

Multiretorno y retorno de funciones

Índice

- 01** [Multiretorno](#)
- 02** [Retorno de valores nombrados](#)
- 03** [Retorno de funciones](#)

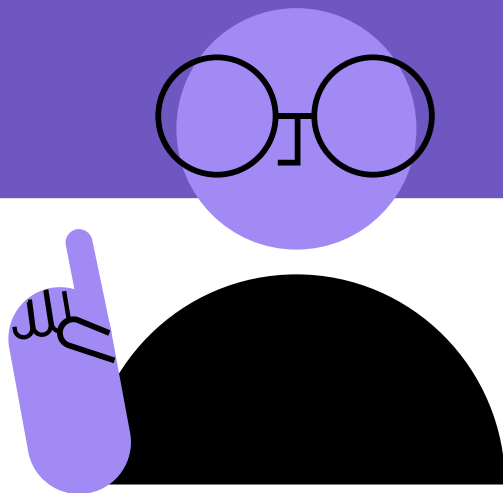


01

Multiretorno

Funciones de múltiples retornos

Una de las características que tiene Go es que podemos crear funciones que **retornan más de un valor**.



Para empezar tenemos que indicar los tipos de datos de los valores que retornarán, separados por coma y entre paréntesis.

```
func miFuncion(valor1, valor2 float64)
{
    (float64, string, int, bool)
}
```

Luego, vamos a generar una función que nos devuelva los cuatro resultados de las operaciones aritméticas: suma, resta, multiplicación y división.



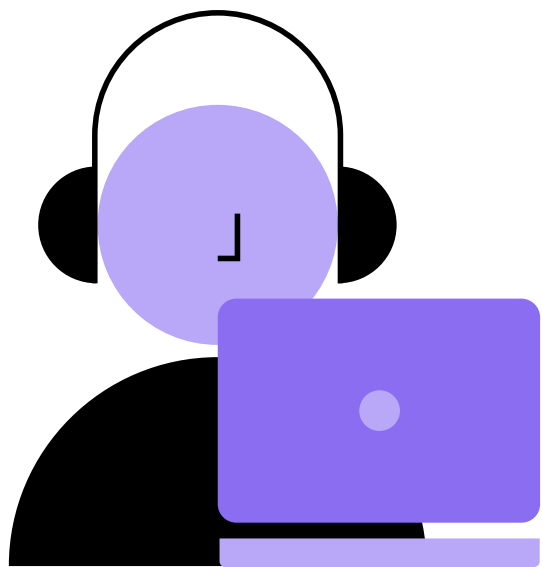
```
{}
```

```
func operaciones(valor1, valor2 float64) (float64, float64, float64, float64) {  
    suma := valor1 + valor2  
    resta := valor1 - valor2  
    multip := valor1 * valor2  
    var divis float64  
  
    if valor2 != 0 {  
        divis = valor1 / valor2  
    }  
  
    return suma, resta, multip, divis  
}
```

Al llamar a nuestra función, debemos recibir todos los valores que retorna.

}

```
func main() {  
    s, r, m, d := operaciones(6, 2)  
  
    fmt.Println("Suma:\t\t", s)  
    fmt.Println("Resta:\t\t", r)  
    fmt.Println("Multiplicación:\t", m)  
    fmt.Println("Division:\t", d)  
}
```





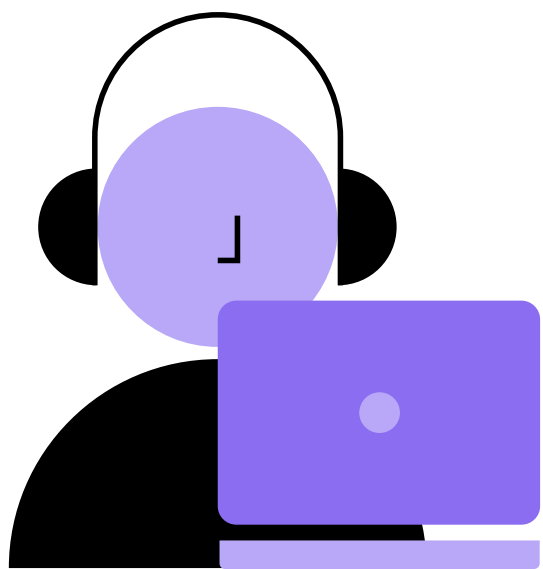
En Go el retorno de multivalores se utiliza por lo general cuando necesitamos retornar un valor y un error, y necesitamos validar si se produjo un error o no.

Para ello vamos a realizar un ejemplo de una división que nos retorna error en caso de que el divisor sea cero. Utilizaremos el paquete **errors** que nos permite trabajar con la interfaz de error.

```
{}
```

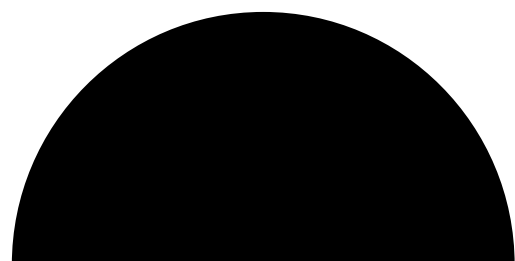
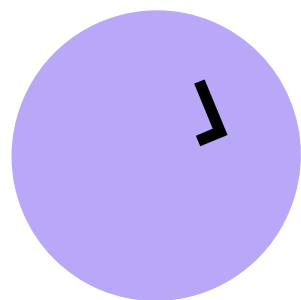
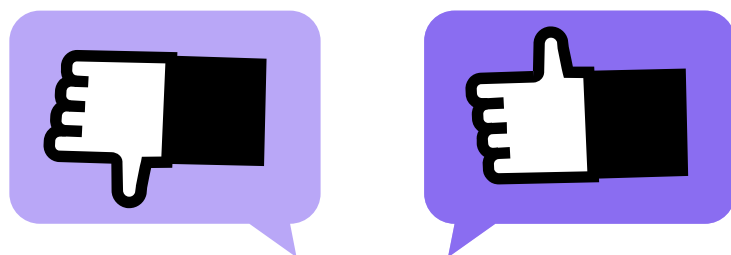
```
import (  
    "errors"  
)
```

Implementamos nuestra función división y validamos si el divisor es cero. En caso de que lo sea, retornará un error; de lo contrario, realizará la división.



```
func division(dividendo,divisor float64) (float64, error) {  
  
    if divisor == 0 {  
        return 0, errors.New("El divisor no puede ser cero")  
    }  
  
    return dividendo/divisor, nil  
}
```


Ejecutamos nuestra función **main()** y validamos si la operación fue realizada correctamente.

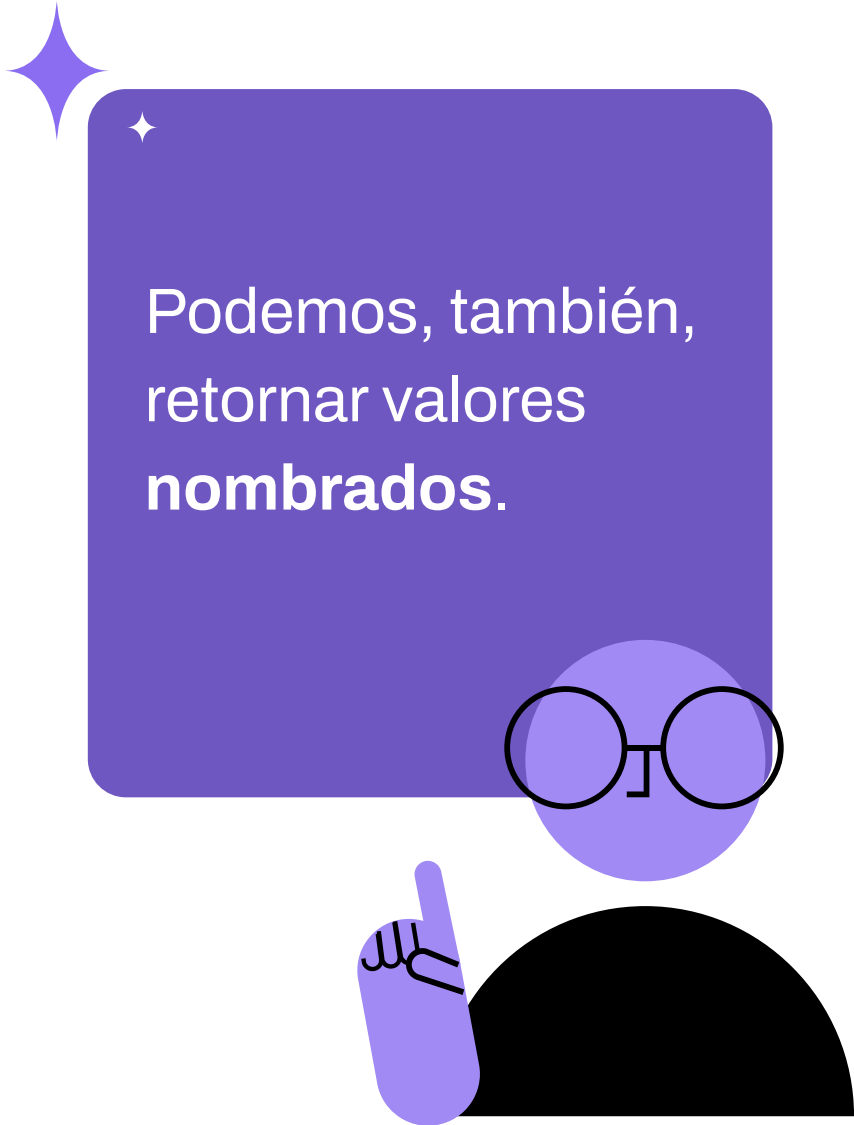


```
func main() {  
    res, err := division(2, 0)  
  
    if err != nil {  
        // Si hubo error  
    } else {  
        // Si terminó correctamente  
    }  
}
```

02

Retorno de valores nombrados

Retorno de valores nombrados



Podemos, también, retornar valores **nombrados**.

Para esto, debemos definir en la función no solo el tipo de dato a retornar, sino también el nombre de la variable.

```
{ } func operaciones(valor1, valor2 float64) (suma float64, resta float64, multip float64, divis float64)
```

Dentro de la función, tenemos que almacenar el resultado de las operaciones en dichas variables y luego hacer un return.



```
{}
```

```
func operaciones(valor1, valor2 float64) (suma
float64, resta float64, multip float64, divis
float64) {
    suma = valor1 + valor2
    resta = valor1 - valor2
    multip = valor1 * valor2

    if valor2 != 0 {
        divis = valor1 / valor2
    }

    return
}
```




De este modo, Go retornará los valores que guardamos en las variables que definimos en la función.

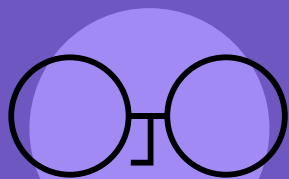
03

Retorno de funciones

Retorno de función



También podemos implementar una función que devuelva otra función.



Para ello debemos indicarle los parámetros y los tipos de datos que retorne dicha función.

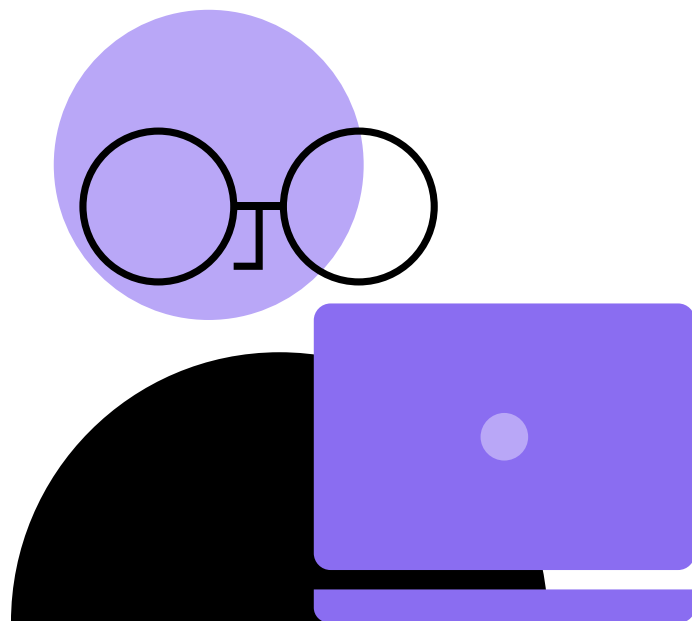
En este caso **miFuncion** nos devolverá otra función que recibe dos parámetros y devuelve un valor en punto flotante.

```
{ }
```

```
func miFuncion(valor string) func(valor1, valor2  
float64) float64
```

Veamos un ejemplo de una función a la cual le indicaremos una operación y nos devolverá una función que realice la operación pasándole dos valores numéricos como parámetros.

Crearemos una función para cada operación. Y cada una de ellas se encargará de una de las operaciones aritméticas: suma, resta, multiplicación y división.



```
{ }
```

```
func opSuma(valor1, valor2 float64) float64 {  
    return valor1 + valor2  
}  
  
func opResta(valor1, valor2 float64) float64 {  
    return valor1 - valor2  
}  
  
func opMultip(valor1, valor2 float64) float64 {  
    return valor1 * valor2  
}  
  
func opDivis(valor1, valor2 float64) float64 {  
    if valor2 == 0 {  
        return 0  
    }  
    return valor1 / valor2  
}
```

Generamos una función que se encargue de orquestar las funciones que realizarán las operaciones.



```
{}
```

```
func operacionAritmetica(operador string) func(valor1, valor2
float64) float64 {
    switch operador {
    case "Suma":
        return opSuma
    case "Resta":
        return opResta
    case "Multip":
        return opMultip
    case "Divis":
        return opDivis
    }

    return nil
}
```


Instanciamos la función indicando la operación a realizar.

Nos devolverá una función a la que le pasaremos los dos valores con los cuales queremos realizar la operación.



```
{}
```

```
func main() {  
    oper := operacionAritmetica("Suma")  
    r := oper(2, 5)  
    fmt.Println(r)  
}
```

¡Muchas gracias!