

TOPICS

Editing



IN THIS ARTICLE

Autocomplete and IntelliSense



(<https://vscode.dev/github/microsoft/vscode-docs/blob/main/docs/python/editing.md>)

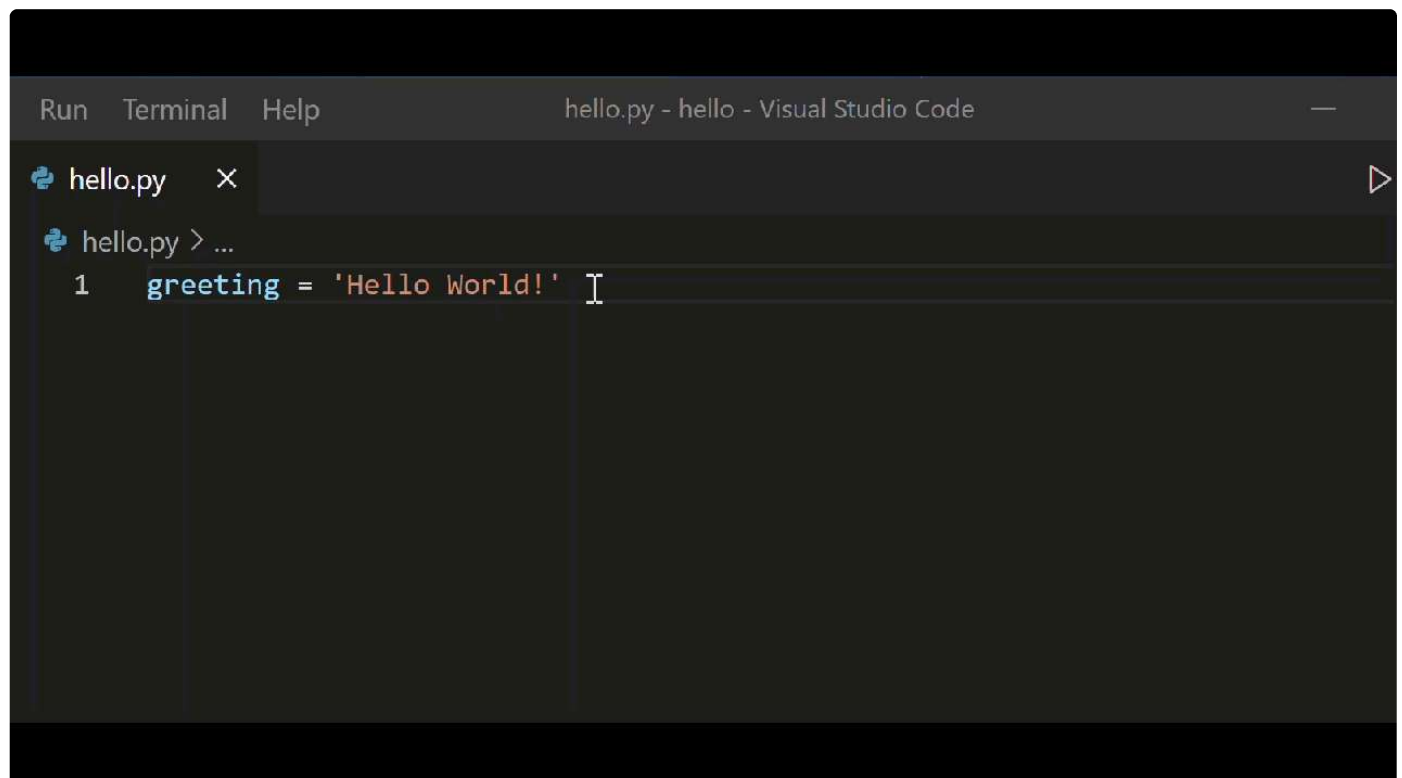
## Edición de Python en Visual Studio Code

Visual Studio Code es una eficaz herramienta de edición para el código fuente de Python. El editor incluye varias funciones para ayudarlo a ser productivo al escribir código. Para obtener más información sobre la edición en Visual Studio Code, vea [Edición básica \(/docs/editing/codebasics\)](https://docs.python.org/3/tutorial/interact.html) y [navegación de código \(/docs/editing/editingevolved\)](https://docs.python.org/3/tutorial/interact.html).

En esta descripción general, describiremos las funciones de edición específicas proporcionadas por la [extensión de Python](https://marketplace.visualstudio.com/items?itemName=ms-python.python) (<https://marketplace.visualstudio.com/items?itemName=ms-python.python>), incluidos los pasos sobre cómo personalizar estas funciones a través de la [configuración \(/docs/configure/settings\)](https://docs.python.org/3/tutorial/interact.html) del usuario y del espacio de trabajo.

### Autocompletar e IntelliSense

IntelliSense es un término general para las características de edición de código relacionadas con la finalización de código. Tómese un momento para ver el ejemplo a continuación. Cuando se escribe **print**, observe cómo IntelliSense rellena las opciones de autocompletado. El usuario también recibe una lista de opciones cuando comienza a escribir la variable llamada **saludo**.



Autocompletar e IntelliSense se proporcionan para todos los archivos de la carpeta de trabajo actual. También están disponibles para paquetes de Python que se instalan en ubicaciones estándar.

[Pylance](https://marketplace.visualstudio.com/items?itemName=ms-python.vscode-pylance) (<https://marketplace.visualstudio.com/items?itemName=ms-python.vscode-pylance>) es el servidor de lenguaje predeterminado para Python en VS Code y se instala junto con la extensión de Python para proporcionar características de IntelliSense.

Pylance se basa en la herramienta de verificación de tipos estáticos [Pyright](https://github.com/microsoft/pyright) (<https://github.com/microsoft/pyright>) de Microsoft, que aprovecha los [códigos auxiliares de tipo](https://typing.readthedocs.io/en/latest/source/stubs.html) (<https://typing.readthedocs.io/en/latest/source/stubs.html>) (archivos) y la inferencia de tipos diferidos para proporcionar una experiencia de desarrollo de alto rendimiento. `.pyi`

Para obtener más información sobre IntelliSense en general, consulte [IntelliSense \(/docs/editing/intellisense\)](/docs/editing/intellisense).

**Sugerencia:** Consulte la [extensión IntelliCode para VS Code](https://go.microsoft.com/fwlink/?linkid=2006060) (<https://go.microsoft.com/fwlink/?linkid=2006060>). IntelliCode proporciona un conjunto de funcionalidades asistidas por IA para IntelliSense en Python, como inferir las finalizaciones automáticas más relevantes en función del contexto de código actual. Para obtener más información, consulte las [preguntas más frecuentes sobre IntelliCode para VS Code](https://learn.microsoft.com/visualstudio/intellicode/intellicode-visual-studio-code) (<https://learn.microsoft.com/visualstudio/intellicode/intellicode-visual-studio-code>).

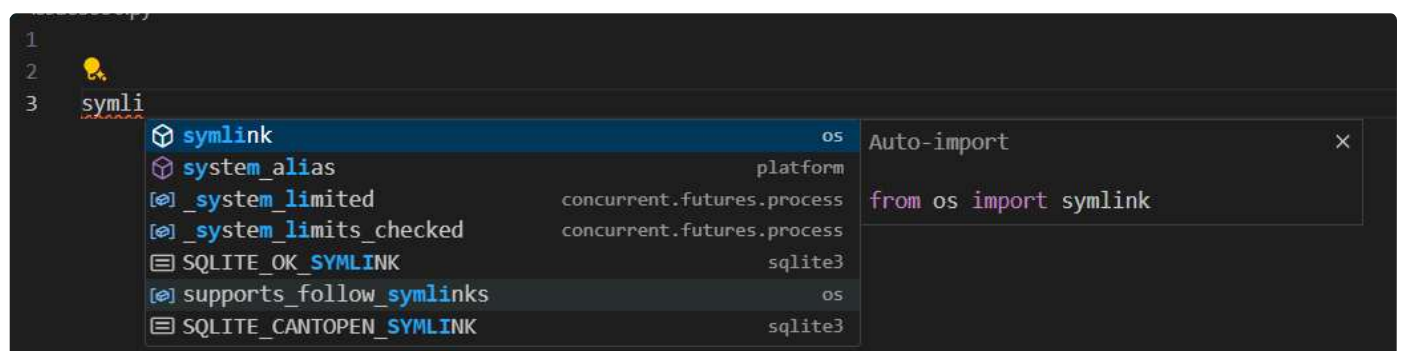
## Personalización del comportamiento de IntelliSense

Habilitar el conjunto completo de características de IntelliSense de forma predeterminada podría terminar haciendo que la experiencia de desarrollo se sienta más lenta, por lo que la extensión de Python habilita un conjunto mínimo de características que le permiten ser productivo sin dejar de tener una experiencia de rendimiento. Sin embargo, puede personalizar el comportamiento del motor de análisis a su gusto a través de múltiples configuraciones.

## Habilitar importaciones automáticas

Pylance ofrece sugerencias de importación automática para módulos en su espacio de trabajo y para paquetes que instaló en su entorno. A medida que escribe en el editor, es posible que reciba sugerencias de finalización. Cuando acepta la sugerencia, la importación automática agrega automáticamente la instrucción de importación correspondiente al archivo.

Puede habilitar las importaciones automáticas configurándolas en su configuración. De forma predeterminada, las importaciones automáticas están deshabilitadas. `python.analysis.autoImportCompletions true`



## Habilitación de IntelliSense para ubicaciones de paquetes personalizados

Para habilitar IntelliSense para paquetes que se instalan en ubicaciones no estándar, agregue esas ubicaciones a la colección del archivo (la colección predeterminada está vacía). Por ejemplo, es posible que tengas Google App Engine instalado en ubicaciones personalizadas, especificadas en si usas Flask. En este caso, especificaría esas ubicaciones de la siguiente manera: `python.analysis.extraPaths settings.json app.yaml`

**Windows:**

```
"python.analysis.extraPaths": [  
  "C:/Program Files (x86)/Google/google_appengine",  
  "C:/Program Files (x86)/Google/google_appengine/lib/flask-0.12"]
```

[Copy](#)

macOS/Linux:

```
"python.analysis.extraPaths": [  
  "~/local/lib/Google/google_appengine",  
  "~/local/lib/Google/google_appengine/lib/flask-0.12" ]
```

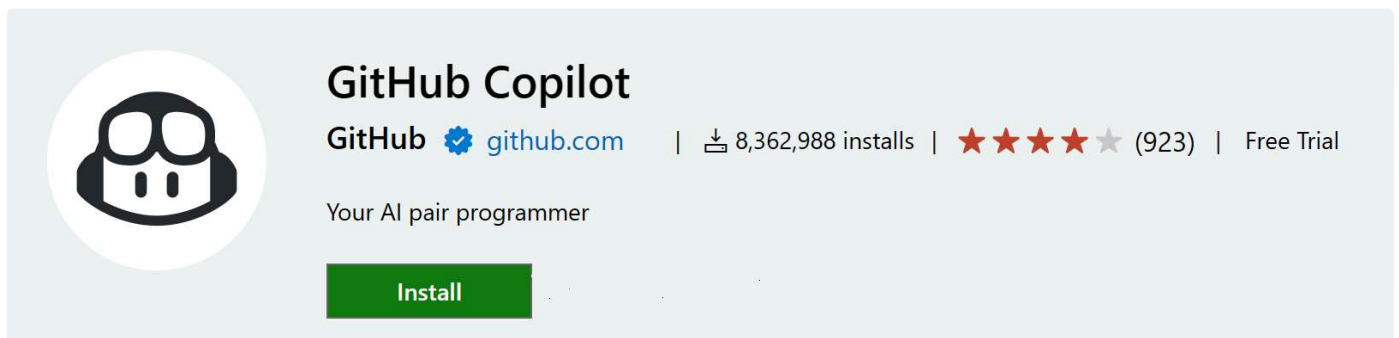
[Copy](#)

Para obtener la lista completa de controles IntelliSense disponibles, puede hacer referencia a la [configuración de análisis de código](#) (/docs/python/settings-reference#\_code-analysis-settings) de extensión de Python y a la configuración de [autocompletar](#) (/docs/python/settings-reference#\_autocomplete-settings).

También puede personalizar el comportamiento general de autocompletar e IntelliSense, incluso deshabilitar las características por completo. Puede obtener más información en [Personalización de IntelliSense](#) (/docs/editing/intellisense#\_customizing-intellisense).

## Mejore las finalizaciones con IA

[GitHub Copilot](https://copilot.github.com/) (<https://copilot.github.com/>) es una herramienta de finalización de código impulsada por IA que lo ayuda a escribir código de manera más rápida e inteligente. Puede usar la [extensión de GitHub Copilot](#) (<https://marketplace.visualstudio.com/items?itemName=GitHub.copilot>) en VS Code para generar código o para aprender del código que genera.



(<https://marketplace.visualstudio.com/items?itemName=GitHub.copilot>)

GitHub Copilot proporciona sugerencias para numerosos lenguajes y una amplia variedad de marcos, y funciona especialmente bien para Python, JavaScript, TypeScript, Ruby, Go, C # y C ++.

Puede obtener más información sobre cómo empezar a utilizar Copilot en la [documentación de Copilot](#) (/docs/editor/github-copilot).

## Navegación

Mientras edita, puede hacer clic con el botón derecho en diferentes identificadores para aprovechar varios comandos convenientes

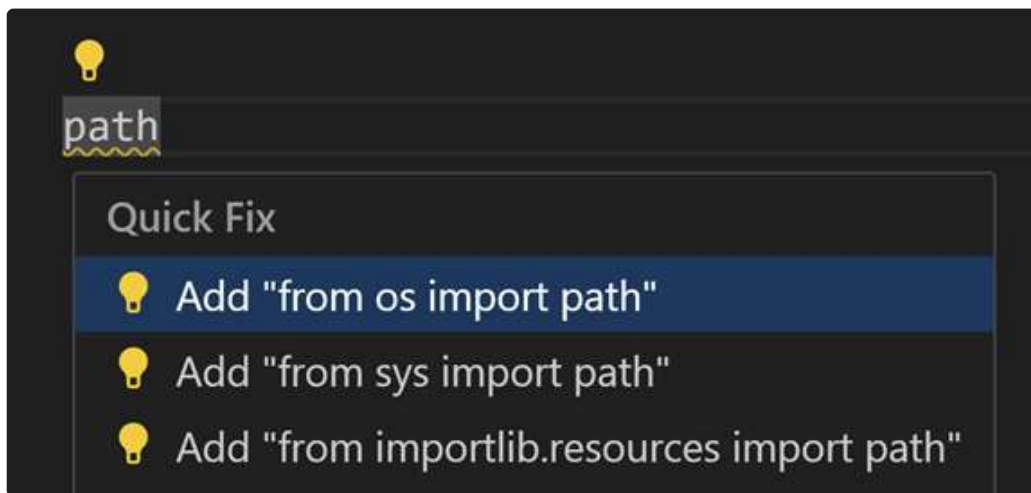
- **Ir a definición** ( F12 ) salta del código al código que define un objeto. Este comando es útil cuando se trabaja con bibliotecas.
- **Peek Definition** ( Alt+F12 ), es similar, pero muestra la definición directamente en el editor (haciendo espacio en la ventana del editor para evitar oscurecer cualquier código). Presione **Escape** para cerrar la ventana Ver o use la **x** en la esquina superior derecha.

- **Ir a Declaración** salta al punto en el que se declara la variable u otro objeto en el código.
- **La declaración de Peek** es similar, pero muestra la declaración directamente en el editor. Nuevamente, use `Escape` o la `x` en la esquina superior derecha para cerrar la ventana Peek.

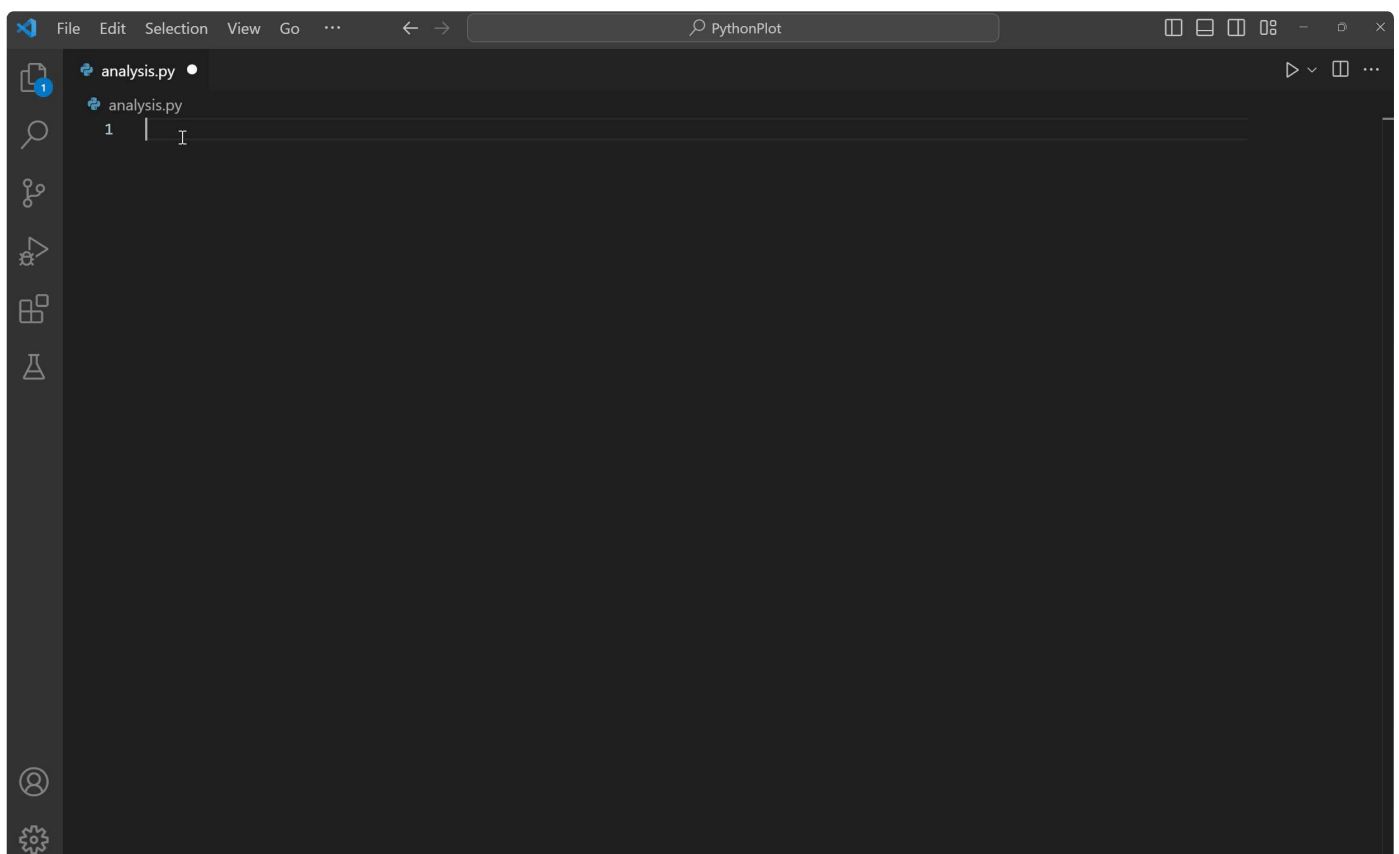
## Soluciones rápidas

### Agregar importación

Cuando se utiliza Pylance, la solución rápida `add import` le permite completar rápidamente las instrucciones de importación para los módulos instalados en su entorno. A medida que comienza a escribir un nombre de paquete en el editor, hay una acción de código disponible para completar automáticamente la línea de código fuente. Pase el cursor sobre el texto (marcado con un subrayado ondulado) y seleccione la bombilla de acción de código. A continuación, puede seleccionar de la lista de posibles importaciones.



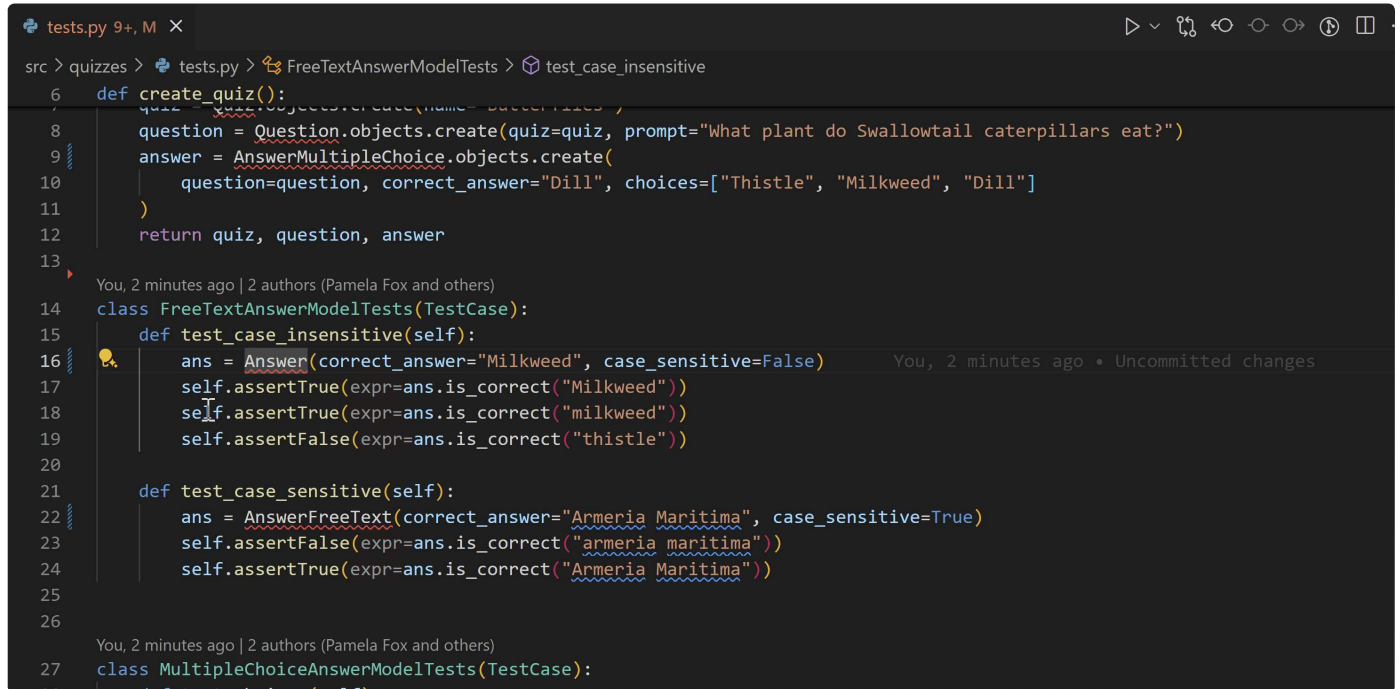
Esta acción de código también reconoce algunas de las abreviaturas populares para los siguientes paquetes comunes de Python: como np, como tf, como pd, como plt, como mpl, como m, como spio y como sp, como pn y como hv. numpy tensorflow pandas matplotlib.pyplot matplotlib math scipy.io scipy panel holoviews



La lista de sugerencias de importación muestra las 3 opciones de importación de alta confianza, priorizadas en función de: importaciones utilizadas más recientemente, símbolos del mismo módulo, símbolos de la biblioteca estándar, símbolos de módulos de usuario, símbolos de paquetes de terceros y, finalmente, clasificación por módulo y nombre de símbolo.

## Buscar coincidencias de importación adicionales

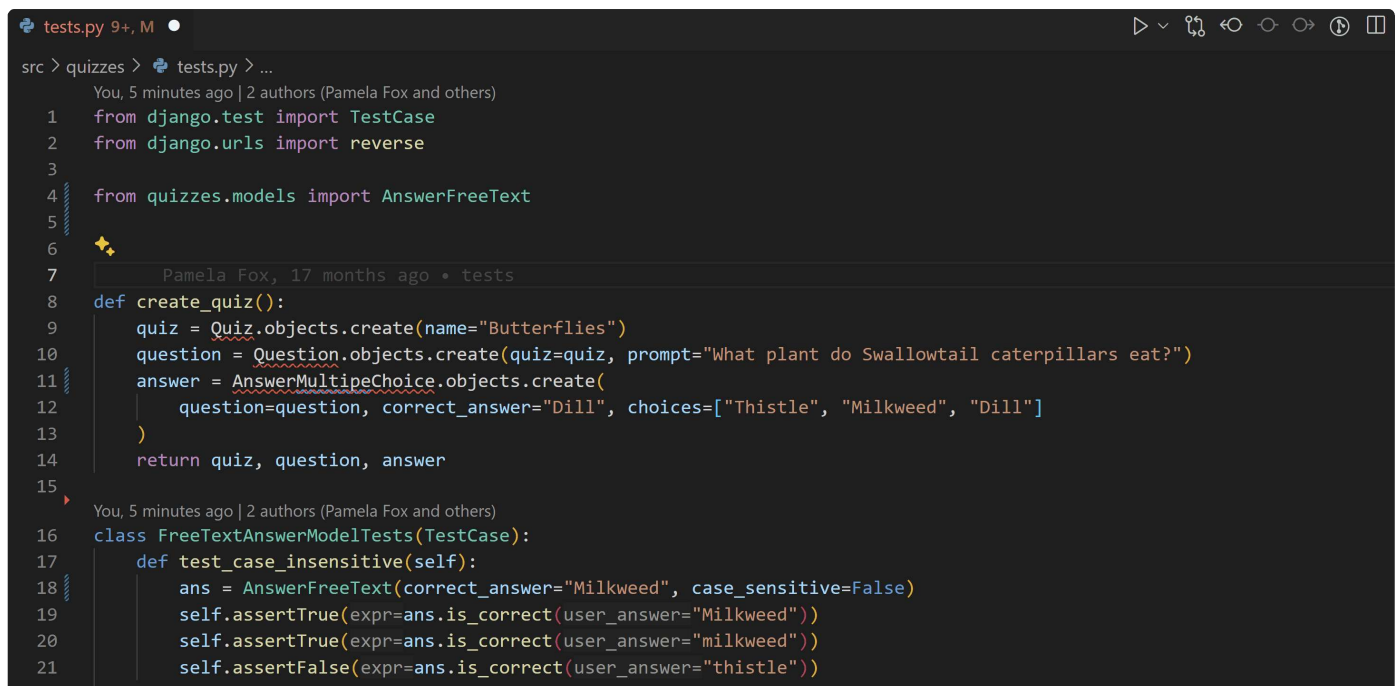
De forma predeterminada, la solución rápida de importación de agregar solo muestra 3 opciones de importación de alta confianza. Si no enumeran lo que está buscando, puede usar Pylance **Search para coincidencias de importación adicionales** Solución rápida para errores de importación faltantes. Esta solución rápida muestra un menú de selección rápida que le permite buscar opciones de importación que coincidan con el prefijo del símbolo de importación que falta.



```
tests.py 9+, M X
src > quizzes > tests.py > FreeTextAnswerModelTests > test_case_insensitive
6 def create_quiz():
7     quiz = Quiz.objects.create(name="Butterflies")
8     question = Question.objects.create(quiz=quiz, prompt="What plant do Swallowtail caterpillars eat?")
9     answer = AnswerMultipleChoice.objects.create(
10         question=question, correct_answer="Dill", choices=["Thistle", "Milkweed", "Dill"]
11     )
12     return quiz, question, answer
13
14 You, 2 minutes ago | 2 authors (Pamela Fox and others)
15 class FreeTextAnswerModelTests(TestCase):
16     def test_case_insensitive(self):
17         ans = Answer(correct_answer="Milkweed", case_sensitive=False)
18         self.assertTrue(expr=ans.is_correct("Milkweed"))
19         self.assertTrue(expr=ans.is_correct("milkweed"))
20         self.assertFalse(expr=ans.is_correct("thistle"))
21
22     def test_case_sensitive(self):
23         ans = AnswerFreeText(correct_answer="Armeria Maritima", case_sensitive=True)
24         self.assertFalse(expr=ans.is_correct("armeria maritima"))
25         self.assertTrue(expr=ans.is_correct("Armeria Maritima"))
26
27 You, 2 minutes ago | 2 authors (Pamela Fox and others)
28 class MultipleChoiceAnswerModelTests(TestCase):
29     def test_case_insensitive(self):
```

## Cambiar ortografía

Pylance muestra la corrección rápida de **cambio ortográfico** en variables no resueltas o diagnósticos de importaciones faltantes cuando es probable que se deban a errores tipográficos. Esta acción de código sugiere la ortografía correcta del símbolo, en función de las coincidencias más cercanas encontradas en el espacio de trabajo.



```
tests.py 9+, M
src > quizzes > tests.py > ...
1 from django.test import TestCase
2 from django.urls import reverse
3
4 from quizzes.models import AnswerFreeText
5
6
7 Pamela Fox, 17 months ago • tests
8 def create_quiz():
9     quiz = Quiz.objects.create(name="Butterflies")
10    question = Question.objects.create(quiz=quiz, prompt="What plant do Swallowtail caterpillars eat?")
11    answer = AnswerMultipleChoice.objects.create(
12        question=question, correct_answer="Dill", choices=["Thistle", "Milkweed", "Dill"]
13    )
14    return quiz, question, answer
15
16 You, 5 minutes ago | 2 authors (Pamela Fox and others)
17 class FreeTextAnswerModelTests(TestCase):
18     def test_case_insensitive(self):
19         ans = AnswerFreeText(correct_answer="Milkweed", case_sensitive=False)
20         self.assertTrue(expr=ans.is_correct(user_answer="Milkweed"))
21         self.assertTrue(expr=ans.is_correct(user_answer="milkweed"))
22         self.assertFalse(expr=ans.is_correct(user_answer="thistle"))
23
```

**Nota:** Para los símbolos de usuario, estas correcciones rápidas sugerirán las importaciones solo desde los archivos donde están definidos. No se admiten sugerencias de importación de archivos en los que los símbolos de usuario son externos o importados.

También tenga en cuenta que para los símbolos provenientes de paquetes instalados (generalmente ubicados en la carpeta de su entorno de Python), solo los definidos en la carpeta raíz del paquete, como en su archivo, son sugeridos por estas Correcciones rápidas. Puede personalizar este comportamiento para paquetes específicos a través de la configuración, pero tenga en cuenta que puede afectar el rendimiento de Pylance. `site-packages __init__.py python.analysis.packageIndexDepths`

## Refactorizaciones

The Python extension adds the following refactoring functionalities via the Pylance extension: **Extract Variable**, **Extract Method**, **Rename Module**, **Move Symbol** and **Implement All Inherited Abstract Classes**. It also supports extensions that implement additional refactoring features, such as **Sort Imports**.

### Extract Variable

Extracts all similar occurrences of the selected text within the current scope, and replaces it with a new variable.

You can invoke this command by selecting the line of code you wish to extract as a variable. Then select the light-bulb that is displayed next to it.

```
logo_path = mask_path.parent / "logo.png"

if not logo_path.exists():
    wc = generate_wordcloud(parse_content(url), np.array(Image.open(mask_path)))

    generate_image(logo_path, wc, np.array(Image.open(mask_path)))

return "/static/images/logo.png"
```

### Método de extracción

Extrae todas las apariciones similares de la expresión o bloque seleccionado dentro del ámbito actual y lo reemplaza por una llamada de método.

Puede invocar este comando seleccionando las líneas de código que desea extraer como método. Luego seleccione la bombilla que se muestra junto a ella.

```
logo_path = mask_path.parent / "logo.png"

if not logo_path.exists():
    parsed_content = parse_content(url)
    wc = generate_wordcloud(parsed_content, np.array(Image.open(mask_path)))
    generate_image(logo_path, wc, np.array(Image.open(mask_path)))

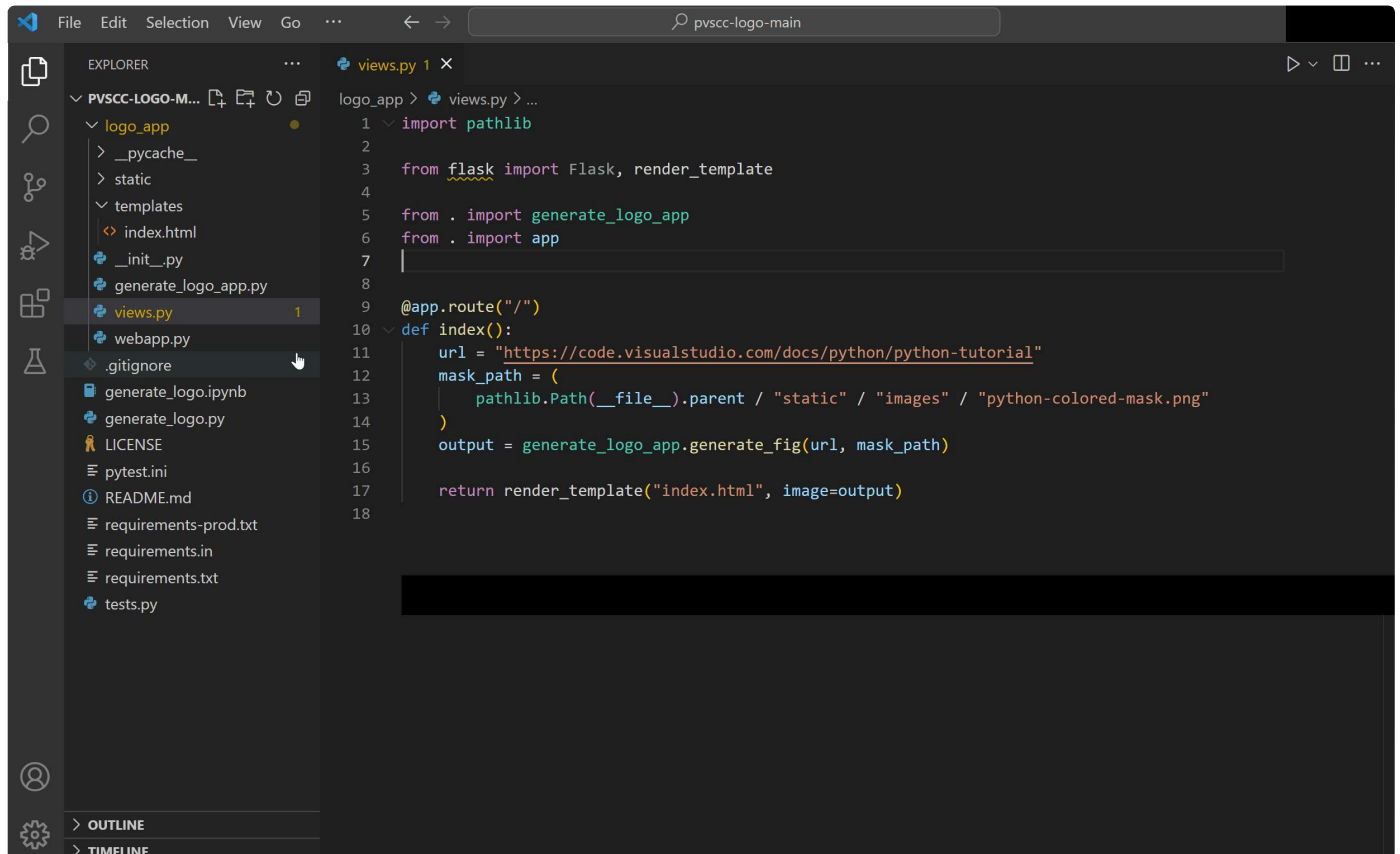
return "/static/images/logo.png"
```



## Cambiar nombre del módulo

Después de cambiar el nombre de un archivo/módulo de Python, Pylance puede encontrar todas las instancias que deban actualizarse y proporcionarle una vista previa de todos los cambios.

Para personalizar qué referencias deben actualizarse, puede activar o desactivar las casillas de verificación en la línea o desde el nivel de archivo en **Refactor Preview**. Una vez que haya realizado las selecciones, puede seleccionar **Aplicar refactorización** o **Descartar refactorización**.



## Mover símbolo

La extensión Pylance ofrece dos acciones de código para simplificar el proceso de mover símbolos a diferentes archivos:

- **Mover símbolo a...:** muestra un selector de archivos para seleccionar el archivo de destino al que se va a mover el símbolo.
- **Mover símbolo a nuevo archivo:** crea un nuevo archivo con el nombre del símbolo, ubicado en el mismo directorio que el archivo de origen donde se invocó la acción de código.

Puede acceder a estas acciones de código pasando el cursor sobre el símbolo que desea mover y luego seleccionando la bombilla que aparece junto a la acción deseada. Alternativamente, puede hacer clic derecho en el símbolo y seleccionar **Refactorizar...** en el menú contextual.

```
answer_tests.py U  display_tests.py U X
src > quizzes > tests > display_tests.py > DisplayQuizViewTests > test_quiz_redirects
4  from django.test import TestCase
5  from django.urls import reverse
6
7
8  class DisplayQuizViewTests(TestCase):
9      def test_quiz_404(self):
10         url = reverse("quizzes:display_quiz", args=(12,))
11         response = self.client.get(url)
12         self.assertEqual(response.status_code, 404)
13
14     def test_quiz_redirects(self):
15         quiz, question, _ = create_quiz()
16         url = reverse("quizzes:display_quiz", args=(quiz.pk,))
17         response = self.client.get(url)
18         self.assertRedirects(response, reverse("quizzes:display_question", args=(quiz.pk, question.pk)))
```

## Implementar todas las clases abstractas heredadas

En Python, las clases abstractas sirven como "planos" para otras clases y ayudan a construir código modular y reutilizable al promover una estructura clara y requisitos para que las subclases se adhieran. Para definir una clase abstracta en Python, puede crear una clase que herede de la clase en el módulo y anotar sus métodos con el decorador. A continuación, puede crear nuevas clases que hereden de esta clase abstracta y definir una implementación para los métodos

```
base.ABC abc @abstractmethod
```

Pylance ofrece una acción de código para simplificar el proceso de creación de estas clases. Cuando define una nueva clase que hereda de una abstracta, ahora puede usar la acción de código **Implementar todas las clases abstractas heredadas** para implementar automáticamente todos los métodos abstractos y propiedades de la clase primaria:



```
from abc import ABC, abstractmethod
```

```
class Product(ABC):
```

```
    @abstractmethod
```

```
    def get_price(self):
```

```
        pass
```

```
    @abstractmethod
```

```
    def get_name(self):
```

```
        pass
```

```
    @abstractmethod
```

```
    def get_category(self):
```

```
        pass
```



```
class Book(Product):
```

```
    ~
```

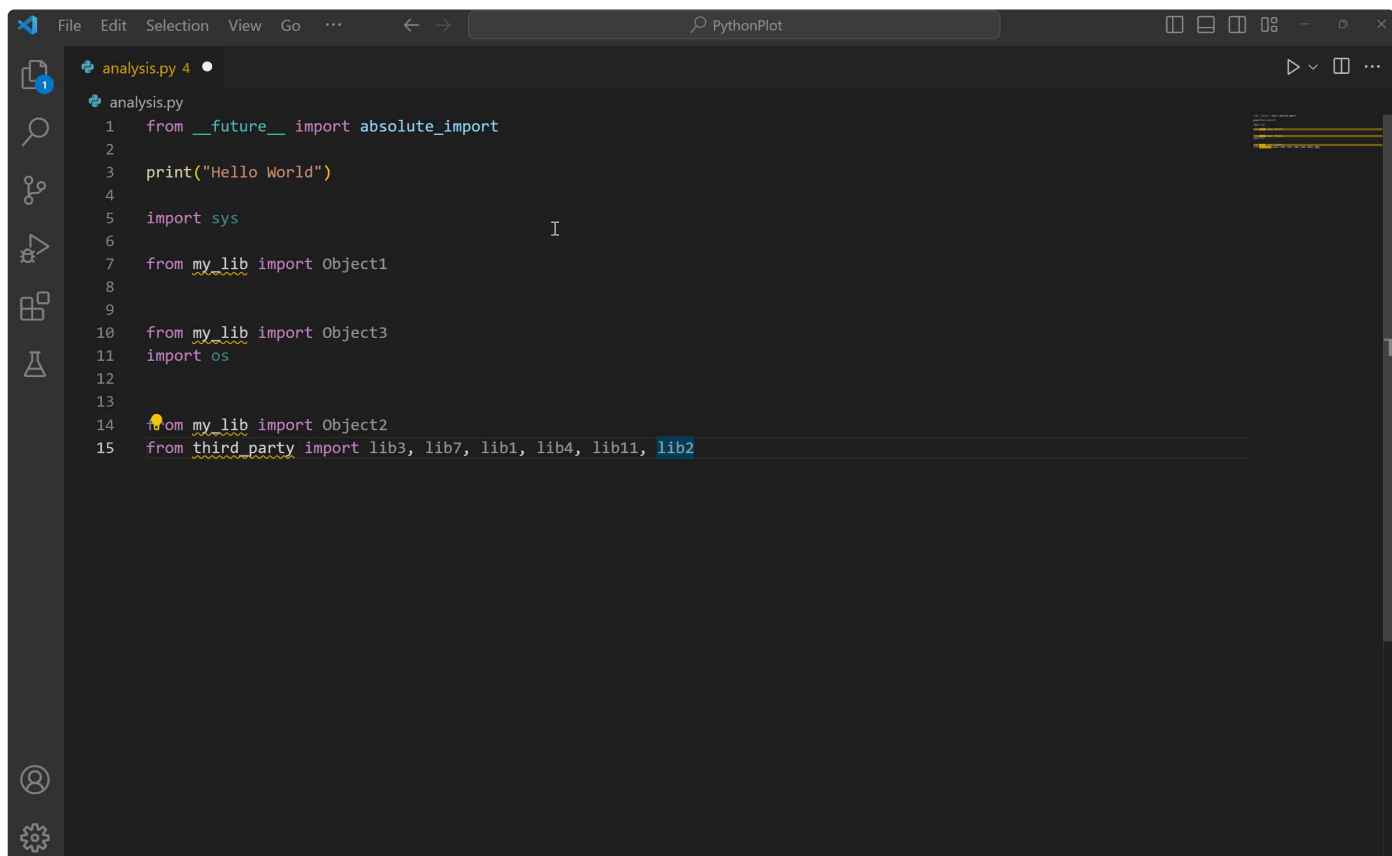
I

## Ordenar importaciones

La extensión de Python admite extensiones como [isort](https://marketplace.visualstudio.com/items?itemName=ms-python.isort) (<https://marketplace.visualstudio.com/items?itemName=ms-python.isort>) y [Ruff](https://marketplace.visualstudio.com/items?itemName=charliermarsh.ruff) (<https://marketplace.visualstudio.com/items?itemName=charliermarsh.ruff>) que implementan la funcionalidad **Ordenar importaciones**. Este comando consolida importaciones específicas del mismo módulo en una sola instrucción y organiza las instrucciones en orden alfabético. `import import`

Puede invocar esto instalando una extensión que admita la ordenación de importaciones, luego abriendo la paleta de comandos ( `Ctrl+Mayús+P` ) y ejecutando **Organizar importaciones**.

**Consejo:** Puede asignar un método abreviado de teclado al comando. `editor.action.organizeImports`



## Solución de problemas

Para obtener ayuda con problemas comunes de edición de IntelliSense y Python, consulte la tabla siguiente:

Problema	Causa	Solución
Pylance solo ofrece opciones de símbolos de nivel superior al agregar importaciones.	De forma predeterminada, solo se indexan los módulos de nivel superior (depth=1). Por ejemplo, puede verlo como una sugerencia, pero no de forma predeterminada. <code>import matplotlib import matplotlib.pyplot</code>	Intente aumentar la profundidad a la que Pylance puede indexar sus bibliotecas instaladas a través del archivo <code>.pyproject.toml</code> . Verifique <a href="#">la configuración de análisis de código (/docs/python/settings-reference#_code-analysis-settings)</a> . <code>python.analysis.packageIndexDepths</code>
Pylance no agrega automáticamente las importaciones faltantes	Es posible que la configuración de finalización de importación automática esté deshabilitada.	Marca la <a href="#">sección Habilitar importaciones automáticas (/docs/python/editing#_customize-intellisense-behavior)</a> .

Problema	Causa	Solución
Las importaciones automáticas están habilitadas, pero Pylance no importa automáticamente los símbolos definidos en otros archivos del espacio de trabajo.	Los símbolos definidos por el usuario (aquellos que no provienen de paquetes o bibliotecas instalados) solo se importan automáticamente si ya se han utilizado en archivos abiertos en el editor. De lo contrario, solo estarán disponibles a través de la <a href="#">solución rápida de agregar importaciones</a> ( <a href="#">/docs/python/editing#_quick-fixes</a> ).	Utilice la solución rápida para agregar importaciones o asegúrese de abrir primero los archivos relevantes en su espacio de trabajo.
Pylance parece lento o consume demasiada memoria cuando se trabaja en un espacio de trabajo grande.	El análisis de Pylance se realiza en todos los archivos presentes en un espacio de trabajo determinado.	Si hay subcarpetas que sabe que se pueden excluir del análisis de Pylance, puede agregar sus rutas a la configuración. Como alternativa, puede intentar configurar para deshabilitar el indexador de Pylance ( <b>Nota:</b> esto también afectará la experiencia de finalizaciones e importaciones automáticas. Obtenga más información sobre la indexación en <a href="#">la configuración de análisis de código</a> ( <a href="#">/docs/python/settings-reference#_code-analysis-settings</a> )). <code>python.analysis.exclude python.analysis.indexing false</code>
No puede instalar un módulo personalizado en el proyecto de Python.	El módulo personalizado se encuentra en una ubicación no estándar (no se instala mediante pip).	Agregue la ubicación a la configuración y reinicie VS Code. <code>python.autoComplete.extraPaths</code>

## Diagnóstico de Pylance

Pylance proporciona de forma predeterminada diagnósticos para archivos de Python en el panel Problemas.

La siguiente lista son algunos de los diagnósticos más comunes proporcionados por Pylance y cómo solucionarlos.

### importResolveSourceFailure

Este error se produce cuando Pylance puede encontrar códigos auxiliares de tipo para el paquete importado, pero no puede encontrar el paquete en sí. Esto puede suceder cuando el paquete que está intentando importar no está instalado en el entorno de Python seleccionado.

#### Cómo solucionarlo

- Si el paquete ya está instalado en un intérprete o kernel diferente, [seleccione el intérprete correcto](#) ([/docs/python/environments#\\_select-and-activate-an-environment](#)).
- Si el paquete no está instalado, puede instalarlo ejecutando el siguiente comando en una terminal activada: `. python -m pip install {package_name}`

### importResolveFailure

Este error ocurre cuando Pylance no puede encontrar el paquete o módulo que está importando, ni sus códigos auxiliares de tipo.

#### Cómo solucionarlo

- Si está importando un módulo, asegúrese de que exista en su espacio de trabajo o en una ubicación que esté incluida en la configuración. `python.autoComplete.extraPaths`
- Si está importando un paquete que no está instalado, puede instalarlo ejecutando el siguiente comando en un terminal activado: `. python -m pip install {package_name}`
- Si está importando un paquete que ya está instalado en un intérprete o kernel diferente, [seleccione el intérprete correcto \(/docs/python/environments#\\_select-and-activate-an-environment\)](#).
- Si está trabajando con una instalación editable y actualmente está configurada para usar enlaces de importación, considere cambiar al uso de archivos que solo contengan rutas de archivo, para mejorar la compatibilidad y garantizar un comportamiento de importación más fluido. Obtenga más información en la [documentación de Pyright \(https://microsoft.github.io/pyright/#/import-resolution?id=editable-installs\)](#). `.pth`

## importCycleDetected

Este error se produce cuando Pylance detecta una dependencia circular entre dos o más módulos.

### Cómo solucionarlo

Intente reordenar las instrucciones de importación para romper la dependencia circular.

---

La gravedad de los diagnósticos de Pylance se puede personalizar a través de la configuración. Consulte la [referencia de configuración \(/docs/python/settings-reference\)](#) para obtener más información. `python.analysis.diagnosticSeverityOverrides`

## Pasos siguientes

- [Linting \(/docs/python/linting\)](#): habilite, configure y aplique varios linters de Python.
- [Depuración \(/docs/python/debugging\)](#): aprenda a depurar Python tanto local como remotamente.
- [Pruebas \(/docs/python/testing\)](#): configure entornos de prueba y detecte, ejecute y depure pruebas.
- [Edición básica \(/docs/editing/codebasics\)](#): obtenga información sobre el eficaz editor de VS Code.
- [Navegación de código \(/docs/editing/editingevolved\)](#): muévase rápidamente por el código fuente.
- [IntelliSense \(/docs/editing/intellisense\)](#): obtenga información sobre las características de IntelliSense.
- [Soporte técnico de Jupyter \(/docs/datascience/jupyter-notebooks\)](#): aprenda a empezar a trabajar con Jupyter Notebooks.
- [Plantilla de extensión de Python \(/api/advanced-topics/python-extension-template\)](#): cree una extensión para integrar sus herramientas favoritas de Python.

¿Te ha resultado útil esta documentación?

☐ Sí ☐ No


---


07/09/2025

 [RSS Feed\(/feed.xml\)](#)  [Ask questions\(https://stackoverflow.com/questions/tagged/vscode\)](#)

 [Follow @code\(https://go.microsoft.com/fwlink/?LinkID=533687\)](#)

 [Request features\(https://go.microsoft.com/fwlink/?LinkID=533482\)](#)

 [Report issues\(https://www.github.com/Microsoft/vscode/issues\)](https://www.github.com/Microsoft/vscode/issues)

 [Watch videos\(https://www.youtube.com/channel/UCs5Y5\\_7XK8HLDX0SLNwkd3w\)](https://www.youtube.com/channel/UCs5Y5_7XK8HLDX0SLNwkd3w)



[\(https://www.microsoft.com\)](https://www.microsoft.com)

[\(https://go.microsoft.com/fwlink/?LinkID=533687\)](https://go.microsoft.com/fwlink/?LinkID=533687)



[\(https://github.com/microsoft/vscode\)](https://github.com/microsoft/vscode)



[\(https://www.youtube.com/@code\)](https://www.youtube.com/@code)

Apoyo (<https://support.serviceshub.microsoft.com/supportforbusiness/create?sapId=d66407ed-3967-b000-4cfb-2c318cad363d>)

Privacidad (<https://go.microsoft.com/fwlink/?LinkId=521839>)      Términos de uso (<https://www.microsoft.com/legal/terms-of-use>)

Licencia (/License)