**Golang – GO**

Lenguaje compilado open source, desarrollado por **Google**

Para chequear si se instaló correctamente, en cmd:

**go version**

Genera también variables de entorno para un uso más cómodo

Instalar en **Visual Studio Code** la extensión Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Se ejecuta:

Con el comando **go run archivo.go**

En caso de necesitar ejecutar varios archivos

**go run archivo1.go archivo2.go**

**go build archivo.go** 🡪 para compilar y generar el ejecutable, luego podremos hacer archivo.exe

Si tenemos problemas para ejecutar y depurar desde el visual code, usar este comando 

**Lenguaje de tipado estático** 🡪 los tipos de datos deben estar definidos y mantenerse a lo largo de la ejecución del programa

**Tipos de datos numéricos**

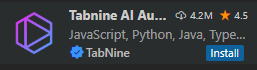
Enteros 🡪 **int**

Enteros sin signo (no permite negativos) 🡪 **uint**

Flotantes 🡪 **float32 / float64**

No puedo sumar un entero con un flotante

Extensión de Visual Studio Code que autocompleta el código



La , me permite concatenar strings con números, el signo + permite solo concatenar entre strings. Además, la coma (,) me agrega un espacio entre los objetos concatenados.

**Creación de tipos propios dentro de GO**

Texto

Descripción generada automáticamentesimilar a la definición de una clase en otros lenguajes

Texto

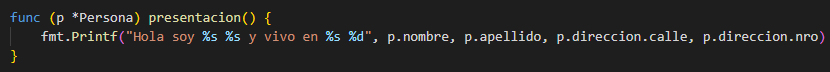
Descripción generada automáticamenteproceso similar a la instanciación de una clase (generación de un objeto de tipo persona)

Podemos incluir mas tipos propios dentro del creado

Dentro de un Printf podemos incluir variables en las comillas de la siguiente forma:

%s 🡪 strings %d 🡪 int %T 🡪 tipo de dato %v 🡪 variable

Definición de un método para la clase Persona

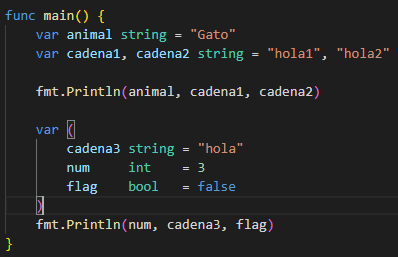


Código completo

Texto

Descripción generada automáticamente

**Definición de variables**



de esta forma el mismo compilador asume cual es el tipo que posee la variable

**Valores por defecto que toman las variables**

Numéricos 🡪 toman 0 como valor por predeterminado

Booleanos 🡪 false

Strings 🡪 cadena vacía

**Alcance o scope de las variables**

Global: todo el código

Local: dentro de una función (cuidado con las llaves)

**Constantes**

Valores que se van a mantener a lo largo de la ejecución del programa

También tienen un scope, como las variables

**Punteros**

Ubicación en memoria de una variable

Con le ampersand podemos acceder a ella 🡪 &var

Paso por valor 🡪 copia del valor

Paso por referencia 🡪 ubicación de memoria