

Go Piscine Go 02

Summary: THIS document is the subject for the Go 02 module of the Go Piscine @ 42Tokyo.

# Contents

Ι	Instructions	2
II	Exercise 00: iterative factorial	3
III	Exercise 01 : recursive factorial	4
IV	Exercice 02: iterativepower	6
V	Exercice 03: recursivepower	7
VI	Exercise 04 : fibonacci	9
VII	Exercise 05 : sqrt	11
VIII	Exercise 06 : isprime	12
IX	Exercise 07: findnextprime	14
X	Exercise 08: eightqueens	16

#### 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 <u>cannot</u> leave <u>any</u> 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: iterative factorial

	Exercise 00	
	iterativefactorial	
Turn-in directory : $ex00/$		
Files to turn in : *		
Allowed packages : fmt		
Allowed builtin functions : Non	le	/

Write an iterative function that returns the factorial of the int passed as parameter.

- Errors (non possible values or overflows) will return 0.
- Expected function

```
func IterativeFactorial(nb int) int {
}
```

• Usage

```
package main

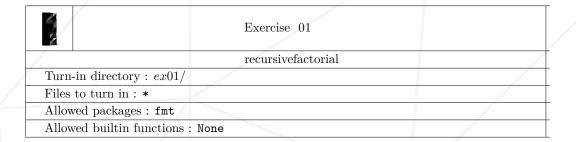
import (
        "fmt"
        "piscine"
)

func main() {
        arg := 4
        fmt.Println(piscine.IterativeFactorial(arg))
}
```

```
$ go mod init ex00
$ go run .
24
$
```

## Chapter III

#### Exercise 01: recursive factorial



Write a recursive function that returns the factorial of the int passed as parameter.

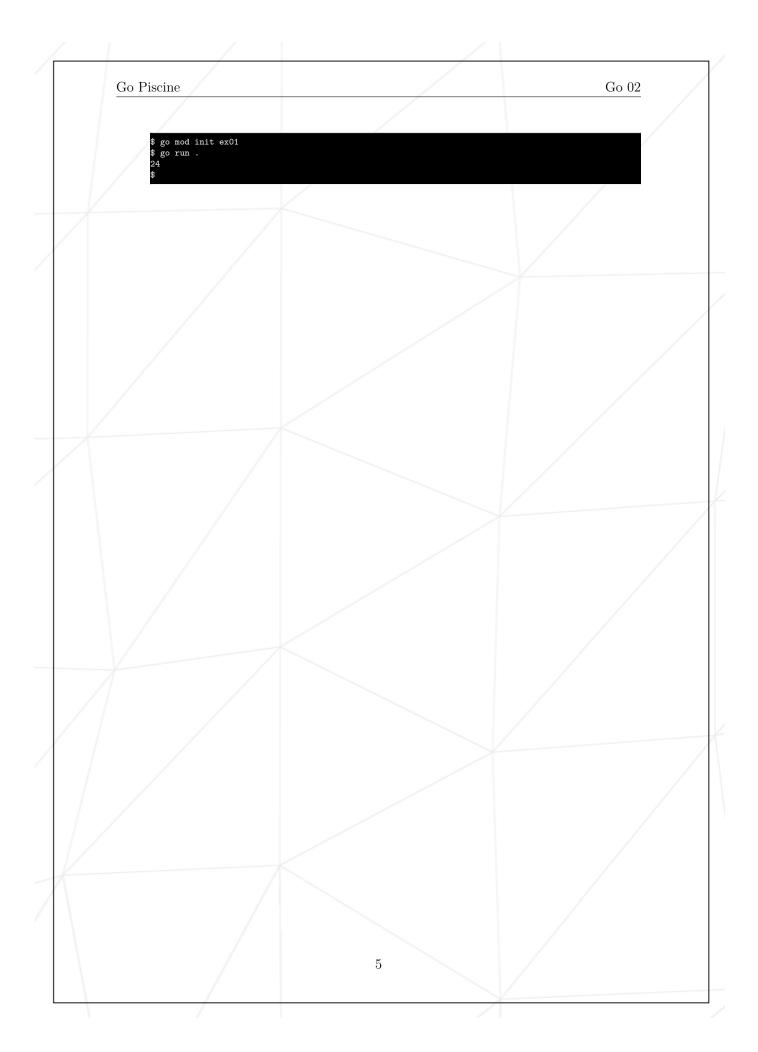
- Errors (non possible values or overflows) will return 0.
- for is forbidden for this exercise.
- Expected function

```
func RecursiveFactorial(nb int) int {
}
```

 $\bullet$  Usage

```
package main
import (
     "fmt"
     "piscine"
)
func main() {
     arg := 4
     fmt.Println(piscine.RecursiveFactorial(arg))
}
```

 $\bullet\,$  Output of usage



## Chapter IV

## Exercice 02: iterativepower

	Exercise 02	
	iterativepower	
Turn-in directory : $ex02/$		
Files to turn in : *		
Allowed packages: fmt		
Allowed builtin functions: 1	Vone	

Write an iterative function that returns the value of nb to the power of power.

- Negative powers will return 0. Overflows do not have to be dealt with.
- Expected function

```
func IterativePower(nb int, power int) int {
}
```

• Usage

```
package main
import (
     "fmt"
     "piscine"
)

func main() {
     fmt.Println(piscine.IterativePower(4, 3))
}
```

```
$ go mod init ex02
$ go run .
64
$
```

# Chapter V

## Exercice 03: recursivepower

	Exercise 03	
	recursivepower	
Turn-in directory : $ex03/$		
Files to turn in : *		
Allowed packages: fmt		
Allowed builtin functions	: None	

Write an iterative function that returns the value of nb to the power of power.

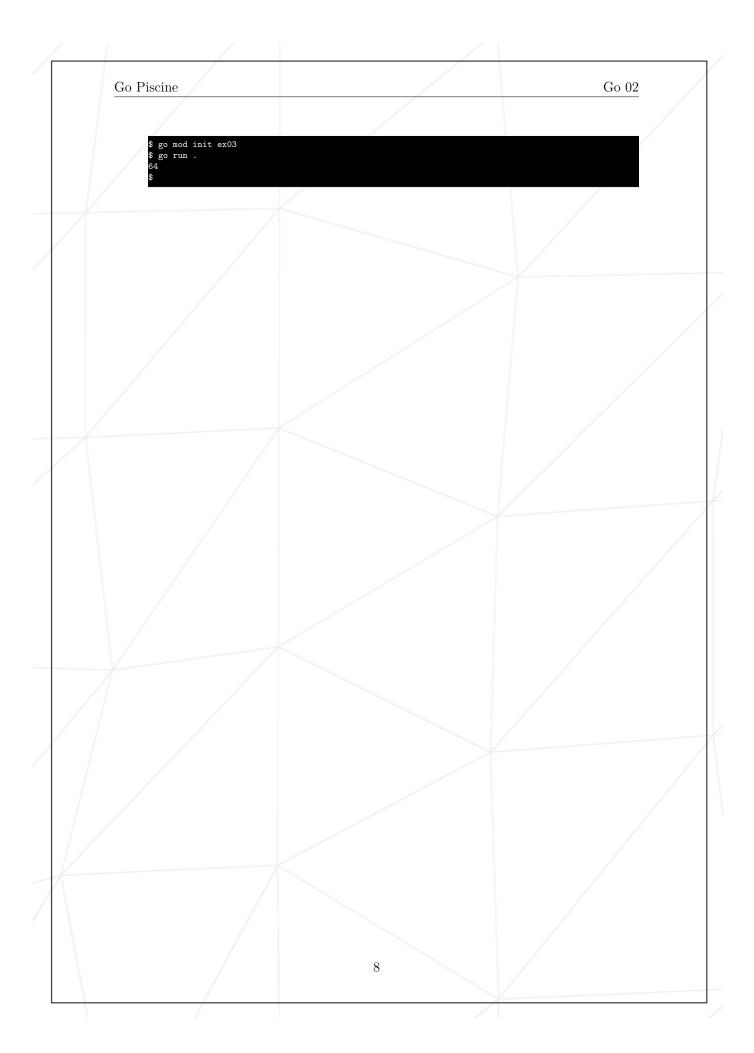
- Negative powers will return 0. Overflows do not have to be dealt with.
- for is forbidden for this exercise.
- Expected function

```
func RecursivePower(nb int, power int) int {
}
```

• Usage

```
package main
import (
    "fmt"
    "piscine"
)

func main() {
    fmt.Println(piscine.RecursivePower(4, 3))
}
```



## Chapter VI

## Exercise 04: fibonacci

	Exercise 04	
/	fibonacci	
Turn-in directory : $ex04/$		
Files to turn in : *		
Allowed packages : fmt		
Allowed builtin functions:	None	

Write a recursive function that returns the value at the position index in the fibonacci sequence.

- The first value is at index 0.
- The sequence starts this way: 0, 1, 1, 2, 3 etc...
- A negative index will return -1.
- for is forbidden for this exercise.
- Expected function

func Fibonacci(index int) int {

Go Piscine Go 02

• Usage

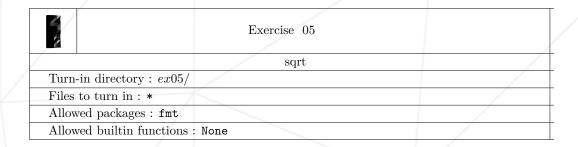
```
package main
import (
    "fmt"
    "piscine"
)

func main() {
    arg1 := 4
    fmt.Println(piscine.Fibonacci(arg1))
}
```

```
$ go mod init ex04
$ go run .
3
```

## Chapter VII

Exercise 05: sqrt



Write a function that returns the square root of the int passed as parameter, if that square root is a whole number. Otherwise it returns 0.

• Expected function

```
func Sqrt(nb int) int {
}
```

• Usage

```
package main
import (
    "fmt"
        "piscine"
)
func main() {
        fmt.Println(piscine.Sqrt(4))
        fmt.Println(piscine.Sqrt(3))
}
```

```
$ go mod init ex05
$ go run .
2
0
$
```

## Chapter VIII

## Exercise 06: isprime

	Exercise 06	
/	isprime	
Turn-in directory : $ex06/$		
Files to turn in : *		
Allowed packages : fmt		
Allowed builtin functions:	None	

Write a function that returns true if the int passed as parameter is a prime number. Otherwise it returns false.

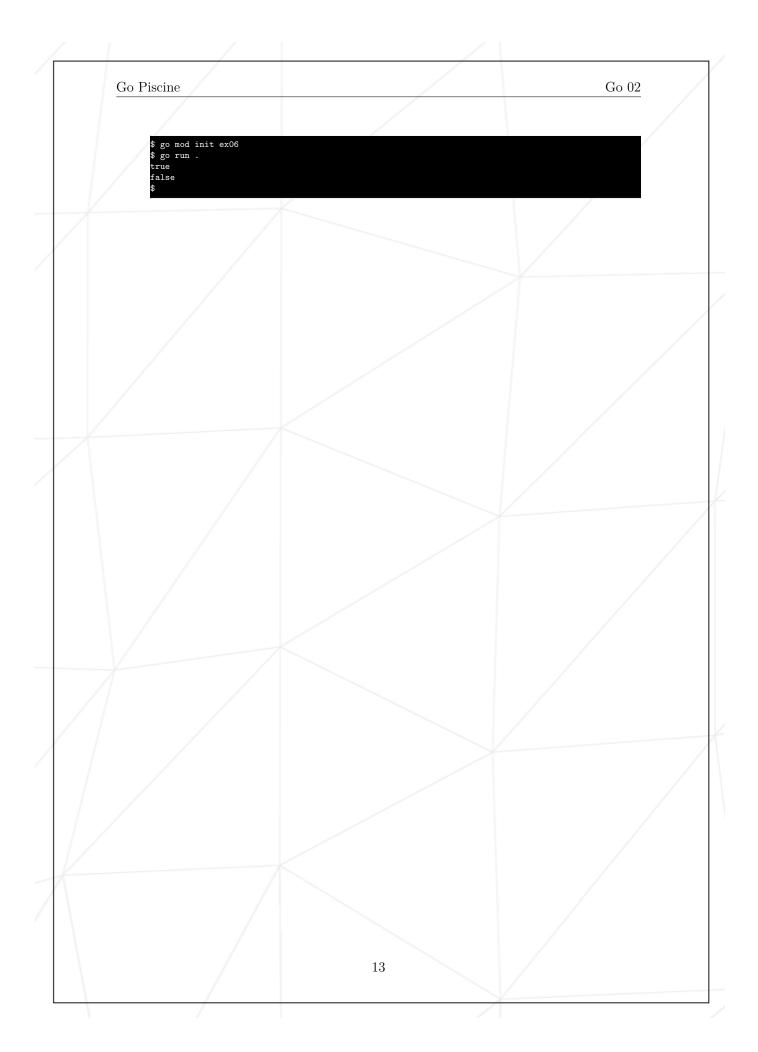
- (We consider that only positive numbers can be prime numbers)
- (We also consider that 1 is not a prime number)
- Expected function

```
func IsPrime(nb int) bool {
}
```

• Usage

```
package main
import (
    "fmt"
    "piscine"
)

func main() {
    fmt.Println(piscine.IsPrime(5))
    fmt.Println(piscine.IsPrime(4))
}
```



## Chapter IX

## Exercise 07: findnextprime

	Exercise 07	
	findnextprime	
Turn-in directory : ex07/		
Files to turn in : *		
Allowed packages : fmt		
Allowed builtin functions	: None	

Write a function that returns the first prime number that is equal or superior to the int passed as parameter.

- (We consider that only positive numbers can be prime numbers)
- Expected function

```
func FindNextPrime(nb int) int {
}
```

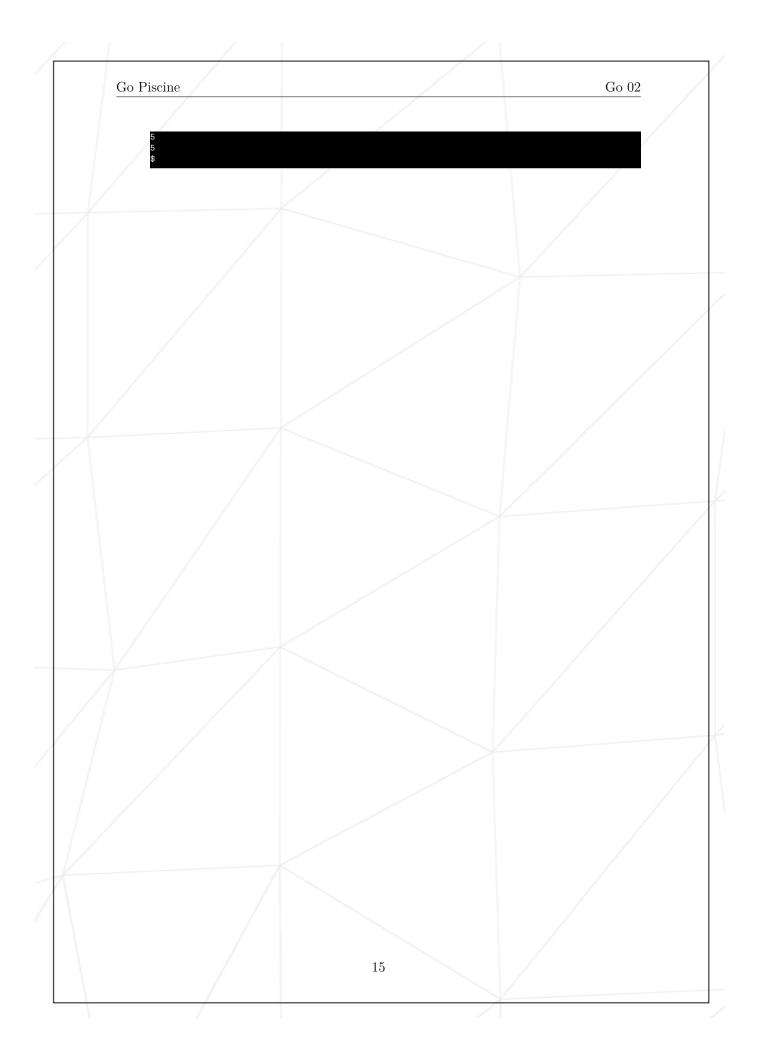
• Usage

```
package main

import (
     "fmt"
     "piscine"
)

func main() {
     fmt.Println(piscine.FindNextPrime(5))
     fmt.Println(piscine.FindNextPrime(4))
}
```

```
$ go mod init ex07
$ go run .
```



# Chapter X

# Exercise 08: eightqueens

	Exercise 08	
/	eightqueens	
Turn-in directory : ex08/		
Files to turn in : *		
Allowed packages : None		
Allowed builtin functions:	None	

Write a function that prints the solutions to the eight queens puzzle.

- Recursivity must be used to solve this problem.
- Expected function

```
func EightQueens() {
}
```

• Your function should print something like this:

```
$ go mod init ex08
$ go run .
15863724
16837425
17468253
...
$
```