Challenge: Make a Stack with Variable Internal Types

This lesson brings you a challenge to solve.

WE'LL COVER THE FOLLOWING

- Problem statement

Problem statement

In the last chapter, we developed some **Stack** struct-types. However, they were limited to a certain fixed internal type. Now, develop a general `stack` type using a slice. That slice should be holding elements of type `interface{ }`. Implement the following stack-methods: `Len() int`, `IsEmpty() bool`, `Push(x interface{})` and `Pop()(x interface{}, error)`.

`Pop()` returns the top most element and removes it from the stack. Also, write a method `Top()`, which only returns this element and does not remove it. Note that the stack will be implemented in the file **mystack.go**, and its functions will be called in **main.go**.

Try to solve the challenge below. Good Luck!

Environment Variables

Key:	Value:
GOROOT	/usr/local/go
GOPATH	//root/usr/local/go/src
PATH	//root/usr/local/go/src/bin:/usr/local/go...

```
package mystack
import "errors"

type Stack []interface{}

func (stack Stack) Len() int {
        return 0
```

```
}

func (stack Stack) Cap() int {

        return 0
}

func (stack Stack) IsEmpty() bool {
        return true
}

func (stack *Stack) Push(e interface{}) {
        return
}

func (stack Stack) Top() (interface{}, error) {
        return nil, nil
}

func (stack *Stack) Pop() (interface{}, error) {
        return nil, nil
}
```

We hope that you were able to solve the challenge. The next lesson brings you the solution to this challenge.