

ALGORITMIA

Juan Carlos Molina Lozano
Docente

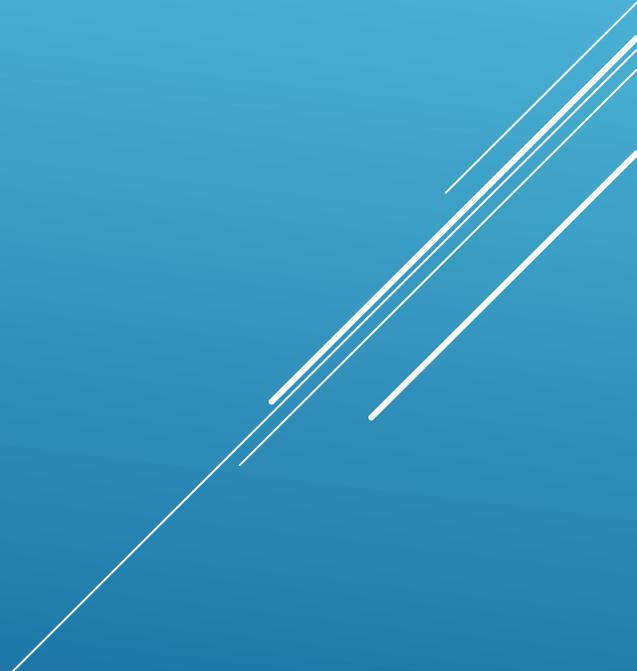
OBJETIVOS DE LA CLASE

1. Comprender qué es un ciclo do-while y su propósito.
 2. Analizar por qué Python no tiene una estructura do-while directa.
 3. Aprender a simular correctamente un ciclo do-while en Python.
 4. Aplicar esta simulación en ejemplos prácticos y resolución de problemas.
- 
- Decorative white lines consisting of several parallel diagonal strokes in the bottom right corner of the slide.

METODOLOGÍA

1. Se aplicará un enfoque constructivista donde los estudiantes aprenderán a través de la exploración y la resolución de problemas.
2. Se fomentará el aprendizaje activo mediante el desarrollo de ejercicios prácticos y proyectos sencillos.
3. Se utilizarán ejemplos del mundo real para contextualizar el uso de Python.
4. Se promoverá el trabajo colaborativo a través de discusiones y resolución de problemas en grupo.
5. Se incentivará la autoevaluación y el aprendizaje basado en errores para fortalecer la comprensión del lenguaje.

CONTENIDO

1. Introducción al ciclo do-while
 2. Por qué Python no tiene do-while?
 3. Simulación del do-while en Python
 4. Sintaxis y estructura
 5. Ventajas y desventajas
 6. Ejercicios prácticos
 7. Actividad en claseConclusiones
- 

INTRODUCCIÓN A ESTRUCTURA REPETITIVA DO WHILE – HACER MIENTRAS

El ciclo do-while es una estructura de control utilizada en muchos lenguajes de programación para ejecutar un bloque de instrucciones al menos una vez, y luego repetirlo mientras se cumpla una condición.

Estructura general en pseudocódigo:

```
hacer
    Instrucciones
mientras (condición)
```

Ejemplo en Java:

```
int x = 0;
do {
    System.out.println(x);
    x++;
} while (x < 5);
```

Este ciclo garantiza que el bloque se ejecute por lo menos una vez, incluso si la condición nunca se cumple desde el inicio.

¿POR QUÉ PYTHON NO TIENE DO-WHILE?

Python promueve la claridad y simplicidad en la sintaxis. El equipo de diseño de Python decidió no incluir do-while (hacer-mientras) como una estructura nativa para evitar duplicación y mantener el lenguaje limpio.

Sin embargo, este comportamiento puede simularse fácilmente usando un ciclo while infinito (while True) y una instrucción break.

Sintaxis simulada del do-while en Python

```
while True:
```

```
    # instrucciones a ejecutar  
    al menos una vez
```

```
if not condición:
```

```
    break
```

¿Cómo funciona?

- while True: inicia un ciclo infinito.
- El bloque dentro del while se ejecuta al menos una vez.
- La condición se evalúa al final del ciclo.
- Si la condición ya no se cumple, se usa break para salir del bucle.

VENTAJAS Y DESVENTAJAS

✓ Ventajas

- Permite ejecutar un bloque de código al menos una vez
- Ideal para validación de entrada
- Flexible, fácil de implementar con `while True`

x Desventajas

- No tiene una sintaxis propia en Python
- Uso de `break` puede confundir a principiantes
- Menos legible que `do-while` en otros lenguajes

SOLUCIÓN DE PROBLEMAS UTILIZANDO DO WHILE – HACER MIENTRAS

Problema 1: Ingreso de número positivo

```
while True:
    num = int(input("Ingrese un número positivo: "))
    if num > 0:
        break
```

Problema 2: Validación de contraseña

```
password = "python123"
while True:
    entrada = input("Ingrese la contraseña: ")
    if entrada == password:
        break
    print("Incorrecta, intente de nuevo.")
```

SOLUCIÓN DE PROBLEMAS UTILIZANDO FOR

Problema 3: Pedir números hasta que el usuario escriba uno mayor que 100

```
while True:
    num = int(input("Ingrese un número mayor que 100: "))
    if num > 100:
        print("¡Correcto! Número aceptado.")
        break
    else:
        print("Número no válido, intente de nuevo.")
```

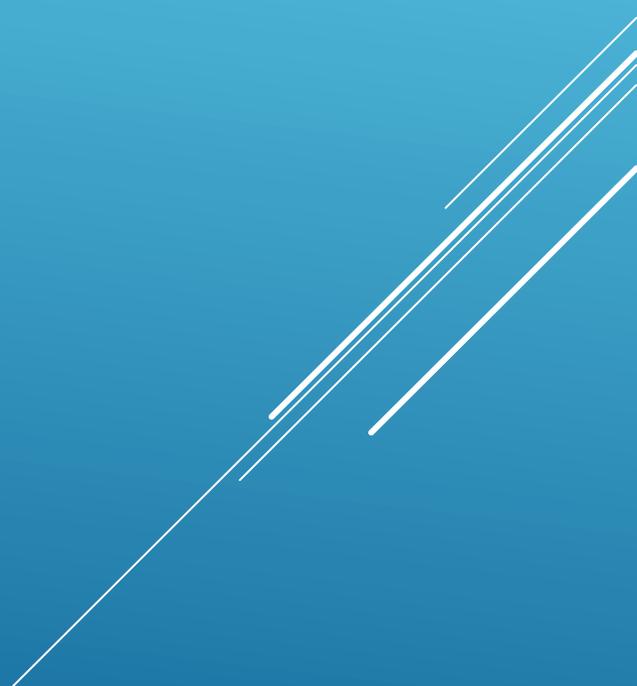
Problema 4: Solicitar una letra hasta que sea una vocal

```
while True:
    letra = input("Ingrese una letra vocal: ").lower()
    if letra in "aeiou" and len(letra) == 1:
        print("¡Es una vocal!")
        break
    else:
        print("No es una vocal. Intente de nuevo.")
```

CONCLUSIONES

1. Python no tiene ciclo do-while como estructura propia.
 2. Puede ser simulado con while True y una condición de ruptura (break).
 3. Este patrón se usa comúnmente para validaciones de entrada y menús interactivos.
 4. Es importante controlar correctamente la condición de salida para evitar bucles infinitos.
- 

EJERCICIOS

1. Pedir un PIN hasta que sea correcto (predefinido).
 2. Mostrar un menú que se repite hasta seleccionar "Salir".
 3. Leer una palabra hasta que tenga más de 5 letras.
 4. Sumar números ingresados por el usuario hasta que escriba "0".
 5. Simular un login que permite 3 intentos máximos.
 6. Calcular promedio de notas hasta que el usuario escriba "fin".
 7. Adivinar un número aleatorio hasta acertar.
 8. Simular una calculadora simple que se repite hasta salir.
- 

PREGUNTAS

