Module 1: Functions and Organization
Topic 1.4: Passing Arrays and Slices

Passing Array Arguments

- Array arguments are copied
- Arrays can be big, so this can be a problem

```
func foo(x [3]int) int {
    return x[0]
}
func main() {
    a := [3]int{1, 2, 3}
    fmt.Print(foo(a))
}
```

Passing Array Pointers

- Possible to pass array pointers

```
func foo(x *[3]int) {
    (*x)[0] = (*x)[0] + 1
}
func main() {
    a := [3]int{1, 2, 3}
    foo(&a)
    fmt.Print(a)
}
```

- Messy and unnecessary

Pass Slices Instead

- **Slices contain a pointer** to the array
- Passing a slice copies the pointer

```
func foo(sli []int) {
    sli[0] = sli[0] + 1
}
func main() {
    a := []int{1, 2, 3}
    foo(a)
    fmt.Print(a)
}
```