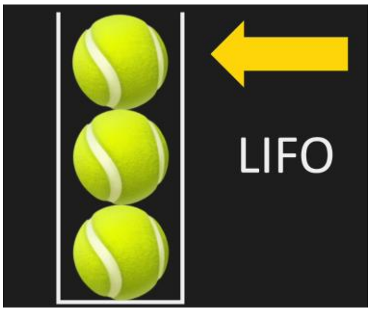
Stack:

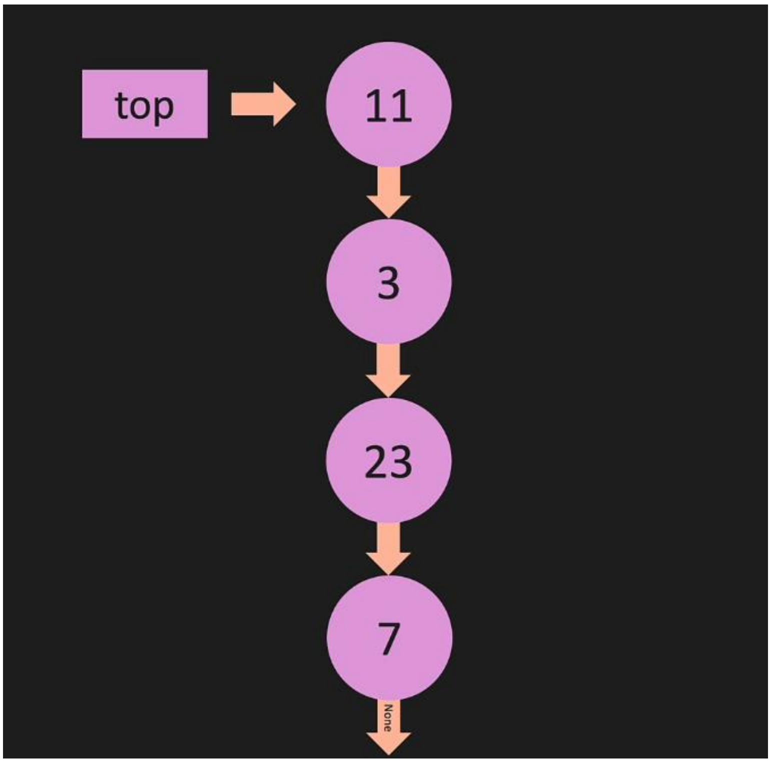
LIFO – Last In, First Out



Used in web browser, to go the previous sites

Data structure that can be used

- **List**: pop and push on the **end O(1)**
- **LinkedList**: pop and push on the **beginning O(1)**



Stack: Constructor

Create a **Stack** class that represents a last-in, first-out (LIFO) data structure using a linked list implementation.

The Stack class should contain the following components:

1. A **Node** class, which serves as the building block for the linked list. The **Node** class should have an **__init__** method that initializes the following attributes:

    - **value**: The value of the node.
    - **next**: A reference to the next node in the list, initialized to None.

2. The **Stack** class should have an **__init__** method that initializes the stack with a single node, using the given value. The **__init__** method should perform the following tasks:
    - Create a new instance of the **Node** class using the provided value.
    - Set the **top** attribute of the Stack class to point to the new node.
    - Initialize a **height** attribute for the Stack class, which represents the current number of nodes in the stack, and set it to 1.

Stack: Push

Implement the **push** method for the Stack class that adds a new node with a given value to the top of the stack.

The method should perform the following tasks:

1. Create a new instance of the **Node** class using the provided value.
2. Set the **next** attribute of the new node to point to the current top node.
3. Update the **top** attribute of the Stack class to point to the new node.
4. Increment the **height** attribute of the Stack class by 1.

```
    def push(self, value):
        new_node = Node(value)
        new_node.next = self.top
        self.top = new_node
        self.height += 1
```

Stack: Pop

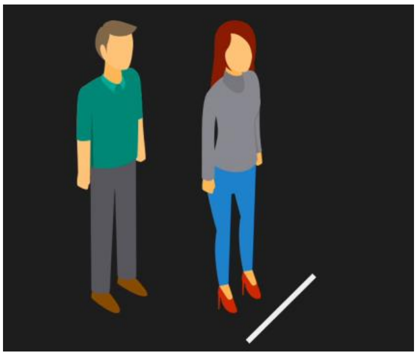
Implement the **pop** method for the Stack class that removes the top node from the stack and returns it.

The method should perform the following tasks:

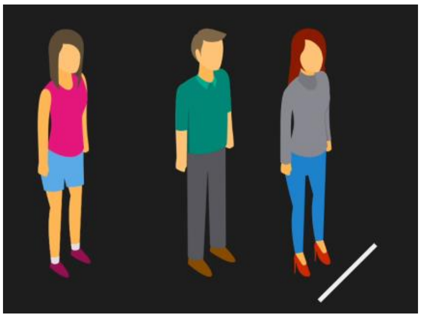
1. If the stack is empty (i.e., the **height** is 0), return None.
2. Store a reference to the current top node in a temporary variable, **temp**.
3. Update the **top** attribute of the Stack class to point to the next node in the stack.
4. Set the **next** attribute of the removed node (stored in the temporary variable) to None.
5. Decrement the **height** attribute of the Stack class by 1.
6. Return the removed node (stored in the temporary variable).

Queue:

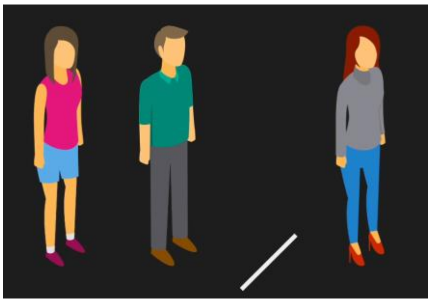
FIFO: First in, First out



Enqueue

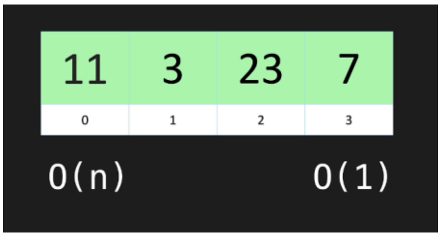


Dequeue



Data Structure usable:

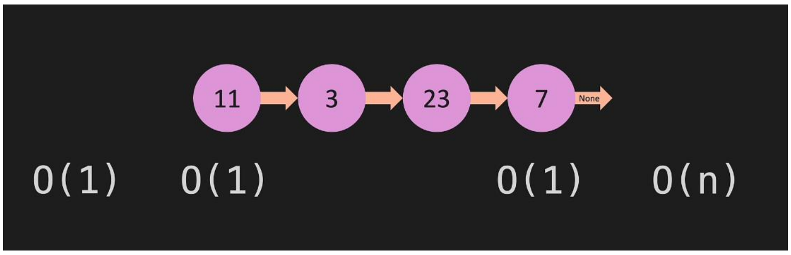
List:



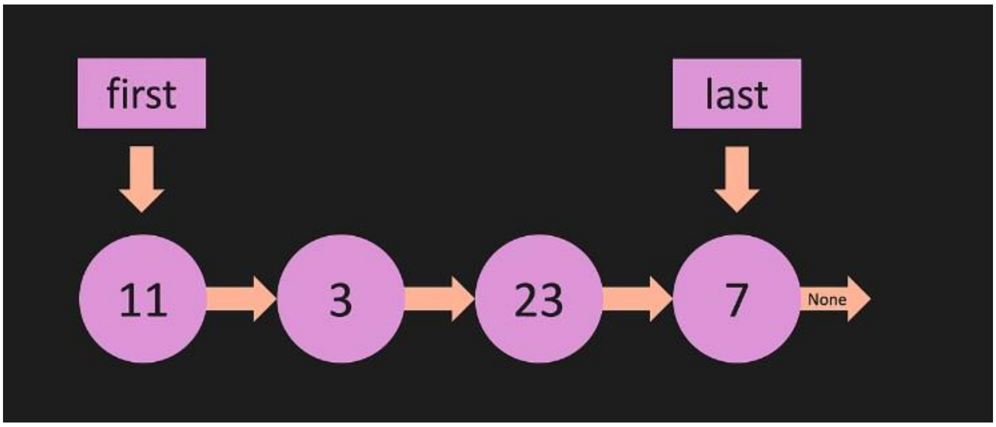
11	3	23	7
0	1	2	3

O(n) O(1)

LinkedList:



Enqueue from **last** and dequeue from **first**



Queue: Constructor

Create a Queue class that represents a first-in, first-out (FIFO) data structure using a linked list implementation.

The Queue class should contain the following components:

1. A **Node** class, which serves as the building block for the linked list. The **Node** class should have an **__init__** method that initializes the following attributes:
    - **value**: The value of the node.
    - **next**: A reference to the next node in the list, initialized to None.
2. The **Queue** class should have an **__init__** method that initializes the queue with a single node, using the given value. The **__init__** method should perform the following tasks:
    - Create a new instance of the **Node** class using the provided value.
    - Set the **first** attribute of the Queue class to point to the new node.
    - Set the **last** attribute of the Queue class to point to the new node.
    - Initialize a **length** attribute for the Queue class, which represents the current number of nodes in the queue, and set it to 1.

Queue: Enqueue

Implement the **enqueue** method for the Queue class that adds a new node with a given value to the end of the queue.

The method should perform the following tasks:

1. Create a new instance of the **Node** class using the provided value.
2. If the queue is empty (i.e., **self.first** is None), set the **first** and **last** attributes of the Queue class to point to the new node.
3. If the queue is not empty, perform the following steps:
    - Set the **next** attribute of the current last node to point to the new node.
    - Update the **last** attribute of the Queue class to point to the new node.
4. Increment the **length** attribute of the Queue class by 1.

Queue: Dequeue

Implement the **dequeue** method for the Queue class that removes the first node from the queue and returns it.

The method should perform the following tasks:

1. If the queue is empty (i.e., the **length** is 0), return None.
2. Store a reference to the current first node in a temporary variable, **temp**.
3. If the queue has only one node (i.e., the **length** is 1), set both the **first** and **last** attributes of the Queue class to None.
4. If the queue has more than one node, perform the following steps:
    - Update the **first** attribute of the Queue class to point to the next node in the queue.
    - Set the **next** attribute of the removed node (stored in the temporary variable) to None.
5. Decrement the **length** attribute of the Queue class by 1.
6. Return the removed node (stored in the temporary variable).