Return a random greeting

In this section, you'll change your code so that instead of returning a single greeting every time, it returns one of several predefined greeting messages.

Note: This topic is part of a multi-part tutorial that begins with Create a Go module.

To do this, you'll use a Go slice. A slice is like an array, except that its size changes dynamically as you add and remove items. The slice is one of Go's most useful types.

You'll add a small slice to contain three greeting messages, then have your code return one of the messages randomly. For more on slices, see Go slices in the Go blog.

1. In greetings/greetings.go, change your code so it looks like the following.

```
package greetings

import (
    "errors"
    "fmt"
    "math/rand"
    "time"
)

// Hello returns a greeting for the named person.
func Hello(name string) (string, error) {
    // If no name was given, return an error with a message.
    if name == "" {
        return name, errors.New("empty name")
    }
    // Create a message using a random format.
    message := fmt.Sprintf(randomFormat(), name)
    return message, nil
}

// init sets initial values for variables used in the function.
func init() {
    rand.Seed(time.Now().UnixNano())
}

// randomFormat returns one of a set of greeting messages. The returned
// message is selected at random.
func randomFormat() string {
    // A slice of message formats.
    formats := []string{
        "Hi, %v. Welcome!",
        "Great to see you, %v!",
        "Hail, %v! Well met!",
    }

    // Return a randomly selected message format by specifying
    // a random index for the slice of formats.
    return formats[rand.Intn(len(formats))]
}
```

   In this code, you:

   - Add a `randomFormat` function that returns a randomly selected format for a greeting message. Note that `randomFormat` starts with a lowercase letter, making it accessible only to code in its own package (in other words, it's not exported).
   - In `randomFormat`, declare a `formats` slice with three message formats. When declaring a slice, you omit its size in the brackets, like this: `[]string`. This tells Go that the size of the array underlying the slice can be dynamically changed.
   - Use the `math/rand` package to generate a random number for selecting an item from the slice.
   - Add an `init` function to seed the `rand` package with the current time. Go executes `init` functions automatically at program startup, after global variables have been initialized. For more about `init` functions, see Effective Go.
   - In `Hello`, call the `randomFormat` function to get a format for the message you'll return, then use the format and `name` value together to create the message.
   - Return the message (or an error) as you did before.

2. In hello/hello.go, change your code so it looks like the following.

   You're just adding Gladys's name (or a different name, if you like) as an argument to the `Hello` function call in hello.go.

```
package main

import (
    "fmt"
    "log"

    "example.com/greetings"
)

func main() {
    // Set properties of the predefined Logger, including
    // the log entry prefix and a flag to disable printing
    // the time, source file, and line number.
    log.SetPrefix("greetings: ")
    log.SetFlags(0)

    // Request a greeting message.
    message, err := greetings.Hello("Gladys")
    // If an error was returned, print it to the console and
    // exit the program.
    if err != nil {
        log.Fatal(err)
    }

    // If no error was returned, print the returned message
    // to the console.
    fmt.Println(message)
}
```

3. At the command line, in the hello directory, run hello.go to confirm that the code works. Run it multiple times, noticing that the greeting changes.

```
$ go run .
Great to see you, Gladys!

$ go run .
Hi, Gladys. Welcome!

$ go run .
Hail, Gladys! Well met!
```

Next, you'll use a slice to greet multiple people.

< Return and handle an error

Return greetings for multiple people >