Texto

El contenido generado por IA puede ser incorrecto.

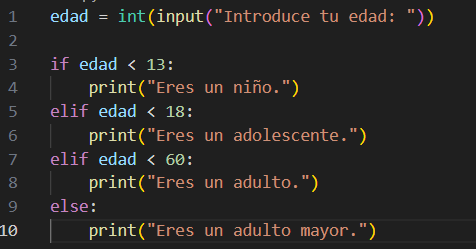
**ACTIVIDAD IA 802**

**Actividad 1: Programación en Python**

A continuación, se presentan ejercicios para practicar estructuras de control (if, for, while, switch, funciones). Cada ejercicio incluye el código y su explicación para que puedas entender el **por qué** y **para qué** de su uso. Tu tarea es **replicar, modificar y comentar** cada fragmento de código.

**Estructura condicional if**

**Ejercicio 1.** Verifica si un número es positivo, negativo o cero.



**¿Por qué if?**  
La estructura if-elif-else permite clasificar casos mutuamente excluyentes. Aquí usamos rangos para decidir en qué etapa de la vida se encuentra una persona.

**Preguntas de lógica**:

1. ¿Qué sucedería si no incluyéramos el else?

En caso de no incluirlo no tendríamos una opción por defecto entonces si alguien tuviera mas de 60 años el código no le daría ninguna salida

1. ¿Cómo cambiarías el código para incluir una categoría “adulto joven” de 18 a 29 años?

Agregaría un and al código esto para que si esta en ese rango le imprimiera un valor, también modifique el código para tener and en cada uno de los intervalos para que sea más claro

# Solicita la edad al usuario y la convierte a número entero

edad = int(input("Introduce tu edad: "))

# Si la edad es menor que 13

if edad < 13:

print("Eres un niño.")

# Si la edad está entre 13 y 17 (inclusive)

elif edad >= 13 and edad < 18:

print("Eres un adolescente.")

# Si la edad está entre 18 y 29 (inclusive)

elif edad >= 18 and edad < 30:

print("Eres un adulto joven.")

# Si la edad está entre 30 y 59 (inclusive)

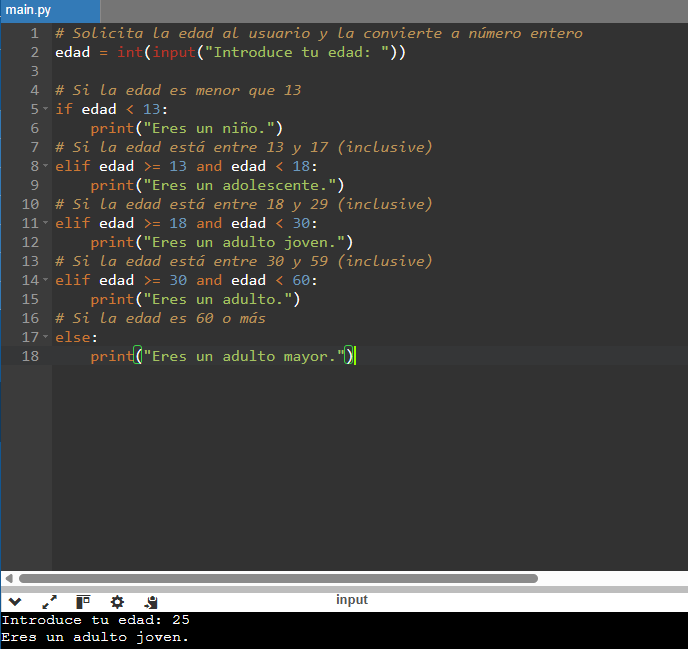
elif edad >= 30 and edad < 60:

print("Eres un adulto.")

# Si la edad es 60 o más

else:

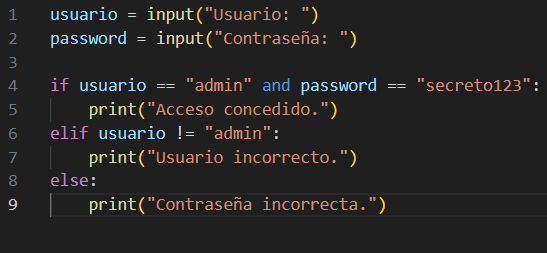
print("Eres un adulto mayor.")



1. ¿Qué pasaría si alguien introduce una edad negativa?

En caso de ser negativa el programa le mostraría que es un adulto mayor, por lo tanto el else debería ser una opción para edades no validad

**Ejercicio 2: Validación de contraseña con advertencia**



**¿Por qué if?**  
Permite validar condiciones combinadas con operadores lógicos (and, or). Aquí se usa para diferenciar fallas de autenticación.

**Preguntas de lógica**:

1. ¿Qué operadores lógicos se usan en el código?

Usa el operador lógico “and” y el operador “!=”

1. ¿Por qué es importante validar el usuario antes que la contraseña?

Por que en caso de que el usuario exista no tiene caso que preguntemos la contraseña ya que solo sería desperdiciar memoria en nuestro programa

1. ¿Cómo implementarías un intento fallido limitado a 3 veces?

Crearía un while True para preguntar la contraseña o usuario hasta que se ingresen de manera correcta pero en caso de tener 3 intentos fallidos el programa finalizaría informándoselo al usuario

# intentos

intentos = 0

# iniciamos el bucle

while True:

# Solicita el nombre de usuario al usuario

usuario = input("Usuario: ")

# Solicita la contraseña al usuario

password = input("Contraseña: ")

# Verifica si el usuario y la contraseña son correctos

if usuario == "admin" and password == "secreto123":

print("Acceso concedido.")

break

# Verifica si el usuario es incorrecto

elif usuario != "admin":

print("Usuario incorrecto.")

# Si el usuario es correcto pero la contraseña es incorrecta

else:

print("Contraseña incorrecta.")

# Incrementa el contador de intentos

intentos += 1

# Verifica si se han agotado los intentos

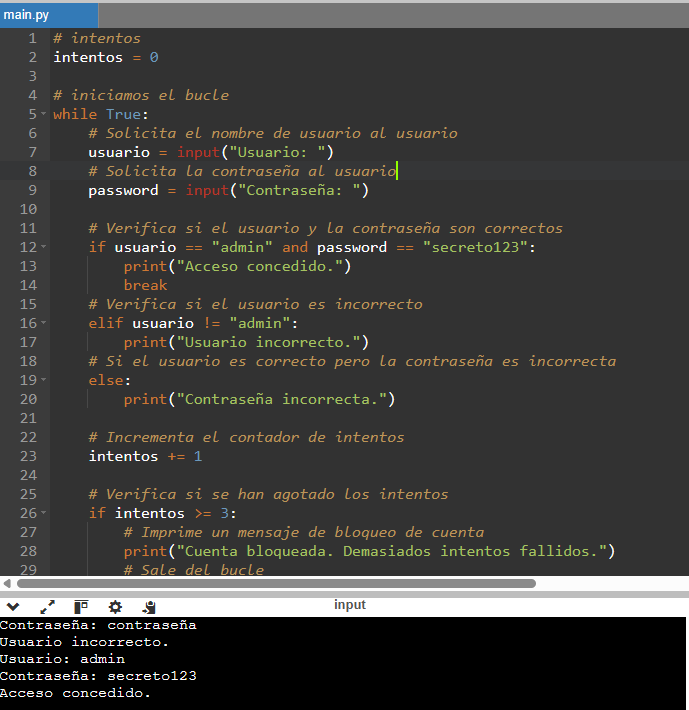
if intentos >= 3:

# Imprime un mensaje de bloqueo de cuenta

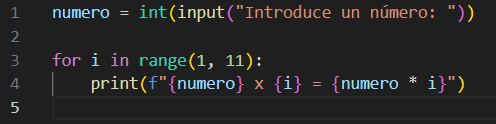
print("Cuenta bloqueada. Demasiados intentos fallidos.")

# Sale del bucle

break



**Ejercicio 3: Mostrar tabla de multiplicar de un número**



**¿Por qué for?**  
El for se usa cuando conoces de antemano el número de repeticiones. Aquí se usa para imprimir del 1 al 10.

**Preguntas de lógica**:

1. ¿Qué hace range(1, 11) exactamente?

Hace que la variable i recorra los valores desde 1 hasta 10

1. ¿Cómo podrías modificar el código para imprimir la tabla al revés?

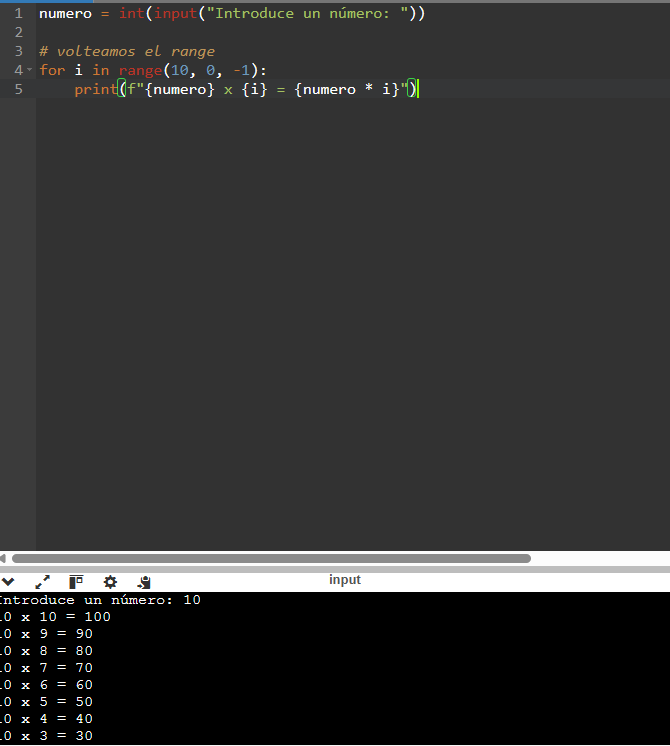
Modificaría los valores que se le dan al rango para darlos al revés y decir que los quiero de -1 en -1

numero = int(input("Introduce un número: "))

# volteamos el range

for i in range(10, 0, -1):

print(f"{numero} x {i} = {numero \* i}")



1. ¿Cómo harías que imprima todas las tablas del 1 al 10?

Simplemente en vez de preguntar el numero metería todos los números en un for

# Itera a través de los números del 1 al 10 (tablas de multiplicar)

for numero in range(1, 11):

# Imprime el encabezado de la tabla actual

print(f"Tabla de multiplicar del {numero}:")

# Itera a través de los números del 1 al 10 (multiplicadores)

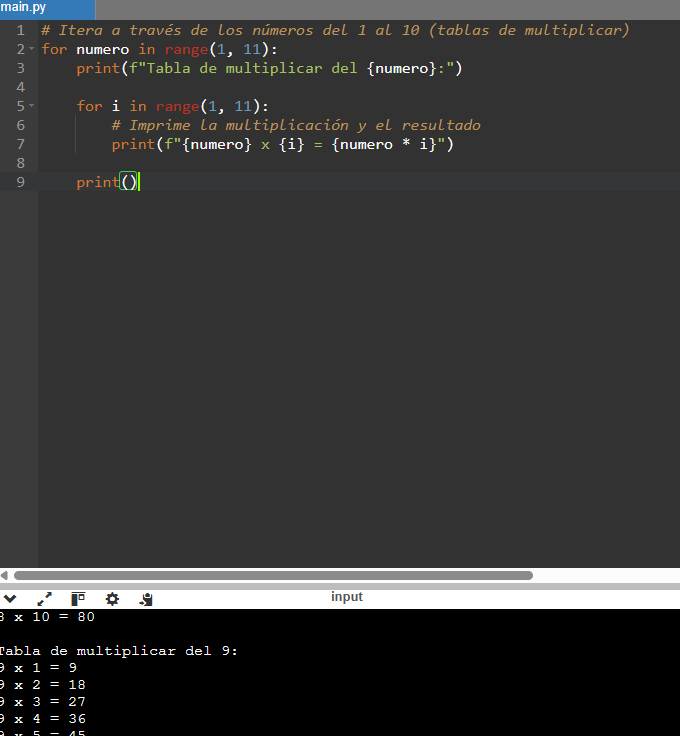
for i in range(1, 11):

# Imprime la multiplicación y el resultado

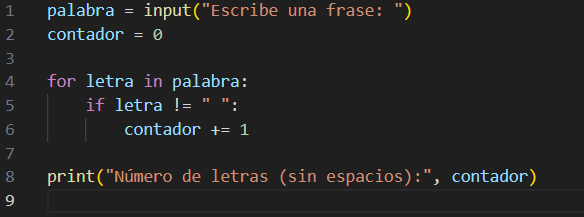
print(f"{numero} x {i} = {numero \* i}")

# Imprime una línea en blanco para separar las tablas

print()



**Ejercicio 4: Contar letras en una palabra (sin espacios)**



**¿Por qué for con if?**  
El for recorre carácter por carácter, y el if filtra los espacios. Esto combina lógica de control con análisis de datos.

**Preguntas de lógica**:

1. ¿Qué otros caracteres podrías querer ignorar además de espacios?

Las comas, así como los puntos y signos de interrogación y admiración

1. ¿Cómo podrías contar solo vocales?

Crearía una cadena con todas las vocales después iteraría y en caso de que la letra este en mi cadena aumentaría un valor

# Solicita al usuario que escriba una frase

palabra = input("Escribe una frase: ")

# Inicializa el contador de vocales

contador\_vocales = 0

# vocales

vocales = "aeiouAEIOU"

# Itera a través de cada letra en la palabra

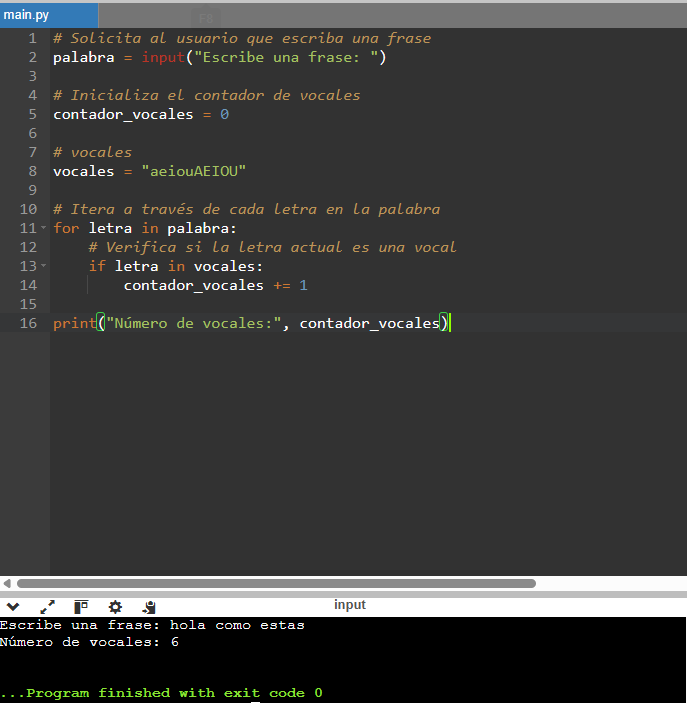
for letra in palabra:

# Verifica si la letra actual es una vocal

if letra in vocales:

contador\_vocales += 1

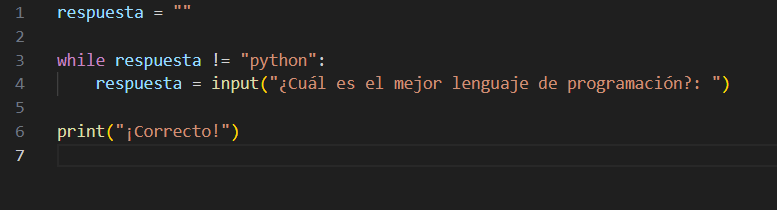
print("Número de vocales:", contador\_vocales)



1. ¿Qué pasa si no usas if?

En caso de no usar contaríamos todos los caracteres de la cadena que el usuarion ingrese

**Ejercicio 5: Validar entrada hasta que sea correcta**



**¿Por qué while?**  
Se utiliza cuando no sabes cuántas veces se va a repetir el ciclo. El programa sigue preguntando hasta que se cumpla la condición.

**Preguntas de lógica**:

1. ¿Qué pasaría si el usuario nunca escribe la palabra correcta?

En caso de nunca escribirla el bucle serio infinito

1. ¿Cómo agregarías un número máximo de intentos?

Mediante la creación de una variable y un if dentro del bucle , en caso de alcanzar los intentos máximos lo que haría seria que usaría un break

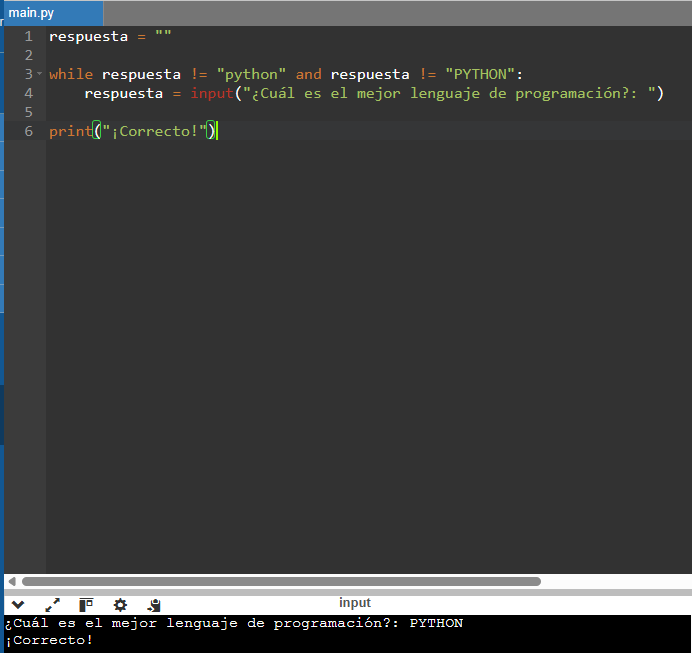
1. ¿Puedes cambiar el programa para que no importe si escriben “Python” con mayúscula?

respuesta = ""

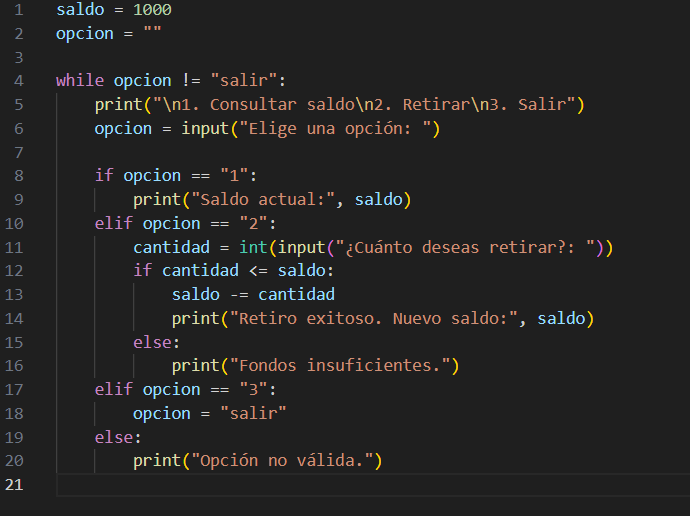
while respuesta != "python" and respuesta != "PYTHON":

respuesta = input("¿Cuál es el mejor lenguaje de programación?: ")

print("¡Correcto!")



**Ejercicio 6: Cajero automático simulado**



**¿Por qué match-case?**  
Permite manejar múltiples casos claramente, como un switch. Es más legible que varios if.

**Preguntas de lógica**:

1. ¿Qué ventaja tiene match sobre if en este caso?

Considero que ayudaría simplemente a mejorar la legibilidad del código ya que es análogo a if o varios if

1. ¿Qué pasa si el usuario pone otro símbolo?

Dado que tenemos un else lo que pasaría es que se le mostraría al usuario que su opción no es valida

1. ¿Qué deberías validar antes de dividir?

Validaría que la cantidad a retirar sea un entero positivo

saldo = 1000

opcion = ""

while opcion != "salir":

print("\n1. Consultar saldo\n2. Retirar\n3. Salir")

opcion = input("Elige una opción: ")

if opcion == "1":

print("Saldo actual:", saldo)

elif opcion == "2":

while True: # Bucle para validar la entrada

try:

cantidad = int(input("¿Cuánto deseas retirar?: "))

if cantidad > 0:

break # Salir del bucle si la cantidad es válida

else:

print("Por favor, introduce un número entero positivo.")

except ValueError:

print("Entrada inválida. Debes introducir un número entero.")

if cantidad <= saldo:

saldo -= cantidad

print("Retiro exitoso. Nuevo saldo:", saldo)

else:

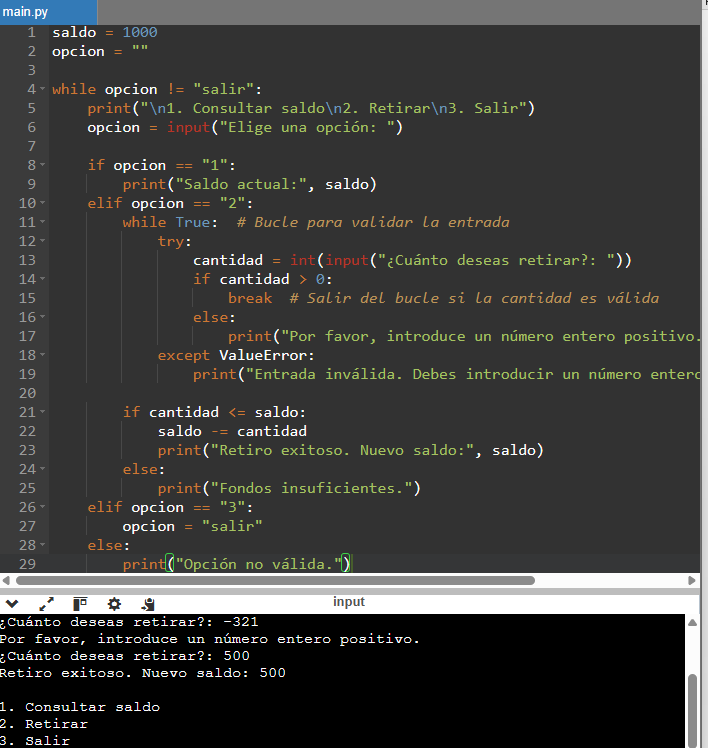
print("Fondos insuficientes.")

elif opcion == "3":

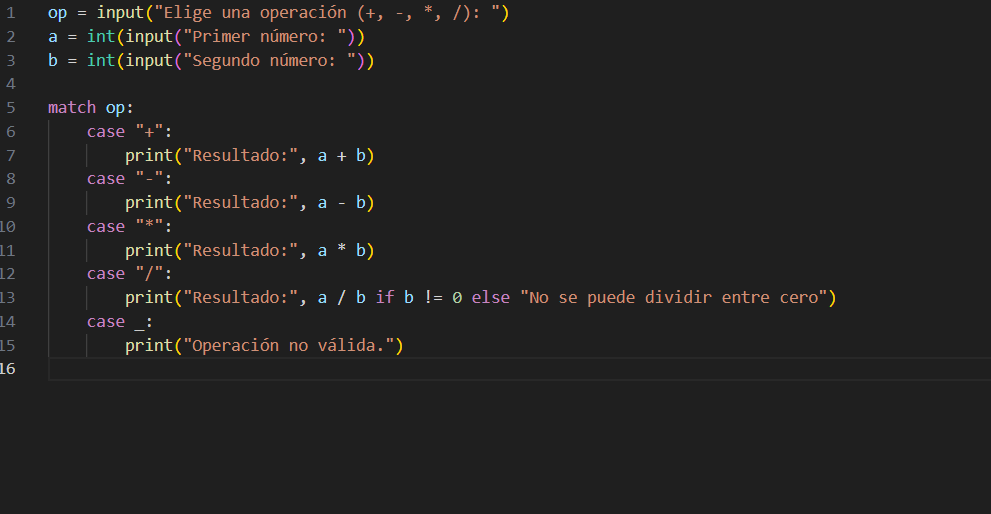
opcion = "salir"

else:

print("Opción no válida.")



**Ejercicio 7: Menú de operaciones matemáticas**



**¿Por qué match-case?**  
Permite manejar múltiples casos claramente, como un switch. Es más legible que varios if.

**Preguntas de lógica**:

1. ¿Qué ventaja tiene match sobre if en este caso?

Da una mejor legibilidad al código

1. ¿Qué pasa si el usuario pone otro símbolo?

Se le imprimiría que su opción elegida no es valida

1. ¿Qué deberías validar antes de dividir?

Validaría que el divisor sea diferente de 0, esto para no tener un numero infinito

# Solicita al usuario que elija una operación (+, -, \*, /)

op = input("Elige una operación (+, -, \*, /): ")

# Solicita al usuario que introduzca el primer número

a = int(input("Primer número: "))

# Solicita al usuario que introduzca el segundo número

b = int(input("Segundo número: "))

# Utiliza la estructura match para realizar la operación seleccionada

match op:

# Caso para la suma

case "+":

print("Resultado:", a + b)

# Caso para la resta

case "-":

print("Resultado:", a - b)

# Caso para la multiplicación

case "\*":

print("Resultado:", a \* b)

# Caso para la división

case "/":

# Valida que el divisor no sea cero para evitar la división infinita

if b != 0:

print("Resultado:", a / b)

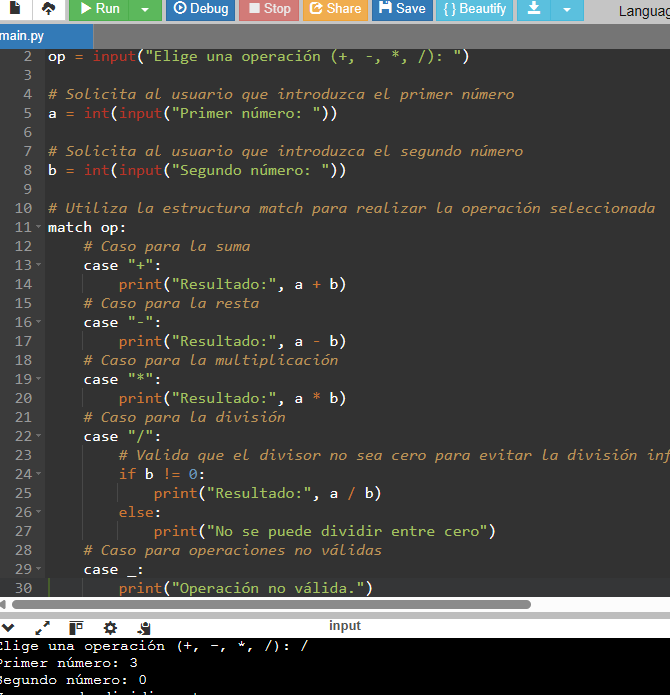
else:

print("No se puede dividir entre cero")

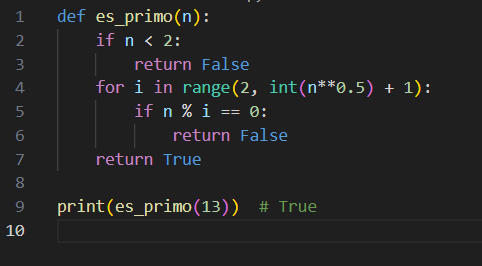
# Caso para operaciones no válidas

case \_:

print("Operación no válida.")



**Ejercicio 8: Función que devuelve si un número es primo**



**¿Por qué función?**  
Porque encapsulamos una tarea reutilizable. Y se usa la raíz cuadrada como optimización.

**Preguntas de lógica**:

1. ¿Qué hace exactamente el bucle for dentro de la función?

Se encarga de verificar que el número tenga algún divisor exacto desde 2 hasta la raíz de n

1. ¿Por qué se empieza desde 2 y no desde 1?

Dado que el 1 no es un divisor útil ya que todos los números pordiran ser divididos por el

1. ¿Qué pasaría si no usáramos return dentro del if?

Tendriamos errores de encontrar falsos números primos

def es\_primo(n):

if n < 2:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

pass # No se retorna False, solo se pasa al siguiente número

return True

print(es\_primo(13)) # True

