Getting Started with Python in VS Code

In this tutorial, you use Python 3 to create the simplest Python "Hello World" application in Visual Studio Code. By using the Python extension, you make VS Code into a great lightweight Python IDE (which you may find a productive alternative to PyCharm).

This tutorial introduces you to VS Code as a Python environment, primarily how to edit, run, and debug code through the following tasks:

- Write, run, and debug a Python "Hello World" Application
- Learn how to install packages by creating Python virtual environments
- Write a simple Python script to plot figures within VS Code

This tutorial is not intended to teach you Python itself. Once you are familiar with the basics of VS Code, you can then follow any of the programming tutorials on python.org within the context of VS Code for an introduction to the language.

If you have any problems, feel free to file an issue for this tutorial in the VS Code documentation repository.

> **Note**: You can use VS Code with Python 2 with this tutorial, but you need to make appropriate changes to the code, which are not covered here.

Prerequisites

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

- VS Code
- VS Code Python extension
- Python 3

Install Visual Studio Code and the Python Extension

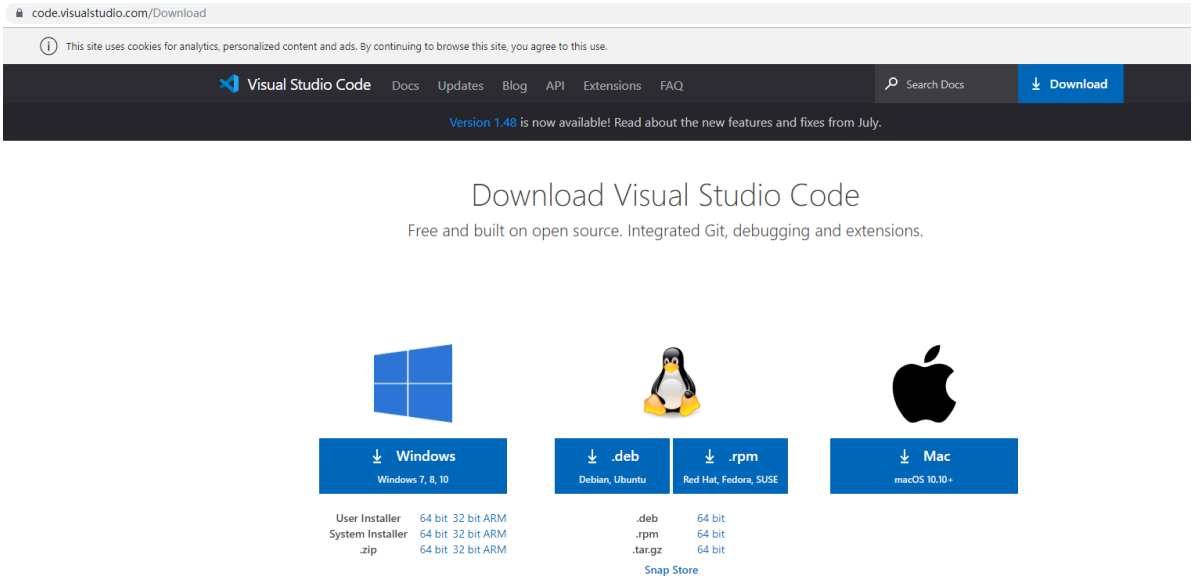
Install Studio Code

Visual Studio Code is a powerful open-source code editor developed by Microsoft. It has built-in debugging support, embedded Git control, syntax highlighting, code completion, integrated terminal, code refactoring, and snippets. Visual Studio Code is cross-platform, available on Windows, Linux, and macOS.

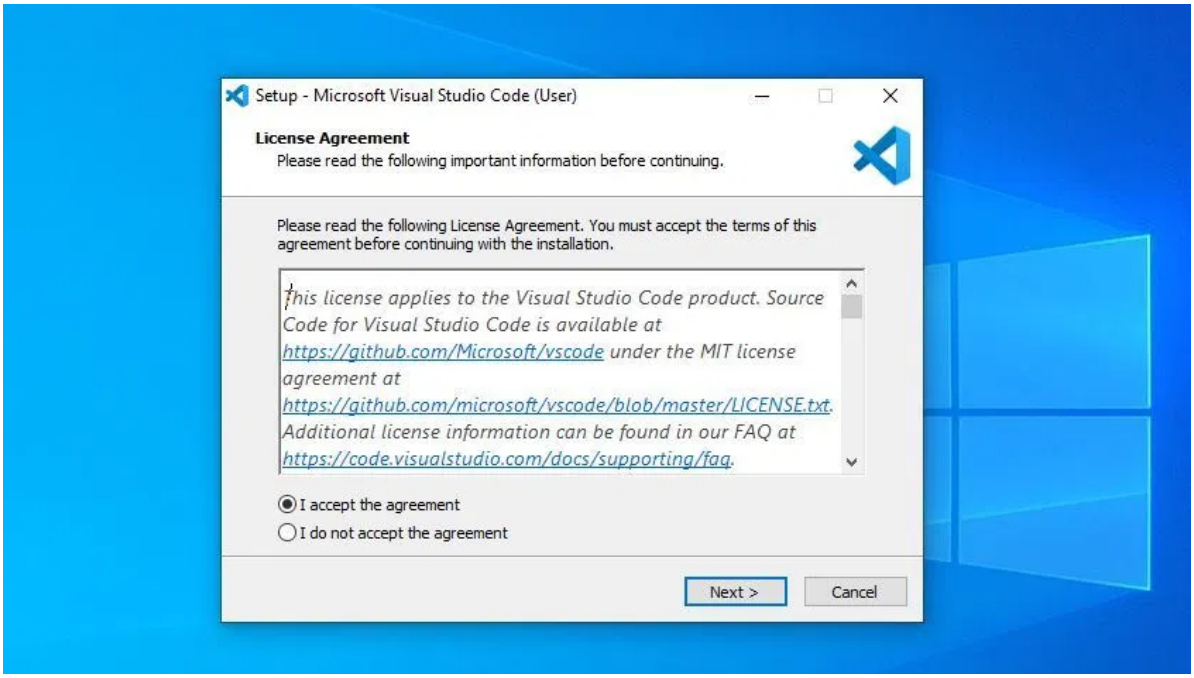
In this section, we'll discuss how to download and install the VS Code on Windows.

Step 1: First of all, we need to download the installer file for Windows operating system. For that visit code.visualstudio.com and download the windows version of Vustua Studio Code or click the download button below.

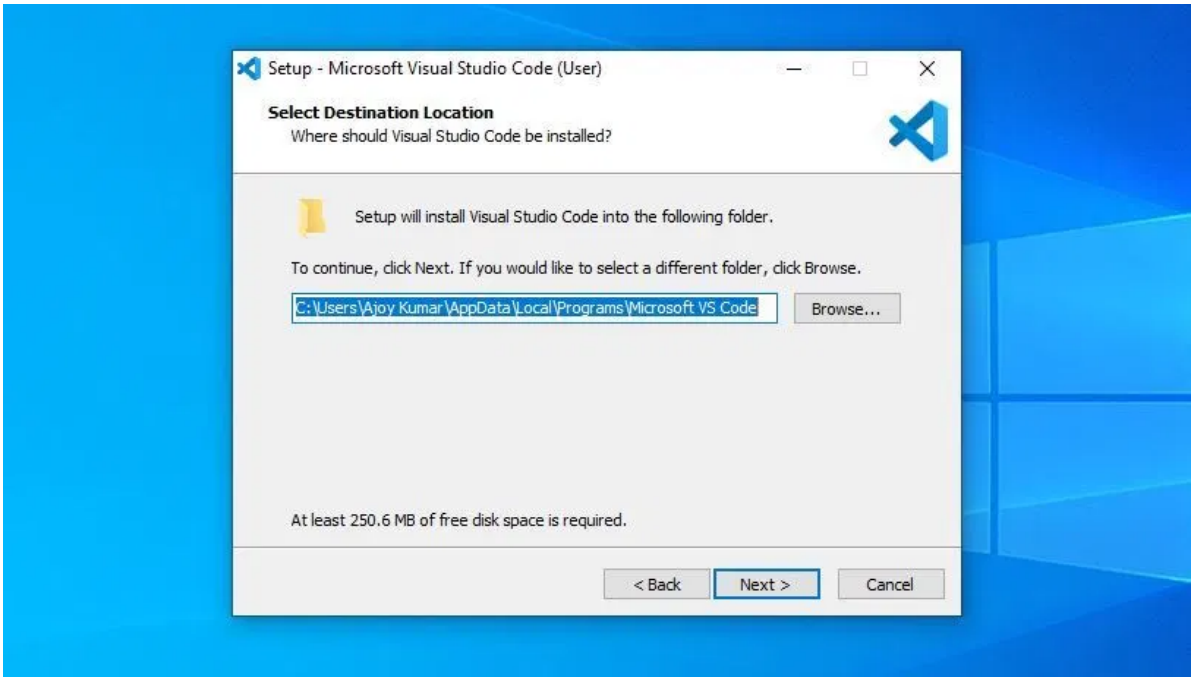
> **Note**:- Visual Studio Code is also available for other operating systems like MacOS and Linux (Ubuntu, Debian, Red Hate, Fedora).



Step 2: After you have downloaded the installer file open it and *accept the licence agreement* then click on "*Next*".

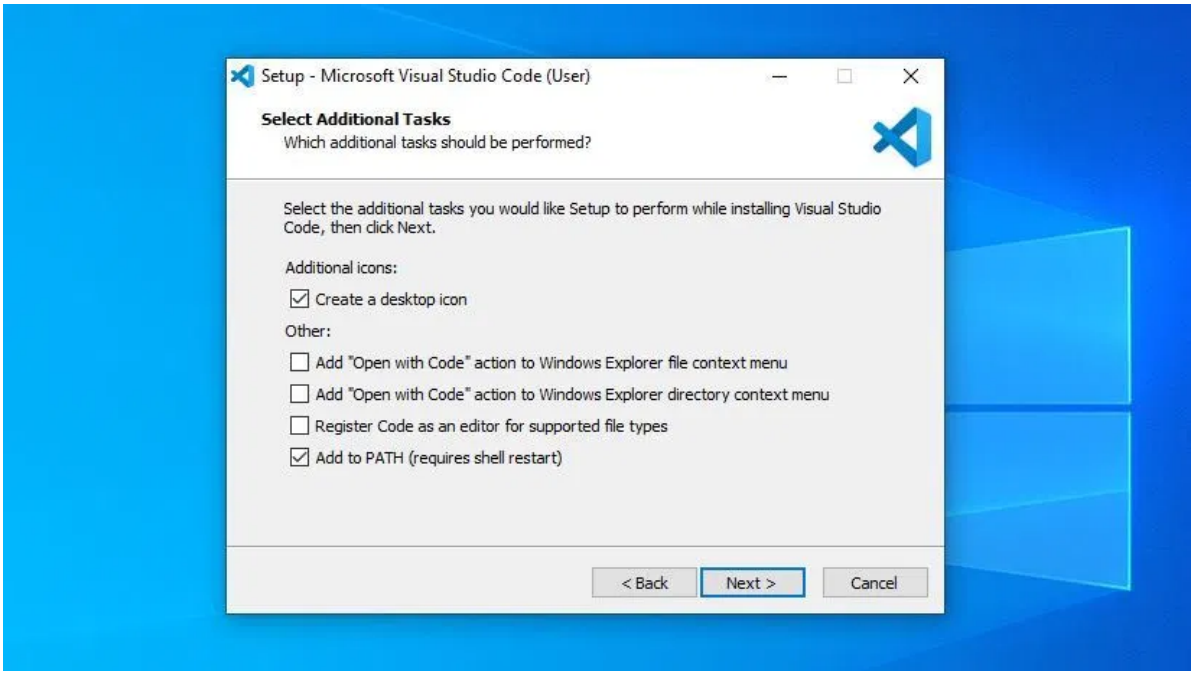


Step 3: Now select your installation location, where you want to install Visual Studio Code. If you don't have any good reason to change the installation location then keep it to default.

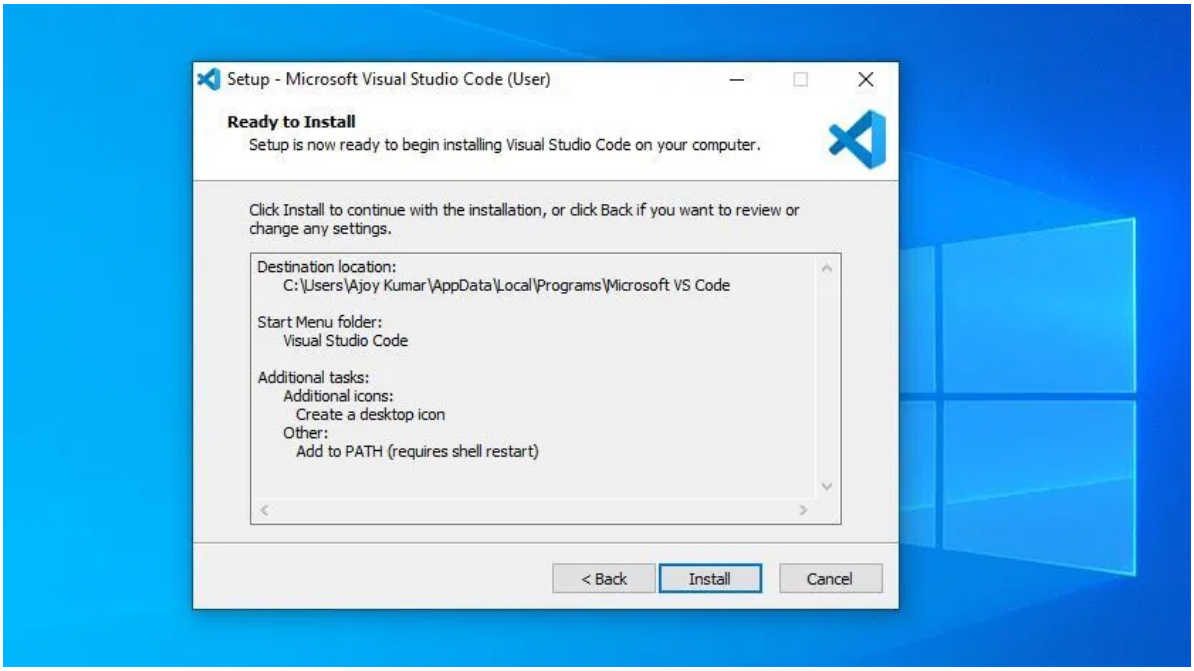


Step 4: After that select Start Menu Folder and add some additional tasks, "*create a desktop icon*" and "*Add to Path*".

> **Note:**– It is very important to add Visual Studio Code to PATH.



Step 5: Finally you are ready to install the VS Code on Windows 10. Just review all your selections then click on "Install".

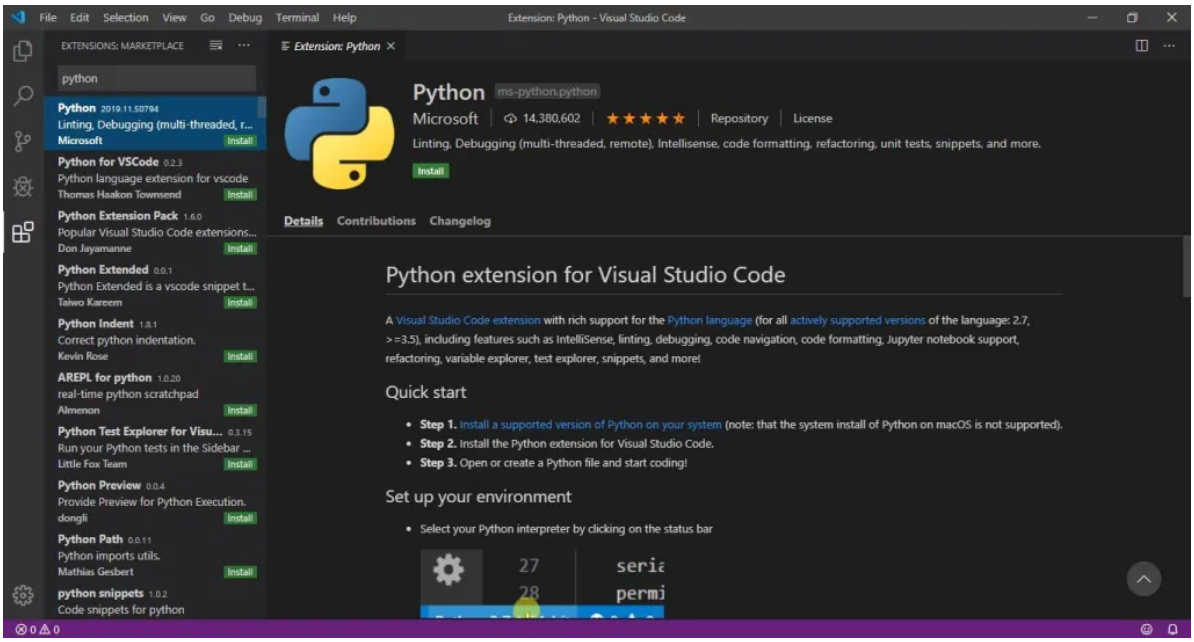


It will take some time to install so wait until the installation completed. After that, you're ready to use to Visual Studio Code.

Python Extension for VS Code

To install extensions from within Visual Studio Code:

1. Click on Extensions icon , search the extension you want to install. (If you know the name or part of the name of the extension, you can search in the **Search** window.)
2. Select the extension, review its Details, contributions, changelog and more.
3. Finally when you're ready to install the extension click on the "Install Button".



After installation complete reopen the Visual Studio Code.

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. You can typically 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 Python 3.7 and Python 3.8. Be aware that you might have compatibility issues with some packages using this method.

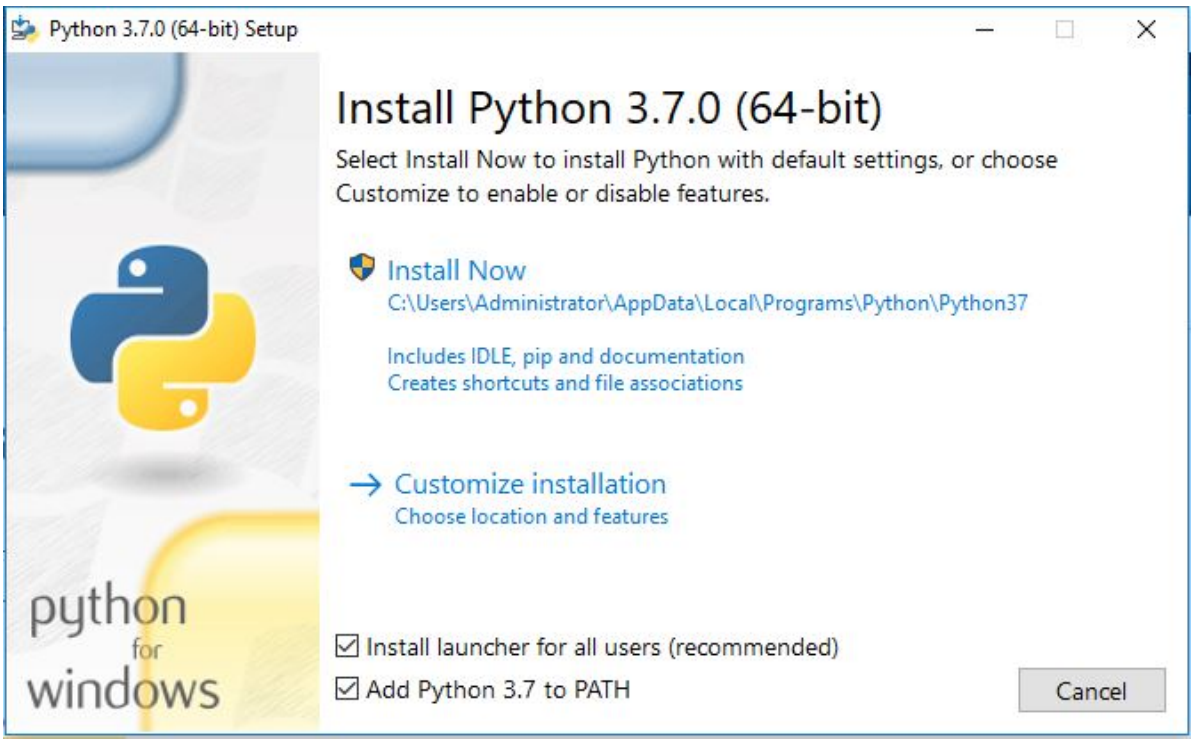
For additional information about using Python on Windows, see Using Python on Windows at Python.org

1. Download Python on Windows 10

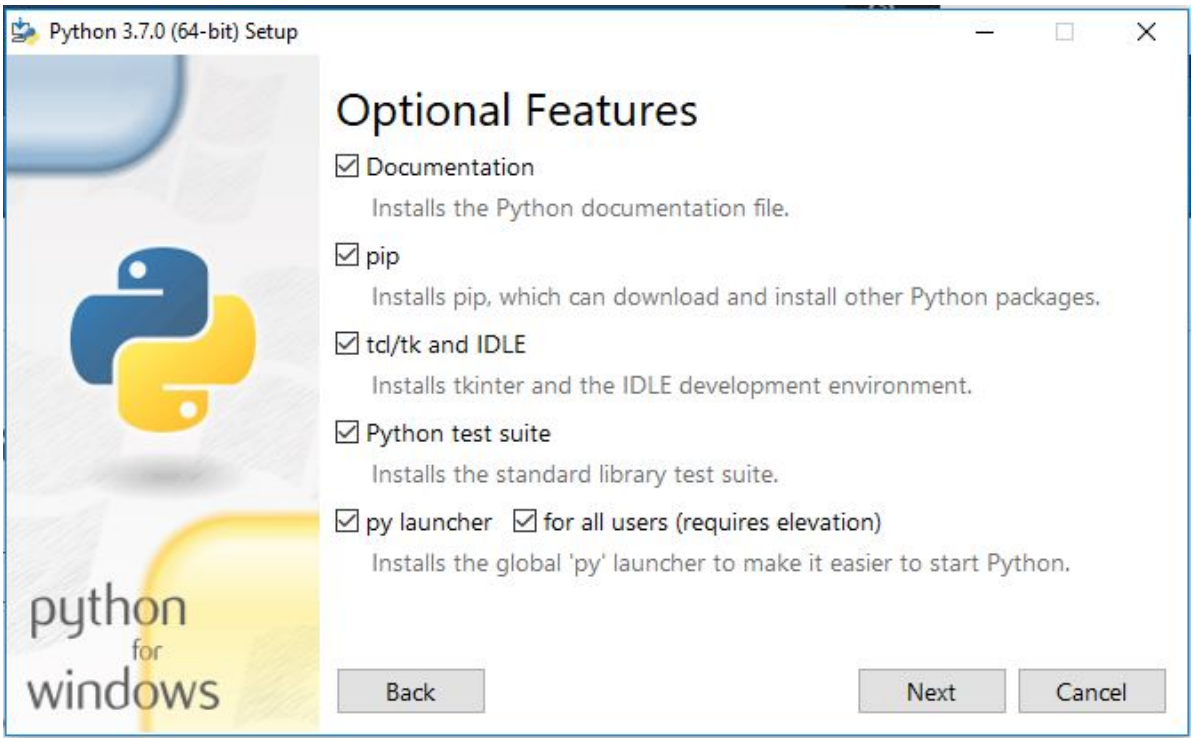
1. Click on the following link to download the Python for Windows Download Python for Windows 10, and you can select any one of the following, as shown in the below image. I chose the Windows x86-64 executable installer for the installation.

2. Install Python 3.7 on Windows 10

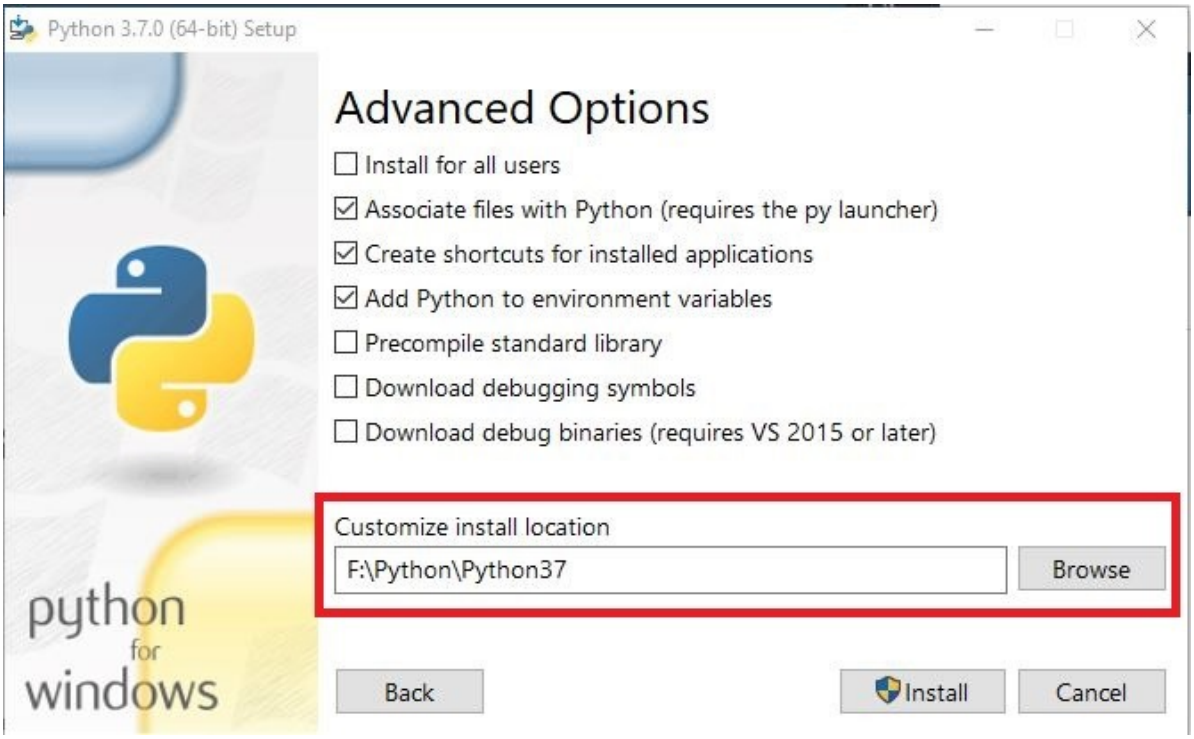
1. After completing the download, run the executable to start the installation for Python 3.7 on Windows 10. The first installation screen will display as shown in the below image. Select the check-boxes for install launcher for all users and Add Python 3.7 to PATH and click on the **Customize installation** option.



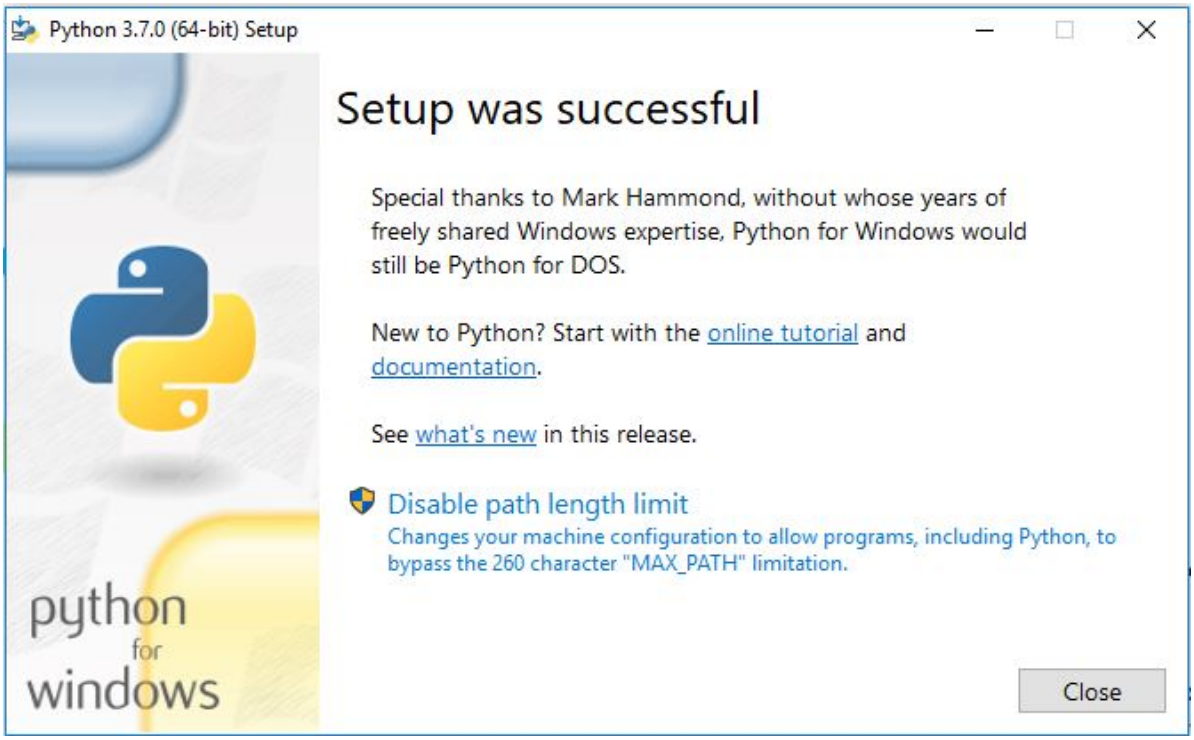
1. The installation customization screen will appear, as shown in below image. You can select all options and then Click on the Next button.



1. Specify the location of installation for Python, and select the check-boxes as per your need. Then click on the Install button.



1. The installation will be complete, and it will show the successful installation screen, as shown in the below image.



macOS

The system install of Python on macOS is not supported. Instead, an installation through Homebrew 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 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.

Install Visual Studio Code on Ubuntu 18.04

To install Visual Studio Code on your Ubuntu system, follow these steps:

1. First, update the packages index and install the dependencies by typing:

   ```
   sudo apt updatesudo apt install software-properties-common apt-transport-https wget
   ```

2. Next, import the Microsoft GPG key using the following wget command :

   ```
   wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -
   ```

   And enable the Visual Studio Code repository by typing:

   ```
   sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"
   ```

3. Once the apt repository is enabled , install the latest version of Visual Studio Code with:

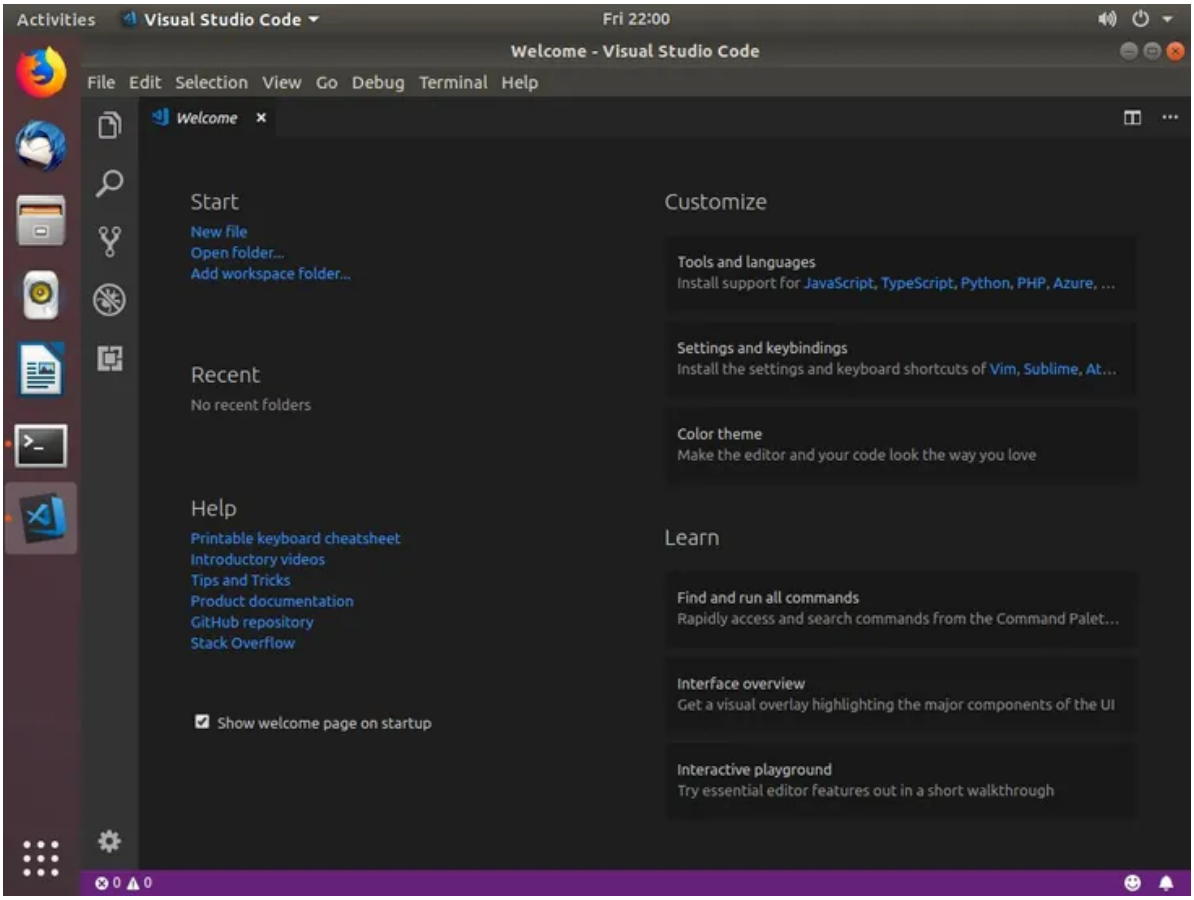
   ```
   sudo apt updatesudo apt install code
   ```

That's it. Visual Studio Code has been installed on your Ubuntu desktop and you can start using it.

Starting Visual Studio Code

Now that VS Code is installed on your Ubuntu system you can launch it either from the command line by typing `code` or by clicking on the VS Code icon (`Activities -> Visual Studio Code`).

When you start VS Code for the first time, a window like the following should appear:



You can now start installing extensions and configuring VS Code according to your preferences.

Updating Visual Studio Code

When a new version is released you can update the Visual Studio Code package through your desktop standard Software Update tool or by running the following commands in your terminal:

```
sudo apt updatesudo apt upgrade
```

Other options

- **Data Science**: If your primary purpose for using Python is Data Science, then you might consider a download from Anaconda. 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 (WSL) is an option for you. If you choose this option, you'll also want to install the Remote - WSL extension. For more information about using WSL with VS Code, see VS Code Remote Development or try the Working in WSL tutorial, which will walk you through setting up WSL, installing Python, and creating a Hello World application running in WSL.

Verify the Python installation

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
  ```

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

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

If the installation was successful, the output window should show the version of Python that you installed.

> **Note** 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 project (workspace) folder

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 .
```

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

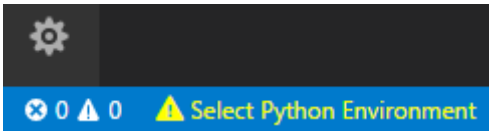
By starting VS Code in a folder, that folder becomes your "workspace". VS Code stores settings that are specific to that workspace in `.vscode/settings.json`, which are separate from user settings that are stored globally.

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

Select a Python interpreter

Python is an interpreted language, and in order to run Python code and get Python IntelliSense, you must tell VS Code which interpreter to use.

From within VS Code, select a Python 3 interpreter by opening the **Command Palette** (Ctrl+Shift+P), start typing the **Python: Select Interpreter** command to search, then select the command. You can also use the **Select Python Environment** option on the Status Bar if available (it may already show a selected interpreter, too):



The command presents a list of available interpreters that VS Code can find automatically, including virtual environments. If you don't see the desired interpreter, see Configuring Python environments.

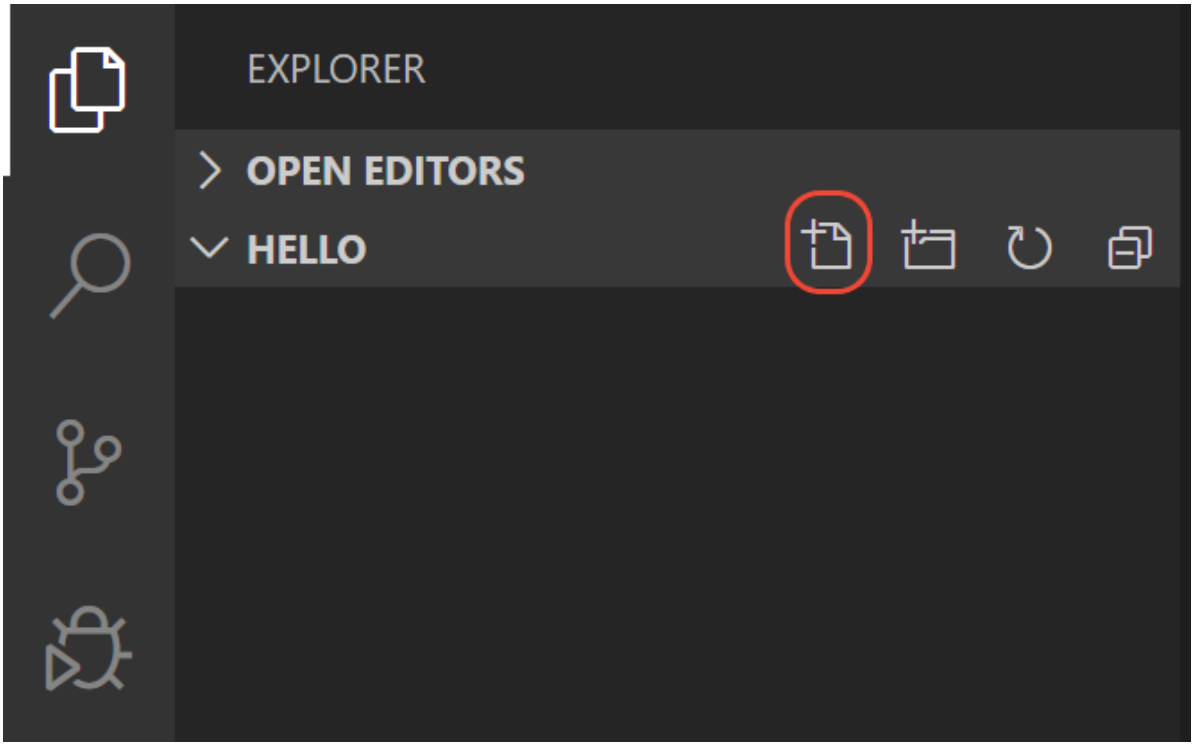
> **Note**: When using an Anaconda distribution, the correct interpreter should have the suffix `('base':conda)`, for example `Python 3.7.3 64-bit ('base':conda)`.

Selecting an interpreter sets the `python.pythonPath` value in your workspace settings to the path of the interpreter. To see the setting, select **File** > **Preferences** > **Settings** (**Code** > **Preferences** > **Settings** on macOS), then select the **Workspace Settings** tab.

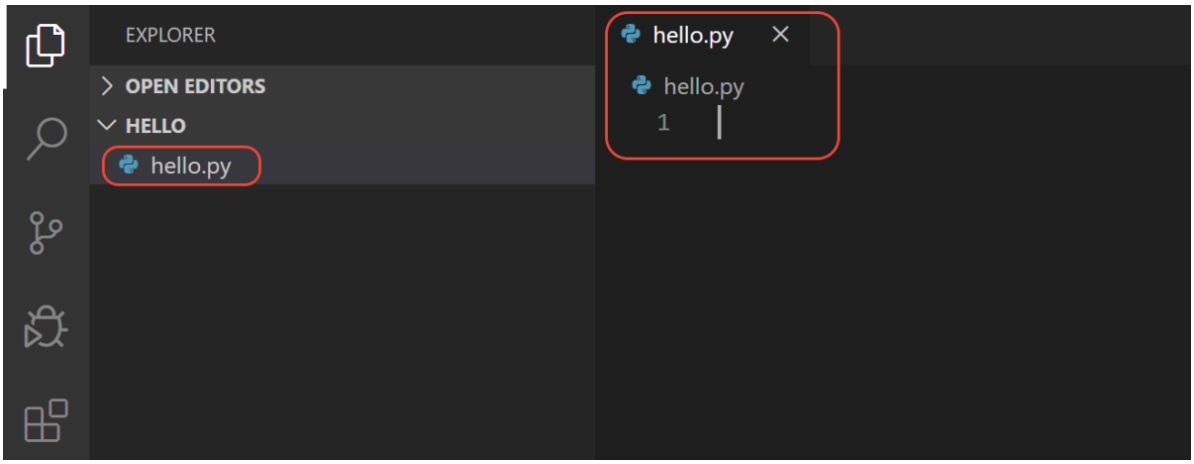
> **Note**: If you select an interpreter without a workspace folder open, VS Code sets `python.pythonPath` in your user settings instead, which sets the default interpreter for VS Code in general. The user setting makes sure you always have a default interpreter for Python projects. The workspace settings lets you override the user setting.

Create a Python Hello World source code file

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



Name the file `hello.py`, and it automatically opens in the editor:



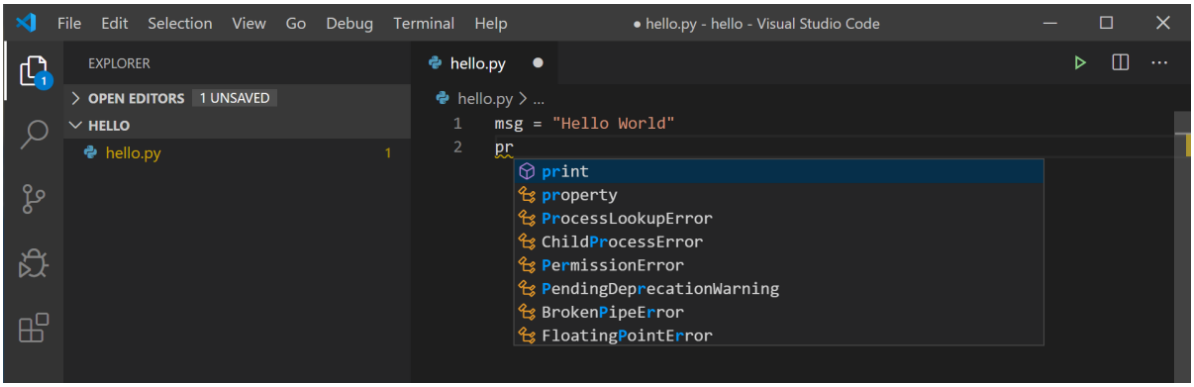
By using the `.py` file extension, you tell VS Code to interpret this file as a Python program, so that it evaluates the contents with the Python extension and the selected interpreter.

> **Note**: The File Explorer toolbar also allows you to create folders within your workspace to better organize your code. You can use the **New folder** button to quickly create a folder.

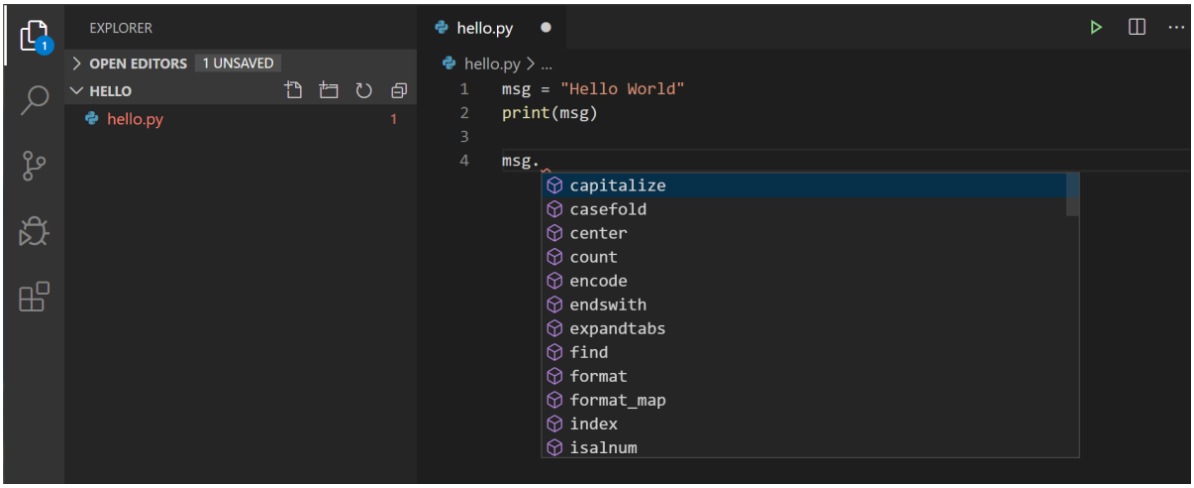
Now that you have a code file in your Workspace, enter the following source code in `hello.py`:

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

When you start typing `print`, notice how IntelliSense presents auto-completion options.



IntelliSense and auto-completions work for standard Python modules as well as other packages you've installed into the environment of the selected Python interpreter. It also provides completions for methods available on object types. For example, because the `msg` variable contains a string, IntelliSense provides string methods when you type `msg.`:

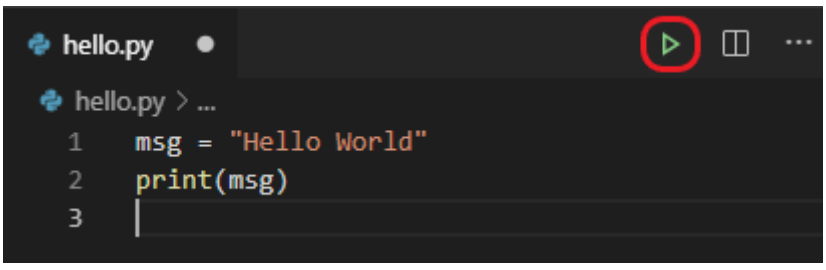


Feel free to experiment with IntelliSense some more, but then revert your changes so you have only the `msg` variable and the `print` call, and save the file (Ctrl+S).

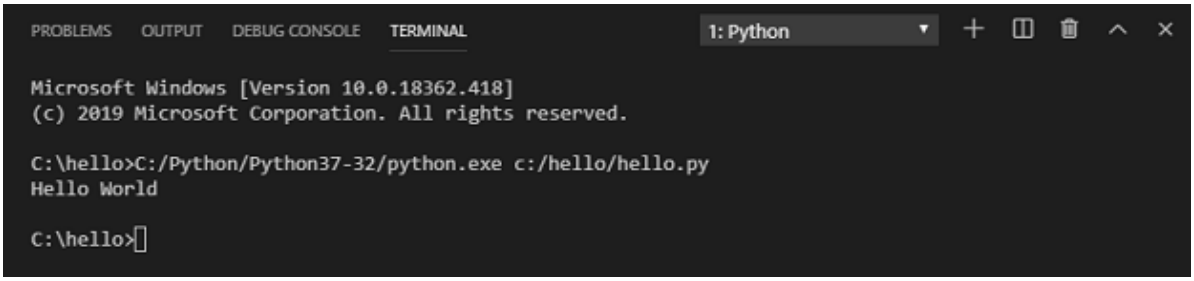
For full details on editing, formatting, and refactoring, see Editing code. The Python extension also has full support for Linting.

Run Hello World

It's simple to run `hello.py` with Python. Just click the **Run Python File in Terminal** play button in the top-right side of the editor.

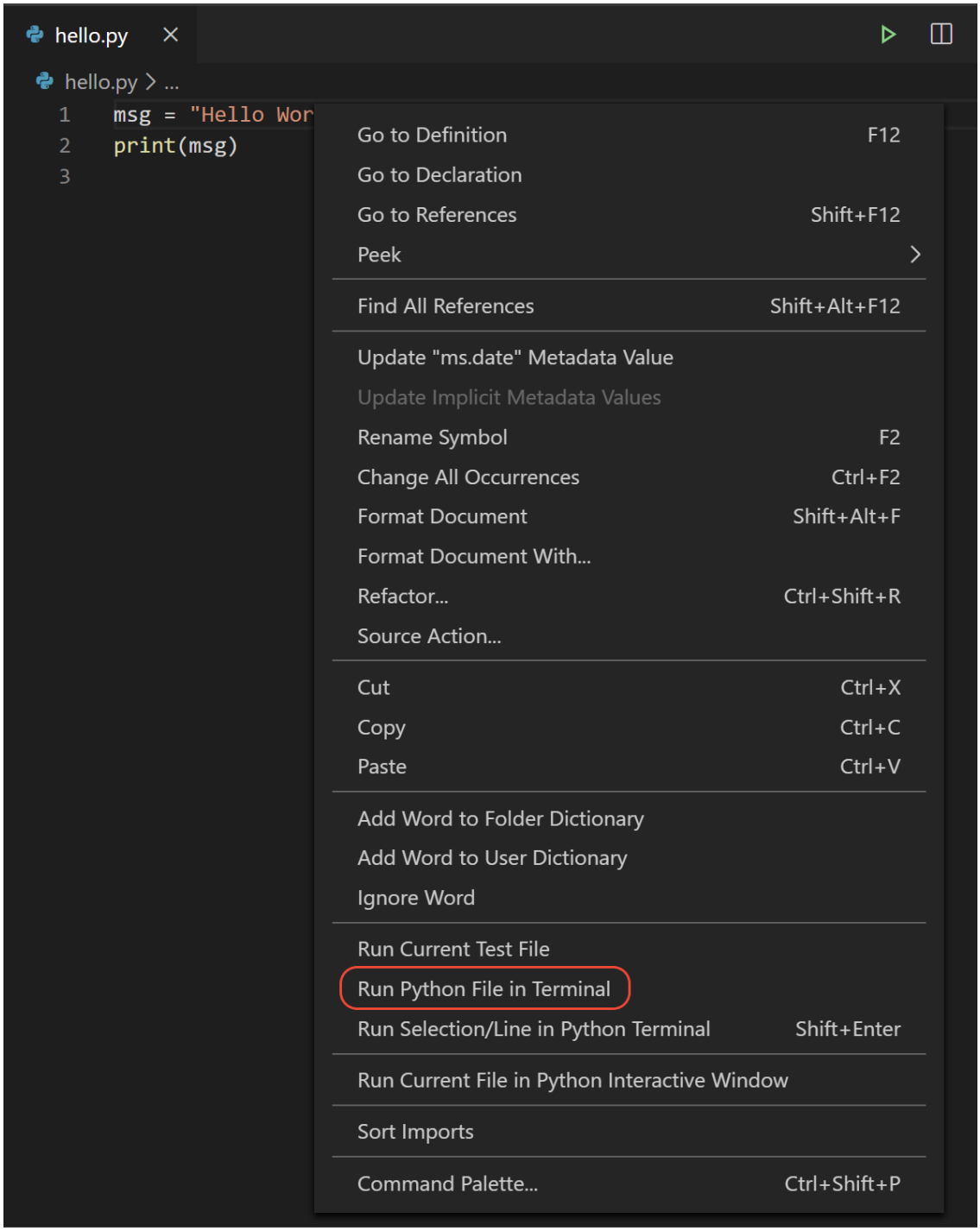


The button opens a terminal panel in which your Python interpreter is automatically activated, then runs `python3 hello.py` (macOS/Linux) or `python hello.py` (Windows):



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

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

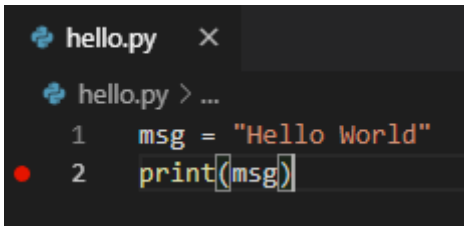


- Select one or more lines, then press Shift+Enter or right-click and select **Run Selection/Line in Python Terminal**. This command is convenient for testing just a part of a file.
- From the Command Palette (Ctrl+Shift+P), select the **Python: Start REPL** command to open a REPL terminal for the currently selected Python interpreter. In the REPL, you can then enter and run lines of code one at a time.

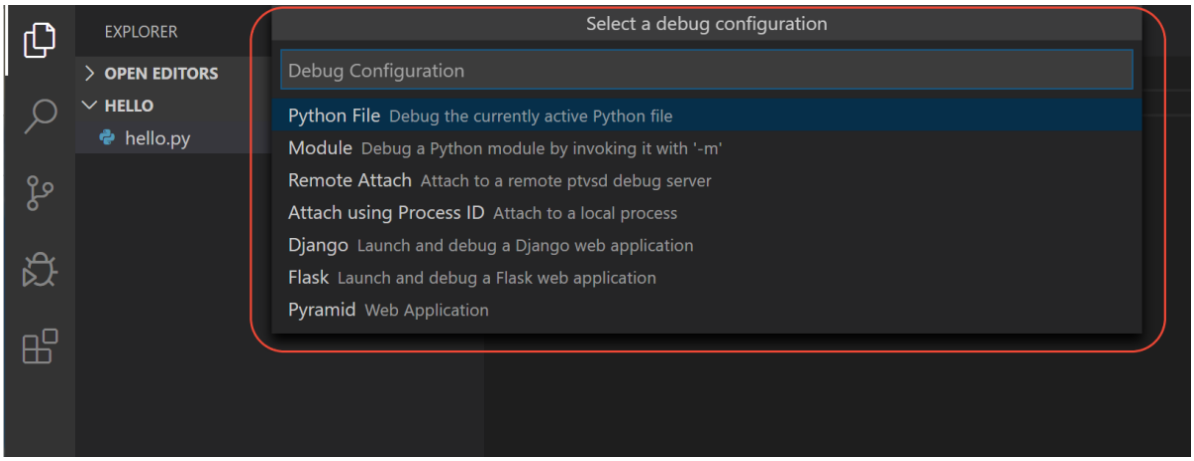
Configure and run the debugger

Let's now try debugging our simple Hello World program.

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



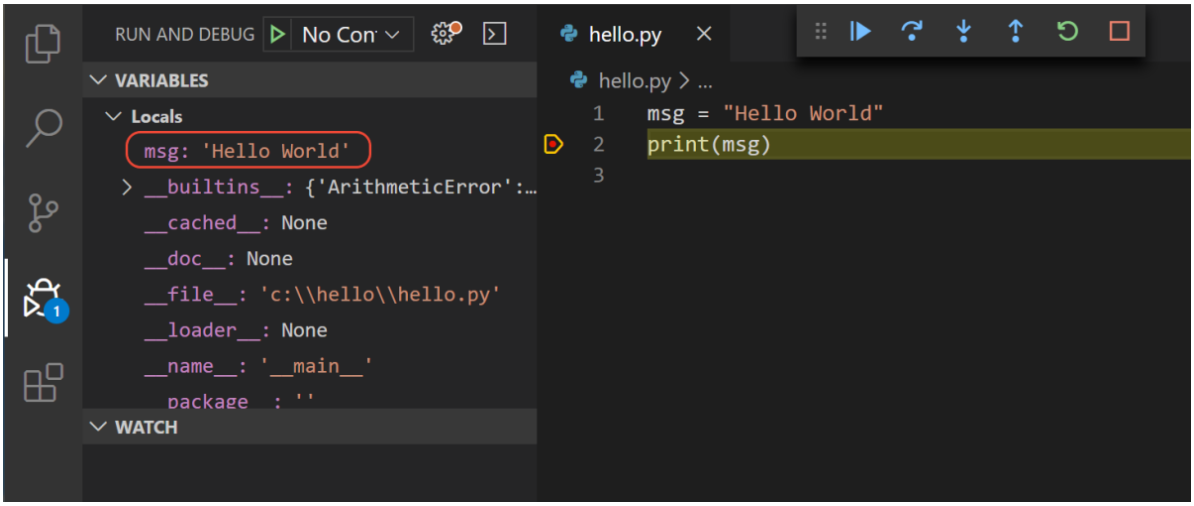
Next, to initialize the debugger, press F5. Since this is your first time debugging this file, a configuration menu will open from the Command Palette allowing you to select the type of debug configuration you would like for the opened file.



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

These different configurations are fully explained in Debugging configurations; for now, just 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 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 will see now defined `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 (Shift+F11), restart (Ctrl+Shift+F5), and stop (Shift+F5).

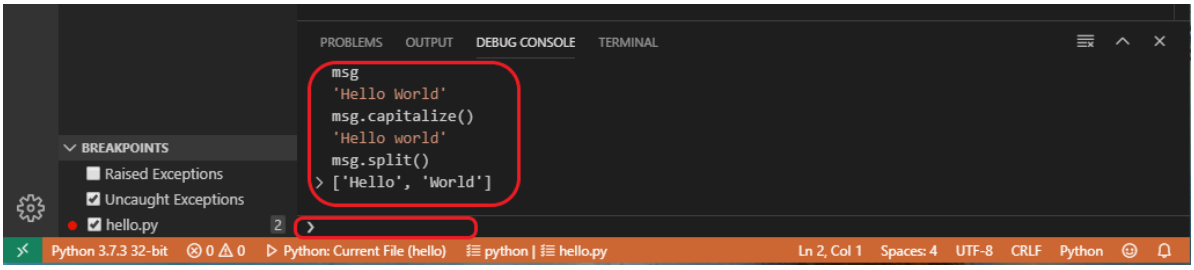
The Status Bar also changes color (orange in many themes) to indicate that you're in debug mode. The **Python Debug Console** also appears automatically in the lower right panel to show the commands being run, along with the program output.

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 `Hello world` 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()
```



Select the blue **Continue** button on the toolbar again (or press F5) to run the program to completion. "Hello World" 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 (Shift+F5), or use the **Run > Stop debugging** menu command.

For full details, see Debugging configurations, 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 in the main VS Code debugging article.

Install and use packages

Let's now run an example that's a little more interesting. In Python, packages are how you obtain any number of useful code libraries, typically from PyPI. For this example, you use the `matplotlib` and `numpy` packages to create a graphical plot as is commonly done with data science. (Note that `matplotlib` cannot show graphs when running in the Windows Subsystem for Linux as it lacks the necessary UI support.)

Return to the **Explorer** view (the top-most icon on the left side, which shows files), create a new file called `standardplot.py`, and paste in the following source code:

```
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
```

> **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, try running the file in the debugger using the "Python: Current file" configuration as described in the last section.

Unless you're using an Anaconda distribution or have previously installed the `matplotlib` package, you should see the message, **"ModuleNotFoundError: No module named 'matplotlib'"**. Such a message indicates that the required package isn't available in your system.

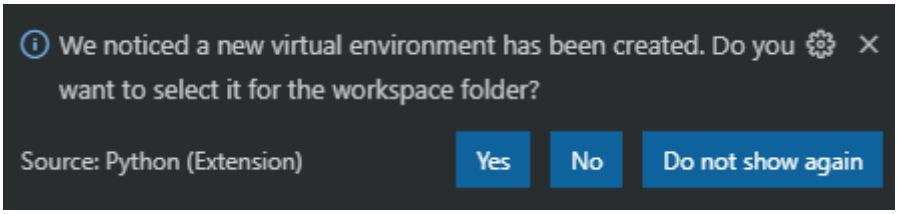
To install the `matplotlib` package (which also installs `numpy` as a dependency), stop the debugger and use the Command Palette to run **Terminal: Create New Integrated Terminal** (Ctrl+Shift+`). This command opens a command prompt for your selected interpreter.

A best practice among Python developers is to avoid installing packages into a global interpreter environment. You instead use a project-specific `virtual environment` that contains a copy of a global interpreter. Once you activate that environment, any packages you then install are isolated from other environments. Such isolation reduces many complications that can arise from conflicting package versions. To create a *virtual environment* and install the required packages, enter the following commands as appropriate for your operating system:

> **Note**: For additional information about virtual environments, see Environments.

1. Create and activate the virtual environment

   > **Note**: When you create a new virtual environment, you should be prompted by VS Code to set it as the default for your workspace folder. If selected, the environment will automatically be activated when you open a new terminal.

   For windows

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

If the activate command generates the message "Activate.ps1 is not digitally signed. You cannot run this script on the current system.", then you need to temporarily change the **PowerShell** execution policy to allow scripts to run (see About Execution Policies in the PowerShell documentation):

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

For macOS/Linux

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

2. Select your new environment by using the **Python: Select Interpreter** command from the **Command Palette**.
3. Install the packages

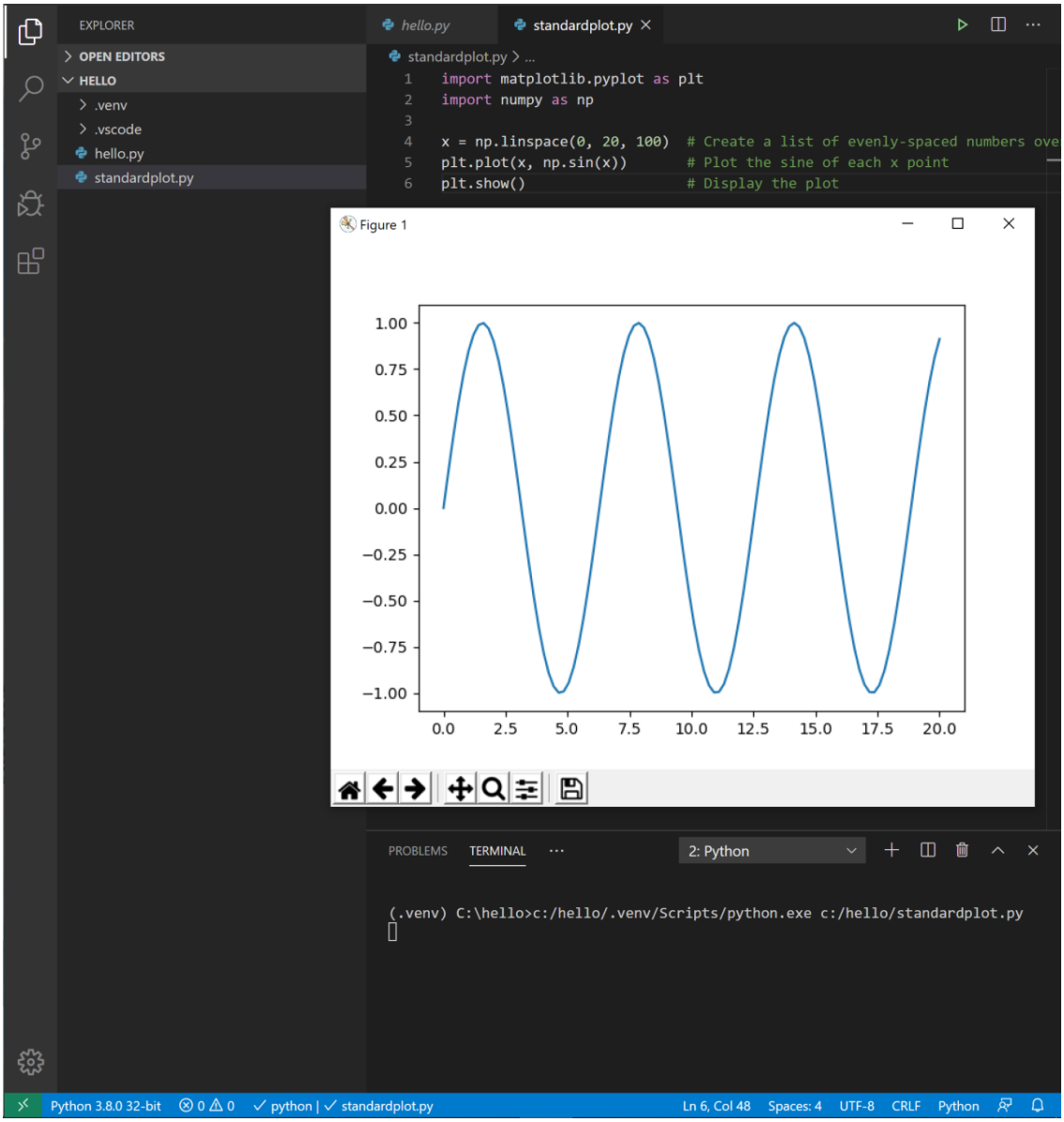
```
# Don't use with Anaconda distributions because they include matplotlib
already.

# macOS
python3 -m pip install matplotlib

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

# Linux (ubuntu18.06)
sudo apt-get install python3-tk
pip3 install matplotlib
```

4. Rerun the program now (with or without the debugger) and after a few moments a plot window appears with the output:



5. Once you are finished, type `deactivate` in the terminal window to deactivate the virtual environment.