Notas de Conferencia 02: Tomando Decisiones en el Código

1. El Flujo de Control: De la Secuencia a la Decisión

Hasta ahora, nuestros programas han sido como una receta de cocina muy simple: una lista de instrucciones que se ejecutan una tras otra, de arriba hacia abajo, sin excepción. A esto lo llamamos **flujo secuencial**.

Sin embargo, el mundo real no es tan lineal. Constantemente tomamos decisiones basadas en ciertas condiciones: "Si está lloviendo, tomo el paraguas. Si no, tomo las gafas de sol". Para que nuestros programas sean realmente útiles y puedan modelar situaciones complejas, necesitan poder hacer lo mismo.

La solución a este problema son las **estructuras de control condicionales**. En Python, las herramientas principales para esto son las sentencias `if`, `elif` y `else`.

---

2. La Sentencia `if`: El Guardián del Código

La sentencia `if` es la estructura de decisión más fundamental. Actúa como un guardián que protege un bloque de código. Este bloque de código **solo se ejecutará si la condición que custodia el `if` es verdadera (`True`)**.

Anatomía de un `if`

Toda sentencia `if` se compone de tres partes clave:

1. La palabra clave `if`.
2. Una **condición**: Una pieza de código que, al ser evaluada, produce un resultado booleano (`True` o `False`).
3. Un **bloque de código indentado**: Una o más líneas de código que están desplazadas hacia la derecha. Este es el código que se ejecutará si la condición es `True`. El uso de los dos puntos (`:`) al final de la línea del `if` es fundamental, ya que le indica a Python: "Aquí comienza el bloque de código que depende de esta condición".

La **indentación** no es una sugerencia estilística en Python; es una regla sintáctica obligatoria que define qué código pertenece a qué estructura.

```python
edad = int(input())

if edad >= 18:
    # Este es el bloque de código.
    # Solo se ejecuta si la variable 'edad' es 18 o más.
    print("Eres mayor de edad.")
```

---

3. ¿Qué es una Condición? Expresiones y Booleanos

La pregunta clave es: ¿qué podemos poner como condición en un `if`? La respuesta es: **cualquier expresión que pueda ser interpretada como `True` o `False`**.

Expresiones vs. Instrucciones

- Una **expresión** es cualquier fragmento de código que **produce un valor**. Por ejemplo: `2 + 2` (produce el valor `4`), `edad > 18` (produce `True` o `False`), o simplemente `True`.
- Una **instrucción** es una acción que altera el estado del programa, como asignar una variable (`x = 5`) o imprimir algo en pantalla (`print("Hola")`).

La condición de un `if` **siempre debe ser una expresión**.

Expresiones Booleanas

Así como los operadores aritméticos (`+`, `-`, `*`) trabajan con números, los **operadores booleanos** trabajan con valores de verdad. Se dividen en dos categorías principales:

1. **Operadores de Comparación**: Toman dos valores y devuelven un booleano. Son el pan de cada día en las condiciones.
    - `==` : Igual a
    - `!=` : Distinto de
    - `>` : Mayor que
    - `<` : Menor que
    - `>=` : Mayor o igual que
    - `<=` : Menor o igual que
    - **¡Cuidado!** Un error común de principiante es usar `=` (asignación) en lugar de `==` (comparación) en un `if`.

2. **Operadores Lógicos**: Toman valores booleanos y devuelven un nuevo booleano. Sirven para combinar condiciones.

    - `and`: Devuelve `True` solo si **ambas** condiciones son `True`.
    - `or`: Devuelve `True` si **al menos una** de las condiciones es `True`.
    - `not`: Invierte el valor booleano (`not True` es `False`).

```
edad = int(input())

if edad >= 18 and (not edad >= 60):
# o if edad >= 18 and edad < 60:
# o if not ( edad < 18 or edad >= 60):
# Hay múltiples formas de escribir una condición
    print("Eres un adulto")
```

---

4. El Plan B: La Sentencia `else`

El `if` nos permite ejecutar código si una condición es verdadera. Pero, ¿qué pasa si es falsa? La sentencia `else` nos proporciona un "plan B".

Un bloque `else` se asocia a un `if` y se ejecuta **únicamente cuando la condición del `if` es `False`**. Un `else` nunca tiene su propia condición; simplemente recoge todos los casos que no cumplieron la condición del `if`.

```
if condicion:
    # Bloque A: Se ejecuta si la condición es True
else:
    # Bloque B: Se ejecuta si la condición es False
```

```
if edad >= 18 and (not edad >= 60):
    print("Eres un adulto")
else:
    print("Eres muy joven o muy mayor")
```

¿Y si hay más de dos caminos? El problema del anidamiento

Cuando tenemos más de dos posibilidades, una primera idea podría ser anidar `if/else`. Por ejemplo, si queremos diferenciar precios de entrada según la edad. En la entrada de un museo desean dejar entrar a los niños gratis, a los mayores le cobran la mitad y a los adultos le cobran el precio total de la entrada: 200 CUP:

```
# Establecemos una variable precio y la establecemos en 0
# por defecto
precio = 0

if edad <=12:
    print("Es un niño")
    # Nótese que aquí no establecemos el precio, nos quedamos con
    # el 0 como valor por defecto
else:
    if edad >=60:
        print("Es una persona mayor")
        price = 100
    else:
        print("Es un adulto")
        price = 200
```

Aunque esto funciona, tiene dos grandes problemas:

1. **Es feo y difícil de leer**: La indentación excesiva crea una estructura llamada "cabeza de flecha" que complica la lectura del flujo lógico.
2. **Es propenso a errores**: Es fácil perderse en los niveles de anidamiento y cometer errores lógicos.

---

5. Múltiples Caminos: La Solución Elegante con `elif`

Para resolver el problema del anidamiento, Python nos ofrece `elif`, que es una contracción de "else if". `elif` nos permite encadenar múltiples condiciones de forma clara y legible.

La estructura completa funciona así:

1. Python evalúa la condición del `if`. Si es `True`, ejecuta su bloque y **salta el resto de la estructura** (`elif`s y `else`).

2. Si la condición del `if` es `False`, pasa a evaluar la condición del **primer `elif`**. Si es `True`, ejecuta su bloque y salta el resto.
3. Este proceso se repite para cada `elif` en orden.
4. Si ninguna de las condiciones anteriores (`if` y todos los `elif`) fue `True`, se ejecuta el bloque del `else` (si existe).

La clave es que **solo uno de los bloques de toda la cadena se ejecutará como máximo**: el primero cuya condición sea `True`.

```
if condicion_A:
    # Bloque A
elif condicion_B:
    # Bloque B
elif condicion_C:
    # Bloque C
else:
    # Bloque D (por defecto)
```

Por ejemplo, modificando el ejemplo anterior:

```
precio = 0

if edad <=12:
    print("Es un niño")
elif edad >=60:
    # Se reduce la indentación
    print("Es una persona mayor")
    price = 100
else:
    print("Es un adulto")
    price = 200
```

---

6. Resumen de la Sesión

- **Estructuras de control**: Son herramientas que nos permiten alterar el flujo secuencial de un programa. Las condicionales (`if`, `elif`, `else`) nos permiten tomar decisiones.
- **Bloques de código**: En Python, los bloques de código se definen mediante la **indentación**. Es una regla sintáctica, no una opción.

- **Expresiones vs. Instrucciones**: Las expresiones producen un valor, mientras que las instrucciones realizan una acción. Las condiciones de un `if` deben ser expresiones.
- **Expresiones Booleanas**: Usamos operadores de **comparación** (`==`, `!=`, `>`) y **lógicos** (`and`, `or`, `not`) para construir condiciones complejas que evalúan a `True` o `False`.
- **Flujo condicional completo**:
  - `if`: Para la condición principal.
  - `elif`: Para condiciones alternativas, si la primera falla.
  - `else`: Para el caso por defecto, si ninguna condición anterior se cumple.