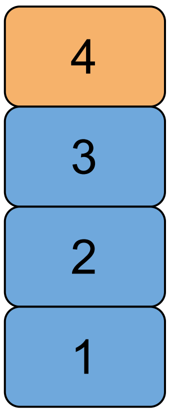
Challenge: Implement Stack Data Structure

This lesson brings you a challenge to solve.

Problem statement

Implement the **stack** data structure. It has cells to contain data. For example, integers `1`, `2`, `3`, `4`, and so on. The cells are *indexed* from the bottom (index 0) to the top (index n). Let's assume **n**=**3** for this exercise, so we have **4** places. A new stack contains **0** in all cells. A new value is put in the highest cell, which is empty (containing 0) on top (of the stack): this is called **push**. To get a value from the stack, take the value in the highest cell which is not 0: this is called **pop**. We can understand why a stack is called a *Last In First Out (LIFO)* structure.



Define a new type `Stack` for this data structure. Make 2 methods `Push` and `Pop`. Make a `String()` method (for *debugging* purposes) that shows the content of the stack as: `[0:i] [1:j] [2:k] [3:l]`. Take the underlying data structure, a **struct** containing an `index`, an array `data` of *int*, and the `ix` contains the first free position.

Generalize the implementation by making the number of elements 4 a constant `LIMIT`.

Note: `Stack` is the struct type, and `ix` and `data[LIMIT]` are its fields. The variable `ix` denotes total elements present in the stack, and `data` holds the element of a stack. Do not change the name of these variables.

Try to implement the function below. Feel free to view the solution, after giving some shots. Good Luck!

```go
package main
import "fmt"
import "strings"
import "strconv"
import "encoding/json"

const LIMIT = 4 // DONOT CHANGE IT!

type Stack struct {
        ix   int // first free position, so data[ix] == 0
        data [LIMIT]int
}

func (st *Stack) Push(n int) {

        return
}

func (st *Stack) Pop() int {

        return 0
}

func (st Stack) String() string {

        return ""
}
```

Implement Stack Data Structure

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