Tema 5: Bucles en Python

Lectura fácil

En programación, los **bucles** nos permiten ejecutar un bloque de código varias veces, lo que es muy útil cuando necesitamos realizar tareas repetitivas. Python tiene dos tipos principales de bucles: el **bucle for** y el **bucle while**. También veremos cómo usar las sentencias **break** y **continue** para controlar el flujo dentro de los bucles.

Bucle for

El bucle **for** se usa cuando queremos **iterar** (repetir) un bloque de código para cada elemento de una secuencia, como una lista, una cadena de texto, o un rango de números y conocemos el número de iteraciones.

Sintaxis básica:

```
for variable in secuencia:
    # Código que se ejecuta en cada iteración
```

- **variable**: Es el nombre de la variable que tomará el valor de cada elemento de la secuencia en cada repetición.
- **secuencia**: Puede ser una lista, una cadena de texto, un rango de números, etc.

Ejemplo con una lista:

```
frutas = ["manzana", "banana", "cereza"]

for fruta in frutas:
    print(fruta)
```

- En este ejemplo, el bucle **for** recorre la lista **frutas** y en cada repetición asigna uno de los elementos de la lista a la variable **fruta**.
- El programa mostrará:

```
manzana
banana
cereza
```

Ejemplo con un rango de números:

```
for numero in range(5):
    print(numero)
```

- La función **range(5)** genera una secuencia de números del 0 al 4 (no incluye el 5).
- El programa mostrará:

```
0
1
2
3
4
```

Bucle while

El bucle **while** repite un bloque de código mientras una condición sea verdadera. Se usa cuando no sabemos cuántas veces se repetirá el bucle, pero sabemos que queremos seguir repitiéndolo mientras se cumpla una condición.

Sintaxis básica:

```
while condición:
    # Código que se ejecuta mientras la condición sea
verdadera
```

- **Condición**: Es una expresión que Python evaluará como **True** (verdadera) o **False** (falsa). Mientras sea **True**, el bucle seguirá repitiéndose.

Ejemplo básico con while:

```
contador = 0

while contador < 5:
    print(contador)
    contador += 1
```

- El bucle se repite mientras **contador** sea menor que 5. Después de cada iteración, aumentamos el valor de **contador** en 1 usando `contador += 1`.
- El programa mostrará:

```
0
1
2
3
4
```

Uso de break y continue

A veces necesitamos controlar los bucles de maneras especiales. Python nos da las herramientas **break** y **continue** para manejar estos casos.

Sentencia break

La sentencia **break** se usa para **salir de un bucle** antes de que haya terminado. Si Python encuentra un **break**, deja de ejecutar el bucle inmediatamente, incluso si aún quedan iteraciones por hacer.

Ejemplo con break:

```
for numero in range(10):
    if numero == 5:
        break
    print(numero)
print("He salido del for")
```

- Este bucle imprimirá los números del 0 al 4, pero cuando **numero** sea igual a 5, el **break** detendrá el bucle.
- El programa mostrará:

```
0
1
2
3
4
He salido del for
```

Sentencia continue

La sentencia **continue** se usa para **saltar una iteración** y continuar con la siguiente. Si Python encuentra un **continue**, no ejecuta el resto del código en esa iteración, pero continúa con la siguiente.

Ejemplo con continue:

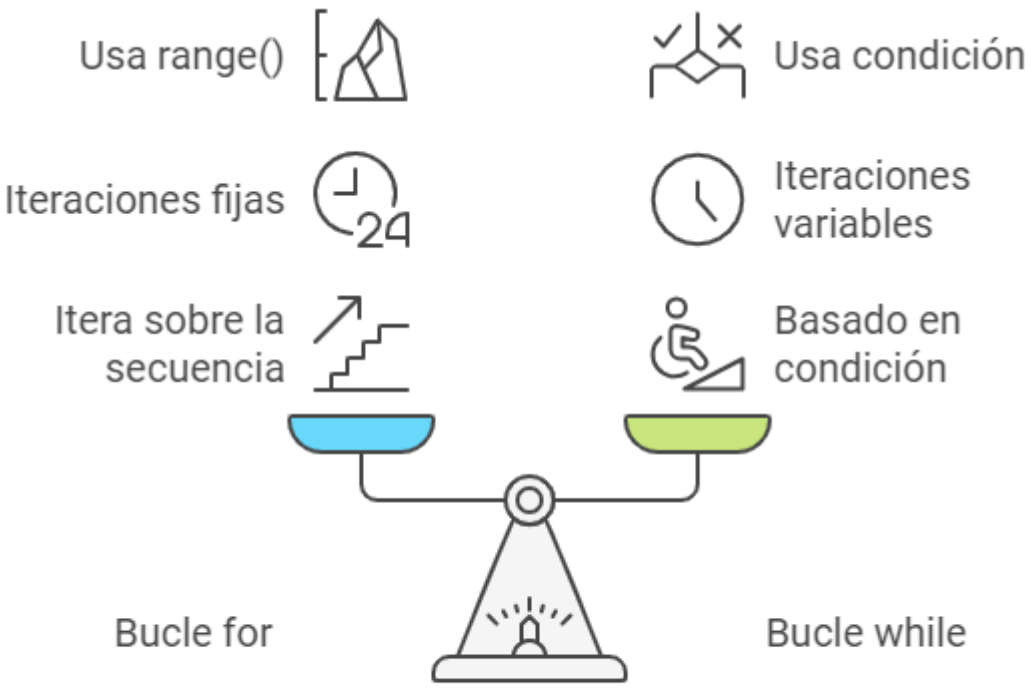
```
for numero in range(5):
    if numero == 3:
        continue
    print(numero)

print("He salido del for")
```

- Cuando el **numero** sea 3, el **continue** hará que el bucle salte esa iteración y pase directamente al siguiente número.
- El programa mostrará:

```
0
1
2
4
He salido del for
```

Fíjate que **3** no se imprime porque el bucle lo saltó.

Comparación de bucles for y while



Comparando bucles for y while en Python

Ejemplos prácticos de bucles

Vamos a ver algunos ejemplos que combinan todo lo que hemos aprendido sobre **for**, **while**, **break**, y **continue**.

Ejemplo 1: Contar hasta 10 con un bucle for

```
for i in range(1, 11):
    print(i)
```

- El bucle **for** va desde el número 1 hasta el 10 (incluye el 10 porque **range(1, 11)** genera números desde 1 hasta 10).
- El programa mostrará:

```
1
2
3
4
5
6
7
8
9
10
```

Ejemplo 2: Sumar los números de una lista

```
numeros = [1, 2, 3, 4, 5]
suma = 0

for numero in numeros:
    suma = suma + numero

print("La suma es", suma)
```

- El bucle recorre la lista **numeros** y va sumando cada número a la variable **suma**.
- El programa mostrará:

```
La suma es 15
```

Ejemplo 3: Encontrar el primer número divisible por 7

```
for numero in range(1, 100):
    if numero % 7 == 0:
        print("El primer número divisible por 7 es",
numero)
        break

///// Con un bucle while ////

encontrado = False
numero = 1

while (numero <= 100) & (encontrado == False):
     if numero % 7 == 0:
          print("El primer número divisible por 7 es",
          numero)
          encontrado = True
     numero += 1
```

- Este bucle busca el primer número divisible por 7 en el rango de 1 a 99. Cuando lo encuentra, usa **break** para detenerse.
- El programa mostrará:

```
El primer número divisible por 7 es 7
```

Ejemplo 4: Contar números los pares existentes hasta el 5 con un bucle while

```
contador = 0
numero = 0

while numero < 5:
    if numero % 2 == 0:
        print(numero)
        contador += 1
    numero += 1
```

print (f"La cantidad de números pares es: {contador}")

print("La cantidad de números pares es: ", contador)

- El bucle **while** cuenta los primeros 5 números pares.
- El programa mostrará:

Los **bucles** son una herramienta fundamental en Python para repetir tareas de manera eficiente. Con los bucles **for** y **while**, podemos recorrer listas, rangos de números o realizar repeticiones controladas. Además, las sentencias **break** y **continue** nos permiten controlar el flujo de los bucles para hacer nuestro código más flexible y dinámico.

EJERCICIOS BUCLES PARA PRACTICAR EN CLASE

Ejercicio 1: Contar números pares

Descripción: Crea un programa que solicite un número entero positivo, y luego use un bucle `for` para contar cuántos números pares hay desde 1 hasta ese número.

Instrucciones:

1. Solicita un número entero positivo.
2. Usa un bucle `for` para iterar desde 1 hasta el número ingresado.
3. Cuenta cuántos números son pares.
4. Imprime el resultado.

Ejemplo de entrada:

```
numero = 10
```

Salida esperada:

```
Hay 5 números pares entre 1 y 10.
```

Ejercicio 2: Suma de números hasta que se introduce un negativo

Descripción: Crea un programa que solicite al usuario números enteros de manera repetida. El programa debe sumar los números introducidos y terminar cuando el usuario ingrese un número negativo.

Instrucciones:

1. Usa un bucle `while` para seguir solicitando números.
2. Si el número ingresado es negativo, termina el bucle.
3. Si el número es positivo, añádelo a una suma acumulada.
4. Al finalizar, imprime la suma total.

Ejemplo de entrada:

```
Entrada: 5
Entrada: 3
Entrada: 8
Entrada: -1
```

Salida esperada:

```
La suma total es 16.
```

Ejercicio 3: Tabla de multiplicar

Descripción: Escribe un programa que solicite un número entero positivo y luego imprima la tabla de multiplicar de ese número (del 1 al 10) usando un bucle `for`.

Instrucciones:

1. Solicita un número entero.
2. Usa un bucle `for` para multiplicar el número por los valores del 1 al 10.
3. Imprime el resultado de cada multiplicación.

Ejemplo de entrada:

```
numero = 7
```

Salida esperada:

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
...
7 x 10 = 70
```

Ejercicio 4: Adivinar un número

Descripción: Crea un programa que seleccione un número al azar entre 1 y 100 (puedes usar `random.randint(1, 100)`). El usuario debe intentar adivinar el número. El programa debe darle pistas diciendo si su conjetura es demasiado alta o demasiado baja, hasta que adivine el número correcto.

Instrucciones:

1. Usa la función `random.randint(1, 100)` para generar un número aleatorio.

2. Solicita al usuario que adivine el número.
3. Usa un bucle `while` para seguir solicitando conjeturas hasta que el número correcto sea adivinado.
4. Si la conjetura es mayor que el número, indica "Demasiado alto". Si es menor, indica "Demasiado bajo".
5. Al final, felicita al usuario por adivinar correctamente.

Ejemplo de interacción:

```
Número generado: 42
Entrada: 50
Salida: Demasiado alto
Entrada: 30
Salida: Demasiado bajo
Entrada: 42
Salida: ¡Felicidades! Has adivinado el número.
```

Ejercicio 5: Contar vocales en una palabra

Descripción: Escribe un programa que solicite una palabra y use un bucle `for` para contar cuántas vocales (a, e, i, o, u) tiene esa palabra.

Instrucciones:

1. Solicita una palabra al usuario.
2. Usa un bucle `for` para recorrer cada letra de la palabra.
3. Cuenta cuántas letras son vocales (a, e, i, o, u, ya sea minúscula o mayúscula).
4. Imprime el número total de vocales encontradas.

Ejemplo de entrada:

```
palabra = "programacion"
```

Salida esperada:

```
La palabra "programacion" tiene 5 vocales.
```