ALGORITMIA

Juan Carlos Molina Lozano

Docente

OBJETIVOS DE LA CLASE

1. Comprender el funcionamiento del ciclo for.
2. Identificar sus diferencias con el ciclo while.
3. Aplicar la estructura for en la solución de problemas reales.
4. Promover la reflexión a través de ejercicios prácticos guiados.

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. ¿Qué es un ciclo for?
2. Estructura del for en Python
3. Uso de range()
4. Comparación con el ciclo while
5. Ejemplos básicos y prácticos
6. Solución de problemas con for
7. Ventajas y desventajas del ciclo for
8. Ejercicios de aplicación

INTRODUCCIÓN A ESTRUCTURA REPETITIVA FOR

En programación, una estructura repetitiva permite ejecutar un bloque de código varias veces. El ciclo for es una de estas estructuras, utilizada para repetir instrucciones un número determinado de veces o para recorrer secuencias como listas, cadenas, tuplas o rangos numéricos.

El ciclo for se caracteriza por ser determinado, es decir, se conoce de antemano la cantidad de repeticiones que va a realizar. A diferencia del ciclo while, que depende de una condición lógica que puede cumplirse o no en tiempo de ejecución, el for itera sobre los elementos de una secuencia, desde el primero hasta el último, de forma ordenada.

INTRODUCCIÓN A ESTRUCTURA REPETITIVA FOR

En el lenguaje Python, el ciclo for se emplea principalmente con la función range() o con colecciones de datos, lo que lo hace muy versátil y adecuado para tareas como:

- Repetir procesos controlados por un contador.
- Recorrer listas, cadenas o diccionarios.
- Procesar datos en estructuras iterables.
- Generar patrones, listas numéricas o cálculos acumulativos.

SINTAXIS ESTRUCTURA REPETITIVA FOR (PARA)

```
for variable in secuencia:
    # Instrucciones que se repiten.
```

Donde:

- **variable** es una variable temporal que toma el valor de cada elemento dentro de la secuencia.
- **secuencia** puede ser una lista, cadena, tupla o un rango generado por range().

Ejemplo:

```
for i in range(5):
    print(i)
```

Salida

```
0
1
2
3
4
```

En este ejemplo, i toma los valores desde 0 hasta 4, ejecutando la instrucción print(i) cinco veces.

VENTAJAS Y DESVENTAJAS

✓ Ventajas

- Simplicidad cuando el número de repeticiones es conocido.
- Código más corto y claro.
- Ideal para recorrer listas, cadenas o rangos.

x Desventajas

- Menor flexibilidad para condiciones dinámicas.
- No apto para repetir "hasta que" se cumpla una condición sin saber cuántas veces.

USO DE RANGE() EN UN CICLO FOR

Uno de las iteraciones mas comunes que se realizan, es la de iterar un número entre por ejemplo 0 y n. Si ya programas, estoy seguro de que estas cansado de escribir esto, aunque sea en otro lenguaje. Pongamos que queremos iterar una variable i de 0 a 5. Haciendo uso de lo que hemos visto anteriormente, podríamos hacer lo siguiente.

```
for i in (0, 1, 2, 3, 4, 5):
    print(i) #0, 1, 2, 3, 4, 5
```

Se trata de una solución que cumple con nuestro requisito. El contenido después del in se trata de una clase que como ya hemos visto antes, es iterable, y es de hecho una tupla. Sin embargo, hay otras formas de hacer esto en Python, haciendo uso del range().

```
for i in range(6):
    print(i) #0, 1, 2, 3, 4, 5
```

USO DE RANGE() EN UN CICLO FOR

El range() genera una secuencia de números que van desde 0 por defecto hasta el número que se pasa como parámetro menos 1. En realidad, se pueden pasar hasta tres parámetros separados por coma, donde el primer es el inicio de la secuencia, el segundo el final y el tercero el salto que se desea entre números. Por defecto se empieza en 0 y el salto es de 1.

#range(inicio, fin, salto)

Por lo tanto, si llamamos a range() con (5,20,2), se generarán números de 5 a 20 de dos en dos. Un truco es que el range() se puede convertir en list.

print(list(range(5, 20, 2)))

Y mezclándolo con el for, podemos hacer lo siguiente.

```
for i in range(5, 20, 2):
    print(i) #5,7,9,11,13,15,17,19
```

USO DE RANGE() EN UN CICLO FOR

Se pueden generar también secuencias inversas, empezando por un número mayor y terminando en uno menor, pero para ello el salto deberá ser negativo.

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

SINTAXIS	SIGNIFICADO
range(n)	0 hasta n-1
range(inicio, fin)	desde inicio hasta fin-1
range(inicio, fin, paso)	controla también el incremento

DIFERENCIAS CON WHILE

Diferencias del ciclo FOR y ciclo WHILE

Aspecto	for	while
Uso común	Repeticiones conocidas	Repeticiones condicionadas
Control	Automático con range()	Manual (condición + actualización)
Riesgo de bucle infinito	Bajo	Alto si no se actualiza la condición

SOLUCIÓN DE PROBLEMAS UTILIZANDO FOR

Problema 1: Recorrer una cadena

```
palabra = "Python"
for letra in palabra:
    print(letra)
```

Problema 2: Sumar los números del 1 al 100

```
suma = 0
for i in range(1, 101):
    suma += i
print("Suma:", suma)
```

SOLUCIÓN DE PROBLEMAS UTILIZANDO FOR

Problema 3: Mostrar solo los múltiplos de 3 hasta el 30

```
for i in range(3, 31, 3):
    print(i)
```

Problema 4: Contar cuántas vocales tiene una palabra

```
palabra = "algoritmo"
contador = 0
for letra in palabra:
    if letra in "aeiou":
        contador += 1
print("Cantidad de vocales:", contador)
```

SOLUCIÓN DE PROBLEMAS UTILIZANDO WHILE

```
numero = int(input("Ingrese un número (0 para terminar): "))
suma = 0
while numero != 0:
    suma += numero
    numero = int(input("Ingrese otro número: "))
print("La suma total es:", suma)
```

EJERCICIOS

1. Imprimir los números del 1 al 10.
2. Calcular el factorial de un número.
3. Imprimir los caracteres de una palabra al revés.
4. Generar los cuadrados de los números del 1 al 5.
5. Imprimir una escalera de asteriscos.
6. Leer 5 notas, calcular el promedio.
7. Contar cuántos números pares hay en una lista.
8. Imprimir los números pares del 1 al 20
9. Realizar la tabla del 7
10. Contar cifras de un número positivo

PREGUNTAS