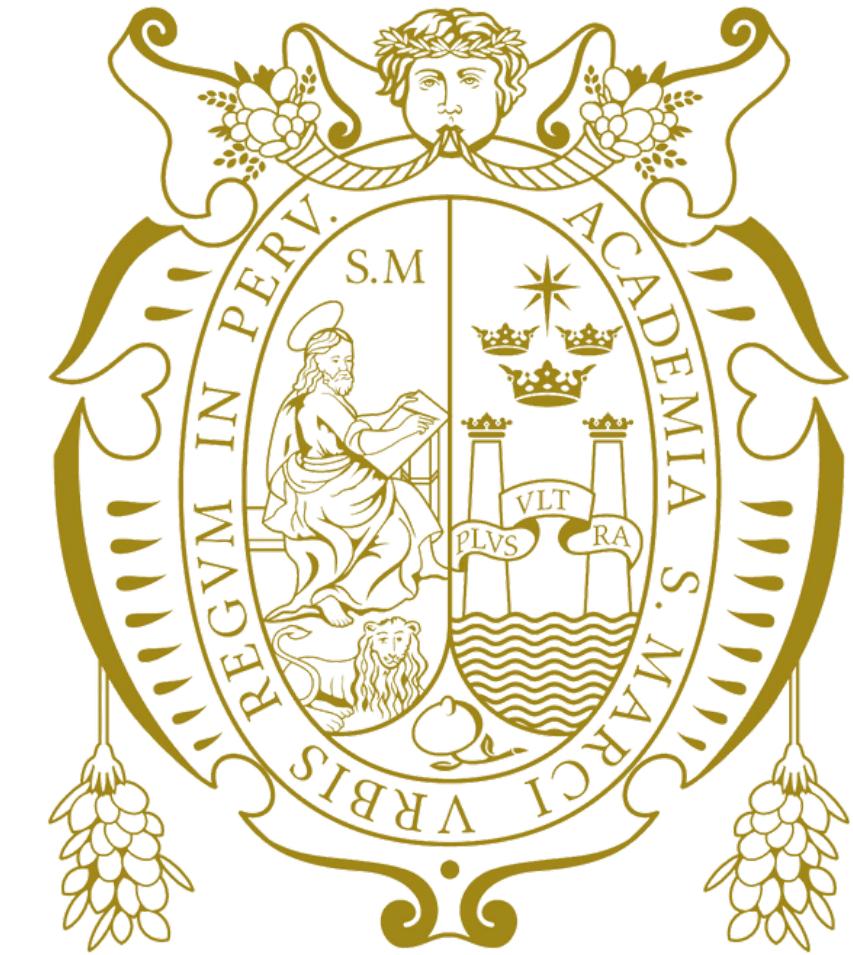


CLASE 3: WHILE, FOR, ITERADORES, ACUMULABLES, CONTADORES

CURSO DE PYTHON

Pensando en acciones repetitivas para el control de casos

PROF: KEVIN T.A. | ING. SOFTWARE



INTRODUCCION

ALGUNA VEZ HAS TENIDO QUE REPETIR UNA TAREA MUCHAS VECES? POR EJEMPLO, ENVIAR VARIOS CORREOS, REVISAR NOTAS UNA POR UNA O CONTAR HASTA 100. IMAGINA SI PUDIERAS DECIRLE A UNA COMPUTADORA: “REPITE ESTO HASTA QUE TERMINES”.

AHÍ ES DONDE ENTRAN LOS BUCLES. SON ESTRUCTURAS QUE PERMITEN REPETIR INSTRUCCIONES AUTOMÁTICAMENTE SIN ESCRIBIRLAS UNA Y OTRA VEZ.”



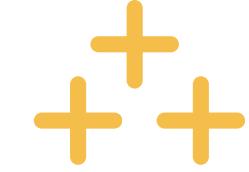
OBJETIVO

¿Qué valores?



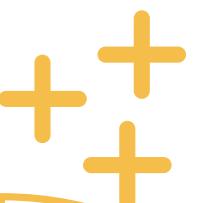
OBJETIVO GENERAL:

- Comprender y aplicar las estructuras de bucle (for y while) para repetir acciones de manera eficiente y controlada dentro de un programa.



OBJETIVOS ESPECÍFICOS:

1. Identificar cuándo usar un bucle: Reconocer situaciones donde se requiere la repetición de instrucciones para resolver un problema.
2. Diferenciar los tipos de bucles en Python: Entender el uso y sintaxis de for y while, y cuándo es más conveniente usar cada uno.
3. Escribir bucles correctamente: Implementar bucles para recorrer rangos, listas o repetir acciones hasta que se cumpla una condición.
4. Prevenir errores comunes: Comprender la importancia de las condiciones de salida para evitar bucles infinitos.
5. Aplicar bucles en problemas reales: Resolver ejercicios prácticos donde se automatice un proceso repetitivo mediante bucles.



BUCLES

Un bucle while en Python se utiliza para ejecutar repetidamente un bloque de código mientras se cumpla una condición específica. Es ideal cuando no sabes cuántas veces necesitas iterar, pero sí sabes la condición que debe cumplirse.



Bucle while

```
contador = 0  
  
while contador < 5:  
  
    print("Contador:", contador)  
  
    contador += 1
```

Break: Rompe el bucle.

```
contador = 0  
  
while True: # Bucle infinito  
  
    print("Contador:", contador)  
  
    contador += 1  
  
    if contador == 5:  
  
        break
```

Salida:

```
Contador: 0  
Contador: 1  
Contador: 2  
Contador: 3  
Contador: 4
```

Continue: Itera de nuevo el bucle

```
contador = 0  
  
while contador < 5:  
  
    contador += 1  
  
    if contador == 3:  
  
        continue  
  
    print("Contador:", contador)
```

Salida:

```
Contador: 1  
Contador: 2  
Contador: 4  
Contador: 5
```



BUCLES

El bucle for en Python se utiliza para iterar sobre una secuencia (como listas, tuplas, cadenas, rangos, etc.) y ejecutar un bloque de código para cada elemento de esa secuencia. Es ideal cuando sabes de antemano cuántas iteraciones necesitas o cuando trabajas con colecciones de datos.

Bucle For

```
numeros = [1, 2, 3, 4, 5]  
  
for numero in numeros:  
    print("Número:", numero)
```

Enumerate: indice y valor

```
frutas = ["manzana", "plátano", "cereza"]  
  
for índice, fruta in enumerate(frutas):  
    print(f"Índice: {índice}, Fruta: {fruta}")
```

Salida:

Índice: 0, Fruta: manzana
Índice: 1, Fruta: plátano
Índice: 2, Fruta: cereza

Range: itera sobre el rango asignada

```
for i in range(5):  
    print("i:", i)    Salida:  
                    i: 0  
                    i: 1  
                    i: 2  
                    i: 3  
                    i: 4
```

Range: itera sobre el rango, pero con un razon de 2

```
for i in range(2, 10, 2):  
    print("i:", i)    Salida:  
                    i: 2  
                    i: 4  
                    i: 6  
                    i: 8
```



WHILE - FOR

¿Cuándo utilizar cada bucle? Ambos bucles permiten iterar, y en muchos casos se pueden usar con pequeñas modificaciones.

¿Cómo puedo saber cuando utilizar cada tipo de bucle?



¿Ejemplos?

El bucle **While** se utiliza puede utilizar cuando no sabemos cuantas iteraciones se harán. Ademas si la condición no se cumple, se salta el bucle y continua con su recorrido.

El bucle **For** se utiliza cuando sabemos la cantidad de iteraciones.

Ejemplos de **While**, cuando un usuario ingresa mal una contraseña, en este caso no sabemos cuantas veces intentara ingresar la contraseña correcta.

Ejemplos de **For**, cuando tenemos a un conjunto de personas o cosas y queremos hacer acciones con ellas.



EJEMPLOS DE WHILE

1. Un jugador gana puntos a lo largo de un juego, y cada vez que alcanza 100 puntos, recibe un premio. El jugador comienza con 0 puntos y gana puntos en cada ronda. Escribe un programa que calcule **cuántas rondas le toma al jugador llegar o superar los 100 puntos.**
2. Tienes un número secreto entre 1 y 50, y el jugador intenta adivinarlo. Después de cada intento, el programa le dice al jugador si el número ingresado es mayor, menor o igual al número secreto. **El jugador sigue intentando hasta acertar.**
3. Un usuario tiene un saldo inicial de 500 soles en su cuenta bancaria. El usuario puede realizar retiros, pero el cajero no permitirá que el saldo baje de 0. **El programa debe seguir pidiendo al usuario que retire una cantidad hasta que no quede suficiente dinero.**
4. Tienes una lista de números que incluye tanto valores positivos como negativos. Escribe un programa que cuente cuántos números positivos hay en la lista, **deteniéndose cuando encuentre un número negativo.**
5. Tienes una máquina expendedora que acepta monedas de 1, 2 y 5 soles. El usuario va introduciendo monedas hasta que **el total de dinero ingresado sea suficiente para comprar un producto** que cuesta 15 soles. Escribe un programa que permita al usuario seguir ingresando monedas hasta alcanzar esa cantidad.



EJEMPLOS DE FOR

1. Dada una lista de temperaturas diarias **durante un mes**, escribe un programa que cuente cuántos días hubo temperaturas extremas, es decir, aquellas que fueron menores de 0°C o mayores de 35°C .
2. Un comerciante registra las ventas diarias de su tienda **durante una semana** en una lista. Escribe un programa que calcule el total de las ventas de la semana y determine en qué día se alcanzó el mayor volumen de ventas.
3. Un supermercado realiza un seguimiento de la cantidad de veces que se vende un producto específico **cada día del mes**. Dado un listado con las ventas diarias, escribe un programa que cuente cuántas veces se vendió un producto que aparece en el listado.
4. Imagina que tienes una cuenta bancaria en la que **los clientes pueden realizar depósitos y retiros**. La cuenta comienza con un saldo de 0. Escribe un programa que reciba una lista de transacciones (positivas para depósitos y negativas para retiros) y calcule el saldo final de la cuenta después de todas las transacciones.
5. En una escuela, se registra la calificación de **cada estudiante** en un examen. Escribe un programa que, dado un conjunto de calificaciones, calcule el promedio de las calificaciones y cuente cuántos estudiantes aprobaron (una calificación mayor o igual a 60).
6. En una sala de cine, los tickets se venden en **varias sesiones** a lo largo del día. Cada sesión tiene un número limitado de tickets disponibles. Escribe un programa que simule la venta de boletos para diferentes sesiones y determine cuántos tickets quedan disponibles después de que un número específico de personas haya comprado entradas.



ITERADORES

Un iterable es cualquier objeto en Python (o en otros lenguajes de programación) que puede ser recorrido o iterado, es decir, un objeto sobre el cual se puede realizar un ciclo.

Código:

```
frutas = ["manzana", "banana", "cereza"]  
  
for fruta in frutas:  
    print(fruta)
```

Lo que realiza es lo siguiente

Tenemos la lista Frutas y dentro de ella hay elementos: manzana, banana, cereza. Para acceder a cada elemento ello utilizamos un bucle.

- 1 frutas = ["manzana", "banana", "cereza"]
 ↑
 for fruta in frutas:
 print(fruta) --> manzana

- 2 frutas = ["manzana", "banana", "cereza"]
 ↑
 for fruta in frutas:
 print(fruta) --> banana

- 3 frutas = ["manzana", "banana", "cereza"]
 ↑
 for fruta in frutas:
 print(fruta) --> cereza



ACUMULABLES

Un acumulador es una variable que **se utiliza para almacenar y acumular valores durante la ejecución de un bucle o de una operación**. Generalmente, se usa para sumar, multiplicar o concatenar valores de una colección de datos.

"En los acumuladores, el valor inicial depende de la operación que se deseé realizar. Para una suma, siempre se inicia en 0, ya que este es el elemento neutro de la suma (cualquier número sumado a 0 no cambia su valor)."

"En el caso de una multiplicación, el acumulador comienza en 1, porque este es el elemento neutro de la multiplicación (cualquier número multiplicado por 1 permanece igual)."



Ejemplo:

```
numeros = [1, 2, 3, 4, 5]
suma = 0
for numero in numeros:
    suma += numero
print("La suma es:", suma)
```

```
numeros = [1, 2, 3, 4, 5]
producto = 1
for numero in numeros:
    producto *= numero
print("La multiplicación es:", producto)
```



CONTADORES

Un contador es una variable que se utiliza para **contar cuántas veces ocurre un evento específico**, generalmente incrementando su valor en cada iteración de un bucle. Es común usarlo cuando se quiere contar cuántos elementos cumplen una determinada condición.

```
# Lista de números
numeros = [1, 2, 3, 4, 5, 6]

# Inicializamos el contador
contador = 0

# Iteramos sobre la lista
for numero in numeros:
    if numero % 2 == 0: # Comprobamos si el número es par
        contador += 1 # Incrementamos el contador si es par

# Imprimimos el resultado
print("Cantidad de números pares:", contador)
```



EJERCICIO PARA PENSAR

1. En toda empresa hay procesos repetitivos. Tu misión es identificar un proceso cíclico real en una empresa y explicarlo como si fuera un bucle de programación.

- ¿Qué se repite?
- ¿Cuál es la condición para que se repita?
- ¿Qué evento o dato rompe ese ciclo?
- ¿Qué recursos o resultados se acumulan con cada repetición?

2. Imagina que tienes que explicar el concepto de bucle a alguien que nunca ha programado, sin usar código. Crea una analogía, historia o situación real que represente un bucle.

Puedes basarte en actividades del hogar, rutinas diarias, procesos empresariales o situaciones naturales.

- Tu ejemplo debe incluir: qué se repite, cuándo se detiene y cuál es su propósito.

3. Imagina que en una empresa hay un proceso que se repite constantemente y nunca se detiene.

Reflexiona:

- ¿Cómo se vería un "bucle infinito" en la vida real o en una empresa?
- ¿Qué consecuencias tendría?
- ¿Qué se debería hacer para evitarlo o detenerlo?

NOTA: RESOLVER CON LA MENTE





UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA

**"REALIZAR LOS EJERCICIOS PROPUESTOS
Y REVISAR LOS MATERIALES DEL PROFESOR"**

