Exercise 3

CST8333 19F

Professor: Stanley Pieda

By: Lucas Estienne (esti0011)

040 819 959

## Evidence of Learning

* 1. Variables:

// declare variables

var records []Record

numRecords := 5 // number of records to get

* 1. Loops:

// Get subslice of lines slice, not including column names

for i := 0; i < numRecords; i++ {

   // append a Record object to Record slice

   records = append(records, Record { strings.Join(lines[i][:], ",") })

}

// loop through records object

for \_, v := range(records) {

    fmt.Println(v.content) // print row

}

* 1. File I/O & functions:

// helper function to read CSV

func getLinesFromCSV(filePath string) (lines [][]string, err error) {

    // open file

    file, err := os.Open(filePath)

    check(err)

    defer file.Close() // defer closing the file until function returns

    // create CSV Reader from file

    reader := csv.NewReader(file)

    return reader.ReadAll()

}

* 1. Simple Data Structure (Slice):

// simple data structure containing a string

type Record struct {

    content string

}

## Research

1. Memory management

Go is a language which performs automatic memory allocation and automatic garbage collection. There aren’t any memory consumption concerns that aren’t caused by programmer error (i.e. not closing files, loading ridiculous amounts of data into memory, etc.)

[1]"Memory Blocks - Go 101 (an online book for Go programming language + Golang knowledge base)", *Go101.org*, 2019. [Online]. Available: https://go101.org/article/memory-block.html. [Accessed: 16- Sep- 2019].

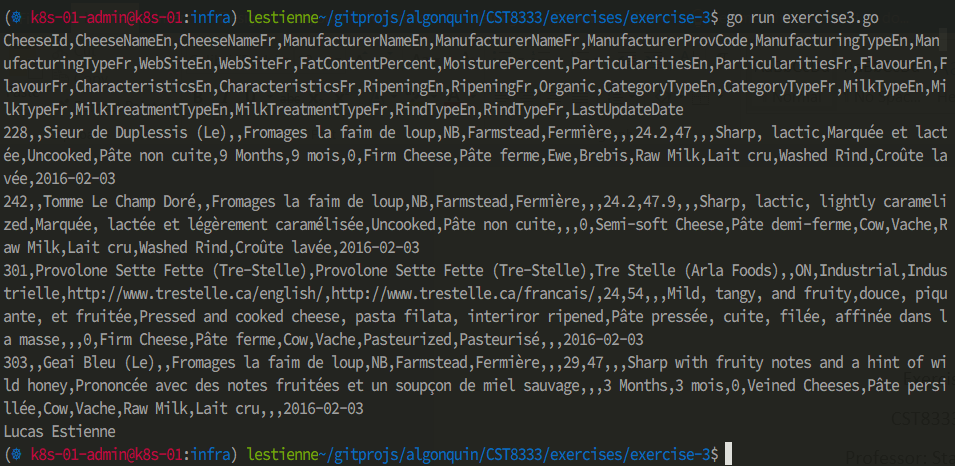
1. Unit Testing

Unit Testing in Go is done using the standard “testing” package. Besides mocking, it’s fully featured.

[2]"testing - The Go Programming Language", *Golang.org*, 2019. [Online]. Available: https://golang.org/pkg/testing/. [Accessed: 16- Sep- 2019].

[3]A. Ellis, "Golang basics - writing unit tests", *alex ellis' blog*, 2017. [Online]. Available: https://blog.alexellis.io/golang-writing-unit-tests/. [Accessed: 16- Sep- 2019].

## Program Demonstration via Screen Shot



## Source Code

// CST8333 Exercise 3 - Lucas Estienne

package main

import (

    "fmt"

    "os"

    "log"

    "strings"

    "encoding/csv"

)

// simple data structure containing a string

type Record struct {

    content string

}

// main function, this is the entrypoint

func main() {

    // declare variables

    var records []Record

    numRecords := 5 // number of records to get

    // Load lines from CSV

    lines, err := getLinesFromCSV("data/canadianCheeseDirectory.csv")

    check(err)

    // Get subslice of lines slice, not including column names

    for i := 0; i < numRecords; i++ {

        // append a Record object to Record slice

        records = append(records, Record { strings.Join(lines[i][:], ",") })

    }

    // loop through records object

    for \_, v := range(records) {

        fmt.Println(v.content) // print record content

    }

    fmt.Println("Lucas Estienne")

}

// helper function to do error handling

func check(e error) {

    if e != nil {

        log.Fatal("Error", e)

        panic(e)

    }

}

// helper function to read CSV

func getLinesFromCSV(filePath string) (lines [][]string, err error) {

    // open file

    file, err := os.Open(filePath)

    check(err)

    defer file.Close() // defer closing the file until function returns

    // create CSV Reader from file

    reader := csv.NewReader(file)

    return reader.ReadAll()

}