LEARN GO: FUNDAMENTALS

Gopher It

An effective way to learn something new is by making something! In this project we're going to incorporate the use of `fmt` to make ASCII art. One suggestion is to make a gopher:

But feel free to make what you want. In other words, *gopher it*.

Tasks

8/8 Complete

Mark the tasks as complete by checking them off

Basic Go Structure

1.

Every Go program requires a package declaration, add in `package main` at the top of **main.go**.

Hint

At the top of the file add the line:

```
package main
```

2.

Now let's import the packages that we need. In this case, we want to create ASCII art and print it to the terminal. Therefore, we want to import the `"fmt"` package.

Hint

Under the package declaration, use the `import` keyword followed by `"fmt"`. Now we have access to the `fmt` package's functions.

3.

Time to define the `main` function that will later house the code we need to print to the terminal.
Hint
The syntax for the `main` function is as follows:

```
func main () {}
```

You can also add in a line break to insert code inside the function later like:

```
func main () {

}
```

Printing

4.
Inside the main function, we can use functions from the `fmt` package and print to the terminal. We're already familiar with `Println` to print and add a new line each time.

`fmt` also has many other functions that we can explore. Remember, we can:

- directly go to Go's fmt package documentation.
- enter the command `go doc fmt`.
- use a search engine and find out more about `fmt`.

5.
After searching the different functions, let's get to printing stuff!

You can print anything you want to the terminal, from messages to ASCII art:

```
 __      _
o'')}____//
 `_/      )
 (_(_/-(_/
```

Hint
If you want to print the dog above call `fmt.Println` for each following line:

```
"          "
"o'')}____//"
" `_/      )"
" (_(_/-(_/ "
```

For example, the first line will be:

```
fmt.Println("  __      _")
```

6.
If you want to re-create the gopher and need the template, check the hint!

Hint
To see the complete gopher, add each line in their own individual `fmt.Println()` statement:

7.
Click the **Save** button to save your code.

After saving, build an executable file and run that file to print out your ASCII art.

Or directly run the file, you decide!
Hint
Use the command `go build main.go` to build an executable file named **main**.

Then, to run the executable type in `./main` in the command line.

Another way to see the printed ASCII is to use the command `go run main.go`.
8.
Great job creating a Go program from scratch!

If you want to challenge yourself and learn more about Go:

- Change the design of your ASCII art.
- Add the time to show when your art was printed.
    - Use the `time` package.
    - Explore different `time` functions.
- Explore different `fmt` functions (using `go doc`) and see what else you can print/do.

main.go

```
package main

import "fmt"

func main() {
  fmt.Println("    `.-::::::-.`    ")
  fmt.Println(".:-:::::::::::::::-:.")
  fmt.Println("`_:::    ::    :::_`")
  fmt.Println(" .:( ^   :: ^   ):. ")
  fmt.Println(" `:::   (..)   :::. ")
  fmt.Println(" `:::::::UU:::::::` ")
  fmt.Println(" .:::::::::::::::::. ")
  fmt.Println(" O:::::::::::::::::O ")
  fmt.Println(" -:::::::::::::::::- ")
  fmt.Println(" `:::::::::::::::::` ")
  fmt.Println("  .:::::::::::::::.  ")
  fmt.Println("    oO:::::::Oo     ")
}
```

```
$ go run main.go
    `.-::::::-.`
.:-:::::::::::::::-:.
`_:::    ::    :::_`
 .:( ^   :: ^   ):.
 `:::   (..)   :::.
 `:::::::UU:::::::`
 .:::::::::::::::::.
 O:::::::::::::::::O
 -:::::::::::::::::-
 `:::::::::::::::::`
  .:::::::::::::::.
    oO:::::::Oo
```