

«Talento Tech»

# Data Analytics con Python

Clase 03





## Clase N° 3 | conceptos básicos

### Temario:

- Bucles en Programación: Concepto y Aplicaciones
- Tipos de Bucles en Python: for y while Explorados y Comparados





## Bucles

### ¿Qué es un bucle?

Un bucle es una estructura de control en programación que permite ejecutar un bloque de código repetidamente. Es especialmente útil cuando necesitas realizar una misma acción múltiples veces o cuando quieres iterar a través de una colección de datos, como una lista o un rango de números.

### ¿Para qué sirve un bucle?

Los bucles son esenciales para automatizar tareas repetitivas y manejar grandes volúmenes de datos de manera eficiente. Por ejemplo, puedes usar un bucle para sumar todos los números en una lista, procesar elementos de un archivo de texto línea por línea, o generar una serie de números.

## Entendiendo el concepto de iteración

La iteración se refiere al proceso de repetir una serie de instrucciones. En el contexto de los bucles, cada repetición se llama "iteración". Un bucle puede iterar sobre una secuencia de elementos, como una lista, y ejecutar el mismo bloque de código para cada elemento. Por ejemplo, si tienes una lista de nombres y quieres imprimir cada nombre en la lista, el bucle iterará sobre cada nombre, ejecutando la misma instrucción de impresión para cada uno.

En resumen, los bucles y la iteración son herramientas poderosas que te permiten escribir código más corto, eficiente y fácil de mantener, evitando la repetición de instrucciones de manera manual.

## Bucle for

El bucle for se usa para iterar: realizar varias veces, una acción, sobre elementos que tengan una secuencia. Estos elementos por propiedad, serán iterables.

Iterar en Python significa realizar una acción repetidamente sobre un conjunto de elementos. Es como ejecutar una tarea varias veces de manera automática en tu programa.

Por ejemplo, si necesitas realizar una acción con cada elemento de una lista o ejecutar un proceso varias veces, utilizas la iteración para recorrer esos elementos uno por uno y realizar la acción deseada para cada uno.

Este bucle se usa cuando se tiene un fragmento de código que se quiera repetir "n" veces, por lo que el número de iteraciones de un for está definido de antemano.

## Sintaxis del bucle for

La sintaxis tiene como elementos a un objeto que será iterado y a una variable donde iremos guardando los elementos de la secuencia.

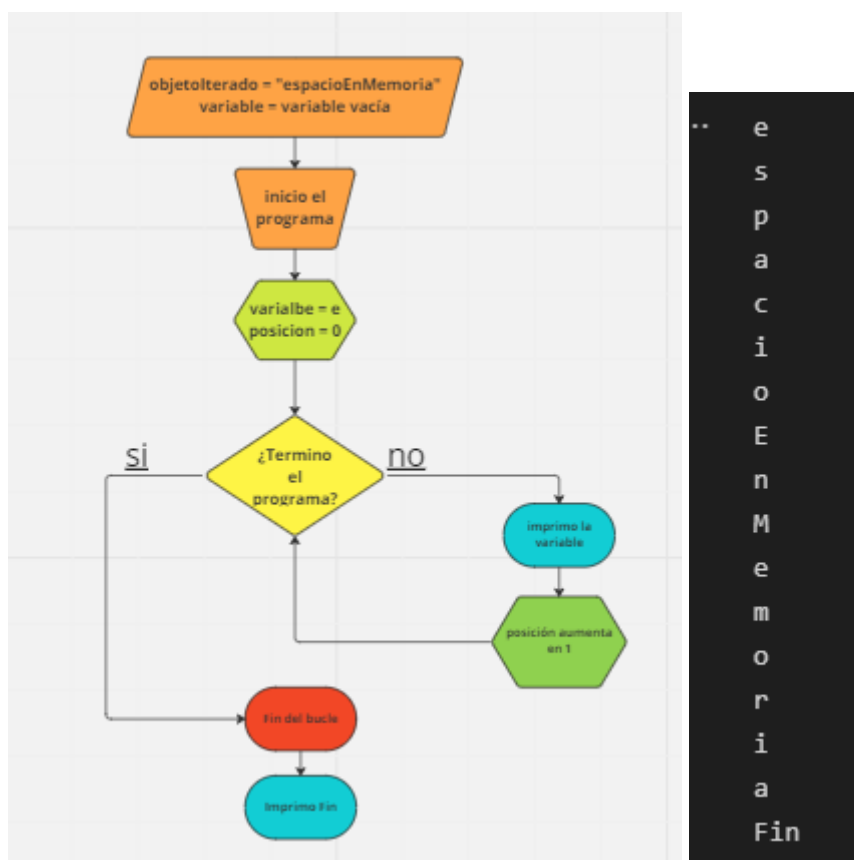
```
Código
##
for variable in objetoIterado:
    Declaracion
##
Ejemplo
##
objetoIterado = "espacioEnMemoria"

for variable in objetoIterado:
    print(variable)
print("Fin")
##
```

En este caso el objeto que iteramos es la variable "objetoIterador" que es una variable de tipo String, que tiene el valor de "espacioEnMemoria". Este es un objeto iterable el cual tiene una posición para cada uno de sus elementos.

objetoIterado	e	s	p	a	c	i	o	E	n	M	e	m	o	r	i	a
Posición	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Lo que hacemos es recorrer el objeto iterado e imprimir cada uno de los elementos que guardamos en la variable. Como lo que guardamos en la variable es cada elemento del objetoIterado, es decir la cadena “espacioEnMemoria”, y está formada por letras que forman una cadena de caracteres, lo que se guarda e imprime es letra por letra tantas veces como posiciones tenga el elemento.





Mientras la posición de la variable no sea mayor, en este caso, a 15, se imprimirá el valor de la variable.

También podemos iterar una acción "x" cantidad de veces usando la función range.

Código:

```
##  
for x in range(0, 10, 2):  
    print(x)
```

##

Consola:

##

0

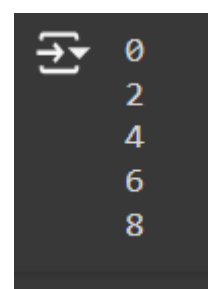
2

4

6

8

##



En este código estamos diciendo que en x guardaremos en memoria los valores que vayan desde 0 hasta 10, yendo de dos en dos. Los valores que tenemos por consola serán entonces, 0,2,4,6,8 sin incluir el 10. Esto se debe a que una vez que llega a diez, el programa se corta y no se llega a ejecutar el código. Una solución a esto es siempre poner un +1 después del 10 o poner directamente 11.



## Enumerate

Vimos que cada objeto iterable tiene una posición para cada uno de sus elementos.

¿Qué pasa si queremos obtener no solo el valor del objeto iterable sino también su posición?

¿Para qué nos sirve?


Para esos casos, existe la posibilidad de enumerar cada uno de los elementos.

Código:

```
##  
for x, i in enumerate(range(1, 21, 3)):  
    print(f"en la posición {x} esta {i}")  
##
```

Consola

```
##  
en la posición 0 esta 1  
en la posición 1 esta 4  
en la posición 2 esta 7  
en la posición 3 esta 10  
en la posición 4 esta 13  
en la posición 5 esta 16  
en la posición 6 esta 19  
##
```



```
en la posición 0 esta 1
en la posición 1 esta 4
en la posición 2 esta 7
en la posición 3 esta 10
en la posición 4 esta 13
en la posición 5 esta 16
en la posición 6 esta 19
```

Esto nos permite más adelante, controlar y guardar los elementos que tengamos en por ejemplo, un diccionario. Para así, buscar los elementos por su posición.

## Bucle While

En un bucle while, el bloque de código se va ejecutar varias veces hasta que se cumpla una determinada condición. Su modo de ejecutarse, es similar a la declaración "If", pero en lugar de ejecutar el bloque de código una vez, regresa al punto donde comenzó el código y repite todo el proceso nuevamente.

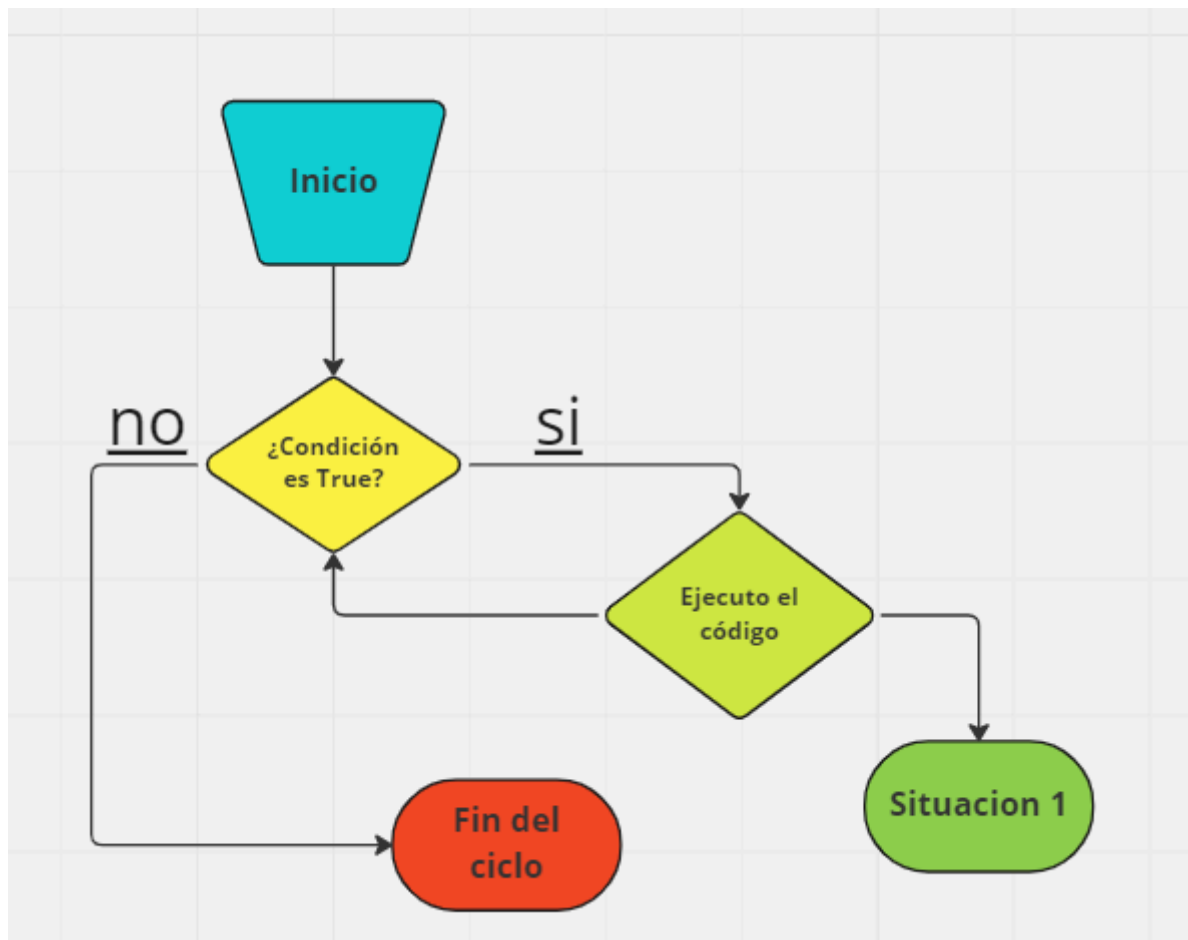
Pero usar esta instrucción necesita de un cuidado especial

### 🛑 ¡Evitar que sean bucles infinitos!

Hay que tener en cuenta que mientras la condición se cumpla, el código va a ejecutarse. Esto es lo que puede dar lugar a bucles infinitos ¡Y como consecuencia harán que nuestra memoria colapse!

Para evitar eso, solemos usar contadores y avisar con un operador lógico, hasta donde queremos que llegue nuestro bucle.





## Sintaxis:

```
##  
while(condición):  
    declaración  
##
```

Ejemplo en código:

```
##  
y = 0
```

```
while(y <= 5):  
    print(y)  
    y+=1  
##
```

Consola:

```
##  
0  
1  
2  
3  
4  
5  
##
```



```
0  
1  
2  
3  
4  
5
```

En este caso, arrancamos con la variable x= 5 y vamos disminuyendo su valor hasta que x sea mayor a cero.



## Ejemplos prácticos:

Crea un programa que solicite al usuario un número y muestre su tabla de multiplicar del 1 al 10.

Código:

```
##  
n = int(input("Ingrese un número: "))  
suma = 0  
  
for i in range(1, n + 1):  
    suma += i  
  
print(f"La suma de los números del 1 al {n} es: {suma}")  
##
```

Escribe un programa que imprima los números pares del 2 al 10 utilizando un bucle while.

Código:

```
##  
numero = 2  
  
while numero <= 10:  
    print(numero)  
    numero += 2  
  
##
```



## Desafío N° 3:

Desarrolla un programa que solicite al usuario un número  $n$  y calcule la suma de los números del 1 al  $n$ .

Ejemplo: Si el usuario coloca el número 4, debe imprimir el número 10 ( $1+2+3+4$ )

[Colab Desafío 3](#)





**Buenos Aires**  
*aprende*  
Agencia de Actividades para el Futuro

**BA** Buenos  
Aires  
Ciudad