



Piscine Golang - Day00

🕒 Created @October 24, 2021 10:33 PM

요약: 이 문서는 Golang의 기본 문법 내용을 담고 있습니다.

서브젝트 오류 문의 : bigpel66@icloud.com

Contents

- III* Exercise 00 : Are you ready?
- IV* Exercise 01 : Welcome to GoLand!
- V* Exercise 02 : Greetings Everyone!
- VI* Exercise 03 : Boss is watching us...
- VII* Exercise 04 : We're going to launch lottery...

Chapter I - General Rules

- 모든 Exercise들은 Golang 1.17로 진행하며 테스트 코드와 함께 작성되어야 합니다. 작성된 테스트 코드들은 Exercise와 함께 제출되어야 합니다.
- 제출된 테스트 코드 내에 테스트 케이스는 반드시 존재해야 합니다.
- 적극적인 테스트 케이스는 권장되며, 특히 예외 케이스에 신경써야 합니다.
- 평가자는 자신이 생각했던 테스트 케이스가 존재하지 않는 경우, 피평가자에게 테스트 케이스를 요청할 수 있습니다. 이 때 피평가자의 코드가 동작하지 않으면 평가는 종료됩니다.
- 테스트 진행은 "testing" 패키지를 이용해야 하며, "testing"과 "github.com/stretchr/testify/assert" 둘 중 하나를 일관성 있게 사용해야 합니다.
- 테스트 진행 시 testing 객체를 이용한 벤치마킹 역시 적극 권장됩니다.
- Exercise의 제출은 동료 평가 및 Moulinette 채점으로 **PASS** 혹은 **FAIL**이 결정됩니다.
- Exercise들은 gofmt를 이용하여 포매팅이 되어 있어야 합니다. 동료 평가 진행 시, 제출된 Exercise가 포매팅이 되어 있지 않다면 **FAIL**입니다.
- Exercise들은 golanci-lint를 이용했을 때, 별다른 문제가 없다는 것이 검증되어야 합니다. golanci-lint에서 문제가 있는 부분이 발견되면 **FAIL**입니다.
- 동료 평가 진행을 위한 주석은 적극 허용됩니다. 각 함수 및 변수들에게 적용된 주석은 Golang에서 정식으로 지원하는 godoc을 이용하여 Document로 만들 수 있습니다.
- 컴파일은 **go build**를 이용하며, 이를 위해 **go mod**를 먼저 이해해야 합니다. **go mod**를 이용하지 않고 제출된 Exercise 역시 **FAIL**입니다.
- 제공된 서브젝트를 철저히 파악하여 Golang의 기초 지식을 모두 얻을 수 있도록 하십시오.
- 서브젝트에서 요구하는 내용들이 짧더라도, 이 내용들을 이해하고 예상한대로 작동되도록 시간을 쏟는 것은 굉장히 가치 있는 행동입니다. 다양한 시도를 권장합니다.
- *Go ? Ahead ! Write in Go.*



go mod를 이용하기 전에 GOPATH, GOROOT를 알아보십시오.

Chapter II - Preamble

누가 만들었을까?

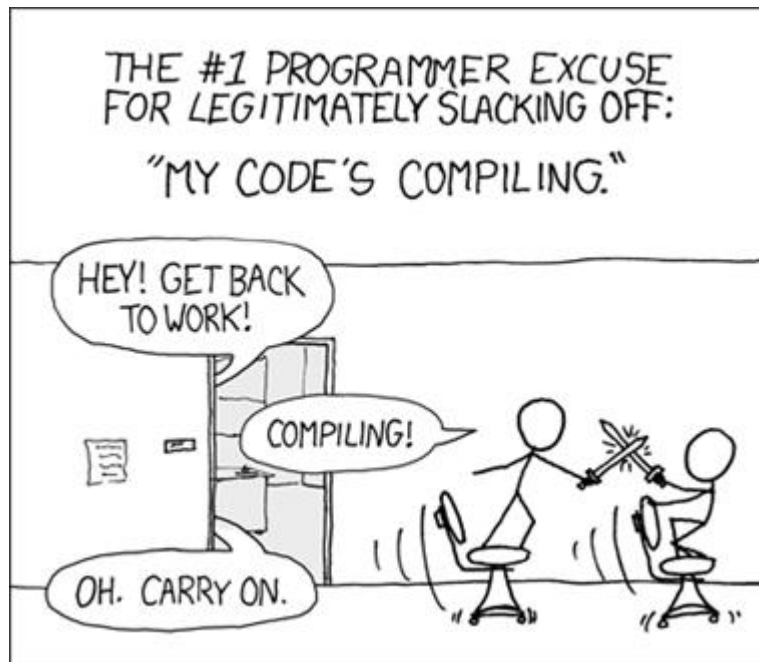
2007년 샌프란시스코의 구글 사무실에는 화이트보드를 앞에 두고 세 사람이 모여 앉았다. 세 사람은 서로 생각한 것들을 화이트보드에 그리기도 하고, 중간 중간 언쟁을 하기도 하며 시간을 보내고 있었다. 세 사람이 얘기를 나누고 있던 것은 새로운 개발 언어에 대한 내용이었다. 이 사람들은 바로 Robert Griesemer, Rob Pike, Ken Thompson이고, 새롭게 만드는 언어는 Golang이었다. 개발 중간에 2명의 멤버가 합류하게 된다. Ian Taylor는 Golang을 GCC 컴파일러에 적용하는 일을 맡았고, Russ Cox는 개발 업무를 담당하면서 Golang의 완성도를 담당했다. 여러 사람이 모여 결국 Golang은 2009년 11월에 정식으로 릴리즈되었다.

Robert Griesemer는 Java HotSpot Compiler를 만들었고, Chrome 및 JavaScript 엔진 개발에 참여했다. Rob Pike는 UTF-8과 분산 운영체제를 개발했다. Russ Cox는 수년간 세계 최고 수준의 프로그래머로 인정받아 왔는데, 대학 진학 전에 USACO, IOI의 국가 대표로 출전한 이력이 있고 운영체제, 컴파일러, 알고리즘 분야에서 다양하게 논문을 썼다.

어떤 점이 Golang을 매력적으로 만드는가?

대단한 사람들끼리 어떤 목적으로 Golang을 만들었는지 살짝 확인하면, 매력 포인트를 쉽게 찾을 수 있다. Golang의 목표는 시스템 개발 분야였다. 구체적으로 Golang의 개발진들이 밝힌 적용 가능 예상 분야는 웹 서버, 웹 브라우저, 웹 로봇, 검색 엔진, 컴파일러, 프로그래밍 도구, IDE, 운영체제가 있다. Golang은 C와 비슷한 성능을 내면서 C/C++로 개발되어 온 분야를 대체할 수 있다고 볼 수 있다. 하지만 Golang은 여기서 그치지 않고 더 많고 다양한 분야의 개발에서 활용될 수 있다. 실제로 출시 후 1년이 지났을 때는 일반 개발 언어 (General Purpose Language)로 사용해도 손색이 없다는 주장들도 있었다. 특히 이에 대해선 Rob Pike는 "내가 써 본 언어들 중에서 가장 생산성이 높다"라고 했다.

개발 분야 및 개발 속도 외에도 Golang이 매력적인 이유들은 더 있다. 메모리 관리로부터 자유로울 수 있는 Garbage Collection을 지원하고, 강타입이지만 동적타입 언어의 속성을 갖고 있으며, 굉장히 빠른 컴파일을 지원하며, 동시성을 위한 손쉬운 프로그래밍을 제공한다.



Golang이 동시성을 언어 차원에서 지원하는 만큼 GC는 반드시 필요하며, Golang은 GC를 완벽히 지원한다. 특히 Golang의 GC는 다른 GC보다 빠른 처리를 자랑하며, 고루틴 덕분에 성능 저하를 최소화할 수 있었다. 또한 동적타입 언어의 특성 때문에 런 타임에 에러를 직면하는 안정성 문제를 강타입을 고수하면서 컴파일러로 해당 문제들을 사전에 찾아낼 수 있도록 했다. 무엇보다도 C에서 기본 헤더 파일을 포함할 때 많은 파일에 접근하면서 컴파일에 많은 시간을 쏟아야 했던 문제를 Golang에서는 빌드 순서 조정을 통해 모듈 참조를 최소화 시킴으로써 빌드 시간을 대폭 줄일 수 있었다. 그리고 Golang은 고루틴이라는 경량 쓰레드 덕분에 Java 및 C++처럼 쓰레드 풀을 돌 필요도 없으며, 두 언어가 쓰레드 생성 시 영역 보호를 위해 1MB 가드 페이지를 두는 것에 비해 Golang은 가드 페이지 없이 단순히 2KB의 적은 공간을 이용하면서 필요에 따라 힙 공간을 확보하며 이용된다.

어떻게 배울 수 있을까?

구글에서 제공하는 교육 자료는 굉장히 탄탄하다. 아래 링크에서 Golang의 문법들을 꼼꼼히 학습할 수 있다.

A Tour of Go



<https://tour.golang.org/welcome/1>

Chapter III

Exercise 00 : Are you ready?

<u>Aa</u> Name	<u>≡</u> Tags
<u>Turn-in directory</u>	ex00
<u>Files to turn in</u>	go.mod
<u>Module name</u>	ex00
<u>Allowed packages</u>	None

당신은 General Rules에 명시된 GOPATH, GOROOT에 대해서 충분히 학습하고 왔습니다. 현재 당신이 이용하고 있는 Golang 버전은 반드시 1.17일 것이므로, go mod를 적극적으로 이용하여 더욱 세련된 방법으로 모듈들을 관리할 것입니다.

아래 명령어와 같은 형태로 go.mod를 작성해보십시오.

```
go mod init <module-name>
```

명령어를 실행하면 go.mod의 내용은 다음과 같이 나와야 합니다.

```
> cat go.mod
module ex00



go 1.17
```

디렉토리 구조는 아래와 같이 만들어져야 합니다.

```
ex00
└─ go.mod
```

Chapter IV

Exercise 01 : Welcome to GoLand!

 Name	 Tags
<u>Turn-in directory</u>	ex01
<u>Files to turn in</u>	main.go go.mod
<u>Module name</u>	ex01
<u>Allowed packages</u>	"bufio" , "os"

당신은 go.mod를 이해했고, 활용할 준비가 된 것 같습니다. GoLand를 개장해봅시다. 모름지기 테마 파크는 고유한 인사가 있어야겠죠.

우리가 개장할 GoLand는 환영 인사를 "Welcome to GoLand!\n"라고 정했습니다. 프로그램이 실행되면 "Welcome to GoLand!\n"를 출력해주도록 코드를 작성하십시오.



Golang의 입력과 출력은 내장 패키지인 fmt를 이용하면 편리합니다. fmt는 Print, Println, Printf, Scan, Scanln, Scanf 등 사뭇 익숙한 입출력 함수들을 지원합니다.

단, 조건이 있습니다. fmt 패키지를 이용하지 않고 출력하십시오. fmt의 입력과 출력 함수들은 자동으로 표준 입력과 표준 출력을 대상으로 동작하게 됩니다. 다만 버퍼가 없어서 그 행동이 꽤나 느리기 때문에, 우리는 어느 상황에서도 긴 인사를 할 수 있도록 버퍼를 두는 형식으로 입력과 출력이 가능하도록 표준 입력과 표준 출력을 이용할 것입니다.

힌트는 Allowed packages에 있습니다. 출력 함수로 반환되는 값들은 무시해도 됩니다. 마지막으로 출력 형태는 아래 그림과 일치해야 합니다. Welcomd to GoLand!


```
> ./ex01
Welcome to GoLand!
```

디렉토리 구조는 아래와 같이 만들어져야 합니다.

```
ex01
├─ go.mod
└─ main.go
```

Chapter V

Exercise 02 : Greetings Everyone!

<u>Aa</u> Name	 Tags
<u>Turn-in directory</u>	ex02
<u>Files to turn in</u>	greet/korean.go , greet/english.go , greet/mandarin.go main.go , go.mod
<u>Module name</u>	ex02
<u>Allowed packages</u>	"bufio" , "os"

GoLand의 개장은 매우 성공적이었습니다. 덕분에 세계적인 명소가 되어 다양한 나라에서 찾아오게 되었습니다. GoLand는 늘 State-of-the-Art 이기 때문에 다양한 언어를 구사하는 직원을 고용했습니다. 이에 따라 각 언어의 환영인사를 별도의 패키지로 관리하려고 합니다.

일단 가장 많이 사용되는 언어 3가지, 한국어, 영어, 한자로 환영 인사를 할 수 있는 함수를 만들어야 합니다. 각 함수의 원형은 아래와 같습니다.

```
func Korean() string // GoLand에 어서오세요!
```

```
func English() string // Welcome to GoLand!
```

```
func Mandarin() string // 欢迎光临 GoLand!
```

만들어진 함수를 이용하여 main 함수에서 호출하고, 아래와 같은 결과가 나오도록 코드를 작성하십시오.

```
> ./ex02
GoLand에 어서 오 세 요 !
Welcome to GoLand!
欢迎光临 GoLand!
```






Golang에서 별도의 작업을 하지 않았는데도, 영어 이외의 다른 문자들을 뽑아낼 수 있다는 점에 주목해야 합니다. 기본적으로 Golang의 문자는 UTF-8으로 구성되면서 다국어를 지원하도록 되어 있습니다. 이에 따라 문자 하나에 할당되는 바이트가 문자마다 다르다는 문제점이 있습니다. 특히 문자열을 바이트로 읽을 때는 UTF-8으로 된 바이트를 읽는 행위가 되기 때문에 원하는 문자를 얻지 못할 수 있습니다. 이처럼 하나의 문자를 온전히 담아내려면 Golang에서는 rune이라는 int32의 별칭인 타입을 이용해야 합니다. 이후 챕터에서 슬라이스라는 개념을 다루게 되겠지만, 문자열과 rune 슬라이스의 관계는 굉장히 중요합니다. 더 자세한 내용을 꼭 찾아보십시오.

디렉토리 구조는 아래와 같이 만들어져야 합니다.

```
ex02
├─ go.mod
├─ main.go
└─ greet
    ├── korean.go
    ├── english.go
    └─ mandarin.go
```

Chapter VI

Exercise 03 : Boss is watching us...

 Name	 Tags
<u>Turn-in directory.</u>	ex03
<u>Files to turn in</u>	greet/greet_test.go , greet/korean.go , greet/english.go , greet/mandarin.go main.go , go.mod
<u>Module name</u>	ex03
<u>Allowed packages</u>	"bufio" , "os" , "testing" , "github.com/stretchr/testify/assert"

GoLand 개장은 성공적이지만, 직원들의 근태를 확인하고자 합니다. 몇몇 고객들로부터 인사를 제대로 하지 않는 직원이 있다는 제보를 받았습니다. 따라서 우리는 각 언어별로 환영 인사가 제대로 출력되는지 확인하려고 합니다. 이에 따른 테스트 코드를 작성하십시오. 테스트 코드에서는 각 환영 인사 결과가 예측한대로 출력되는지 확인할 수 있어야 합니다.

내장되어 있는 "testing" 패키지를 이용할 수 있지만, 이번 만큼은 Allowed packages에 제시된 외장 패키지를 이용하십시오. 만일 외장 패키지를 불러왔는데, 이용할 수 없다면 이는 모듈이 정상적으로 정리되지 않아서 그런 것입니다. 아래 명령어를 이용하여 외장 패키지를 모듈 내에서 이용할 수 있게 정리할 수 있습니다.

테스트 자체는 testing 객체를 이용하기 때문에, "testing"은 반드시 불러와야 합니다. 대신, 테스트 코드 작성 시 각 케이스는 외장 패키지를 이용하여 검증되어야 합니다.

go mod tidy

"github.com/stretchr/testify/assert"의 이용법을 [pkg.go.dev](https://pkg.go.dev/github.com/stretchr/testify/assert) 에서 쉽게 확인할 수 있습니다. 이를 이용하여 main 함수에서 호출했던 함수들이 제 기능을 다하는지 확인하십시오.

아래와 같은 결과를 얻을 수 있으면, 당신은 Golang의 강력한 테스트 기능을 기반으로 코드를 작성할 준비가 되었습니다! Golang의 TDD를 꼭 찾아서 연습해보십시오.

```
> go test -v
=== RUN   TestGreet
--- PASS: TestGreet (0.00s)
PASS
ok      ex02/greet    0.098s
```

디렉토리 구조는 아래와 같이 만들어져야 합니다.

```
ex03
├─ go.mod
├─ main.go
└─ greet
    ├─ greet_test.go
    ├─ korean.go
    ├─ english.go
    └─ mandarin.go
```

Chapter VII

Exercise 04 : We're going to launch lottery...

<u>Aa</u> Name	<u>≡</u> Tags
<u>Turn-in directory.</u>	ex04
<u>Files to turn in</u>	greet/greet_test.go , greet/korean.go , greet/english.go , greet/mandarin.go , lottery/lottery.go , lottery/lottery_test.go , main.go , go.mod , numbers.txt
<u>Module name</u>	ex04
<u>Allowed packages</u>	"bufio" , "math/rand" , "os" , "strconv" , "testing" , "time" , "github.com/stretchchr/testify/assert"

GoLand가 1주년을 맞이하여 이벤트를 런칭했습니다. GoLand에서는 입장하는 고객들에게 0 ~ 99 사이의 무작위 값을 4개 선택하도록 하고, 오후 5시가 되면 메인 홀에서 하나의 수를 발표하여 맞춘 고객들에게는 GoLand 퇴장 시 상품을 지급하기로 했습니다. 당신은 아직 값을 선택하지 않았습니다. 곧 5시가 되어갑니다.

당신이 작성한 로또 번호는 Number 함수를 이용하여 처리됩니다. 조건은 다음과 같습니다.

- Number 함수는 uint 타입의 크기 4짜리 배열을 반환하는 함수입니다.
- Number 함수는 Relative Path를 인자로 받습니다.
- 로또 번호는 numbers.txt에 있는 값을 읽어서 이용합니다. numbers.txt의 경로는 "os" 패키지의 Getwd로 얻은 경로 + Relative Path + numbers.txt로 이용되어야 합니다. numbers.txt 파일 내부에 기록된 값들은 공백 문자로 구분될 수 있어야 합니다. (즉, 한 줄에 모든 값이 존재할 필요는 없습니다.)
- 최종 제출 시 numbers.txt도 제출되어야 하므로, 파일 내에는 올바른 값이 들어가 있어야 합니다.
- 함수의 원형은 다음과 같습니다.

```
func Number(rp string) (arr [4]uint)
```



"os" 패키지의 다양한 기능을 확인할 수 있어야 합니다. 특히 열었던 파일은 닫는 것을 잊지 마십시오. defer를 활용해보십시오.



numbers.txt를 읽어낼 때는 반드시 "os"패키지의 Getwd를 이용하고, Getwd가 동작하는 경로인 Working Directory는 반드시 go.mod가 있는 경로를 기준으로 한다고 가정합니다. 만일 Working Directory가 다른 디렉토리로 바뀌게 되면 Getwd로 읽어낸 경로가 바뀔 수 있기 때문에 numbers.txt를 읽어낼 수 없는 경우가 생깁니다. 따라서 코드를 작성하거나 평가를 진행할 때의 Working Directory는 go.mod가 존재하는 경로가 이용될 수 있도록 하십시오.



Number 함수에서 Relative Path를 받는 이유를 테스트 코드와 잘 엮어서 생각해보십시오. 이와 관련해서 테스트 파일에 대한 결과를 잘 확인해보십시오.

정답 번호는 Answer 함수를 이용하여 만들어집니다. 조건은 다음과 같습니다.

- 정답 번호는 0 ~ 99 사이의 무작위 값으로 결정되어야 하며, "math/rand"를 이용합니다.
- "math/rand"에서의 시드 값은 "time" 패키지로 현재 시간을 UnixNano로 표현한 값을 사용합니다.
- Answer 함수의 테스트 코드는 만들지 않습니다.
- 함수의 원형은 아래와 같습니다.

```
func Answer() uint
```

추출된 로또 번호는 Match 함수를 이용하여 검증하게 됩니다. Match 함수 구현 조건은 아래와 같습니다.

- 로또 번호가 정답 번호와 하나라도 일치하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.
- 로또 번호와 정답 번호의 비교를 진행할 때, range 반복을 이용합니다.

- Match 함수와 테스트 코드의 로직이 비슷하더라도, 테스트로 기능에 문제가 없는 것을 확인하십시오.
- 함수의 원형은 아래와 같습니다.

```
func Match(answer uint, arr *[4]uint) bool
```

Match 함수는 main 함수 내에서 호출되며, 그 값이 true일 때는 "I got a golden ticket!\n"을, false라면 "I have to eat cabbage soup...\n"을 출력하도록 main 함수를 적절히 변경해야 합니다.

당신은 GoLand에 입장한 상태에서 로또 번호를 작성했기 때문에 기존의 환영 인사를 받는 상황을 그대로 유지하며, 아래와 같은 형태로 프로그램이 진행될 수 있도록 main 함수를 구성하십시오. 출력 예시에 나온 로또 번호 자체는 출력 예시와 다를 수 있습니다.

```
> ./ex04
GoLand에 어서오세요!
Welcome to GoLand!
欢迎光临 GoLand!

Answer is 1

My lottery number is...
1      2      3      4
I got a golden ticket!
```

디렉토리 구조는 아래와 같이 만들어져야 합니다.

```
ex04
├─ numbers.txt
├─ go.mod
├─ main.go
├─ lottery
│  ├─ lottery.go
│  └─ lottery_test.go
└─ greet
   ├─ greet_test.go
   ├─ korean.go
   ├─ english.go
   └─ mandarin.go
```