Naming (variables, functions, packages) Conventions is Go

Naming conventions are important for code readability and maintainability.

Naming conventions:

- Names start with a **letter** or an underscore **(_)**
- **Case matters:** quickSort and QuickSort are different variables
- Go keywords (25) can not be used as names
- Use the first letters of the words

```
var mv int //mv -> max value
```

- Use fewer letters in smaller scopes and the complete word in larger scopes

```
var packetsReceived int // NOT OK, to verbose

var n int //OK -> no. of packets received

var taskDone bool  //ok in larger scopes
```

Naming (variables, functions, packages) Conventions is Go

- Don’t use _ , it’s not idiomatic in GO

const MAX_VALUE = 100 // NOT OK

const N = 100 //OK, IDIOMATIC

- An uppercase first letter has special significance to go (it will be exported).
- The convention in Go is to use MixedCaps or mixedCaps also known as camelCase rather than underscores to write multi word names. This is applicable to variables, constants or functions
- It is convention to write acronyms in all caps.

writeToDB -> recommended, unexported visible only within the package. In this example DB a an acronym for database.

writeToDb -> not recommended

Naming (variables, functions, packages) Conventions is Go

- By convention, **packages are given lower case, single-word names;**
- Go doesn't provide automatic support for **getters** and **setters**. If you have a field in a struct called **owner**, the getter method should be called **Owner** (upper case, exported), **not** GetOwner.

  A setter function, if needed, will likely be called **SetOwner**.

```
owner := obj.Owner()

 if owner != user {

obj.SetOwner(user)

 }
```

- By convention, one-method interfaces are named by the method name plus an -er suffix: Reader, Writer, Formatter, etc.