The regexp Package

This lesson provides information about the regexp package and the functionalities it provides.

WE'LL COVER THE FOLLOWING

- Overview
- Explanation

Overview

Package `regexp` implements regular expression search. For information about regular expressions and their syntax, visit this page.

Explanation

```
package main
import (
  "fmt"
  "regexp"
  "strconv"
)

func main() {
  searchIn := "John: 2578.34 William: 4567.23 Steve: 5632.18" // string to search
  pat := "[0-9]+.[0-9]+" // pattern search in searchIn

  f := func (s string) string {
    v, _ := strconv.ParseFloat(s, 32)
    return strconv.FormatFloat(v * 2, 'f', 2, 32)
  }
  if ok, _ := regexp.Match(pat, []byte(searchIn)); ok {
    fmt.Println("Match found!")
  }
  re, _ := regexp.Compile(pat)

  str := re.ReplaceAllString(searchIn, "##.#") // replace pat with "##.#"
  fmt.Println(str)
  // using a function
  str2 := re.ReplaceAllStringFunc(searchIn, f)
  fmt.Println(str2)
}
```

regexp Package

In the code above, outside `main` at **line 4**, we import the package `regexp`. We want to search a string pattern `pat` (declared in `main` at **line 10**) in a string `searchIn` (declared in `main` at **line 9**). At **line 12**, we have a function `f`. It is taking a string `s`, formatting it in *float32*, and returning the formatted string.

Testing if the pattern occurs is easy, use the function `Match` as: `ok, _ := regexp.Match(pat,[]byte(searchIn))` (see **line 16**); where `ok` will be *true* or *false*. If `ok` is true, it means the match was found, so **Match found!** will be printed on the screen.

For more functionalities, you must first make a (pointer to a) Regexp object from the pattern; this is done through the `Compile` function (see **line 19**). Then we have at our disposal a whole number of `Match`, `Find` and `Replace` functions. At **line 21**, we replace the strings of pattern `pat` in `searchIn` with ##.#, and store the updated string in `str`. In the next line, we are printing `str` to see how `ReplaceAllString` of the package `regexp` works.

At **line 24**, we are again replacing some part of string `searchIn`, but this time through function `f`. We store the result in `str2`. This means `str2` will be a formatted string in *float32*. In the next line, we are printing `str` to see the changes.

The `Compile` function also returns an error, which we have safely ignored here because we have entered the pattern ourselves and know that it is a valid regular expression. Should the expression be entered by the user or taken from a data source, it is necessary to check this parsing error.

In this example, we could also have used the function `MustCompile`, which is like `Compile` but panics (stopping the program with an error message when the pattern is not a valid regular expression.

That's it about `regexp` package and its methods. Now you'll study another package `sync` in the next lesson.