Stacks

Introduction

A **stack** is a fundamental data structure that follows the **Last-In-First-Out (LIFO)** principle. Imagine a stack of plates: when you add a new plate, it goes on top, and the only plate you can remove is the one at the top[13]. Here are the key points:

1. **Definition**: A stack is a linear data structure where elements are inserted and removed from the same end, known as the **top** of the stack.
2. **LIFO Principle**: The last element added is the first one to be removed.
3. **Operations**:
   - `push()`: Adds an element to the top of the stack.
   - `pop()`: Removes the top element from the stack.
   - `top()`: Returns the top element without removing it.
4. **Implementation**:
   - **Fixed Size Stack**: Has a predefined capacity and cannot dynamically grow or shrink.
   - **Dynamic Size Stack**: Can resize as needed, often implemented using linked lists.
5. **Use Cases**: Stacks are used in various applications, such as function calls, expression evaluation, and undo functionality in software[2].

Implementation

Using Built-in List

```python
stack = []
stack.append('a')  # Push 'a' onto the stack
stack.append('b')  # Push 'b' onto the stack
stack.append('c')  # Push 'c' onto the stack
print('Initial stack:', stack)
print('Elements popped from stack:')
print(stack.pop())  # Pop 'c'
print(stack.pop())  # Pop 'b'
print(stack.pop())  # Pop 'a'
print('Stack after elements are popped:', stack)
```

By Creating a Stack Class

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

    def push(self, item):
        """Add an item to the top of the stack."""
        self.items.append(item)

    def pop(self):
        """Remove and return the top item from the stack."""
        if not self.is_empty():
            return self.items.pop()
        else:
            raise IndexError("Stack is empty!")

    def top(self):
        """Return the top item without removing it."""
        if not self.is_empty():
            return self.items[-1]
        else:
            raise IndexError("Stack is empty!")

    def is_empty(self):
        """Check if the stack is empty."""
        return len(self.items) == 0

    def size(self):
        """Return the number of items in the stack."""
        return len(self.items)

# Example usage:
stack = Stack()
stack.push(10)
stack.push(20)
stack.push(30)
print("Top item:", stack.top())  # Should print 30
print("Popped item:", stack.pop())  # Should print 30
print("Is stack empty?", stack.is_empty())  # Should print False
print("Stack size:", stack.size())  # Should print 2
```