A6.1. UT2 Práctica Guiada 1: IDEs (Parte 1)

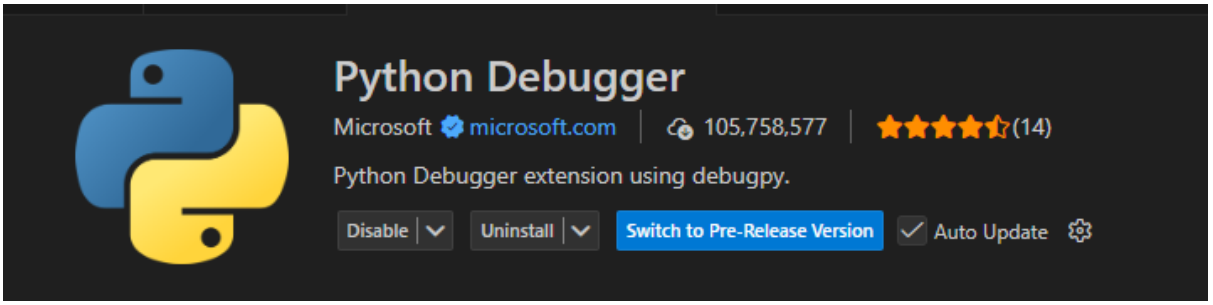
Actividad 2.
Para la realización de estás actividades prácticas hay disponibles una serie de scripts (los ficheros
adjuntos a la actividad en el comprimido “DepurarParte1.zip”) con los que trabajaremos.
Trata de ejecutar el primer script (script_1) y contesta a las preguntas.
¿En que lenguaje de programación está escrito?
Python

¿Es un lenguaje compilado o interpretado?
interpretado

¿Has
podido ejecutarlo directamente nada más instalar el Visual Studio Code?¿Que tienes que instalar
no he podido he tenido instalar python

para ejecutarlo?¿Has tenido que instalar algún plugin adicional?¿Cual?

Si este



**(adjunta capturas de pantalla para documentar lo que has hecho en cada paso hasta conseguir
ejecutar el código)**

Actividad 3. Busca información sobre el lenguaje en el que están escritos los scripts adjuntos y pon sus principales características, sus puntos fuertes, sus puntos débiles y principales campos de aplicación a día de hoy en el mundo de las tecnologías de la información. Busca plugins que te puedan ayudar a trabajar de forma más eficiente e instala alguno (al menos uno) ¿Cual/es has instalado?¿Para qué sirven? Documenta todo el proceso y el plugin funcionando (es decir como estaba el entorno antes y después de aplicar el plugin, que se vea la funcionalidad del plugin)

1. Lenguaje de los scripts: Python

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general, muy utilizado actualmente en el ámbito de las tecnologías de la información.

---

2. Principales características de Python

- **Sintaxis sencilla y legible.**
- **Lenguaje interpretado (no necesita compilación previa).**
- **Tipado dinámico.**

- **Multiplataforma (Windows, Linux, macOS).**
- **Multiparadigma: orientado a objetos, estructurado y funcional.**
- **Amplio ecosistema de librerías y frameworks.**
- **Gestión de paquetes mediante `pip`.**

---

3. Puntos fuertes de Python

- **Fácil de aprender y usar.**
- **Gran comunidad y documentación.**
- **Alta productividad y desarrollo rápido.**
- **Integración sencilla con otras tecnologías.**
- **Muy versátil: sirve para muchos ámbitos distintos.**
- **Ideal para scripts, automatización y prototipos.**

---

4. Puntos débiles de Python

- **Menor rendimiento que lenguajes compilados (C, C++, Java).**
- **Errores pueden aparecer en tiempo de ejecución por el tipado dinámico.**
- **Limitaciones en concurrencia debido al GIL.**
- **No es el más adecuado para sistemas en tiempo real o videojuegos exigentes.**

---

5. Principales campos de aplicación actuales

- **Desarrollo web: Django, Flask.**
- **Inteligencia Artificial y Machine Learning.**
- **Ciencia de datos y Big Data.**
- **Automatización y scripting.**
- **DevOps y administración de sistemas.**
- **Educación.**
- **Aplicaciones de escritorio.**
- **Ciberseguridad.**

---

6. IDE utilizado

Visual Studio Code (VS Code)
Editor multiplataforma muy utilizado para Python gracias a su sistema de extensiones.

Plugins:

Plugin Prettier (VS Code)

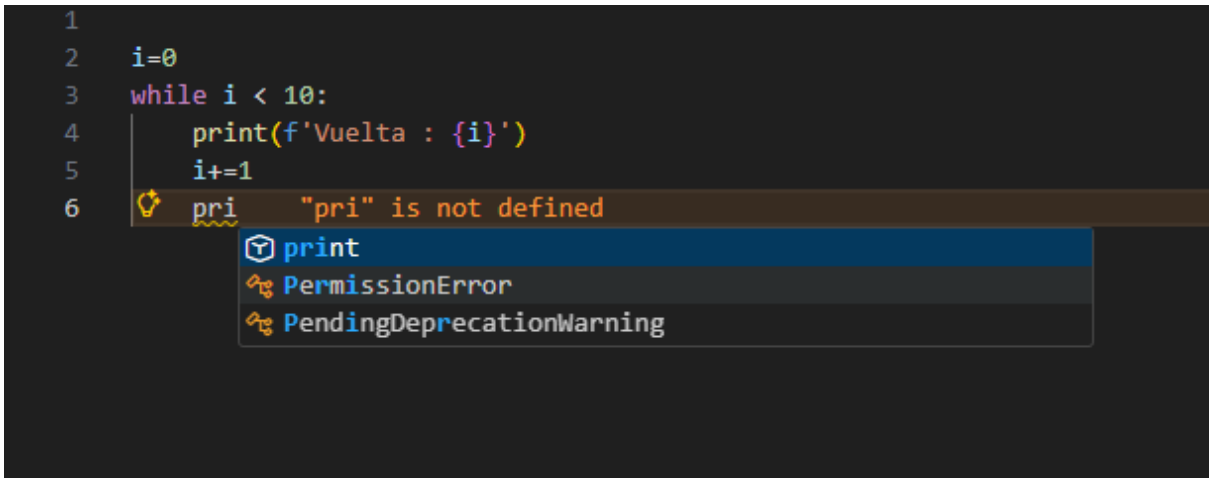
Nombre del plugin

Prettier – Code formatter

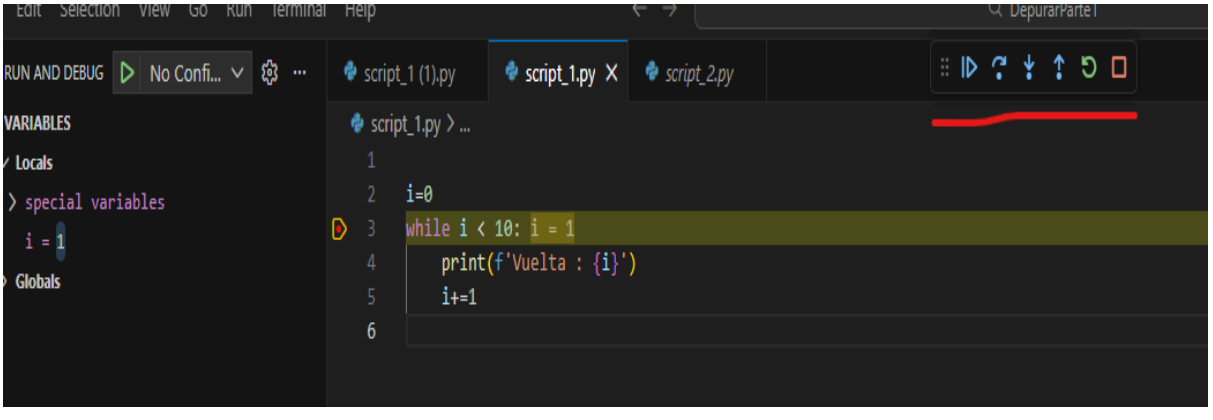
¿Para qué sirve Prettier?

Prettier es un formateador automático de código. Reordena y da formato al código siguiendo un estilo consistente sin que tengas que hacerlo a mano.

Aunque se usa mucho en HTML, CSS y JavaScript, también puede usarse con Python (normalmente junto a Black).



Ahora que sabemos como va la depuración y los puntos de ruptura realiza las siguiente actividades: 4. 1. Pon un punto de ruptura en el script_1.py donde creas conveniente y ejecuta línea a línea viendo como se va ejecutando el código y los valores que van imprimiéndose por pantalla. Prueba con las 4 opciones de ejecución (continue, step over, step into, step out). ¿Que diferencias hay entre los 4 modos de ejecución para este caso?



Modos de ejecución del depurador

1. Continue (Continuar)

- **Reanuda la ejecución normal del programa.**
- **El programa sigue corriendo hasta el siguiente punto de ruptura o hasta que finaliza.**
- **No se detiene en cada línea intermedia.**

En este caso:
Solo sirve para ver el resultado final por pantalla sin analizar paso a paso qué ocurre entre medias.

---

2. Step Over (Paso por encima)

- **Ejecuta la línea actual y pasa a la siguiente.**
- **Si la línea contiene una llamada a una función, la función se ejecuta completa sin entrar en ella.**
- **El depurador se detiene en la siguiente línea del mismo nivel.**

En este caso:
Permite ver cómo cambian las variables principales sin entrar al detalle interno de las funciones.

---

3. Step Into (Paso a paso / Entrar)

- **Ejecuta la línea actual entrando dentro de la función si la hay.**
- **Permite depurar el código línea a línea dentro de la función.**
- **Es el modo más detallado.**

En este caso:
Sirve para ver exactamente cómo se ejecuta el código interno de una función y cómo se calculan los valores antes de imprimirse.

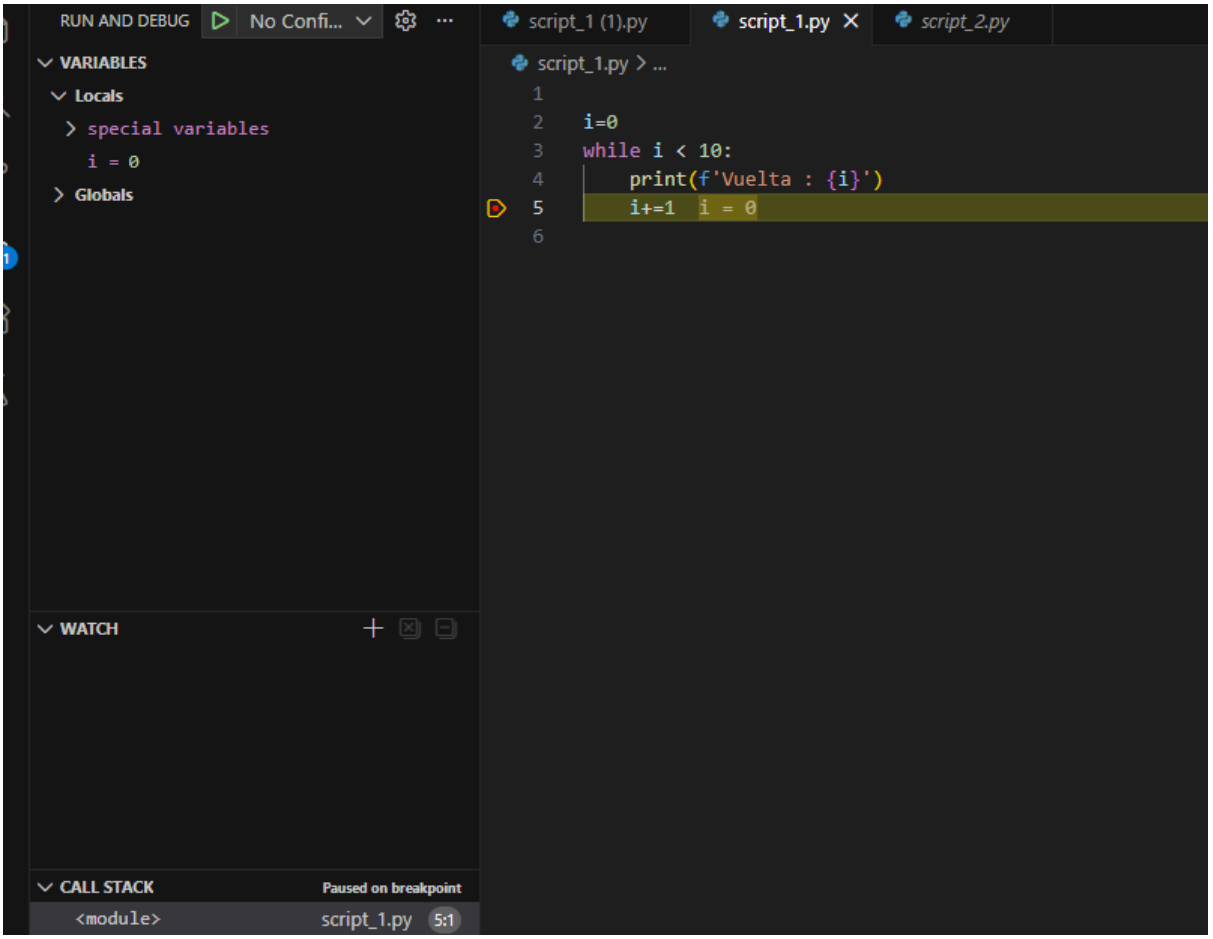
---

4. Step Out (Salir)

- **Ejecuta el resto del código de la función actual de una sola vez.**
- **Sale de la función y se detiene en la línea donde fue llamada.**
- **Útil cuando ya no interesa seguir depurando esa función.**

En este caso:
Permite volver rápidamente al flujo principal del programa sin seguir viendo cada línea interna de la función.

4.2. Ahora ejecuta el script_2.py y coloca nuevamente al menos un punto de ruptura. Prueba con las 4 opciones de ejecución (continue, step over, step into, step out). ¿Que diferencias hay entre los 4 modos de ejecución? (Saca capturas de pantalla dónde se vean las diferencias si es que las hay)



Diferencias entre los modos del depurador

- **Continue**
  Continúa la ejecución normal del programa hasta el siguiente punto de ruptura o hasta que termine. No se detiene en cada línea.

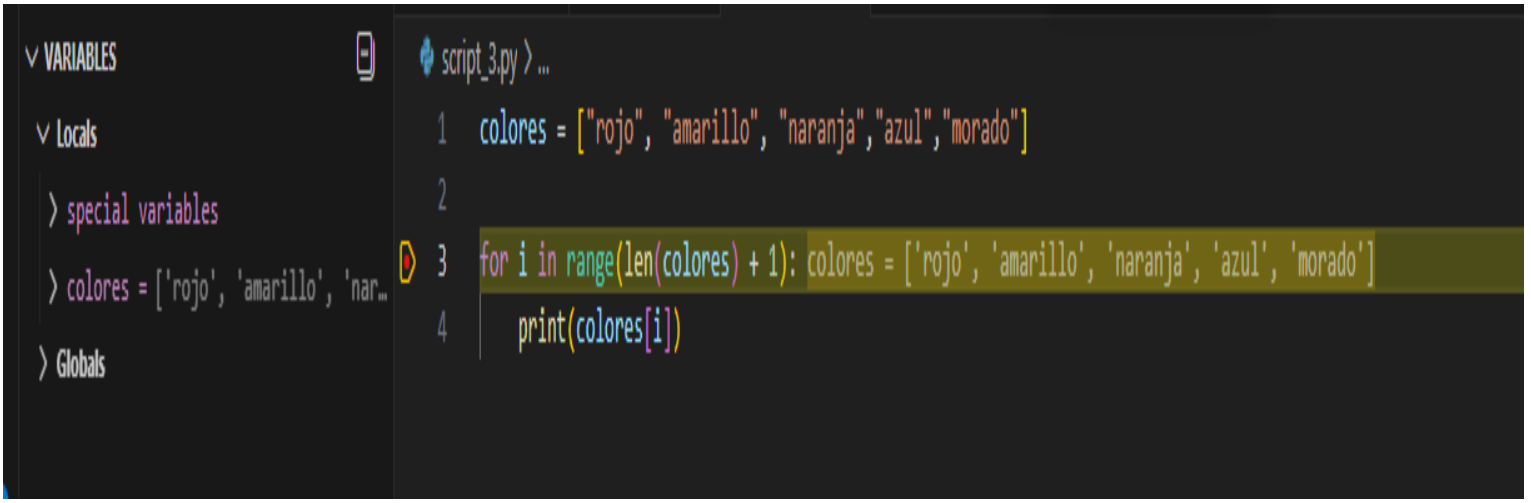
- **Step Over**
  Ejecuta la línea actual y pasa a la siguiente.
  Si la línea contiene una llamada a una función, la ejecuta

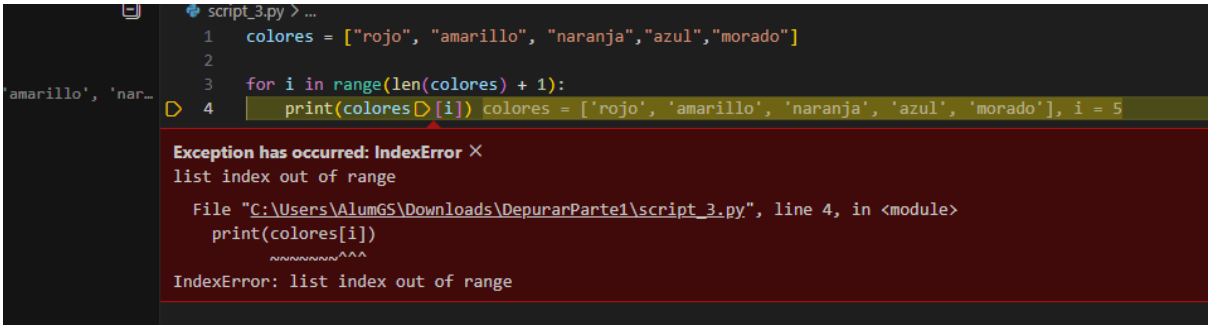
completa sin entrar en ella.

- **Step Into**
  Ejecuta la línea actual y, si hay una llamada a una función, entra dentro de esa función para depurarla línea a línea.

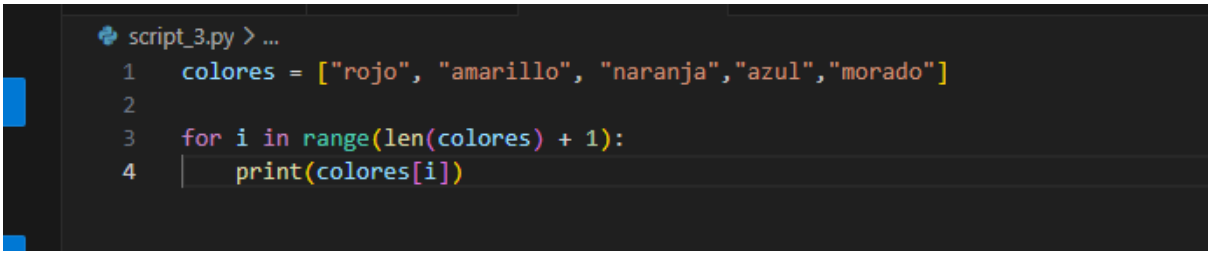
- **Step Out**
  Sale de la función en la que estás y continúa la ejecución hasta volver a la línea desde la que fue llamada.

4.3. Ejecuta el script_3.py en modo normal. Saca captura de pantalla de lo que ocurre. ¿Que está ocurriendo? Pon un punto de ruptura y ejecuta paso a paso para dar con el error y solucionarlo ¿Que has hecho para solucionarlo? (Adjunta capturas de pantalla de la ejecución paso a paso hasta dar con el problema y solucionarlo).





Pulse f5 y me volvio al estado del código sin errores



4.4Ejecuta el script_4.py ¿Se produce algún error?¿Por qué? Explica que está ocurriendo.

El error ocurre porque el programa intenta sumar un número entero a una cadena de texto, algo que Python no permite.

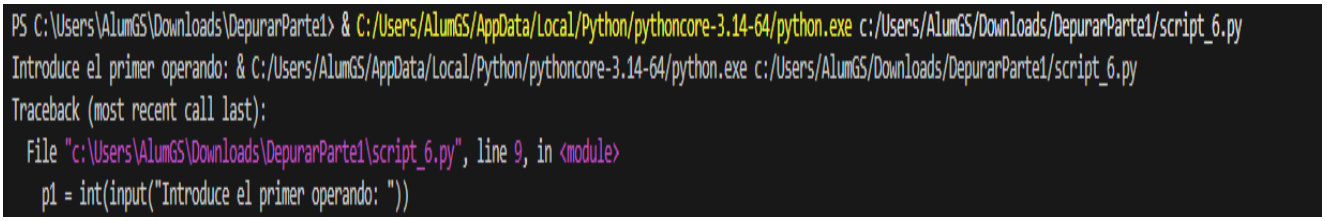
4.5. Ejecuta el script_5.py paso a paso fíjate en los valores que van tomando las variables
(situándote encima con el ratón y también observando en la parte izquierda en la sección de
“Variables”). Comenta lo que va haciendo el código y documenta lo que ves (que tipo de
ejecuciones has hecho: step over, step into,etc).

```
rojo
Traceback (most recent call last):
  File "c:\Users\AlumGS\Downloads\DepurarParte1\script_4.py", line 5, in <module>
    resultado=colores[i]+1
              ~~~~~~~~~~^~
```

Al iniciar la depuración se coloca un punto de ruptura en la llamada a la función `operaciones(5, 10)` y se comienza la ejecución con Continue hasta detenerse en ese punto.
A continuación se utiliza Step Into para entrar en la función `operaciones`, donde se observa que el parámetro a toma el valor 5 y b el valor 10.
Con Step Over se ejecuta la línea `resultado_intermedio = a * 2`, asignándose el valor 10 a la variable.

En la siguiente línea, también con Step Over, `resultado_intermedio_2` pasa a valer 100 al multiplicar el valor anterior por 10.
Después, usando nuevamente Step Over, se calcula `resultado_final`, que toma el valor 120 al sumar todas las variables.
Al ejecutar la sentencia `return`, se utiliza Step Out para salir de la función y volver al programa principal, donde `resultado_operacion` recibe el valor devuelto.
Finalmente, con Continue, se ejecuta la instrucción `print`, mostrando el resultado final por pantalla.

4.6. Ejecuta el script_6.py. Este script como verás, se parece al anterior con la diferencia de que es



el usuario el que pasa por teclado el valor de los operandos.
Prueba a ejecutar el script con varios
valores. ¿Que pasa si no le metes un número y le metes letras? (Documenta el proceso como has
hecho anteriormente donde quede reflejado todas las pruebas que has hecho).

Al ejecutar `script_6.py`, el programa solicita al usuario dos valores por teclado mediante la función `input()`, que posteriormente se convierten a enteros con `int()`.
Cuando se introducen números válidos, el programa continúa su ejecución sin errores, entra en la función `operaciones` y calcula correctamente los valores intermedios y el resultado final.
Durante la depuración se ha colocado un punto de ruptura antes de la lectura de los datos y se ha utilizado Step Over para observar cómo las variables `p1` y `p2` toman los valores introducidos por el usuario.
Al entrar en la función con Step Into, se comprueba cómo se calculan `resultado_intermedio`, `resultado_intermedio_2` y `resultado_final`.
En las pruebas realizadas introduciendo letras en lugar de números, el programa lanza un ValueError, ya que Python no puede convertir una cadena de texto a entero mediante `int()`.
Este error se produce antes de entrar en la función, impidiendo que el resto del código se ejecute.

4.7. Ejecuta el script_7.py en modo normal para entender qué hace el programa.
En teoría el usuario introduce por teclado un número y el programa calcula si es un numero par o
impar; después pregunta si quiere de salir de la ejecución o si quiere seguir. Si se introduce una ‘s’
se sale de la ejecución; si se introduce una ‘n’ sigue pidiendo al usuario que introduzca otro número.
Prueba a introducir distintos valores por teclado para poner a prueba el programa (documenta

indicando los valores que has probado con capturas de pantalla). ¿Funciona correctamente? ¿Hay
algún comportamiento que no debería tener? ¿Si en lugar de poner ‘y’ o ‘n’ ponemos lo mismo pero
con mayúsculas qué pasa?¿Por qué?
Prueba a poner puntos de ruptura y ejecutar el código línea a línea para dar con el error y
solucionarlo.

Al ejecutar el programa, funciona bien al calcular par o impar.
Si se introduce una letra en lugar de número, muestra el error controlado con `ValueError`.
Si se introduce `Y` o `N` en mayúsculas, también funciona porque se usa `.lower()`.
El problema es que al introducir n no se sale del bucle, porque el código usa `continue` en vez de `break`.
Se ha corregido cambiando `continue` por `break` cuando se introduce n.