

Try [agent mode \(vscode://GitHub.Copilot-Chat/chat?mode=agent&referrer=vscode-agentbanner\)](https://vscode.dev/GitHub.Copilot-Chat/chat?mode=agent&referrer=vscode-agentbanner) in VS Code! ✕

TOPICS

Tutorial ▾

IN THIS ARTICLE

Prerequisites ▾

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

Getting Started with Python in VS Code

In this tutorial, you will learn how to use Python 3 in Visual Studio Code to create, run, and debug a Python "Roll a dice!" application, work with virtual environments, use packages, and more! By using the [Python extension \(https://marketplace.visualstudio.com/items?itemName=ms-python.python\)](https://marketplace.visualstudio.com/items?itemName=ms-python.python), you turn VS Code into a great, lightweight Python editor.

If you are new to programming, check out the [Visual Studio Code for Education - Introduction to Python \(https://vscodeedu.com/courses/intro-to-python\)](https://vscodeedu.com/courses/intro-to-python) course. This course offers a comprehensive introduction to Python, featuring structured modules in a ready-to-code browser-based development environment.

To gain a deeper understanding of the Python language, you can explore any of the [programming tutorials \(https://wiki.python.org/moin/BeginnersGuide/Programmers\)](https://wiki.python.org/moin/BeginnersGuide/Programmers) listed on python.org within the context of VS Code.

For a Data Science focused tutorial with Python, check out our [Data Science section \(/docs/datascience/data-science-tutorial\)](/docs/datascience/data-science-tutorial).

Prerequisites

To successfully complete this tutorial, you need to first set up your Python development environment. Specifically, this tutorial requires:

- [Python 3 \(/docs/python/python-tutorial#_install-a-python-interpreter\)](/docs/python/python-tutorial#_install-a-python-interpreter)
- [VS Code \(https://code.visualstudio.com/\)](https://code.visualstudio.com/)
- [VS Code Python extension \(https://marketplace.visualstudio.com/items?itemName=ms-python.python\)](https://marketplace.visualstudio.com/items?itemName=ms-python.python) (For additional details on installing extensions, see [Extension Marketplace \(/docs/configure/extensions/extension-marketplace\)](/docs/configure/extensions/extension-marketplace))

Install a Python interpreter

Along with the Python extension, you need to install a Python interpreter. Which interpreter you use is dependent on your specific needs, but some guidance is provided below.

Windows

Install Python from [python.org](https://www.python.org/downloads/) (<https://www.python.org/downloads/>). Use the **Download Python** button that appears first on the page to download the latest version.

Note: If you don't have admin access, an additional option for installing Python on Windows is to use the Microsoft Store. The Microsoft Store provides installs of [supported Python versions](https://apps.microsoft.com/store/search?publisher=Python%20Software%20Foundation) (<https://apps.microsoft.com/store/search?publisher=Python%20Software%20Foundation>).

For additional information about using Python on Windows, see [Using Python on Windows at Python.org](https://docs.python.org/3.9/using/windows.html) (<https://docs.python.org/3.9/using/windows.html>).

macOS

The system install of Python on macOS is not supported. Instead, a package management system like Homebrew (<https://brew.sh/>) is recommended. To install Python using Homebrew on macOS use `brew install python3` at the Terminal prompt.

Note: On macOS, make sure the location of your VS Code installation is included in your PATH environment variable. See [these setup instructions \(/docs/setup/mac#_launching-from-the-command-line\)](#) for more information.

Linux

The built-in Python 3 installation on Linux works well, but to install other Python packages you must install `pip` with `get-pip.py` (<https://pip.pypa.io/en/stable/installation/#get-pip-py>).

Other options

- **Data Science:** If your primary purpose for using Python is Data Science, then you might consider a download from [Anaconda](https://www.anaconda.com/download/) (<https://www.anaconda.com/download/>). Anaconda provides not just a Python interpreter, but many useful libraries and tools for data science.
- **Windows Subsystem for Linux:** If you are working on Windows and want a Linux environment for working with Python, the [Windows Subsystem for Linux](https://learn.microsoft.com/windows/wsl/about) (<https://learn.microsoft.com/windows/wsl/about>) (WSL) is an option for you. If you choose this option, you'll also want to install the [WSL](#)

[extension \(https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl\)](https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl).

For more information about using WSL with VS Code, see [VS Code Remote Development \(/docs/remote/remote-overview\)](/docs/remote/remote-overview) or try the [Working in WSL tutorial \(/docs/remote/wsl-tutorial\)](/docs/remote/wsl-tutorial), which will walk you through setting up WSL, installing Python, and creating a Hello World application running in WSL.

Note: To verify that you've installed Python successfully on your machine, run one of the following commands (depending on your operating system):

Linux/macOS: open a Terminal Window and type the following command:

```
python3 --version
```

[Copy](#)

Windows: open a command prompt and run the following command:

```
py -3 --version
```

[Copy](#)

If the installation was successful, the output window should show the version of Python that you installed. Alternatively, you can use the `py -0` command in the VS Code integrated terminal to view the versions of python installed on your machine. The default interpreter is identified by an asterisk (*).

Start VS Code in a workspace folder

By starting VS Code in a folder, that folder becomes your "workspace".

Using a command prompt or terminal, create an empty folder called "hello", navigate into it, and open VS Code (code) in that folder (.) by entering the following commands:

```
mkdir hello  
cd hello  
code .
```


[Copy](#)

Note: If you're using an Anaconda distribution, be sure to use an Anaconda command prompt.

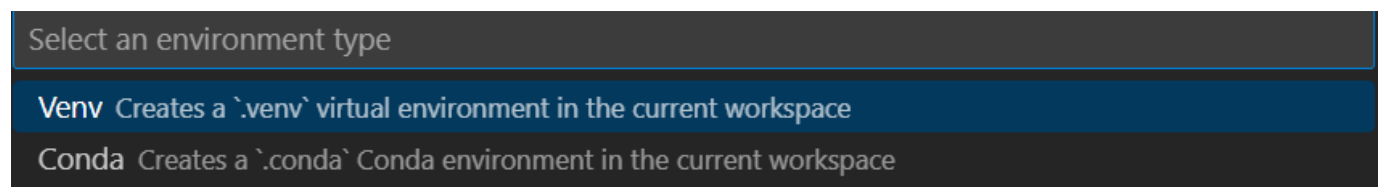
Alternately, you can create a folder through the operating system UI, then use VS Code's **File > Open Folder** to open the project folder.

Create a virtual environment

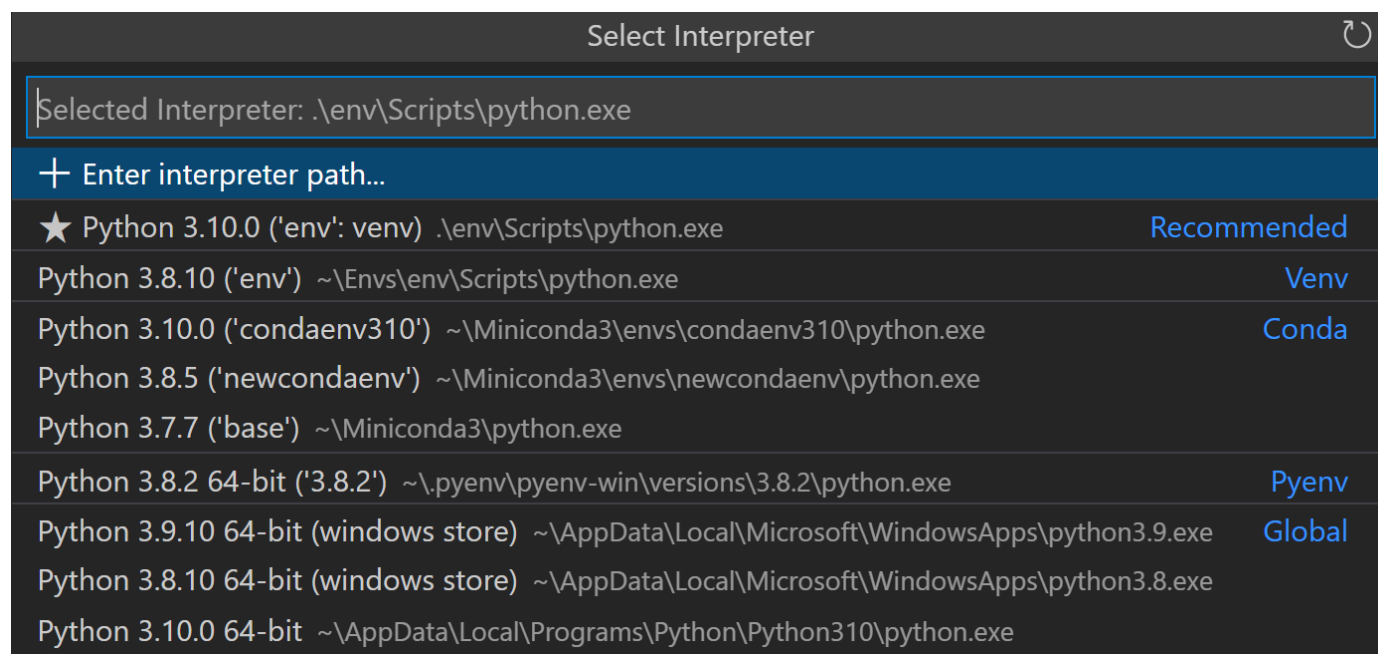
A best practice among Python developers is to use a project-specific `virtual environment`. Once you activate that environment, any packages you then install are isolated from other environments, including the global interpreter environment, reducing many complications that can arise from conflicting package versions. You can create non-global environments in VS Code using Venv or Anaconda with **Python: Create Environment**.

Open the Command Palette (P), start typing the **Python: Create Environment** command to search, and then select the command.

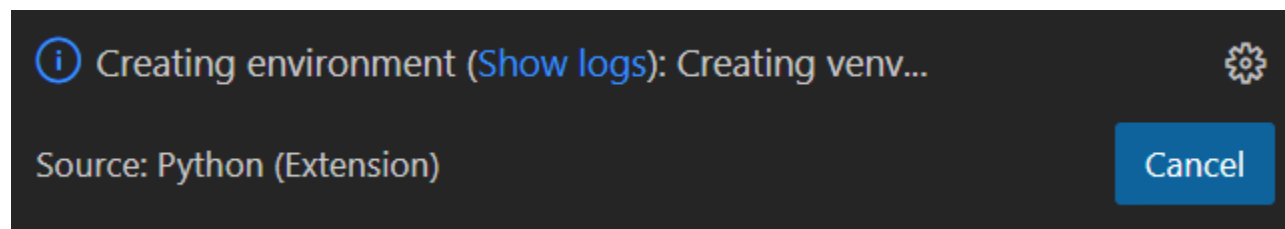
The command presents a list of environment types, Venv or Conda. For this example, select **Venv**.



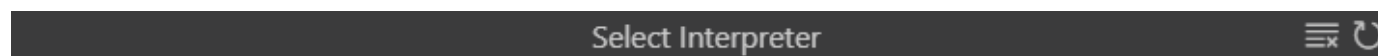
The command then presents a list of interpreters that can be used for your project. Select the interpreter you installed at the beginning of the tutorial.

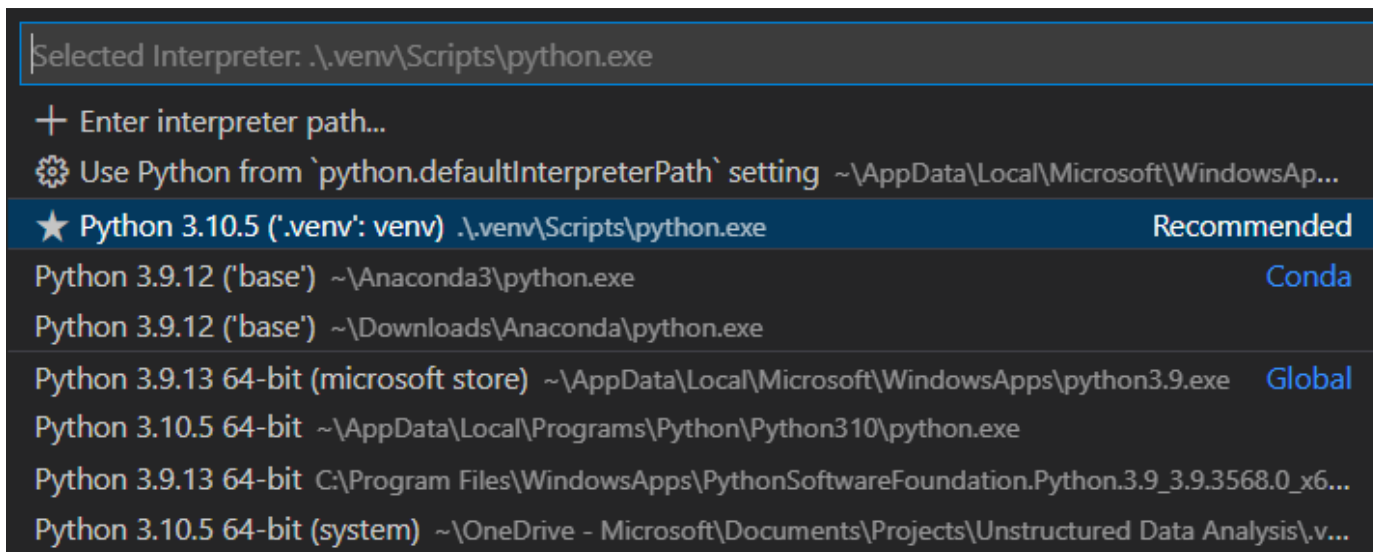


After selecting the interpreter, a notification will show the progress of the environment creation and the environment folder (`/.venv`) will appear in your workspace.



Ensure your new environment is selected by using the **Python: Select Interpreter** command from the **Command Palette**.

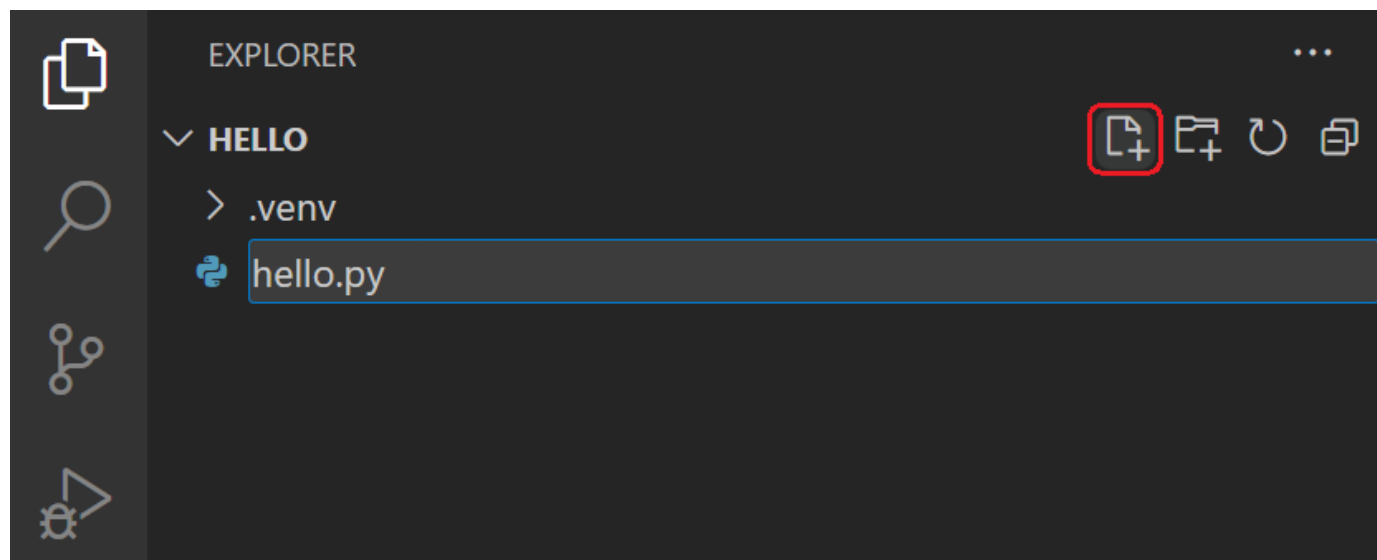




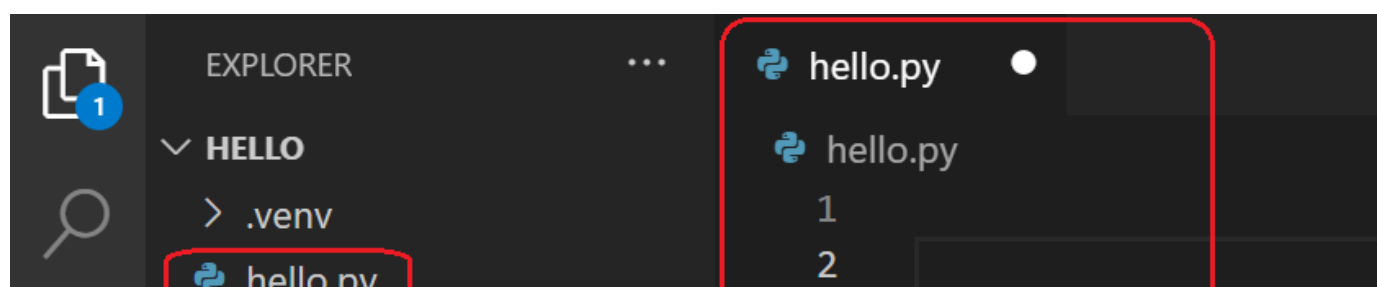
Note: For additional information about virtual environments, or if you run into an error in the environment creation process, see [Environments \(/docs/python/environments#_creating-environments\)](/docs/python/environments#_creating-environments).

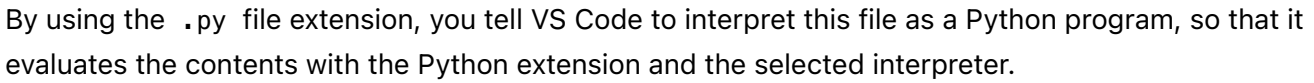
Create a Python source code file

From the File Explorer toolbar, select the **New File** button on the `hello` folder:



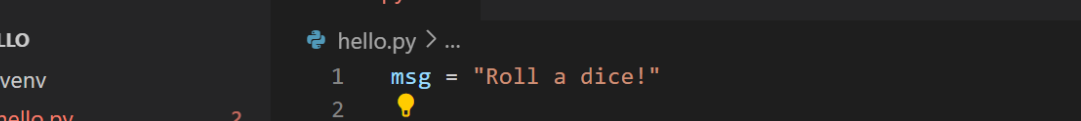
Name the file `hello.py`, and VS Code will automatically open it in the editor:





Now that you have a code file in your Workspace, enter the following source code in `hello.py`:

Copy



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file named 'hello.py' under a folder named 'HELLO'. The main editor area shows the contents of 'hello.py', which includes a docstring and a function definition. The function 'roll_dice' is currently being edited, and a code completion dropdown menu is visible, suggesting various Python built-in functions and exceptions that start with 'pr'.

```
"""
A simple program that simulates rolling a 6-sided die.

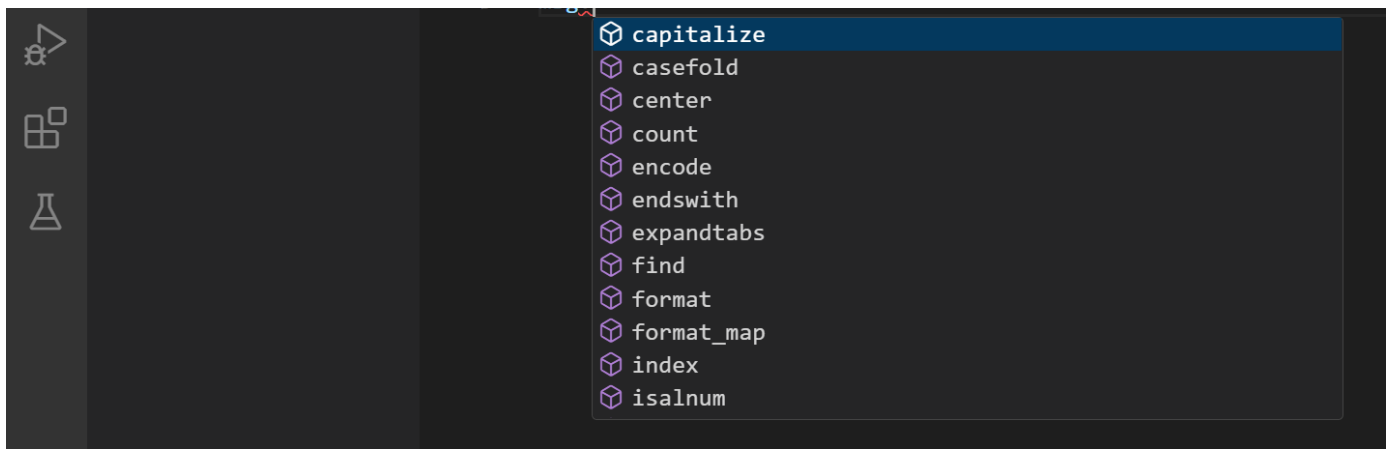
This program uses the random module to generate a random integer
between 1 and 6, representing the outcome of a dice roll.
"""

import random


def roll_dice():
    """Roll a 6-sided die and return the outcome."""
    return random.randint(1, 6)


if __name__ == '__main__':
    # Roll the die once
    outcome = roll_dice()
    print(f"You rolled a {outcome}!")
```

A screenshot of the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'HELLO' with a subdirectory '.venv' and a file 'hello.py'. The Editor pane on the right shows the contents of 'hello.py', which includes a docstring and a function definition. A lightbulb icon is visible next to the function definition, indicating a suggestion or error.

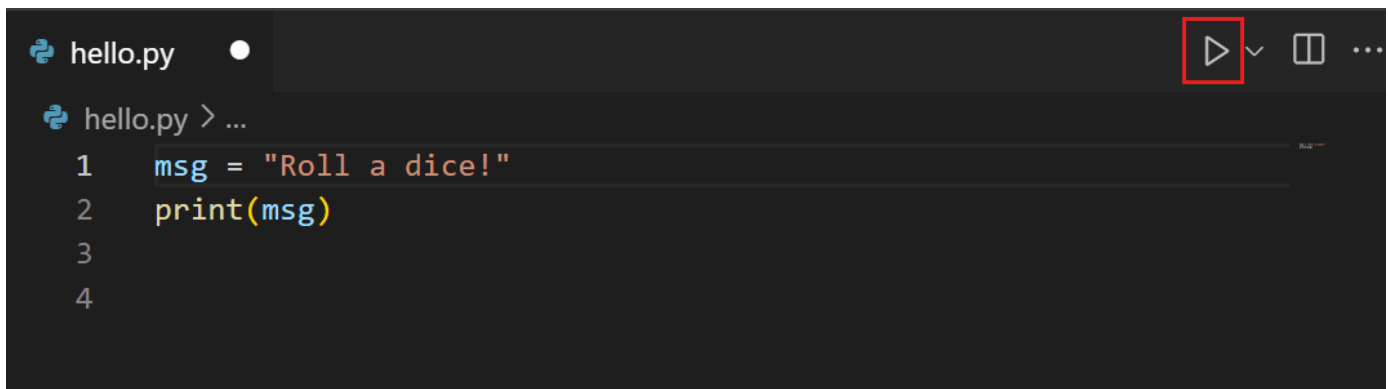


Finally, save the file (`⌘S`). At this point, you're ready to run your first Python file in VS Code.

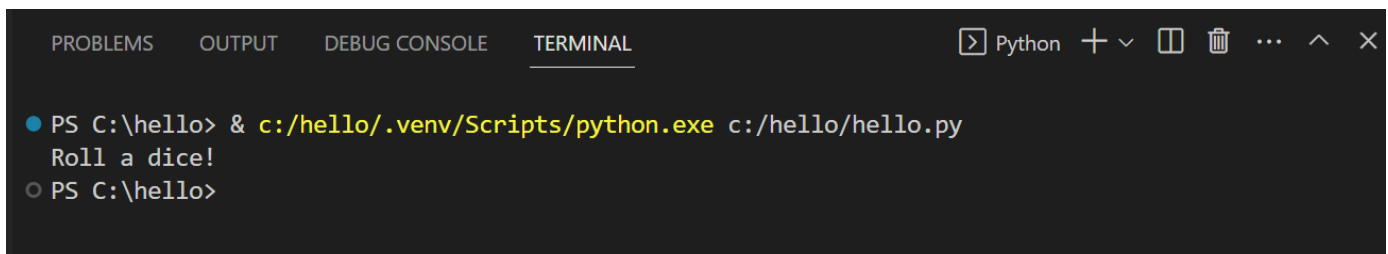
For full details on editing, formatting, and refactoring, see [Editing code \(/docs/python/editing\)](/docs/python/editing). The Python extension also has full support for [Linting \(/docs/python/linting\)](/docs/python/linting).

Run Python code

Click the **Run Python File** play button in the top-right side of the editor.



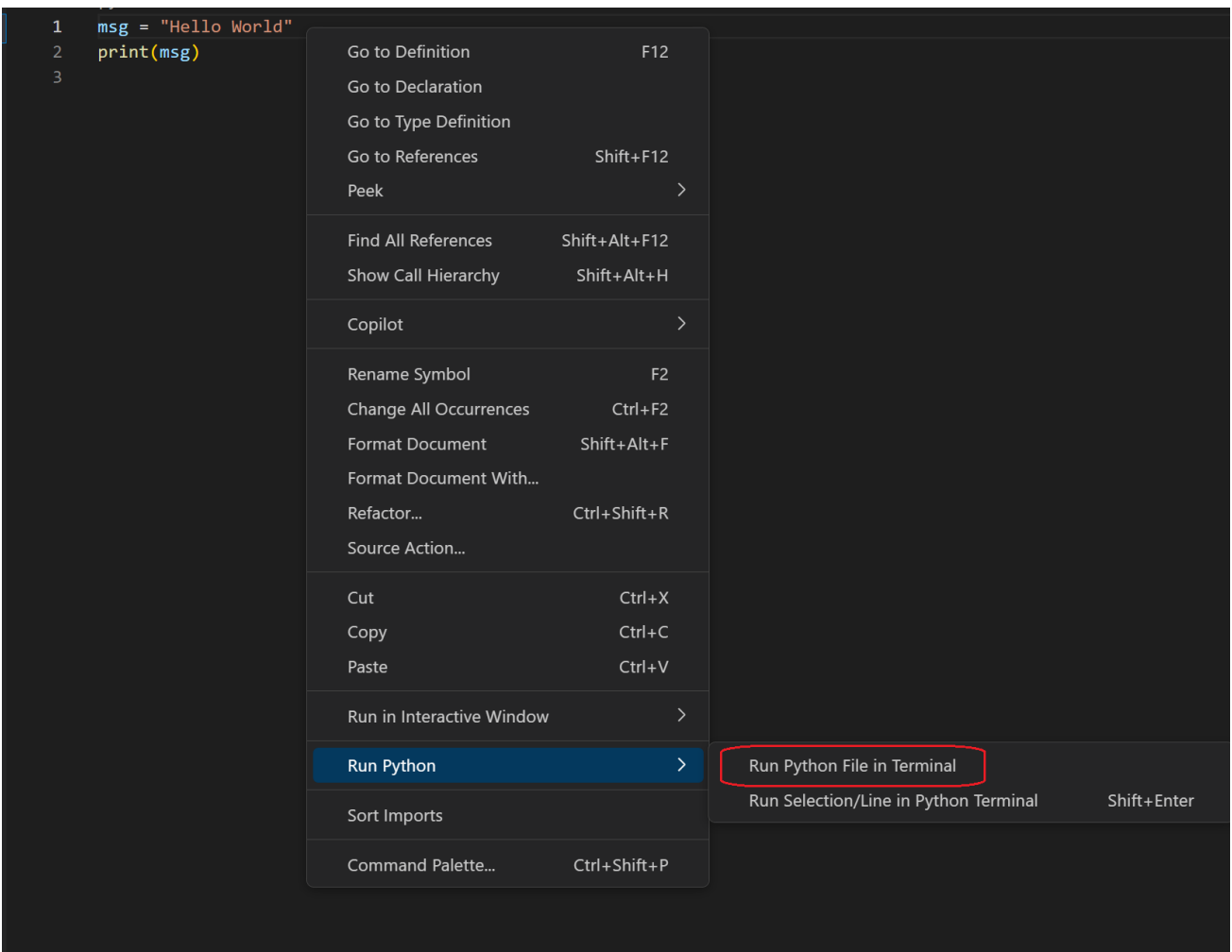
The button opens a terminal panel in which your Python interpreter is automatically activated, then runs `python3 hello.py` (macOS/Linux) or `python hello.py` (Windows):



There are three other ways you can run Python code within VS Code:

- 1 Right-click anywhere in the editor window and select **Run Python > Run Python File in Terminal** (which saves the file automatically):





- 2 Select one or more lines, then press `Shift+Enter` or right-click and select **Run Python > Run Selection/Line in Python Terminal**. Alternatively, you can activate Smart Send using `Shift+Enter` without a selection and the Python extension will send the smallest runnable block of code near where your cursor is placed to the terminal. This command is convenient for testing just a part of a file.

Note: If you prefer to send code at the particular line your cursor is placed, you can turn off Smart Send by setting `python.REPL.enableREPLSmartSend : "false"` in your **User** settings.

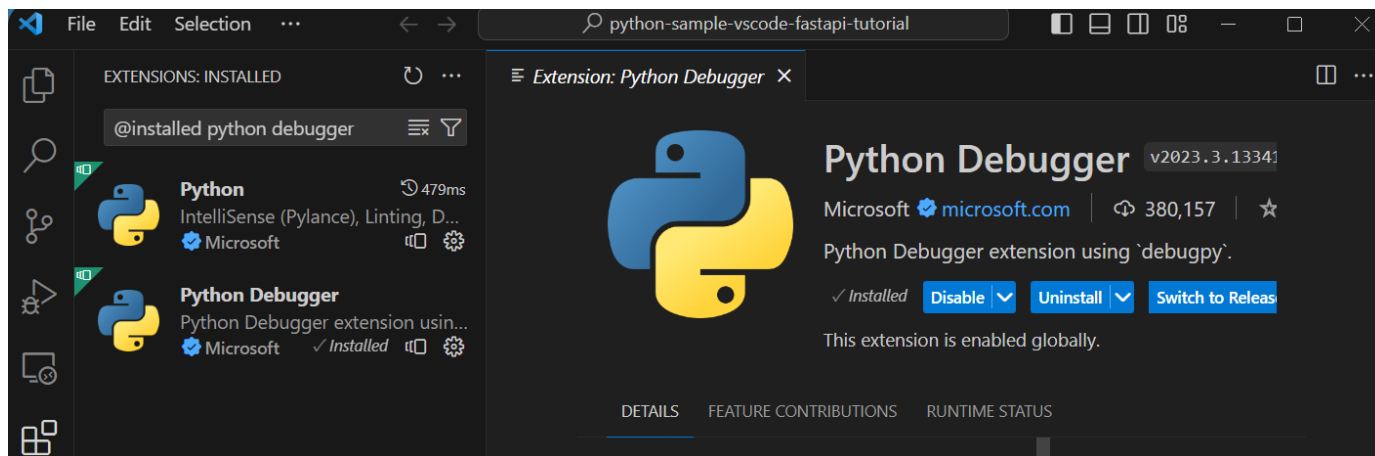
- 3 From the Command Palette (`⇧⌘P`), select the **Python: Start Terminal REPL** command to open a REPL terminal (notated by `>>>`) for the currently selected Python interpreter. In the REPL, you can then enter and run lines of code one at a time.

Congrats, you just ran your first Python code in Visual Studio Code!

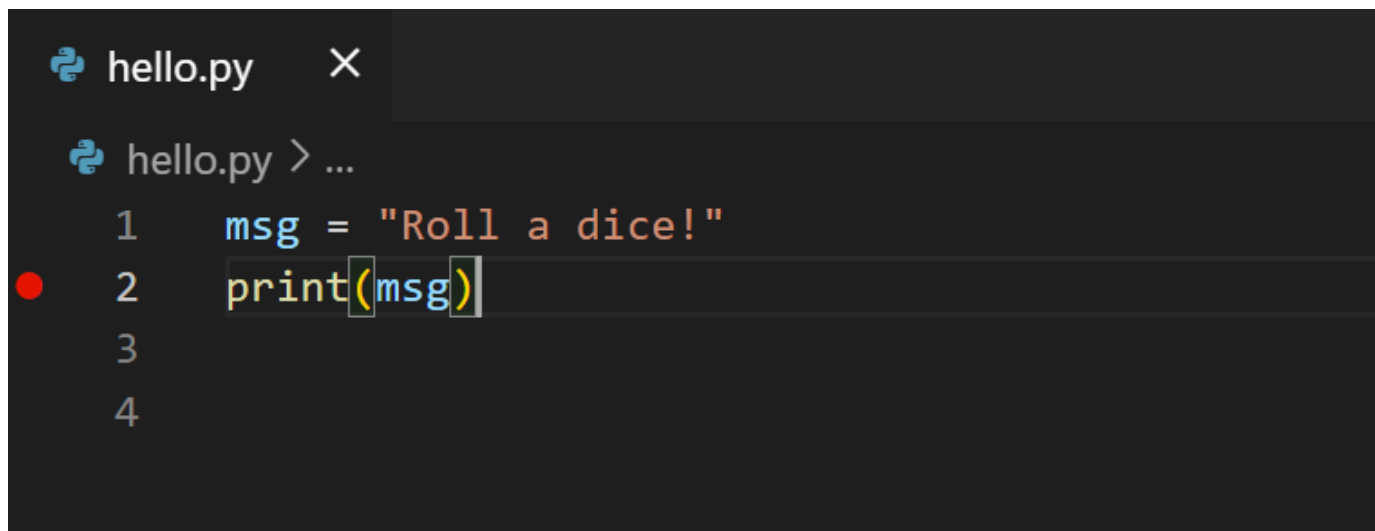
Configure and run the debugger

Let's now try debugging our Python program. Debugging support is provided by the [Python Debugger](#)

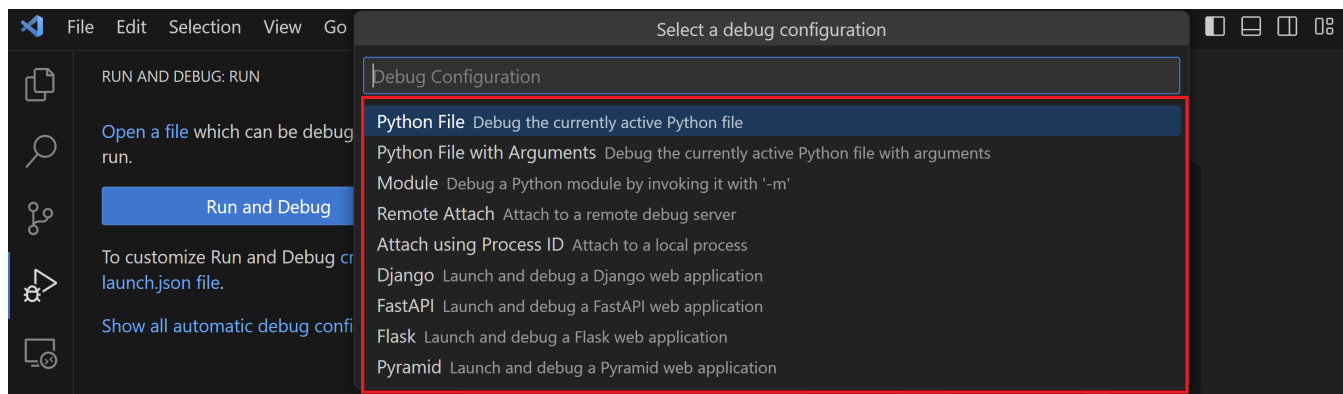
extension (<https://marketplace.visualstudio.com/items?itemName=ms-python.debugpy>), which is automatically installed with the Python extension. To ensure it has been installed correctly, open the **Extensions** view (⇧⌘X) and search for `@installed python debugger`. You should see the Python Debugger extension listed in the results.



Next, set a breakpoint on line 2 of `hello.py` by placing the cursor on the `print` call and pressing `F9`. Alternately, click in the editor's left gutter, next to the line numbers. When you set a breakpoint, a red circle appears in the gutter.



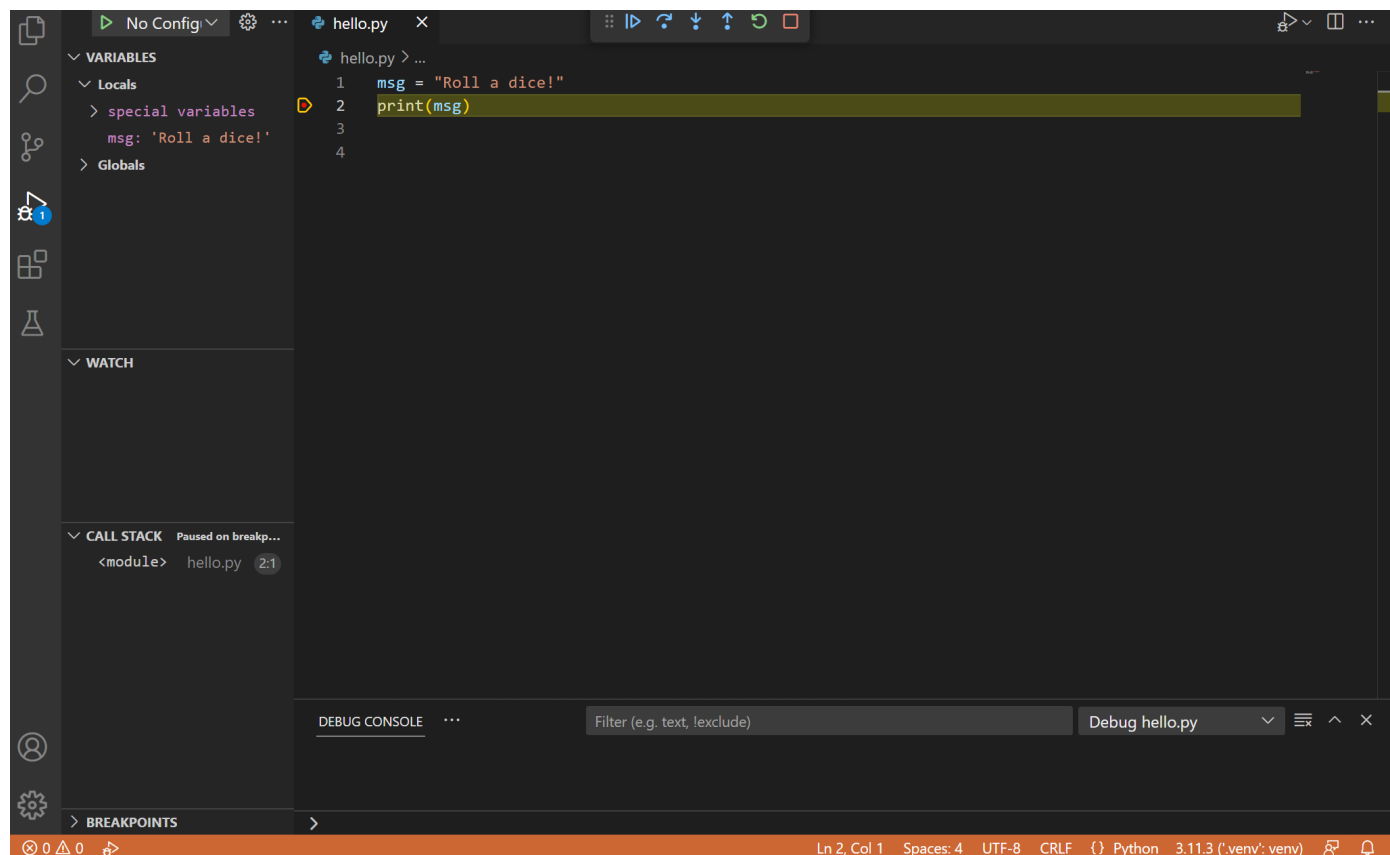
Next, to initialize the debugger, press `F5`. Since this is your first time debugging this file, a configuration menu will open from the Command Palette allowing you to select the type of debug configuration you would like for the opened file.



Note: VS Code uses JSON files for all of its various configurations; `launch.json` is the standard name for a file containing debugging configurations.

Select **Python File**, which is the configuration that runs the current file shown in the editor using the currently selected Python interpreter.

The debugger will start, and then stop at the first line of the file breakpoint. The current line is indicated with a yellow arrow in the left margin. If you examine the **Local** variables window at this point, you can see that the `msg` variable appears in the **Local** pane.



A debug toolbar appears along the top with the following commands from left to right: continue (F5), step over (F10), step into (F11), step out (⇧F11), restart (⇧⌘F5), and stop (⇧F5).



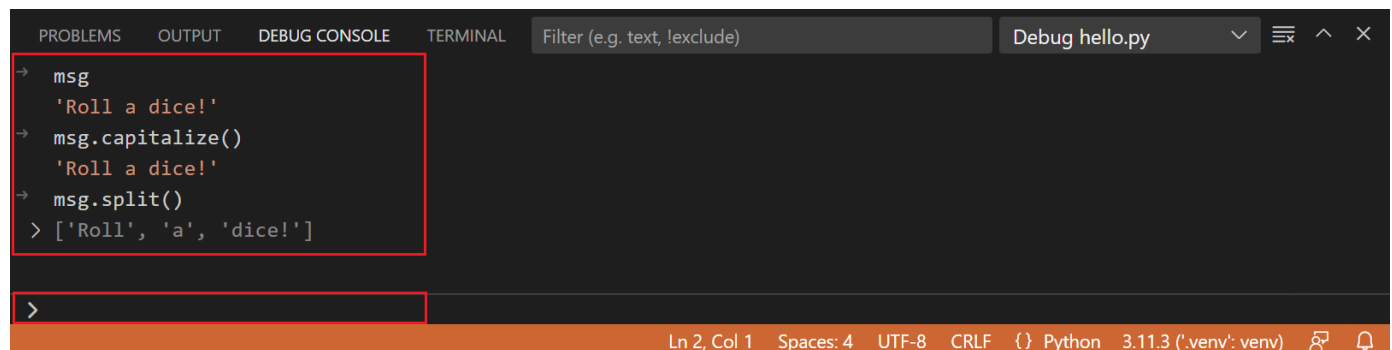
The Status Bar also changes color (orange in many themes) to indicate that you're in debug mode. The **Python Debug Console** also appears automatically in the lower right panel to show the commands being run, along with the program output.

To continue running the program, select the continue command on the debug toolbar (F5). The debugger runs the program to the end.

Tip Debugging information can also be seen by hovering over code, such as variables. In the case of `msg`, hovering over the variable will display the string `Roll a dice!` in a box above the variable.

You can also work with variables in the **Debug Console** (If you don't see it, select **Debug Console** in the lower right area of VS Code, or select it from the ... menu.) Then try entering the following lines, one by one, at the `>` prompt at the bottom of the console:

```
msg
msg.capitalize()
msg.split()
```

[Copy](#)

Select the blue **Continue** button on the toolbar again (or press `F5`) to run the program to completion. "Roll a dice!" appears in the **Python Debug Console** if you switch back to it, and VS Code exits debugging mode once the program is complete.

If you restart the debugger, the debugger again stops on the first breakpoint.

To stop running a program before it's complete, use the red square stop button on the debug toolbar (`⇧F5`), or use the **Run > Stop debugging** menu command.

For full details, see [Debugging configurations \(/docs/python/debugging/\)](/docs/python/debugging/), which includes notes on how to use a specific Python interpreter for debugging.

Tip: Use Logpoints instead of print statements: Developers often litter source code with `print` statements to quickly inspect variables without necessarily stepping through each line of code in a debugger. In VS Code, you can instead use **Logpoints**. A Logpoint is like a breakpoint except that it logs a message to the console and doesn't stop the program. For more information, see [Logpoints \(/docs/debugtest/debugging#_logpoints\)](/docs/debugtest/debugging#_logpoints) in the main VS Code debugging article.

Install and use packages

Let's build upon the previous example by using packages.

In Python, packages are how you obtain any number of useful code libraries, typically from [PyPI \(https://pypi.org/\)](https://pypi.org/)

pypi.org/), that provide additional functionality to your program. For this example, you use the `numpy` package to generate a random number.

Return to the **Explorer** view (the top-most icon on the left side, which shows files), open `hello.py`, and paste in the following source code:

```
import numpy as np

msg = "Roll a dice!"
print(msg)

print(np.random.randint(1,9))
```

[Copy](#)

Tip: If you enter the above code by hand, you may find that auto-completions change the names after the `as` keywords when you press `Enter` at the end of a line. To avoid this, type a space, then `Enter`.

Next, run the file in the debugger using the "Python: Current file" configuration as described in the last section.

You should see the message, **"ModuleNotFoundError: No module named 'numpy'"**. This message indicates that the required package isn't available in your interpreter. If you're using an Anaconda distribution or have previously installed the `numpy` package you may not see this message.

To install the `numpy` package, stop the debugger and use the Command Palette to run **Terminal: Create New Terminal** (`^⇧``). This command opens a command prompt for your selected interpreter.

To install the required packages in your virtual environment, enter the following commands as appropriate for your operating system:

1 Install the packages

```
# Don't use with Anaconda distributions because they include matplotlib already.

# macOS
python3 -m pip install numpy

# Windows (may require elevation)
py -m pip install numpy

# Linux (Debian)
apt-get install python3-tk
python3 -m pip install numpy
```

[Copy](#)

2 Now, rerun the program, with or without the debugger, to view the output!

Managing dependencies across environments

When working on Python projects, it's essential to manage your dependencies effectively. One useful tip is to use the `pip freeze > requirements.txt` command. This command helps you create a `requirements.txt` file that lists all the packages installed in your virtual environment. This file can then be used to recreate the same environment elsewhere.

Follow these steps to create a `requirements.txt` file:

- 1 Activate your virtual environment, if you haven't already.

```
source venv/bin/activate # On macOS/Linux
```

[Copy](#)

```
.\venv\Scripts\activate # On Windows
```

[Copy](#)

- 2 Generate the `requirements.txt` file.

```
pip freeze > requirements.txt
```

[Copy](#)

You can now use the newly generated `requirements.txt` file to install dependencies in another environment. Furthermore, you can continue to add dependencies to it as your project may grow in complexity.

```
pip install -r requirements.txt
```

[Copy](#)

By following these steps, you ensure that your project dependencies are consistent across different environments, making it easier to collaborate with others and deploy your project.

Congrats on completing the Python tutorial! During the course of this tutorial, you learned how to create a Python project, create a virtual environment, run and debug your Python code, and install Python packages. Explore additional resources to learn how to get the most out of Python in Visual Studio Code!

Next steps

To learn how to build web apps with popular Python web frameworks, see the following tutorials:

- [Use Django in Visual Studio Code \(/docs/python/tutorial-django\)](/docs/python/tutorial-django)
- [Use Flask in Visual Studio Code \(/docs/python/tutorial-flask\)](/docs/python/tutorial-flask)
- [Use FastAPI in Visual Studio Code \(/docs/python/tutorial-fastapi\)](/docs/python/tutorial-fastapi)

There is then much more to explore with Python in Visual Studio Code:

- [Python profile template \(/docs/configure/profiles#_python-profile-template\)](/docs/configure/profiles#_python-profile-template) - Create a new [profile \(/docs/configure/profiles\)](/docs/configure/profiles) with a curated set of extensions, settings, and snippets
- [Editing code \(/docs/python/editing\)](/docs/python/editing) - Learn about autocomplete, IntelliSense, formatting, and refactoring for Python.
- [Linting \(/docs/python/linting\)](/docs/python/linting) - Enable, configure, and apply a variety of Python linters.
- [Debugging \(/docs/python/debugging\)](/docs/python/debugging) - Learn to debug Python both locally and remotely.
- [Testing \(/docs/python/testing\)](/docs/python/testing) - Configure test environments and discover, run, and debug tests.
- [Settings reference \(/docs/python/settings-reference\)](/docs/python/settings-reference) - Explore the full range of Python-related settings in VS Code.
- [Deploy Python to Azure App Service \(https://learn.microsoft.com/azure/developer/python/tutorial-containerize-deploy-python-web-app-azure-01\)](https://learn.microsoft.com/azure/developer/python/tutorial-containerize-deploy-python-web-app-azure-01)
- [Deploy Python to Container Apps \(https://learn.microsoft.com/azure/developer/python/tutorial-deploy-python-web-app-azure-container-apps-01\)](https://learn.microsoft.com/azure/developer/python/tutorial-deploy-python-web-app-azure-container-apps-01)

Was this documentation helpful?

Yes


No


06/12/2025

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

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

 [Request features\(https://go.microsoft.com/fwlink/?LinkID=533482\)](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://go.microsoft.com/fwlink/?LinkID=533687>)



(<https://github.com/microsoft/vscode>)



(<https://www.youtube.com/@code>)

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

Privacy (<https://go.microsoft.com/fwlink/?LinkId=521839>)

Terms of Use (<https://www.microsoft.com/legal/terms-of-use>) License (/License)