Stacks

- Consider a stack of pancakes, when one pancake is added, you always put it on top, and always remove fromthe top
  - Very simple
    - Add/Remove from top
    - Don't have to think about it's position
- Characteristics
  - Items leave stack in reverse order
    - LIFO (Last in First out)
- Stack Functions:
  - is_empty() # is the stack empty
  - push() # add item to the stack
  - pop() # remove item from the stack
- Stacks can be used as follows:

```
stack = Stack()
for v in "aeiou"
    stack.push(v)

while not stack.is_empty():
    print(stack.pop())
```

This will add the vowels to the stack in order, then print them from last to first

---

Class Definition

```
class Stack:
    def __init__(self, capacity=10):
        self.data = [0] * capacity
        self.top = 0

    def is_empty(self):
        return self.top == 0

    def push(self, item):
        if self.top < len(self.data):
            self.data[self.top] = item
        else:
            raise Exception("The stack is full")
    self.top += 1
```

```
def pop(self):
    if self.is_empty():
        return None
    else:
        self.top -= 1
        return self.data[self.top]
```

Complexity analysis

Operation	Big O Perfromance
is_empty()	Constant Time
push()	Constant Time
pop()	Constant time

Uses

- Surprisingly Useful:
  - Browser History
  - Reverse Input
  - Matching Brackets
  - Undo Function
