# Contributing to Spyder

:+1::tada: First off, thanks for taking the time to contribute to Spyder! :tada::+1:

## General Guidelines

This page documents at a very high level how to contribute to Spyder. Please check the [Spyder IDE Contributor Documentation](https://github.com/spyder-ide/spyder/wiki/Contributing-to-Spyder) for a more detailed guide on how to do so.

Also, make sure you're familiar with our [Github workflow](https://github.com/spyder-ide/spyder/wiki/Dev:-Github-Workflow).

## Troubleshooting

Before posting a report, \*please\* carefully read our \*\*[Troubleshooting Guide](https://github.com/spyder-ide/spyder/wiki/Troubleshooting-Guide-and-FAQ)\*\* and search the [issue tracker](https://github.com/spyder-ide/spyder/issues) for your error message and problem description, as the great majority of bugs are either duplicates, or can be fixed on the user side with a few easy steps. Thanks!

## Submitting a Helpful Issue

Submitting useful, effective and to-the-point issue reports can go a long way toward improving Spyder for everyone. Accordingly, please read the [relevant section](https://github.com/spyder-ide/spyder/wiki/Troubleshooting-Guide-and-FAQ#calling-for-help-still-have-a-problem) of the Spyder Troubleshooting Guide, which describes in detail how to do just that.

Most importantly, aside from the error message/traceback and the requested environment/dependency information, \*please\* be sure you include a detailed, step by step description of exactly what triggered the problem. Otherwise, we likely won't be able to find and fix it, and your issue will have to be closed after a week (7 days). Thanks!

## Setting Up a Development Environment

### Forking and cloning the repo

First, navigate to the [Spyder repo](https://github.com/spyder-ide/spyder) in your web browser and press the ``Fork`` button to make a personal copy of the repository on your own Github account.

Then, click the ``Clone or Download`` button on your repository, copy the link and run the following on the command line to clone the repo:

```bash

$ git clone <LINK-TO-YOUR-REPO>

```

Finally, set the upstream remote to the official Spyder repo with:

```bash

$ git remote add upstream https://github.com/spyder-ide/spyder.git

```

### Creating a conda environment or virtualenv

If you use Anaconda you can create a conda environment with the following commands:

```bash

$ conda create -n spyder-dev python=3

$ conda activate spyder-dev

```

You can also use `virtualenv` on Linux, but `conda` is \*\*strongly\*\* recommended:

```bash

$ mkvirtualenv spyder-dev

$ workon spyder-dev

```

### Installing dependencies

After you have created your development environment, you need to install Spyder's necessary dependencies. The easiest way to do so (with Anaconda) is

```bash

$ conda install -c spyder-ide/label/alpha --file requirements/conda.txt

```

This installs all Spyder's dependencies into the environment.

If using `pip` and `virtualenv` (not recommended), you need to `cd` to the directory where your git clone is stored and run:

```bash

$ pip install -e .

```

### Running Spyder

To start Spyder directly from your clone, i.e. without installing it into your environment, you need to run (from the directory you cloned it to e.g. `spyder`):

```bash

$ python bootstrap.py

```

To start Spyder in debug mode, useful for tracking down an issue, you can run:

```bash

$ python bootstrap.py --debug

```

\*\*Important Note\*\*: To test any changes you've made to the Spyder source code, you need to restart Spyder or start a fresh instance (you can run multiple copies simultaneously by unchecking the Preferences option <kbd>Use a single instance</kbd> under <kbd>General</kbd> > <kbd>Advanced Settings</kbd> .

### Running tests

To install our test dependencies under Anaconda:

```bash

$ conda install -c spyder-ide --file requirements/tests.txt

```

If using `pip` (for experts only), run the following from the directory where your git clone is stored:

```bash

$ pip install -e .[test]

```

To run the Spyder test suite, please use (from the `spyder` root directory):

```bash

$ python runtests.py

```

## Spyder Branches

When you start to work on a new pull request (PR), you need to be sure that your work is done on top of the correct Spyder branch, and that you base your PR on Github against it.

To guide you, issues on Github are marked with a milestone that indicates the correct branch to use. If not, follow these guidelines:

\* Use the `4.x` branch for bugfixes only (\*e.g.\* milestones `v4.0.1` or `v4.1.2`)

\* Use `master` to introduce new features or break compatibility with previous Spyder versions (\*e.g.\* milestones `v5.0beta1` or `v5.0beta2`).

You should also submit bugfixes to `4.x` or `master` for errors that are only present in those respective branches.

To start working on a new PR, you need to execute these commands, filling in the branch names where appropriate:

```bash

$ git checkout <SPYDER-BASE-BRANCH>

$ git pull upstream <SPYDER-BASE-BRANC>

$ git checkout -b NAME-NEW-BRANCH

```

### Changing the base branch

If you started your work in the wrong base branch, or want to backport it, you can change the base branch using `git rebase --onto`, like this:

```bash

$ git rebase --onto <NEW-BASE-BRANCH> <OLD-BASE-BRANCH> <YOUR-BRANCH>

```

For example, backporting `my\_branch` from `master` to `4.x`:

```bash

$ git rebase --onto 4.x master my\_branch

```

## Making contributions that depend on pull requests in spyder-kernels

Spyder and spyder-kernels are developed jointly because a lot of communication happens between them in order to run code written in the editor in the IPython console. The way the branches on their respective repos are linked appears in the table below:

| Spyder branch | Associated spyder-kernels branch |

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

| 4.x | 1.x |

| master (future 5.x) | master (future 2.x) |

For this reason, a clone of spyder-kernels is placed in the `external-deps` subfolder of the Spyder repository. The instructions on this section will help you in case you need to make changes that touch both repositories at the same time.

The first thing you need to do is cloning the [git-subrepo](https://github.com/ingydotnet/git-subrepo) project and follow these instructions to install it (on Windows you need to use Git Bash in order to run them):

```

git clone https://github.com/ingydotnet/git-subrepo /path/to/git-subrepo

echo 'source /path/to/git-subrepo/.rc' >> ~/.bashrc

source ~/.bashrc

```

As an example, let's assume that (i) your Github user name is `myuser`; (ii) you have two git repositories placed at `~/spyder` and `~/spyder-kernels` that link to `https://github.com/myuser/spyder` and `https://github.com/myuser/spyder-kernels` respectively; and (iii) you have two branches named `fix\_in\_spyder` and `fix\_in\_kernel` in each of these git repos respectively. If you want to open a joint PR in `spyder` and `spyder-kernels` that link these branches, here is how to do it:

\* Go to the `~/spyder` folder, checkout your `fix\_in\_spyder` branch and replace the spyder-kernels clone in the `external-deps` subfolder by a clone of your `fix\_in\_kernel` branch:

```

$ cd ~/spyder

$ git checkout fix\_in\_spyder

$ git subrepo clone https://github.com/myuser/spyder-kernels.git external-deps/spyder-kernels -b fix\_in\_kernel -f

```

\* You can now open a PR on `https://github.com/spyder-ide/spyder` and on `https://github.com/spyder-ide/spyder-kernels` for each of your branches.

\* If you make additional changes to the `fix\_in\_kernel` branch in `spyder-kernels` (e.g. adding a new file, as in the example below), you need to sync them in your Spyder's `fix\_in\_spyder` branch like this:

```

$ cd ~/spyder-kernels

$ git checkout fix\_in\_kernel

$ touch foo.py

$ git add -A

$ git commit -m "Adding foo.py to the repo"

$ git push origin fix\_in\_kernel

$ cd ~/spyder

$ git checkout fix\_in\_spyder

$ git subrepo pull external-deps/spyder-kernels

$ git push origin fix\_in\_spyder

```

\* When your `fix\_in\_kernel` PR is merged, you need to update Spyder's `fix\_in\_spyder` branch because the clone in Spyder's repo must point out again to the spyder-kernel's repo and not to your own clone. For that, please run:

```

$ git subrepo clone https://github.com/spyder-ide/spyder-kernels.git external-deps/spyder-kernels -b <branch> -f

```

where `<branch>` needs to be `1.x` if your `fix\_in\_spyder` branch was done against Spyder's `4.x` branch; and `master`, if you did it against our `master` branch here.

## Making contributions that depend on pull requests in python-language-server

As with spyder-kernels, Spyder is tightly integrated with the [python-language-server](https://github.com/palantir/python-language-server) to provide code completion, linting and folding on its editor.

Due to that, a clone of that project is placed in the `external-deps` directory, which is managed with the `git subrepo` project. If you want to make a pull request in python-language-server that affects functionality in Spyder, please read carefully the instructions in the previous section because they are very similar for this case. A summary of those instructions applied to this project is the following:

\* First you need to create a pull request in python-language-server with the changes you want to make there. Let's assume the branch from which that pull request is created is called `fix\_in\_pyls`.

\* Then you need to create a branch in Spyder (let's call it `fix\_in\_spyder`) with the fixes that require that pull request and update the python-language-server subrepo. For that you need to execute the following commands:

```

$ git checkout -b fix\_in\_spyder

$ git subrepo clone https://github.com/myuser/python-language-server.git external-deps/python-language-server -b fix\_in\_pyls -f

```

and then commit the changes you need to make in Spyder.

\* If you need to add more commits to `fix\_in\_pyls`, you need to update `fix\_in\_spyder` with these commands:

```

$ git checkout fix\_in\_spyder

$ git subrepo pull external-deps/python-language-server

$ git push origin fix\_in\_spyder

```

\* After `fix\_in\_pyls` is merged, you need to update the python-language-server subrepo in your `fix\_in\_spyder` branch with

```

$ git checkout fix\_in\_spyder

$ git subrepo clone https://github.com/palantir/python-language-server.git external-deps/python-language-server -b develop -f

```

## Adding Third-Party Content

All files or groups of files, including source code, images, icons, and other assets, that originate from projects outside of the Spyder organization (regardless of the license), must be first approved by the Spyder team. Always check with us (on Github, Gitter, Google Group, etc) before attempting to add content from an external project, and only do so when necessary.

### Licenses

Code considered for inclusion must be under a permissive (i.e. non-copyleft) license, particularly as the following (in order of preference):

\* MIT (Expat)

\* Public domain (preferably, CC0)

\* ISC license

\* BSD 2-clause ("Simplified BSD")

\* BSD 3-clause ("New" or "Modified BSD")

\* Apache License 2.0

Additionally, external assets (fonts, icons, images, sounds, animations) can generally be under one of the following weak-copyleft and content licenses:

\* Creative Commons Attribution 3.0 or 4.0

\* SIL Open Font License 1.1

\* GNU LGPL 2.1 or 3.0

Additional licenses \*may\* qualify for these lists from time to time, but every effort should be made to avoid it. Regardless, all such licenses must be OSI, FSF, and DSFG approved as well as GPLv3-compatible to ensure maximum free distribution and use of Spyder with minimum ambiguity or fragmentation.

### Steps to take

1. Contact the Spyder team to ensure the usage is justified and compatible.

2. Add the files, preserving any original copyright/legal/attribution header

3. If making non-trivial modifications, copy the standard Spyder copyright header from ``.ciocopyright`` to just below the original headers; if the original headers are unformatted and just consist of a copyright statement and perhaps mention of the license, incorporate them verbatim within the Spyder header where appropriate. Always ensure copyright statements are in ascending chronological order, and replace the year in the Spyder copyright statement with the current one. Modify the license location to be the current directory, or NOTICE.txt.

4. Include the following line at the end of each module's docstring, separated by blank lines:

```rst

Adapted from path/to/file/in/original/repo.py of the

`Project Name <url-to-original-github-repo>`\_.

```

For example,

```rst

Adapted from qcrash/\_dialogs/gh\_login.py of the

`QCrash Project <https://github.com/ColinDuquesnoy/QCrash>`\_.

```

5. Convert the files to project standards where needed.

6. If the copied file(s) reside in a directory dedicated to them, place the source project's LICENSE.txt file there, and any other legal files. Also, mention the same in the \_\_init\_\_.py file in that directory.

7. Add an entry in NOTICE.txt with the instructions and template there.

8. If a non-code visible asset (icons, fonts, animations, etc) or otherwise under a Creative Commons license, include a mention in the appropriate section of the README, as well as Spyder's About dialog, in the same form as the others present there.

## More information

[Main Website](https://www.spyder-ide.org/)

[Download Spyder (with Anaconda)](https://www.anaconda.com/download/)

[Online Documentation](https://docs.spyder-ide.org/)

[Spyder Github](https://github.com/spyder-ide/spyder)

[Troubleshooting Guide and FAQ](

https://github.com/spyder-ide/spyder/wiki/Troubleshooting-Guide-and-FAQ)

[Development Wiki](https://github.com/spyder-ide/spyder/wiki/Dev:-Index)

[Gitter Chatroom](https://gitter.im/spyder-ide/public)

[Google Group](https://groups.google.com/group/spyderlib)

[@Spyder\_IDE on Twitter](https://twitter.com/spyder\_ide)

[@SpyderIDE on Facebook](https://www.facebook.com/SpyderIDE/)

[Support Spyder on OpenCollective](https://opencollective.com/spyder/)