Challenge: Sort People with Sorter Interface

This lesson brings you a challenge to solve.

WE'LL COVER THE FOLLOWING

- Problem statement

Problem statement

Define a struct `Person` with `firstName` and `LastName`, and a type `Persons` as a `[ ]Person`. Implement the `Sorter` interface for `Persons` and test it. Implement the `Sorter` interface in the file **mysort.go** in **mysort** folder, and `Person` in **main.go**.

Try to attempt the challenge below. Feel free to view the solution, after giving some shots. 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 mysort

type Sorter interface {
    Len() int
    Less(i, j int) bool
    Swap(i, j int)
}

func Sort(data Sorter) {
    for pass:=1; pass < data.Len(); pass++ {
        for i:=0; i < data.Len() - pass; i++ {
            if data.Less(i+1, i) {
                data.Swap(i, i+1)
            }
        }
    }
}


func IsSorted(data Sorter) bool {
    n := data.Len()
    for i := n - 1; i > 0; i-- {
        if data.Less(i, i-1) {
            return false
        }
    }
    return true
}

// Convenience types for common cases
type IntSlice []int

func (p IntSlice) Len() int { return len(p) }

func (p IntSlice) Less(i, j int) bool { return p[i] < p[j] }

func (p IntSlice) Swap(i, j int) { p[i], p[j] = p[j], p[i] }

type StringSlice []string

func (p StringSlice) Len() int { return len(p) }


func (p StringSlice) Less(i, j int) bool { return p[i] < p[j] }

func (p StringSlice) Swap(i, j int) { p[i], p[j] = p[j], p[i] }

// Convenience wrappers for common cases
func SortInts(a []int) { Sort(IntSlice(a)) }

func SortStrings(a []string) { Sort(StringSlice(a)) }

func IntsAreSorted(a []int) bool { return IsSorted(IntSlice(a)) }

func StringsAreSorted(a []string) bool { return IsSorted(StringSlice(a)) }
```

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