# Stacks, Queues, and Deques

## Key Terms

- Stack
- Queue
- Deque
- LIFO
- FIFO

## Essential Questions

- How does the ordering of stacks differ from the ordering of items in a queue?
- Where do we see stacks and queues in the real world?
- What are common use cases for stacks and queus in computer science?

# TUTorial STACK

**STACK**

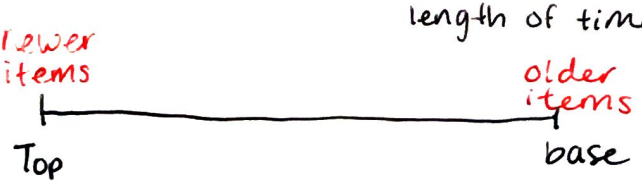
- Ordered collection of items
- Addition of new items and the removal of existing items takes place at same end
  - ↳ the end aka the "top"
  - ↳ opposite to the "top" is the "base"

**BASE of STACK**

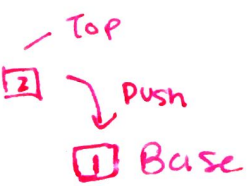
- Items stored closer to the base; represent those that have been in the stack the longest
- Most recent item is the one that is in position to be removed first

**LIFO**

- Last-in First-out principal
  - ↳ provides ordering based on length of time in the collection

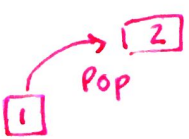



visual Example




Stacks are fundamentally important

- ↳ can be used to reverse the order of items
- ↳ Order of insertion is reverse of removal order



web page navigation is a stack
↳ urls are in a stack
↳ current page - top
first page - base

## Stack Operations

| | |
|---|---|
| Stack() | - creates a new empty stack<br>- needs no parameters<br>- returns an empty stack |
| push(item) | - pushes new item to top of the stack |
| pop() | - Removes the top item from the stack<br>- needs no parameters<br>- returns the item<br>- stack is modified |
| peek() | - Returns the top item from the stack but does not remove it.<br>- no parameters<br>- Stack is not modified |
| isEmpty() | - Check if stack is empty<br>- Returns boolean |
| size() | - Returns # of items in the stack<br>↳ integer |

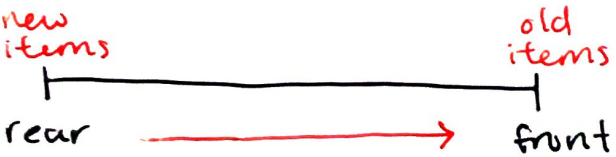
Implementation of Stack.

```
class Stack (object):

        def __init__ (self):
                self.items = []

        def __isEmpty__ (self):
                self.items = []

        def push (self, item ):
                self.items.append (item)

        def pop (self ):
                self.items.pop()

        def peek ( self)
                return self.items[len(self.items) - 1]

        def size ( self):
            return len (self.items)

s = Stack()
s.isEmpty()
```

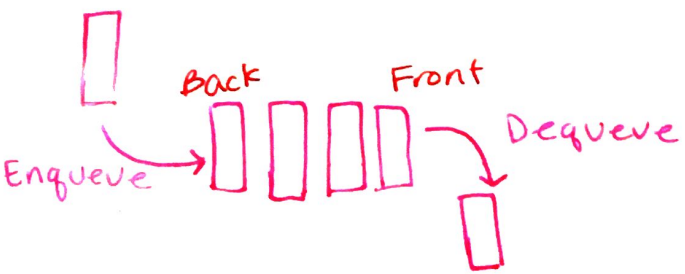
# Queue

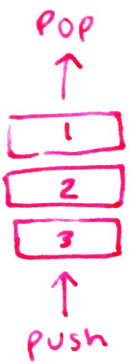
Queue
- Ordered collection of items
- Addition of new items happens at one end called the "rear"
- Removal of existing items occurs at the other end called the "front"
  - ↳ element starts at rear & makes its way toward the front
  - ↳ waits until it is its time to be removed

FIFO
first-in first-out
- Most recently added item must wait at the end of the collection
- Item that has been in the collection the longest is at the front.




Visual Example




Enqueue - Add new item

Dequeue - Removing item from queue

# Queue Operations

| | |
|---|---|
| Queue() | - creates empty queue |
| enqueue(item) | - Adds a new item to the rear of queue<br>- needs an item<br>- returns nothing |
| dequeue() | - removes the front item from the queue<br>- needs no parameters<br>- returns the item<br>- queue is modified |
| isEmpty() | |
| size() | |

# Implementation of Queue

```
class Queue(object):
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def enqueue(self, item):
        self.items.insert(0,item)
    def dequeue(self):
        return self.items.pop()
    def size(self):
        return len(self.items)
```