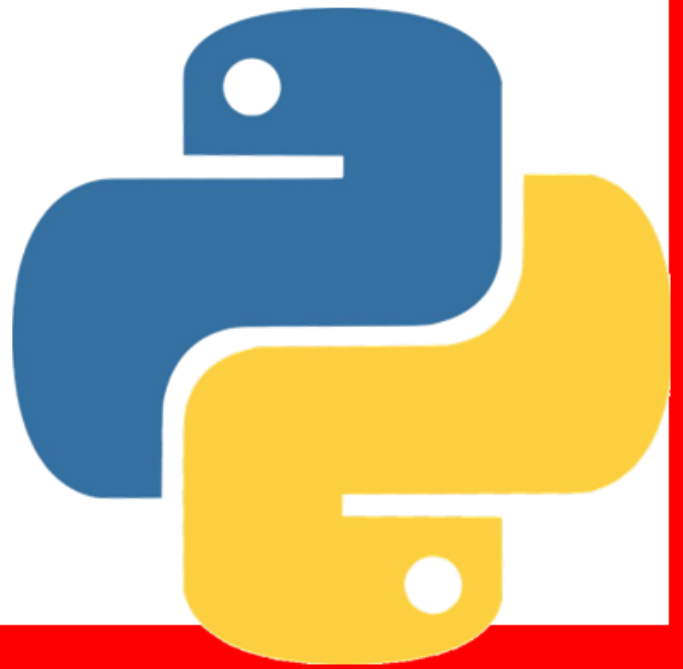
Ciclo while

06

Curso Básico

python



Estructuras iterativas

- Las estructuras o composiciones iterativas permiten **repetir una instrucción** o un **bloque de instrucciones** de manera automática.
- Las estructuras iterativas junto con las alternativas forman la base de la construcción de algoritmos y, de acuerdo al paradigma de la programación estructurada, permiten resolver cualquier problema computable.
- Las iteraciones para ejecutar el bloque de código varias veces se llaman también bucles (loops).

Estructura iterativa while (mientras)

- La sentencia while es la composición algorítmica iterativa por excelencia que sirve para cualquier esquema iterativo. Su implementación es similar en todos los lenguajes de programación.

Estructura iterativa `while`

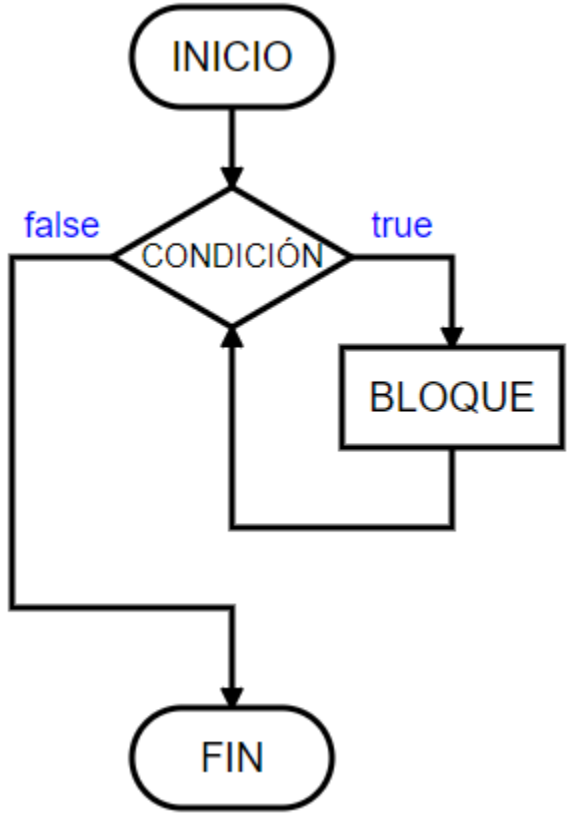
Pseudocódigo	Python	Pascal	C/C++

mientras *condición* **hacer**
  Instrucciones
 	
fin_mientras
	while *condición*:
	  Instrucciones
		while *condición* **do**
		begin
		  Instrucciones;
		end;
			while (*condición*)
			{
			  Instrucciones
			}

Estructura iterativa while (mientras)

- Un bucle while permite repetir la ejecución de un grupo de instrucciones mientras se cumpla una condición (es decir, mientras la condición tenga el valor True).

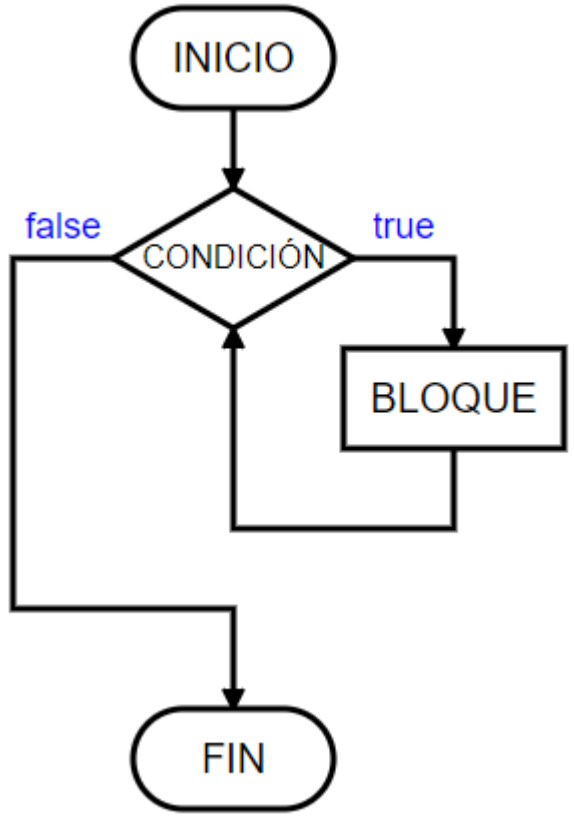
```
while condicion:
        bloque_instrucciones_del_ciclo
```



Estructura iterativa while (mientras)

```
while condicion:
        bloque_instrucciones_del_ciclo
```

```
suma=0
opc='s'

while opc == 's' or opc =='S':
    num = float( input("Introduce un número: "))
    suma+=num
    opc = input("Desea introducir otro número? S/N: ")
print("La suma es: ", suma)
```

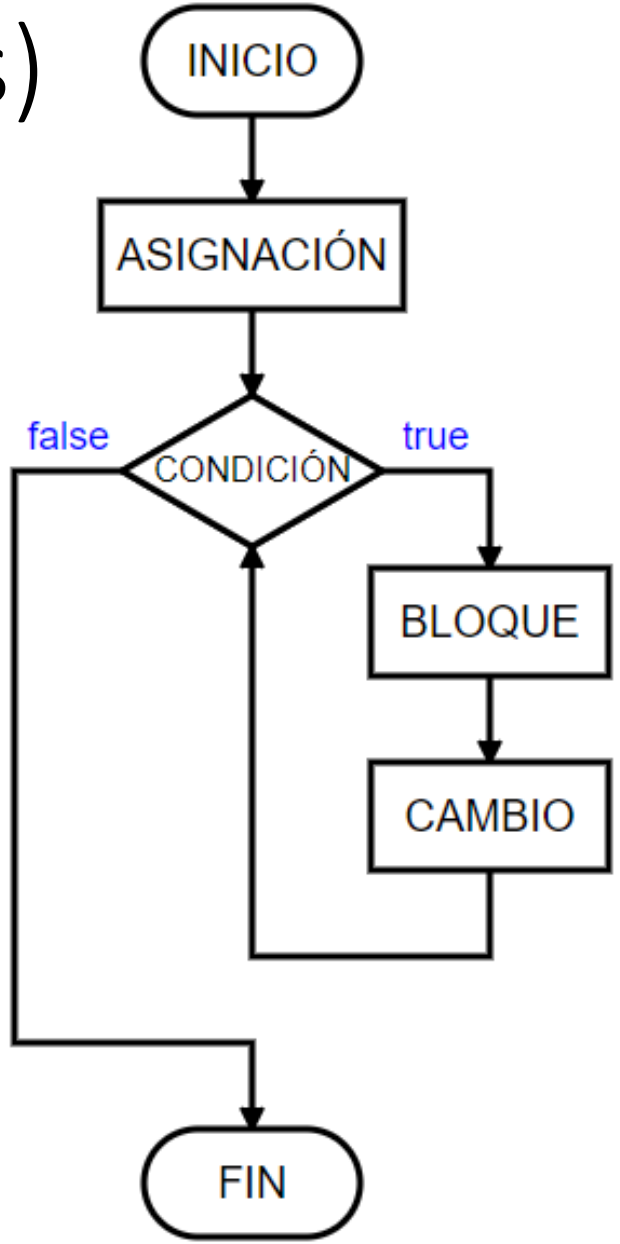


Estructura iterativa while (mientras)

Variables de control

- La variable o las variables que aparezcan en la condición se suelen llamar variables de control. Las variables de control deben definirse antes del bucle while y modificarse en el bucle while.

```
# Tabla de multiplicar (del 1 al 10) del número introducido
n = int(input('Entra un número entero: '))
k = 1
while k <= 10:
    print(n,'x',k,'=',n*k)
    k = k + 1
print('Hemos mostrado la tabla de multiplicar del',n)
```



Bucles infinitos

- Si la condición del bucle se cumple siempre, el bucle no terminará nunca de ejecutarse y tendremos lo que se denomina un **bucle infinito**.
- Para terminarlo hay que pulsar la combinación de teclas Ctrl+C.

```
i = 1
while i <= 10:
    print(i, end=" ")
```

```
i = 1
while i > 0:
    print(i, end=" ")
    i += 1
```

Bucles infinitos

- ¿Por qué este ciclo es infinito?
- Si lo sabes, deja tu respuesta en los comentarios

```
i = 1
while i != 100:
    print(i, end=" ")
    i += 2
```

Sentencias break y continue

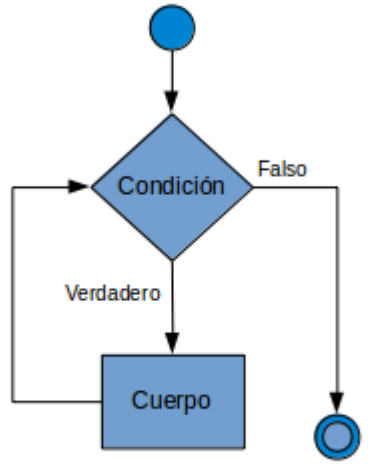
- **break** .- Aquellas instrucciones de la secuencia que están a continuación del break ya no se ejecutan y termina el ciclo for o while.

- **continue** .- Hace que se salten las instrucciones dentro del bloque de la estructura iterativa, pero solo en la iteración actual, el ciclo continuará con la siguiente iteración (for o while).

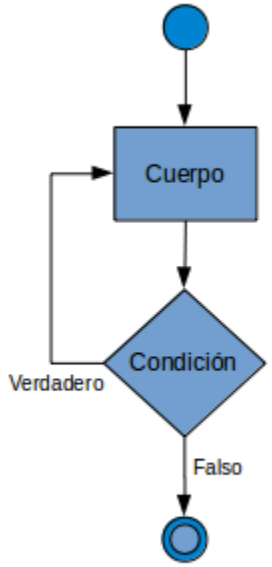
Ciclo do-while

- ¿Qué pasó con el ciclo do-while?
- ¿Como implementamos uno?

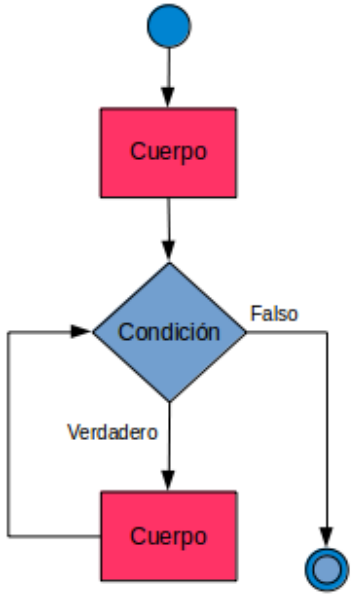
```
instrucciones_del_ciclo

while condicion:
        instrucciones_del_ciclo
```



Ciclo `while`



Ciclo `do-while`

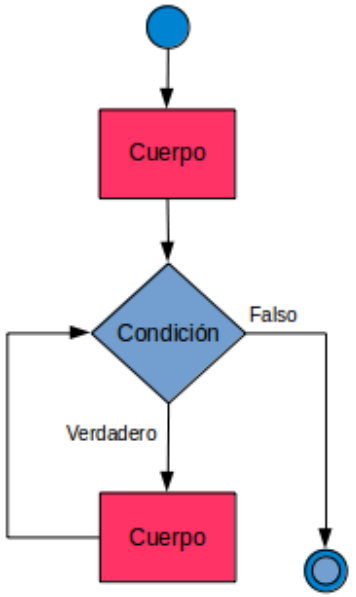


Simulación de un ciclo `do-while` mediante un ciclo `while` y duplicación del cuerpo

Ciclo do-while

- Ejemplo

```
instrucciones_del_ciclo

while condicion:
        instrucciones_del_ciclo
```



Simulación de un ciclo `do-while` mediante un ciclo `while` y duplicación del cuerpo

```
numero = int(input("Escriba un número positivo: "))
while numero < 0:
    print("¡Ha escrito un número negativo! Inténtelo de nuevo")
    numero = int(input("Escriba un número positivo: "))
print("Gracias por su colaboración")
```

Y el ciclo for:

```
# Saludo de cumpleaños
for nombre in ['Leo', 'Ronaldo', 'Andrés', 'Sergio']:
    print('Feliz cumpleaños', nombre)
print('Ya hemos saludado a los amigos')
```

*En otra ocasión, antes veremos **colecciones***

Ciclo while

06

Curso Básico

python