

TOPICS Tutorial

VS Code中的Python入门

<https://github.com/Microsoft/vscode-docs/blob/master/docs/python/python-tutorial.md>

在本教程中，您将使用Python 3在Visual Studio Code中创建最简单的Python“Hello World”应用程序。通过使用Python扩展，您可以将VS Code变成一个功能强大的轻量级Python IDE（您可能会找到PyCharm的高效替代品）。

本教程向您介绍VS Code作为Python环境，主要介绍如何通过以下任务来编辑，运行和调试代码：

- 编写，运行和调试Python“Hello World”应用程序
- 了解如何通过创建Python虚拟环境来安装软件包
- 编写一个简单的Python脚本以在VS Code中绘制图形

本教程无意教您Python本身。熟悉VS Code的基础知识之后，您就可以在VS Code的上下文中遵循python.org上的

(<https://wiki.python.org/moin/BeginnersGuide/Programmers>)任何编程教程，(<https://wiki.python.org/moin/BeginnersGuide/Programmers>)以对该语言进行介绍。

如果您有任何问题，请随时在VS Code文档库中提交 (<https://github.com/Microsoft/vscode-docs/issues>)本教程的问题。

注意：在本教程中，您可以将VS Code与Python 2结合使用，但是您需要对代码进行适当的更改，此处不做介绍。

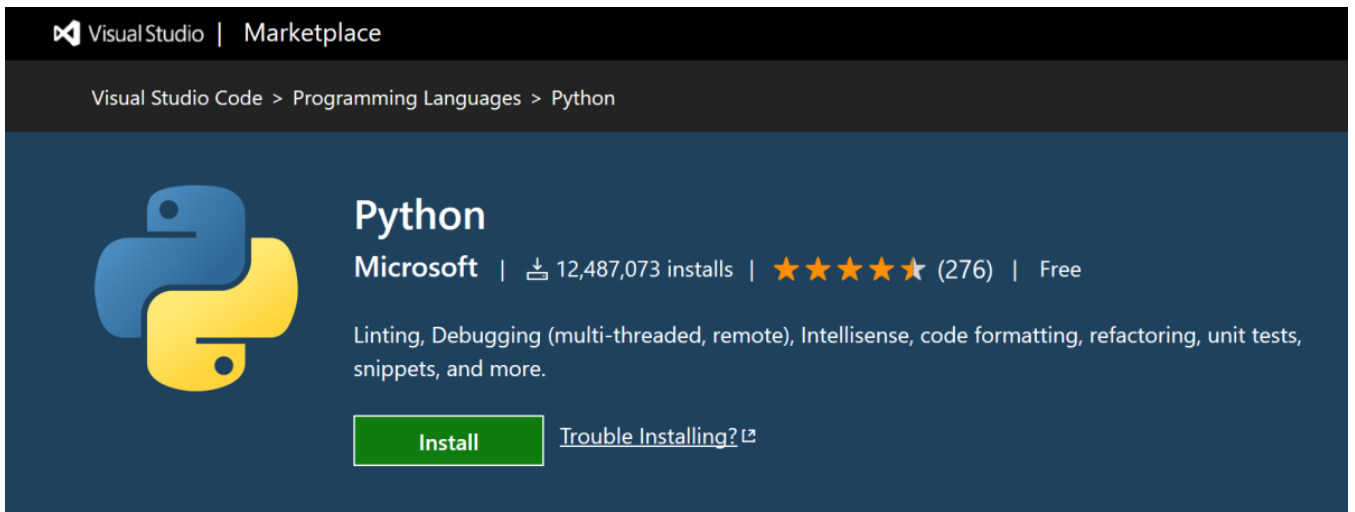
先决条件

要成功完成本教程，您需要首先设置Python开发环境。具体来说，本教程要求：

- VS代码
- VS Code Python扩展
- Python 3

安装Visual Studio代码和Python扩展

1. 如果尚未这样做，请安装VS Code (<https://code.visualstudio.com/>)。
2. 接下来，从Visual Studio Marketplace 安装VS Code (<https://marketplace.visualstudio.com/items?itemName=ms-python.python>)的Python扩展 (<https://marketplace.visualstudio.com/items?itemName=ms-python.python>)。有关安装扩展的更多详细信息，请参阅Extension Marketplace (</docs/editor/extension-gallery>)。Python扩展名为Python，由Microsoft发布。



(<https://marketplace.visualstudio.com/items?itemName=ms-python.python>)

安装Python解释器

与Python扩展一起，您需要安装Python解释器。您使用哪种口译员取决于您的特定需求，但是下面提供了一些指导。

视窗

从python.org (<https://www.python.org/downloads/>)安装Python (<https://www.python.org/downloads/>)。通常，您可以使用页面第一个出现的Download Python按钮下载最新版本。

注意：如果您没有管理员权限，则在Windows上安装Python的另一个选择是使用Microsoft Store。Microsoft Store提供了Python 3.7 (<https://www.microsoft.com/en-us/p/python-37/9nj46sx7x90p>)和Python 3.8的 (<https://www.microsoft.com/en-us/p/python-38/9msszt1n39l>)安装。请注意，使用此方法可能与某些软件包存在兼容性问题。

有关在Windows上使用Python的其他信息，请参阅Python.org上的在Windows上使用Python。 (<https://docs.python.org/3.7/using/windows.html>)

苹果系统

不支持在macOS上系统安装Python。相反，建议通过Homebrew (<https://brew.sh/>)安装。要在macOS上使用Homebrew安装Python `brew install python3`，请在终端提示下使用。

注意在macOS上，请确保PATH环境变量中包含VS Code安装的位置。有关更多信息，请参见这些设置说明 (/docs/setup/mac#_launching-from-the-command-line)。

的Linux

在Linux上的内置Python 3安装效果很好，但是要安装其他Python软件包，必须 `pip` 使用`get-pip.py`进行 (<https://pip.pypa.io/en/stable/installing/#installing-with-get-pip-py>)安装。

其他选择

- **数据科学**：如果使用Python的主要目的是数据科学，那么您可以考虑从Anaconda (<https://www.anaconda.com/download/>)下载。Anaconda不仅提供Python解释器，还提供许多有用的数据科学库和工具。
- **适用于Linux的Windows子系统**：如果您正在Windows上工作，并且希望Linux环境可用于Python，那么可以选择适用于Linux (<https://docs.microsoft.com/windows/wsl/about>)的Windows子系统 (<https://docs.microsoft.com/windows/wsl/about>) (WSL)。如果选择此选项，则还需要安装Remote-WSL扩展 (<https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl>)。有关将WSL与VS Code一起使用的更多信息，请参见VS Code远程开发 (</docs/remote/remote-overview>)或尝试“在WSL中使用”教程 (</remote-tutorials/wsl/getting-started>)，该教程 (</remote-tutorials/wsl/getting-started>)将引导您完成设置WSL，安装Python以及创建在WSL中运行的Hello World应用程序。

验证Python安装

要验证您是否已在计算机上成功安装了Python，请运行以下命令之一（取决于您的操作系统）：

- Linux / macOS：打开“终端窗口”并键入以下命令：

```
python3 --version
```

- Windows：打开命令提示符并运行以下命令：

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

如果安装成功，则输出窗口应显示您安装的Python版本。

注意可以 `py -0` 在VS Code集成终端中使用该命令来查看计算机上安装的python版本。默认解释器由星号 (*) 标识。

在项目（工作区）文件夹中启动VS Code

使用命令提示符或终端，创建一个名为“hello”的空文件夹，导航至该文件夹，然后通过输入以下命令 `code` 在该文件夹（.）中打开VS Code（）：

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

注意：如果您使用的是Anaconda发行版，请确保使用Anaconda命令提示符。

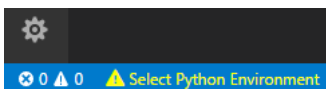
通过在文件夹中启动VS Code，该文件夹将成为您的“工作区”。VS Code将特定于该工作空间的 `.vscode/settings.json` 设置存储在中，这些设置与全局存储的用户设置分开。

或者，您可以通过操作系统UI运行VS Code，然后使用“文件”>“打开文件夹”打开项目文件夹。

选择一个Python解释器

Python是一种解释型语言，为了运行Python代码并获得Python IntelliSense，必须告诉VS Code使用哪个解释器。

在VS Code中，通过打开**命令面板**（`Ctrl + Shift + P`）选择一个Python 3解释器，开始键入**Python：选择要解释的解释器**命令，然后选择该命令。如果可用，您还可以使用状态栏上的“**选择Python环境**”选项（它可能已经显示了选定的解释器）：



该命令显示了VS Code可以自动找到的可用解释器的列表，包括虚拟环境。如果看不到所需的解释器，请参阅“配置Python环境” (</docs/python/environments>)。

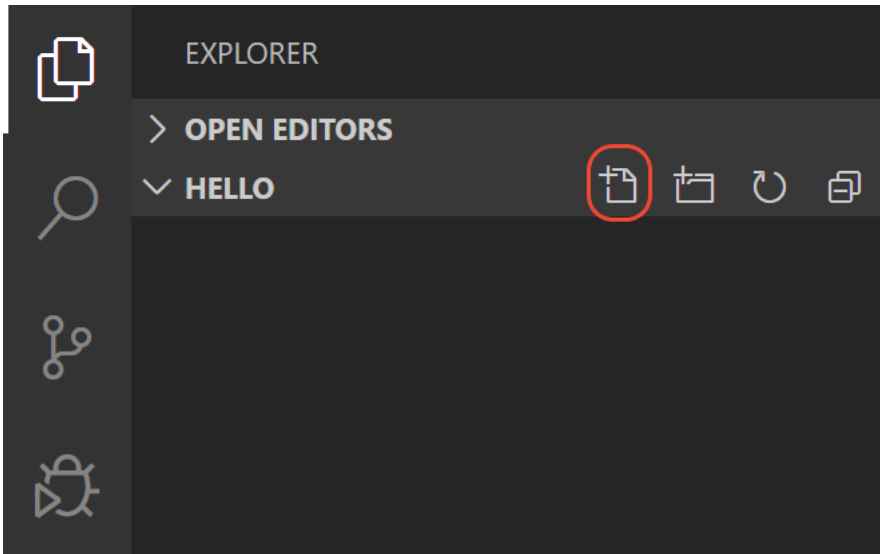
注：当使用蟒蛇分布，正确的解释应该有后缀（`'base':conda`），例如 `Python 3.7.3 64-bit ('base':conda)`。

选择解释器会将 `python.pythonPath` 工作空间设置中的值设置为解释器的路径。要查看设置，请选择**文件** > **首选项** > **设置**（在macOS上为**代码** > **首选项** > **设置**），然后选择**工作区设置**选项卡。

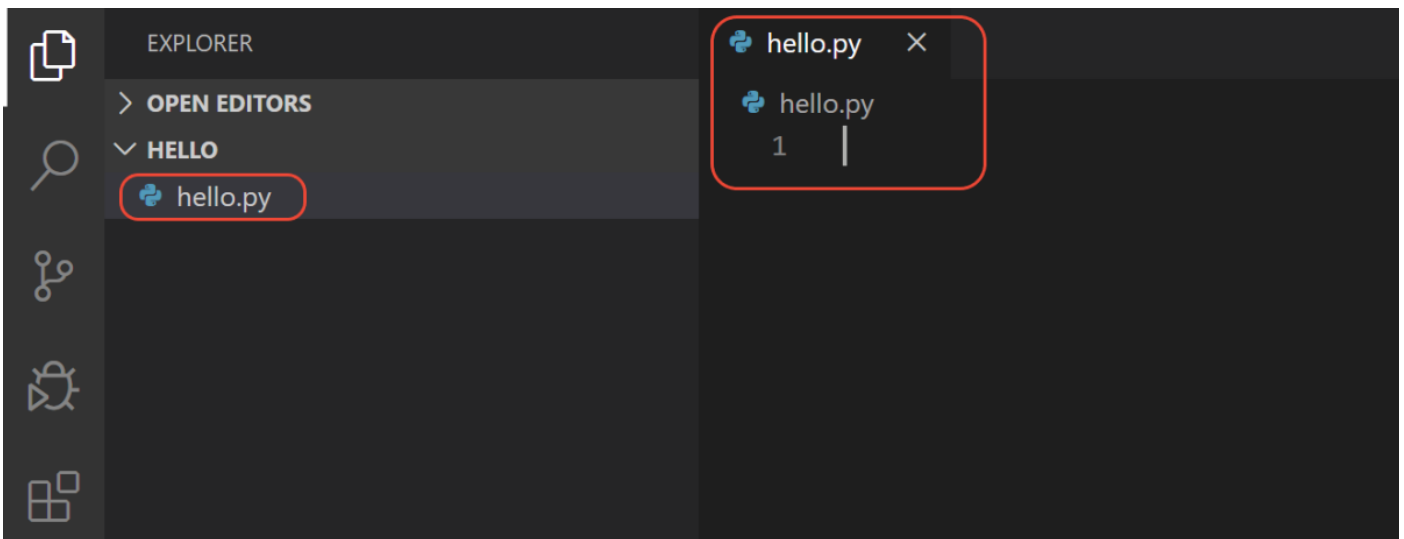
注意：如果选择未打开工作空间文件夹的解释器，则将 `python.pythonPath` 在您的用户设置中设置VS Code，这通常设置VS Code的默认解释器。用户设置可确保您始终具有用于Python项目的默认解释器。通过工作区设置，您可以覆盖用户设置。

创建一个Python Hello World源代码文件

在“文件资源管理器”工具栏中，选择文件 `hello` 夹上的“新建文件”按钮：



命名文件 `hello.py`，它会在编辑器中自动打开：



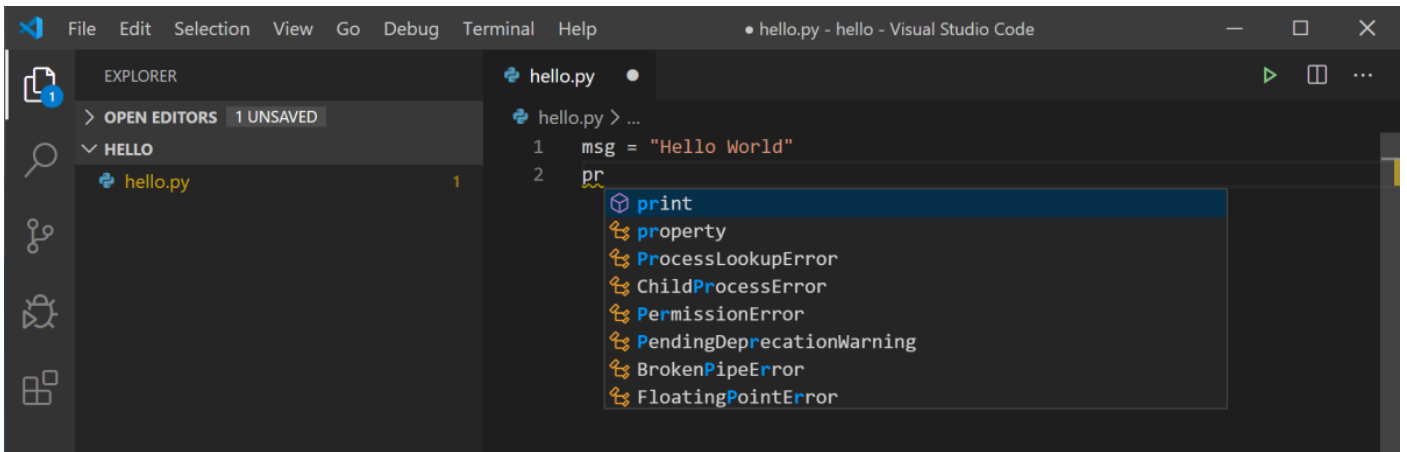
通过使用 `.py` 文件扩展名，您告诉VS Code将这个文件解释为Python程序，以便它使用Python扩展名和选定的解释器评估内容。

注意：“文件资源管理器”工具栏还允许您在工作区中创建文件夹，以更好地组织代码。您可以使用“新建文件夹”按钮快速创建文件夹。

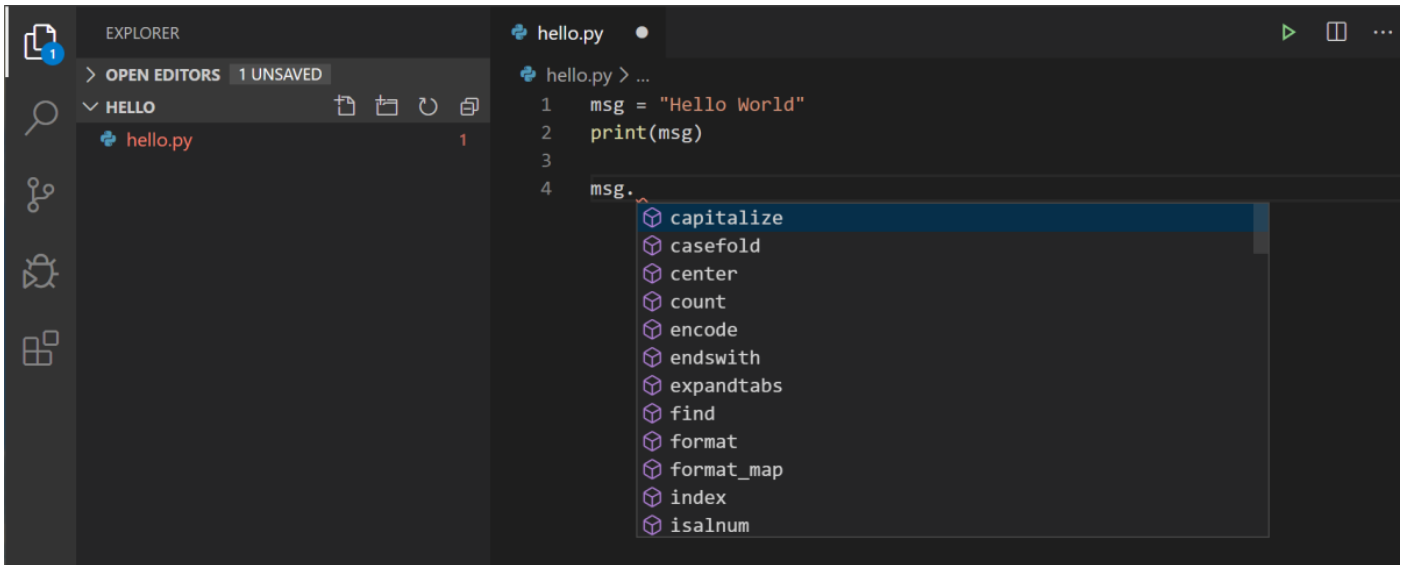
现在您的工作区中已有一个代码文件，在中输入以下源代码 `hello.py`：

```
msg = "Hello World"
print(msg)
```

当您开始键入时 `print`，请注意IntelliSense (</docs/editor/intellisense>)如何显示自动完成选项。



IntelliSense和自动完成功能适用于标准Python模块以及您已安装到所选Python解释器环境中的其他软件包。它还提供了对象类型上可用方法的完成。例如，由于 `msg` 变量包含字符串，因此在键入时IntelliSense提供了字符串方法 `msg.`：

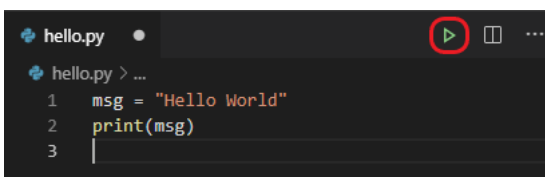


可以尝试使用IntelliSense进行更多尝试，但是请还原所做的更改，以便仅包含 `msg` 变量和 `print` 调用，并保存文件（`Ctrl + S`）。

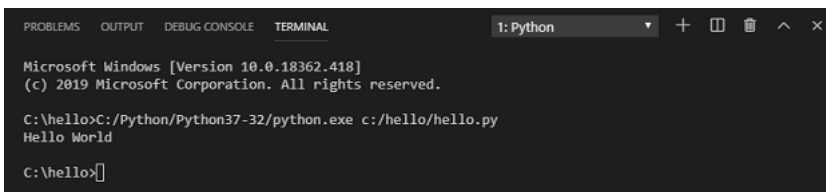
有关编辑，格式化和重构的完整详细信息，请参见编辑代码（/docs/python/editing）。Python扩展还完全支持Linting（/docs/python/linting）。

运行Hello World

`hello.py` 使用Python 运行很简单。只需点击编辑器右上角的“在终端中运行Python文件”播放按钮。

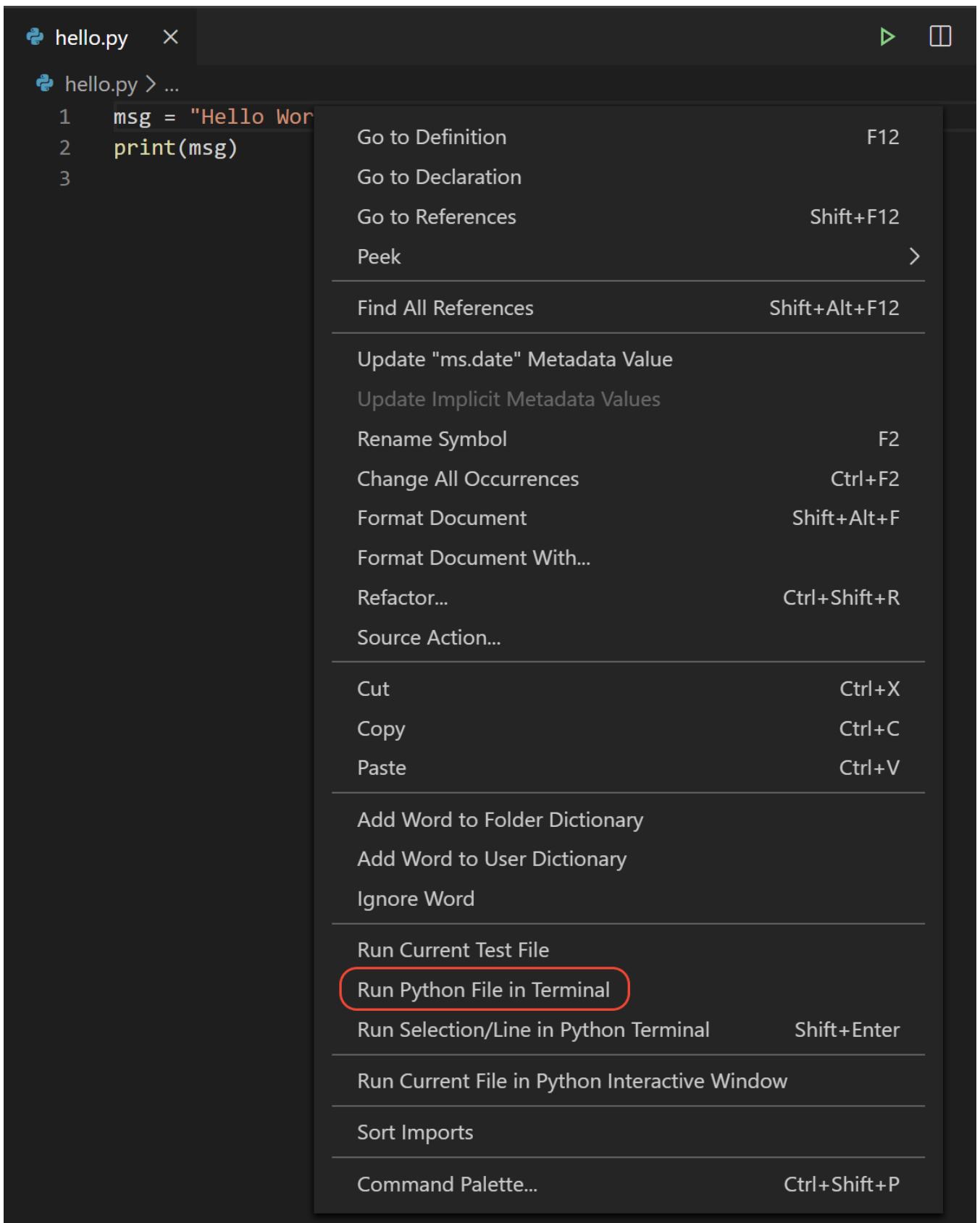


该按钮将打开一个终端面板，在其中自动激活您的Python解释器，然后运行 `python3 hello.py`（macOS / Linux）或 `python hello.py`（Windows）：



您还可以通过三种其他方法在VS Code中运行Python代码：

- 右键单击编辑器窗口中的任意位置，然后选择“在终端中运行Python文件”（这会保存文件）：

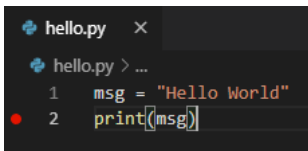


- 选择一个或多个行，然后按 Shift + Enter 或右键单击并在“Python Terminal”中选择“运行选择/行”。该命令对于仅测试文件的一部分非常方便。
- 从命令面板（Ctrl + Shift + P）中，选择Python: Start REPL命令以为当前选择的Python解释器打开REPL终端。然后在REPL中，您可以一次输入并运行一行代码。

配置并运行调试器

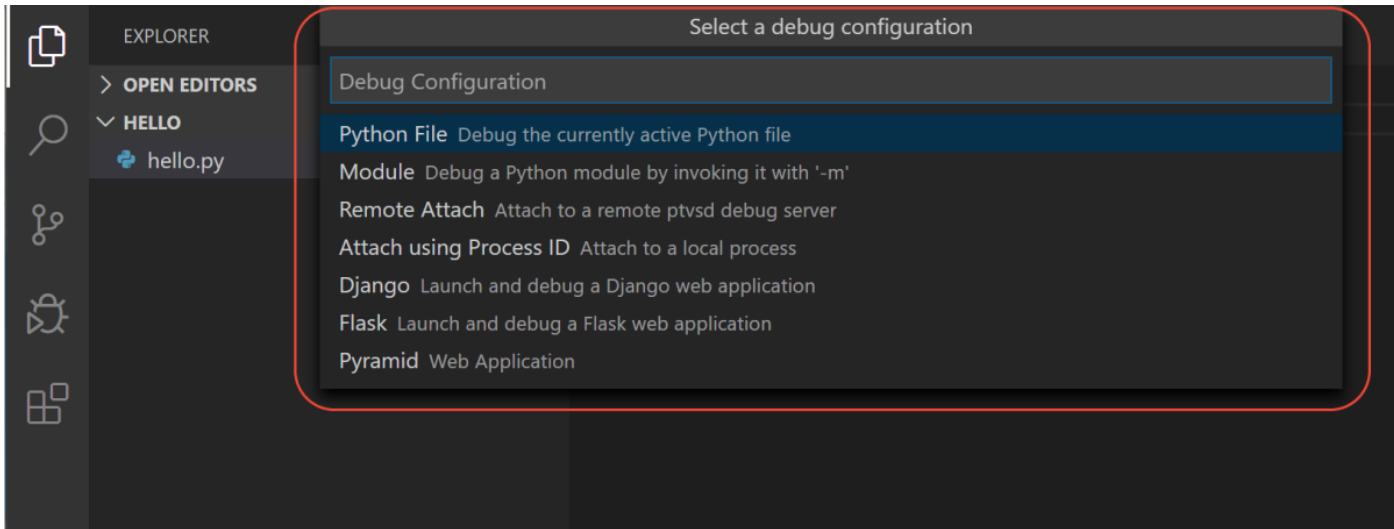
现在让我们尝试调试简单的Hello World程序。

首先，hello.py 通过将光标放在 print 呼叫上并按 F9，在第2行上设置断点。或者，只需单击编辑器的左装订线，在行号旁边。设置断点时，装订线中会出现一个红色圆圈。



```
hello.py x
hello.py > ...
1 msg = "Hello World"
2 print(msg)
```

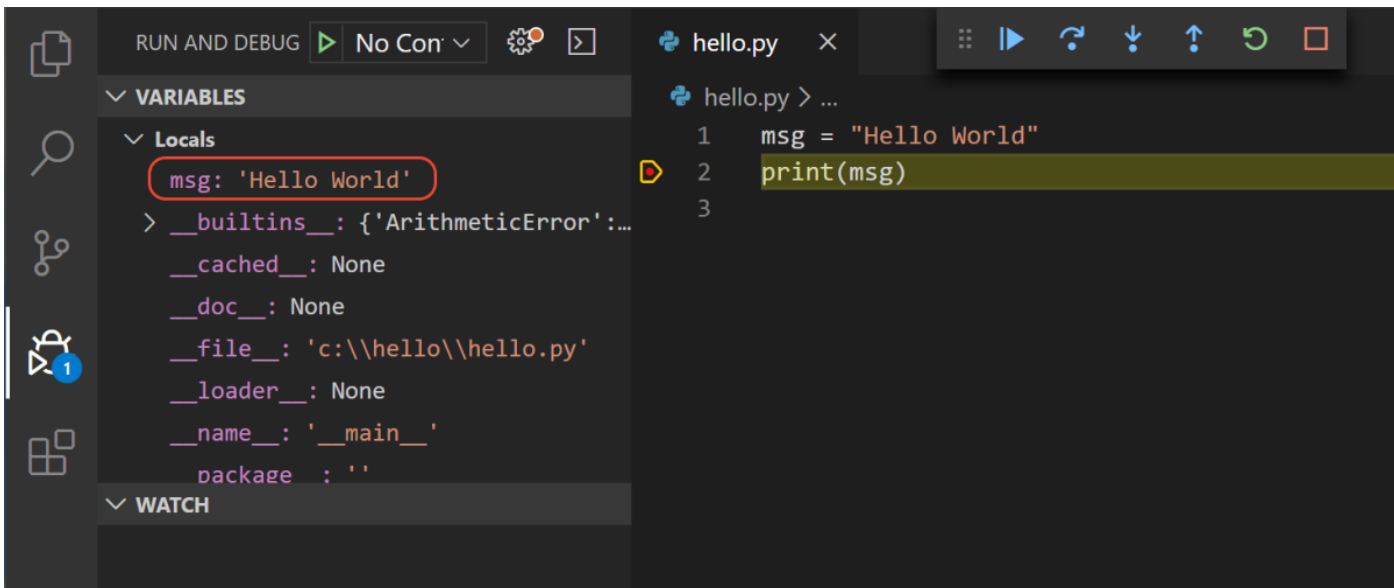
接下来，要初始化调试器，请按 F5。由于这是您第一次调试此文件，因此将从“命令面板”中打开一个配置菜单，您可以为打开的文件选择所需的调试配置类型。



注意：VS Code将JSON文件用于其所有各种配置。`launch.json` 是包含调试配置的文件的标准名称。

在调试配置中对 (/docs/python/debugging)这些不同的配置进行了详细说明；现在，只需选择Python File，这是使用当前选择的Python解释器运行编辑器中显示的当前文件的配置。

调试器将在文件断点的第一行停止。当前行在左边距中用黄色箭头指示。如果此时检查“本地变量”窗口，您将看到 `msg` “本地”窗格中现在显示已定义的变量。



调试工具栏从左到右显示在顶部，带有以下命令：继续（F5），跳过（F10），进入（F11），退出（Shift + F11），重新启动（Ctrl + Shift + F5），然后停止（Shift + F5）。



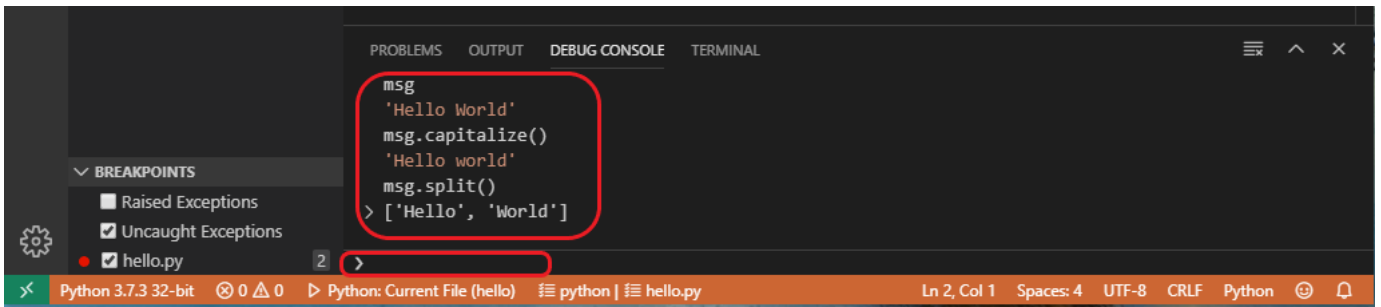
状态栏还会更改颜色（在许多主题中为橙色），以表明您处于调试模式。所述的Python调试控制台还右下面板自动出现以示出命令正在运行，与该程序一起输出。

要继续运行程序，请在调试工具栏（F5）上选择Continue命令。调试器将程序运行到最后。

提示也可以通过将鼠标悬停在代码（例如变量）上来查看调试信息。在的情况下 `msg`，将鼠标悬停在变量上将 `Hello world` 在变量上方的框中显示字符串。

您也可以在调试控制台中使用变量（如果看不到它，请在VS Code右下方区域中选择“调试控制台”，或从...菜单中选择它。）然后尝试输入以下几行，在控制台底部的>提示符下，按一个：

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



再次选择工具栏上的蓝色“继续”按钮（或按F5键）以运行程序以完成操作。如果您切换回“Hello World”，则会在Python调试控制台中显示，程序完成后，VS Code将退出调试模式。

如果重新启动调试器，调试器将再次在第一个断点处停止。

要在程序完成之前停止运行，请使用调试工具栏上的红色正方形停止按钮（Shift + F5），或使用“运行”>“停止调试”菜单命令。

有关完整的详细信息，请参见调试配置（/docs/python/debugging），其中包括有关如何使用特定的Python解释器进行调试的说明。

提示：使用Logpoints代替print语句：开发人员经常在源代码中填充 print 语句，以快速检查变量，而不必单步调试程序中的每一行代码。在VS Code中，您可以改用 Logpoints。日志点类似于断点，不同之处在于它将消息记录到控制台并且不会停止程序。有关更多信息，请参见主要VS Code调试文章中的Logpoints（/docs/editor/debugging#_logpoints）。

安装和使用软件包

现在让我们运行一个更有趣的示例。在Python中，包是如何获取许多有用的代码库的方法，通常是从PyPI中获取（<https://pypi.org/>）。在此示例中，您使用 matplotlib 和 numpy 包来创建图形化绘图，这与数据科学一样。（请注意，由于缺少必需的UI支持，因此 matplotlib 在Windows子系统中（<https://docs.microsoft.com/windows/wsl/about>）运行时无法显示图形。）

返回“资源管理器”视图（左侧最上面的图标，显示文件），创建一个名为的新文件 standardplot.py，并粘贴以下源代码：

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 20, 100) # Create a list of evenly-spaced numbers over the range
plt.plot(x, np.sin(x))      # Plot the sine of each x point
plt.show()                 # Display the plot
```

提示：如果您手动输入上述代码，则在一行的末尾按 Enter 时，您可能会发现自动完成功能会更改关键字后的名称。为避免这种情况，请输入空格，然后按 Enter。

接下来，尝试使用上一节中所述的“Python：当前文件”配置在调试器中运行文件。

除非您使用的是Anaconda发行版或以前已经安装了 matplotlib 软件包，否则应该看到消息“ModuleNotFoundError：没有名为'matplotlib'的模块”。这样的消息表明所需的软件包在您的系统中不可用。

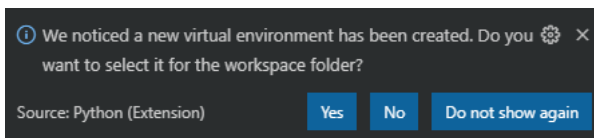
要安装 matplotlib 软件包（也将 numpy 作为依赖项安装），请停止调试器并使用命令面板运行终端：**创建新的集成终端**（Ctrl + Shift + `）。该命令为您选择的解释器打开命令提示符。

Python开发人员中的最佳实践是避免将软件包安装到全局解释器环境中。相反，您使用的是特定 virtual environment 于项目的项目，其中包含全局解释器的副本。激活该环境后，您随后安装的所有软件包都将与其他环境隔离。这种隔离减少了因版本冲突而引起的许多复杂情况。要创建虚拟环境并安装所需的软件包，请根据您的操作系统输入以下命令：

注意：有关虚拟环境的其他信息，请参阅环境（/docs/python/environments#_global-virtual-and-conda-environments）。

1. 创建并激活虚拟环境

注意：创建新的虚拟环境时，VS Code会提示您将其设置为工作区文件夹的默认环境。如果选择此选项，则在打开新终端时将自动激活环境。



对于窗口

```
py -3 -m venv .venv
.venv\scripts\activate
```

如果activate命令生成消息“Activate.ps1没有经过数字签名。您无法在当前系统上运行此脚本。”，则需要临时更改PowerShell执行策略以允许脚本运行（请参阅“关于执行策略（<https://go.microsoft.com/fwlink/?LinkID=135170>）”。PowerShell文档）：

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process
```

对于macOS / Linux

```
python3 -m venv .venv
source .venv/bin/activate
```

2. 使用Python选择新的环境：从“命令面板”中选择“解释器”命令。

3. 安装软件包

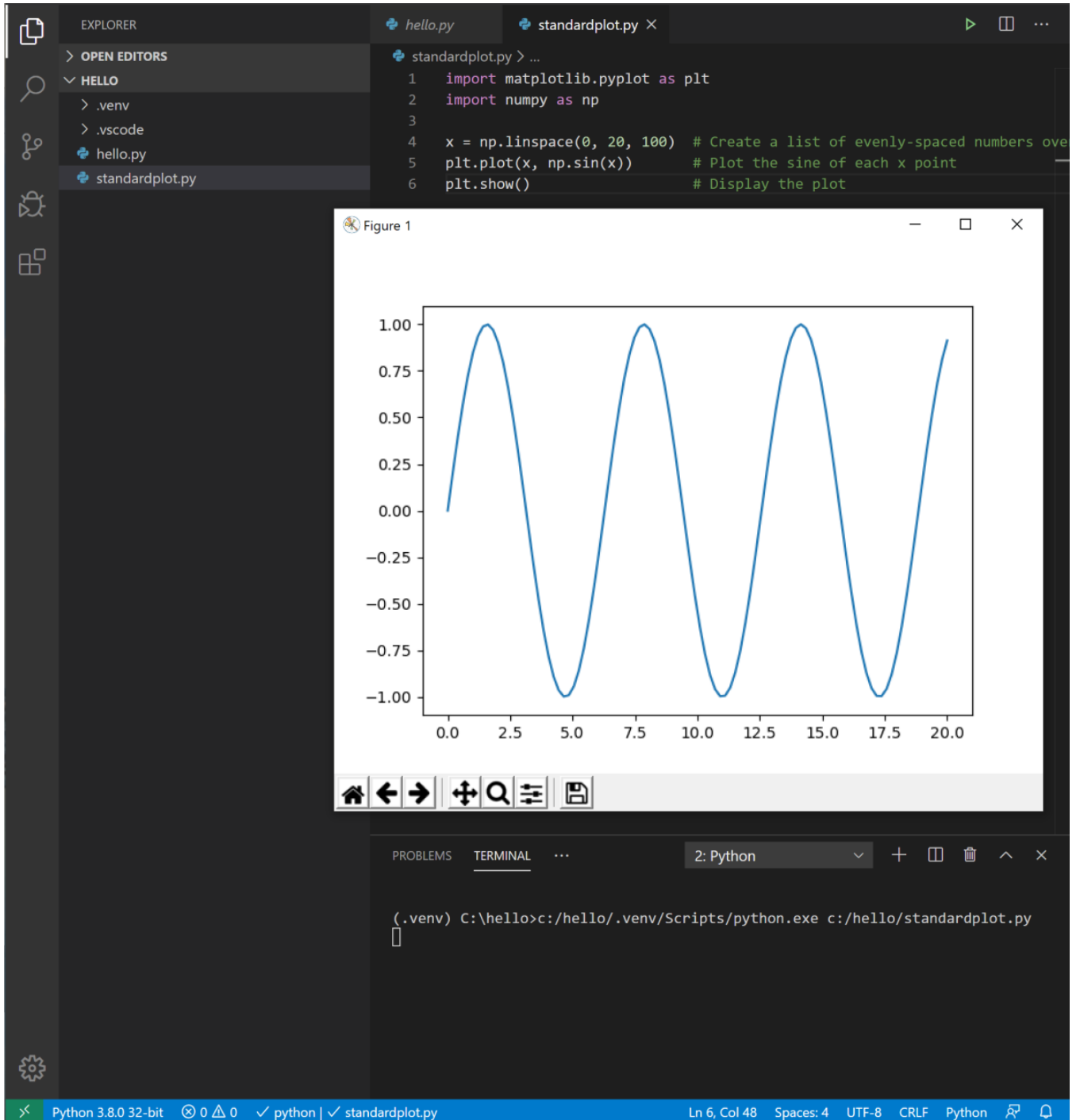
```
# Don't use with Anaconda distributions because they include matplotlib already.

# macOS
python3 -m pip install matplotlib

# Windows (may require elevation)
python -m pip install matplotlib

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

4. 立即重新运行程序（带调试器或不带调试器），过一会儿，将出现一个带有输出的绘图窗口：



5. 完成后，`deactivate` 在终端窗口中键入以停用虚拟环境。

有关创建和激活虚拟环境以及安装软件包的其他示例，请参见Django教程 (/docs/python/tutorial-django)和Flask教程 (/docs/python/tutorial-flask)。

下一步 #