📝 Apuntes de Programación: Control de Flujo (While y Match/Case)

El control de flujo es esencial para determinar el orden en que se ejecutan las instrucciones de un programa. Tu código combina un bucle repetitivo (`while`) con una estructura de decisión moderna (`match/case`).

1. El Bucle `while` (Repetición Condicional)

El bucle `while` se utiliza para repetir un bloque de código **mientras** una condición específica sea verdadera. Es ideal para situaciones donde no sabes de antemano cuántas veces se debe repetir el bloque (como un menú o una entrada de usuario).

A. Estructura

```
numero = 0
while(numero != 5): # Condición de permanencia: el bucle se ejecuta MIENTRAS 'numero' s
    # El código dentro del bucle se ejecuta repetidamente
    numero = int(input("Introduce un nuemro del 1 al 4"))
    # ...
print("FIN") # El programa solo llega aquí cuando la condición 'numero != 5' es Falsa
```

B. Elementos Clave

1. **Condición de Salida:** Debe haber una instrucción dentro del bucle que eventualmente haga que la condición del `while` sea falsa, previniendo un bucle infinito. En tu ejemplo, es cuando el usuario introduce el número `5` .

2. La Sentencia `match/case` (El Python Switch)

La sentencia `match/case` (Coincidencia de Patrones) se introdujo en Python 3.10 como una alternativa más limpia y potente a las largas cadenas de `if/elif/else` cuando se evalúa una única expresión.

A. Estructura y Componentes

```
match numero: # Se utiliza el valor de la variable 'numero' para la coincidencia
    case 2: # Patrón 1: Coincidencia exacta con el valor 2
        print("El numero es el 2")
    case 3 | 4: # Patrón 2: Coincidencia con el valor 3 O con el valor 4 (uso del operc
        print("El numero es el 3 y el 4")
    case 5: # Patrón 3: Coincidencia exacta con 5
        print("Salir")
    case _: # Patrón 4 (Wildcard): Coincidencia por defecto (funciona como el 'default'
        print("No es ni el 2 ni el 3")
```

B. Patrones Clave (Para Examen)

Patrón (`case`)	Descripción	Uso Común
Literales (`case 2:`)	Coincide con valores exactos (números, cadenas, booleanos).	Menús de selección, estados específicos.
**`	`(OR)** (case 3	4:`)
Wildcard (`case _:`)	Captura cualquier valor que no haya coincidido con los patrones anteriores. **Siempre debe ser el último**.	Manejo de entradas inválidas o por defecto.

3. Consejos para Mejorar la Programación con `match/case`

El uso de `match/case` en lugar de `if/elif/else` mejora significativamente la legibilidad y la estructura del código, especialmente cuando las decisiones se basan en la forma de los datos (no solo en su valor).

🎯 Principio 1: Mejor Legibilidad que `if/elif`

- Para las estructuras de menú o decisión basadas en valores literales (como números), `match/case` es mucho más claro que su equivalente en `if/elif` . Es la forma moderna de implementar la lógica *switch* en Python.

🎯 Principio 2: Coincidencia de Tipo y Estructura

- `match/case` no solo coincide valores, sino también **tipos** y **estructuras de datos**.
- **Ejemplo Avanzado (Coincidencia de Tipo):**

```
def procesar(datos):
    match datos:
        case int():    # Si 'datos' es un entero
            print("Es un número")
        case str():    # Si 'datos' es una cadena
            print("Es un texto")
        case _:
            print("Tipo desconocido")
```

🎯 Principio 3: Evitar el Bucle Infinito

- **Doble chequeo del `while` :** Asegúrate siempre de que la variable de control (`numero` en este caso) se actualice dentro del bucle. Un error aquí resultará en un programa que nunca termina. La entrada del usuario (`input`) dentro del `while` es una forma común y efectiva de actualizar esta variable.