# Contributing to the Python extension for Visual Studio Code

---

| `release` branch | `master` branch | Nightly CI | coverage (`master` branch) |

| ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- | ----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------- | ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------ | ---------------------------------------------------------------------------------------------------------------------------------------- |

| [![Build Status](https://dev.azure.com/ms/vscode-python/\_apis/build/status/CI?branchName=release)](https://dev.azure.com/ms/vscode-python/\_build/latest?definitionId=88&branchName=release) | [![Build Status](https://dev.azure.com/ms/vscode-python/\_apis/build/status/CI?branchName=master)](https://dev.azure.com/ms/vscode-python/\_build/latest?definitionId=88&branchName=master) | [![Build Status](https://dev.azure.com/ms/vscode-python/\_apis/build/status/Nightly%20Build?branchName=master)](https://dev.azure.com/ms/vscode-python/\_build/latest?definitionId=85&branchName=master) | [![codecov](https://codecov.io/gh/microsoft/vscode-python/branch/master/graph/badge.svg)](https://codecov.io/gh/microsoft/vscode-python) |

[[Development build](https://pvsc.blob.core.windows.net/extension-builds/ms-python-insiders.vsix)]

---

[For contributing to the [Microsoft Python Language Server](https://github.com/Microsoft/python-language-server) see its own repo; for [Pylance](https://github.com/microsoft/pylance-release) see its own repo; for [debugpy](https://github.com/microsoft/debugpy) see its own repo]

## Contributing a pull request

### Prerequisites

1. [Node.js](https://nodejs.org/) 12.15

1. [Python](https://www.python.org/) 2.7 or later

1. Windows, macOS, or Linux

1. [Visual Studio Code](https://code.visualstudio.com/)

1. The following VS Code extensions:

- [TSLint](https://marketplace.visualstudio.com/items?itemName=ms-vscode.vscode-typescript-tslint-plugin)

- [Prettier](https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode)

- [EditorConfig for VS Code](https://marketplace.visualstudio.com/items?itemName=EditorConfig.EditorConfig)

1. Have an issue which has a "needs PR" label (feel free to indicate you would like to provide a PR for the issue so others don't work on it as well)

### Setup

```shell

git clone https://github.com/microsoft/vscode-python

cd vscode-python

npm ci

python3 -m venv .venv

# Activate the virtual environment as appropriate for your shell, For example, on bash it's ...

source .venv/bin/activate

# The Python code in the extension is formatted using Black.

python3 -m pip install black

# Install Python dependencies using `python3`.

# If you want to use a different interpreter then specify it in the

# CI\_PYTHON\_PATH environment variable.

npx gulp installPythonLibs

```

If you see warnings that `The engine "vscode" appears to be invalid.`, you can ignore these.

### Incremental Build

Run the `Compile` and `Hygiene` build Tasks from the [Run Build Task...](https://code.visualstudio.com/docs/editor/tasks) command picker (short cut `CTRL+SHIFT+B` or `??B`). This will leave build and hygiene tasks running in the background and which will re-run as files are edited and saved. You can see the output from either task in the Terminal panel (use the selector to choose which output to look at).

You can also compile from the command-line. For a full compile you can use:

```shell

npx gulp prePublishNonBundle

```

For incremental builds you can use the following commands depending on your needs:

```shell

npm run compile

npm run compile-webviews-watch # For data science (React Code)

```

Sometimes you will need to run `npm run clean` and even `rm -r out`.

This is especially true if you have added or removed files.

### Errors and Warnings

TypeScript errors and warnings will be displayed in the `Problems` window of Visual Studio Code.

### Run dev build and validate your changes

To test changes, open the `vscode-python` folder in VSCode, and select the workspace titled `vscode-python`.

Then, open the debug panel by clicking the `Run and Debug` icon on the sidebar, select the `Extension`

option from the top menu, and click start. A new window will launch with the title

`[Extension Development Host]`.

### Running Unit Tests

Note: Unit tests are those in files with extension `.unit.test.ts`.

1. Make sure you have compiled all code (done automatically when using incremental building)

1. Ensure you have disabled breaking into 'Uncaught Exceptions' when running the Unit Tests

1. For the linters and formatters tests to pass successfully, you will need to have those corresponding Python libraries installed locally

1. Run the Tests via the `Unit Tests` launch option.

You can also run them from the command-line (after compiling):

```shell

npm run test:unittests # runs all unit tests

npm run test:unittests -- --grep='<NAME-OF-SUITE>'

```

\_To run only a specific test suite for unit tests:\_

Alter the `launch.json` file in the `"Debug Unit Tests"` section by setting the `grep` field:

```js

"args": [

"--timeout=60000",

"--grep", "<suite name>"

],

```

...this will only run the suite with the tests you care about during a test run (be sure to set the debugger to run the `Debug Unit Tests` launcher).

### Running Functional Tests

Functional tests are those in files with extension `.functional.test.ts`.

These tests are similar to system tests in scope, but are run like unit tests.

You can run functional tests in a similar way to that for unit tests:

- via the "Functional Tests" launch option, or

- on the command line via `npm run test:functional`

### Running System Tests

Note: System tests are those in files with extension `.test\*.ts` but which are neither `.functional.test.ts` nor `.unit.test.ts`.

1. Make sure you have compiled all code (done automatically when using incremental building)

1. Ensure you have disabled breaking into 'Uncaught Exceptions' when running the Unit Tests

1. For the linters and formatters tests to pass successfully, you will need to have those corresponding Python libraries installed locally by using the `./requirements.txt` and `build/test-requirements.txt` files

1. Run the tests via `npm run` or the Debugger launch options (you can "Start Without Debugging").

1. \*\*Note\*\* you will be running tests under the default Python interpreter for the system.

You can also run the tests from the command-line (after compiling):

```shell

npm run testSingleWorkspace # will launch the VSC UI

npm run testMultiWorkspace # will launch the VSC UI

```

#### Customising the Test Run

If you want to change which tests are run or which version of Python is used,

you can do this by setting environment variables. The same variables work when

running from the command line or launching from within VSCode, though the

mechanism used to specify them changes a little.

\* Setting `CI\_PYTHON\_PATH` lets you change the version of python the tests are executed with

\* Setting `VSC\_PYTHON\_CI\_TEST\_GREP` lets you filter the tests by name

\_`CI\_PYTHON\_PATH`\_

In some tests a Python executable is actually run. The default executable is

`python` (for now). Unless you've run the tests inside a virtual environment,

this will almost always mean Python 2 is used, which probably isn't what you

want.

By setting the `CI\_PYTHON\_PATH` environment variable you can

control the exact Python executable that gets used. If the executable

you specify isn't on `$PATH` then be sure to use an absolute path.

This is also the mechanism for testing against other versions of Python.

\_`VSC\_PYTHON\_CI\_TEST\_GREP`\_

This environment variable allows providing a regular expression which will

be matched against suite and test "names" to be run. By default all tests

are run.

For example, to run only the tests in the `Sorting` suite (from

[`src/test/format/extension.sort.test.ts`](https://github.com/Microsoft/vscode-python/blob/84f9c7a174111/src/test/format/extension.sort.test.ts))

you would set the value to `Sorting`. To run the `ProcessService` and

`ProcessService Observable` tests which relate to `stderr` handling, you might

use the value `ProcessService.\*stderr`.

Be sure to escape any grep-sensitive characters in your suite name.

In some rare cases in the "system" tests the `VSC\_PYTHON\_CI\_TEST\_GREP`

environment variable is ignored. If that happens then you will need to

temporarily modify the `const grep = ` line in

[`src/test/index.ts`](https://github.com/Microsoft/vscode-python/blob/84f9c7a174111/src/test/index.ts#L64).

\_Launching from VSCode\_

In order to set environment variables when launching the tests from VSCode you

should edit the `launch.json` file. For example you can add the following to the

appropriate configuration you want to run to change the interpreter used during

testing:

```js

"env": {

"CI\_PYTHON\_PATH": "/absolute/path/to/interpreter/of/choice/python"

}

```

\_On the command line\_

The mechanism to set environment variables on the command line will vary based

on your system, however most systems support a syntax like the following for

setting a single variable for a subprocess:

```shell

VSC\_PYTHON\_CI\_TEST\_GREP=Sorting npm run testSingleWorkspace

```

### Testing Python Scripts

The extension has a number of scripts in ./pythonFiles. Tests for these

scripts are found in ./pythonFiles/tests. To run those tests:

- `python2.7 pythonFiles/tests/run\_all.py`

- `python3 -m pythonFiles.tests`

By default, functional tests are included. To exclude them:

`python3 -m pythonFiles.tests --no-functional`

To run only the functional tests:

`python3 -m pythonFiles.tests --functional`

### Standard Debugging

Clone the repo into any directory, open that directory in VSCode, and use the `Extension` launch option within VSCode.

### Debugging the Python Extension Debugger

The easiest way to debug the Python Debugger (in our opinion) is to clone this git repo directory into [your](https://code.visualstudio.com/docs/extensions/install-extension#\_your-extensions-folder) extensions directory.

From there use the `Extension + Debugger` launch option.

### Coding Standards

Information on our coding standards can be found [here](https://github.com/Microsoft/vscode-python/blob/master/CODING\_STANDARDS.md).

We have CI tests to ensure the code committed will adhere to the above coding standards. \\*You can run this locally by executing the command `npx gulp precommit` or use the `precommit` Task.

Messages displayed to the user must be localized using/created constants from/in the [localize.ts](https://github.com/Microsoft/vscode-python/blob/master/src/client/common/utils/localize.ts) file.

## Development process

To effectively contribute to this extension, it helps to know how its

development process works. That way you know not only why the

project maintainers do what they do to keep this project running

smoothly, but it allows you to help out by noticing when a step is

missed or to learn in case someday you become a project maintainer as

well!

### Helping others

First and foremost, we try to be helpful to users of the extension.

We monitor

[Stack Overflow questions](https://stackoverflow.com/questions/tagged/visual-studio-code+python)

to see where people might need help. We also try to respond to all

issues in some way in a timely manner (typically in less than one

business day, definitely no more than a week). We also answer

questions that reach us in other ways, e.g. Twitter.

For pull requests, we aim to review any externally contributed PR no later

than the next sprint from when it was submitted (see

[Release Cycle](#release-cycle) below for our sprint schedule).

### Release cycle

Planning is done as one week sprints. We start a sprint every Thursday.

All [P0](https://github.com/Microsoft/vscode-python/labels/P0) issues are expected

to be fixed in the current sprint, else the next release will be blocked.

[P1](https://github.com/Microsoft/vscode-python/labels/P1) issues are a

top-priority and we try to close before the next release. All other issues are

considered best-effort for that sprint.

The extension aims to do a new release once a month. A

[release plan](https://github.com/Microsoft/vscode-python/labels/release%20plan)

is created for each release to help track anything that requires a

person to do (long-term this project aims to automate as much of the

development process as possible).

All development is actively done in the `master` branch of the

repository. This allows us to have a

[development build](#development-build) which is expected to be stable at

all times. Once we reach a release candidate, it becomes

our [release branch](https://github.com/microsoft/vscode-python/branches).

At that point only what is in the release branch will make it into the next

release.

### Issue triaging

#### Classifying issues

To help actively track what stage

[issues](https://github.com/Microsoft/vscode-python/issues)

are at, various labels are used. The following label types are expected to

be set on all open issues (otherwise the issue is not considered triaged):

1. `needs`/`triage`/`classify`

1. `feature`

1. `type`

These labels cover what is blocking the issue from closing, what is affected by

the issue, and what kind of issue it is. (The `feature` label should be `feature-\*` if the issue doesn't fit into any other `feature` label appropriately.)

It is also very important to make the title accurate. People often write very brief, quick titles or ones that describe what they think the problem is. By updating the title to be appropriately descriptive for what \_you\_ think the issue is, you not only make finding older issues easier, but you also help make sure that you and the original reporter agree on what the issue is.

#### Post-classification

Once an issue has been appropriately classified, there are two keys ways to help out. One is to go through open issues that

have a merged fix and verify that the fix did in fact work. The other is to try to fix issues marked as `needs PR`.

### Pull requests

Key details that all pull requests are expected to handle should be

in the [pull request template](https://github.com/Microsoft/vscode-python/blob/master/.github/PULL\_REQUEST\_TEMPLATE.md). We do expect CI to be passing for a pull request before we will consider merging it.

### Versioning

Starting in 2018, the extension switched to

[calendar versioning](http://calver.org/) since the extension

auto-updates and thus there is no need to track its version

number for backwards-compatibility. In 2020, the extension switched to

having the the major version be the year of release, the minor version the

release count for that year, and the build number is a number that increments

for every build.

For example the first release made in 2020 is `2020.1.<build number>`.

## Releasing

Overall steps for releasing are covered in the

[release plan](https://github.com/Microsoft/vscode-python/labels/release%20plan)

([template](https://github.com/Microsoft/vscode-python/blob/master/.github/release\_plan.md)).

### Building a release

To create a release \_build\_, follow the steps outlined in the [release plan](https://github.com/Microsoft/vscode-python/labels/release%20plan) (which has a [template](https://github.com/Microsoft/vscode-python/blob/master/.github/release\_plan.md)).

## Local Build

Steps to build the extension on your machine once you've cloned the repo:

```bash

> npm install -g vsce

# Perform the next steps in the vscode-python folder.

> npm ci

> python3 -m pip --disable-pip-version-check install -t ./pythonFiles/lib/python --no-cache-dir --implementation py --no-deps --upgrade -r requirements.txt

# For python 3.6 and lower use this command to install the debugger

> python3 -m pip --disable-pip-version-check install -t ./pythonFiles/lib/python --no-cache-dir --implementation py --no-deps --upgrade --pre debugpy

# For python 3.7 and greater use this command to install the debugger

> python3 -m pip --disable-pip-version-check install -r build/debugger-install-requirements.txt

> python3 ./pythonFiles/install\_debugpy.py

> npm run clean

> npm run package # This step takes around 10 minutes.

```

Resulting in a `ms-python-insiders.vsix` file in your `vscode-python` folder.

?? If you made changes to `package.json`, run `npm install` (instead of `npm ci`) to update `package-lock.json` and install dependencies all at once.

## Development Build

If you would like to use the latest version of the extension as committed to `master` that has passed our test suite, then you may set the `"python.insidersChannel"` setting to `"daily"` or `"weekly"` based on how often you would like the extension to check for updates.

You may also download and install the extension manually from the following

[location](https://pvsc.blob.core.windows.net/extension-builds/ms-python-insiders.vsix).

Once you have downloaded the

[ms-python-insiders.vsix](https://pvsc.blob.core.windows.net/extension-builds/ms-python-insiders.vsix)

file, please follow the instructions on

[this page](https://code.visualstudio.com/docs/editor/extension-gallery#\_install-from-a-vsix)

to install the extension. Do note that the manual install will not automatically update to newer builds unless you set the `"python.insidersChannel"` setting (it will get replaced with released versions from the Marketplace once they are newer than the version install manually).