Module 2: Functions and Organization
Topic 2.1: Variadic and Deferred

Variable Argument Number

- Functions can take a variable number of arguments
- Use ellipsis ... to specify
- Treated as a slice inside function

```
func getMax(vals ...int) int {
      maxV := -1
      for _, v := range vals {
            if v > maxV {
                  maxV = v
            }
      }
      return maxV
}
```

Variadic Slice Argument

- Can pass a slice to a variadic function
- Need the ... suffix

```
func main() {
      fmt.Println(getMax(1, 3, 6, 4))

      vslice := []int{1, 3, 6, 4}

      fmt.Println(getMax(vslice...))
}
```

Deferred Function Calls

- Call can be **deferred** until the surrounding function completes
- Typically used for cleanup activities

```
func main() {
      defer fmt.Println("Bye!")

      fmt.Println("Hello!")
}
```

Deferred Call Arguments

- Arguments of a deferred call are evaluated immediately

```
func main() {
      i := 1
      defer fmt.Println(i+1)
      i++
      fmt.Println("Hello!")
}
```