Curso de Go (Golang)

En este curso aprendí los conceptos básicos del lenguaje de programación Go, donde aprendí a usar estructuras, funciones para abrir archivos, Imprimir resultados por terminal entre otras cosas

Imprimir Hello World

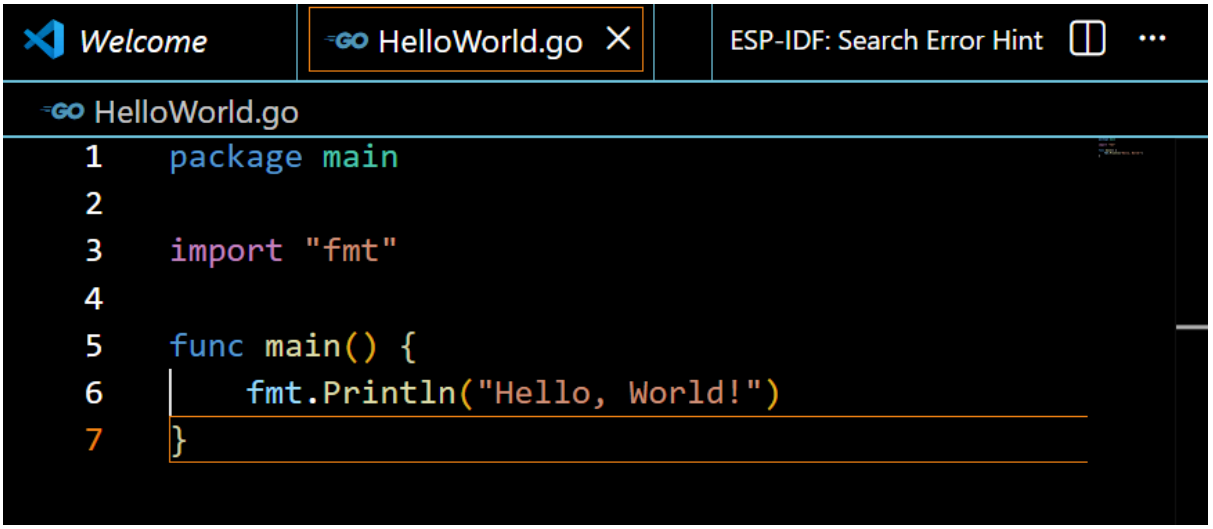


Figura 1. Hello World

`package main`: Le dice a Go que este archivo debe compilarse como un programa ejecutable, no como una librería.
`import "fmt"`: Importa el paquete "format", que contiene las funciones para imprimir texto en la consola.
`func main()`: Es el punto de entrada. Todo lo que pongas dentro de estas llaves `{ }` es lo que se ejecutará primero.

Figura 2. Creación de .exe

Build crea un archivo ejecutable (.exe) que será el archivo que ejecutaremos para activar lo que escribimos en el archivo .go. El último comando de la terminal lo ejecuta

Truncar un número decimal al entero más cercano

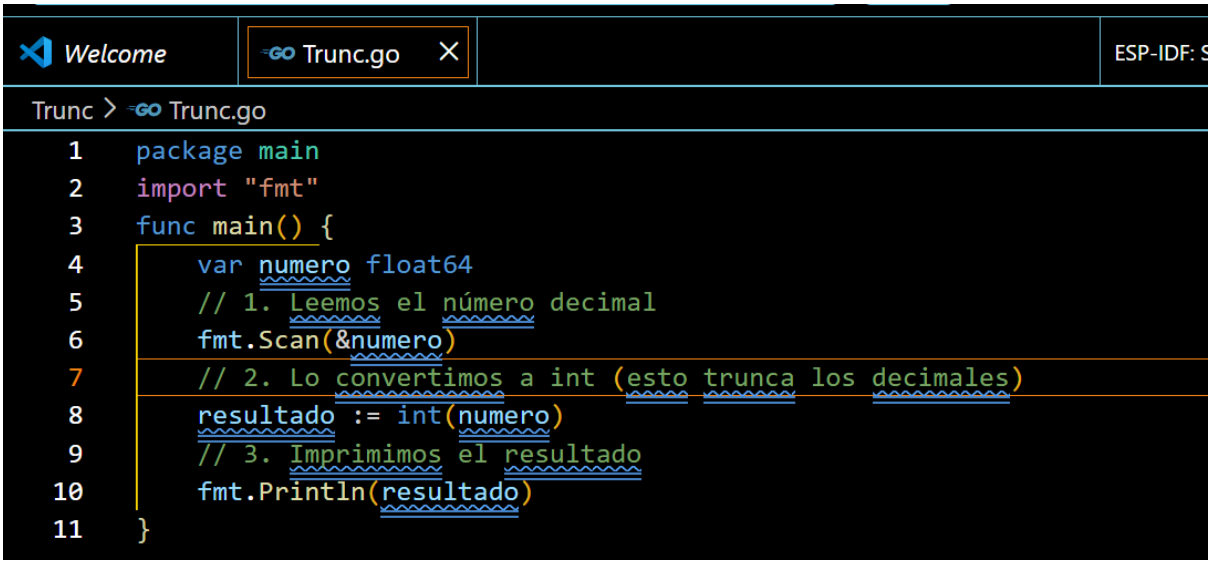


Figura 3. Código de truncar a entero

Creamos el nuevo archivo escribimos el codigo que nos solicitaron

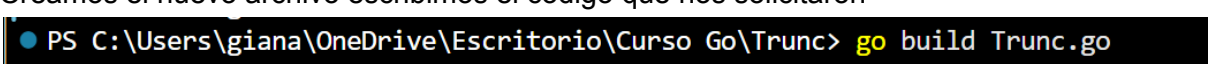


Figura 3. Creación de ejecutable

creamos el ejecutable

```
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\Trunc> ./Trunc
12.5
12
```

Figura 4. Ejecución

Ejecutamos el .exe, tecleamos un número y lo trunca a entero más cercano

Find String

Este código tiene la función de pedir un texto por consola, cuando se ingrese el texto validará mediante condiciones si encuentra las letras especificadas entre las comillas si encuentra las tres, muestra encontrado, si falta alguna mostrará no encontrado

Trunc.go | Trunc.exe | FindString.go | ESP-IDF: Search Error Hint

FindString > FindString.go

```go
package main
import (
    "fmt"
    "strings"
)
func main() {
    var texto string
    // El sistema ingresará la cadena aquí
    fmt.Scan(&texto)
    // Convertimos a minúsculas para asegurar que encuentre 'i', 'a', 'n'
    texto = strings.ToLower(texto)
    // Verificamos la presencia de las tres letras
    if strings.Contains(texto, "i") && strings.Contains(texto, "a") && strings.Contains(texto, "n") {
        fmt.Println("¡Encontrado!")
    } else {
        fmt.Println("¡No encontrado!")
    }
}
```

Figura 5. Código

Creamos el código que se encargará de encontrar las letras que le indicamos en la condición if que letras son las que queremos filtrar para detectar si se encontró o no

```
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\FindString> go build FindString.go
```

Figura 6. Creación de ejecutable

Creamos el archivo ejecutable

```
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\FindString> ./FindString
gian
¡Encontrado!
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\FindString> ./FindString
dana
¡No encontrado!
```

Figura 7. Funcionalidad

Lo ejecutamos y verificamos funcionalidad

Slice

Este código pide números por terminal, los cuales almacena en Slice o “rebanadas” que seria como los arrays de otro lenguaje, para ordenar los numeros ingresados en el Slice

```go
package main
import (
    "fmt"
    "sort"
    "strconv"
)
func main() {
    // Creamos una rebanada vacía de enteros
    rebanada := make([]int, 0)
    for {
        var entrada string
        fmt.Scan(&entrada)
        // Si el usuario escribe 'X' o algo que no es número, el curso suele terminar
        // Pero aquí simplemente convertimos la entrada a entero
        numero, err := strconv.Atoi(entrada)
        if err != nil {
            break
        }
        // Añadimos el número a la rebanada
        rebanada = append(rebanada, numero)
        // Ordenamos la rebanada de menor a mayor
        sort.Ints(rebanada)
        // Imprimimos la rebanada ordenada
        fmt.Println(rebanada)
    }
}
```

Figura 8. Código

Creamos el código para almacenar en una rebanada o Slice, luego se ordenarán

```
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\Slices y Sort> go build Slice.go
```

Figura 9. Ejecutable

Creamos el archivo ejecutable

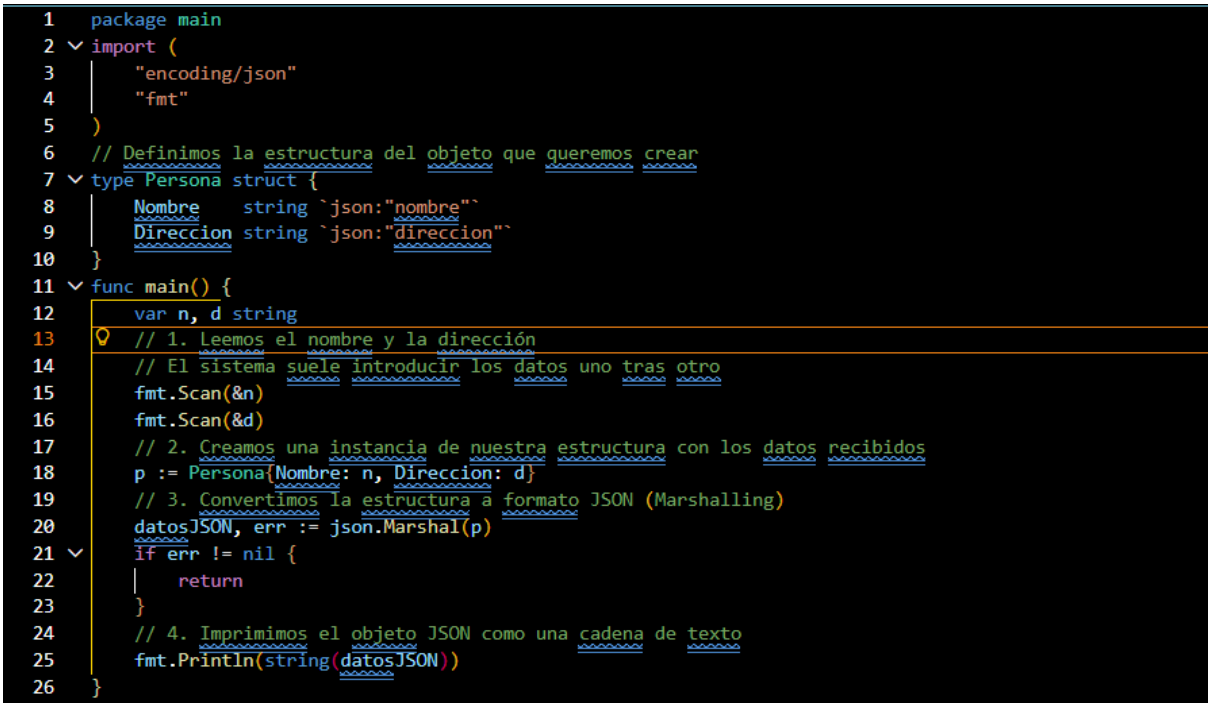
```
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\Slices y Sort> ./Slice
3
[3]
2
[2 3]
6
[2 3 6]
7
[2 3 6 7]
8
[2 3 6 7 8]
1
[1 2 3 6 7 8]
```

Figura 10. Resultado

Ejecutamos y comprobamos que los números ingresados los ordena y guarda en el Slice

JSON de ordenamiento

Código para crear un JSON que según lo ingresado en terminal mostrará un JSON ordenado con la estructura para identificar un mensaje de buena manera



```go
package main
import (
    "encoding/json"
    "fmt"
)
// Definimos la estructura del objeto que queremos crear
type Persona struct {
    Nombre    string `json:"nombre"`
    Direccion string `json:"direccion"`
}
func main() {
    var n, d string
    // 1. Leemos el nombre y la dirección
    // El sistema suele introducir los datos uno tras otro
    fmt.Scan(&n)
    fmt.Scan(&d)
    // 2. Creamos una instancia de nuestra estructura con los datos recibidos
    p := Persona{Nombre: n, Direccion: d}
    // 3. Convertimos la estructura a formato JSON (Marshalling)
    datosJSON, err := json.Marshal(p)
    if err != nil {
        return
    }
    // 4. Imprimimos el objeto JSON como una cadena de texto
    fmt.Println(string(datosJSON))
}
```

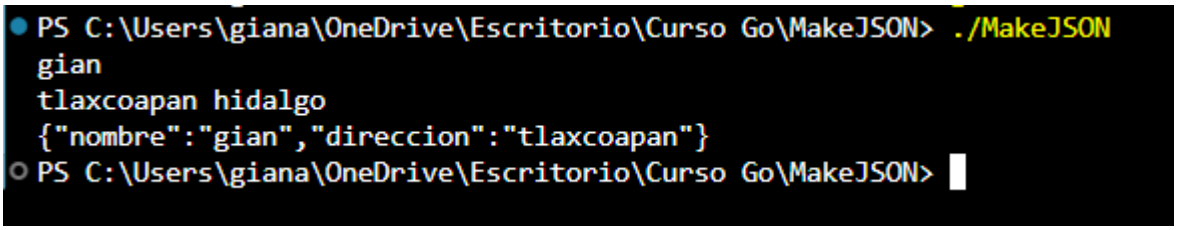
Figura 11. Código

Creamos el codigo para crear un JSON que devolvera en orden nombre y direccion

```
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\MakeJSON> go build MakeJSON.go
```

Figura 12. Ejecutable

Creamos el ejecutable



```
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\MakeJSON> ./MakeJSON
gian
tlaxcoapan hidalgo
{"nombre":"gian","direccion":"tlaxcoapan"}
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\MakeJSON>
```

Figura 13. Usabilidad

Lo ejecutamos y observamos que pidió un nombre, luego el lugar pero solo almacena sino escribimos espacios entre lo que digitalizamos en consola

Leer archivo y almacenar en estructuras

Este código se encargará de abrir un archivo de texto donde tendremos guardados nombres y apellidos como un alista, separado por un espacio como cualquier lista, el código va a pedir abrir el archivo de texto, luego leerá el texto linea por linea y detectara el nombre y apellido para guardarlo en una estructura separada el nombre del apellido

```go
package main
import (
    "bufio"
    "fmt"
    "os"
    "strings"
)
// Definimos la estructura con campos de 20 caracteres (en Go usamos string)
type Nombre struct {
    Fname string
    Lname string
}
func main() {
    var nombreArchivo string
    fmt.Print("Introduce el nombre del archivo: ")
    fmt.Scan(&nombreArchivo)
    // Abrimos el archivo
    archivo, err := os.Open(nombreArchivo)
    if err != nil {
        fmt.Println("Error al abrir el archivo:", err)
        return
    }
    defer archivo.Close()
    // Creamos la slice de structs
    personas := make([]Nombre, 0)
    // Leemos el archivo linea por linea
    scanner := bufio.NewScanner(archivo)
    for scanner.Scan() {
        linea := scanner.Text()
        partes := strings.Split(linea, " ")
        if len(partes) >= 2 {
            // Creamos el struct y limitamos a 20 caracteres si es necesario
            nuevaPersona := Nombre{
                Fname: partes[0],
                Lname: partes[1],
            }
            // Añadimos a la slice
            personas = append(personas, nuevaPersona)
        }
    }
    // Iteramos sobre la slice e imprimimos los resultados
    for _, p := range personas {
        fmt.Printf("Nombre: %s, Apellido: %s\n", p.Fname, p.Lname)
    }
}
```

Figura 14. Código

Creamos el archivo y escribimos el código

```
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\Read> go build Read.go
```

Figura 15. Ejecutable

Creamos el archivo ejecutable de nuestro codigo

```
Read > Nombres.txt
1   Juan Perez
2   Maria Garcia
3   Gian Arellano
4   Dana Castro
```

Figura 16. Lista de nombres

Creamos un archivo de texto donde guardamos nombres y apellidos como si fuera una lista

```
PS C:\Users\giana\OneDrive\Escritorio\Curso Go\Read> .\Read.exe
Introduce el nombre del archivo:
```

Figura 17. Selección de archivo

Ejecutamos el archivo .exe y va a pedirnos el nombre del archivo que va a empezar a leer

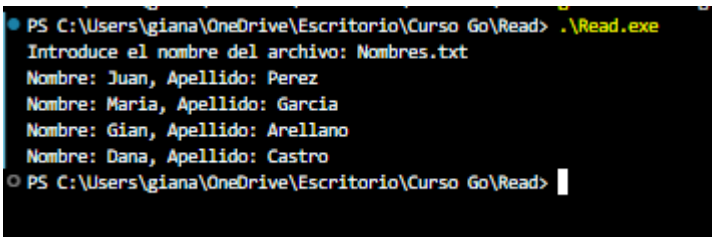


Figura 18. Usabilidad

Le damos el nombre del archivo que contiene los nombres que creamos anteriormente y se verá como leyó el archivo y separó en dos estructuras el nombre y apellido