🔄 ¿Qué son las estructuras repetitivas?

Las estructuras repetitivas, también conocidas como bucles, nos permiten ejecutar un bloque de código múltiples veces. En Python, estas estructuras son fundamentales para automatizar tareas repetitivas y procesar conjuntos de datos.

🔁 Bucle `while`

El bucle `while` ejecuta un bloque de código mientras una condición sea verdadera:

```
realizarBucle = True
while realizarBucle == True:
    terminarElBucle = input("¿Quieres terminar el bucle? [y/N]: ")
    if terminarElBucle == "Y" or terminarElBucle == "y":
        print("Adios mundo cruel..")
        realizarBucle = False
    else:
        print("Gracias por salvarme ✨")
```

> **Nota:** Es importante asegurarse de que la condición del `while` eventualmente se vuelva falsa, de lo contrario, crearás un bucle infinito.

⏹️ Palabra clave `break`

La palabra clave `break` nos permite salir inmediatamente de un bucle, incluso si la condición sigue siendo verdadera:

```
while True:  # Bucle infinito intencional
    terminarElBucle = input("¿Quieres terminar el bucle? [y/N]: ")
    if terminarElBucle == "Y" or terminarElBucle == "y":
        print("Adios mundo cruel..")
        break  # Sale del bucle cuando el usuario ingresa "y" o "Y"
    else:
        print("Gracias por salvarme ✨")
```

🔢 Contadores con `while`

Podemos usar `while` junto con un contador para ejecutar código un número específico de veces:

```
contador = 0
while True:
    if contador == 10: break
    print("Hola", contador)
    contador += 1  # Es una simplificación de: contador = contador + 1
```

> **Tip:** El operador `+=` es una forma abreviada de incrementar una variable. `contador += 1` es equivalente a `contador = contador + 1`.

Bucle `for`

El bucle `for` en Python es especialmente útil para iterar sobre una secuencia (como un rango de números):

```
for posicion in range(0, 10, 1):
    print("Hola", posicion)
```

> **Explicación:** La función `range(comienzo, parada, paso)` genera una secuencia de números desde `comienzo` hasta `parada-1`, avanzando de `paso` en `paso`.

Iterando sobre caracteres con `for`

El bucle `for` también puede iterar sobre cada elemento de una secuencia, como los caracteres de una cadena de texto:

```
palabra = input("Ingresa una palabra sin espacios: ")
tieneEspacios = False

for letra in palabra:
    if letra == " ":
        tieneEspacios = True
        break

if tieneEspacios == True:
    print("Existe un espacio en la palabra. Terminando el programa...")
elif tieneEspacios == False:
    print("La palabra no contiene espacios")
```

Ejercicios prácticos

Ejercicio 1: Suma de números del 1 al N

Escribe un programa que sume todos los números desde 1 hasta un número N ingresado por el usuario.

```
n = int(input("Ingresa un número entero positivo: "))
suma = 0

for i in range(1, n + 1):
    suma += i

print(f"La suma de los números del 1 al {n} es: {suma}")
```

Ejercicio 2: Tabla de multiplicar

Crea un programa que muestre la tabla de multiplicar de un número ingresado por el usuario.

```
numero = int(input("Ingresa un número para ver su tabla de multiplicar: "))

print(f"Tabla de multiplicar del {numero}:")
for i in range(1, 11):
    resultado = numero * i
    print(f"{numero} × {i} = {resultado}")
```

Ejercicio 3: Contador de vocales

Desarrolla un programa que cuente cuántas vocales hay en una frase ingresada por el usuario.

```
frase = input("Ingresa una frase: ").lower()
vocales = "aeiou"
contador = 0

for letra in frase:
    if letra in vocales:
        contador += 1

print(f"La frase tiene {contador} vocales")
```

Ejercicio 4: Menú interactivo

Crea un menú interactivo que permita al usuario elegir entre varias opciones, como sumar dos números, restar, multiplicar o salir del programa.

```
while True:
    print("\nMenú:")
    print("1. Sumar dos números")
    print("2. Restar dos números")
    print("3. Multiplicar dos números")
    print("4. Salir")

    opcion = input("Elige una opción (1-4): ")

    if opcion == "1":
        a = float(input("Ingresa el primer número: "))
        b = float(input("Ingresa el segundo número: "))
        print(f"La suma es: {a + b}")
    elif opcion == "2":
        a = float(input("Ingresa el primer número: "))
        b = float(input("Ingresa el segundo número: "))
        print(f"La resta es: {a - b}")
    elif opcion == "3":
        a = float(input("Ingresa el primer número: "))
        b = float(input("Ingresa el segundo número: "))
```

```
            print(f"La multiplicación es: {a * b}")
        elif opcion == "4":
            print("Saliendo del programa...")
            break
        else:
            print("Opción no válida, intenta de nuevo.")
```

🔍 Diferencias entre `while` y `for`

- Usa `while` cuando no sabes exactamente cuántas iteraciones necesitarás (por ejemplo, hasta que el usuario decida salir).
- Usa `for` cuando conoces el número exacto de iteraciones o cuando necesitas recorrer todos los elementos de una secuencia.

🔄 Conclusión

Las estructuras repetitivas son esenciales en programación, ya que nos permiten automatizar tareas repetitivas y procesar grandes cantidades de datos de manera eficiente. Python ofrece bucles `while` y `for` con diferentes características que se adaptan a distintas necesidades.
