Packages

//Go Bases

IT BOARDING

BOOTCAMP

Packages

“El propósito de un paquete es diseñar y mantener una gran cantidad de programas agrupando funciones relacionadas en unidades individuales para que puedan ser fáciles de mantener y comprender.”.

Índice

IT BOARDING

BOOTCAMP

FMT

En el lenguaje Go, el paquete **fmt** proporciona funciones para **formatear** y **mostrar** datos en la **entrada / salida (I/O) estándar**. Abstrae la función nativa **println()**

De esta forma ofrece capacidades más avanzadas para formatear texto y datos antes de imprimirlos.

{}

```
package main

import "fmt"

func main() {
    // fmt
    fmt.Println("Geometrical shapes properties")
}
```

FMT

El paquete **fmt** utiliza **placeholders** para especificar el **tipo de dato** a **formatear** o **mostrar** en consola

{}

```go
func main() {
    // placeholders
    // - %d -> int
    // - %f -> float
    // - %c -> char
    // - %s -> string
    // - %t -> bool
    // - %p -> pointer
    // - %v -> any value
    // - %T -> type of value

    fmt.Printf("name: %s, age: %d, height: %f, license: %t\n", "John", 25, 1.75, true)
}
```

IO

El paquete **IO** proporciona una serie de interfaces y funciones que se utilizan para realizar **operaciones** de **entrada** y **salida** genéricas (**I/O**).

Nos permite trabajar con **flujos de datos**, como **archivos**, **redes**, y otras operaciones de I/O.

Las **interfaces** más comunes son: **io.Reader**, **io.Writer** y **io.Closer**

{}

```
// Reader allows us to read from a data source as a stream
// -p: slice of bytes where the read data will be stored
// -n: number of bytes read
// -err: error. One of them represents the end of the stream EOF
type Reader interface {
    Read(p []byte) (n int, err error)
}
```

A continuación, proporcionamos un ejemplo leyendo de un **reader** de una **string**

{}

```
func main() {
    // NewReader creates a new reader from a string
    reader := strings.NewReader("I'm gonna be an stream of data")
    // buffer to store the read data
    // -len: 50     -cap: 50
    buffer := make([]byte, 50)
    n, err := reader.Read(buffer)
    if err != nil {
        panic(err)
    }
    // print the buffer up to the number of bytes read
    println(string(buffer[:n]))
}
```

OS

Package **OS** proporciona funcionalidades para interactuar con el sistema operativo y realizar operaciones relacionadas con el entorno del sistema.

Aquí hay un ejemplo simple, abriendo un archivo y tratando leer de él.

{}

```go
func main() {
    // open a file
    f, err := os.Open("file.txt")
    if err != nil {
        panic(err)
    }
    defer f.Close()
    // read the file and store the data in a slice of bytes
    // f: file implements the Reader interface
    data, err := io.ReadAll(f)
    if err != nil {
        panic(err)
    }
    fmt.Println(string(data))
}
```

Gracias.

IT BOARDING

BOOTCAMP