Guía del Entorno de Programación Thonny

Dpto. Lenguajes y Ciencias de la Computación. E.T.S.I. Informática.

Programación-II

Thonny es un entorno de desarrollo (IDE) para programas Python, simple y fácil de utilizar, orientado a personas que están aprendiendo el lenguaje, desarrollado en la Universidad de Tartu (Estonia).

Desarrollo de programas Python en Thonny

El *entorno de programación* (IDE) **Thonny** nos permite crear, editar y ejecutar ficheros de código fuente en Python.

Inicialmente muestra tres ventanas principales.

- La **ventana de edición** (*editor de textos*), etiquetada con el nombre del fichero que estemos editando, permite editar y modificar los ficheros de código fuente en Python.
- La **Consola** muestra el resultado de la ejecución de los programas, así como también se utiliza para introducir datos interactivos para los programas en ejecución.
- El **Asistente**, muestra información de interés sobre el análisis, realizado por las herramientas **Pylint** y **Mypy**, aplicadas al programa que se está ejecutando. Este análisis es muy importante para la mejora y depuración de los programas.

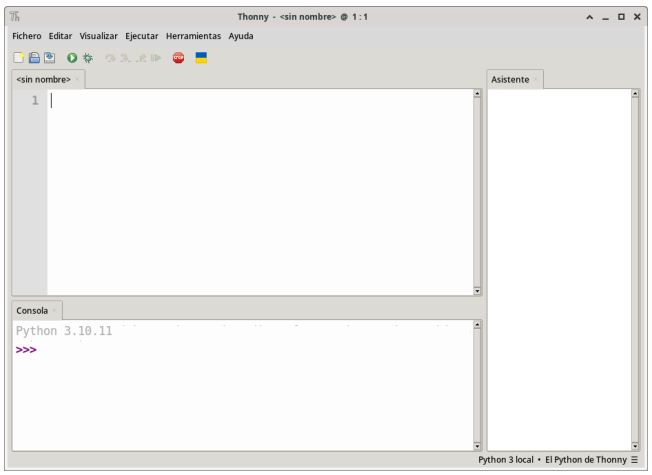

Fig.1: Ventana de edición, Consola y Asistente.

Pulsar sobre el icono **Nuevo Programa (Ctrl+N)** nos permite crear un nuevo fichero de código fuente en Python. Sin embargo, pulsar sobre el icono **Abrir fichero (Ctrl+O)** permite abrir y editar un fichero de código ya existente.

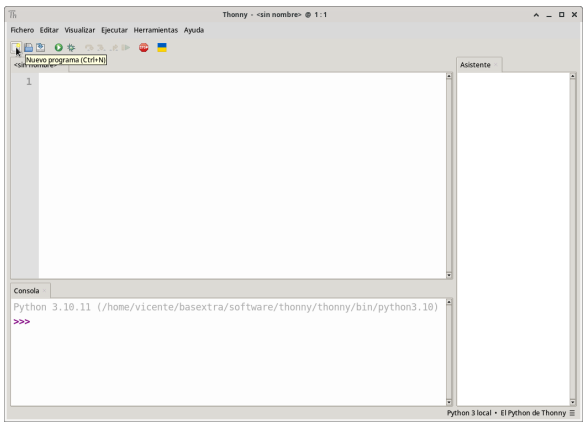

Fig.2: Creación de nuevo programa.

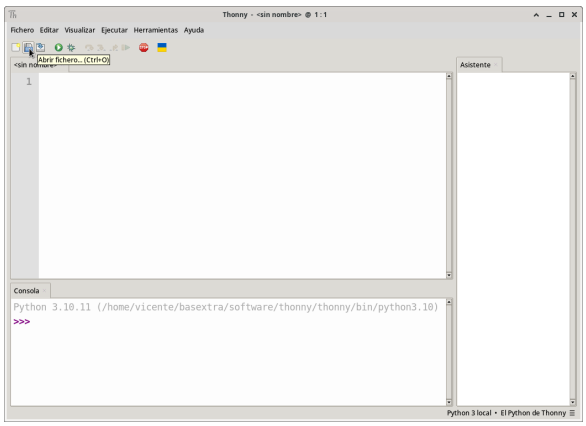

Fig.2: Abrir y editar un programa ya existente.

Codifique, utilizando el editor de textos del entorno de desarrollo, el siguiente programa Python, exactamente igual que aparece en el siguiente texto *(incluyendo los errores que contiene)*.

```
from typing import Final

PTS_EUR: Final = 166.386

def conv_euros(pts: int) -> float
    """Convierte las pesetas recibidas a euros """
    return ptas / PTS_EUR

def main() -> None
    """Lee la cantidad de pesetas, y muestra equivalente en euros"""
    pts = int(input("Introduce la cantidad de pesetas: "))
    euros = conv_euros(pts)
    print(pts, "pesetas equivalen a", euros, "euros")

if __name__ == "__main__":
    main()
```

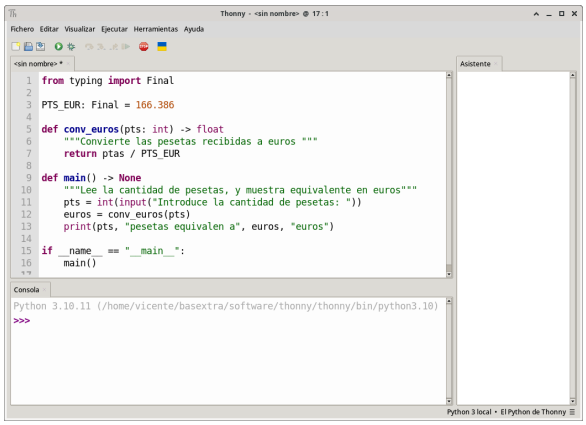

Fig.3: Programa modificado, sin guardar.

Mientras se edita un programa, si la versión actual no ha sido **guardada** en el fichero en memoria secundaria (permanente), entonces aparece un símbolo asterisco (*) en la pestaña junto al nombre del fichero.

Para guardar en memoria secundaria el programa modificado, es necesario pulsar el icono **Guardar** (**Ctrl+S**) o pulsar la combinación de teclas **Ctrl+S** (nótese como desaparece el símbolo * de la pestaña). Si el fichero es nuevo, en la operación de guardar habrá que seleccionar la carpeta y el nombre del fichero (con la extensión `.py` para todos los programas Python). En el caso contreto de este ejercicio, el fichero se llamará **p1_euros.py**, y lo guardaremos en la carpeta **python/src/clases/practica_1** (si no existe, la deberemos crear).

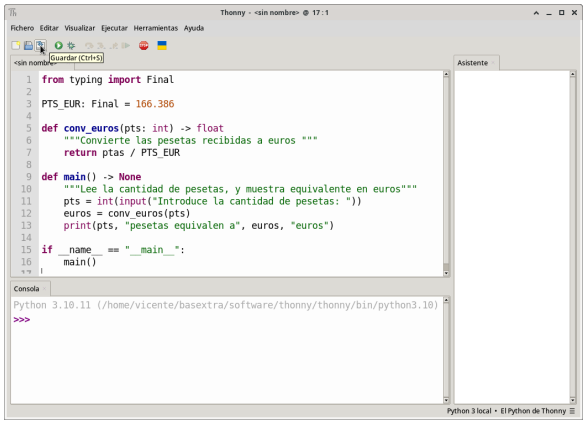

Fig.4: Guardar el programa.

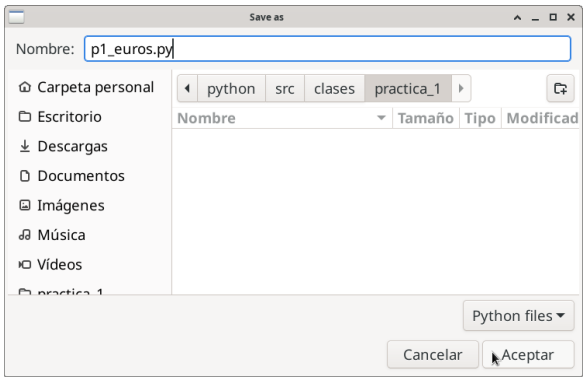

Fig.5: Seleccionar carpeta y nombre del fichero.

Es conveniente que guardemos **frecuentemente** las modificaciones de nuestros programas, pulsando la combinación de teclas **Ctrl+S**, o **Ctrl+Alt+S** para guardar todos los ficheros de todas las pestañas que estemos editando.

Podemos ejecutar el programa seleccionado en la pestaña activa pulsando sobre el icono **Ejecutar (F5)**, o pulsando la tecla **F5**.

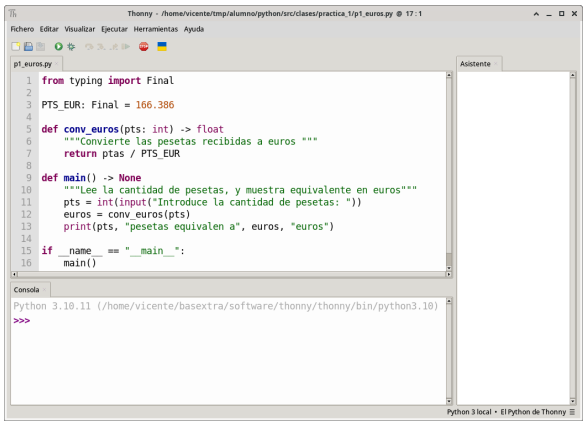

Fig.6: Programa guardado con nombre p1_euros.py.

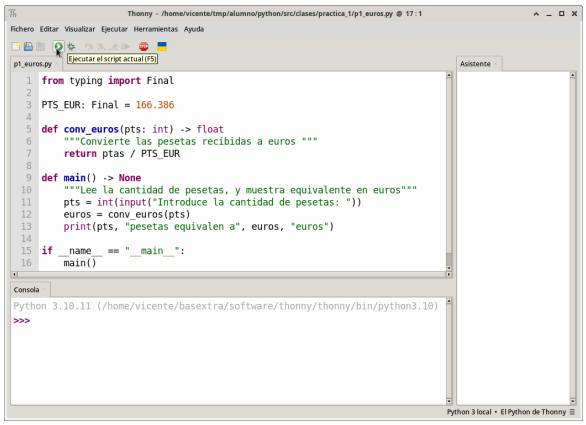

Fig.7: Ejecutar el programa de la pestaña activa.

En este caso, el programa tiene algunos errores sintácticos, por lo que no puede ser ejecutado. Tanto la **Consola** como el **Asistente** indican que falta poner el símbolo dos-puntos (**:**) al final de la línea 5. Corregiremos el error (añadir el símbolo dos-puntos al final de la línea 5), guardaremos el fichero (**Ctrl+S**), y volveremos a ejecutar (**F5**).

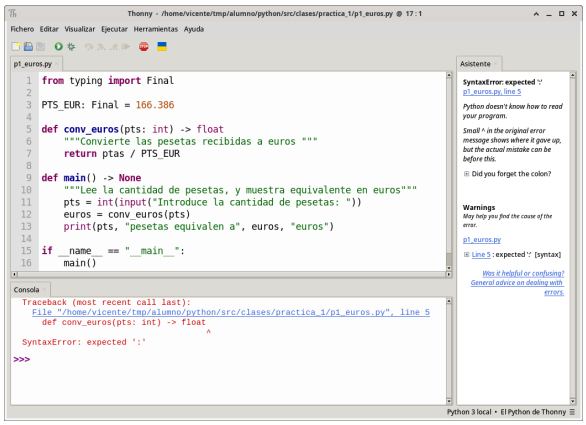

Fig.8: Error en línea 5.

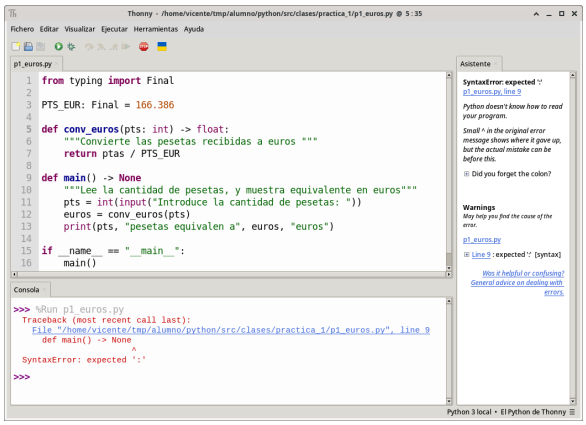

Fig.9: Error en línea 9.

En este caso, el programa tiene algunos errores sintácticos, por lo que no puede ser ejecutado. Tanto la **Consola** como el **Asistente** indican que falta poner el símbolo dos-puntos (**:**) al final de la línea 9. Corregiremos el error (añadir el símbolo dos-puntos al final de la línea 9), guardaremos el fichero (**Ctrl+S**), y volveremos a ejecutar (**F5**).

Ahora se ejecuta nuestro programa, en la **Consola** nos muestra un mensaje, proveniente de nuestro programa, solicitando que introduzcamos una cantidad de pesetas. Introduciremos el número **500** y pulsaremos la tecla **ENTER**.

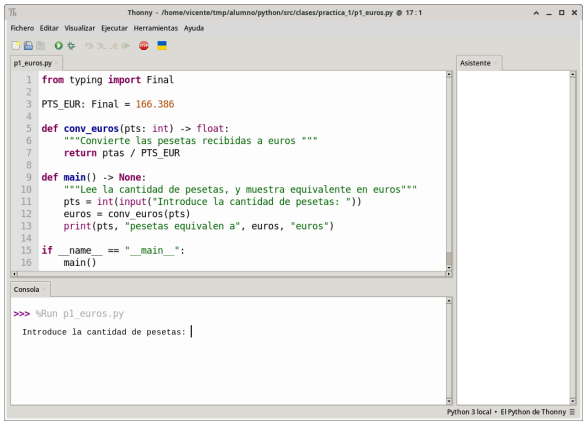

Fig.10: Ejecución del programa. Espera por entrada de teclado.

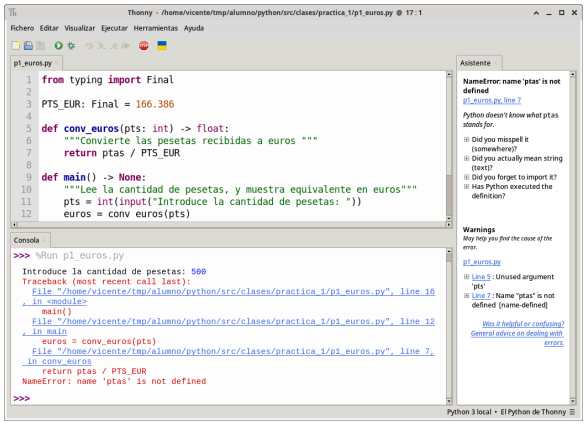

Fig.11: Error de ejecución. Variable ptas no existe.

Ahora tenemos un error durante la ejecución de nuestro programa. En este caso, tanto la **Consola** como el **Asistente** nos indican que en la línea 7 hemos utilizado el nombre de variable **ptas**, que no existe, además, el asistente también indica que el parámetro **pts** no ha sido utilizado, por lo tanto, para corregir el error, debemos cambiar el nombre de la variable de la línea 7, y nombrarlo como **pts**. Volveremos a guardar el fichero (**Ctrl+S**) y volveremos a ejecutarlo (**F5**). Ahora se ejecuta correctamente y muestra en **Consola** el mensaje para introducir la cantidad de pesetas (introduciremos **500**), y muestra el resultado de convertir 500 pesetas a euros. Nótese también como el **Asistente** muestra que el análisis de nuestro programa ha sido correcto.

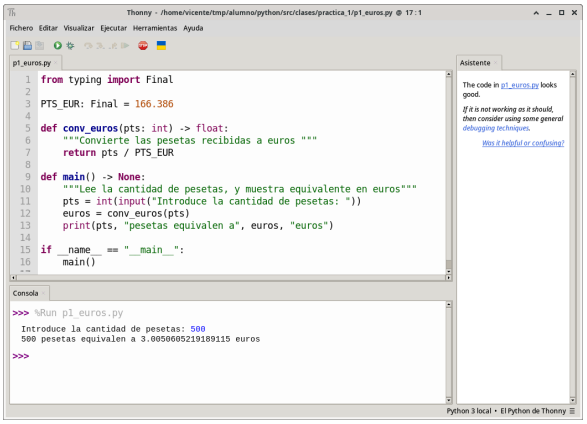

Fig.12: Ejecución correcta del programa.

La combinación de una tecla **Windows** y la **Barra Espaciadora** (pulsadas simultáneamente) permite commutar el idioma del teclado.

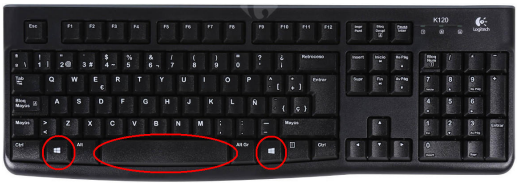

Fig.13: Conmutar el idioma del teclado.

Fig.14: Conmutar el idioma del teclado.

Ejecución de pruebas unitarias

Durante la ejecución de **pruebas unitarias**, por problemas internos, la **consola** de *Thonny* no muestra el resultado completamente. En caso de ejecución completa, la prueba unitaria debería mostrar al final del resultado unas líneas similares a las siguientes:

```
----------------------------------------------------------------------
Ran 8 tests in 0.014s

FAILED (failures=2)
```

```
----------------------------------------------------------------------
Ran 8 tests in 0.010s

OK
```

Cuando la ejecución de las pruebas unitarias no muestra el resultado completamente, puede ser conveniente ejecutar nuestro programa (prueba unitaria) en un **terminal externo**. Para ello, podemos seleccionar la opción **Ejecutar el script actual en terminal** (**Ctrl+T**) del menú **Ejecutar** de la barra de menú superior.

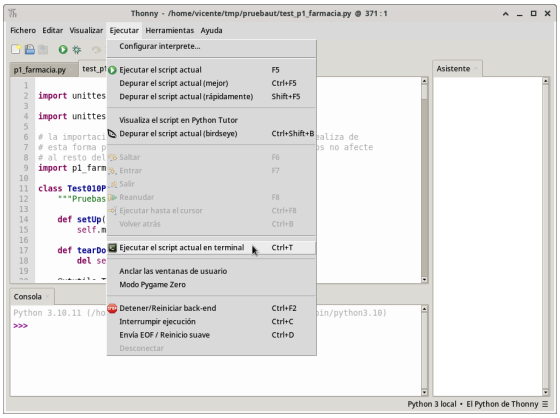

Fig.15: Ejecución del programa en terminal externo.

Depuración de programas Python en Thonny

A veces, es posible que nuestro programa no funcione adecuadamente, y no produzca los resultados y la salida esperada.

En estos casos, el **análisis mental y comprensión** del funcionamiento de nuestro programa es muy importante para poder detectar y corregir los errores.

Sin embargo, a veces el análisis y comprensión del funcionamiento del programa no es suficiente para encontrar los errores, por lo que mostrar la información de las variables durante la ejecución del programa puede llegar a ser una ayuda importante.

Estos valores de las variables se pueden mostrar simplemente con sentencias `print("DBG", ...)`, que después de corregir el error, deberán ser eliminadas o puestas como comentarios.

No obstante, a veces, puede ser conveniente utilizar herramientas más sofisticadas, tales como los **depuradores**, que ayudan a ejecutar el programa **paso-a-paso**, y permiten ver el valor de las variables durante la ejecución.

Nótese que en la ejecución de un programa Python, primero se cargan (ejecutan) todas las definiciones del programa (importación de módulos, constantes, definiciones de funciones y clases, etc.) y finalmente se ejecuta la función **main** invocada desde la sentencia

```
if __name__ == "__main__":
    main()
```

El entorno de programacion Thonny tiene dos modos de depuración, **depuración rápida** (*faster*), más fácil de usar, y la **depuración detallada** (*nicer*), que realiza una depuración más detallada de los componentes individualizados de las sentencias. En los siguientes ejemplos utilizaremos el modo de **depuración rápida** (*faster*).

Cuando depuramos un programa, es conveniente visualizar el valor de las variables que se están utilizando en un determinado momento de la ejecución del programa. Para ello, se debe activar la **visualización de las variables** durante la ejecución del programa, que podemos volver a desactivar después de realizar la depuración.

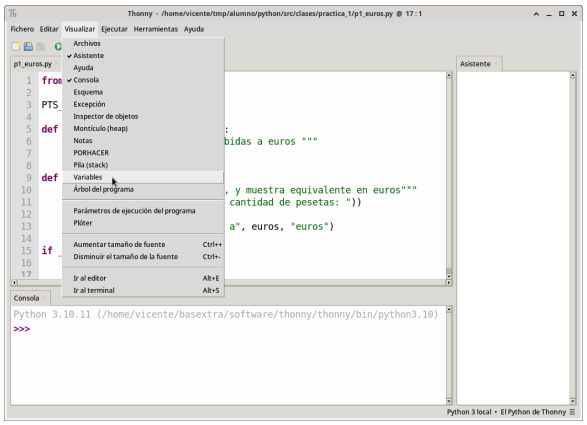

Fig.16: Activar **visualizar variables** durante la depuración.

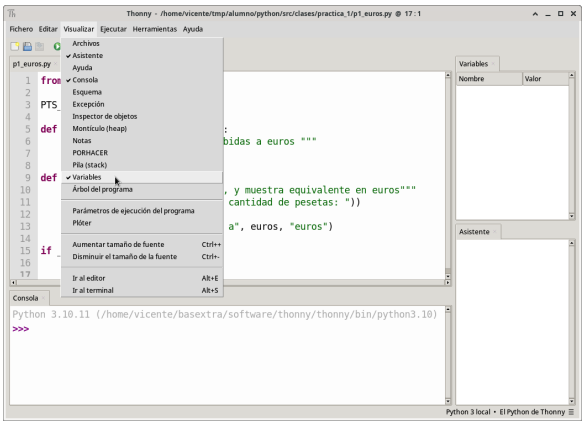

Fig.17: Activar **visualizar variables** durante la depuración.

Para ejecutar un programa utilizando el depurador, pulsaremos sobre el icono **Depurar**, realiza la depuración en el modo seleccionado en la configuración general, o la combinación de teclas **Mayúsculas+F5** para depurar en *modo rápido*, o la combinación de teclas **Ctrl+F5** para depurar en *modo detallado*.

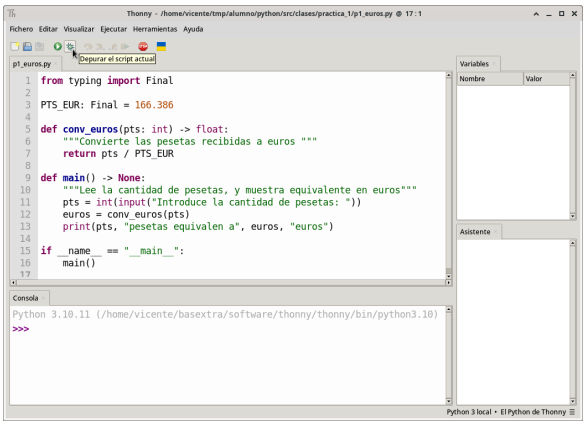

Fig.18: **Depurar** (ejecutar paso-a-paso) el programa.

Disponemos de los siguiente iconos para poder ejecutar **paso-a-paso** el programa:

- **Saltar (F6)**: ejecuta completamente la sentencia sobre la que se encuentra el cursor de ejecución actual. Por ejemplo, si es una invocación a una función, ejecuta completamente la función, sin entrar en las sentencias que la componen.
- **Entrar (F7)**: en caso de invocación a función, la ejecución paso-a-paso entra en las sentencias que componen la función. Además, en el modo de depuración detallado, ejecuta paso-a-paso los componentes internos de las sentencias, de forma detallada.
- **Salir**: ejecuta de forma continua todas las sentencias que componen la función actual, hasta que la ejecución de la función actual termina. Permite que la ejecución salga de la función actual, y continuar paso-a-paso después.
- **Reanudar (F8)**: reanuda la ejecución del programa, desde el punto actual, de forma continua, sin parar hasta el final del programa.
- Durante la ejecución paso-a-paso del programa, se van **mostrando los valores de las variables locales de las funciones**.

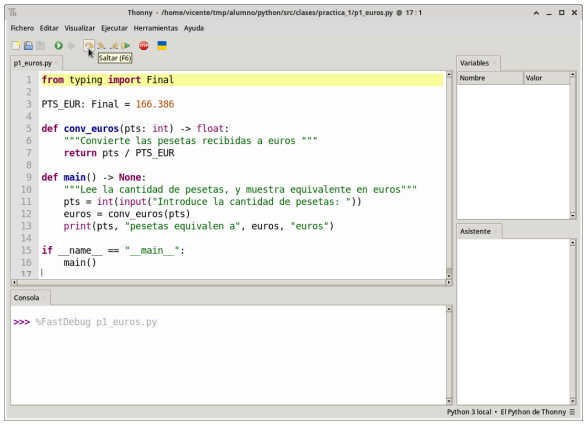

Fig.19: **Saltar**: ejecuta completamente la sentencia actual.

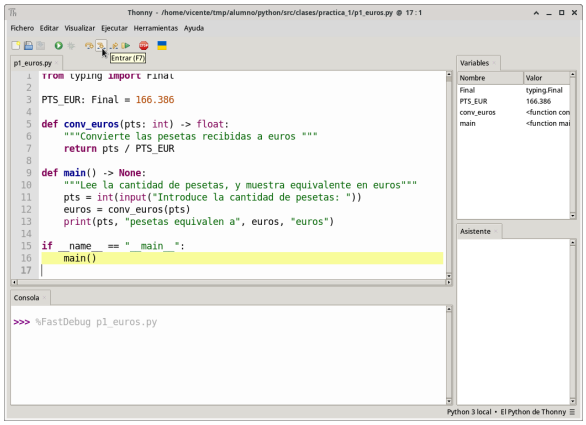

Fig.20: **Entrar**: entra paso-a-paso dentro de la función.

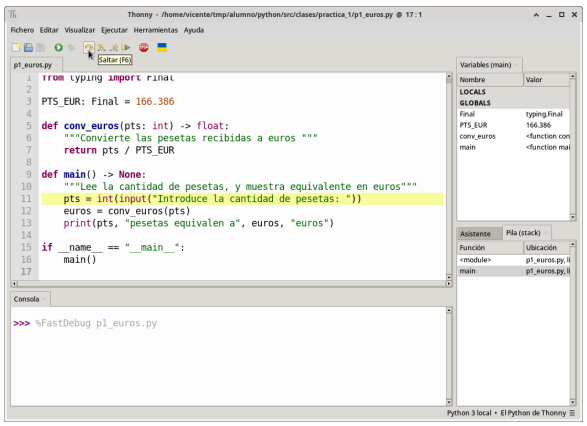

Fig.21: **Saltar**: ejecuta completamente la sentencia actual.

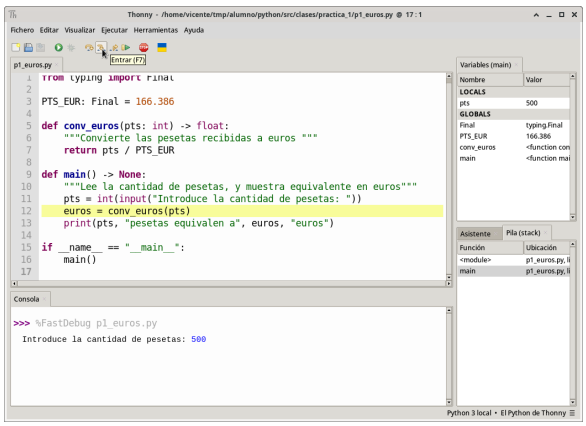

Fig.22: **Entrar**: entra paso-a-paso dentro de la función.

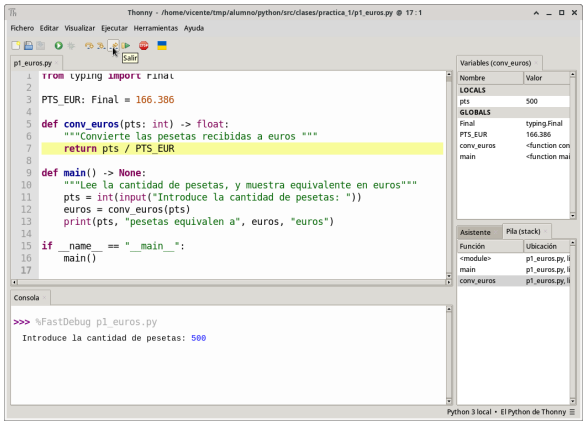

Fig.23: **Salir**: ejecuta hasta el final de la función.

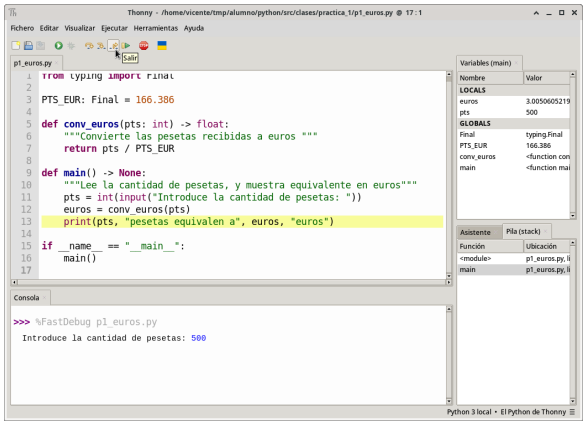

Fig.24: **Salir**: ejecuta hasta el final de la función.

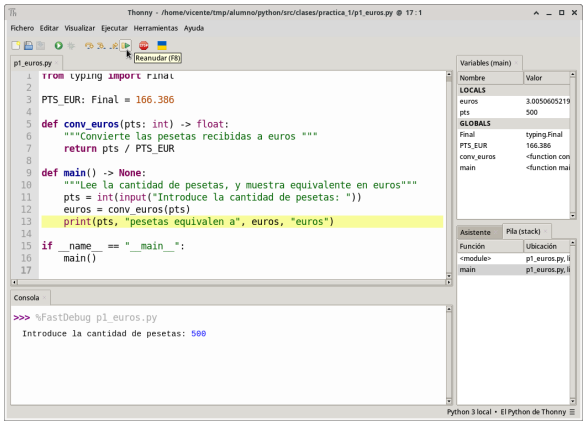

Fig.25: **Reanudar**: ejecuta hasta el final del programa.

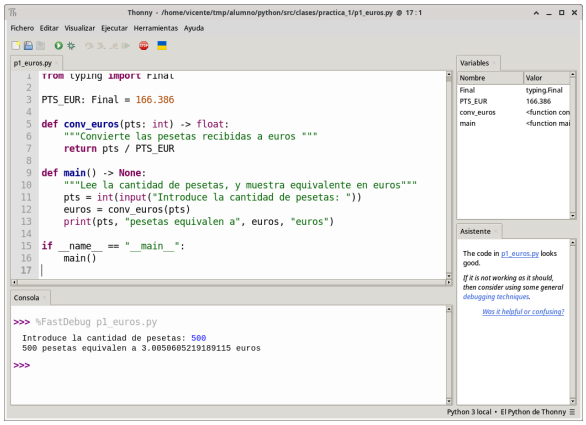

Fig.26: Final de la depuración del programa.