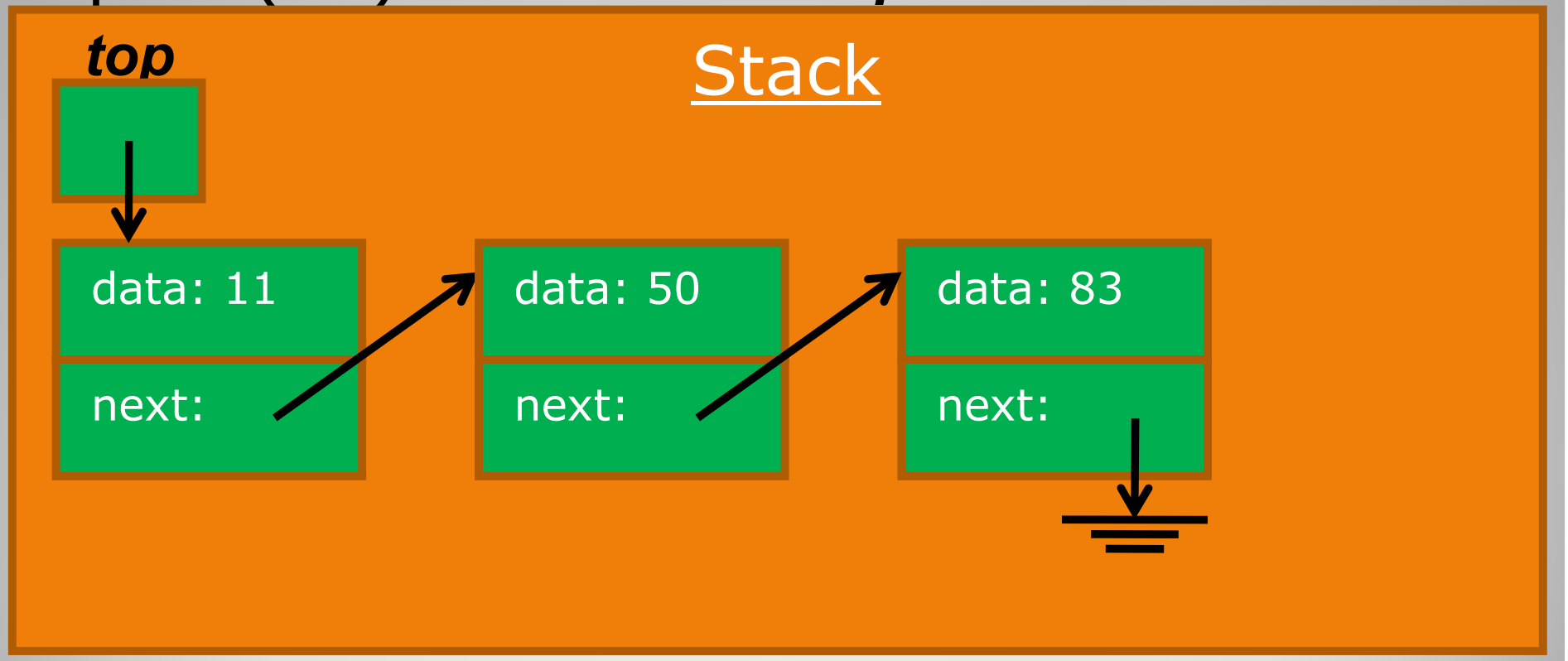
```
class StackLinked implements Stack {
    Declare Node top

   // Public members go here…
}
```

Stack (Linked)

s.push(83)
s.push(50)
s.push(11)

Assume the stack has 3 elements pushed on to it

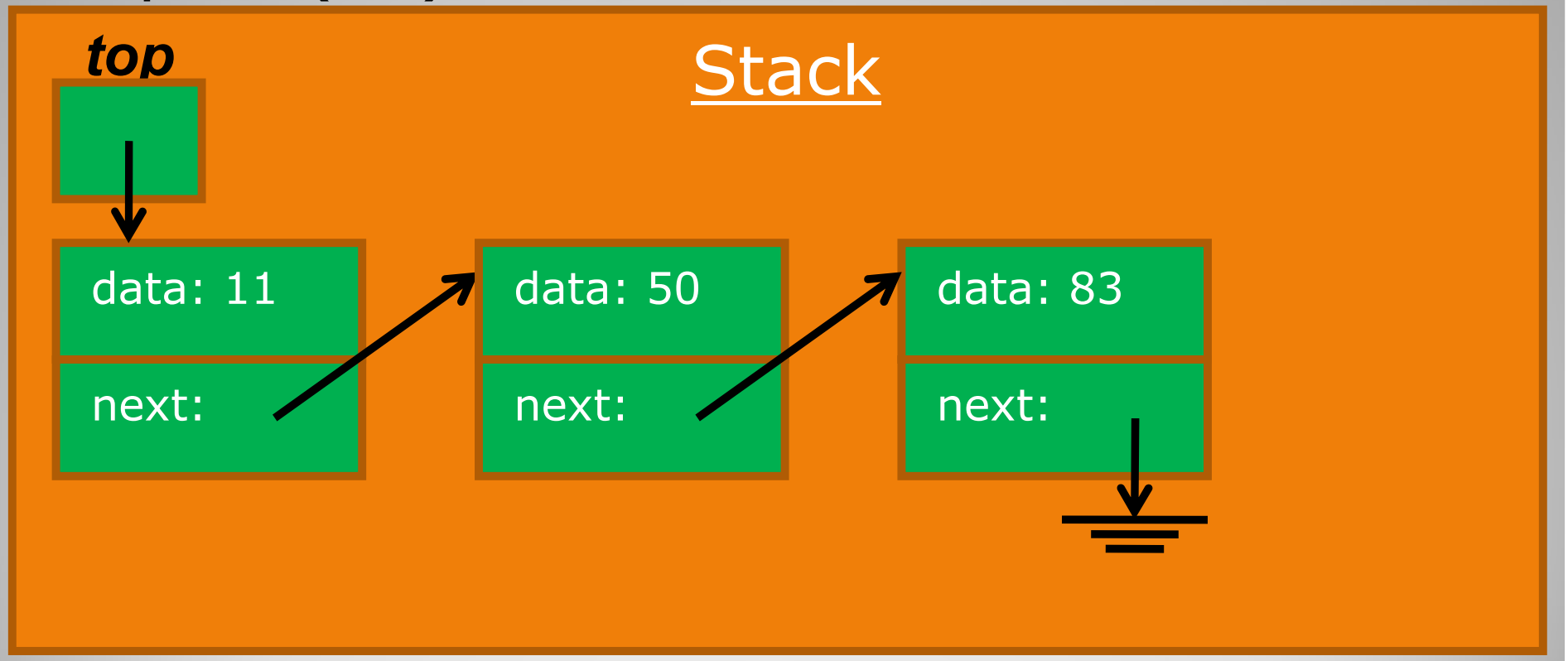


Stack (Linked)

Stack (Linked) - Push

- Where would a new item go? How is it inserted?

  s.push(77)



Stack (Linked) - push

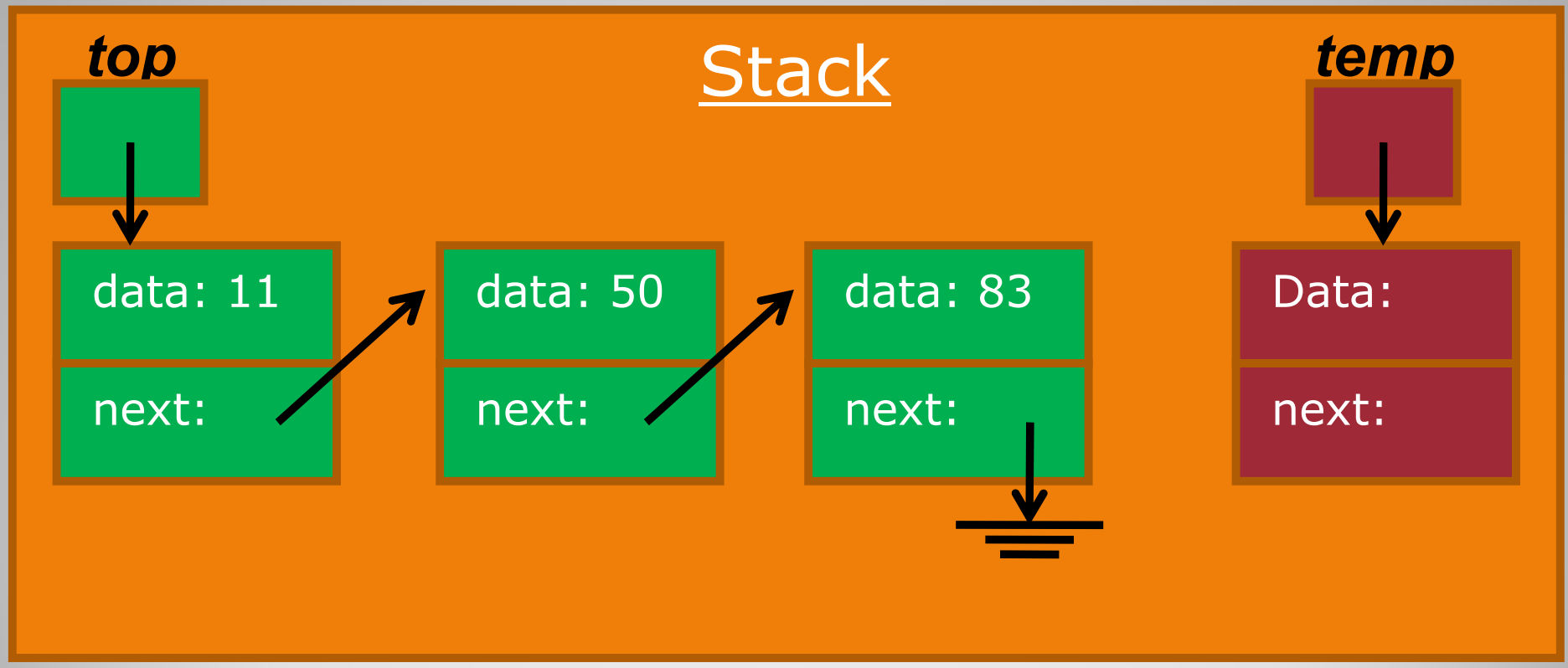
You ***MUST*** push the item on to the top of the stack.

push Pseudocode

1. Create a new Node item (dynamically allocate).
2. Set the fields on the new Node item. This means setting the data item and the next pointer. The next pointer should be set to the current top.
3. Set the top pointer to the new Node Item.

s.push(77)

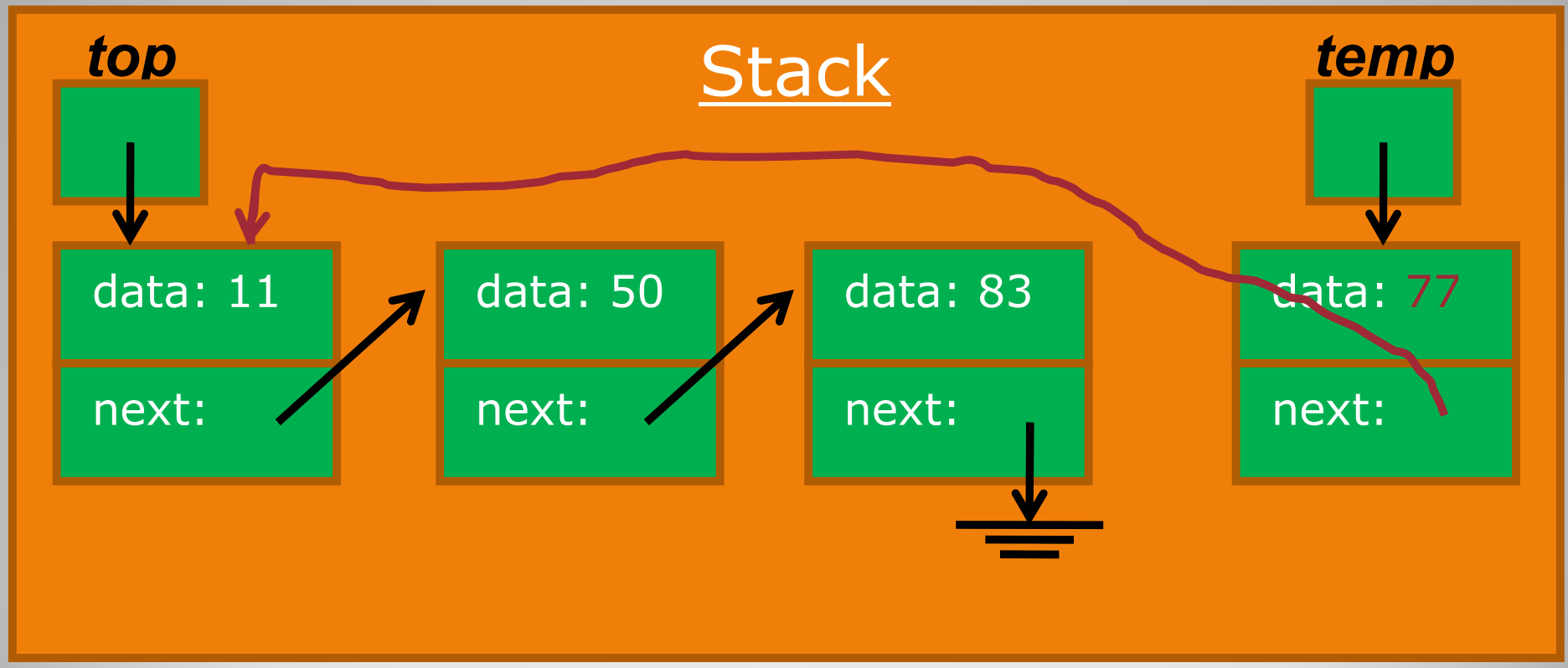
1. Create a new Node item (dynamically allocate).



Stack (Linked) - push

s.push(77)

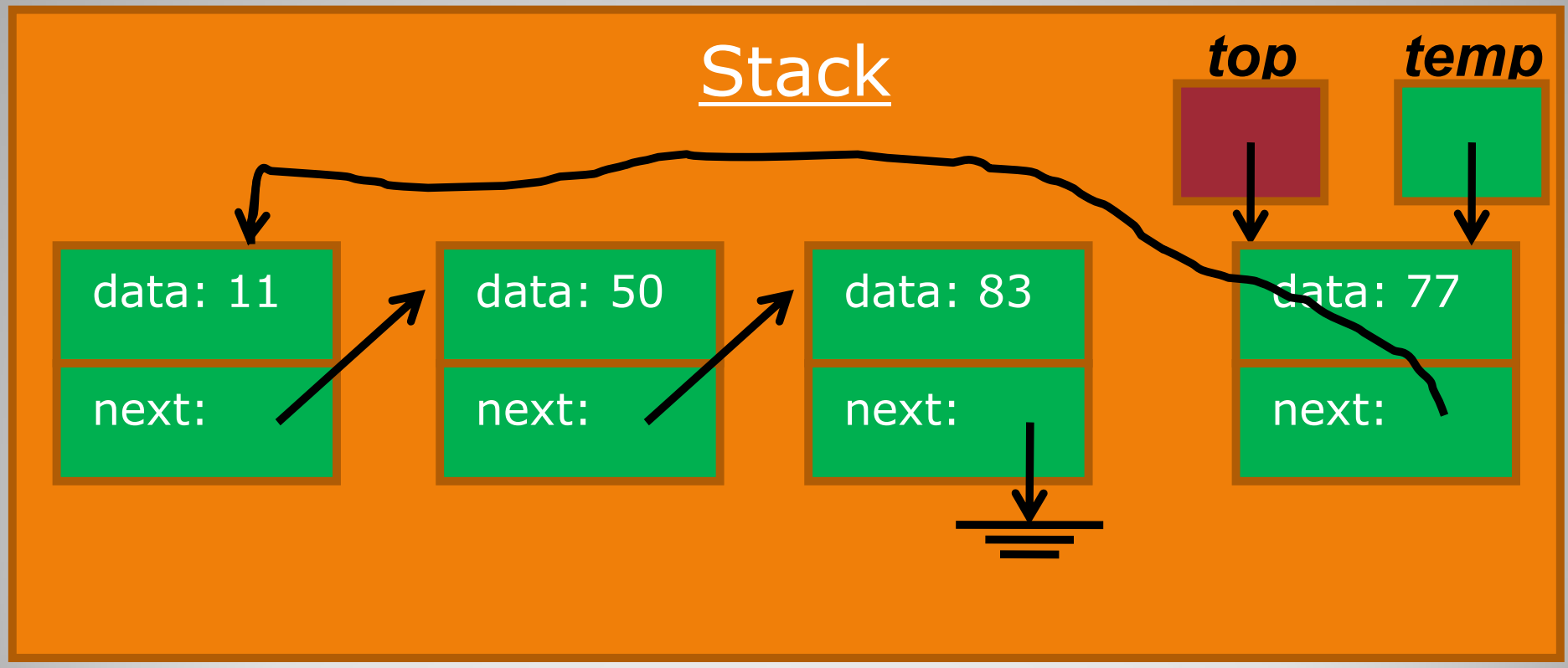
2. Set the fields on the new Node item. Set data item and next pointer. Next points to current top.



Stack (Linked) - push

s.push(77)

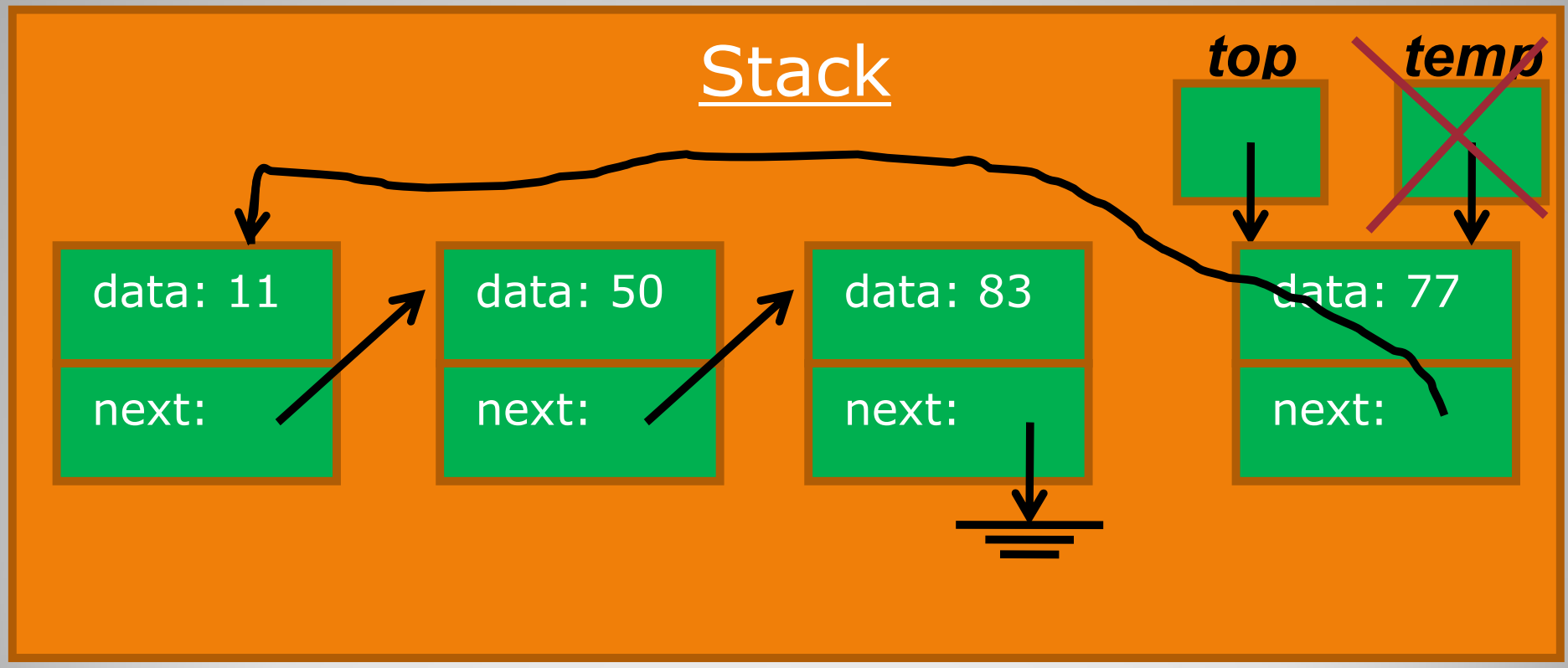
3. Set the top pointer to the new NodeType Item.



Stack (Linked) - push

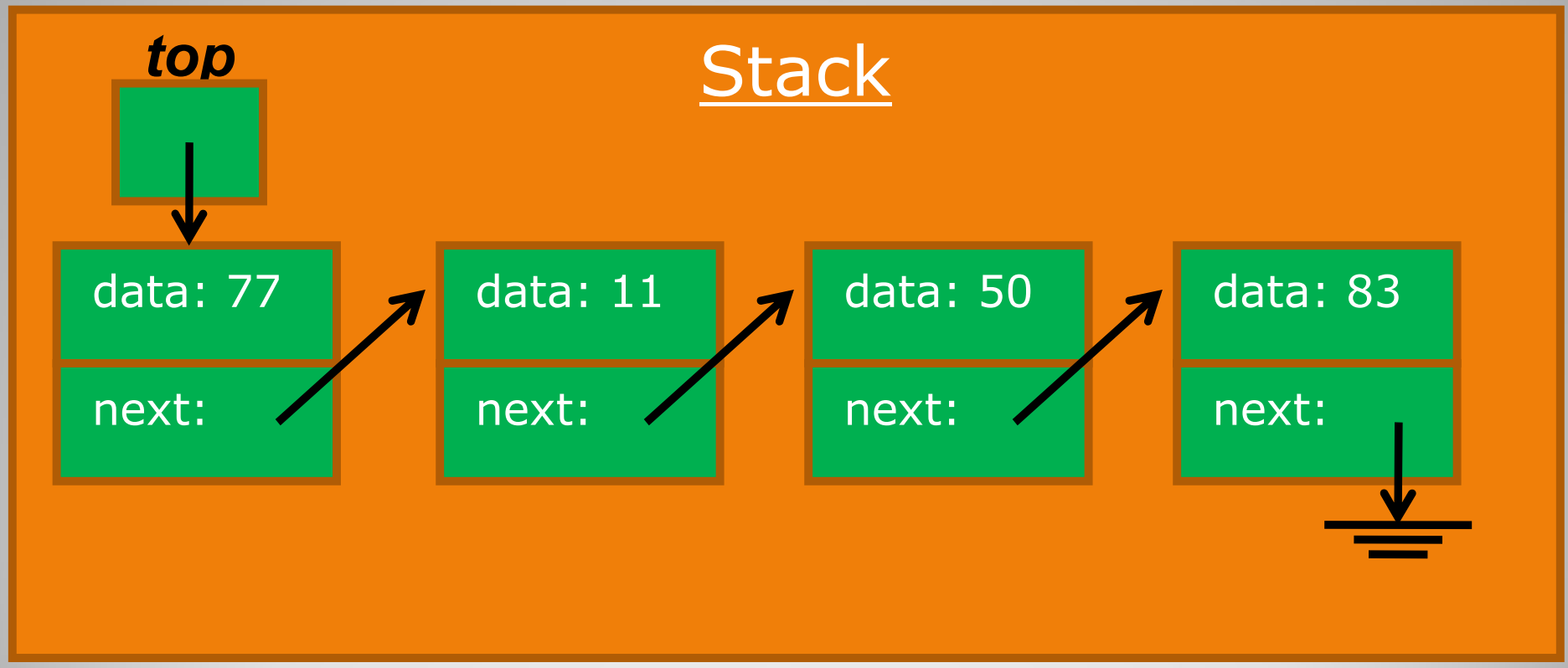
s.push(77)

When the push method ends the temp pointer will go out of scope and disappear.



Stack (Linked) - push

This picture is ***LOGICALLY EQUIVALENT*** to the previous slide!!!



Stack (Linked) - push

```
push(int item) throws Exception
   if (stack is full)
      throw exception "Full Stack"
   else


      Declare Node temp
      Set temp to new Node instance
      Set temp.data to item
      Set temp.next to top
      Set top to temp
```

Allocate a new node

Set values on the new node

Place the new node at the top of the stack

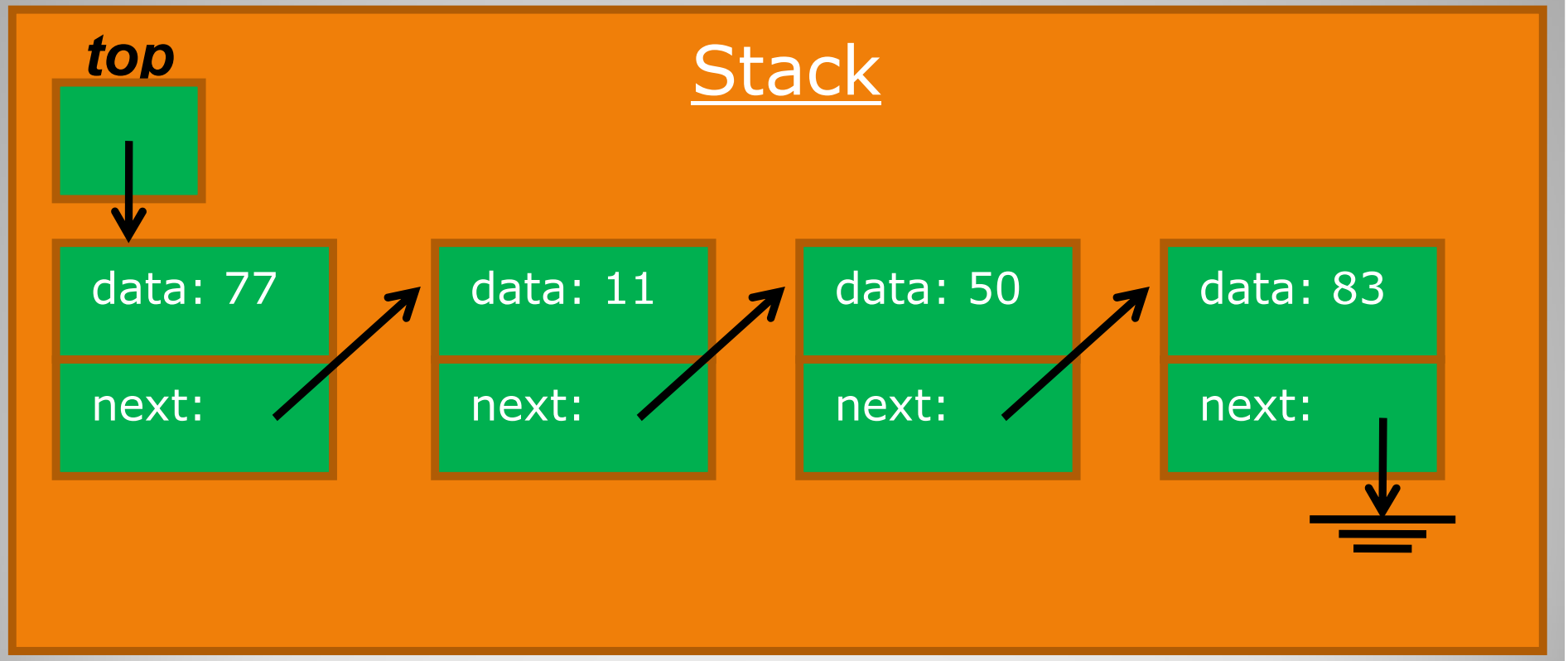
Stack (Linked) - push

pop Pseudocode

1. Create a temporary pointer to the top node.
2. Update the top pointer so that it points to the second item in the stack.
3. Deallocate the memory pointed to by the temporary pointer (the old top node).

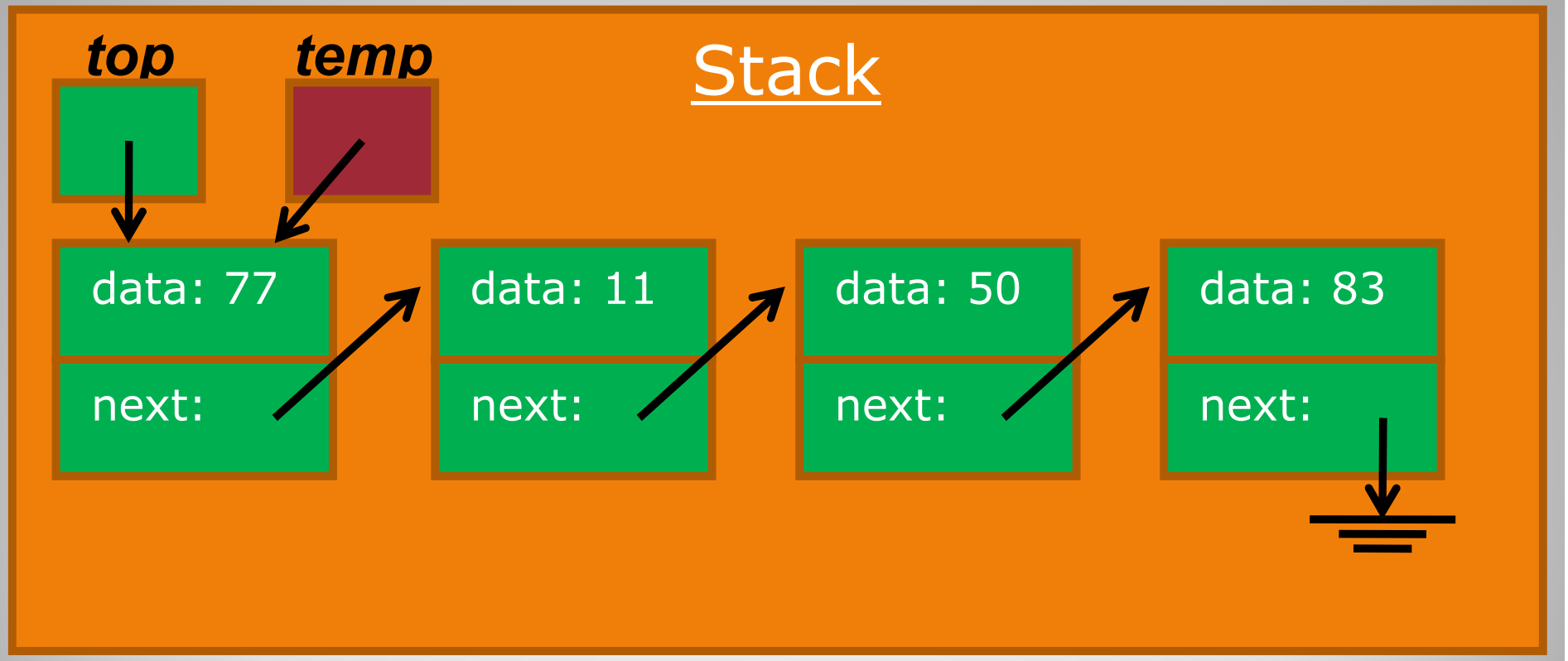
Stack (Linked) - pop

s.pop()



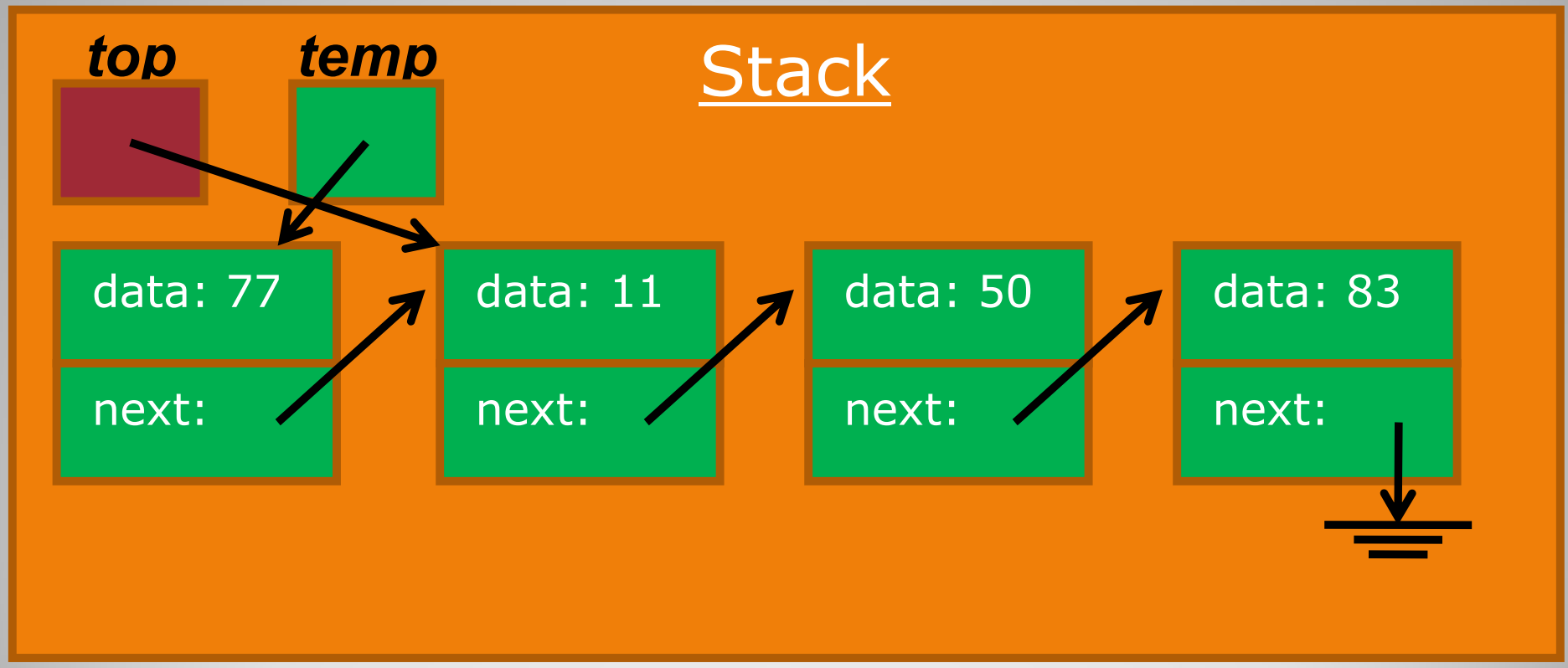
Stack (Linked) - pop

1. Create a temporary pointer to the top node.



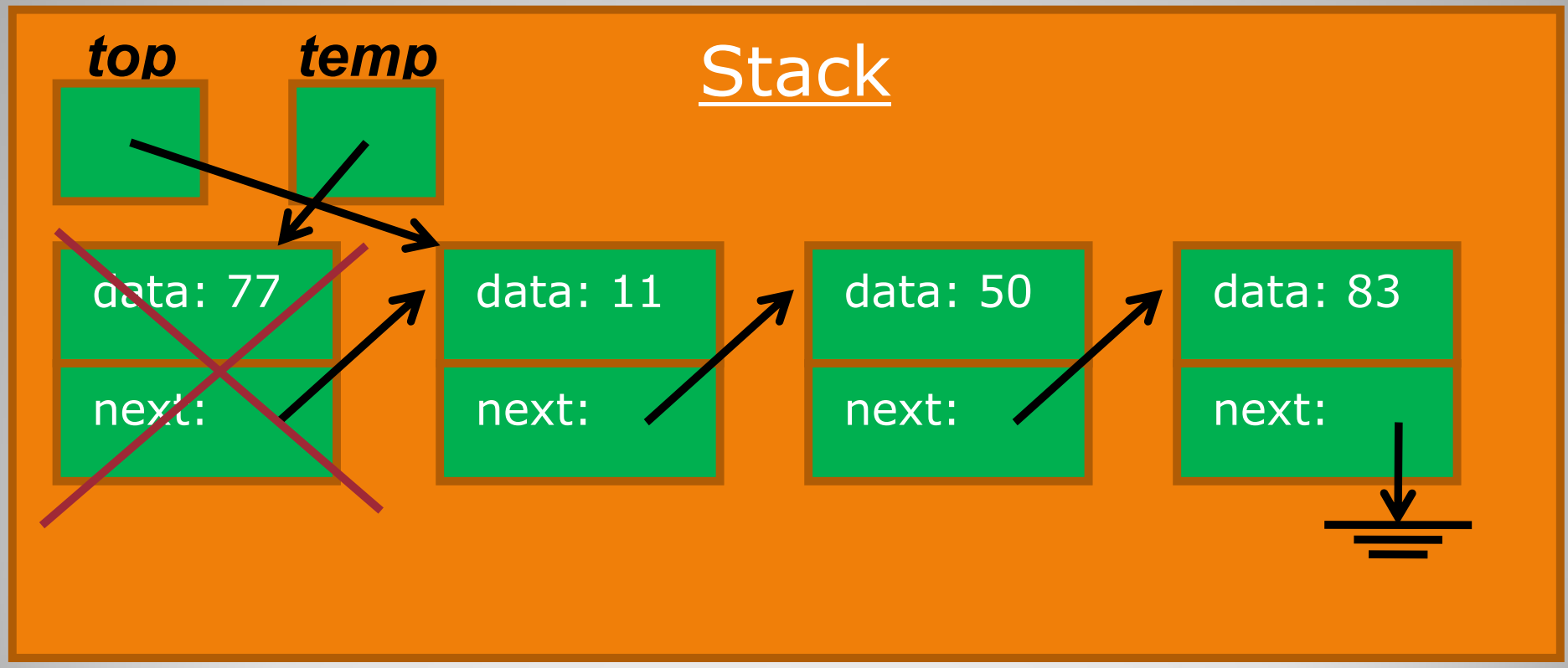
Stack (Linked) - pop

2. Update the top pointer so that it points to the second item in the stack.



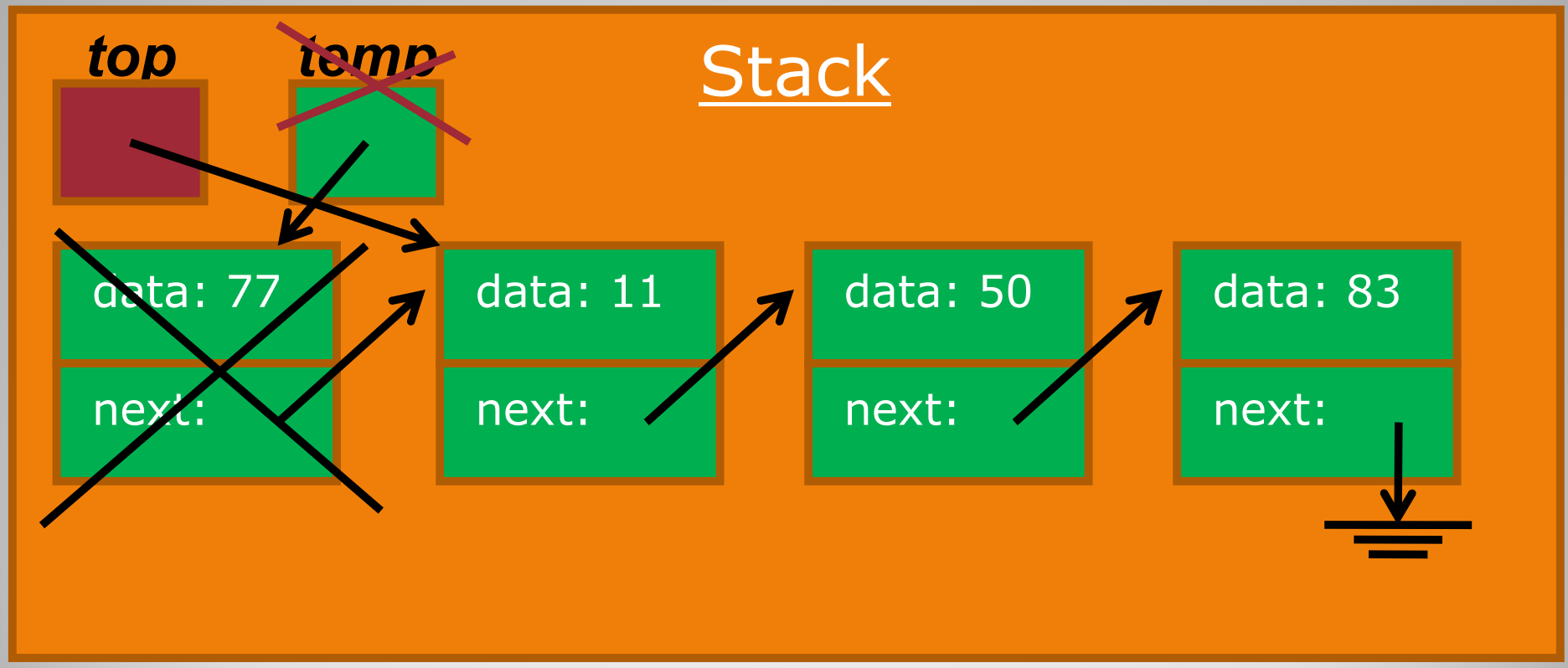
Stack (Linked) - pop

3. Deallocate the memory pointed to by the temporary pointer (the old top node).



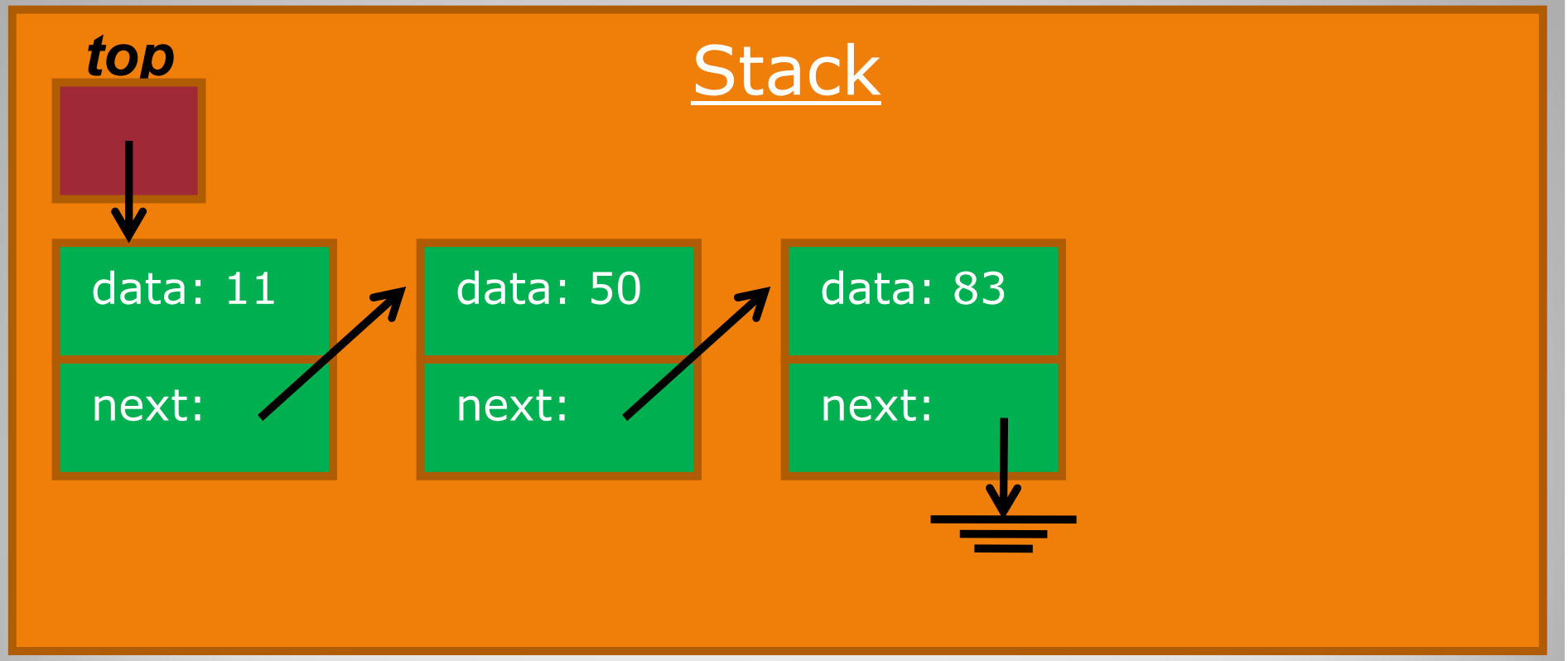
Stack (Linked) - pop

The temp pointer will disappear when the Pop method ends.



Stack (Linked) - pop

This picture is ***LOGICALLY EQUIVALENT*** to the previous slide!!!



Stack (Linked) - pop

```
pop() returns int and throws exception
    Declare int data
    if (stack is empty)
        throw exception "Empty Stack"
    else
        Set data to top.data

        Declare Node temp
        Set temp to top

        Set top to top.next
        Set temp to null

    return data
```

Create a temporary pointer to the top node

Now set top to the second item

Release memory for the old top node (will become a candidate for garbage collection)

Stack (Linked) - pop

StackLinked Constructor

Set top to null

Constructor

isEmpty() returns boolean

return top equals null

IsEmpty

Stack (Linked) – Constructor and isEmpty

```
isFull() returns boolean
    Declare Node location
    try
        Set location to new Node instance
        Set location to null
        return false
    catch (OutOfMemoryError ome)
        return true
```

Check to see if you can allocate memory.

If you CAN, then the list is NOT full so return false.

If you CANNOT allocate memory, then the list is full.

Stack (Linked) - isFull

- To get data from the stack you need to call the top() method (for this particular ADT).

- Note: There are other Stack ADTs where the pop() method actually returns the value of the top item.

Stack

```
int top() throws Exception
   if (stack is empty)
      throw exception "Empty Stack"

   return top.data
```

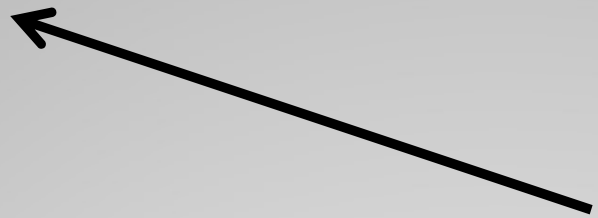
Return the data in the top node.

Stack (Linked) - top

Stack (Linked) – makeEmpty

makeEmpty()
Set top to null

Deletes <u>ALL</u> of the elements in the stack.



Make top null.

All nodes in the stack are now unreferenced so they will become candidates for garbage collection

- What are the Big-O runtimes for the linked implementation of a stack?

Stack (Linked) – Big O

Operation	Cost
isFull	O(1)
isEmpty	O(1)
push	O(1)
pop	O(1)
makeEmpty	O(1)
Constructor	O(1)

Stack (Linked) – Big O

- **End of Slides**

End of Slides