Módulo 4.3: While

Prof. Carlos Cedeño

Bucle `while` en Python

El bucle `while` es una estructura de control que permite repetir un bloque de código mientras una condición especificada sea verdadera. A diferencia del bucle `for`, que itera sobre una secuencia, el `while` es ideal cuando no sabemos de antemano cuántas veces se debe repetir el ciclo.

Sintaxis

La estructura básica del bucle `while` es la siguiente:

```
while condicion:
    # Bloque de código que se ejecuta
    # mientras la condición sea verdadera
```

Puntos clave:

- **Condición:** Es una expresión que se evalúa como `True` or `False`. El bucle continúa mientras la condición sea `True`.
- **Bloque de código:** Son las instrucciones que se repiten. Debe estar indentado.
- **Actualización:** Es fundamental que dentro del bucle algo modifique la variable de la condición. De lo contrario, se podría crear un **bucle infinito**.

Ejemplo 1: Contador simple

Un ejemplo clásico para empezar es un contador que imprime números hasta un límite.

```
# Imprimir números del 1 al 5
contador = 1
while contador <= 5:
    print(f"Número: {contador}")
    contador += 1  # ¡Importante! Actualizamos el contador para evitar un bucle infinito

print("¡Fin del bucle!")
```

En este caso, la variable `contador` se inicializa en 1. El bucle se ejecuta mientras `contador` sea menor o igual a 5. En cada iteración, se imprime el valor y se incrementa en 1. Cuando `contador` llega a 6, la condición `6 <= 5` se vuelve `False` y el bucle termina.

Ejemplo 2: Validación de entrada de usuario

El `while` es perfecto para asegurar que el usuario ingrese el tipo de dato correcto.

```
# Pedir al usuario que ingrese un número positivo
numero = 0
while numero <= 0:
    entrada = input("Por favor, ingresa un número positivo: ")
    if entrada.isdigit():
        numero = int(entrada)
        if numero <= 0:
            print("El número debe ser mayor que cero.")
    else:
        print("Entrada no válida. Debes ingresar un número.")

print(f"¡Excelente! Ingresaste el número {numero}.")
```

Aquí, el bucle se repetirá indefinidamente hasta que el usuario ingrese un valor que, al ser convertido a entero, sea mayor que cero.

Ejemplo 3: Menú interactivo

Se puede usar un `while` para mantener un programa en ejecución hasta que el usuario decida salir.

```
opcion = ""
while opcion != "3":
    print("\n--- MENÚ PRINCIPAL ---")
    print("1. Saludar")
    print("2. Contar un chiste")
    print("3. Salir")
    opcion = input("Elige una opción: ")

    if opcion == "1":
        print("\n¡Hola, estudiante de Python!")
    elif opcion == "2":
        print("\n¿Por qué los programadores prefieren el modo oscuro? Porque la luz atrae a los bugs.")
    elif opcion == "3":
        print("\n¡Hasta luego!")
    else:
        print("\nOpción no válida. Inténtalo de nuevo.")
```

Estructuras `else`, `break` y `continue`

El bucle `while` puede combinarse con otras palabras clave para un control más fino.

- **`else`**: El bloque `else` se ejecuta una única vez, solo si el bucle `while` termina de forma natural (es decir, cuando su condición se vuelve `False`), pero no si se interrumpe con `break`.
- **`break`**: Termina el bucle inmediatamente, sin importar el estado de la condición.
- **`continue`**: Salta el resto del código de la iteración actual y vuelve al inicio del bucle para evaluar de nuevo la condición.

Ejemplo combinado:

```
# Adivina un número secreto (con un número limitado de intentos)
numero_secreto = 7
intentos = 3

while intentos > 0:

    adivinanza = int(input(f"Adivina el número (tienes {intentos} intentos): "))
    if adivinanza == numero_secreto:
        print("¡Felicidades! ¡Adivinaste!")
        break  # Termina el bucle porque el usuario ganó
    else:
        intentos -= 1
        if intentos > 0:
            print("Incorrecto. Intenta de nuevo.")
        continue # Salta a la siguiente iteración
else:
    # Este bloque solo se ejecuta si el while termina por "intentos > 0" siendo falso
    print(f"¡Lo siento! Te quedaste sin intentos. El número era {numero_secreto}.")
```

Ejercicio 1: Suma de dígitos

Escribe un programa que pida al usuario un número entero y, usando un bucle `while`, calcule y muestre la suma de sus dígitos.

Ejemplo de ejecución:

```
Ingresa un número entero: 12345
La suma de los dígitos es: 15
```

Pista: Puedes usar el operador módulo (`%`) para obtener el último dígito y la división entera (`//`) para removerlo.

Ejercicio 2: Juego de adivinanza con pistas

Crea un juego donde el programa elige un número aleatorio entre 1 y 100. El usuario debe adivinarlo. En cada intento fallido, el programa debe indicar si el número a adivinar es **mayor** o **menor** que el ingresado. El juego termina cuando el usuario adivina el número.

Pista: Usa el módulo `random` para generar el número aleatorio (`import random; numero_secreto = random.randint(1, 100)`).

Ejercicio 3: Sucesión de Fibonacci

Escribe un programa que genere los primeros `n` números de la sucesión de Fibonacci, donde `n` es un número ingresado por el usuario. La sucesión de Fibonacci comienza con 0 y 1, y cada número siguiente es la suma de los dos anteriores (0, 1, 1, 2, 3, 5, 8, ...).

Ejemplo de ejecución:

```
¿Cuántos números de la sucesión de Fibonacci quieres generar? 8
0 1 1 2 3 5 8 13
```

Pista: Necesitarás dos variables para almacenar los dos números anteriores de la serie y una variable de contador para controlar el bucle `while`.