



# Estructura repetitiva

## While





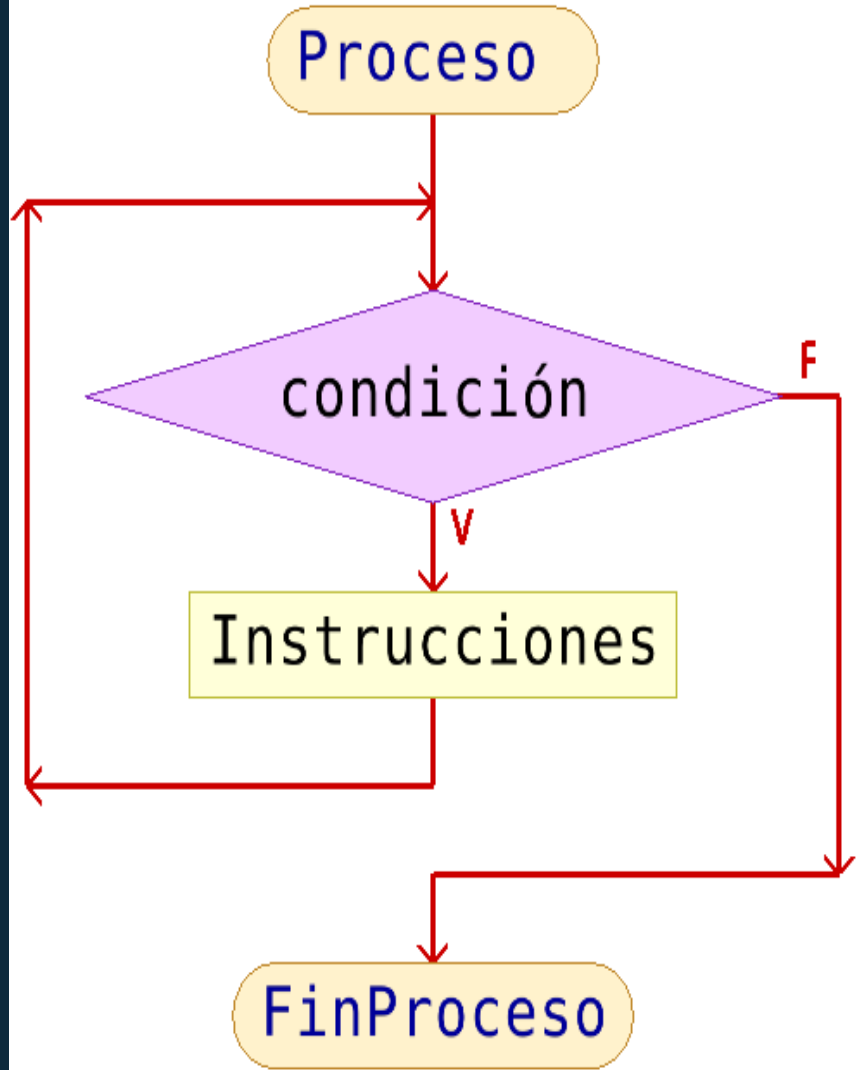
# ¿Qué es un ciclo?

Es una estructura de control esencial que permite repetir una o varias veces la misma instrucción o bloque de instrucciones de forma automática.

El número de veces que el bloque de instrucciones se ejecutará se puede especificar a través de una **condición** lógica que indica si se ejecuta de nuevo o no.



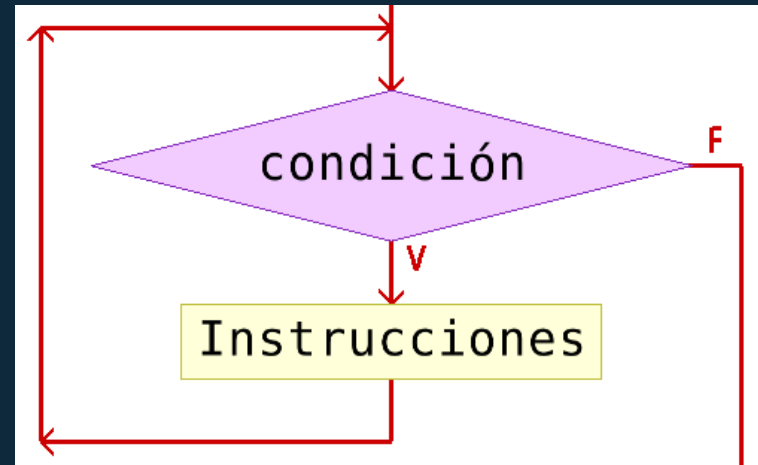
# Estructura repetitiva While



# Estructura repetitiva While

La estructura repetitiva While nos permite repetir la ejecución de una secuencia de instrucciones.

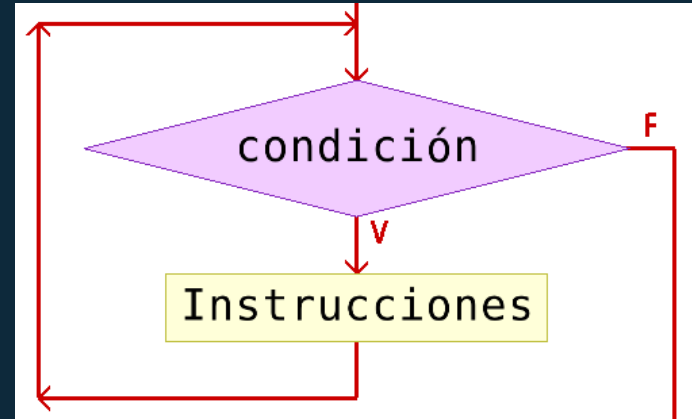
La repetición es controlada por la evaluación de una **condición**, mientras esta condición sea verdadera, entonces ejecuta las instrucciones.



# Estructura repetitiva While

Las instrucciones dentro del while se ejecutan en forma repetida, en secuencia de arriba a abajo, mientras la **condición** (*expresión lógica*) sea **verdadera**.

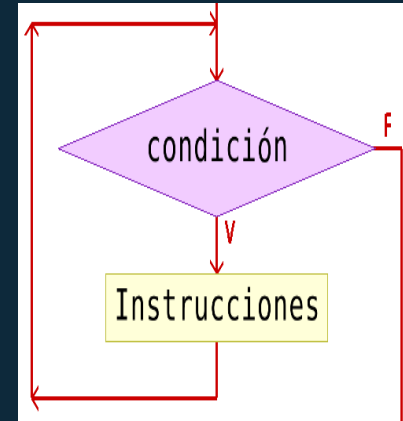
Si la **condición** (*expresión lógica*) se evalúa como **falsa** en la primera ocasión, las instrucciones dentro del while nunca se ejecutan.



# Estructura repetitiva While

La ejecución del “Estructura repetitiva While” sucede así:

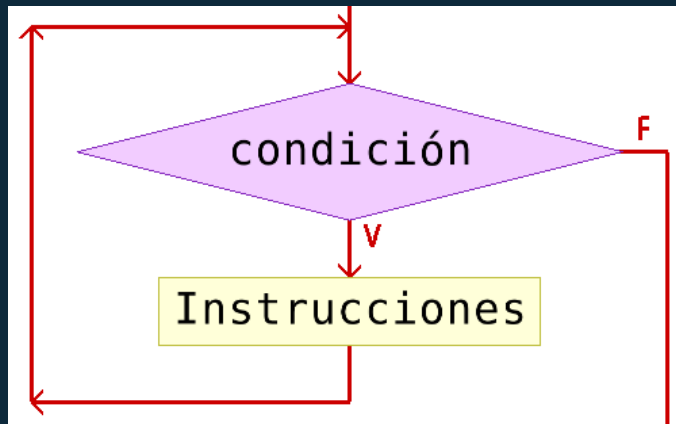
1. Se evalúa la **condición** (*expresión lógica*).
2. Si el resultado de la **condición** (*expresión lógica*) es **falso**, las instrucciones no se ejecutan y se pasa a ejecutar la siguiente instrucción en el programa.
3. Si el resultado de la **condición** (*expresión lógica*) es **verdadero**, se ejecuta(n) la(s) instrucción(es) y el proceso se repite comenzando desde el inicio del ciclo.



# Estructura repetitiva While

La estructura básica de la estructura repetitiva While en Python tiene la siguiente forma:

**while** **condición:** → Expresión lógica o booleana: condición  
#Instrucciones a repetir



A decorative graphic on the left side of the slide consists of a cluster of hexagons in various shades of blue and cyan. Some hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. A network of dots and lines is also visible. A large, solid cyan hexagon is positioned in the center of this cluster.

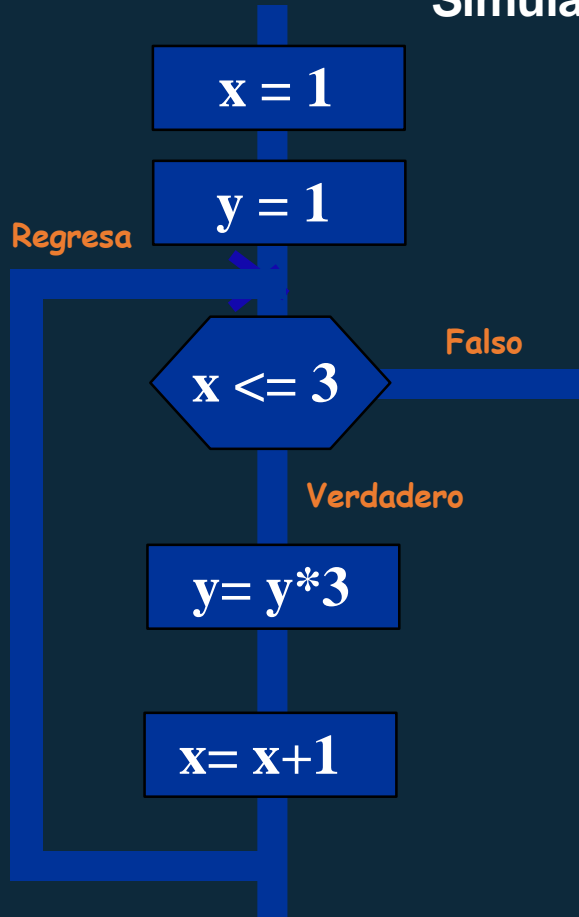
# Actividad

**Analizar la ejecución de la siguiente simulación en Python**



# Estructura repetitiva While

## Simulación de uso



x = 1

y = 1

**while** x <= 3 :

y = y \* 3

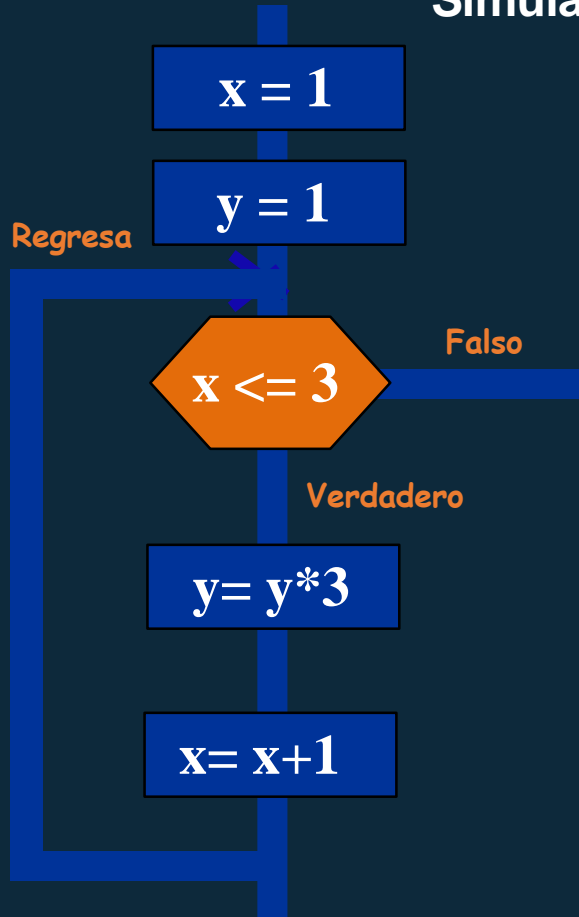
x = x + 1

y = 1

x = 1

# Estructura repetitiva While

## Simulación de uso



$x = 1$

$y = 1$

**while**  $x \leq 3$  :

$y = y * 3$

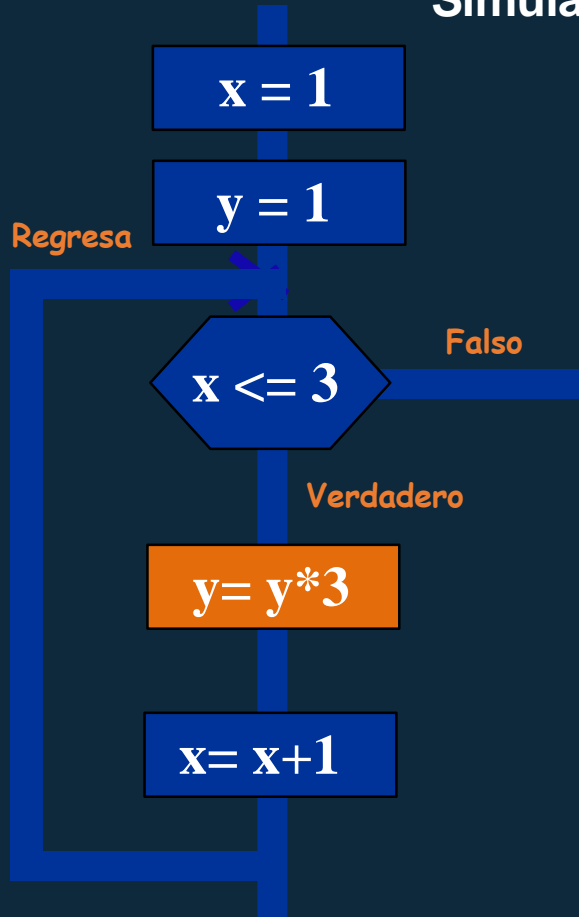
$x = x + 1$

$y =$  1

$x =$  1

# Estructura repetitiva While

## Simulación de uso



x = 1

y = 1

**while** x <= 3 :

y = y \* 3

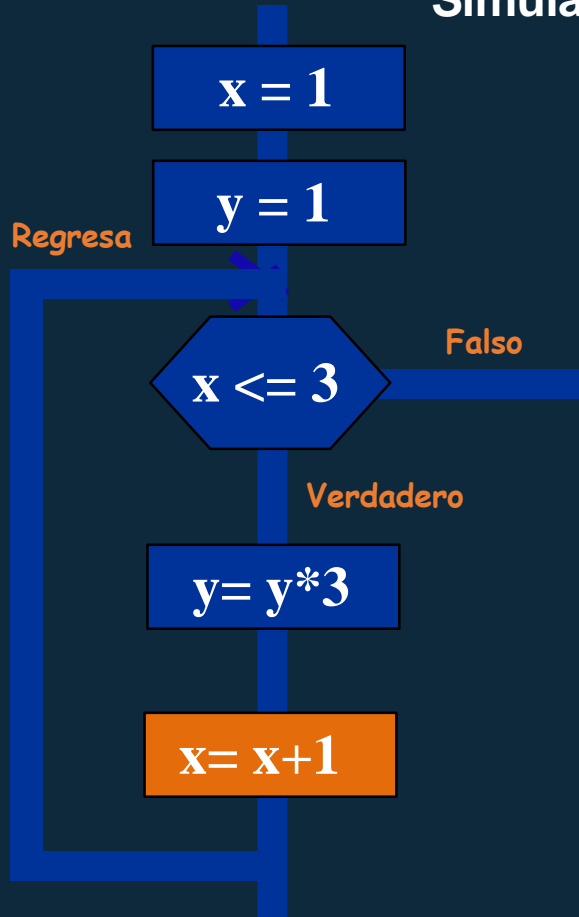
x = x + 1

y = **3**

x = **1**

# Estructura repetitiva While

Simulación de uso



x = 1

y = 1

**while** x <= 3 :

y = y \* 3

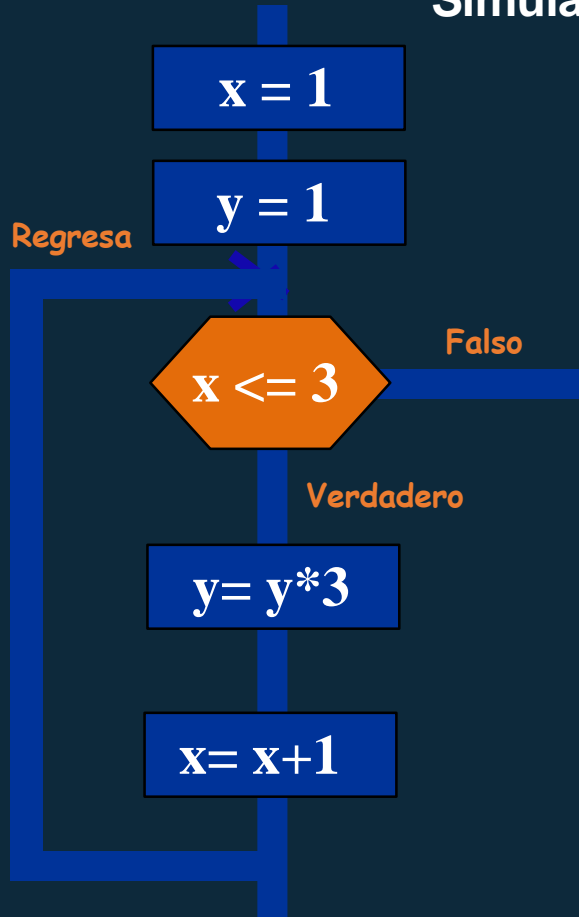
x = x + 1

y = **3**

x = **2**

# Estructura repetitiva While

## Simulación de uso



`x = 1`

`y = 1`

**while** `x <= 3` :

`y = y * 3`

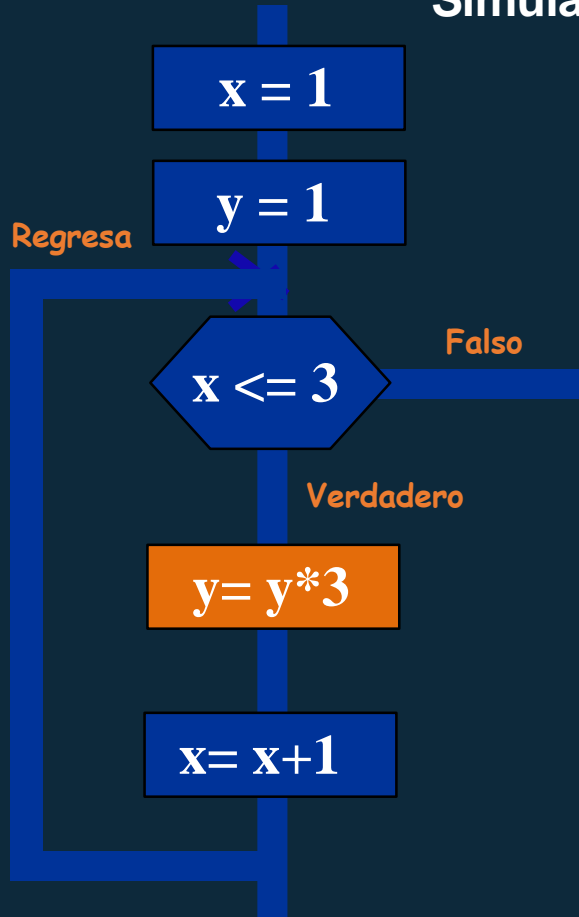
`x = x + 1`

**y =** 3

**x =** 2

# Estructura repetitiva While

Simulación de uso



$x = 1$

$y = 1$

**while**  $x \leq 3$  :

$y = y * 3$

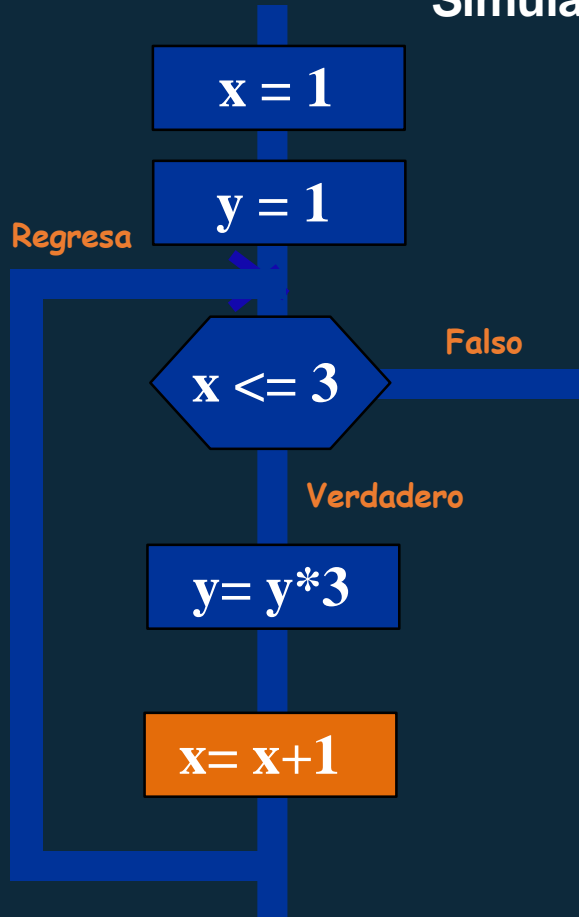
$x = x + 1$

$y =$  9

$x =$  2

# Estructura repetitiva While

## Simulación de uso



$x = 1$

$y = 1$

**while**  $x \leq 3$  :

$y = y * 3$

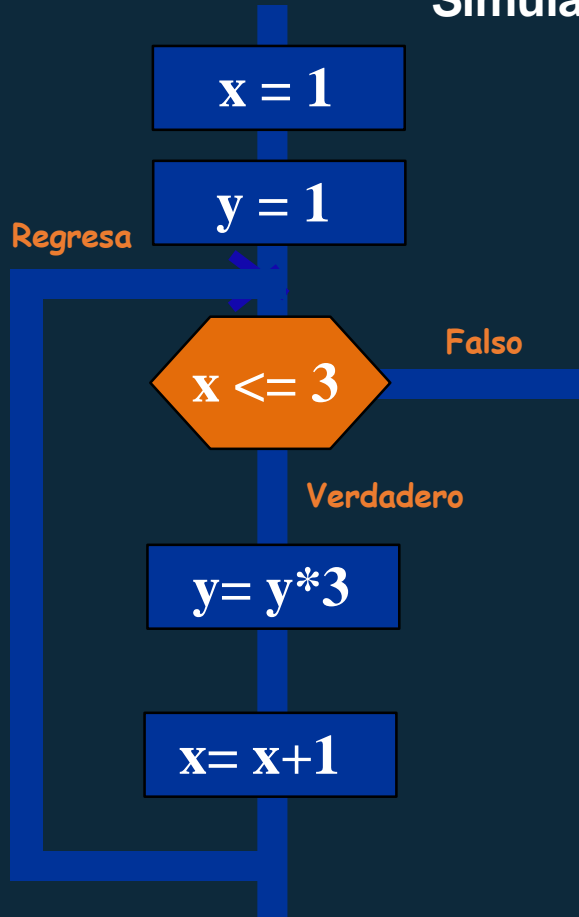
$x = x + 1$

$y =$  9

$x =$  3

# Estructura repetitiva While

## Simulación de uso



x = 1

y = 1

**while** x <= 3 :

y = y \* 3

x = x + 1

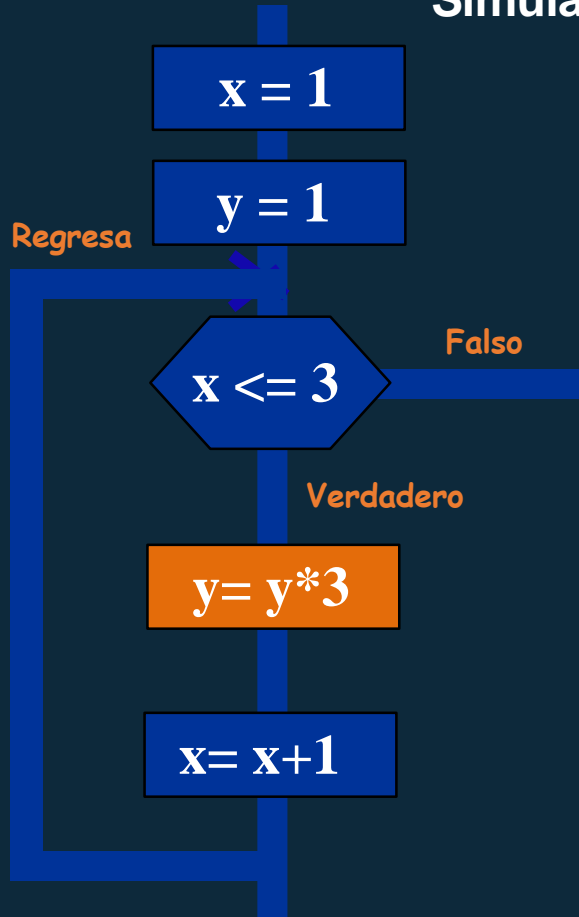
y = 9

x = 3



# Estructura repetitiva While

Simulación de uso



x = 1

y = 1

**while** x <= 3 :

y = y \* 3

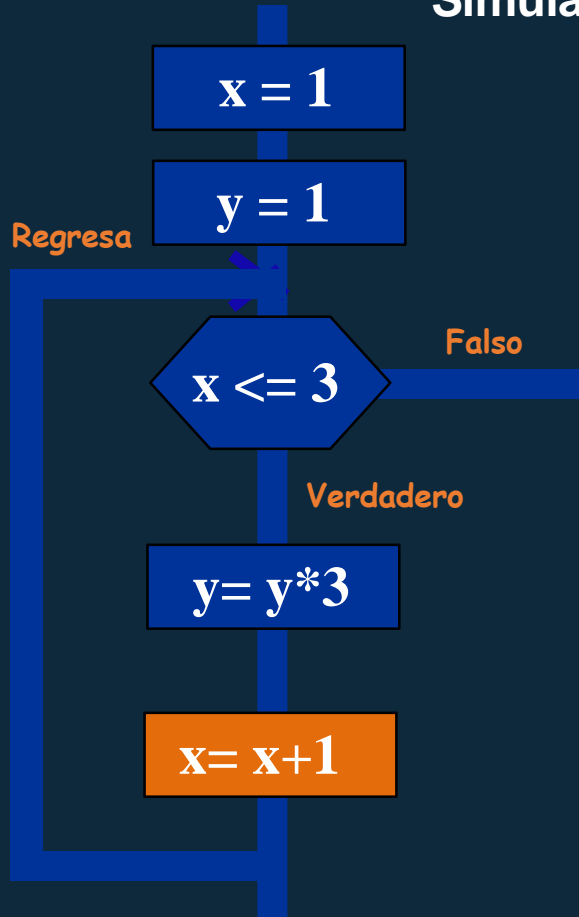
x = x + 1

y = **27**

x = **3**

# Estructura repetitiva While

## Simulación de uso



$x = 1$

$y = 1$

**while**  $x \leq 3$  :

$y = y * 3$

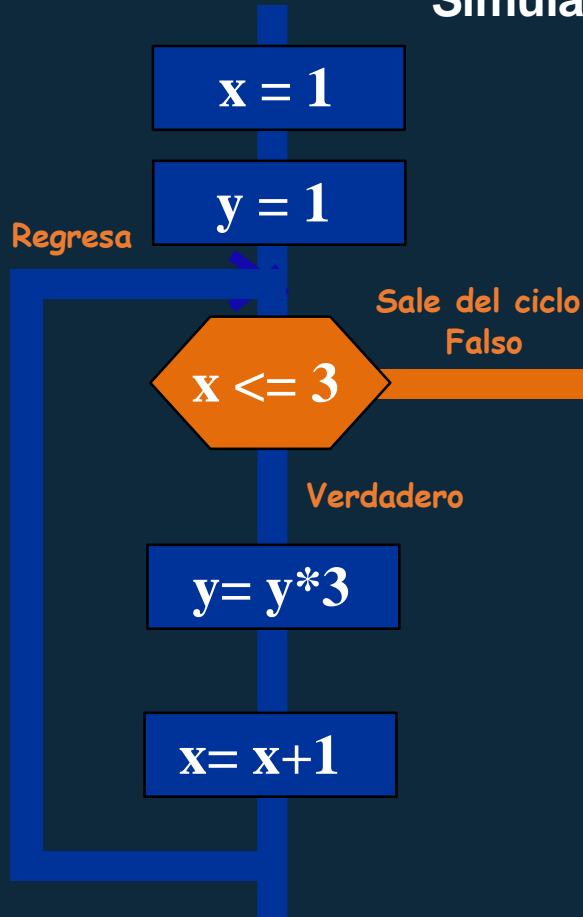
$x = x + 1$

$y =$  27

$x =$  4

# Estructura repetitiva While

## Simulación de uso



$x = 1$

$y = 1$

**while**  $x \leq 3$  :

$y = y * 3$

$x = x + 1$

$y =$  27

$x =$  4

# Actividad de reflexión

¿Qué hace el siguiente código?

1

```
x = 5
```

```
while x >= 5:
```

```
    print(" Hola a todos ")
```



# Actividad de reflexión

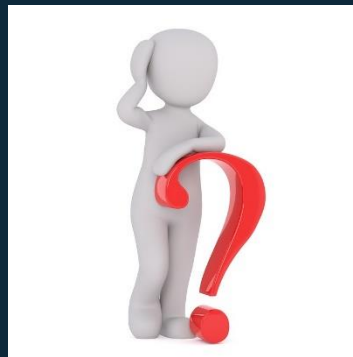
- ❖ ¿Cuántas veces se ejecutará este ciclo?
- ❖ ¿Cuál será el valor final de x?

2

**x = 10**

**while x > 0:**

**x = x - 1**



# Actividad de reflexión

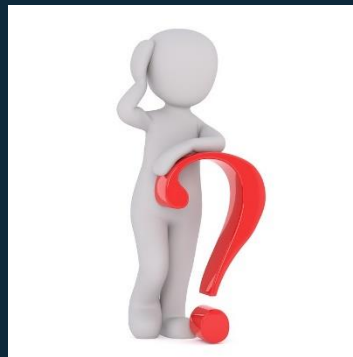
- ❖ ¿Qué hace el siguiente código?
- ❖ ¿Por qué?


3

```
x=0
```

```
while x > 0:
```

```
    print(" Hasta luego ")
```



A decorative graphic on the left side of the slide consists of a large cyan hexagon in the center. Surrounding it are several smaller hexagons of varying shades of blue and cyan. Some of these hexagons contain white icons: a lightbulb (top left), a thumbs-up (top left), a network node (top left), a smartphone (bottom left), a magnifying glass (bottom left), a gear (bottom left), and a speech bubble (bottom left).

# Operadores simplificados de operación-asignación



Operador	Ejemplo	Es lo mismo que _
<b>+=</b>	y+=10	y=y+10
<b>--</b>	y-=10	y=y-10
<b>*=</b>	y*=10	y=y*10
<b>/=</b>	y/=10	y=y/10
<b>//=</b>	y//=10	y=y//10
<b>%=</b>	y%=10	y=y%10



# Cuatro preguntas que nos debemos hacer antes de codificar un ciclo

1. ¿Qué quiero repetir?
2. ¿Cuántas veces deseo repetirlo?
3. ¿Qué debe cambiar en cada repetición?
4. ¿Qué debo hacer para que termine el ciclo y no quede como ciclo infinito?

# Contadores y Acumuladores

## Contadores

Variables que incrementan/decrementan en un valor fijo

```
cont=cont+1
```

```
cont+=1
```

## Acumuladores

Variables que incrementan/decrementan con valores diferentes:

```
acum=acum+x
```

```
acum+=x
```



# Estructura repetitiva While

En muchos casos la Estructura repetitiva While, lleva el control del ciclo a través de una variable que llamaremos **contador**, que permite controlar el número de repeticiones.

Por ejemplo, si queremos repetir una determinada instrucción cinco veces, es necesario definir una variable que vaya contando en qué pasada del ciclo se encuentra.



# Actividad grupal

Definir el programa en **Python**:

- ❖ Imprima 5 veces “hola mundo”

1



# Actividad grupal

## Programa

1

```
# Programa que imprima 5 veces "hola mundo".  
  
contador = 1  
while contador <= 5:  
    print("Hola mundo")  
    contador = contador + 1
```

Th

# Actividad grupal

Definir el programa en **Python**:

- ❖ Imprima los números del 5 al 1.



# Actividad grupal

## Programa

2

```
# Programa que imprime los número del 5 al 1.  
  
contador = 5  
while contador >= 1:  
    print (contador)  
    contador = contador - 1
```

Th

# Actividad grupal

Definir el programa en **Python**:

- ❖ Pedir al usuario un número **n** mayor o igual a 1.
- ❖ Calcular la suma de todos los números naturales desde el 1 hasta el número **n**.

$$1 + 2 + 3 + 4 + \dots + n$$

Por ejemplo, si  $n = 3$ , el resultado sería:

$$1 + 2 + 3 = 6$$





# Actividad grupal

## Programa

3

```
# Programa que calcula la suma de números del 1 a n.  
# int recibe sólo datos entero.  
  
acum = 0  
contador = 1  
n = int(input("Introduce un número entero mayor o igual a 1: "))  
while contador <= n:  
    acum = acum + contador  
    contador = contador + 1  
print(acum)
```

Th



# Ejercicios

Utiliza Thonny para codificar las siguientes funciones en **Python** y ejecútalas.

Instala **Thonny** en: <https://thonny.org/>





# Ejercicio 1

- Escriba la función **tablaMultiplicar**, que recibe un número entero e imprime la tabla de multiplicar de ese número.
- En el **script principal**, pedir un número y mandar llamar la función.

Por ejemplo: Si el número que da el usuario es 5, se deberá desplegar la tabla del 5:

5 X 1=5  
5 X 2 = 10  
5 X 3 = 15  
5 X 4 =20  
5 X 5 =25  
5 X 6 = 30  
5 X 7 =35  
5 X 8 =40  
5 X 9 =45  
5 X 10 =50

Guarda tu programa:  
**ciclos\_matricula.py**





## Ejercicio 2

- Escriba la función **numerosAscendentes**, que recibe dos números enteros: **inicio** y **fin**. La función deberá desplegar los números en orden ascendente y de dos en dos, comenzando por el número de **inicio** y terminando con el número **fin** (sin pasarse del límite).
- En el **script principal**, pedir dos números (inicio y fin) y mandar llamar la función. Validar que los datos que te proporcione el usuario sean adecuados para resolver el problema, de lo contrario, manda un mensaje de error.

Por ejemplo: si inicio es = 3 y fin = 20. Los números que se deben desplegar a pantalla son el 3 5 7 9 11 13 15 17 19.

Guarda tu programa: **ciclos\_matricula.py**





## Ejercicio 3

Escriba la función **menu**, que imprima el siguiente menú en pantalla:

1. Tabla de multiplicar
2. Escribe números ascendentemente
3. Salir

Pide una opción y **regresa el valor de la opción**. Si la opción es inválida regresar 0.

Guarda tu programa: **ciclos\_matricula.py**





# Ejercicio 4

Escriba el **script principal**, que mande llamar la función **menu** y de acuerdo a la opción seleccionada por el usuario le dé la oportunidad de ejecutar cualquiera de las funciones que han sido construidas, haciendo uso del **if - anidado**. Utiliza el ciclo **while** para que se cicle el programa hasta que el usuario introduzca la opción de salir (3).

```
opcion = 0
while opcion != 3:
    opcion = menu()
    if opcion == 1:

    elif opcion == 2:

    elif opcion == 3:
        break
    else:
        print("Opción inválida")
```

Guarda tu programa:  
**ciclos\_matricula.py**





# Situación problema 1

## Función: Comprueba clave de acceso

Escribe la función **compruebaClave(listaClaves)**, que recibe la lista de claves permitidas (listaClaves). La función debe pedirle al usuario su clave y mediante un ciclo while debe comprobar que la clave esté dentro de la lista de claves permitidas, de lo contrario, imprimir “Intenta de nuevo, introduce tu clave de acceso”, hasta que introduzca una clave válida.

En el script principal, definir la lista de claves (**listaClaves**) y manda llamar la función **compruebaClave**.





# Fuentes para consultar

- ◇ <https://www.w3resource.com/python/python-while-loop.> (While loop)
- ◇ <http://www.mclibre.org/consultar/python/lecciones/python-while.html> (Bucle While)

