Suggestions and Considerations

This lesson lists some useful pieces of advice, tips and things a programmer should consider to program efficiently in Go.

WE'LL COVER THE FOLLOWING

General

Stopping a program in case of error

```
if err != nil {
  fmt.Printf("Program stopping with error %v", err)
  os.Exit(1)
}
```

or:

```
if err != nil {
  panic("ERROR occurred: " + err.Error())
}
```

Performance best practices and advice

- Use `[]rune`, if possible, instead of strings.
- Use slices instead of arrays.
- Use arrays or slices instead of a map where possible.
- Use a `for range` over a slice if you only need the value and not the index; this is slightly faster than having to do a slice lookup for every element.

- When the array is sparse (containing many 0 or nil-values), using a map can result in lower memory consumption.
- Specify an initial capacity for maps.
- When defining methods, use a pointer to a type (struct) as a receiver.
- Use constants or flags to extract constant values from the code.
- Use caching whenever possible when large amounts of memory are being allocated.
- Use template caching.

Memory considerations

For long-lived processes, deploy your Go applications on a **64-bit** OS. Due to the current nature of the GC, big chunks of memory are pre-allocated, which could bring a long-lived Go process on a 32-bit machine to crash due to memory depletion. Other causes of memory leaks could be:

- You keep creating goroutines that don't ever exit.
- You keep appending to the same persistent slice (or writing to the same `bytes.Buffer`) and never resetting it.
- You keep adding items to a persistent map without ever removing them.

Here is a code snippet to monitor memory usage:

```
memstats = new(runtime.MemStats)
runtime.ReadMemStats(memstats)
log.Printf("memstats before GC: bytes = %d footprint = %d",
memstats.HeapAlloc, memstats.Sys)
```

After learning all basic and advanced concepts of programming in Golang, how about making a web application of our own, keeping in view all the concepts and suggestions? The next chapter is about building a complete application.