# B Editores de texto e IDE



Los programadores pasan mucho tiempo escribiendo, leyendo y editando código, y es esencial utilizar un editor de texto o un IDE (entorno de desarrollo integrado) para que este trabajo sea lo más eficiente posible. Un buen editor realizará tareas sencillas, como resaltar la estructura de tu código para que puedas detectar errores comunes mientras trabajas. Pero no hará tanto que te distraiga de tus pensamientos. Los editores también tienen funciones útiles como la sangría automática, marcadores para mostrar la longitud de línea adecuada y atajos de teclado para operaciones comunes.

Un *IDE* es un editor de texto con otras herramientas incluidas, como depuradores interactivos e introspección de código. Un IDE examina tu código a medida que lo introduces e intenta aprender sobre el proyecto que estás construyendo. Por ejemplo, cuando empiezas a escribir el nombre de una función, un IDE puede mostrarte todos los argumentos que acepta esa función. Este comportamiento puede ser muy útil cuando todo funciona y entiendes lo que estás viendo. Pero también puede resultar abrumador para un principiante y difícil de solucionar cuando no estás seguro de por qué tu código no funciona en el IDE.

Hoy en día, las líneas se han difuminado entre los editores de texto y los IDE. Los editores más populares tienen algunas funciones que antes eran exclusivas de los IDE. Asimismo, la mayoría de los IDE pueden configurarse para funcionar en un modo más ligero que te distraiga menos mientras trabajas, pero que te permita utilizar las funciones más avanzadas cuando las necesites.

Si ya tienes instalado un editor o IDE que te gusta, y si ya está configurado para funcionar con una versión reciente de Python instalada en tu sistema, te animo a que sigas con lo que ya conoces. Explorar diferentes editores puede ser divertido, pero también es una forma de evitar el trabajo de aprender un nuevo lenguaje.

Si aún no tienes instalado un editor o IDE, te recomiendo VS Code por varias razones:

* Es gratuito y está publicado bajo una licencia de código abierto.
* Se puede instalar en los principales sistemas operativos.
* Es fácil de usar para principiantes, pero también lo suficientemente potente como para que muchos programadores profesionales lo utilicen como editor principal.
* Encuentra las versiones de Python que tienes instaladas, y normalmente no requiere ninguna configuración para ejecutar tus primeros programas.
* Tiene un terminal integrado, por lo que tu salida aparece en la misma ventana que tu código.
* Dispone de una extensión Python que hace que el editor sea muy eficaz para escribir y mantener código Python.
* Es altamente personalizable, por lo que puedes ajustarlo a tu forma de trabajar con el código.

En este apéndice, aprenderás cómo empezar a configurar VS Code para que te funcione bien. También aprenderás algunos atajos que te permitirán trabajar de forma más eficiente. Ser un mecanógrafo rápido no es tan importante como mucha gente piensa en programación, pero entender tu editor y saber cómo utilizarlo de forma eficiente es bastante útil.

Dicho todo esto, VS Code no funciona para todo el mundo. Si por alguna razón no funciona bien en tu sistema, o si te distrae mientras trabajas, hay otros editores que pueden resultarte más atractivos. Este apéndice incluye una breve descripción de algunos de los otros editores e IDEs que deberías tener en cuenta.

## Trabajar eficazmente con VS Code

En el Capítulo 1, instalaste VS Code y añadiste también la extensión Python. Esta sección te mostrará algunas configuraciones adicionales que puedes hacer, además de atajos para trabajar eficientemente con tu código.

### Configurar VS Code

Hay algunas formas de cambiar los ajustes de configuración por defecto de VS Code. Algunos cambios pueden hacerse a través de la interfaz, y otros requerirán cambios en en los archivos de configuración. Estos cambios a veces tendrán efecto para todo lo que hagas en VS Code, mientras que otros sólo afectarán a los archivos dentro de la carpeta que contiene el archivo de configuración.

Por ejemplo, si tienes un archivo de configuración en tu carpeta *python\_work*, esos ajustes sólo afectarán a los archivos de esa carpeta (y sus subcarpetas). Esta es una buena característica, porque significa que puedes tener ajustes específicos del proyecto que anulen tus ajustes globales.

#### Uso de tabulaciones y espacios

Si utilizas una mezcla de tabuladores y espacios en tu código, puede causar problemas en tus programas que son difíciles de diagnosticar. Cuando trabajas en un archivo *.py* con la extensión Python instalada, VS Code está configurado para insertar cuatro espacios cada vez que pulsas la tecla TAB. Si sólo escribes tu propio código y tienes instalada la extensión Python, es probable que nunca tengas problemas con los tabuladores y los espacios.

Sin embargo, puede que tu instalación de VS Code no esté configurada correctamente. Además, en algún momento, puedes acabar trabajando en un archivo que sólo tenga tabuladores o una mezcla de tabuladores y espacios. Si sospechas que hay algún problema con las pestañas y los espacios, mira la barra de estado en la parte inferior de la ventana de VS Code y haz clic en **Espacios** o en **Tamaño de las pes**tañas. Aparecerá un menú desplegable que te permite cambiar entre usar tabuladores y usar espacios. También puedes cambiar el nivel de sangría por defecto y convertir toda la sangría del archivo en tabuladores o espacios.

Si estás mirando código y no estás seguro de si la sangría consiste en tabuladores o espacios, resalta varias líneas de código. Esto hará visibles los caracteres invisibles de los espacios en blanco. Cada espacio aparecerá como un punto, y cada tabulación como una flecha.

## Nota

En programación, se prefieren los espacios a las tabulaciones porque los espacios pueden ser interpretados sin ambigüedad por todas las herramientas que trabajan con un archivo de código. El ancho de los tabuladores puede ser interpretado de forma diferente por las distintas herramientas, lo que da lugar a errores que pueden ser extremadamente difíciles de diagnosticar.

#### Cambiar el tema de color

VS Code utiliza un tema oscuro por defecto. Si quieres cambiarlo, haz clic en **Archivo(Código** en la barra de menús en macOS), luego en **Preferencias** y elige **Tema** de color. Aparecerá una lista desplegable que te permitirá elegir un tema que te vaya bien.

#### Configurar el indicador de longitud de línea

La mayoría de los editores te permiten establecer un indicador visual, normalmente una línea vertical, para mostrar dónde deben terminar tus líneas. En la comunidad Python, la convención es limitar las líneas a 79 caracteres o menos.

Para configurar esta función, haz clic en **Código** y luego en **Preferencias**, y elige **Configuración**. En el cuadro de diálogo que aparece, introduce rulers. Verás una opción para Editor: Reglas; haz clic en el enlace etiquetado *Edit in settings.json*. En el archivo que aparece, añade lo siguiente a la configuración de editor.rulers:

**settings.json**

"editor.rulers": [  
 80,  
 ]

Esto añadirá una línea vertical en la ventana de edición en la posición de 80 caracteres. Puedes tener más de una línea vertical; por ejemplo, si quieres una línea adicional a 120 caracteres, el valor de tu configuración sería [80, 120]. Si no ves las líneas verticales, asegúrate de haber guardado el archivo de configuración; también es posible que tengas que salir y volver a abrir VS Code para que los cambios surtan efecto en algunos sistemas.

#### Simplificar la salida

Por defecto, VS Code muestra la salida de tus programas en una ventana de terminal incrustada. Esta salida incluye los comandos que se están utilizando para ejecutar el archivo. Para muchas situaciones, esto es ideal, pero puede distraerte más de lo que deseas cuando estás aprendiendo Python por primera vez.

Para simplificar la salida, cierra todas las pestañas que estén abiertas en VS Code y luego sal de VS Code. Inicia de nuevo VS Code y abre la carpeta que contiene los archivos Python en los que estás trabajando; puede ser simplemente la carpeta *python\_work* donde se guarda *hello\_world.py*.

Haz clic en el icono Ejecutar/Depurar (que parece un triángulo con un pequeño error) y, a continuación, haz clic en **Crear un** **archivo** ***launch.json***. Selecciona las opciones de Python en las indicaciones que aparecen. En el archivo *launch.json* que se abre, realiza el siguiente cambio:

**launch.json**

{  
 --snip--  
 "configurations": [  
 {  
 --snip--  
 "console": "internalConsole",  
 "justMyCode": true  
 }  
 ]  
}

Aquí, estamos cambiando la configuración de console de integratedTerminal a internalConsole. Después de guardar el archivo de configuración, abre un archivo *.py* como *hello\_world.py*, y ejecútalo pulsando CTRL-F5. En el panel de salida de VS Code, haz clic en **Consola de depuración** si no está ya seleccionada. Sólo deberías ver la salida de tu programa, y la salida debería actualizarse cada vez que ejecutes un programa.

## Nota

La Consola de depuración es de sólo lectura. No funcionará con archivos que utilicen la función input(), que empezarás a utilizar en el Capítulo 7. Cuando necesites ejecutar estos programas, puedes volver a cambiar la configuración de console a la predeterminada integratedTerminal, o puedes ejecutar estos programas en una ventana de terminal independiente, como se describe en "Ejecutar programas Python desde un terminal" en la página 11.

#### Explorar otras personalizaciones

Puedes personalizar VS Code de muchas maneras para ayudarte a trabajar de forma más eficiente. Para empezar a explorar las personalizaciones disponibles, haz clic en **Código** y luego en **Preferencias** y, a continuación, elige **Configuración**. Verás una lista titulada Usos comunes; haz clic en cualquiera de los subtítulos para ver algunas formas comunes de modificar tu instalación de VS Code. Tómate tu tiempo para ver si hay alguna que haga que VS Code funcione mejor para ti, ¡pero no te pierdas tanto en la configuración de tu editor que dejes para más tarde el aprendizaje del uso de Python!

### Atajos de VS Code

Todos los editores e IDEs ofrecen formas eficientes de realizar tareas comunes que todo el mundo necesita hacer al escribir y mantener código. Por ejemplo, puedes sangrar fácilmente una sola línea de código o un bloque entero de código; con la misma facilidad puedes mover un bloque de líneas hacia arriba o hacia abajo en un archivo.

Hay demasiados atajos para describirlos aquí en su totalidad. Esta sección compartirá sólo algunos que probablemente te resultarán útiles cuando escribas tus primeros archivos Python. Si acabas utilizando un editor distinto de VS Code, asegúrate de aprender a realizar estas mismas tareas de forma eficaz en el editor que hayas elegido.

#### Sangrar y Desangrar Bloques de Código

Para sangrar un bloque entero de código, resáltalo y pulsa CTRL-], o ⌘-] en macOS. Para eliminar la sangría de un bloque de código, resáltalo y pulsa CTRL-[, o ⌘-[ en macOS.

#### Comentar bloques de código

Para desactivar temporalmente un bloque de código, puedes resaltarlo y comentarlo para que Python lo ignore. Resalta la sección de código que quieras ignorar y pulsa CTRL-/, o ⌘-/ en macOS. Las líneas seleccionadas se comentarán con una almohadilla (#) sangrada al mismo nivel que la línea de código, para indicar que no son comentarios normales. Cuando quieras descomentar el bloque de código, resalta el bloque y vuelve a ejecutar el mismo comando.

#### Subir o bajar líneas

A medida que tus programas se hacen más complejos, puede que quieras mover un bloque de código hacia arriba o hacia abajo dentro de un archivo. Para ello, resalta el código que quieras mover y pulsa ALT-flecha arriba, u Opción-flecha arriba en macOS. La misma combinación de teclas con la flecha hacia abajo moverá el bloque hacia abajo en el archivo.

Si vas a mover una sola línea hacia arriba o hacia abajo, puedes pulsar en cualquier lugar de esa línea; no necesitas resaltar toda la línea para moverla.

#### Ocultar el explorador de archivos

El explorador de archivos integrado en VS Code es realmente cómodo. Sin embargo, puede distraerte cuando estás escribiendo código y puede ocupar un espacio valioso en una pantalla más pequeña. El comando CTRL-B, o ⌘-B en macOS, alterna la visibilidad del panel del explorador de archivos.

#### Encontrar atajos adicionales

Trabajar eficazmente en un entorno de edición requiere práctica, pero también intención. Cuando estés aprendiendo a trabajar con código, intenta fijarte en las cosas que haces repetidamente. Cualquier acción que realices en tu editor probablemente tenga un atajo; si haces clic en elementos del menú para realizar tareas de edición, busca los atajos para esas acciones. Si cambias con frecuencia entre el teclado y el ratón, busca los atajos de navegación que evitan que tengas que coger el ratón tan a menudo.

Puedes ver todos los atajos de teclado en VS Code haciendo clic en **Código** y luego en **Preferencias** y, a continuación, seleccionando **Atajos de** **teclado**. Puedes utilizar la barra de búsqueda para encontrar un atajo concreto, o puedes desplazarte por la lista para encontrar atajos que te ayuden a trabajar de forma más eficiente.

Recuerda que es mejor centrarse en el código en el que estás trabajando y evitar dedicar demasiado tiempo a las herramientas que utilizas.

## Otros editores de texto e IDEs

Oirás hablar y verás a gente utilizando otros editores de texto. La mayoría de ellos pueden configurarse para ayudarte del mismo modo que has personalizado VS Code. He aquí una pequeña selección de editores de texto de los que puedes oír hablar.

### IDLE

*IDLE* es un editor de texto que se incluye con Python. Es un poco menos intuitivo para trabajar que otros editores más modernos. Sin embargo, verás referencias a él en otros tutoriales dirigidos a principiantes, así que quizá quieras probarlo.

### Geany

*Geany* es un sencillo editor de texto que muestra toda tu salida en una ventana de terminal separada, lo que te ayuda a sentirte cómodo utilizando terminales. Geany tiene una interfaz muy minimalista, pero es lo suficientemente potente como para que un número significativo de programadores experimentados sigan utilizándolo.

Si VS Code te distrae demasiado y está lleno de demasiadas funciones, considera la posibilidad de utilizar Geany en su lugar.

### Sublime Text

*Sublime Text* es otro editor minimalista que deberías considerar utilizar si VS Code te parece demasiado recargado. Sublime Text tiene una interfaz realmente limpia y es conocido por funcionar bien incluso con archivos muy grandes. Es un editor que se apartará de tu camino y te permitirá centrarte en el código que estás escribiendo.

Sublime Text tiene una prueba gratuita ilimitada, pero no es gratuito ni de código abierto. Si decides que te gusta y puedes permitirte comprar una licencia completa, deberías hacerlo. La compra es un pago único; no es una suscripción de software.

### Emacs y Vim

*Emacs* y *Vim* son dos editores populares preferidos por muchos programadores experimentados, porque están diseñados para que puedas utilizarlos sin que tus manos tengan que abandonar nunca el teclado. Esto hace que escribir, leer y modificar código sea muy eficiente, una vez que aprendes cómo funciona el editor. También significa que ambos editores tienen una curva de aprendizaje bastante pronunciada. Vim está incluido en la mayoría de las máquinas Linux y macOS, y tanto Emacs como Vim pueden ejecutarse completamente dentro de un terminal. Por esta razón, a menudo se utilizan para escribir código en servidores a través de sesiones de terminal remotas.

Los programadores suelen recomendarte que los pruebes, pero muchos programadores expertos olvidan lo mucho que los nuevos programadores ya están intentando aprender. Es bueno que conozcas estos editores, pero deberías dejar de utilizarlos hasta que te sientas cómodo trabajando con código en un editor más fácil de usar que te permita centrarte en aprender a programar, en lugar de aprender a utilizar un editor.

### PyCharm

*PyCharm* es un IDE popular entre los programadores de Python porque se creó para trabajar específicamente con Python. La versión completa requiere una suscripción de pago, pero también está disponible una versión gratuita llamada PyCharm Community Edition, y muchos desarrolladores la encuentran útil.

Si pruebas PyCharm, ten en cuenta que, por defecto, configura un entorno aislado para cada uno de tus proyectos. Esto suele ser bueno, pero puede dar lugar a comportamientos inesperados si no entiendes lo que está haciendo por ti.

### Cuadernos Jupyter

*Jupyter Notebook* es un tipo de herramienta diferente de los editores de texto o IDE tradicionales, en el sentido de que es una aplicación web construida principalmente con bloques; cada bloque es un bloque de código o un bloque de texto. Los bloques de texto se representan en Markdown, por lo que puedes incluir un formato sencillo en tus bloques de texto.

Los Cuadernos Jupyter se desarrollaron para apoyar el uso de Python en aplicaciones científicas, pero desde entonces se han ampliado para ser útiles en una amplia variedad de situaciones. En lugar de limitarte a escribir comentarios dentro de un archivo *.py*, puedes escribir texto claro con un formato sencillo, como encabezados, listas con viñetas e hipervínculos entre secciones de código. Cada bloque de código puede ejecutarse independientemente, lo que te permite probar pequeñas partes de tu programa, o puedes ejecutar todos los bloques de código a la vez. Cada bloque de código tiene su propia área de salida, y puedes activar o desactivar las áreas de salida según necesites.

Los Cuadernos Jupyter pueden resultar confusos a veces debido a las interacciones entre las distintas celdas. Si defines una función en una celda, esa función está disponible también para otras celdas. Esto es beneficioso la mayor parte del tiempo, pero puede resultar confuso en cuadernos más largos y si no entiendes bien cómo funciona el entorno de los Cuadernos.

Si realizas algún trabajo científico o centrado en datos en Python, es casi seguro que en algún momento verás Cuadernos Jupyter.