Go/Golang References

Important Packages:

*Note: packages can be imported into your program using import “packageName”

- fmt – implements formatted I/O with functions analogous to C’s printf and scanf.
    - fmt.Println(x)
    - fmt.Printf(“%v”, x)
        - %v – Value
        - %T – Type
        - %d – Decimal
        - %t – Boolean
        - %% – Percent Symbol
- math – implements several math related functions and constants.
    - math constants
        - Pi
        - E
        - Phi

Basic Syntax:

```
package main ← (all programs are named a package, code runs in the package named “main”)

import ( ← (additional packages can be imported by using import, you can do this one by one,
        “fmt”                                   or list them under parenthesis like this instance.
)                                                    Variables can be defined this way too.)

func add() int {
        if x > 5 {
                for i := 0; i < 2; i++ { ← (unlike C++, not many parenthesis are used in Go)
                        x -= 1
                }
        }
        return x + y ← (methods must return their associated type)
}

func main() { ← (functions are outlined as such: func name(var type, var2 type) returnType { })
        var w string
        var x int = 5 ← (standard format for defining a variable)
        y := 10 ← (this method is simpler, however it gives the compiler the work to guess what
                                                                                type it is.)
        z := add(x, y) ← (standard format for calling a method/function)
        fmt.Println(“%v + %v = %v”, x, y, z) ← (variables can be inserted into Println as such)
}
```

Go/Golang Programming Workshops

1. Making a basic calculator in Go.
   a. Navigate to https://play.golang.org/p/Eiepk5Cw3Kt.
      i. This is the Go Playground. This allows you a full web-based IDE to program these next assignments in. At any point you can click the 'Run' button in the top left to test your code.
   b. Your first program is to create a very basic calculator. This calculator will only have options to add, subtract, and perform exponents.
   c. The add function is provided to you as an idea of what a function looks like in Go. First you will be tasked with making the subtract function. Once this is completed, remove the '//' in front of the test code to test if this method is working correctly.
   d. You will now be manually creating an exponent function for your calculator. This means that you cannot use the built-in exponent feature of Go, but instead will be using this function and a for loop to perform this problem. Your function should take in a variable, and second, the power it will be raised to. After implementing this method, delete the '//' on the third test to make sure you code is working as intended.
      i. You will not need to import any other existing packages than fmt.
   e. Once you are completely done, click the 'Share' button at the top left and record the string of characters after the "/p/".

____

2. Shadowing in Go.
   a. This program will have you walk through steps and record your observations as changes are made to the program. This will make you aware of shadowing and why should you keep an eye on it.
   b. Navigate to https://play.golang.org/p/YRg5YSP15yC.
      i. This is a working version of the previous problem's exponent function, we will be playing with a few things in this program and recording what happens.
   c. Begin by adding a full declaration of x outside of function main, "var x int = 10" for example.
      i. How do you think this will affect the program, will any errors be thrown?

d. Run the code, does anything change?

e. Delete the original declaration of x “x := 3” and run your code. Does the code run as expected?

f. Navigate now to https://play.golang.org/p/W4ncI_9RUA4. I have added a few lines of code into the exponent function. What do you think the first Println will output? What about the second?

g. Run the code and record your observations. Is this what you expected?

3. Calculate π in Go.
    a. Navigate to https://play.golang.org/.
    b. For this final activity, no initial code will be provided.
    c. Using 100,000 iterations of the Gregory-Leibniz series, calculate π. Use math.Pi to compare your solution. Display your answer and the percentage of how close you are to the exact solution of π.
        i. If you are not aware of what the Gregory-Leibniz series is, it would be recommend to research it first (It’s quite simple!).
    d. Once you are finished, click the ‘Share’ button at the top left and record the string of characters after the “/p/”.

____

Answer Key:

Answer to 1: https://play.golang.org/p/nSLbLbaMvvd

```go
package main

import (
	"fmt"
)

func add(x int, y int) int {
	return x + y
}

func subtract(x int, y int) int {
	return x - y
}

func exponent(x int, ex int) int {
	total := x
	for i := 1; i < ex; i++ {
		total = total * x
	}
	return total
}

func main() {
	x := 3
	y := 5

	fmt.Printf("The sum of %v and %v is %v\n", x, y, add(x, y))
	fmt.Printf("The difference of %v and %v is %v\n", x, y, subtract(x, y))
	fmt.Printf("%v to the power of %v is %v", x, y, exponent(x, y))
}
```

Answer to 2:

Run the code, does anything change?

No, the program should run identically and return no errors.

Run the code and record your observations. Is this what you expected?

The redeclaration should be accepted and change value exclusively in that scope.

Answer to 3: https://play.golang.org/p/SDv4EEtu-PX

```
package main

import (
	"fmt"
	"math"
)

func calculatePi() float64 {
	x := 4.0
	y := 1.0
	solution := 0.0
	for i := 0; i < 100000; i++ {
		if i%2 == 0 {
			solution += x/y
		} else {
			solution -= x/y
		}
		y += 2
	}
	return solution
}

func main() {
	estimatedPi := calculatePi()
	percentOff := (1 - estimatedPi/math.Pi) * 100

	fmt.Printf("Our estimated pi is eqaul to %v, this is %v%% off." , estimatedPi, percentOff)
}
```