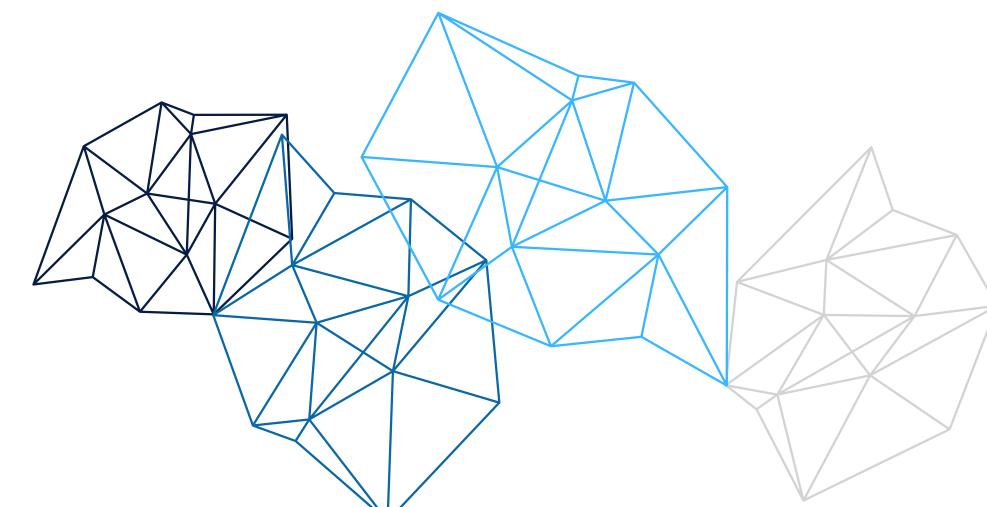
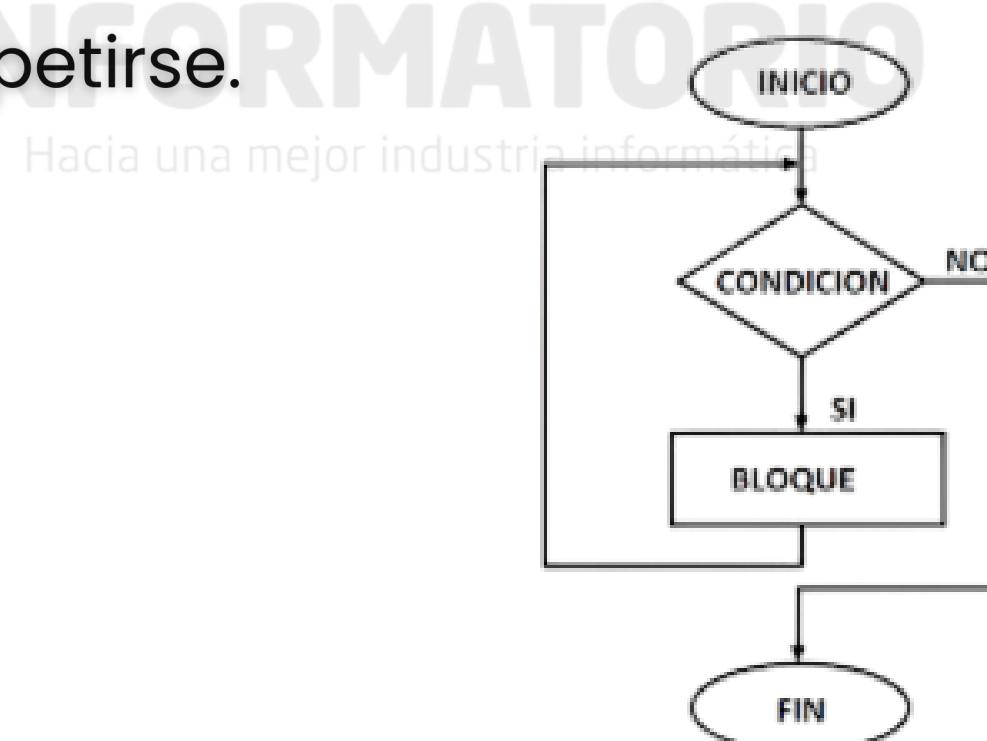


# Estructuras Repetitivas

Durante el proceso de creación de programas, es muy común, encontrarse con que una operación o conjunto de operaciones deben repetirse muchas veces. Para ello es importante conocer las estructuras que permiten repetir una o varias acciones, un número determinado de veces. Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan estructuras repetitivas, iterativas o cíclicas. Cada conjunto de instrucciones a ejecutar se denomina **BUCLE**. Y cada repetición del bucle se llama **ITERACIÓN**. Todo bucle tiene que llevar asociada una condición, que es la que va a determinar cuándo se repite el bucle y cuándo deja de repetirse.



# Antes de continuar veremos tipos de variables especiales que podríamos requerir para diseñar nuestros programas

En muchos programas se necesitan variables que cuenten cuántas veces ha ocurrido algo (**contadores**), que indiquen si simplemente ha ocurrido un evento (**banderas**) o que acumulen valores (**acumuladores**). Las situaciones pueden ser muy diversas, por lo que en este apartado simplemente se ofrecen unos ejemplos para entiendas cómo utilizar estos recursos:

- >**Contador**< Los procesos repetitivos requieren contar los sucesos y acciones internas, una forma de hacerlo es mediante un contador. Un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante en cada repetición.
- >**Acumulador**< Un acumulador o totalizador es una variable cuya función es almacenar cantidades resultantes de operaciones sucesivas. Realiza la misma función que un contador con la diferencia de que el incremento o decremento es variable en lugar de constante.
- >**Banderas**< Una bandera, es una variable que puede tomar uno de dos valores (verdadero o falso) a lo largo de la ejecución del programa y permite comunicar información de una parte a otra del mismo.

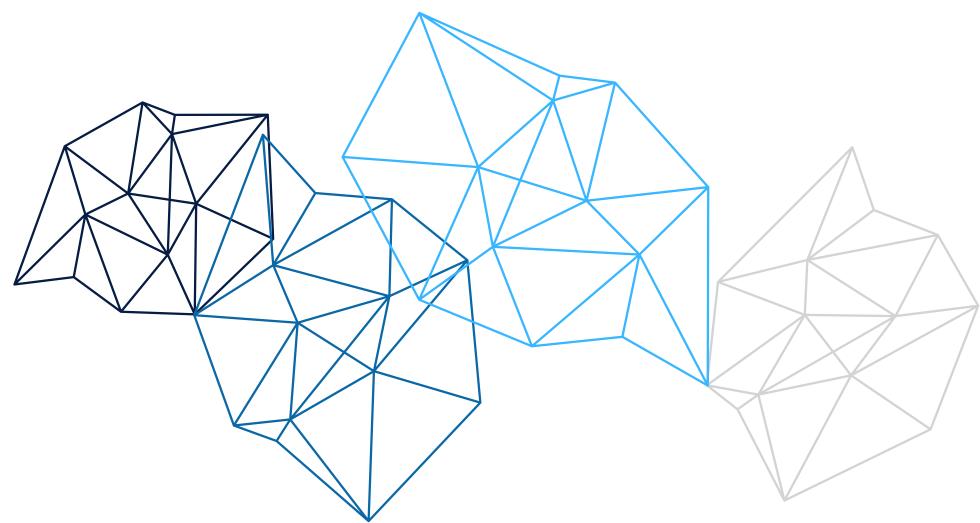
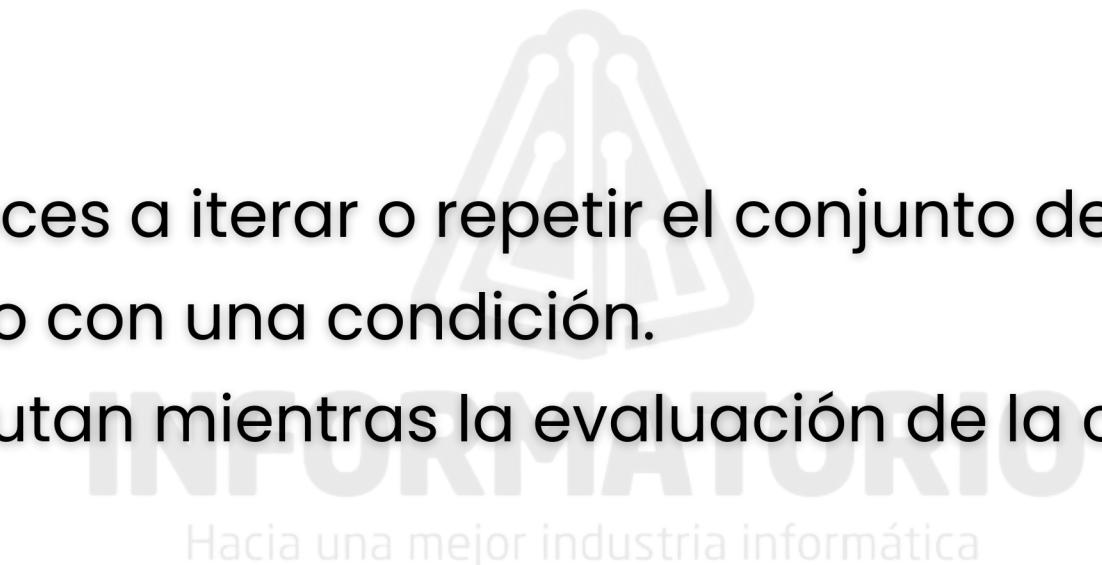
# Sentencia while (mientras)

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).

Queda en las manos del programador decidir el momento en que la condición cambie a False para hacer que el ciclo **While** finalice.

Características:

- No se conoce la cantidad de veces a iterar o repetir el conjunto de acciones
- El final del bucle está controlado con una condición.
- El conjunto de acciones se ejecutan mientras la evaluación de la condición devuelva un resultado verdadero
- El ciclo se puede ejecutar 0 o más veces.



# Ejemplo

Dado un número entero N calcular la suma de todos los números entre 1 y N.

```
>>cont = 0  
suma = 0  
N = int(input('Ingrese tope màximo: '))  
while cont <= N:  
    suma = suma + cont  
    cont = cont + 1  
print('La suma total es:', suma)
```

Resultado : (?)

Otra ventaja del bucle **while** es que el número de iteraciones no está definida antes de empezar el bucle, por ejemplo porque los datos los proporciona el usuario.



**INFORMATORIO**  
Hacia una mejor industria informática



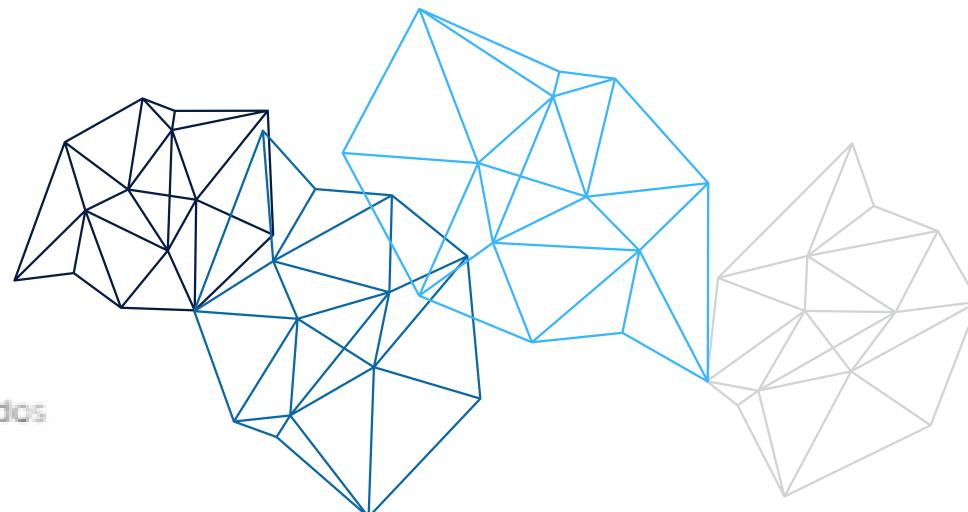
Subsecretaría de  
Empleo



Ministerio de  
Producción, Industria y Empleo



**CHACO**  
Gobierno de todos



# Uso de else en while

Se encadena al While para ejecutar un bloque de código una vez la condición ya no devuelve True (normalmente al final):

```
>>c = 0  
while c <= 5:  
    c+=1  
    print("c vale", c)  
else:  
    print("Se ha completado toda la iteración y c vale", c)
```



**INFORMATARIO**  
Hacia una mejor industria informática



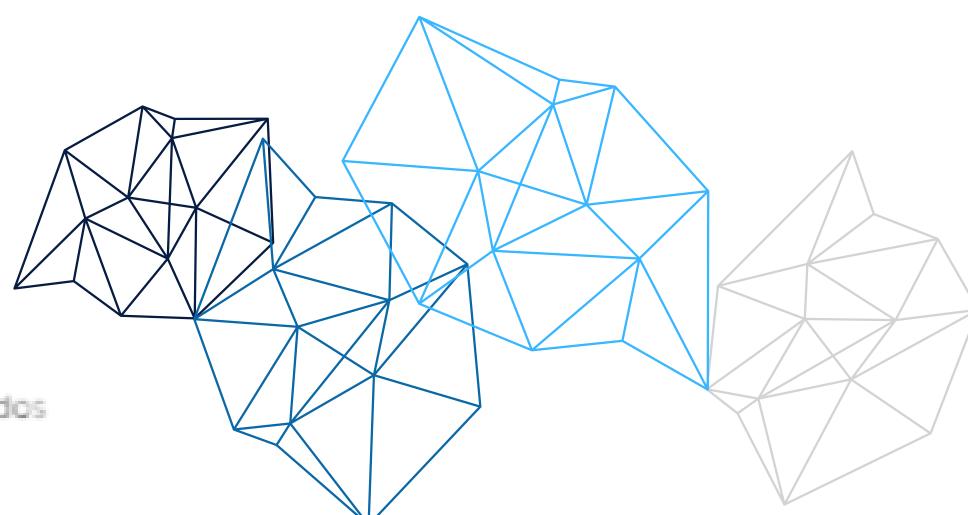
Subsecretaría de  
**Empleo**



Ministerio de  
**Producción, Industria y Empleo**



**CHACO**  
Gobierno de todos



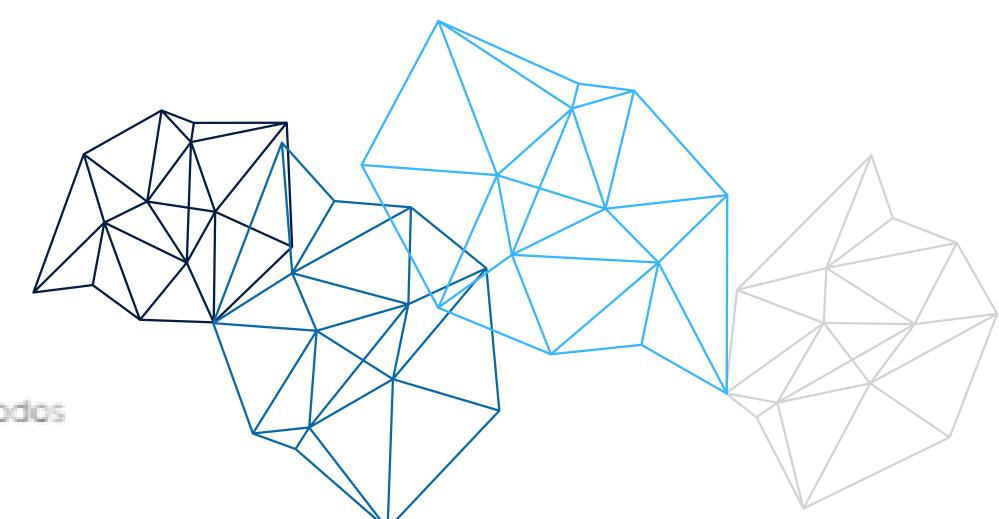
# Sentencia for (para)

- Ahora SI se conoce la cantidad de veces a iterar o repetir el conjunto de acciones
- El final de bucle está controlado un contador (indica cantidad de iteraciones).
- La variable contador que maneja el bucle se incrementa automáticamente de acuerdo al incremento indicado.
- No necesita inicialización

La sintaxis de un bucle for es la siguiente:

>>for variable in elemento iterable (lista, cadena, range, etc.):  
    cuerpo del bucle

El cuerpo del bucle se ejecuta tantas veces como elementos tenga la estructura iterable(elementos de una lista o de un range(), caracteres de una cadena, etc.).



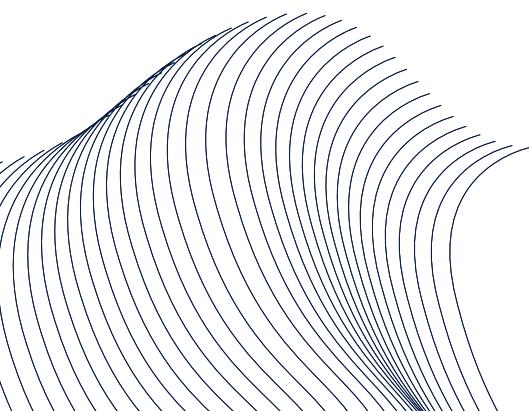
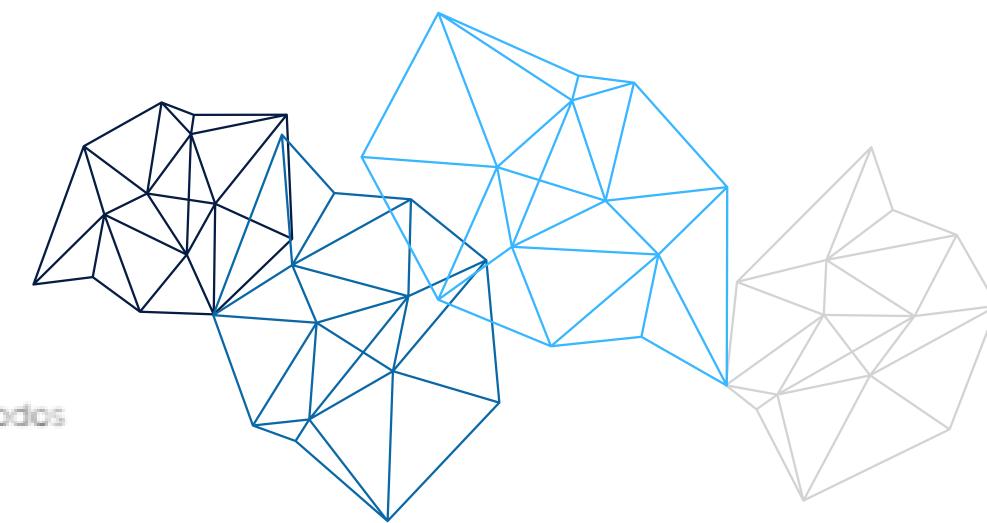
# Utilizando tipos range()

Una las ventajas de utilizar tipos **range()** es que el argumento del tipo **range()** controla el número de veces que se ejecuta el bucle. Sirve para generar una lista de números que podemos recorrer fácilmente, pero no ocupa memoria porque se interpreta sobre la marcha:

```
>>print("Comienzo")
   for i in range(3):
       print("Hola ", end="")
   print("\nFinal")
#Comienzo
Hola Hola Hola
Final
```



**INFORMATORIO**  
Hacia una mejor industria informática



Subsecretaría de  
Empleo



Ministerio de  
Producción, Industria y Empleo



**CHACO**  
Gobierno de todos

# Control de Bucles

## Instrucción **break**:

Se puede usar en bucles for y while y simplemente termina el bucle actual y sigue con la ejecución de la próxima instrucción

### a) Controlar fin de bucle

>>while True:

```
    op = input('Ingrese cualquier palabra, termina con FIN--> ')
```

```
    if op == 'FIN':
```

```
        break
```

```
    else:
```

```
        print(op)
```

```
print('Terminó la ejecución con FIN')
```

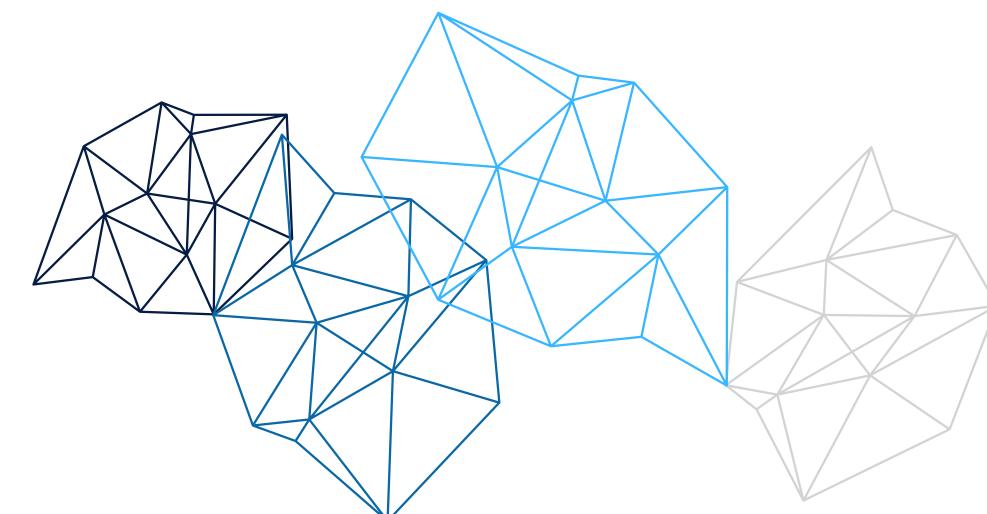
### b) La sentencia **break** es usada también para terminar un ciclo aun cuando la evaluación de la condición no devuelva False

>>for letra in "Python":

```
    if letra == "h":
```

```
        break
```

```
print("Letra actual :", letra)
```



# Control de Bucles

Sentencia **continue**:

Al aparecer un **continue** en Python, este regresa al comienzo del bucle, ignorando todos los estamentos que quedan en la iteración actual del bucle e inicia la siguiente iteración.

Queda más claro con un ejemplo:

```
>>for letra in "Python":  
    if letra == "h":  
        continue  
    print("Letra actual :", letra)
```

Otro ejemplo

```
>>var = 10  
while var > 0:  
    var = var -1  
    if var == 5:  
        continue  
    print("Valor actual de la variable :", var)
```



**INFORMATORIO**  
Hacia una mejor industria informática

