Golanng

1. Ejercicio Go - hello world

```
package main
import ("fmt")
func main (){
    fmt.Println("hello world")
}
```

Explicaccion por lineas

1. linea: en GO cada programa es parte de un paquete , definimos esto usando **package** como palabra clave en este ejemplo el programa pertenece a el paquete **main** o principal = ***package main***
2. linea: **import ("fmt")** nos deja importar ficheros incluidos en el paquete **fmt**
3. linea: un espacio en blanco en GO es ignorado pero queda mas legible para un usuario
4. liena:func main() {} es una funcion cualquier codigo dentro de las llaves sera ejecutado
5. linea:fmt.Println() es una funcion hace posible desde el paquete **fmt** es usado en el ejemplo esto imprimira "hello world"
6. EXTRA: en GO cualquier ejecutable pertenece al paquete main

2. Declaraciones en GO

- fmt.Println(Hello World") es una declaracion
- en go las declaraciones estan separadas por finalizacion de linea presionando la tecla enter o poniendo un punto y coma ; darle al enter añade un punto y coma a el final de la frase (no se muestra en el codigo final) las llaves no pueden ponerse al principio de una linea

3. Codigo compacto en GO

- puedes escribir codigo mas compacto en go como se muestra en el ejemplo

```
package main; import ("fmt"); func main () { fmt.Println("hello world");}
```

4. Comentarios en GO

- un comentario es un texto que es ignorado en la ejecucion
- los comentarios sirven para hacer mas legible el codigo y se suele usar para explicarlo
- tambien son utiles para evitar la ejecucion de un codigo cuando se esta probando un codigo alternativo
- GO permite comentarios de una o varias lineas

5. Comentarios de una sola linea

- los comentarios empiezan con doble barra (//)
- todo codigo que este comprendido entre (//) no sera ejecutado

```
//esto es un comentario
package main
import ("fmt")
func main (){
    //esto es un comentario
    fmt.Print("hello world")
}
```

6. Comentarios de multiples lineas

- cada comentario en una linea multiple empieza con /* y termina con */
- todo texto entre estos caracteres sera ignorado

```
package main
import ("fmt")

func main() {
  /* The code below will print Hello World
  to the screen, and it is amazing */
  fmt.Println("Hello World!")
}
```

7. Variables

- en GO tenemos varios tipos de variantes por ejemplo:

> int: guarda numeros enteros como por ejemplo 123 o -321 float32: guarda numeros con decimales por ejemplo 19.99 o -19.99 string: guarda texto como "hello world" los valores de la string estan entre comillas bool: guarda booleanos : true y false

8. Declaraciones o creacion de variables

- En go hay dos maneras de declarar una variable
    1. usando la palabra clave "var"

```
var nombre_de_la_variable type = valor
```

2. con el caracter :=

```
nombre_de_la_variable := valor
```

9. Declaracion de variables con un valor inicial

- si el valor de una variable es conocido desde el inicio puedes declararlo y asignarle un valor en una linea

```
package main
import ("fmt")

func main(){
  var student1 string = "john"
  var studemt2 string = "jane"
  x := 2
  fmt.Println(student1)
  fmt.Println(student2)
  fmt.Println(x)
}
```

10. Valor asignado despues de la declaracion

- es posible asignar un valor a una variable despues de declararla , esto es muy util en casos donde la variable no se conoce

```
package main
import("fmt")

func main() {
  var student1 string
  student1 = "John"
  fmt.Println(student1)
}
```

11. Diferencias entre var y :=

- var: puede usarse dentro o fuera de funciones, la declaracion de la variable y el valor asignado pueden hacerse por separado
- := :solo se puede usar dentro de las funciones, la declaracion de la variable no puede hacerse por separado

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

var a int
var b int = 2
var c = 3
```

```
func main() {
  a = 1
  fmt.Println(a)
  fmt.Println(b)
  fmt.Println(c)
}
```

12. Multiples declaraciones en GO

- en GO es posibe declarar multiples variables en una misma linea

```
package main
import ("fmt")

func main(){
  var a, b, c, d int = 1, 2, 3, 4
  fmt.Println(a)
  fmt.Println(b)
  fmt.Println(c)
  fmt.Println(d)
}
```

- si la palabra clave type no esta especificada puedes declar diferentes tipos de variables al mismo tiempo

```
package main
import("fmt")

func main() {
  var a, b = 6, "hello"
  c, d := 7, "world"
  fmt.Println(a)
  fmt.Println(b)
  fmt.Println(c)
  fmt.Println(d)
}
```

13. Declaracion de variables en bloque

```
package main
import ("fmt")

func main() {
  var (
    a int
    b int = 1
    c string = "hello"
```

```
    )

    fmt.Println(a)
    fmt.Println(b)
    fmt.Println(c)


}
```

14. Las reglas en los nombres de las variables en GO

- una variable puede tener un nombre corto como por ejemplo (x o y) o un nombre mas descriptivo como (edad precio nombre etc...)

> las reglas de las variables son:
>
> > la variable debe emprzr por letra o por barrabaja (_)
> >
> > una variable no puede empezar con un numero
> >
> > una variable solo puede contener caracteres alfanumericos y barrabajas (a-z A-Z 0-9 y _)
> >
> > el nombre de las variables es sensible no es lo mismo (edad que Edad que EDAD)
> >
> > no hay un limite de extension en el nombre de las variables
> >
> > una variable no puede contener espacios
> >
> > en el nombre de la variable no puede ser ninguna de las palabras clave de GO

Multi-mundo Nombres de variables

- estas son algunas de las tecnicas que puedes usar para hacer las variables mas legibles
- Estilo Pascal

```
MyVariableName = "john "
```

- Estilo snake

```
my_variable_name = "john"
```

15. las constantes en GO

- las variables deben tener un valor que no se puede cambiar puedes usar const
- la palabra clave const declara que una variable es constante lo que significa que no se puede cambiar y es de solo lectura

> Sintaxis

```
const nombre_de_constante type = valor
```

16. Declarando una Constante

```
package main
import ("fmt")

const PI = 3.14

func main() {
  fmt.Println(PI)
}
```

17. Reglas de las constantes

- los nombres de las constantes siguen las mismas reglas que los nombres de las variables
- los nombres de las constantes normalmente se escriben en letras mayusculas
- constantes pueden declararse dentro y fuera de una funcion

18. Tipos de constantes

- constantes definidas
- constantes indefinidas

Constantes definidas

- son constantes que estan declaradas con un tipo definido

```
package main
import("fmt")

const A int = 1

func main() {
  fmt.Println(A)
}
```

Constantes indefinidas

- constantes indefinidas son constantes que no estan declaradas

```

```

```
package main
import("fmt")

const A = 1

func main() {
fmt.Println(A)
}
```

Constantes incambiables de solo lectura

- cuando una constante es declarada no es posible cambiar su valor despues

```
package main
import ("fmt")

func main () {
  const A = 1
  A = 2
  fmt.Println(A)
}
```

- resultado:

```
./prog.go:8:7: cannot assign to A
```

19. Declaracion de varias constantes

- varias constantes pueden ser agrupadas juntas para hacer todo mas legible

```
package main
import("fmt")

const(
  A int = 1
  B = 3,14
  C = "hola"
)
func main(){
fmt.Println(A)
fmt.Println(B)
fmt.Println(C)
}
```

20. Las funciones de salida en GO

- hay tres funciones de salida en GO

> Print()

> Println()

> Printf()

La funcion Print()

- la funcion Print() imprime los arguments en su manera por defecto

```
package main
import ("fmt")

func main(){
  var a string = "hola mundo"
  fmt.Print(a)
}
```

- si queremos imprimir los argumentos en otras lineas tenemos que utilizar \n

```
package main
import ("fmt")

func main(){
var i , j string = "hello" , "world"

fmt.Print(i "\n")
fmr.print(j "\n")
}
```

> resultado

```
hello
world
```

- tambien se puede usar Print () para multiples variables

```
package main
import ("fmt")

func main() {
  var i,j string = "hello world"
```

```
  fmt.Print(i, "\n",j)
}
```

> resultado

```
hola
mundo
```

- para un espacio entre los argumentos se usa " " por ejemplo

```
fmt.Print(i," ", j)
```

- Print tambien crea un espacio entre los argumentos si ninguno es una string

```
package main
import ("fmt")

func main() {
var i,j = 10,20

fmt.Print(i,j)
}
```

> resultado

```
10 20
```

La funcion Println

- es similar a Print() con la diferenca de que un espacio y una linea en blanco se genera entre y despues de los argumentos

```
package main
import ("fmt")

func main() {
var i,j string = "hola","mundo"

fmt.Println(i,j)
}
```

> resultado

```
hola mundo
```