

Python / Guía 3

Control

de flujos

## 

## 

## OBJETIVOS DE LA GUÍA

En esta guía aprenderemos a:

* Crear comentarios en el código
* Comprender la indentación
* Manejar condicionales
* Manejar bucles
* Buenas prácticas de programación en Python

## 

# Comentarios

Un comentario es una línea de texto no ejecutable, esto quiere decir que, el intérprete no lo tomará como una línea de código. Los comentarios en Python, así como en otros lenguajes de programación, sirven para dejar explicaciones breves sobre qué es lo que hace el programa.

En Python se puede comentar en las siguientes formas:

* Escribiendo el símbolo de numeral # al comienzo de la línea de texto donde queremos nuestro comentario.

# ¡Esto es un comentario!

* Escribiendo el símbolo de numeral # al final de una instrucción (se aconseja dos espacios para mayor legibilidad:

print(f"{'texto':>15}") # 15 caracteres a la derecha

* Escribiendo triples comillas """ al principio y al final del comentario, En este caso, los comentarios pueden ocupar más de una línea.

"""Este es otro comentario

multilínea"""

''' Otro uso de los comentarios de Python es ayudarte a encontrar rápidamente código con el buscador '''

| Usa comentarios concisos y no más de lo necesario, evitando así los comentarios que afirman lo obvio. |
| --- |

# 

# 

# Indentación

Python utiliza sangría para delimitar bloques de código; esta sangría se llama indentación. No existen comandos para finalizar las líneas, ni llaves que delimitan el código. Los únicos delimitadores existentes son los dos puntos ( : ) y la indentación del código.

La indentación indica el inicio de un bloque. Si las posteriores líneas no tuvieran sangrado, significa el final de dicho bloque de código. Para finalizar un bloque de código, sólo tenemos que dejar de introducir el sangrado.

Tenemos que ser muy conscientes de la indentación que realizamos, ya que de esto dependerá la lógica de nuestro código.

A continuación podemos ver un ejemplo de indentación de bloques de código:

### 

Para la indentación se aconseja 4 espacios en blanco. La tecla 'tab' puede generarlos automáticamente en tu editor de código.

Si abres el intérprete de Python y ejecutas: print("hola") con un espacio (o más) antes de print, Python devolverá: IndentationError: unexpected indent. Esto significa que debes borrar los espacios que no deben ir, para que Python pueda entender el comando.

Se entenderá por completo la indentación en los próximos temas.

# Condicionales

Un programa de Python es un conjunto de instrucciones analizadas y ejecutadas por el intérprete de arriba hacia abajo. Cuando todas las instrucciones se han ejecutado, el programa termina. No obstante, contamos con herramientas para alterar el flujo natural del programa: hacer que se saltee una porción de código según se cumpla tal o cual condición, repetir un conjunto de instrucciones, etc.

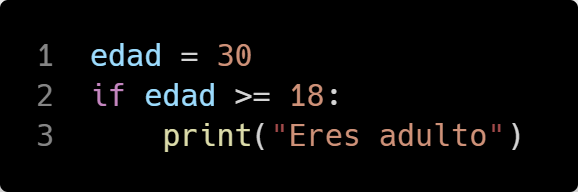
## If

Una de las herramientas de control de flujo es el condicional. A partir de la palabra reservada if indicamos a Python que queremos ejecutar una porción de código sólo si se cumple una determinada condición, es decir, si el resultado de la condición es True. La estructura de if es la siguiente:

if <condición 1>:

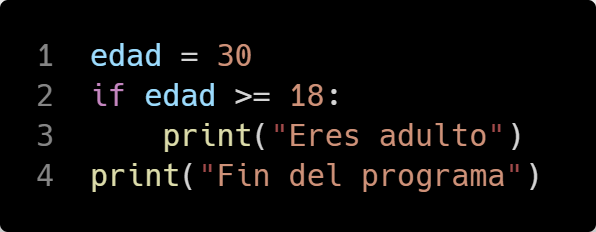
<bloque de código>

**¿Necesitas un ejemplo?**



Primero definimos una variable edad, cuyo valor es un entero, 18. Luego, a través del condicional, indicamos que queremos imprimir 'Eres adulto' si se cumple la condición de que el valor de edad sea mayor o igual a 18. Aquí es donde hay una indentación.

Observa el código:



En este código, solamente la segunda línea está dentro del bloque del condicional: está indentada con cuatro espacios. Por el contrario, la última línea se ejecuta siempre, independientemente del resultado de la condición.

## If - Else

Para ejecutar un bloque de código que no cumpla la condición, usamos la palabra reservada else, seguida de dos puntos. La estructura de if - else es la siguiente:

if <condición 1>:

<bloque de código>

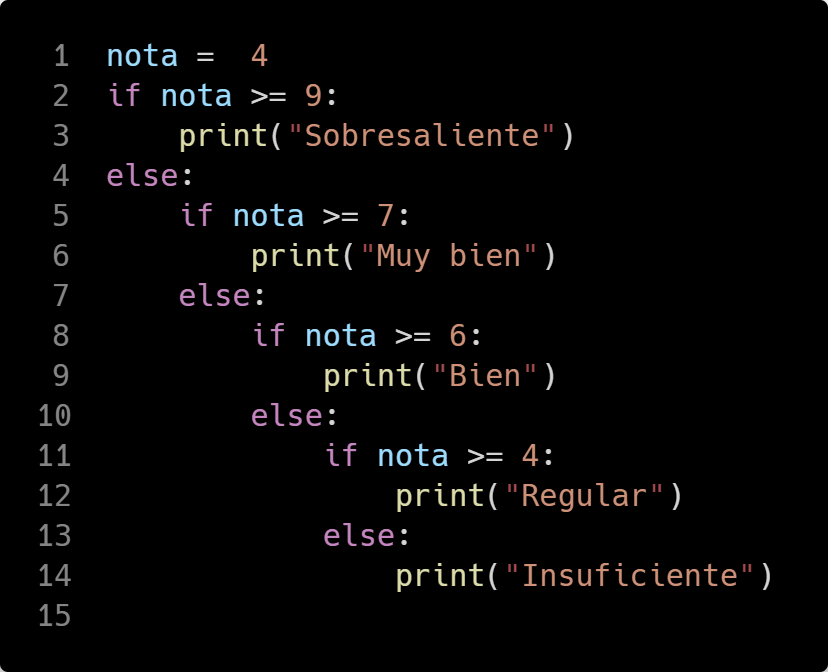
else:

<bloque de código>

## If - Elif

Podemos crear condicionales anidadas, es decir, if dentro de un bloque de otro if.

**¿Necesitas un ejemplo?**



Pero esto es poco legible... Viene a nuestra ayuda la palabra reservada elif, y su estructura lógica es la siguiente:

if <condición 1>:  
 <bloque de código>  
elif <condición 2>:  
 <bloque de código>  
else:  
 <bloque de código>

**¡MANOS A LA OBRA!**

**Ejercicio 1**

Mejorar el código anterior usando elif.

**Ejercicio 2**

Escribe un programa donde el usuario ingrese su edad y sexo, y muestre por pantalla:

Si el usuario es de sexo masculino y es mayor a 65 años, mostrarle que le corresponde el beneficio de jubilación, pero, si es de sexo femenino, debe mostrarle este mensaje si es mayor a 60 años. De lo contrario, mostrar que no le corresponde el beneficio.

| Los operadores 'and' y 'or' son fundamentales para el correcto uso de los condicionales, por eso debes manejar muy bien su concepto. |
| --- |

## Revisemos lo aprendido hasta aquí

* Comentar en una línea o múltiples líneas
* Comprender el concepto de indentación y bloques de código
* Controlar el flujo de datos con los condicionales if, else y elif.

# Bucles

Los bucles son otra herramienta para alterar el flujo normal de un programa. Nos permiten repetir un bloque de código tantas veces o ciclos como queramos. Python incluye únicamente dos tipos de bucle: while y for.

## While

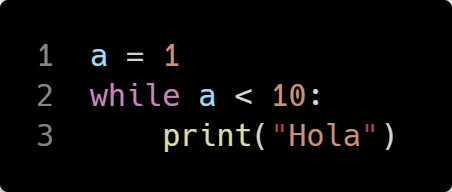
While ejecuta un bloque de código mientras que la condición sea True.

La estructura lógica de while es la siguiente:

while <condición>:

<bloque de código>

**¿Necesitas un ejemplo?**



| Cuando crees un bucle infinito en la consola, presiona las teclas control + C para salir. Python devolverá el error 'KeyboardInterrupt' cada vez que terminemos un programa de esta forma. |
| --- |

**¡MANOS A LA OBRA!**

**Ejercicio 3**

Corregir el ejemplo anterior para que el bucle termine cuando 'a' deje de ser menor que 10.

## While - Else

Al igual que la instrucción if, la instrucción while también puede combinarse con la instrucción else: se crea un bloque lógico while-else.

El bloque lógico while-else no se comporta como el bloque lógico if-else, y aquí la explicación: Cuando la condición de while es True, se ejecuta el bloque else, y cuando la condición de while es False, también se ejecuta el bloque else.

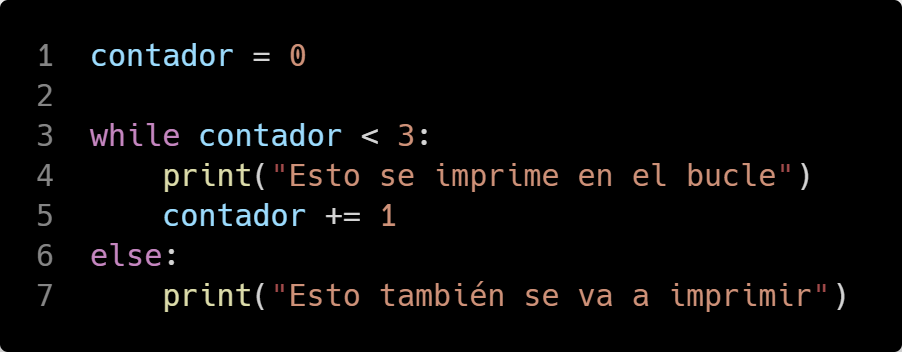
while <condición>:

<bloque de código>

else:

<bloque de código>

**¿Necesitas un ejemplo?**



**¡MANOS A LA OBRA!**

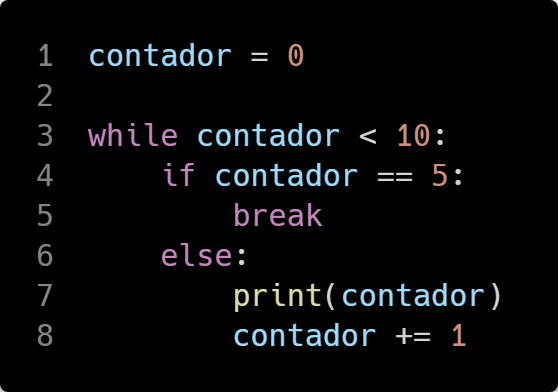
**Ejercicio 4**

Escribir un programa que contenga la estructura while - else: Pedir al usuario que ingrese cuántas veces quiere ingresar texto. Una vez que comienza el bucle, el usuario ingresa texto mediante input(), y se guarda en una variable. Cuando el bucle termina, mostrar la cantidad de caracteres que el usuario escribió en total, es decir, la suma de los caracteres de todas sus entradas. Mostrarlo dentro del bloque else.

## While - Break

La instrucción break sale de un bucle. Generalmente, se pone después de un condicional.

**¿Necesitas un ejemplo?**



**¡MANOS A LA OBRA!**

**Ejercicio 5**

Crea un bucle infinito. Pedir al usuario que ingrese una contraseña, luego que vuelva a repetirla, hasta que ambos ingresos coincidan.

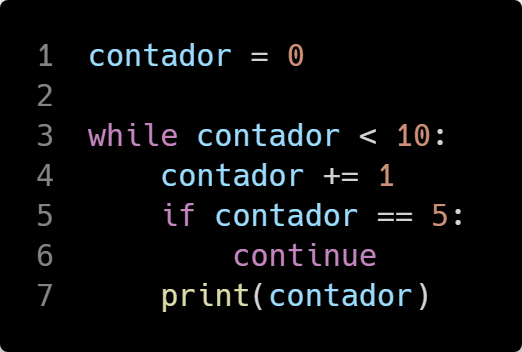
| Es muy habitual en Python generar bucles escribiendo while True:. |
| --- |

## 

## While - Continue

La instrucción continue vuelve a la parte superior del bucle, interrumpiéndolo, pero sin salir de él. Generalmente, se pone después de un condicional.

**¿Necesitas un ejemplo?**



**¡MANOS A LA OBRA!**

**Ejercicio 6**

Crear una variable que guarde la cadena '5'. Iniciar un bucle infinito. Mostrar: 'Adivina el número: '. Si el usuario ingresa un número igual a la variable, sale del bucle, si no, vuelve a repetir la pregunta.

## 

## For

A la instrucción while se le llama bucle indefinido, porque repite un bloque hasta que la condición se hace False, mientras que el bucle for se le llama '**bucle definido**', porque repite un bloque de código recorriendo un conjunto conocido de elementos, de modo que ejecuta tantas iteraciones como elementos hay en el conjunto. Estructura lógica:

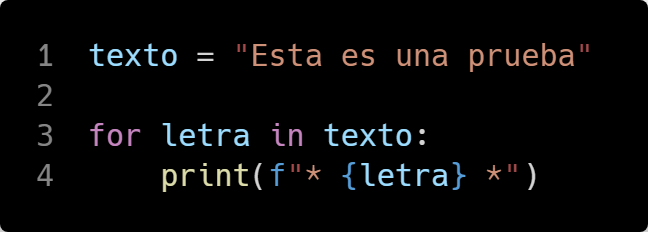
for <elemento> in <iterable>:

<bloque de código>

**<iterable>** Es una variable previamente definida. El tipo de dato debe ser iterable, como, por ejemplo, lo es una cadena. Los tipos numéricos y booleanos no son iterables.

**<elemento>** Al empezar el ciclo, for toma el primer elemento de <iterable> y lo pasa a la variable <elemento>.

**¿Necesitas un ejemplo?**



| Puedes usar la instrucción **continue** y **break** en un bucle for. |
| --- |

## For - Else

for <elemento> in <iterable>:

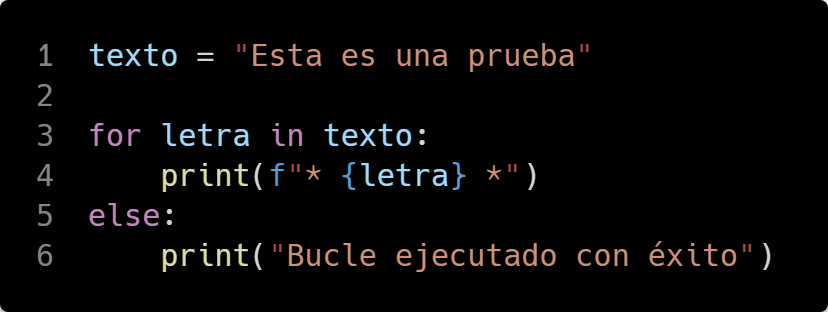
<bloque de código>

else:

<bloque de código>

Else es opcional: una vez que se termina de recorrer <iterable>, se ejecuta el bloque de código anidado en else.

**¿Necesitas un ejemplo?**



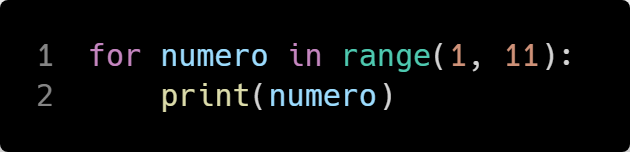
## 

## For y Range

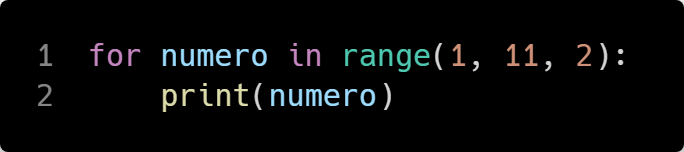
For es bastante diferente al que se usa en otros lenguajes de programación. No obstante, ese comportamiento puede conseguirse en Python empleando otra de sus funciones incorporadas: range. Estructura lógica:

for <elemento> in range(<desde>, <hasta>, <paso:opcional>):  
 <bloque de código>

Por ejemplo, si queremos imprimir números del 1 al 10:



Y, si queremos imprimir números del 1 al 10 de dos en dos:



| La comunidad Python utiliza el guión bajo para definir las variables for que no se utilicen en el bloque de código. Generalmente, se usa para repetir algo. Ejemplo: for\_in range(0, 100): |
| --- |

**¡MANOS A LA OBRA!**

**Ejercicio 7**

Servirse de range y mostrar los números múltiplos de 5 comprendidos entre -50 y -10.

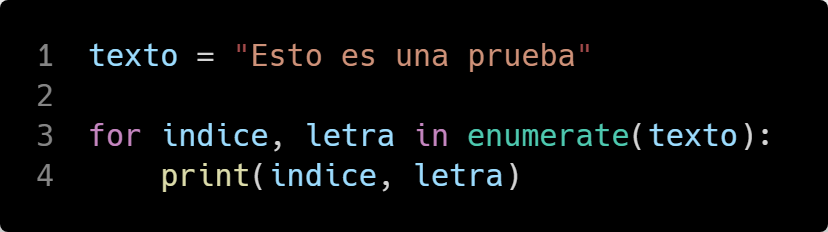
## 

## For y Enumerate

Otra función incorporada, enumerate, permite tener el índice y el valor de cada elemento iterado por for. Estructura lógica:

for <índice>, <elemento> in enumerate(<iterable>):  
 <bloque de código>

**¿Necesitas un ejemplo?**



Enumerate hace más simple el manejo de índices. El siguiente ejemplo origina lo mismo que el ejemplo anterior, pero es más difícil de leer:

