Toma de decisiones con lógica: Estructuras condicionales en algoritmos y Python

Una introducción completa a las estructuras de control condicional para programadores nóveles.

By OscarLondero

Objetivos del curso

1

Comprensión de las estructuras de control

Identificar y utilizar estructuras condicionales **simples** (if) y **compuestas** (if-elif-else) para **controlar** el flujo del programa en función de las **condiciones** lógicas.

2

Diseño y desarrollo de algoritmos

Crear algoritmos que empleen **decisiones** simples, múltiples y anidadas, aplicando estructuras condicionales para resolver problemas reales.

3

Resolución de problemas y pensamiento crítico

Analizar situaciones, dividirlas en partes más simples y aplicar la **lógica** condicional para resolverlas a través del código.

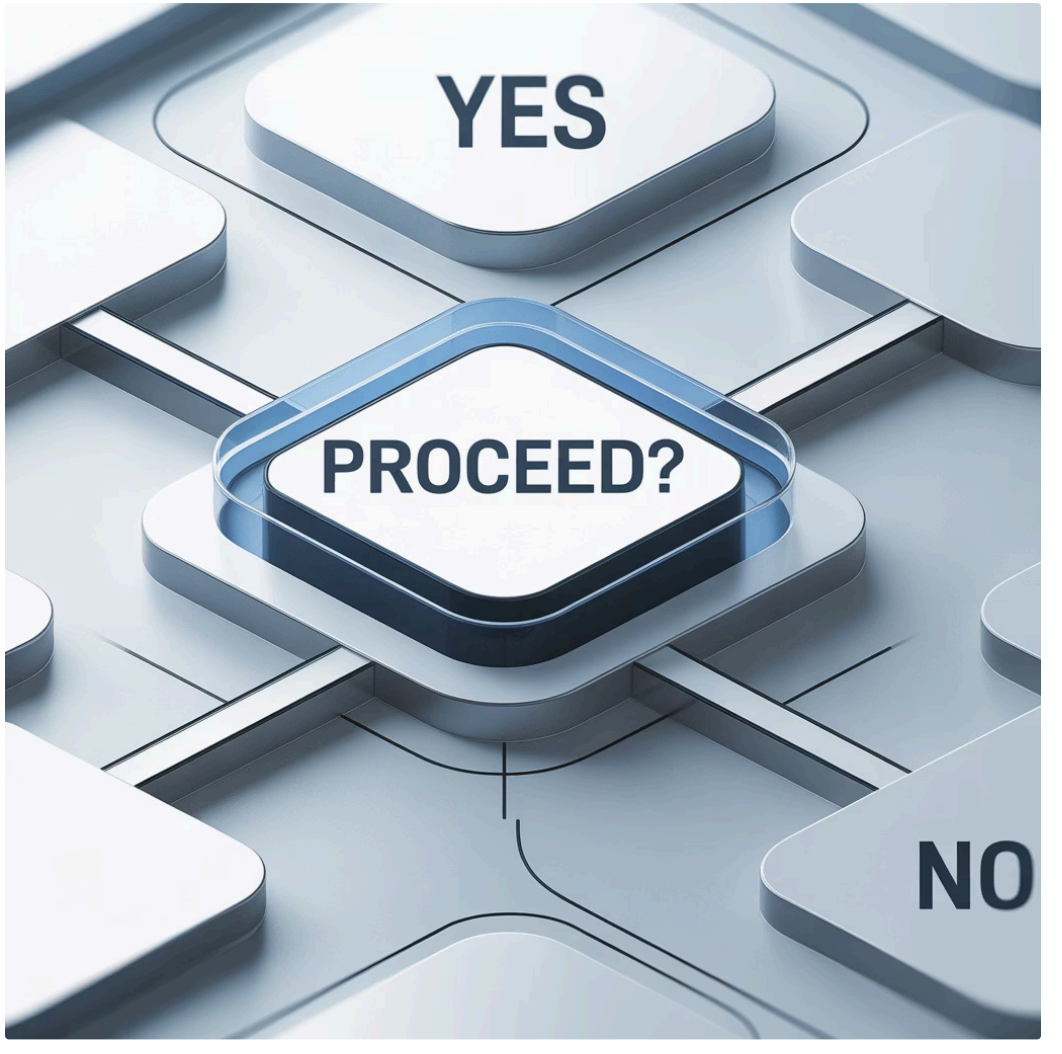
¿Qué son las estructuras condicionales?

Las estructuras condicionales son mecanismos de control que permiten a los algoritmos tomar decisiones basadas en **si se cumple o no** una condición.

Se podría pensar en ellas como "caminos alternativos" para el fluir de los datos dentro un programa:

- Si sucede algo, haz una cosa
- Si **no**, haz otra cosa

Representan el pensamiento lógico detrás de los programas informáticos y son esenciales para crear software dinámico y receptivo.



Analogías del mundo real

Semáforo

Si la luz está en rojo, detente. Si está en verde, avanza. Si está en amarillo, prepárate para detenerte. Estas son decisiones condicionales que tomamos todos los días.

Termostato

Si la temperatura está por debajo del ajuste, enciende la calefacción. Si está por encima, enciende el enfriamiento. Un sistema condicional simple en un hogar inteligente.

Planificación del clima

Si está lloviendo, lleva un paraguas. De lo contrario, déjalo en casa. Usamos la lógica condicional constantemente en la toma de decisiones diarias.

Tipos de estructuras condicionales en Python

Python ofrece varias formas de implementar la toma de decisiones en el código:

Condicional simple (if)

Ejecuta un bloque de código solo si una condición es verdadera

Condicional compuesto (if-else)

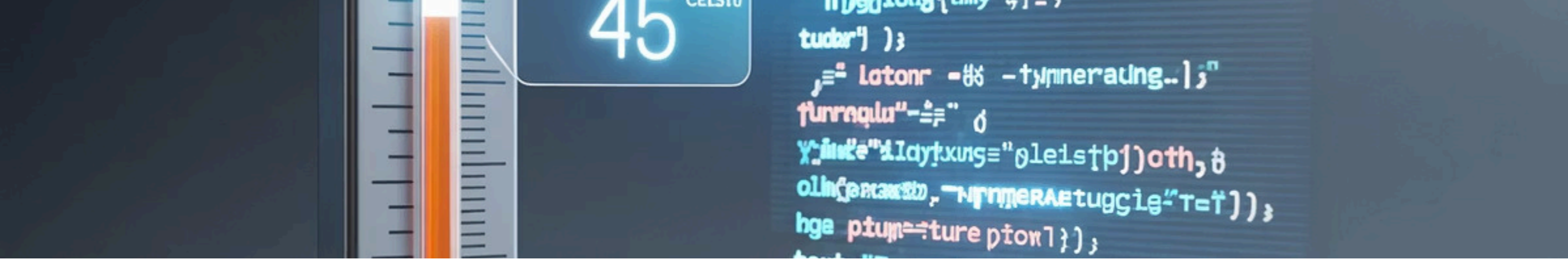
Ejecuta un bloque si la condición es verdadera, otro si es falsa

Condicional múltiple (if-elif-else)

Maneja múltiples condiciones y rutas posibles

Condicional anidado

Coloca estructuras condicionales dentro de otras condicionales



Condicional simple (if)

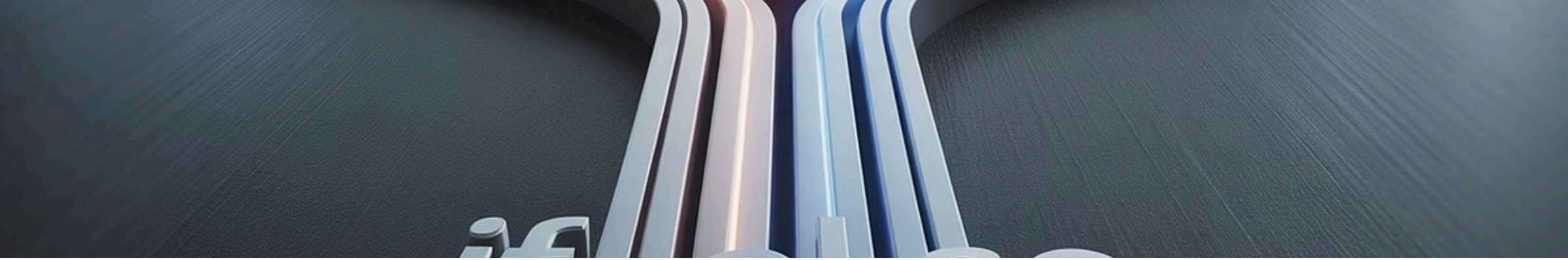
El condicional simple ejecuta un bloque de código solo cuando una condición específica se evalúa como Verdadera. Si la condición es Falsa, el bloque de código se omite.

Esta es la forma más básica de toma de decisiones en la programación.

```
# Ejemplo de condicional simple
if temperature > 30:
    print("¡Hace calor afuera!")
    print("Recuerda mantenerte hidratado")

# El programa continúa aquí independientemente
print("Fin de la verificación del clima")
```

Se puede observar que en python la indentación define el bloque de código que pertenece al condicional.



Condicional compuesto (if-else)

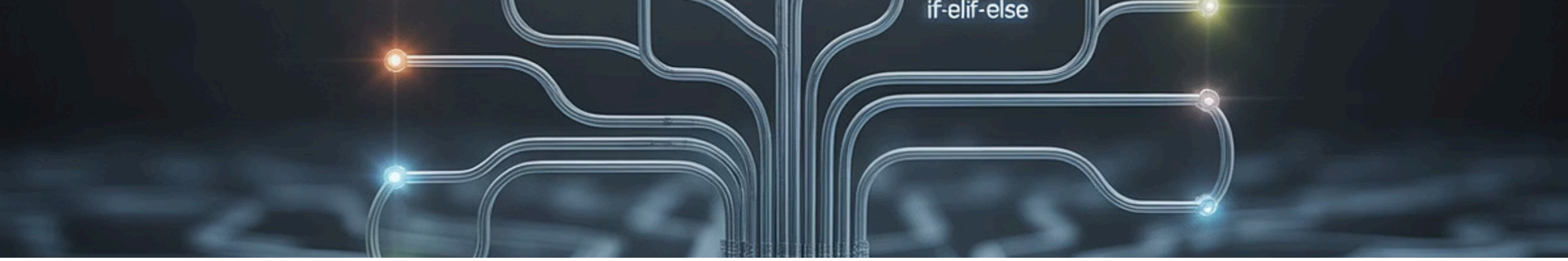
El condicional compuesto proporciona una ruta alternativa cuando la condición es Falsa. Esto crea una estructura de decisión binaria:

- Si la condición es Verdadera → Ejecutar el bloque "if"
- Si la condición es Falsa → Ejecutar el bloque "else"

```
# Ejemplo de condicional compuesto
if age >= 18:
    print("Eres un adulto")
    print("Puedes votar")
else:
    print("Eres menor de edad")
    print("Aún no puedes votar")

print("Verificación de edad completada")
```

Esta estructura asegura que exactamente uno de los dos bloques de código **siempre se ejecutará**.



Condicional Múltiple (if-elif-else)

Cuando se necesite verificar múltiples condiciones y tomar diferentes acciones **basadas en cuál es verdadera,** se debe usar la estructura if-elif-else (en python).

El programa **verifica** cada condición en orden **hasta** que una **se evalúa como Verdadera**, luego ejecuta ese bloque y se **salta** el resto.

```
# Ejemplo de condicional múltiple
if score >= 90:
    print("Calificación: A")
elif score >= 80:
    print("Calificación: B")
elif score >= 70:
    print("Calificación: C")
elif score >= 60:
    print("Calificación: D")
else:
    print("Calificación: F")
```

El bloque "**else**" es **opcional** y sirve como un comodín para cuando ninguna de las condiciones es Verdadera.

Condicionales anidados

Los condicionales anidados colocan una estructura condicional dentro de otra, lo que permite árboles de decisión más complejos y lógica multinivel.

Esto es útil cuando una segunda decisión depende del resultado de una primera decisión.

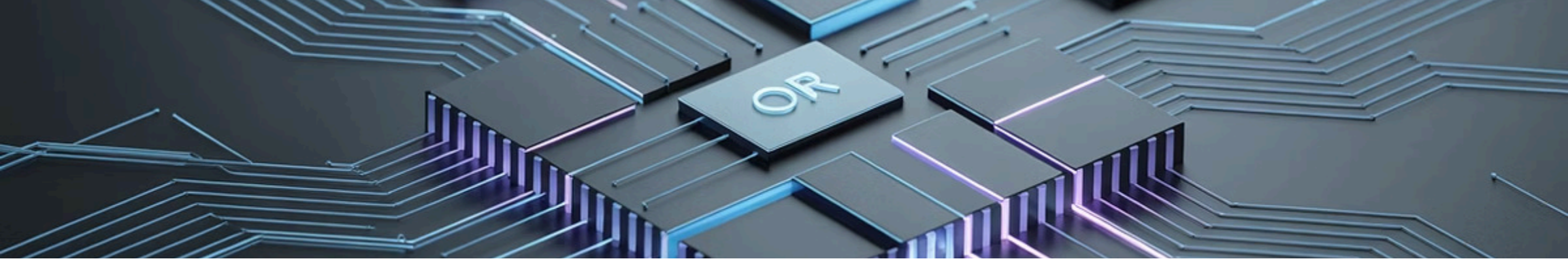
```
# Ejemplo de condicional anidado
if username == "admin":
 if password == "1234":
 print("Acceso de administrador concedido")
 else:
 print("Contraseña incorrecta")
else:
 print("Usuario desconocido")
```

Operadores relacionales en Python

Los operadores relacionales comparan valores y son fundamentales en las condiciones lógicas:

Operador	Significado	Ejemplo
==	Igual a	x == 5
!=	No es igual a o es distinto	x != 3
>	Mayor que	x > 10
<	Menor que	x < 7
>=	Mayor o igual que	x >= 18
<=	Menor o igual que	x <= 15

Estos operadores siempre se evalúan como **valores booleanos** (Verdadero o Falso) y forman la **base** de las expresiones condicionales.



Operadores lógicos en Python

and

Devuelve True si ambas expresiones son True

```
if age >= 18 and has_id == True:
```

Ambas condiciones deben ser True para que toda la expresión sea True

or

Devuelve True si al menos una expresión es True

```
if is_weekend or is_holiday:
```

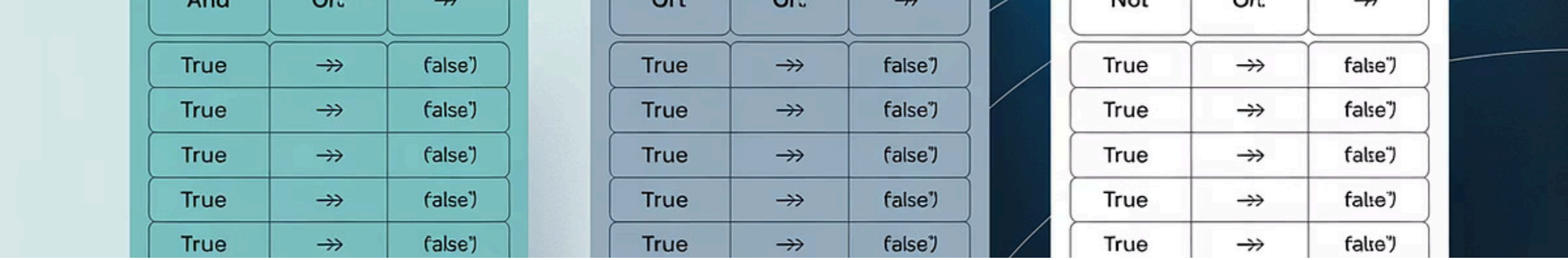
Solo una condición debe ser True para que toda la expresión sea True

not

Invierte el resultado de una expresión

```
if not is_working_hours:
```

Si la expresión es True, not la convierte en False y viceversa



Ejemplos de tablas de verdad

Operador AND

A	B	A y B
True	True	True
True	False	False
False	True	False
False	False	False

Operador OR

A	B	A o B
True	True	True
True	False	True
False	True	True
False	False	False

Operador NOT

A	no A
True	False
False	True

Combinación de operadores lógicos

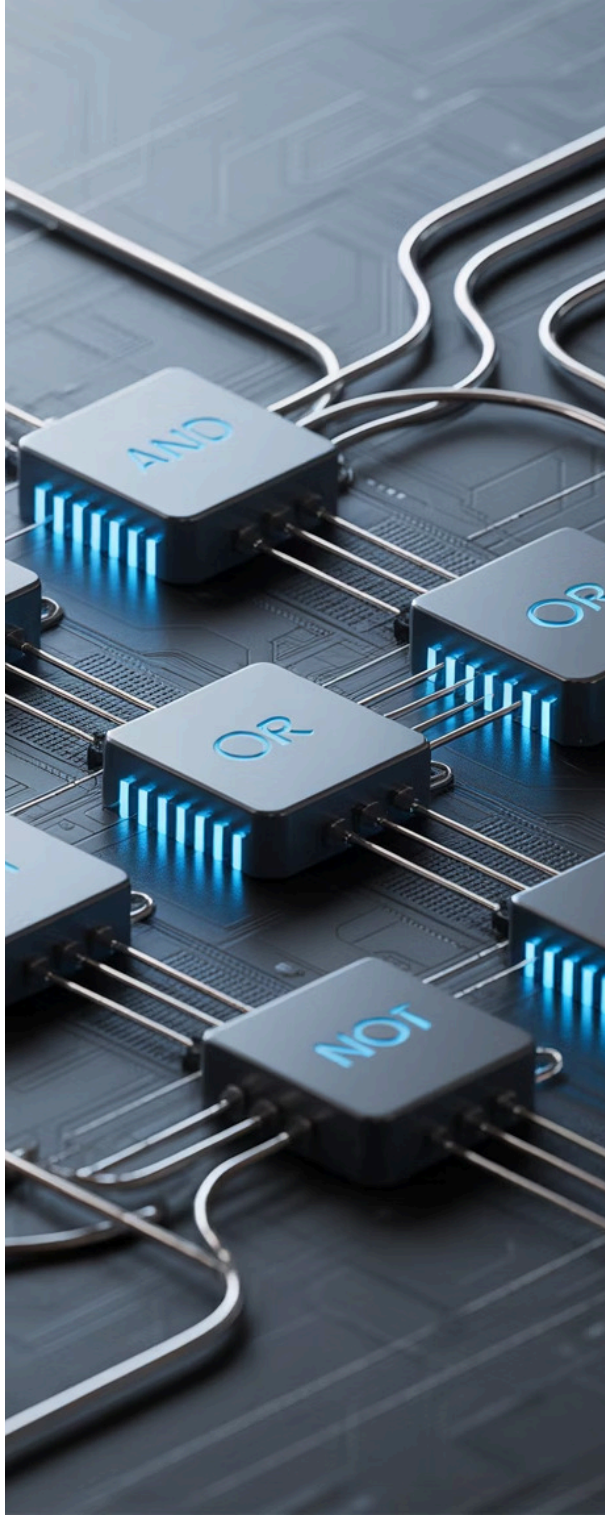
Puedes combinar múltiples operadores lógicos para crear condiciones complejas. Python los evalúa en este orden:

1. Paréntesis ()
2. not
3. and
4. or

Usa paréntesis para controlar el orden de evaluación y hacer que el código sea más legible.

```
# Ejemplo de condicional complejo
if (edad >= 18 and tiene_id) or acompañado_por_adulto:
 print("Entrada permitida")
else:
 print("Entrada denegada")

# Usando not con otros operadores
if not (es_fin_de_semana or es_festivo):
 print("Es un día laboral")
```



Ejemplo práctico 1: Aviso meteorológico

Construyamos un programa de avisos meteorológicos que ofrezca recomendaciones basadas en la temperatura y las condiciones de precipitación.

```
temperature = float(input("Ingrese la temperatura en °C: "))
is_raining = input("¿Está lloviendo? (s/n): ") == "s"

if temperature > 30:
 if is_raining:
 print("Calor y lluvia. Traiga un paraguas y use ropa liviana.")
 else:
 print("¡Mucho calor! Use protector solar y manténgase hidratado.")
elif temperature > 20:
 if is_raining:
 print("Cálido y lluvioso. Traiga un paraguas.")
 else:
 print("Clima agradable. ¡Disfrute su día!")
elif temperature > 10:
 if is_raining:
 print("Frío y lluvioso. Traiga una chaqueta y un paraguas.")
 else:
 print("Clima fresco. Es posible que necesite una chaqueta ligera.")
else:
 if is_raining:
 print("Frío y lluvioso. Lleve un abrigo abrigado y un paraguas.")
 else:
 print("Clima frío. ¡Abríguese!")
```

Errores comunes en las sentencias condicionales

Usar = en lugar de ==

El operador = asigna valores, mientras que == los compara.

```
# INCORRECTO
if age = 18:  # ¡Esto asigna 18 a age!

# CORRECTO
if age == 18:  # Esto compara age con 18
```

Indentación incorrecta

Python usa la indentación para definir bloques de código. La indentación inconsistente causa errores.

```
# INCORRECTO
if score > 90:
print("¡Excelente!")  # Falta la indentación

# CORRECTO
if score > 90:
    print("¡Excelente!") # ok
```

Olvidar los dos puntos (:)

Todas las sentencias condicionales en Python deben terminar con dos puntos.

```
# INCORRECTO
if temperature > 30
    print("¡Hace calor!")

# CORRECTO
if temperature > 30:
   print("¡Hace calor!")
```

Actividad práctica: Comparación de números

Desafío:

Crea un programa que lea dos números y determine cuál es mayor o si son iguales.

1. Obtén dos números del usuario
2. Compáralos usando condicionales
3. Muestra cuál es mayor o si son iguales
4. Calcula y muestra la diferencia

Solución:

```
# Comparación de números
num1 = float(input("Ingresa el primer número: "))
num2 = float(input("Ingresa el segundo número: "))

if num1 > num2:
 larger = num1
 smaller = num2
 print(f"{num1} es mayor que {num2}")
elif num2 > num1:
 larger = num2
 smaller = num1
 print(f"{num2} es mayor que {num1}")
else:
 print(f"Ambos números son iguales: {num1} = {num2}")
 larger = smaller = num1

if num1 != num2:
 difference = larger - smaller
 print(f"La diferencia es: {difference}")
```

Desafío avanzado: Calculadora de IMC

Problema:

Cree una calculadora de IMC (Índice de Masa Corporal) que:

1. Tome la altura (en metros) y el peso (en kg) como entrada
2. Calcule el IMC utilizando la fórmula: peso / (altura^2)
3. Clasifique el resultado según las categorías estándar
4. Proporcione recomendaciones de salud personalizadas

Solución de la calculadora de IMC

```
# Calculadora de IMC
altura = float(input("Ingrese su altura en metros: "))
peso = float(input("Ingrese su peso en kilogramos: "))

# Calcular el IMC
imc = peso / (altura ** 2)
print(f"Su IMC es: {imc:.2f}")

# Clasificar el IMC y proporcionar recomendaciones
if imc < 18.5:
 categoría = "Bajo peso"
 recomendación = "Considere consultar con un proveedor de atención médica sobre estrategias saludables para aumentar de peso."
elif imc < 25:
 categoría = "Peso normal"
 recomendación = "Mantenga su estilo de vida saludable con una dieta equilibrada y ejercicio regular."
elif imc < 30:
 categoría = "Sobrepeso"
 recomendación = "Considere incorporar más actividad física y revisar su dieta."
else:
 categoría = "Obeso"
 recomendación = "Se recomienda consultar con un proveedor de atención médica para obtener asesoramiento personalizado."

print(f"Categoría: {categoría}")
print(f"Recomendación: {recomendación}")
```

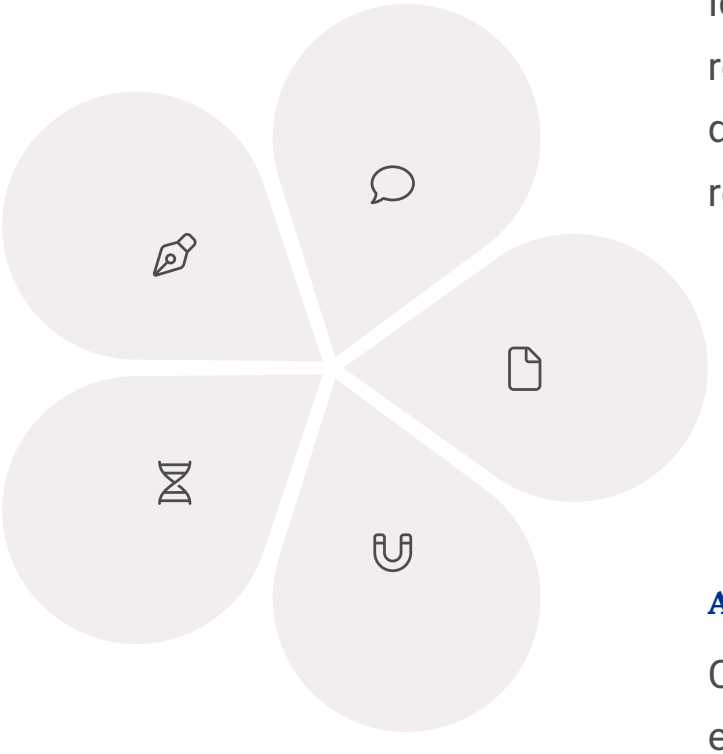
Pensamiento computacional con condicionales

Descomposición

Divida los problemas complejos en partes más pequeñas y manejables que se puedan resolver con condicionales simples.

Evaluación

Pruebe sus estructuras condicionales con diferentes entradas para asegurarse de que manejen todos los casos correctamente.



Reconocimiento de patrones

Identifique los puntos de decisión recurrentes en los problemas y desarrolle estructuras condicionales reutilizables.

Diseño de algoritmos

Construya soluciones paso a paso utilizando lógica condicional para manejar diferentes escenarios.

Abstracción

Concéntrese en los puntos de decisión esenciales e ignore los detalles innecesarios en su lógica condicional.

Aprender a formular condiciones y traducirlas en código desarrolla el pensamiento lógico y las habilidades sistemáticas para resolver problemas.

Conclusión: El poder de las estructuras condicionales

Las estructuras condicionales son la base de la toma de decisiones en la programación, permitiendo que el código:

- Responda de manera diferente según las diferentes entradas
- Cree aplicaciones dinámicas e interactivas
- Modele procesos de decisión del mundo real
- Maneje errores y casos excepcionales
- Implemente lógica de negocios compleja y dinámica.

Dominar estas estructuras es un paso crucial en la programación.