1. Análisis del enunciado
¿Qué pide el enunciado?

Dado un número proporcionado por el usuario, verificar si ese número es múltiplo de cada uno de los siguientes: 2, 3, 5, 7, 9, 10 y 11. Para cada verificación, debe imprimirse un mensaje indicando si sí o no es múltiplo.

¿Qué tipo de datos y operadores utilicé?

Tipo de dato: int (trabajamos con enteros para múltiplos).

Operador principal: el operador módulo %, que devuelve el resto de la división entera.

¿Qué validaciones implementé o podría haber hecho?

Verificar que la entrada sea un entero válido (try/except ValueError).

(Opcional) Comprobar que sea mayor que cero, si se desea limitar la definición de múltiplo a positivos.

2. Código Python

```
# --- Desafío 4: Verificar múltiplos ---
# 1. Leer número desde el usuario
try:
    numero = int(input("Ingrese un número entero: "))
except ValueError:
    raise ValueError("Debes ingresar un número entero válido.")
# (Validación opcional)
if numero <= 0:
    print("Nota: definiendo múltiplos solo para enteros positivos.")
# 2. Lista de divisores a verificar
divisores = [2, 3, 5, 7, 9, 10, 11]
# 3. Para cada divisor, comprobamos con el operador módulo
for d in divisores:
    if numero % d == 0:
        print(f"{numero} **SÍ** es múltiplo de {d}.")
    else:
        print(f"{numero} **NO** es múltiplo de {d}.")
```

3. Explicación del código
Lectura y validación

Usamos input() y int() dentro de un bloque try/except para asegurarnos de que el usuario ingresó un entero.

(Opcional) Se advierte si el número no es positivo.

Lista de divisores

Creamos una lista divisores = [2, 3, 5, 7, 9, 10, 11] para iterar de forma dinámica y evitar repetir código.

Verificación con módulo

El operador numero % d devuelve el resto de dividir numero entre d.

Si el resto es cero, entonces numero es múltiplo de d; en caso contrario, no lo es.

Mostramos un mensaje con f-strings, resaltando "SÍ" o "NO".

4. Validaciones adicionales sugeridas

Permitir que el usuario repita la verificación en bucle while, hasta que ingrese un comando de salida (p. ej. "salir").

Añadir manejo de números negativos, definiendo si consideras múltiplos de valores absolutos.

Registrar los resultados en un archivo de texto o CSV para consulta posterior.

5. Webgrafía

Documentación oficial de Python sobre operadores aritméticos:

https://docs.python.org/3/reference/expressions.html#binary-arithmetic-operations

Explicación del operador módulo (%):

https://es.wikipedia.org/wiki/Operador_módulo