🤔 ¿Qué son las estructuras de decisión?

Las estructuras de decisión nos permiten ejecutar diferentes bloques de código dependiendo de si una condición es verdadera o falsa. En Python, estas estructuras nos ayudan a crear programas que toman decisiones basadas en datos.

⌨️ Operadores lógicos básicos

Python utiliza operadores lógicos para realizar comparaciones:

Operador	Significado	Ejemplo
`==`	¿Es igual que?	`a == b`
`<`	¿Es menor que?	`a < b`
`>`	¿Es mayor que?	`a > b`
`<=`	¿Es menor o igual que?	`a <= b`
`>=`	¿Es mayor o igual que?	`a >= b`
`!=`	¿Es diferente de?	`a != b`

🔀 Estructura `if`

La estructura `if` es la más básica y permite ejecutar un bloque de código solo si una condición es verdadera:

```
luzPrendida = False
if luzPrendida == False:
    print("La luz está apagada")
```

> **Nota:** En Python, es importante la indentación (espacios al inicio de línea) para definir los bloques de código. Todos los comandos que tengan la misma indentación después de un `if` pertenecen al mismo bloque. Te recomiendo útilizar la tecla `Tab` para mantener una indentación consistente.

🔄 Estructura `if-else`

Podemos usar `else` para ejecutar un bloque de código cuando la condición del `if` es falsa. Podemos mirar esta estructura de 2 posibilidades: si la condición es verdadera, se ejecuta un bloque; si es falsa, se ejecuta otro.

```
if a == b:
    print("Son iguales")
else:
    print("No son iguales")
```

🧩 Estructura `if-elif-else`

Para evaluar múltiples condiciones en secuencia, usamos `elif` (abreviatura de "else if"). Esto nos permite verificar varias condiciones y ejecutar el bloque correspondiente a la primera condición verdadera.

```
opcion = input("Ingresa una opción (1,2,3): ")

if opcion == "1":
    print("Presionaste la opción 1")
elif opcion == "2":
    print("Presionaste la opción 2")
elif opcion == "3":
    print("Presionaste la opción 3")
else:
    print("Opción inválida 😠😠")
```

> **Importante:** Python evalúa las condiciones en orden y ejecuta el bloque de la primera condición verdadera. Una vez que encuentra una condición verdadera, ignora las demás.

🔗 Operadores lógicos compuestos

Muchas veces necesitamos combinar varias condiciones para tomar decisiones más complejas. Python permite combinar condiciones lógicas usando operadores como `and`, `or` y `not`.

Python permite combinar condiciones lógicas usando operadores:

Operador `and`

Requiere que TODAS las condiciones sean verdaderas:

```
if a == b and a < c:
    print("A es igual a B y menor que C")
```

Operador `or`

Requiere que AL MENOS UNA condición sea verdadera:

```
if a == b or a < c:
    print("A es igual a B o menor que C")
```

Operador `not`

Invierte el valor de la condición:

```
if not a == b:
    print("A NO es igual a B")
```

> **Consejo:** Puedes usar paréntesis para agrupar condiciones y controlar el orden de evaluación, como en matemáticas.

💡 Ejercicios prácticos

Ejercicio 1: Verificador de número par o impar

Escribe un programa que determine si un número es par o impar.

```
numero = int(input("Ingresa un número: "))
if numero % 2 == 0:
    print(f"{numero} es un número par")
else:
    print(f"{numero} es un número impar")
```

Esto funciona ya que gracias al operador `%` (módulo), podemos determinar si un número es divisible por 2. Si el resultado es 0, el número es par; de lo contrario, es impar.

Ejercicio 2: Calculadora de IMC (Índice de Masa Corporal)

Crea un programa que calcule el IMC y clasifique según las categorías estándar.

```
peso = float(input("Ingresa tu peso en kg: "))
altura = float(input("Ingresa tu altura en metros: "))

imc = peso / (altura ** 2)
print(f"Tu IMC es: {imc:.2f}")

if imc < 18.5:
    print("Categoría: Bajo peso")
elif imc < 25:
    print("Categoría: Peso normal")
elif imc < 30:
    print("Categoría: Sobrepeso")
else:
    print("Categoría: Obesidad")
```

> **Nota**: En python puedes ocupar :.2f para limitar la cantidad de decimales que se muestran en el resultado del IMC. En este caso, estamos mostrando 2 decimales. Si quisieras mostrar 3 decimales, podrías usar :.3f, o a su vez si quisieras mostrar 0 decimales, podrías usar :.0f.

Gracias a la fórmula del IMC, podemos clasificar el estado nutricional de una persona según su peso y altura. Las categorías son: bajo peso, peso normal, sobrepeso y obesidad. Y al utilizar `elif`, podemos evaluar múltiples rangos de IMC de manera clara y ordenada.

Ejercicio 3: Verificador de año bisiesto

Desarrolla un programa que determine si un año es bisiesto.

```python
año = int(input("Ingresa un año: "))

if (año % 4 == 0 and año % 100 != 0) or (año % 400 == 0):
    print(f"{año} es un año bisiesto")
else:
    print(f"{año} no es un año bisiesto")
```

> **Explicación:** Un año es bisiesto si es divisible por 4 y no por 100, o si es divisible por 400.

🔄 Conclusión

Las estructuras de decisión son fundamentales en programación ya que permiten que nuestros programas tomen diferentes caminos según las condiciones que especificamos. En Python, estas estructuras son muy legibles y flexibles, lo que facilita la creación de lógica compleja de manera clara.

A coninuación, puedes seguir aprendiendo sobre las estructuras repetitivas, que te permitirán ejecutar bloques de código varias veces según ciertas condiciones.

Continuar ➡️