
ronaldpetty updated syntax-and-flow-control lab 620166a on Sep 8

2 contributors  

169 lines (124 sloc) 3.28 KB

Go

Syntax and Flow Control

In this lab we will get more comfortable writing basic Go code.

1. Variables and Expressions

There are three different ways to declare a variable called x that is equal to 1. What are they?

- 1:
- 2:
- 3:

Create a new folder called lab-syntax (in your `$GOPATH/src`), and a file called `syntax.go`. Write a function called `vars()` that initializes three variables:

- a to "hello"
- b to 3
- c to 7

The vars() function should return all three variables. Have the main() function call vars and print out all of the variables on one line. When you run your program, the output should look like this:

```
user@ubuntu:~/go/src/lab-syntax$ go run syntax.go

a is hello b is 3 and c is 7
user@ubuntu:~/go/src/lab-syntax$
```

How many lines is your vars() function? Recall that you can initialize multiple variables on one line.

2. Conditional and Switch Statements

In the same file, create a new function called `guess1()` that utilizes user input via command line arguments and `if/else` statements to print a different color for each integer between 1 and 5 (inclusive). If the input is out of range, print the string "Enter a number between 1-5".

Write a new function called `guess2()` that does the same thing, but uses a `switch` statement rather than if/else statements. When you run the file, the first command line arg should be used in guess1 and the second should be used in guess2. The "os" package will give you simple access to the command line.

```
user@ubuntu:~/go/src/lab-syntax$ go run syntax.go 1 5

a is hello b is 3 and c is 7
red
blue
user@ubuntu:~/go/src/lab-syntax$
```

```
user@ubuntu:~/go/src/lab-syntax$ go run syntax.go 2 6

a is hello b is 3 and c is 7
orange
Enter a number between 1-5
user@ubuntu:~/go/src/lab-syntax$
```

Which way is easier? Why might you want to use switch statements over if/else statements, and vice versa?

3. Loops

Create a new function called isPrime() that takes an integer and checks if it is prime. Then write a function called primeList() that calls isPrime() and prints all the prime numbers from 0 to 100.

> A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself.

People get their PhDs solving this problem efficiently so don't get crazy.

```
user@ubuntu:~/go/src/lab-syntax$ go run syntax.go 2 6

a is hello b is 3 and c is 7

orange
Enter a number between 1-5

2
3
5
7
11
13
17
19
23
29
31
```

```
37
41
43
47
53
59
61
67
71
73
79
83
89
97
user@ubuntu:~/go/src/lab-syntax$
```

4. Challenge

Write the function `longestIncreasingRun()` that takes in a positive int value n and returns the longest increasing run of digits. For example longestIncreasingRun(1232) would return 123 and longestIncreasingRun(27648923679) returns 23679. Don't worry about handling ties.

```
user@ubuntu:~/go/src/lab-syntax$ go run syntax.go 2 6
a is hello b is 3 and c is 7

orange
Enter a number between 1-5

2
3
5
7
11
13
17
19
23
```

```
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97

longest run ( 1232 ):  123
longest run ( 27648923679 ):  23679
user@ubuntu:~/go/src/lab-syntax$
```

Congratulations you have completed the lab!!