

# PROGRAMACIÓN

---

LÓGICA, OPERACIONES BÁSICAS, BLOQUES  
FUNDAMENTALES.



# VARIABLE

---

Una **variable** está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio.

Ese espacio contiene una cantidad o información conocida o desconocida, es decir un valor.



# CONSTANTE

---

Una **constante** es un valor que no puede ser alterado/modificado durante la ejecución de un programa, únicamente puede ser leído.

Una constante corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos.

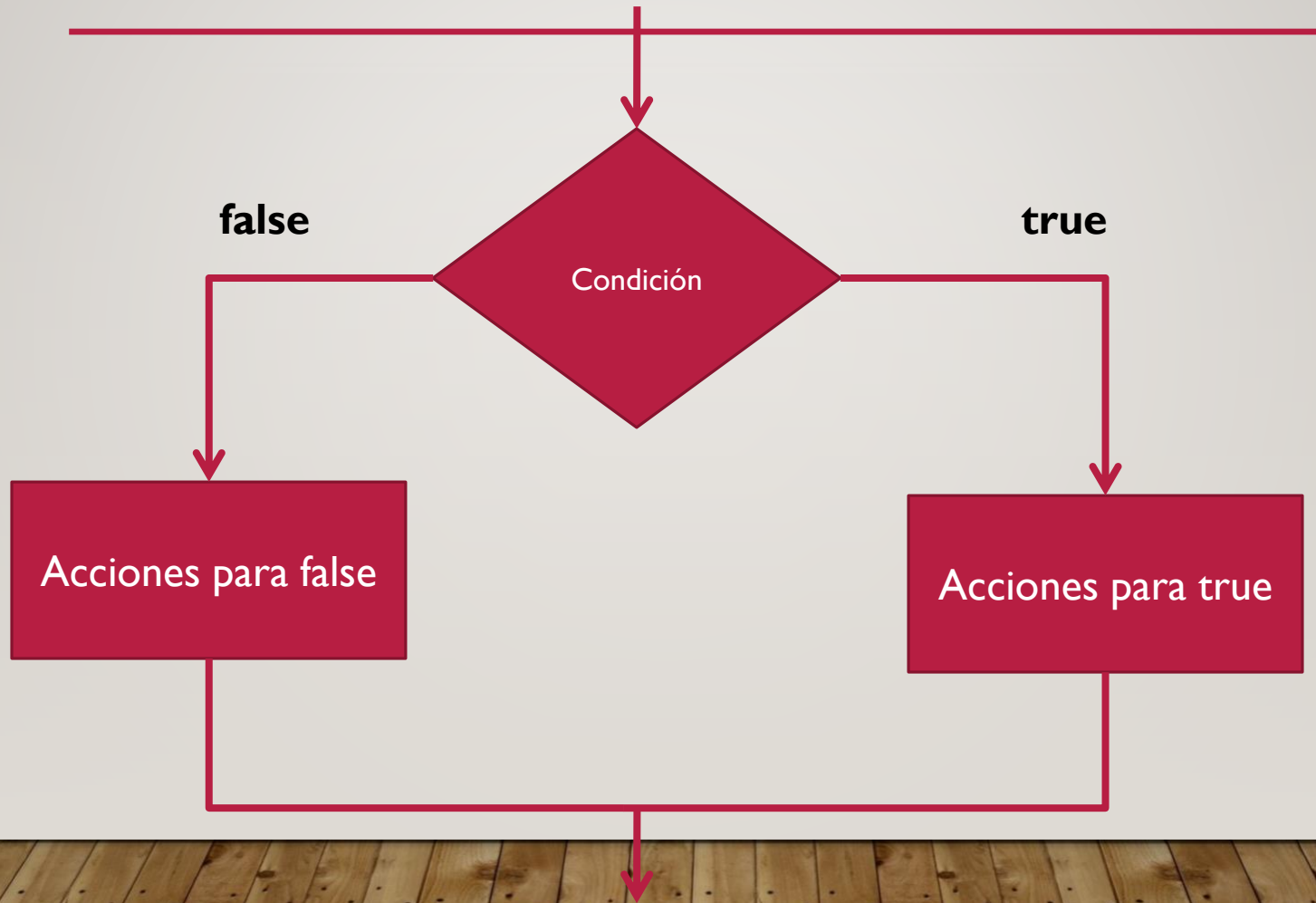


# CONDICIONALES

---

Los bloques condicionales nos permiten evaluar si una condición es verdadera o falsa y a partir de allí elegir las acciones que realizará nuestro código.

# DIAGRAMA IF



# CÓMO ESCRIBIR UNA CONDICIÓN

---

- Una condición es aquella que me retorne true o false.
- Usando operadores de comparación:
  - == (igual)
  - != o <> (distinto)
  - > (mayor)
  - >= (mayor igual)
  - <= (menor igual)



# PSEUCÓDIGO IF- EJEMPLO

---

Si edad > 18

Entonces

Imprimir (“mayor de edad”)

Sino

Imprimir (“menor de edad”)

# BLOQUE LÓGICO IF – ELSE

Bloque que se ejecuta si es **true** la condición.

```
If ( edad > 18)
```

```
{
```

```
    print("mayor de edad");
```

```
}
```

Bloque que se ejecuta si es **false** la condición.

```
else
```

```
{
```

```
    print("menor de edad");
```

```
}
```

Condición que sólo puede ser verdadera o falsa.

El bloque **else** puede omitirse si no hay acciones



# CONDICIONES COMPUESTAS

---

UNA CONDICIÓN COMPUESTA SE DA CUANDO UNA SENTENCIA IF EVALÚA MÁS DE UNA CONDICIÓN PARA DETERMINAR ACCIONES.



# LÓGICA

---

Si (user == “pepito” y pass == “1234”)

true

true

true

Si (user == “pepito” o pass == “1234”)

true/false

false/true

true

# FUNCIONES LÓGICAS

---

AND (&&)

X	Y	AND
0	0	0
0	1	0
1	0	0
1	1	1

OR (||)

X	Y	OR
0	0	0
0	1	1
1	0	1
1	1	1

Las utilizamos para crear condiciones más complejas.

# IMPLEMENTACIÓN

---

```
if(user == "pepito" and pass == "1234")  
{  
    print("Bienvenido");  
}  
else  
{  
    print("Error");  
}
```

# CONDICIONES MÚLTIPLES

---

Cuando evaluamos  $N$  valores posibles en nuestras condiciones.

Intervalos de valores.

# IF-ELSEIF-ELSE

---

```
If( numero > 100) {
```

```
...
```

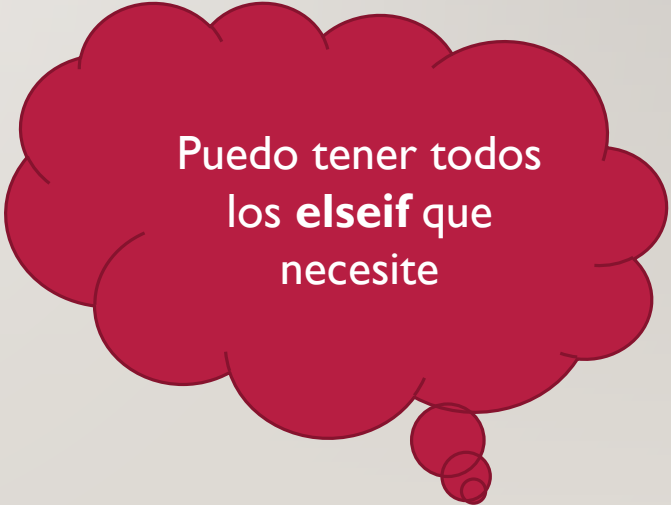
```
}
```

```
elseif(numero > 50 && numero < 100){
```

```
...
```

```
}
```

```
else { ... }
```



Puedo tener todos  
los **elseif** que  
necesite



# IF VS. ELSEIF

---

## BLOQUE DE IF

```
If(x > 10) {  
    print("mayor que 10")  
}  
  
If( x > 15) {  
    print ("mayor que 15")  
}
```

No son condiciones  
excluyentes.  
Podría ver ambos  
mensajes.

## BLOQUE ELSEIF

```
If(x > 10) {  
    print("mayor que 10")  
}  
  
Else If( x > 15) {  
    print ("mayor que 15")  
}
```

Son condiciones  
excluyentes o una o  
la otra.

# CONDICIONES EXCLUYENTES

---

¿Qué pasa si tengo  $N$  posibles condiciones excluyentes?

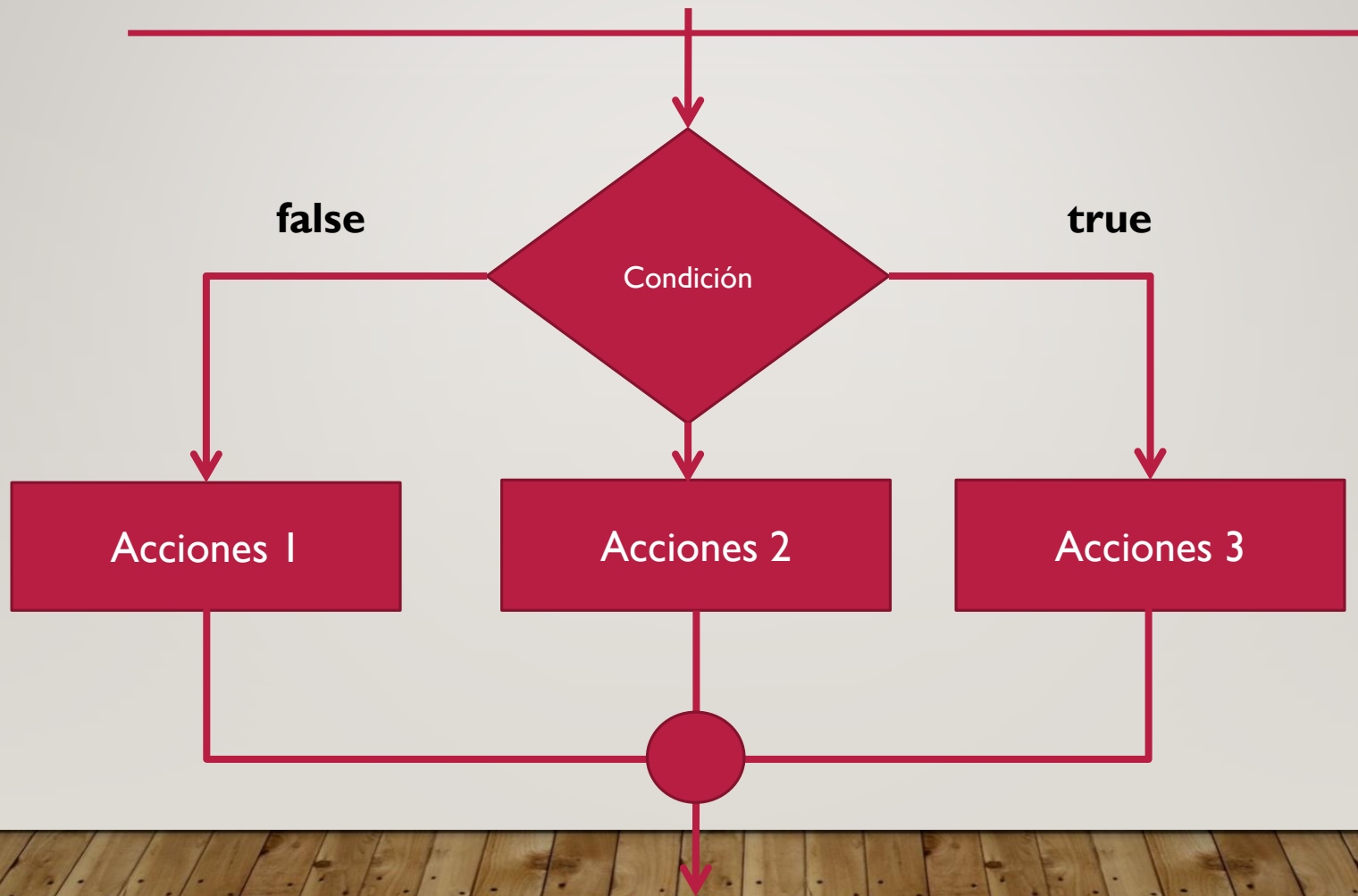


# SWITCH

---

El switch es un bloque de programación básico que permite evaluar un valor y asociar acciones a cada uno de esos posibles valores.

# DIAGRAMA SWITCH



# ¿CÓMO LO IMPLEMENTAMOS?

---

```
switch( condicion) {  
    case 1: imprimir (“valor 1”); break;  
    case 2: imprimir (“valor 2”); break;  
    default: imprimir (“valor desconocido”);  
             break;  
}
```

# ANALICEMOS PASO A PASO

---

`switch (condicion) { ... }`

Palabra reservada que  
indica el bloque.  
Similar al If.

Todas las opciones posibles.  
Las llaves siempre  
representan el alcance del  
bloque.

Lo que voy a evaluar contra  
los valores establecidos.



# Y LAS OPCIONES...

---

Case 3 : imprimir(“opción 3”); break;

Comparo la condición  
del switch con el valor  
definido.

Todas las acciones asociadas  
a ese caso (valor).

El (:) y el break, reemplazan a las llaves  
({ }) y determinan que acciones van para  
ese valor de la condición.

# ¿QUÉ PASA CON EL DEFAULT?

---

Si la condición no cumple con ninguno de los casos que definí, entonces realizo las acciones definidas por el ***default***.

Se puede ver que el default se comporta como un ***else***.

# BUCLÉS(LOOPS)

---

WHILE, FOR, DO-WHILE

# MIENTRAS...

---

Mientras condición hacer modificar condición

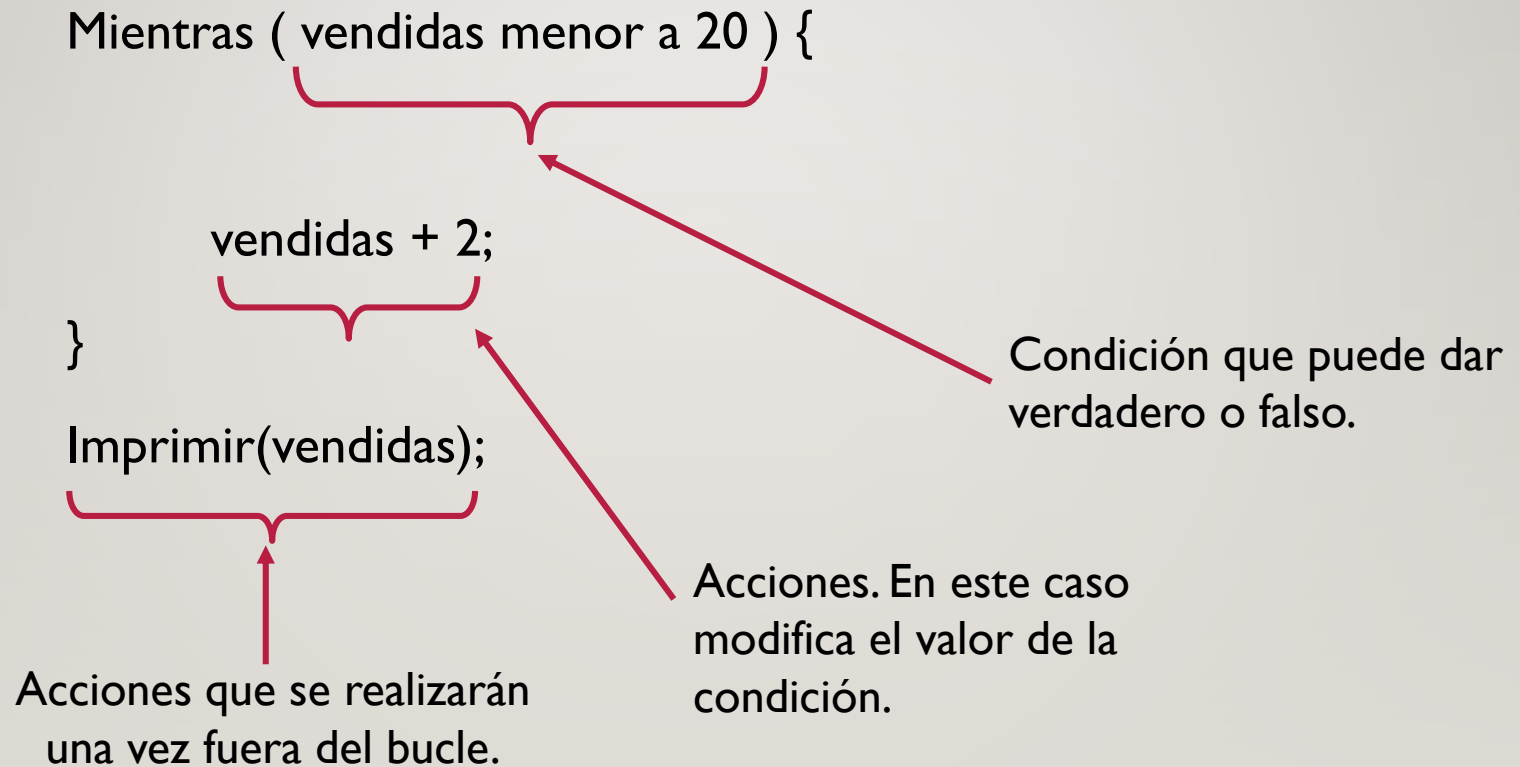
Se evalúa si es verdadera  
o falsa

De forma que deje de  
ser verdadera y poder  
dejar de repetir las  
acciones.

Todas las acciones que  
voy a realizar ***mientras*** la  
condición sea verdadera.

# LE DAMOS UN POCO DE FORMA

---



# IMPLEMENTADO

---

```
while(entradas > 20) {  
    entradas = entradas - 2;  
}
```



# ¿CUÁNTAS VECES SE EJECUTA UN MIENTRAS?

---

Ninguna o infinitas veces.

Si la condición nunca es verdadera, no entro al bloque.

Si la condición es verdadera y jamás se vuelve falsa infinitas veces (esto no es deseado).

Y si todo va bien, una cantidad de veces mientras sea verdadera la condición y luego sale.



Y SI QUEREMOS ENTRAR AL  
MENOS UNA VEZ

---

¿Cómo podría hacer eso?



# HACER - MIENTRAS

---

Hacer modificar condición    Mientras condición

Todas las acciones que voy a realizar **mientras** la condición sea verdadera. Al menos una vez, las voy a hacer.

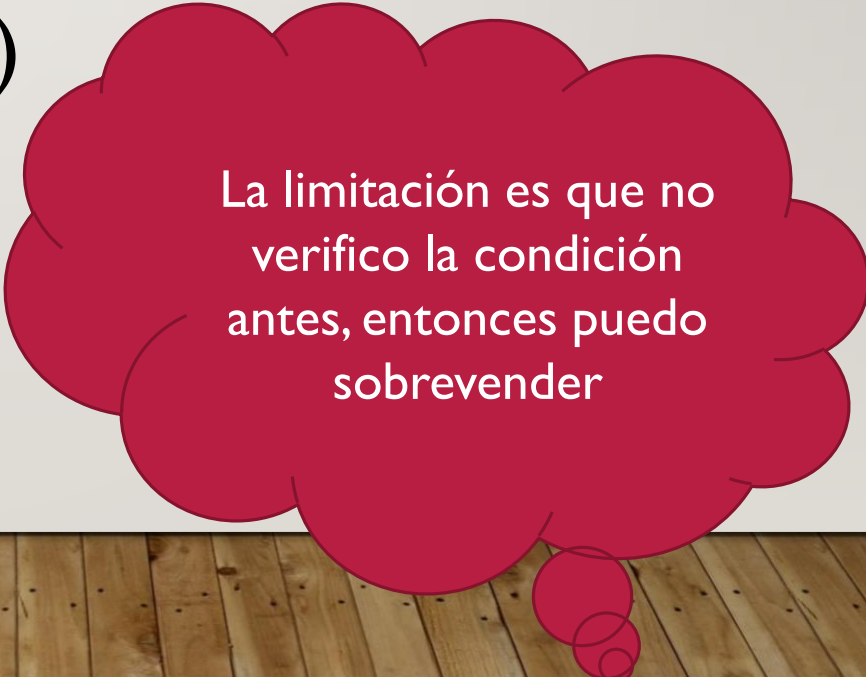
De forma que deje de ser verdadera y poder dejar de repetir las acciones.

Se evalúa si es verdadera o falsa

# COMO SE ESCRIBE

---

```
do {  
    ventas -2;  
}while(ventas > 20)
```



La limitación es que no verifico la condición antes, entonces puedo sobrevender

# FUNCIONES

---



# ¿QUÉ ES UNA FUNCIÓN?

---

Una función es un bloque de código funcional, que recibe parámetros y puede o no devolver un resultado.

Una función tiene como objetivo agrupar sentencias que pueden ser utilizadas en numerosas oportunidades dentro de nuestro código.





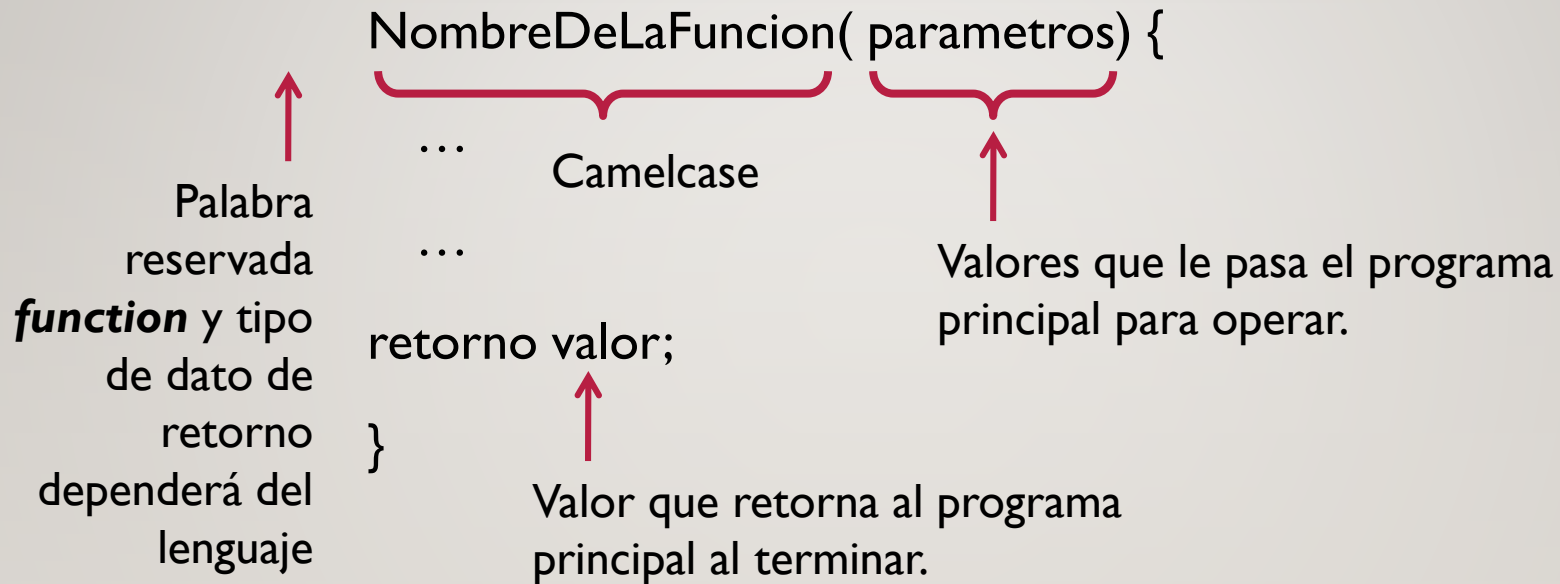
# DECLARACIÓN E INVOCACIÓN

---

Una función consta de dos instancias, la declaración en la cuál se escribe el bloque funcional y la invocación es la oportunidad en el que el programa principal hace uso de la función. Puede invocarse una gran cantidad de veces dentro del código, mientras que la declaración se realiza sólo una vez.

# COMPONENTES DE UNA FUNCIÓN

---



# VARIABLES GLOBALES Y LOCALES

---

Ahora que vimos funciones es importante diferenciar estos dos tipos de variables:

**Variable local:** declarada dentro una función, lo que implica que no existirá fuera de ella.

**Variable global:** está declarada en el programa principal y podrá ser utilizada por todos los bloques.

