Práctica 2b:
Python y Visual Studio Code

¿Qué es Python?

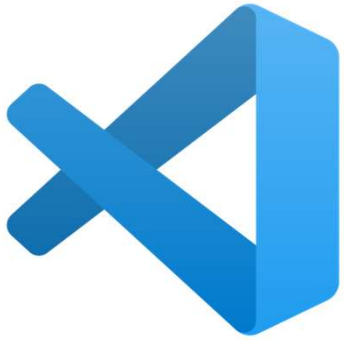
- Lenguaje de programación de alto nivel, fácil de leer y escribir.
- Muy usado en: desarrollo web, ciencia de datos, automatización, inteligencia artificial, etc.

¿Qué es Visual Studio Code?

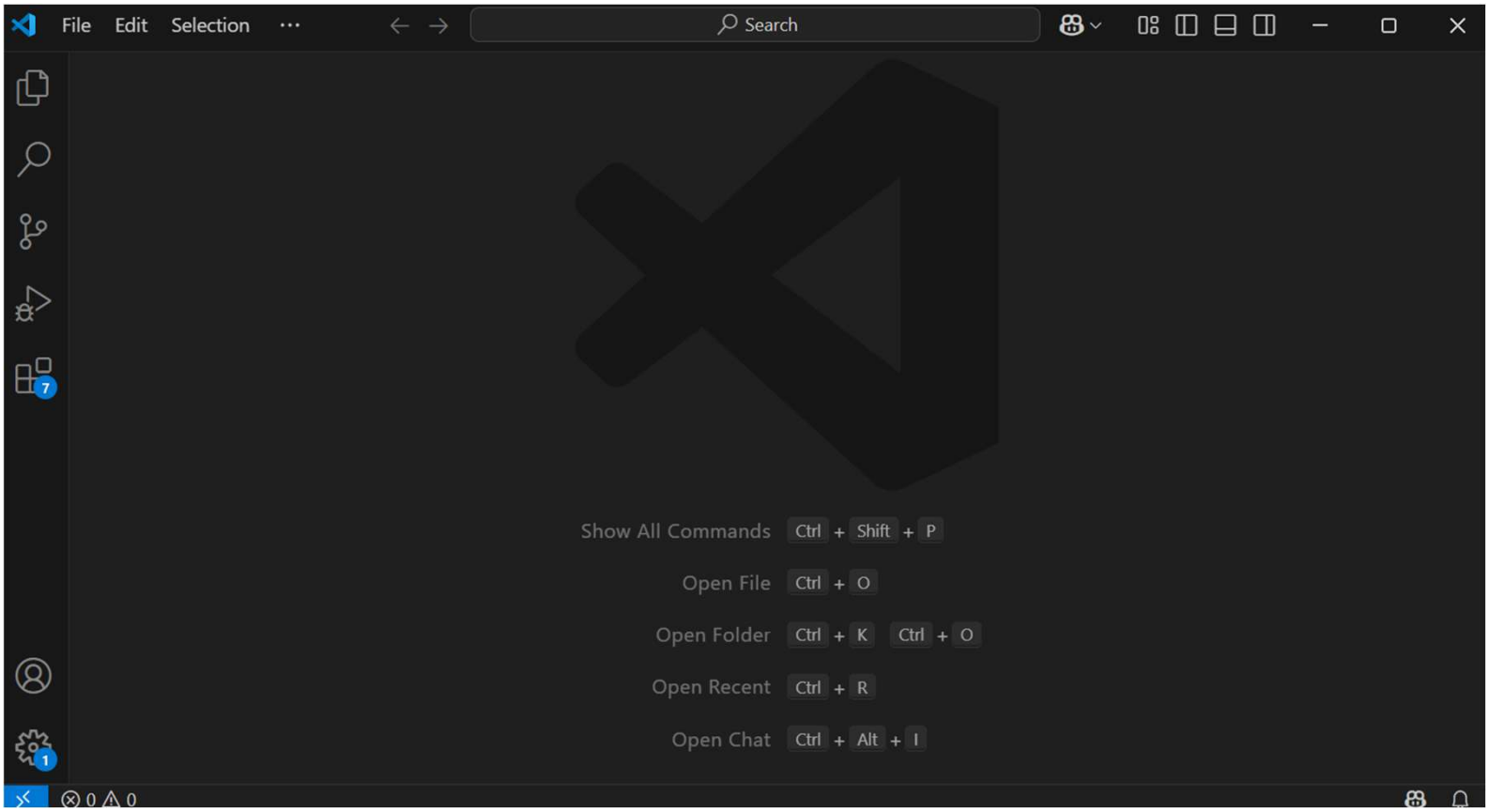
- Editor de código fuente gratuito de Microsoft.
- Soporta múltiples lenguajes (Python, JavaScript, C++, etc.)
- Integración con Git, terminal, depuración, y más extensiones.

Instalación

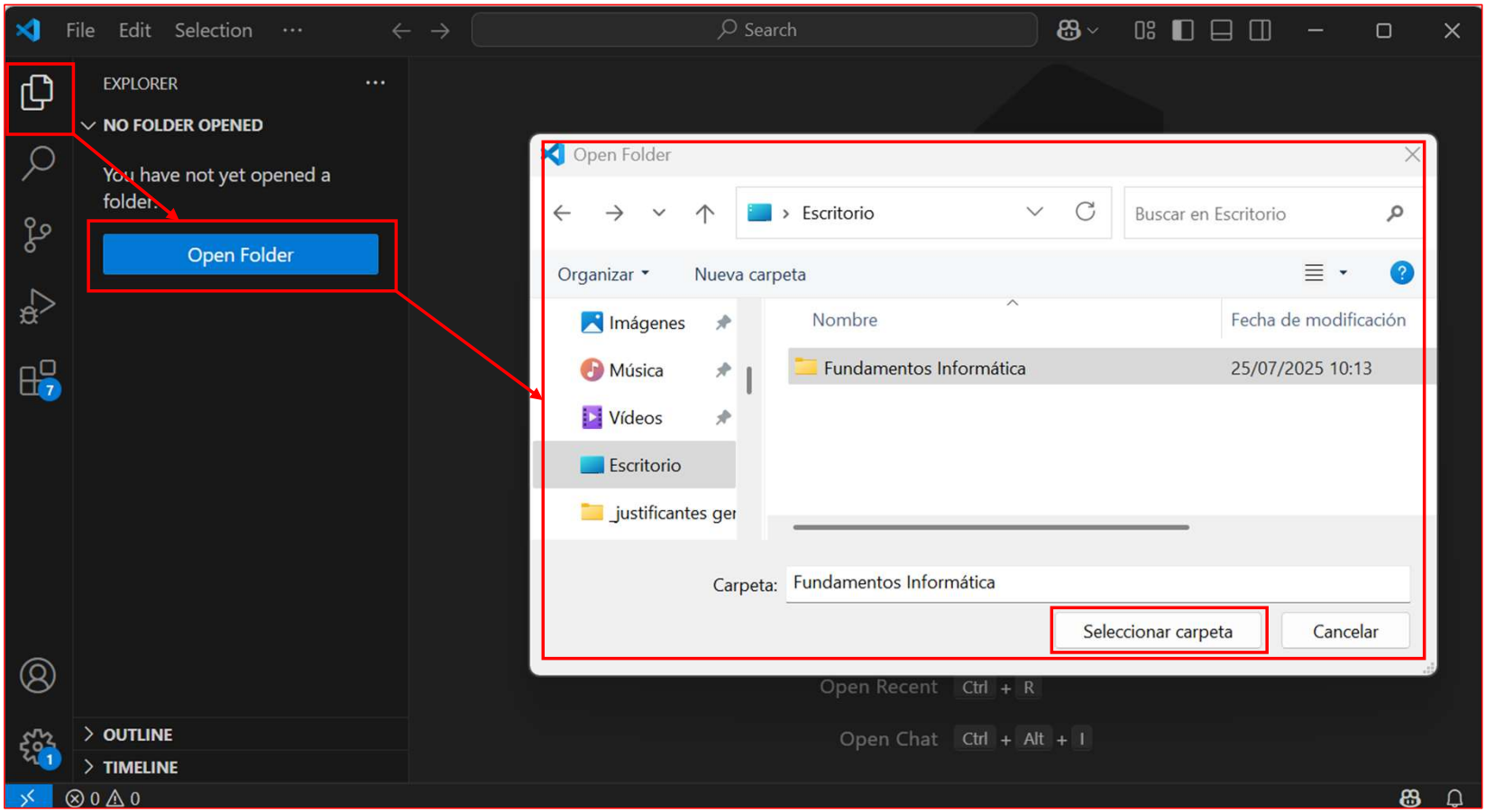
- Python
  - https://python.org
- Visual Studio Code
  - https://code.visualstudio.com/



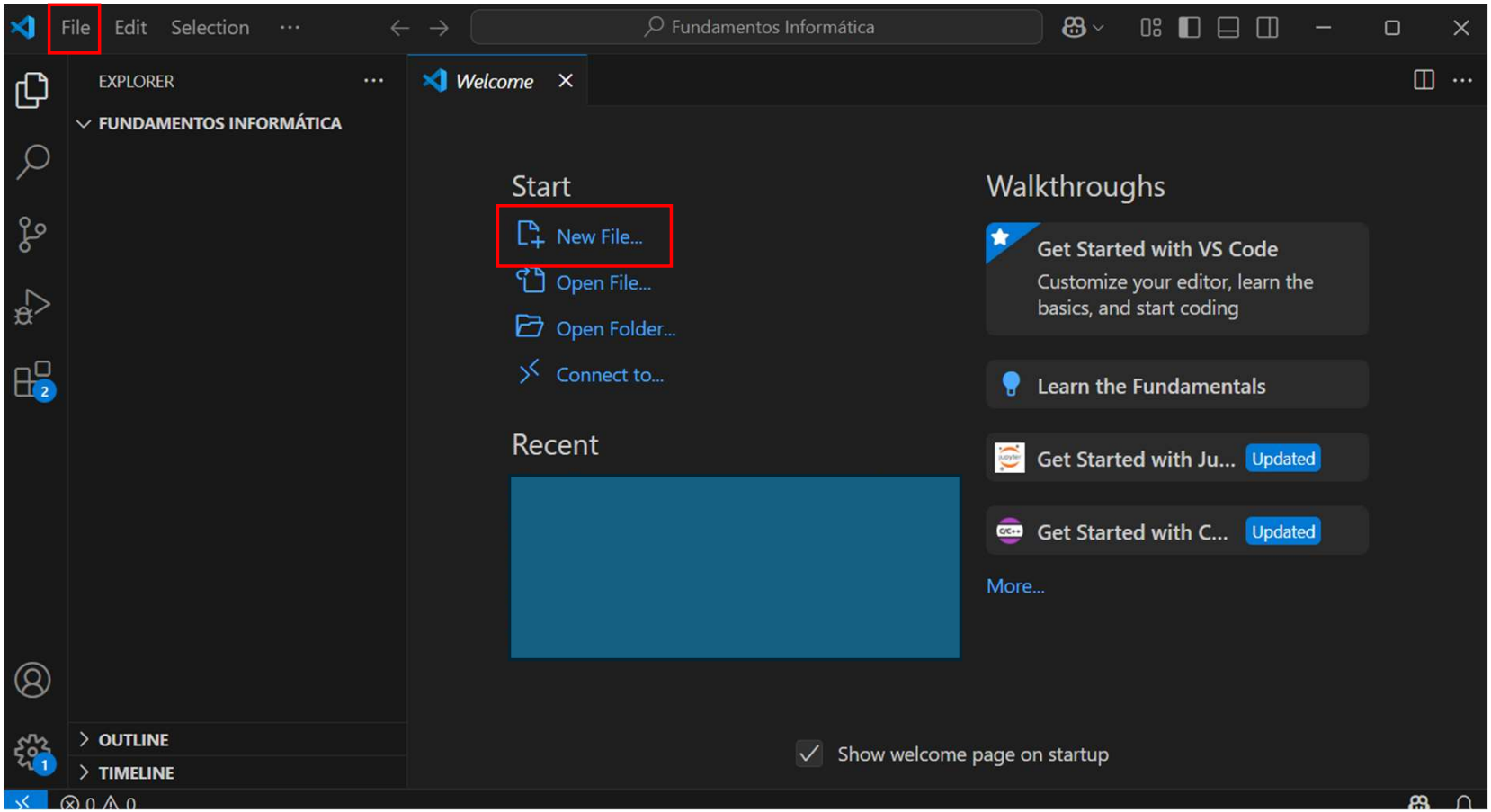
Introducción a Visual Studio Code



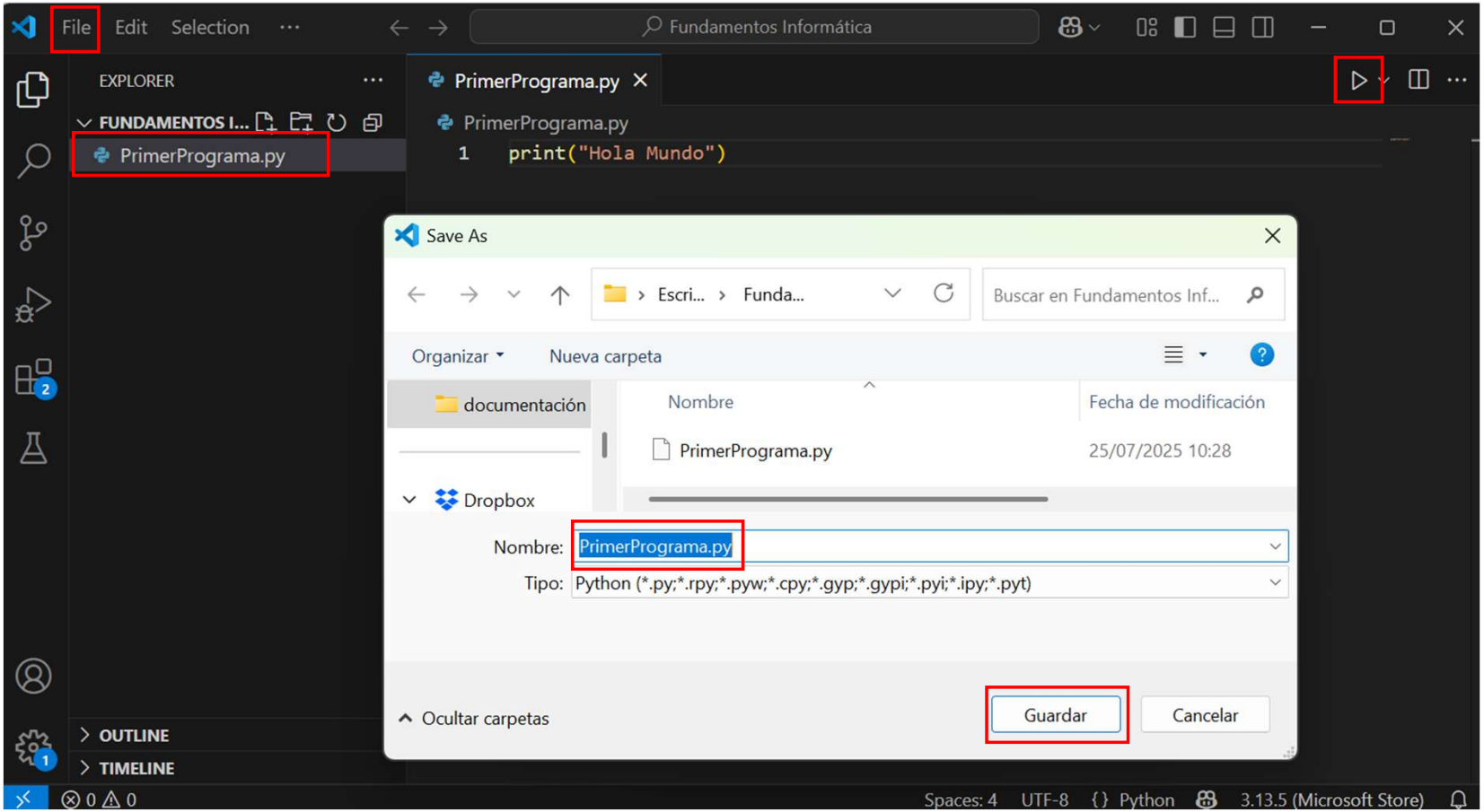
Seleccionar un directorio de trabajo



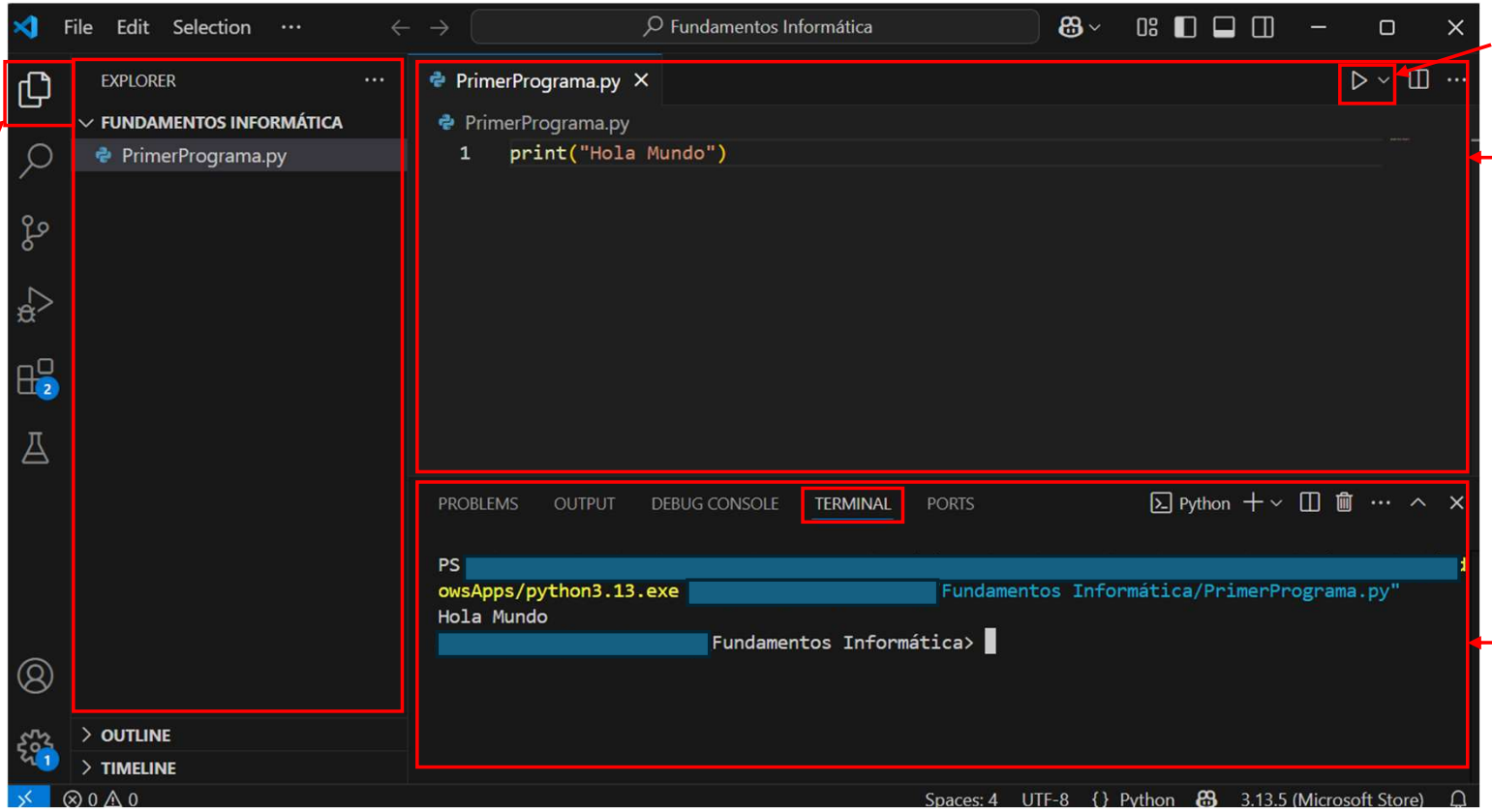
Crear un primer programa “Hola Mundo”



Hola Mundo



Hola Mundo



Navegador

Ejecutar el programa.

El editor

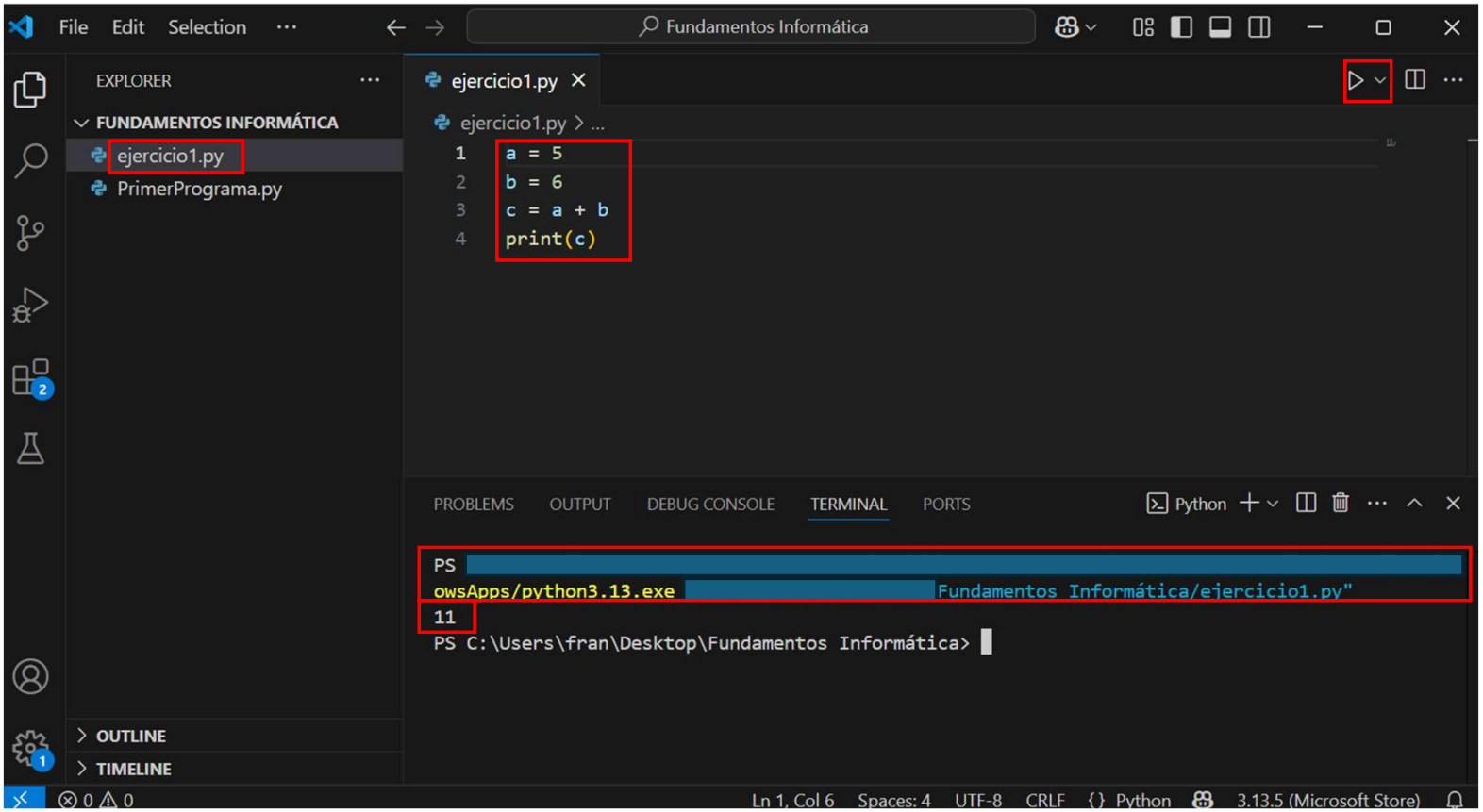
La consola, terminal, o intérprete

Primer ejercicio

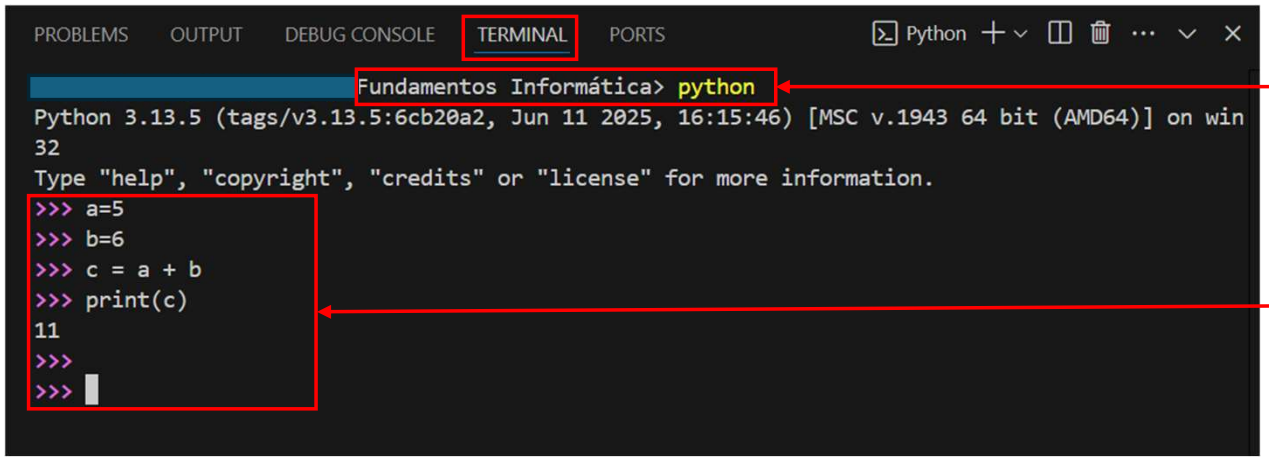
Realiza un programa que consista en asignar el valor 5 a la variable “a”, el valor 6 a la variable “b” y que la suma de ambas variables se guarde en la variable “c”. A continuación, muestra el resultado por consola.

Guarda el programa como “ejercicio1.py” y ejecútalo.

Primer ejercicio



La consola



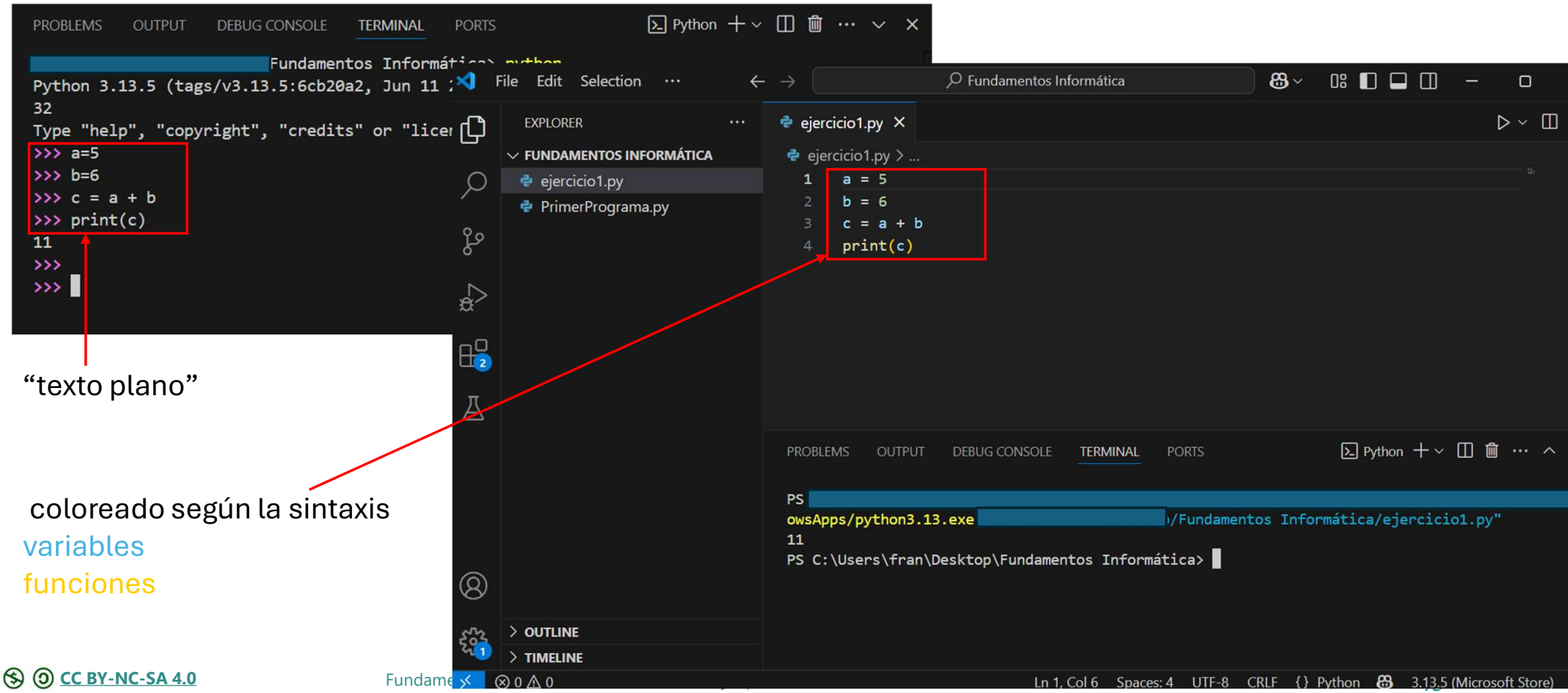
Ejecutar código sin crear un fichero desde la consola.

Se escribe la línea y se pulsa “intro”. Si no se escribe nada es un salto de línea.

Problemas:

1- No se guarda en el código generado.
2- El código es “texto plano”.
3- No integra algunas herramientas como depurador...

La consola vs Editor



"texto plano"

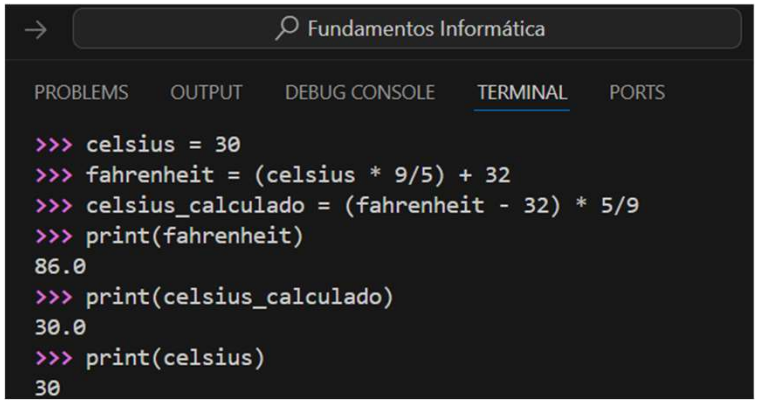
coloreado según la sintaxis
variables
funciones

Python como calculadora

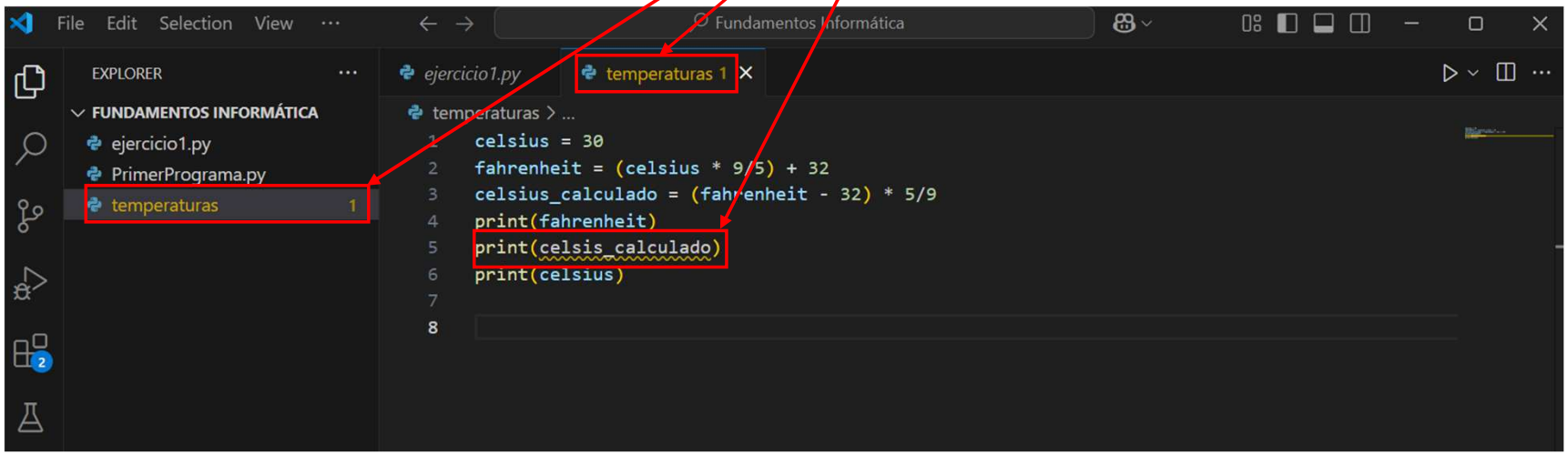
Utilizando la consola de Python:

1. Asigna una temperatura de 38 grados Celsius a una variable.
2. Convierte ese valor a Fahrenheit. $$F = (C \times \frac{9}{5}) + 32$$
3. Convierte el resultado nuevamente a Celsius (variable C2): .
4. Muestra ambos resultados. $$C = (F - 32) \times \frac{5}{9}$$

Python como calculadora



Más sencillo localizar errores usando el editor



Operadores básicos en Python

- **Aritméticos**:

  + (suma), - (resta), * (multiplicación), / (división), // (división entera), % (módulo o resto), ** (potencia o exponente)

- **Comparación**:

  == (igual a), != (distinto de), > (mayor que), < (menor que), >= (mayor o igual que), <= (menor o igual que)

- **Lógicos**:

  and (y lógico), or (o lógico), not (negación lógica)

- **Asignación**:

  = (asignación), += (suma y asignación), -= (resta y asignación), *= (multiplicación y asignación), /= (división y asignación), //= (división entera y asignación), %= (módulo y asignación), **= (potencia y asignación)

- **Identidad**:

  is (es el mismo objeto), is not (no es el mismo objeto)

- **Pertenencia**:

  in (está en), not in (no está en)

- **Bit a Bit**:

  & (AND bit a bit), | (OR bit a bit), ^ (XOR bit a bit), ~ (NOT bit a bit), << (desplazamiento a la izquierda), >> (desplazamiento a la derecha)

Funciones básicas en Python

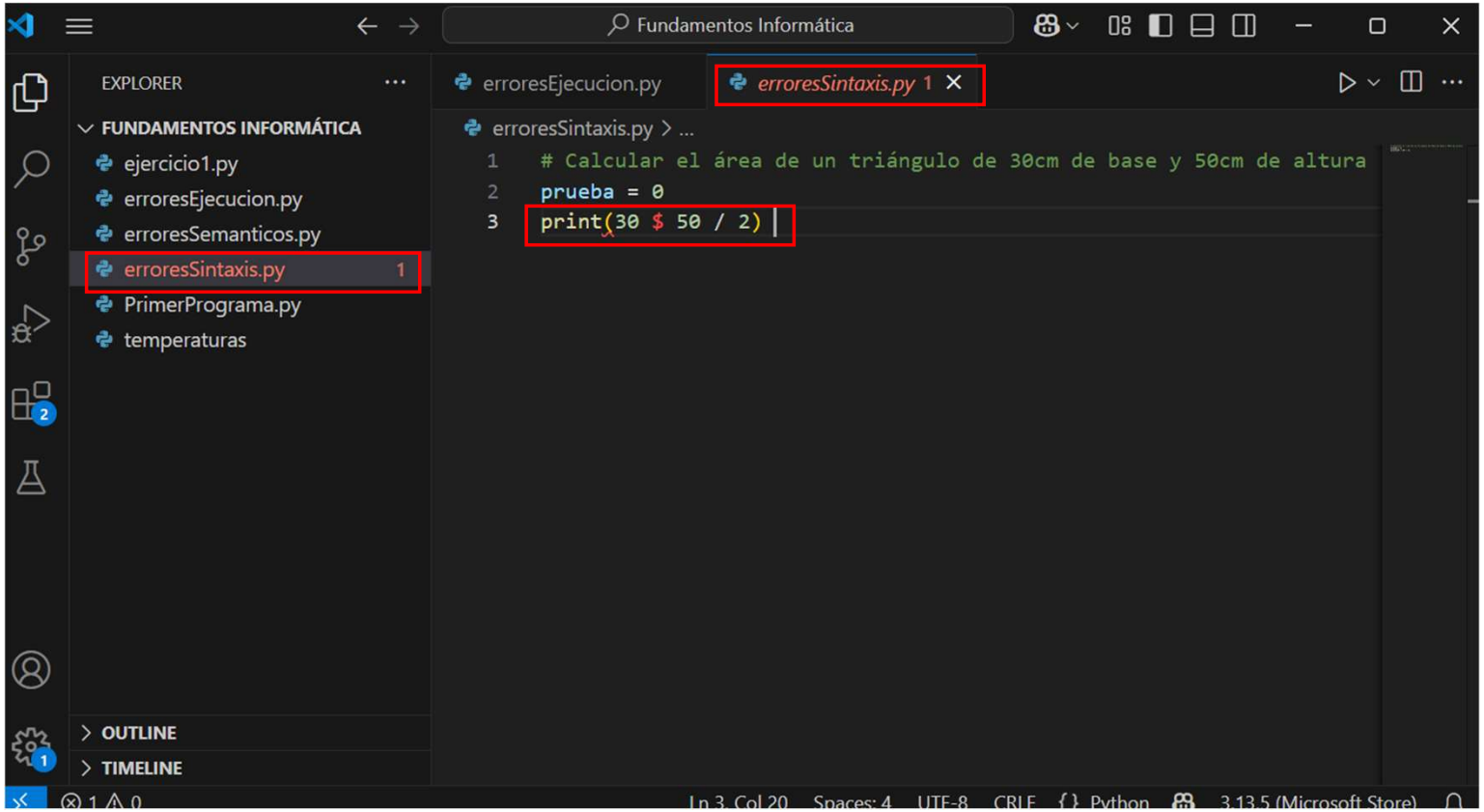
- **Aritméticas**:
  - abs() → Valor absoluto
  - round() → Redondea un número
  - pow(x, y) → Potencia x y x
  - max() / min() → Máximo / Mínimo entre varios valores
  - …
- **Lógicos**:
  - and → Verdadero si ambas condiciones son verdaderas
  - or → Verdadero si una o ambas condiciones son verdaderas
  - not → Invierte el valor lógico (negación)
  - …
- **Entrada / salida**:
  - input() → Lee datos desde la consola como texto
  - print() → Muestra datos en la consola
  - …
- **Tipos**:
  - type() → Muestra el tipo de dato
  - int() → Convierte a entero
  - float() → Convierte a número decimal
  - str() → Convierte a texto
  - …

Errores

1. Los errores en programación se denominan "gazapos ("bugs" en inglés), y encontrar esos errores se llama "depuración" ("debugging" en inglés).
2. Podemos catalogar los errores en tres tipos, por orden creciente de su dificultad de detección

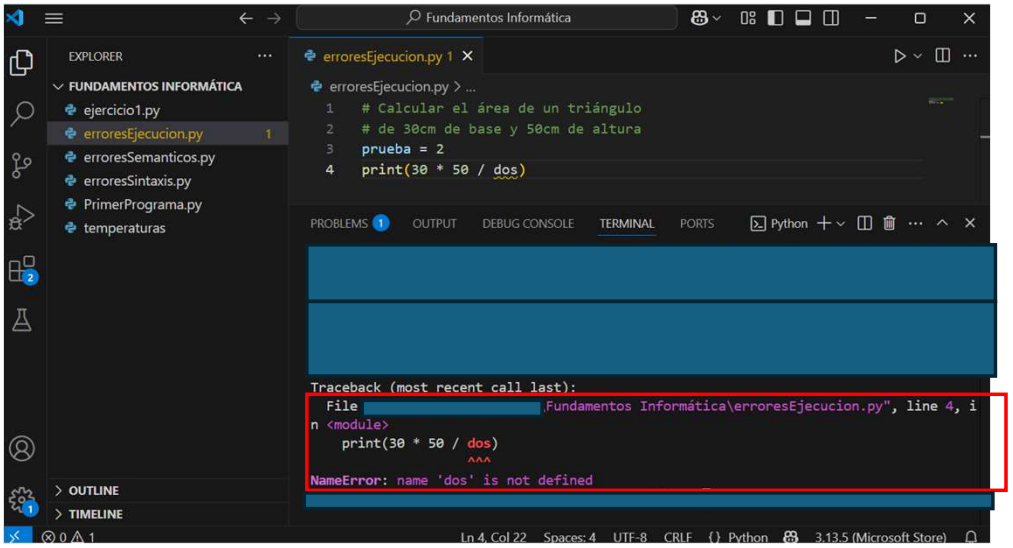
Errores sintácticos

- Son fáciles de encontrar porque Python los señala.
- El error de sintaxis de poner “$” en lugar de “*”.

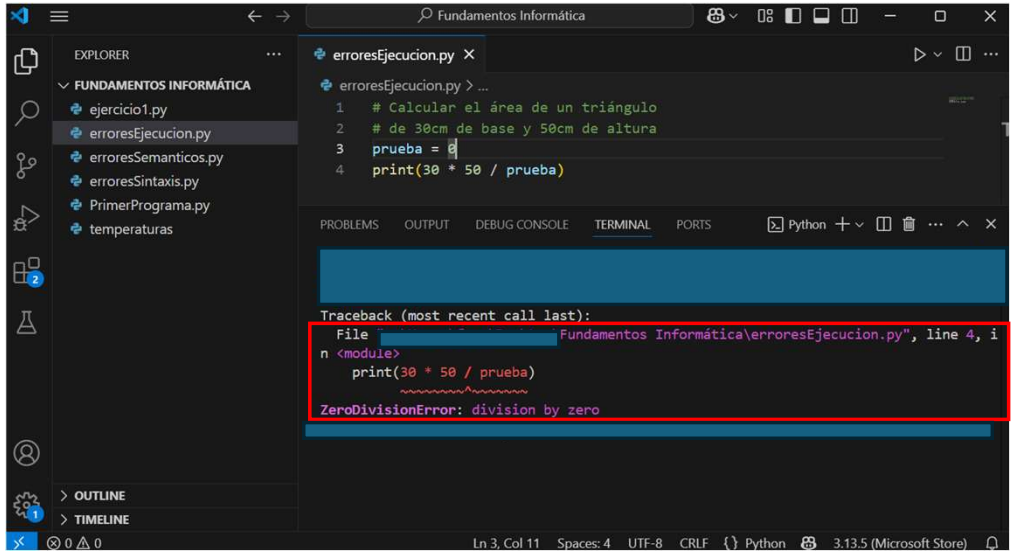


Errores en tiempo de ejecución

- Python no detectará el problema hasta que no llegue a ejecutar esa línea. (aunque Visual Code a veces si)
- Ocurre un error al utilizar la variable “dos” que no está definida.

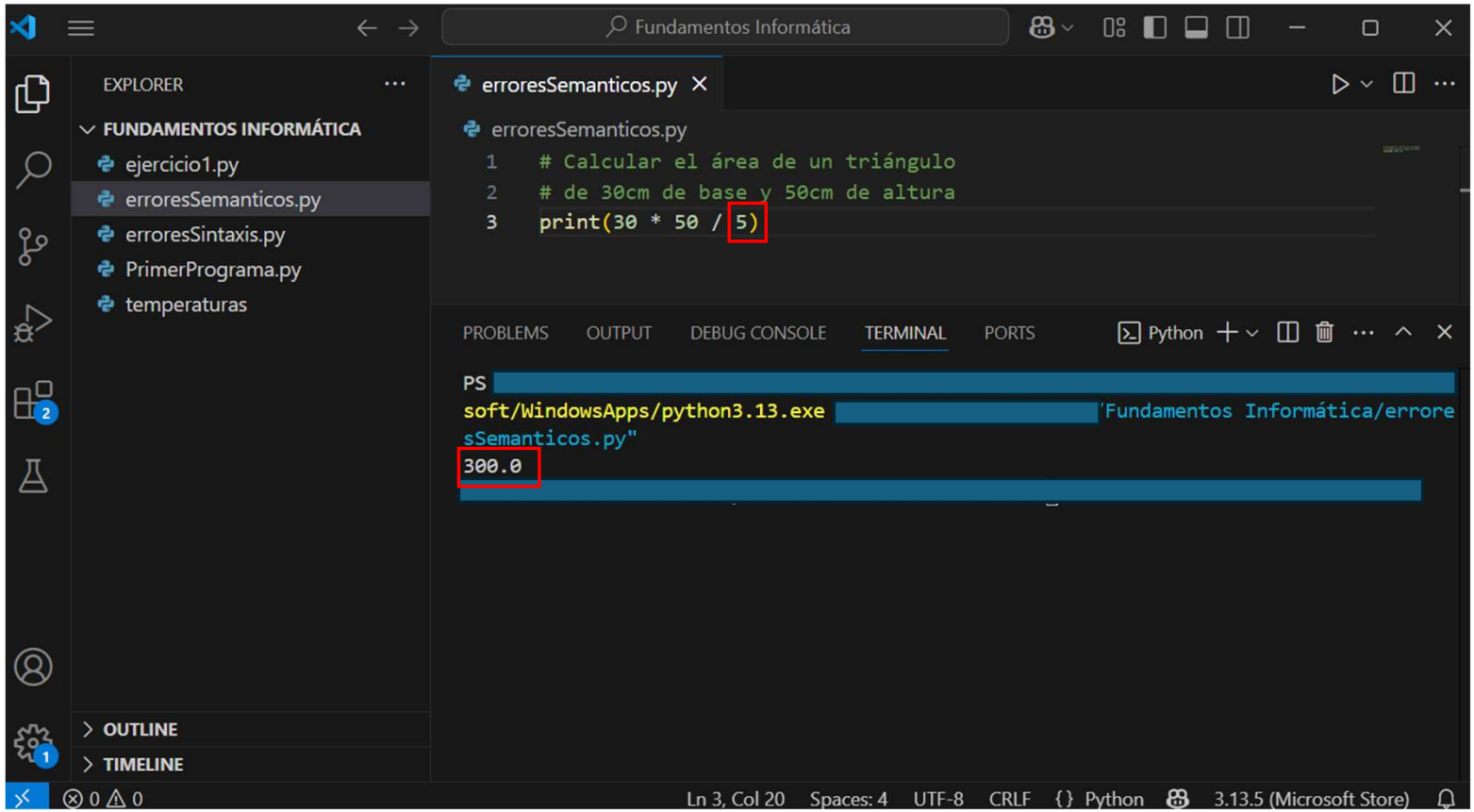


- Otros no se pueden detectar hasta que el código se ejecuta.
- Ocurre un error al dividir entre cero.



Errores semánticos

- Son difíciles de encontrar.
- El error semántico al calcular el área de un triangulo.
- En lugar de dividir por 2, hemos dividido por 5.



Ejercicios

1. Encuentra cuál es el mayor de estos números: 2^15, 3^12 y 5^10.
2. Usando las funciones chr y ord, encuentra cuál es la letra que está 10 posiciones más adelante de la A
3. Usando las funciones chr y hex (y tu cabeza) encuentra cuál es el código binario de la letra 'a' y de la letra 'A' y verifica que se diferencian tan solo en un bit
4. Escribe una expresión que calcule el resto de dividir 500 entre 7. Comprueba que sale 3.
5. Escribe un programa que haga salir en pantalla una línea formada por 80 asteriscos.
6. Calcula la raíz cuadrada de 2 con cinco decimales.
7. ¿Qué crees que debería salir al poner type(1/2)? Comprueba qué sale. Comprueba también el resultado de la operación. ¿Y con type(1//2)? ¿Cuál es la diferencia entre el operador / y el //?
8. Prepara una variable n con el valor 5. Calcular la raíz n-sima de 2. Debe salir 1.1486983549970351.