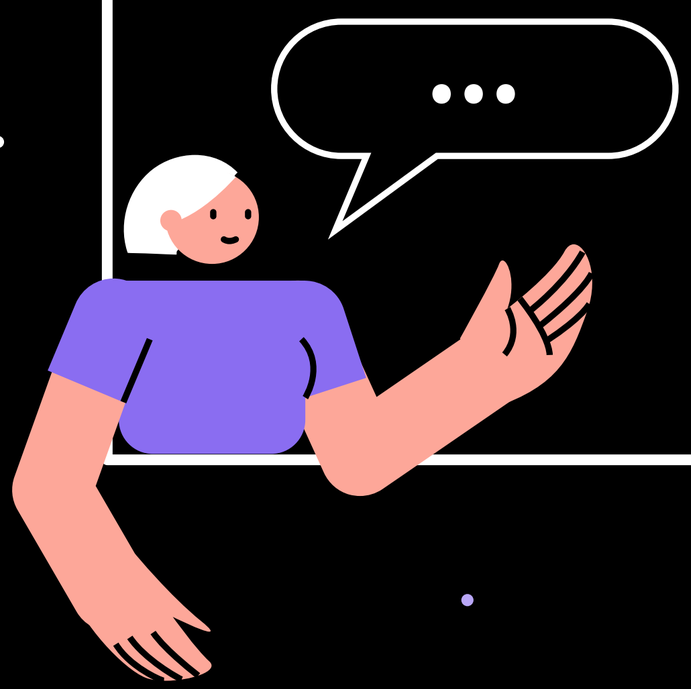
DIGITALHOUSE

Ellipsis

Certified Tech Developer

The Ultimate Degree

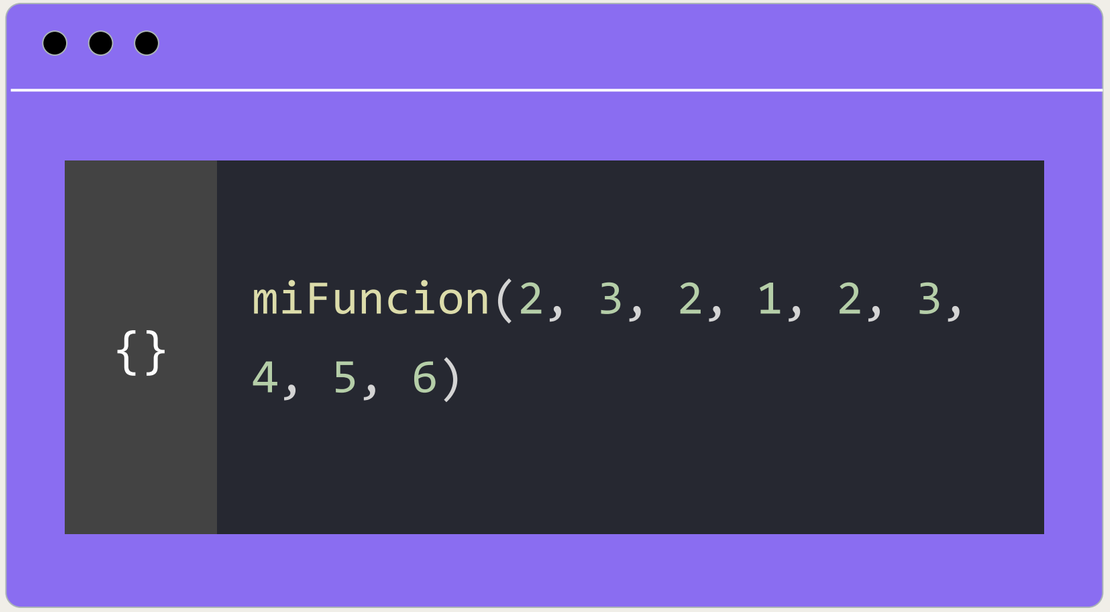
Go nos proporciona la notación de puntos suspensivos (**Ellipsis**). Esta permite que nuestras funciones reciban una cantidad dinámica de parámetros.



Notación de puntos suspensivos (Ellipsis)

Para utilizar esta notación, vamos a definir una función de la siguiente manera:

Al momento de llamar a esta función, podremos pasarle la cantidad de valores que queramos, siempre del mismo tipo de dato. Y nuestra función recibirá los parámetros como si fueran un array.



Ejemplo 1

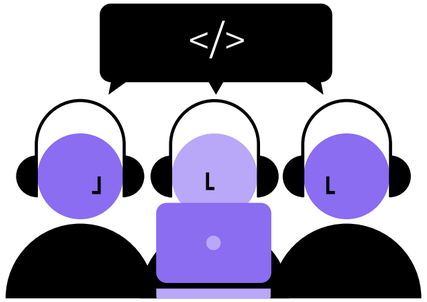
Vamos a crear una función que reciba, mediante la notación de puntos suspensivos, un número variable de valores numéricos y devolveremos la sumatoria de todos ellos.

```
{}
func suma(values ...float64)
float64 {
    var resultado float64
    for _, value := range values {
        resultado += value
    }
    return resultado
}
```

Al llamar a esta función, le podemos pasar todos los valores que queramos sumar.

```
{}
suma(2, 3, 2, 1, 2, 3, 4, 5, 6)
```

También podemos pasar otros parámetros adicionales, pero —en ese caso— el parámetro de notación de puntos suspensivos siempre tiene que estar al final.




```
func miFuncion(valor1 string, valor2 string, valores
...float64)
```

Ejemplo 2

Vamos a realizar un ejemplo más interesante que el anterior. Crearemos una función a la cual le indicaremos la operación a realizar y todos los números a los que se le realizará dicha operación.

Por ejemplo: le indicaremos a la función que queremos realizar una suma y le pasaremos todos los valores que queramos sumar.

```
{}  operacionAritmetica(Suma, 2, 3, 2, 1, 2, 3, 4, 5, 6)
```

1 Declararemos las constantes con las operaciones a realizar:

```
{}  const (
        Suma   = "+"
        Resta  = "-"
        Multip = "*"
        Divis  = "/"
    )
```

2

Luego, creamos las funciones que realizarán las operaciones.

```
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
}
```

3

También crearemos la función que se encargará de recibir la operación a realizar y los valores a los cuales se le aplicará la operación.

Por cada operación, llamaremos a una función que reciba los valores y la función que vamos a ejecutar por ese operador.

```
func operacionAritmetica(operador string, valores
...float64) float64 {
    switch operador {
    case Suma:
        return orquestadorOperaciones(valores, opSuma)
    case Resta:
        return orquestadorOperaciones(valores, opResta)
    case Multip:
        return orquestadorOperaciones(valores, opMultip)
    case Divis:
        return orquestadorOperaciones(valores, opDivis)
    }

    return 0
}
```

4

Crearemos esa función que se encargará de orquestar las operaciones:

```
func orquestadorOperaciones(valores []float64,
operacion func(value1, value2 float64) float64)
float64 {
    var resultado float64
    for i, valor := range valores {
        if i == 0 {
            resultado = valor
        } else {
            resultado = operacion(resultado, valor)
        }
    }

    return resultado
}
```

5

Por último, probamos nuestra aplicación pasándole la operación que queramos realizar.

```
{}
func main() {
    fmt.Println(operacionAritmetica(suma, 2,
3, 2, 1, 2, 3, 4, 5, 6))
}
```

DIGITALHOUSE

¡Muchas gracias!