

Estructuras de Control en Python

Programación en Python

Introducción

¿Qué son las estructuras de control?

Son elementos fundamentales que permiten controlar el flujo de ejecución de un programa.

- Control condicional: `if-else`
- Bucles: `for`, `while`, `do-while`
- Operadores condicionales

Importancia

Permiten crear programas dinámicos que toman decisiones y repiten tareas.

Operadores Condicionales

Operadores de Comparación

Operador	Descripción
==	Igual a
!=	Diferente de
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

Operadores Lógicos

Operador	Descripción
and	Y lógico
or	O lógico
not	Negación lógica

Estructura if-else

Sintaxis Básica

```
1 if condicion:
2     # codigo si la condicion es True
3 else:
4     # codigo si la condicion es False
```

Ejemplo

```
1 edad = 18
2 if edad >= 18:
3     print("Mayor de edad")
4 else:
5     print("Menor de edad")
```

Estructura if-elif-else

Múltiples Condiciones

```
1 if condicion1:
2     # codigo si condicion1 es True
3 elif condicion2:
4     # codigo si condicion2 es True
5 else:
6     # codigo si ninguna es True
```

Ejemplo Práctico

```
1 nota = 85
2 if nota >= 90:
3     print("A")
4 elif nota >= 80:
5     print("B")
6 else:
7     print("C")
```

Bucle for

Sintaxis Básica

```
1 for variable in secuencia:  
2     # codigo a repetir
```

Ejemplos Comunes

```
1 # Con range()  
2 for i in range(5):  
3     print(i)  
4  
5 # Con lista  
6 frutas = ["manzana", "banana", "naranja"]  
7 for fruta in frutas:  
8     print(fruta)
```

Bucle while

Sintaxis Básica

```
1 while condicion:
2     # codigo a repetir mientras
3     # la condicion sea True
```

Ejemplo Práctico

```
1 contador = 0
2 while contador < 5:
3     print(f"Contador: {contador}")
4     contador += 1
```

Bucle do-while (simulado)

Python no tiene do-while nativo

Se simula usando while True con break

Implementación

```
1 while True:
2     # codigo que se ejecuta al menos una vez
3     respuesta = input(" Continuar ? (s/n): ")
4     if respuesta.lower() != 's':
5         break
```

Característica

El código dentro del bucle se ejecuta **al menos una vez**.

Control de Bucles

Palabras Clave

break	Termina el bucle completamente
continue	Salta a la siguiente iteración
pass	No hace nada (marcador de posición)
else	Se ejecuta si el bucle termina normalmente

Ejemplo con break y continue

```
1 for i in range(10):  
2     if i == 3:  
3         continue    # Salta el 3  
4     if i == 7:  
5         break       # Termina en 7  
6     print(i)  
7 else:  
8     print("Bucle completado")
```