Go

Ben Gavan

August 23, 2020

Contents

Abstract

$$\int_{1}^{-1} dx \int_{1}^{-1} dy f(x,y) \tag{1}$$

$$D_{it} = \begin{cases} 1 & \text{if bank } i \text{ issues ABs at time } t \\ 2 & \text{if bank } i \text{ issues CBs at time } t \\ 0 & \text{otherwise} \leq \end{cases} \tag{2}$$

$$I = \prod_{i=1}^{n} \int_{-r}^{r} dx_i f(x_1, ..., x_n) \tag{3}$$

$$f(x,y) = \begin{cases} 1 & \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}} \leq r \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

1 Advice

- Never User Global Variables

2 TODO

- *request.FormValue("KEY")*
- *request.FormFile("KEY")*

3 Goland Keyboard Short-cuts

3.1 Format File

sbift + option + command + f

- Format File

  sbift + option + command + f

4 fmt

4.1 fmt.printf()

- %T - prints the type of the data

4.2 fmt.Sprintf(...,...)

float to string with specifying the number of decimal places.

```go
s := fmt.Sprintf("%.2f", 12.3456) // s == "12.35"
```

5 byte

The type of *byte* is 'an alias for *uint8* an is equivalent in all ways'. 'It is used, by convention, to distinguish byte values from 8-bit unsigned integer values'.

```go
// byte is an alias for uint8 and is equivalent to uint8 in all
    ways. It is
// used, by convention, to distinguish byte values from 8-bit
    unsigned
// integer values.
type byte = uint8
```

[1]

6 Interface Type Assertion

```go
s, ok := v.(string)
if !ok {
  // the assertion failed.
}

// OR //

switch t := v.(type) {
case string:
  // t is a string
case int :
  // t is an int
default:
  // t is some other type that we didn't name.
}
```

7 Slice

7.1 append slice to slice

```go
var ts []byte
var exs []byte
ts = append(ts, exs...)
```

7.2 append multiple elements to slice

```go
var ts []byte
var exs []byte
ts = append(ts, exs...)
ts = append(ts, 1, 3, 3)
```

7.3 Special case: append string to bytes slice

As a special case, it is legal to append a string to a byte slice, like this:

```go
slice = append([]byte("hello "), "world"...)
```

```go
var ts []byte
var exs []byte
ts = append(ts, exs...)
ts = append(ts, 1, 3, 3)
ts = append(ts, "string"...)
```

8 Networking

8.1 Creating a HTTP Server

```

  content...
```

8.2 Routing

To route traffic to a specific path, emit the final forward slash, such that

```
  "base/path/specific"
```

To route traffic from all sub-routes of a base route (excluding specific registered sub routes), include the last forward slash, such that,

```
  "/base/path/all/"
```

8.3 Cookies

8.3.1 Creation

To create a cookie, use the *http.Cookie* type to create it. To then write to the *Set-Cookie* HTTP header, use the *http.SetCookie(w ResponseWriter, cookie *Cookie)* method.
*** Make sure the header is set BEFORE any response is written ***
Cookies could be silently dropped. (*SetCookie(...)* does not return an error)

Requirements for Cookie name:

- name can NOT have spaces in the name. (the value can, though (any data can be stored))

```
expires := time.Now().AddDate(1, 0, 0) // Expires one year from now
c := http.Cookie{
  Name:    name,
  Value:   value,
  MaxAge:  360000,
  Expires: expires,
}
http.SetCookie(w, &c)
```

8.3.2 Changing the value for a given name

To change the value of a previously stored cookie, just create a new cookie and save it as the same name - (it will override the old value with the new value)

8.3.3 Get Value

```
  c, err := r.Cookie(name)
  value := c.Value
```

8.4 Deletion

```
c := http.Cookie{
  Name:   name,
  MaxAge: -1,
}
http.SetCookie(w, &c)
```

9 Neo4j

9.1 Delete all nodes & relationships

```
MATCH (n) DETACH DELETE n;
```

All relationships attached to a node need to be detached (deleted) before a node is deleted.

10 Files

10.1 Write to file

10.2 Append to file

```
  content...
```

11 Logging

12 Logging to file

```
f, err := os.OpenFile("text.log", os.O_APPEND | os.O_CREATE | os.
    O_WRONLY, 0644)
if err != nil {
  log.Println(err)
}
defer f.Close()

logger := log.New(f, "prefix", log.LstdFlags)
logger.Println("text to append")
logger.Println("more text to append")
```

[2]

13 Regex

```
  matched, err := regexp.MatchString(`a.b`, "aaxbb")
  fmt.Println(matched) // true
  fmt.Println(err)     // nil (regexp is valid)
```

14 Testing

14.1 Run Test

To run a test, navigate to the package directory you want to test, then type and run:

```
go test -v
```

For more tips and tricks see [3]

14.2 Check coverage of tests

```
go test -coverage
```

References

[1] *builtin.go* line 88

[2] *https://yourbasic.org/golang/log-to-file/*
Accessed 16/04/2020

[3] *https://medium.com/@matryer/5-simple-tips-and-tricks-for-writing-unit-tests-in-golang-619653f90742*
Accessed 11/05/2020