

Python Básico

Prof. Ariel Palazzesi

Comenzamos a grabar la clase

Módulo 2:

Control de flujo

Clase 04

Condicionales I

1. Operadores relacionales.
2. Toma de decisiones: if, else, elif.
3. Operadores Lógicos.

Clase 05

Bucles while

1. Bucle `while`.
2. Contadores.
3. Acumuladores.

Clase 06

Bucle for

1. Bucle for.
2. Recorriendo cadenas con for.
3. Uso de range()

Estructuras repetitivas

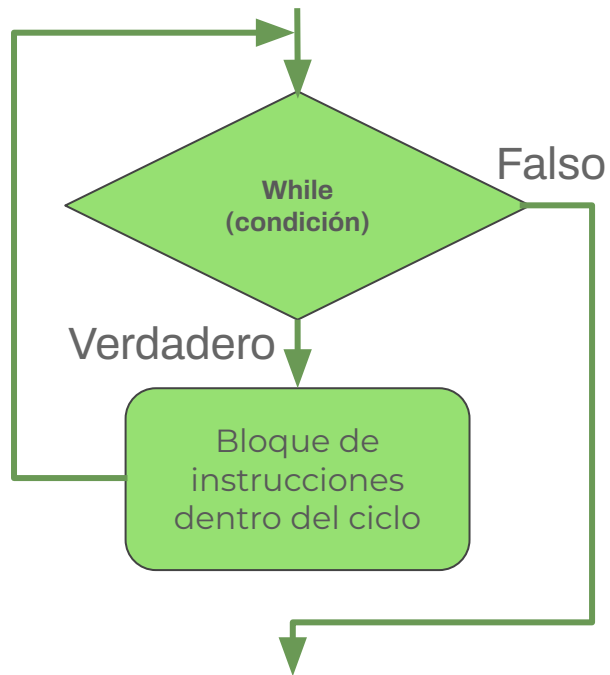
Estructuras repetitivas (bucles)

✓ Son las que repiten un bloque de código dependiendo de una **condición**. Mientras se cumpla la condición, se ejecuta el bloque de código. En cada ciclo se comprueba la condición. Estas estructuras pueden ser de dos clases:

- **Ciclos Exactos:** Se conoce la cantidad exacta de repeticiones a realizar, es un valor establecido antes de que se inicie el bucle.
- **Ciclos Condicionales:** No se conoce de antemano la cantidad de repeticiones, pues depende de una condición que se debe evaluar. Se puede repetir una vez, varias veces o ninguna vez.

Bucle while

Bucle while



✓ **while** ejecuta un bloque de código mientras la condición del while es **verdadera**. Finaliza cuando la condición es **falsa**. No sabemos de antemano el número de veces que se va a repetir.

```
while condición:  
    sentencia 1  
    sentencia 2  
siguiente sentencia fuera del while
```


Bucle while



Veamos un ejemplo sencillo para entender cómo funciona:

Código Python

```
contador = 1

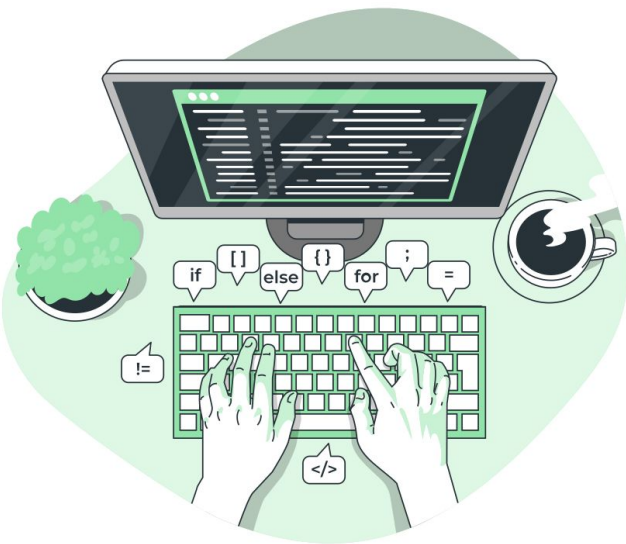
while contador <= 5:
    print("Bucle número:", contador)
    contador = contador + 1

print("Bucle terminado.")
```

Terminal

```
Este es el bucle número: 1
Este es el bucle número: 2
Este es el bucle número: 3
Este es el bucle número: 4
Este es el bucle número: 5
Bucle terminado.
```


Bucle while | Bucles infinitos



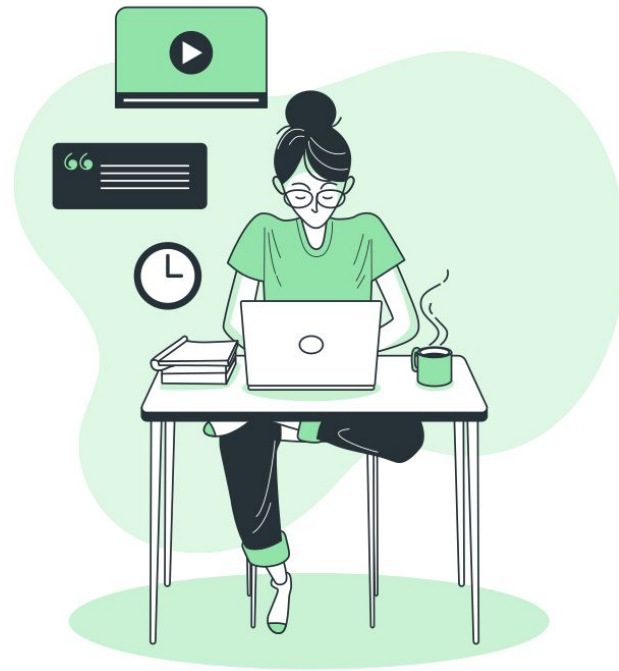
✓ Es importante que dentro del bucle haya algo que eventualmente haga que la condición sea False, porque si no, el programa entraría en lo que llamamos un "**bucle infinito**".

Un bucle infinito sigue ejecutándose sin detenerse y puede hacer que tu programa deje de responder. Se pueden crear intencionalmente, con **True** como condición.

Bucle while | Acumulador

✓ **continue** se utiliza para omitir el resto del código en la iteración actual y pasar directamente a la siguiente. A diferencia de **break**, **continue** sólo salta la ejecución restante en la iteración actual, permitiendo que el bucle continúe funcionando como está diseñado.

Esto es útil cuando para evitar ciertas condiciones específicas dentro del bucle, pero sin detener todo el proceso.



Bucle while | Acumulador | Ejemplo

Código Python

```
# Obtener la suma y promedio de 5 valores.
cont = 1
suma = 0
while cont <= 5:
    num = int(input("Número: "))
    suma = suma + num    # Acumulador
    cont = cont + 1      # Contador

print("La suma es:", suma)
print("El promedio es:", suma/cont)
```

Terminal

```
Número: 6
Número: 8
Número: 10
Número: 4
Número: 2
La suma es: 30
El promedio es: 5.0
```

Bucle while | Break



✓ **break** se utiliza para terminar un bucle de forma inmediata, sin esperar a que se cumpla la condición del bucle. Cuando se ejecuta break, el programa salta automáticamente a la siguiente línea de código después del bucle, ignorando cualquier iteración restante.

Es útil para salir de un bucle antes de que su condición se vuelva **False**, por ejemplo, al encontrar un resultado deseado o al detectar una situación específica.

Bucle while | Break | Ejemplo

Código Python

```
# Obtener la suma de valores positivos.
# Termina cuando el usuario ingresa 0 (ceo).
suma = 0
while True:
    numero = int(input("Ingresa un número: "))
    if numero < 0:
        print("Número negativo. Intentá de nuevo.")
    else:
        suma = suma + numero
    if numero == 0:
        break # Salimos del bucle
print("La suma es:", suma)
```

Terminal

```
Ingresa un número: 10
Ingresa un número: 20
Ingresa un número: 5
Ingresa un número: -2
Número negativo. Intentá de
nuevo.
Ingresa un número: 3
Ingresa un número: 0
La suma es: 38
```

Bucle while | Continue

✓ **continue** se utiliza para omitir el resto del código en la iteración actual y pasar directamente a la siguiente. A diferencia de **break**, **continue** sólo salta la ejecución restante en la iteración actual, permitiendo que el bucle continúe funcionando como está diseñado.

Esto es útil cuando para evitar ciertas condiciones específicas dentro del bucle, pero sin detener todo el proceso.



Bucle while | Continue | Ejemplo

Código Python

```
# Obtener la suma de valores positivos.
# Salimos cuando el usuario ingresa 0
suma = 0
while True:
    numero = int(input("Ingresa un número: "))
    if numero < 0:
        print("Número negativo. Intentá de nuevo.")
        continue # Saltamos el resto de la iteración
    if numero == 0:
        break
    suma = suma + numero
print("La suma es:", suma)
```

Terminal

```
Ingresa un número: 10
Ingresa un número: 20
Ingresa un número: 5
Ingresa un número: -2
Número negativo. Intentá de
nuevo.
Ingresa un número: 3
Ingresa un número: 0
La suma es: 38
```

Desafíos

Desafíos de la clase:



✓ Desafío 1: Ingresar nombre de usuario

Escribe un programa que pida el nombre del usuario.

Si el nombre es una cadena vacía, volver a pedirlo hasta que se ingrese un nombre válido.

Desafíos de la clase (Resolución):

✓ Desafío 1: Ingresar nombre de usuario.

Código Python

```
nombre = ""

# El bucle while se ejecuta mientras la cadena esté vacía
while not nombre.strip():
    nombre = input("Por favor, ingresa tu nombre: ")
    if not nombre:
        print("El nombre no puede estar vacío. Inténtalo de nuevo.")

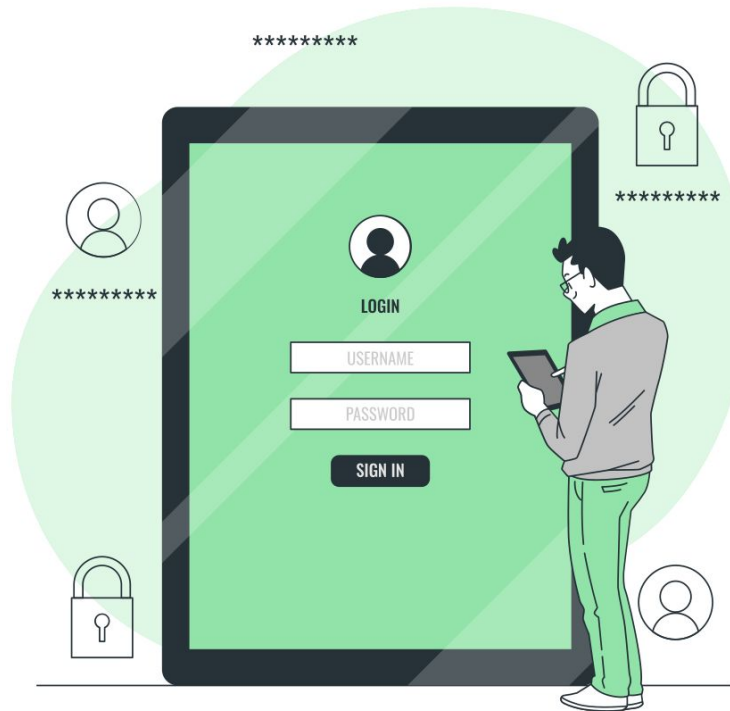
print(";Hola,", nombre, "! Bienvenido.")
```

Desafíos de la clase:

✓ Desafío 2: Sistema de login

Escribe un programa que pida el nombre del usuario y su contraseña. Si no son correctos luego de tres intentos, mostrar un texto indicando que el acceso ha fallado.

Caso contrario, permitir el acceso al sistema.



Desafíos de la clase (Resolución):



Desafío 2: Sistema de login (parte 1 de 2)

Código Python

```
# Datos correctos predefinidos
USUARIO_CORRECTO = "admin"
CONTRASENA_CORRECTA = "1234"
intentos = 0
max_intentos = 3

# Bucle while para permitir hasta tres intentos
while intentos < max_intentos:
    usuario = input("Ingresa tu nombre de usuario: ")
    contrasena = input("Ingresa tu contraseña: ")
```


Desafíos de la clase (Resolución):

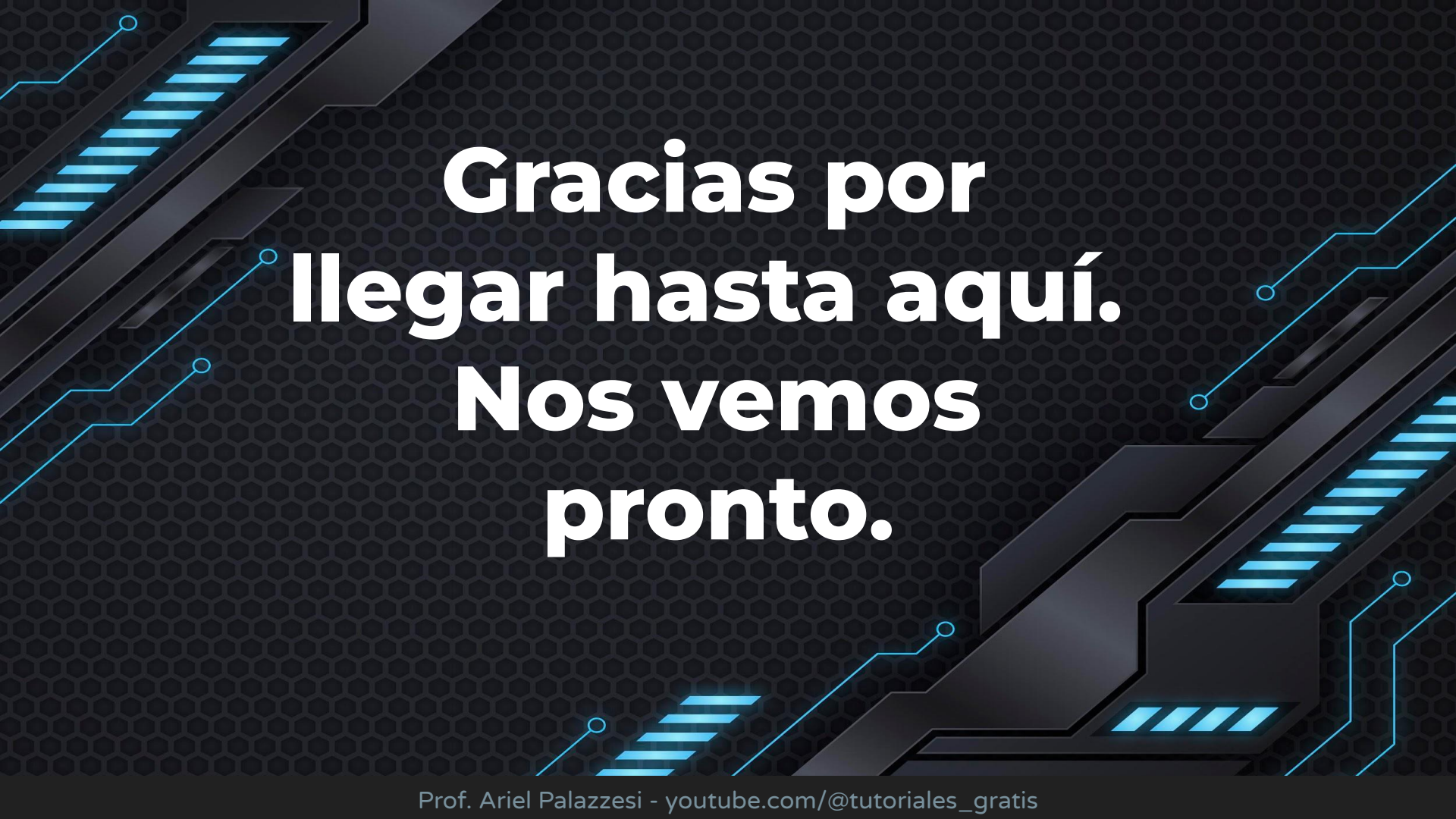


Desafío 2: Sistema de login (parte 2 de 2)

Código Python

```
if usuario == USUARIO_CORRECTO and contrasena == CONTRASENA_CORRECTA:
    print(";Acceso concedido! Bienvenido al sistema.")
    break
else:
    intentos += 1
    print("Usuario o contraseña incorrectos. Intento N°", intentos)

# Si se superan los tres intentos
if intentos == max_intentos:
    print("Acceso fallido. Has superado el número máximo de intentos.")
```



**Gracias por
llegar hasta aquí.
Nos vemos
pronto.**