Go Piscine

Go 01

Summary: THIS document is the subject for the Go 01 module of the Go Piscine @ 42Tokyo.

Contents

Chapter I

Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for every exercise.
- Your exercises will be checked and graded by your fellow classmates.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / man / the Internet / ....`
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- If no other explicit information is displayed, you must use the latest versions of Go.
- Your turn-in directory for each exercise should look something like this:

```
ex[XX]
|-- main.go
|-- vendor
    |-- ft
        |-- printrune.go
    |-- piscine
        |-- [excercisename].go
```

Chapter II

Exercise 00 : pointone

	Exercise 00
pointone	
Turn-in directory : *ex*00/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that takes a pointer to an int as argument and gives to this int the value of 1.

- Expected function

```
func PointOne(nb *int) {

}
```

- Usage

```
package main

import (
        "fmt"
        "piscine"
)

func main() {
        n := 0
        piscine.PointOne(&n)
        fmt.Println(n)
}
```

- Output of usage

```
$ go mod init ex00
$ go run .
1
$
```

Chapter III

Exercise 01 : ultimatepointone

	Exercise 01
ultimatepointone	
Turn-in directory : *ex*01/	
Files to turn in : *	
Allowed packages : fmt	
Allowed builtin functions : None	

Write a function that takes a pointer to a pointer to a pointer to an int as argument and gives to this int the value of 1.

- Expected function

```go
func UltimatePointOne(n ***int) {

}
```

- Usage

```go
package main

import (
        "fmt"
        "piscine"
)

func main() {
        a := 0
        b := &a
        n := &b
        piscine.UltimatePointOne(&n)
        fmt.Println(a)
}
```

- Output of usage

```
$ go mod init ex01
$ go run .
1
$
```

Chapter IV

Exercise 02 : divmod

	Exercise 02
divmod	
Turn-in directory : *ex*02/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that does the following:

- This function will divide an int a by another int b.
- The result of this division will be stored in the int pointed by div.
- The remainder of this division will be stored in the int pointed by mod.
- Expected function

```
func DivMod(a int, b int, div *int, mod *int) {

}
```

- Usage

```go
package main

import (
        "fmt"
        "piscine"
)

func main() {
        a := 13
        b := 2
        var div int
        var mod int
        piscine.DivMod(a, b, &div, &mod)
        fmt.Println(div)
        fmt.Println(mod)
}
```

- Output of usage

```
$ go mod init ex02
$ go run .
6
1
$
```

Chapter V

Exercise 03 : ultimatedivmod

	Exercise 03
ultimatedivmod	
Turn-in directory : *ex*03/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that does the following:

- This function will divide an int a by another int b.
- The result of this division will be stored in the int pointed by a.
- The remainder of this division will be stored in the int pointed by b.
- Expected function

```
func UltimateDivMod(a *int, b *int) {

}
```

- Usage

```go
package main

import (
	"fmt"
	"piscine"
)

func main() {
	a := 13
	b := 2
	piscine.UltimateDivMod(&a, &b)
	fmt.Println(a)
	fmt.Println(b)
}
```

- Output of usage

```
$ go mod init ex03
$ go run .
6
1
$
```

Chapter VI

Exercise 04 : printstr

	Exercise 04
printstr	
Turn-in directory : *ex*04/	
Files to turn in : `*`	
Allowed packages : `None`	
Allowed builtin functions : `None`	

Write a function that prints one by one the characters of a string on the screen.

- Expected function

```go
func PrintStr(s string) {

}
```

- Usage

```go
package main

import "piscine"

func main() {
	piscine.PrintStr("Hello World!")
}
```

- Output of usage

```
$ go mod init ex04
$ go run . | cat -e
Hello World!%
$
```

Chapter VII

Exercise 05 : strlen

	Exercise 05
strlen	
Turn-in directory : *ex*05/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that counts the runes of a string and that returns that count.

- Expected function

```
func StrLen(s string) int {

}
```

- Usage

```
package main

import (
        "fmt"
        "piscine"
)

func main() {
        l := piscine.StrLen("Hello World!")
        fmt.Println(l)
}
```

- Output of usage

```
$ go mod init ex05
$ go run .
12
$
```

Chapter VIII

Exercise 06 : swap

	Exercise 06
swap	
Turn-in directory : *ex*06/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that takes two pointers to an int (*int) and swaps their contents.

- Expected function

```
func Swap(a *int, b *int) {

}
```

- Usage

```
package main

import (
        "fmt"
        "piscine"
)

func main() {
        a := 0
        b := 1
        piscine.Swap(&a, &b)
        fmt.Println(a)
        fmt.Println(b)
}
```

- Output of usage

```
$ go mod init ex06
$ go run .
1
0
$
```

Chapter IX

Exercise 07 : strrev

	Exercise 07
strrev	
Turn-in directory : *ex*07/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that takes a string and returns that string reversed.

- Expected function

```
func StrRev(s string) string {

}
```

- Usage

```
package main

import (
        "fmt"
        "piscine"
)

func main() {
        s := "Hello World!"
        s = piscine.StrRev(s)
        fmt.Println(s)
}
```

- Output of usage

```
$ go mod init ex07
$ go run .
!dlroW olleH
$
```

Chapter X

Exercise 08 : basicatoi

	Exercise 08
basicatoi	
Turn-in directory : *ex*08/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that simulates the behaviour of the Atoi function in Go. Atoi transforms a number defined as a string in a number defined as an int.

- Atoi returns 0 if the string is not considered as a valid number. For this exercise only valid string will be tested. They will only contain one or several digits as characters.
- For this exercise, the handling of the signs + or - does not have to be taken into account.
- This function will only have to return the int. For this exercise, the error return of Atoi is not required.
- Expected function

```go
func BasicAtoi(s string) int {

}
```

- Usage

```go
package main

import (
        "fmt"
        "piscine"
)

func main() {
        fmt.Println(piscine.BasicAtoi("12345"))
        fmt.Println(piscine.BasicAtoi("0000000012345"))
        fmt.Println(piscine.BasicAtoi("000000"))
}
```

- Output of usage

```
$ go mod init ex08
$ go run .
12345
12345
0
$
```

Chapter XI

Exercise 09 : basicatoi2

	Exercise 09
basicatoi2	
Turn-in directory : *ex*09/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that simulates the behaviour of the Atoi function in Go. Atoi transforms a number defined as a string in a number defined as an int.

- Atoi returns 0 if the string is not considered as a valid number. For this exercise non-valid string chains will be tested. Some will contain non-digits characters.
- For this exercise the handling of the signs + or - does not have to be taken into account.
- This function will only have to return the int. For this exercise the error return of Atoi is not required.
- Expected function

```
func BasicAtoi2(s string) int {

}
```

- Usage

```
package main

import (
        "fmt"
        "piscine"
)

func main() {
        fmt.Println(piscine.BasicAtoi2("12345"))
        fmt.Println(piscine.BasicAtoi2("0000000012345"))
        fmt.Println(piscine.BasicAtoi2("012 345"))
        fmt.Println(piscine.BasicAtoi2("Hello World!"))
}
```

- Output of usage

```
$ go mod init ex09
$ go run .
12345
12345
0
0
$
```

Chapter XII

Exercise 10 : atoi

	Exercise 10
atoi	
Turn-in directory : *ex*10/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that simulates the behaviour of the Atoi function in Go. Atoi transforms a number represented as a string in a number represented as an int.

- Atoi returns 0 if the string is not considered as a valid number. For this exercise non-valid string chains will be tested. Some will contain non-digits characters.
- For this exercise the handling of the signs + or - does have to be taken into account.
- This function will only have to return the int. For this exercise the error result of Atoi is not required.
- Expected function

```
func Atoi(s string) int {

}
```

- Usage

```go
package main

import (
        "fmt"
        "piscine"
)

func main() {
        fmt.Println(piscine.Atoi("12345"))
        fmt.Println(piscine.Atoi("0000000012345"))
        fmt.Println(piscine.Atoi("012 345"))
        fmt.Println(piscine.Atoi("Hello World!"))
        fmt.Println(piscine.Atoi("+1234"))
        fmt.Println(piscine.Atoi("-1234"))
        fmt.Println(piscine.Atoi("++1234"))
        fmt.Println(piscine.Atoi("--1234"))
}
```

- Output of usage

```
$ go mod init ex10
$ go run .
12345
12345
0
0
1234
-1234
0
0
$
```

Chapter XIII

Exercise 11 : sortintegertable

	Exercise 11
sortintegertable	
Turn-in directory : *ex*11/	
Files to turn in : `*`	
Allowed packages : `fmt`	
Allowed builtin functions : `None`	

Write a function that reorders a slice of int in ascending order.

- Expected function

```
func SortIntegerTable(table []int) {

}
```

- Usage

```
package main

import (
        "fmt"
        "piscine"
)

func main() {
        s := []int{5,4,3,2,1,0}
        piscine.SortIntegerTable(s)
        fmt.Println(s)
}
```

- Output of usage

```
$ go mod init ex11
$ go run .
[0 1 2 3 4 5]
$
```