CLASE 6

Loops: While y For

Resumen de Estudio

Curso de Python - Fundamentos

Indice de Contenidos

1. Que es un Loop?

Un **loop** (bucle) es una estructura de control que permite ejecutar un bloque de codigo multiples veces. En lugar de escribir el mismo codigo repetidamente, usamos loops para automatizar la repeticion.

Analogia:

Imagina que quieres contar del 1 al 100. Sin loops, necesitarias escribir 100 lineas de print(). Con un loop, solo necesitas 3-4 lineas de codigo.

Ejemplo comparativo:

Sin Loop	Con Loop

`print(1)`
`print(2)`
`print(3)`
`print(4)`
`print(5)`
	`contador = 1`
	`while contador <= 5:`
	`    print(contador)`
	`    contador += 1`

2. While Loop - Sintaxis y Uso

El loop **while** repite un bloque de codigo MIENTRAS una condicion sea verdadera (True). Cuando la condicion se vuelve falsa (False), el loop termina.

Sintaxis:

```
while condicion:
    # codigo que se repite
    # mientras condicion sea True
    # actualizar variable de control
```

Componentes:

Componente	Descripcion
while	Palabra reservada que inicia el loop
condicion	Expresion booleana (True/False)
:	Dos puntos obligatorios al final de la linea
indentacion	4 espacios para el bloque de codigo que se repite

Ejemplo - Contador del 1 al 5:

```
# Contador basico
contador = 1

while contador <= 5:
    print(f"Numero: {contador}")
```

```
    contador += 1  # IMPORTANTE: actualizar!

print("Fin del loop")
```

Output:

```
Numero: 1
Numero: 2
Numero: 3
Numero: 4
Numero: 5
Fin del loop
```

Diagrama de Flujo - While Loop:

Paso	Accion
1	Inicializar variable de control
2	Evaluar condicion
3	Si True: ejecutar bloque de codigo
4	Actualizar variable de control
5	Volver al paso 2
6	Si False: salir del loop

3. break y continue

break:

La instruccion **break** termina el loop COMPLETAMENTE. El programa continua ejecutando el codigo que viene despues del loop.

```
i = 0
while i < 10:
    i += 1
    if i == 5:
        break  # Sale del loop cuando i es 5
    print(i)

# Output: 1, 2, 3, 4
```

continue:

La instruccion **continue** salta a la SIGUIENTE iteracion del loop, ignorando el resto del codigo en la iteracion actual.

```
i = 0
while i < 10:
    i += 1
    if i == 5:
        continue  # Salta el print cuando i es 5
    print(i)

# Output: 1, 2, 3, 4, 6, 7, 8, 9, 10
```

break	continue
Sale del loop completamente	Salta a la siguiente iteracion
El codigo despues del loop se ejecuta	El loop continua normalmente
Uso: terminar cuando encuentras lo buscado	Uso: ignorar casos especificos

4. Patron while True

while True crea un loop que se repite infinitamente. Usamos **break** para salir del loop cuando se cumple una condicion especifica. Este patron es muy util para validar entrada del usuario.

Patron de validacion:

```
while True:
    entrada = input("Ingresa tu edad: ")

    if entrada.isdigit():  # Verifica solo numeros
        edad = int(entrada)
        break  # Sale si es valido
    else:
```

```
        print("Error: solo numeros")

print(f"Tu edad es {edad}")
```

TIP: El metodo isdigit() devuelve True si TODOS los caracteres son digitos (0-9).

5. For Loop - Sintaxis y Uso

El loop **for** itera sobre cada elemento de una secuencia (string, lista, range, etc.). A diferencia de while, no necesitas actualizar una variable de control manualmente.

Sintaxis:

```
for variable in secuencia:
    # codigo que se repite
    # para cada elemento
```

Ejemplo con string:

```
for letra in "PYTHON":
    print(letra)

# Output:
# P
# Y
# T
# H
# O
# N
```

Diagrama de Flujo - For Loop:

Paso	Accion
1	Tomar la secuencia a iterar
2	Asignar el siguiente elemento a la variable
3	Ejecutar el bloque de codigo
4	Si hay mas elementos: volver al paso 2
5	Si no hay mas elementos: salir del loop

6. La funcion range()

La funcion **range()** genera una secuencia de numeros. Es perfecta para usar con for cuando necesitas repetir algo un numero especifico de veces.

Tres formas de usar range():

Forma	Ejemplo	Resultado
`range(stop)`	`range(5)`	`0, 1, 2, 3, 4`
`range(start, stop)`	`range(1, 6)`	`1, 2, 3, 4, 5`
`range(start, stop, step)`	`range(0, 10, 2)`	`0, 2, 4, 6, 8`
`Con step negativo`	`range(5, 0, -1)`	`5, 4, 3, 2, 1`

IMPORTANTE: range() NUNCA incluye el numero final (stop). Si quieres del 1 al 10, usa range(1, 11).

Ejemplos practicos:

```
# Repetir 5 veces (0 a 4)
for i in range(5):
    print(f"Iteracion {i}")

# Numeros del 1 al 10
for i in range(1, 11):
    print(i)

# Numeros pares del 0 al 10
for i in range(0, 11, 2):
    print(i)  # 0, 2, 4, 6, 8, 10

# Cuenta regresiva
for i in range(10, 0, -1):
    print(i)  # 10, 9, 8, ..., 1
```

7. Iterando sobre Strings

Un string es una secuencia de caracteres. Podemos usar for para recorrer cada caracter individualmente.

Ejemplo - Contar vocales:

```
texto = "Hola Mundo"
vocales = "aeiouAEIOU"
contador = 0

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

print(f"Vocales encontradas: {contador}")
# Output: Vocales encontradas: 4
```

TIP: El operador **in** verifica si un elemento esta en una secuencia. Ejemplo: 'a' in 'aeiou' devuelve True.

8. La funcion enumerate()

La funcion **enumerate()** te da el INDICE y el VALOR de cada elemento al mismo tiempo. Es perfecta para crear menus numerados.

Sintaxis:

```
for indice, valor in enumerate(secuencia):
    print(f"{indice}. {valor}")

# Con start=1 para empezar desde 1
for i, item in enumerate(secuencia, start=1):
    print(f"{i}. {item}")
```

Ejemplo - Menu de opciones:

```
frutas = ["Manzana", "Banana", "Cereza"]

print("=== MENU ===")
for i, fruta in enumerate(frutas, start=1):
    print(f"{i}. {fruta}")

# Output:
# === MENU ===
# 1. Manzana
# 2. Banana
# 3. Cereza
```

9. Patrones Visuales con For

Podemos usar loops para crear patrones visuales con texto. El truco clave es que multiplicar un string por un numero lo repite.

Multiplicacion de strings:

```
"*" * 3    # "***"
"=" * 10   # "=========="
"Hola " * 2 # "Hola Hola "
```

Triangulo de asteriscos:

```
for i in range(1, 6):
    print("*" * i)

# Output:
# *
# **
# ***
# ****
# *****
```

Triangulo invertido:

```
for i in range(5, 0, -1):
    print("*" * i)

# Output:
# *****
# ****
# ***
# **
# *
```

10. Comparacion While vs For

Aspecto	WHILE	FOR
Uso	No sabes cuantas veces repetir	Sabes cuantas veces o tienes secuencia
Condicion	Se evalua antes de cada iteracion	Itera sobre cada elemento de secuencia
Riesgo	Loop infinito si olvidas actualizar	Casi imposible loop infinito
Casos tipicos	- Validar input - Juegos - Menus interactivos	- Recorrer listas - Repetir N veces - Procesar strings
Control	Variable de control manual	Variable se asigna automaticamente

REGLA SIMPLE:
- Si puedes contar las iteraciones o tienes una coleccion -> usa **FOR**
- Si dependes de una condicion que puede cambiar -> usa **WHILE**

11. Errores Comunes

1. Loop Infinito (while):

```
# ERROR: Falta actualizar variable
i = 0
while i < 5:
    print(i)
    # FALTA: i += 1

# SOLUCION:
i = 0
while i < 5:
    print(i)
    i += 1  # No olvidar!
```

2. Off-by-one con range():

```
# ERROR: Quiero 1-10 pero obtengo 1-9
for i in range(1, 10):
    print(i)

# SOLUCION: El stop no se incluye
for i in range(1, 11):  # 1 a 10
    print(i)
```

3. Indentacion incorrecta:

```
# ERROR: IndentationError
for i in range(3):
print(i)  # Falta indentacion!

# SOLUCION:
for i in range(3):
    print(i)  # 4 espacios
```

4. Modificar variable de for:

```
# ERROR: Esto no funciona como esperas
for i in range(5):
    print(i)
    i += 10  # No tiene efecto!

# La variable i se reasigna en cada iteracion
# Output sigue siendo: 0, 1, 2, 3, 4
```

TIP: Si tu programa se "congela", probablemente tienes un loop infinito. Usa **Ctrl+C** para detenerlo.

12. Ejercicios Resueltos

Ejercicio 1: Cuenta Regresiva

Crear una cuenta regresiva del 10 al 1, luego imprimir 'Despegue!'

```
# Solucion
numero = 10

while numero >= 1:
    print(numero)
    numero -= 1

print("Despegue!")
```

Ejercicio 2: Numeros Selectivos

Imprimir 1-20 saltando multiplos de 3, detenerse en 15

```
# Solucion
numero = 0

while numero < 20:
    numero += 1

    if numero == 15:
        break

    if numero % 3 == 0:
        continue

    print(numero)
```

Ejercicio 3: Validador de Password

Pedir password hasta que sea 'python123'

```
# Solucion
password = "python123"

while True:
    intento = input("Contrasena: ")

    if intento == password:
        print("Acceso concedido!")
        break
    else:
        print("Incorrecta, intenta de nuevo")
```

Ejercicio 4: Tabla de Multiplicar

Mostrar la tabla de multiplicar de un numero

```
# Solucion
numero = int(input("Numero: "))

for i in range(1, 11):
    resultado = numero * i
    print(f"{numero} x {i} = {resultado}")
```

Ejercicio 5: Contador de Vocales Detallado

```
# Solucion
frase = input("Frase: ").lower()

cont_a = cont_e = cont_i = cont_o = cont_u = 0

for letra in frase:
    if letra == 'a': cont_a += 1
    elif letra == 'e': cont_e += 1
    elif letra == 'i': cont_i += 1
    elif letra == 'o': cont_o += 1
    elif letra == 'u': cont_u += 1

print(f"a:{cont_a}, e:{cont_e}, i:{cont_i}, o:{cont_o}, u:{cont_u}")
```

Ejercicio 6: Menu con enumerate

```
# Solucion
comidas = ["Pizza", "Hamburguesa", "Tacos", "Sushi", "Ensalada"]

print("=== MENU ===")
for i, comida in enumerate(comidas, start=1):
    print(f"{i}. {comida}")

opcion = int(input("Elige (1-5): "))
elegida = comidas[opcion - 1]
print(f"Elegiste: {elegida}")
```

Ejercicio 7: Generador de Patrones

```
# Solucion
tamano = int(input("Tamano: "))

# Triangulo
print("\n=== TRIANGULO ===")
for i in range(1, tamano + 1):
    print("*" * i)

# Cuadrado
print("\n=== CUADRADO ===")
for i in range(tamano):
    print("*" * tamano)

# Triangulo invertido
print("\n=== INVERTIDO ===")
for i in range(tamano, 0, -1):
    print("*" * i)
```

Ejercicio Integrador: Juego de Adivinanza

```
# Solucion completa
numero_secreto = 27
intentos_max = 7
intentos = 0

print("Adivina el numero (1-50)")
print(f"Tienes {intentos_max} intentos")

while intentos < intentos_max:
    entrada = input("Tu numero: ")

    if not entrada.isdigit():
        print("Ingresa un numero valido")
        continue

    numero = int(entrada)
    intentos += 1

    if numero == numero_secreto:
        print(f"Felicidades! En {intentos} intentos")
```

```
        break
    elif numero < numero_secreto:
        print(f"Muy bajo! Quedan {intentos_max - intentos}")
    else:
        print(f"Muy alto! Quedan {intentos_max - intentos}")
else:
    print(f"Game Over! Era {numero_secreto}")
```

13. Proyecto Final - Calculadora Interactiva

Especificaciones:

- Menu que se repite hasta elegir Salir (while True)
- Opciones: Sumar, Restar, Multiplicar, Dividir, Historial, Salir
- Sumar: pedir cuantos numeros, usar for para pedirlos
- Guardar resultados en historial
- Validar division por cero

Codigo completo:

```
# Calculadora Interactiva
historial = ""

print("=== CALCULADORA ===")

while True:
    print("\n1.Sumar 2.Restar 3.Multiplicar")
    print("4.Dividir 5.Historial 6.Salir")

    op = input("Opcion: ")

    if op == "1":  # Sumar
        n = int(input("Cuantos numeros? "))
        suma = 0
        for i in range(1, n + 1):
            num = float(input(f"Numero {i}: "))
            suma += num
        print(f"Resultado: {suma}")
        historial += f"Suma: {suma}\n"

    elif op == "2":  # Restar
        a = float(input("Primer numero: "))
        b = float(input("Segundo numero: "))
        res = a - b
        print(f"Resultado: {res}")
        historial += f"{a} - {b} = {res}\n"

    elif op == "3":  # Multiplicar
        a = float(input("Primer numero: "))
        b = float(input("Segundo numero: "))
        res = a * b
        print(f"Resultado: {res}")
        historial += f"{a} * {b} = {res}\n"

    elif op == "4":  # Dividir
        a = float(input("Dividendo: "))
        b = float(input("Divisor: "))
        if b == 0:
            print("Error: division por cero!")
        else:
```

```
            res = a / b
            print(f"Resultado: {res}")
            historial += f"{a} / {b} = {res}\n"

    elif op == "5":  # Historial
        print("\n=== HISTORIAL ===")
        if historial == "":
            print("Vacio")
        else:
            print(historial)

    elif op == "6":  # Salir
        print("Adios!")
        break

    else:
        print("Opcion invalida")
```

Proxima Clase: Listas y Tuplas - Colecciones de datos que funcionan perfecto con loops!