Stacks

- A Stack is a collection of objects that are inserted and removed according to the last-in, first-out (LIFO) principle.
- A user can insert objects into a stack at anytime, but may only access or remove the most recently inserted object that remains (at the so-called "top" of the stack).
- Think like a PEZ candy dispenser.
- Stacks are an fundamental data structure

Examples:

1: Web browsing back and forth b/w a page.

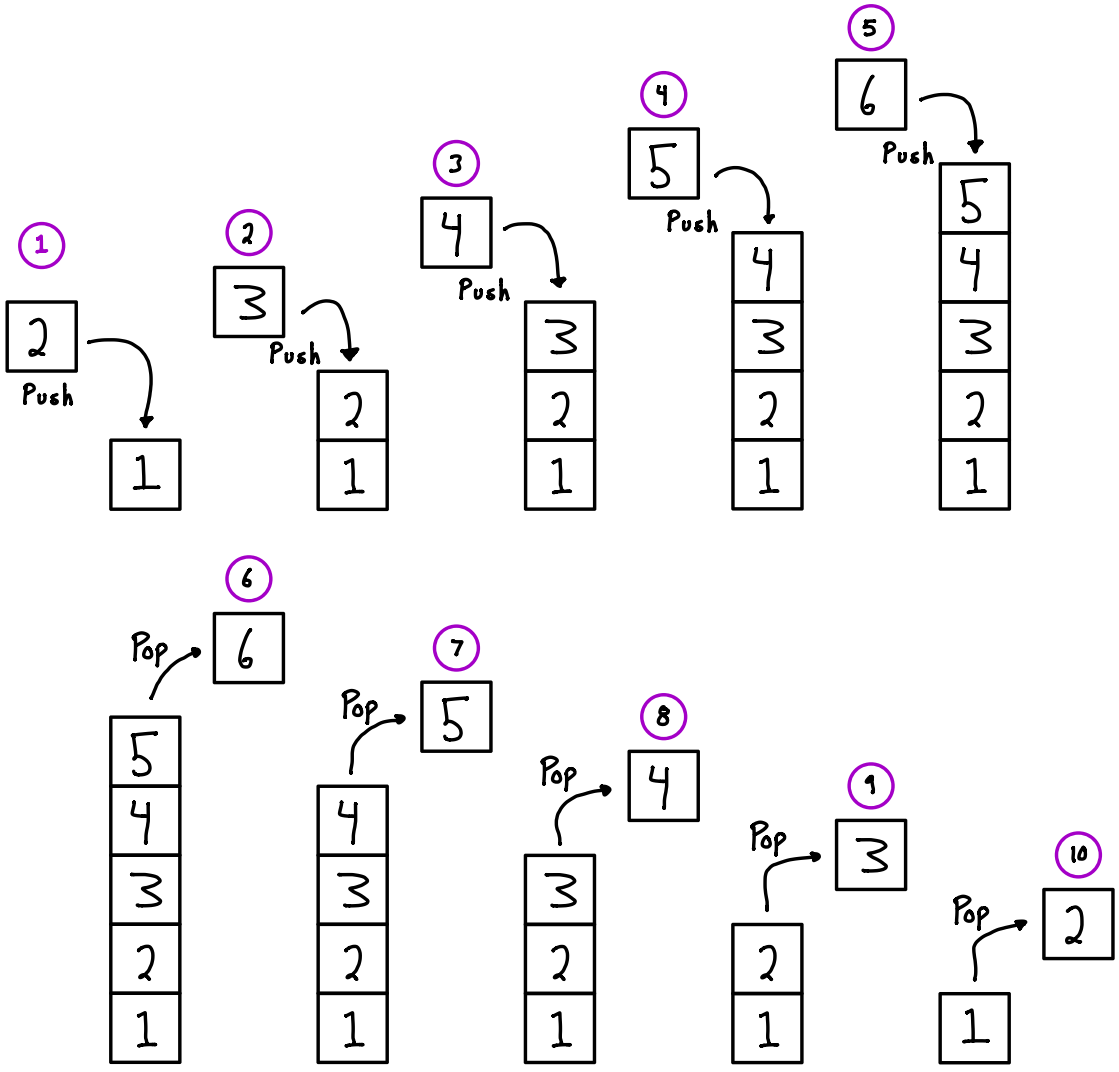
2: Text Editors, like notability provide an "undo" mechanism, keeping changes in a stack.

The Stack Abstract Data Type:

- Stacks are the simplest of all data structures, yet among the most important
- Stacks are an Abstract Data Type (ADT), that an instance S supports:

Process:

- "Push" onto the stack and "Pop" of the stack



Simple Array-Based Stack Implementation

- Can implement a stack using a Python list.

Python Code:

```
class Stack():

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

    def push(self, item):
        '''Pushes element on the stack'''
        self.items.append(item)

    def pop(self):
        '''Remove element from the top of the stack'''
        return self.items.pop()

    def is_empty(self):
        '''Check if Stack is empty'''
        return self.items == []

    def peek(self):
        '''Peek the top element on the stack'''
        if not self.is_empty():
            return self.items[-1]

    def get_stack(self):
        '''Return Stack list'''
        return self.items

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


s = Stack()
s.push("A")
s.push("B")
s.push('C')
s.push('D')
print('What did we pop?:',s.pop())

print(s.get_stack())
```

Analyzing the Array-Based Stack Implementation

Operation	Running Time
S.push(e)	O(1)
S.pop()	O(1)
S.peek()	O(1)
S.is_empty()	O(1)
len(S)	O(1)

- Running times of ArrayStack methods

~ O(n)-time worst case, where n is the # of elements in the stack

~ Space storage usage for a stack is O(n)

Implementation of a Stack to print reverse characters

```
class Stack():
    def __init__(self):
        self.items = []

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

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

    def is_empty(self):
        return self.items == []

    def peek(self):
        if not self.is_empty():
            return self.items[-1]

    def get_stack(self):
        return self.items


def reverse_string(stack, input_str):

    # Loop through the string and push contents
    # character by character onto stack.
    for i in range(len(input_str)):
        stack.push(input_str[i])

    rev_str = ""
    while not stack.is_empty():
        rev_str += stack.pop()

    return rev_str


stack = Stack()
input_str = "Devin Powers"

print(reverse_string(stack, input_str))
```

Added Function

→ sreoP nivcD