Programming in Go

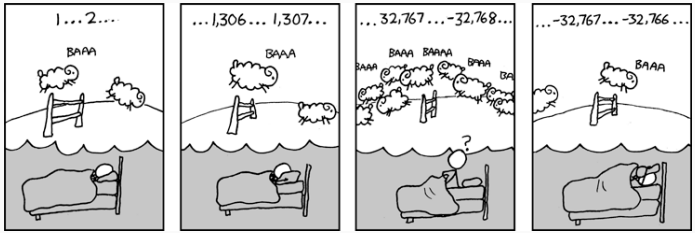
Matt Holiday
Christmas 2020

Homework

Homework #3

Exercise 4.12 from *GOPL:* fetching from the web

The popular web comic “xkcd” has a JSON interface. For example, a request to https://xkcd.com/571/info.0.json produces a detailed description of comic 571, one of many favorites. Download each URL (once!) and build an offline index. Write a tool xkcd that, using this index, prints the *URL, date, and title* of each comic whose *title or transcript* matches a *list of search terms* provided on the command line.



Homework #3

What the raw data looks like:


```
{
 "month":      "4",
 "num":        571,
 "link":       "",
 "year":       "2009",
 . . .
 "transcript": "[[Someone is in bed, . . . long int.",
 "img":        "https://imgs.xkcd.com/comics/cant_sleep.png",
 "title":      "Can't Sleep",
 "day":        "20"
}
```

Homework #3

Sample output:

```
$ go run xkcd-load.go xkcd.json
skipping 404: got 404
skipping 2403: got 404
skipping 2404: got 404
read 2401 comics

$ go run xkcd-find.go xkcd.json someone bed sleep
read 2318 comics
https://xkcd.com/571/ 4/20/2009  "Can't Sleep"
found 1 comics
```

Find the solution at: matt4biz/go-class-exer-4.12

First program: downloader

The first program downloads comic metadata

1. We will read until we get two 404 responses in a row
2. Each request will generate a JSON object as a string
3. We will bracket them with `[` and `]` and a comma between
   (so no comma before the first object)
4. The result will be a file with a JSON array of metadata objects,
   so we won't need to decode
5. We will optionally take a filename from the command line for output

First program: downloader

```
package main

import (
    "bytes"
    "fmt"
    "io"
    "io/ioutil"
    "net/http"
    "os"
)
```

First program: downloader

This function gets the data for a single comic strip

```
func getOne(i int) []byte {
    url := fmt.Sprintf("https://xkcd.com/%d/info.0.json", i)
    resp, err := http.Get(url)

    if err != nil {
        fmt.Fprintf(os.Stderr, "stopped reading: %s\n", err)
        os.Exit(-1)
    }

    defer resp.Body.Close()
```

First program: downloader

```
    if resp.StatusCode != http.StatusOK {
        // easter egg: #404 returns HTTP 404 - not found

        fmt.Fprintf(os.Stderr, "skipping %d: got %d\n",
                    i, resp.StatusCode)
        return nil
    }

    body, err := ioutil.ReadAll(resp.Body)

    if err != nil {
        fmt.Fprintf(os.Stderr, "bad body: %s\n", err)
        os.Exit(-1)
    }

    return body
}
```

First program: downloader

```
func main() {
    var (
        output     io.WriteCloser = os.Stdout
        err        error
        cnt, fails int
        data       []byte
    )

    if len(os.Args) > 0 {
        output, err = os.Create(os.Args[1])

        if err != nil {
            fmt.Fprintln(os.Stderr, err)
            os.Exit(-1)
        }

        defer output.Close()
    }
```

First program: downloader

```
// the output will be in the form of a JSON array,
// so add the brackets before and after

fmt.Fprint(output, "[")
defer fmt.Fprint(output, "]")

for i := 1; fails < 2; i++ { // stop on 2 404s in a row
    if data = getOne(i); data == nil {
        fails++
        continue
    }

    if cnt > 0 {
        fmt.Fprint(output, ",") // OB1
    }
    . . .
```

First program: downloader

```
        . . .

        _, err = io.Copy(output, bytes.NewBuffer(data))

        if err != nil {
            fmt.Fprintf(os.Stderr, "stopped: %s", err)
            os.Exit(-1)
        }

        fails = 0
        cnt++
    }

    fmt.Fprintf(os.Stderr, "read %d comics\n", cnt)

    // $ go run ./xkcd-load.go xkcd.json
    // read 2318 comics
}
```

Second program: searcher

The second program reads in and searches all comics

1. We will require a filename from the command line for input
2. We will read in and decode to a slice of objects
3. We will take multiple search terms from the command line
4. We will select comics that match all words by doing a quadratic search (nested loops)
5. We will compare all strings in lower case

Second program: searcher

```
// { "month":      "4",
//   "num":        571,
//   "year":       "2009",
//   "transcript": "[[Someone is in bed, . . . long int.",
//   "img":        "https://imgs.xkcd.com/comics/cant_sleep.png",
//   "title":      "Can't Sleep",
//   "day":        "20"
// }

type xkcd struct {
    Num        int    `json:"num"`
    Day        string `json:"day"`
    Month      string `json:"month"`
    Year       string `json:"year"`
    Title      string `json:"title"`
    Transcript string `json:"transcript"`
}
```

Second program: searcher

```
func main() {
    if len(os.Args) < 2 {
        fmt.Fprintln(os.Stderr, "no file given")
        os.Exit(-1)
    }

    fn := os.Args[1]

    if len(os.Args) < 3 {
        fmt.Fprintln(os.Stderr, "no search term")
        os.Exit(-1)
    }
```

Second program: searcher

```
var items []xkcd
var terms []string
var input io.ReadCloser
var cnt   int
var err   error

if input, err = os.Open(fn); err != nil {
    fmt.Fprintf(os.Stderr, "invalid file: %s", err)
    os.Exit(-1)
}

// read the file in one big step
err = json.NewDecoder(input).Decode(&items)

if err != nil {
    fmt.Fprintf(os.Stderr, "decode failed: %s\n", err)
    os.Exit(-1)
}
```

Second program: searcher

```
    fmt.Fprintf(os.Stderr, "read %d comics\n", len(items))

    for _, t := range os.Args[2:] {
        terms = append(terms, strings.ToLower(t))
    }

outer:
    for _, item := range items {
        title := strings.ToLower(item.Title)
        transcript := strings.ToLower(item.Transcript)

        for _, term := range terms {
            if !strings.Contains(title, term) &&
                !strings.Contains(transcript, term) {
                continue outer
            }
        }
        . . .
```

Second program: searcher

```
        . . .

        fmt.Printf("https://xkcd.com/%d/ %s/%s/%s\n",
            item.Num, item.Month, item.Day, item.Year)
        cnt++
    }

    fmt.Fprintf(os.Stderr, "found %d comics\n", cnt)

    // $ go run ./xkcd-find.go xkcd.json someone bed sleep
    // read 2318 comics
    // https://xkcd.com/571/ 4/20/2009
    // found 1 comics
}
```