

بسم الله الرحمن الرحيم

تکنولوژی کامپیوتر

جلسه دوم

جلسه‌ی گذشته

قوانین درس، شروع گولنگ

خلاصه‌ای از قوانین

- ۵ سری تمرین عملی، ۱۵ نمره
- آزمون میان‌ترم و پایان‌ترم، حضوری و هر کدام ۳ نمره
- گروهی با گروه‌های ۳ نفره.

تاخیر در تمرین‌ها؟

■ هر سری تمرین، ۷ روز تاخیر مجاز خواهد داشت.

Go Tour

- <https://go.dev/tour>

Go Module

- برای شروع این دستور رو زدیم:
- `go mod init <module_name>`
- اسم ماژول به این شکل هست:
 - `<prefix>/<descriptive-text>`
 - عموماً `prefix` رو اسم شرکت یا اسم ریپازیتوری `git` می‌گذارند.
 - مثلاً: `github.com/Atofighi/felan`

Go.Mod

■ بعد از زدن `go mod init ...`، یک فایل `go.mod` ساخته می‌شود:

■ محتوای فایل `go.mod`:



```
module example.com/mymodule

go 1.14

require (
    example.com/othermodule v1.2.3
    example.com/thismodule v1.2.3
    example.com/thatmodule v1.2.3
)

replace example.com/thatmodule => ../thatmodule
exclude example.com/thismodule v1.3.0
```

اسم فایل‌ها

- Lowercase Letters
 - *main.go* 
- Underscores for Readability:
 - *user_auth.go* 
- File Extension: .go
- Test Files: *_test.go

Main.go 

userAuth.go 

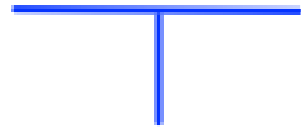
یک فایل go.

- هر statement از جمله‌های global با یک کلیدواژه، مثل package، import، var، func و ... شروع شده.

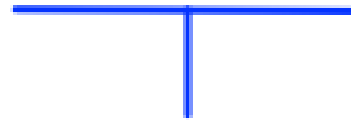
```
greetings > go hello.go
1  package greetings
2
3  import "fmt"
4
5  // Hello returns a greeting for the named person.
6  func Hello(name string) string {
7      // Return a greeting that embeds the name in a message.
8      message := fmt.Sprintf("Hi, %v. Welcome!", name)
9      return message
10 }
11
```

تعریف یک تابع

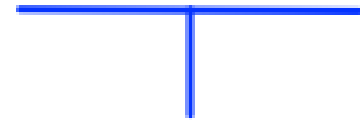
```
func Hello(name string) string
```



Function
name



Parameter
type



Return
type

تعریف یک متغیر

■ Using the var Keyword

- `var <variable_name> <type> = <value>`

- بر خلاف جاوا، تایپ متغیر بعد از اسمش میاد.

- `var age int = 30`

■ Type Inference

- `var name = "Alice"`

تعریف یک متغیر

■ Declaration Without Initialization:

- *var isActive bool*

– بر خلاف جاوا، می‌تونیم مقدار دهی نکنیم و مقدار پیش‌فرض می‌گیره (برای *boolean* ها *false*، برای عددها صفر، برای رشته "" و برای بقیه *nil* (که شبیه *null* هست))

■ Multiple Variables

- *var a, b, c int = 1, 2, 3*

پکیج fmt

func Println

```
func Println(a ...any) (n int, err error)
```

Println formats using the default formats for its operands and writes to standard output. Spaces are always added between operands and a newline is appended. It returns the number of bytes written and any write error encountered.

پکیج fmt

func Printf

```
func Printf(format string, a ...any) (n int, err error)
```

Printf formats according to a format specifier and writes to standard output. It returns the number of bytes written and any write error encountered.

پکیج fmt

```
package main

import (
    "fmt"
)

func main() {
    const name, age = "Kim", 22
    fmt.Printf("%s is %d years old.\n", name, age)

    // It is conventional not to worry about any
    // error returned by Printf.

}
```

Output:

Kim is 22 years old.

فرمت در fmt

- %v the value in a default format
 - *when printing structs, the plus flag (%+v) adds field names*
- %T a Go-syntax representation of the type of the value
- %% a literal percent sign; consumes no value
- More details on: <https://pkg.go.dev/fmt@go1.24.0>

فرمت کردن رشته با fmt

func Sprintf

```
func Sprintf(format string, a ...any) string
```

Sprintf formats according to a format specifier and returns the resulting string.

تعریف یک متغیر

■ Using Short Variable Declaration (Inside Functions)

- *count := 10*
- *a, ok := 10, false*

■ در این حالت حداقل یکی از متغیرهای سمت چپ باید جدید باشد و تعریف شود، بقیه‌ی متغیرها اگر از قبل وجود داشته باشند فقط مقداردهی می‌شوند.

مقدار دهی متغیر

- $a = 10$
- $a, b = 10, 20$

نام گذاری متغیرها

■ Valid Identifier:

- *Variable names can consist of letters (a-z, A-Z), digits (0-9), and underscores (_).*
- *They must not start with a digit.*

■ Case Sensitivity:

- *Variable names are case-sensitive. For example, count and Count are considered different variables.*

■ Reserved Keywords:

- *You cannot use Go's reserved keywords (such as if, else, func, package, etc.) as variable names.*

متغیرهای Exported و Unexported

- If a variable name starts with an **uppercase** letter (e.g., Name), it is exported
 - *meaning it is accessible from other packages.*
 - چیزی شبیه *public* در جاوا
- If it starts with a lowercase letter (e.g., name), it is unexported
 - *... and only accessible within its own package.*
 - چیزی شبیه *private* در جاوا

Built-in Types

■ Boolean Type

- *bool:*
- *Represents a boolean value, which can be either true or false.*
- *var isValid bool = true*

Built-in Types

- Numeric Types

- *Integers:*

- Signed Integers:

- int (platform-dependent, typically 32 or 64 bits)

- int8, int16, int32, int64

- var a int = 42

- var b int8 = -8

Built-in Types

- Numeric Types

- *Integers:*

- Unsigned Integers:

- uint (platform-dependent)

- uint8 (also known as byte), uint16, uint32, uint64

- uintptr (an integer type that can store the uninterpreted bits of a pointer)

Built-in Types

- Numeric Types

- *Floating-Point Numbers*

- float32 and float64:

- Used for representing decimal numbers.

- *Complex Numbers*

- complex64 and complex128:

- Represent complex numbers (numbers with a real and imaginary part).

- `var cNum complex64 = complex(1, 2) // 1 + 2i`

Built-in Types

■ String Type

- *string:*
- *A sequence of characters. Strings are immutable in Go.*
- *var greeting string = "Hello, Go!"*

■ بر خلاف جاوا، برای تایپ char نداریم، ولی دو تایپ alias داریم به نام‌های rune (که int32 هست برای Unicode char) و byte (که unit8 هست)

Constants

- Constants are declared like variables, but with the `const` keyword.
- Constants can be character, string, boolean, or numeric values.
- Numeric constants are high-precision values.
- An untyped constant takes the type needed by its context.

جلسه‌ی جدید

ادامه‌ی سینتکس گولنگ...