## Triaging Issues

On the IPython repository, we strive to trust users and give them responsibility.

By using one of our bots, any user can close issues or add/remove

labels by mentioning the bot and asking it to do things on your behalf.

To close an issue (or PR), even if you did not create it, use the following:

> @meeseeksdev close

This command can be in the middle of another comment, but must start on its

own line.

To add labels to an issue, ask the bot to `tag` with a comma-separated list of

tags to add:

> @meeseeksdev tag windows, documentation

Only already pre-created tags can be added. So far, the list is limited to:

`async/await`, `backported`, `help wanted`, `documentation`, `notebook`,

`tab-completion`, `windows`

To remove a label, use the `untag` command:

> @meeseeksdev untag windows, documentation

We'll be adding additional capabilities for the bot and will share them here

when they are ready to be used.

## Opening an Issue

When opening a new Issue, please take the following steps:

1. Search GitHub and/or Google for your issue to avoid duplicate reports.

Keyword searches for your error messages are most helpful.

2. If possible, try updating to master and reproducing your issue,

because we may have already fixed it.

3. Try to include a minimal reproducible test case.

4. Include relevant system information. Start with the output of:

python -c "import IPython; print(IPython.sys\_info())"

And include any relevant package versions, depending on the issue, such as

matplotlib, numpy, Qt, Qt bindings (PyQt/PySide), tornado, web browser, etc.

## Pull Requests

Some guidelines on contributing to IPython:

\* All work is submitted via Pull Requests.

\* Pull Requests can be submitted as soon as there is code worth discussing.

Pull Requests track the branch, so you can continue to work after the PR is submitted.

Review and discussion can begin well before the work is complete,

and the more discussion the better.

The worst case is that the PR is closed.

\* Pull Requests should generally be made against master

\* Pull Requests should be tested, if feasible:

- bugfixes should include regression tests.

- new behavior should at least get minimal exercise.

\* New features and backwards-incompatible changes should be documented by adding

a new file to the [pr](docs/source/whatsnew/pr) directory, see [the README.md

there](docs/source/whatsnew/pr/README.md) for details.

\* Don't make 'cleanup' pull requests just to change code style.

We don't follow any style guide strictly, and we consider formatting changes

unnecessary noise.

If you're making functional changes, you can clean up the specific pieces of

code you're working on.

[Travis](http://travis-ci.org/#!/ipython/ipython) does a pretty good job testing

IPython and Pull Requests, but it may make sense to manually perform tests,

particularly for PRs that affect `IPython.parallel` or Windows.

For more detailed information, see our [GitHub Workflow](https://github.com/ipython/ipython/wiki/Dev:-GitHub-workflow).

## Running Tests

All the tests can by running

```shell

iptest

```

All the tests for a single module (for example \*\*test\_alias\*\*) can be run by using the fully qualified path to the module.

```shell

iptest IPython.core.tests.test\_alias

```

Only a single test (for example \*\*test\_alias\_lifecycle\*\*) within a single file can be run by adding the specific test after a `:` at the end:

```shell

iptest IPython.core.tests.test\_alias:test\_alias\_lifecycle

```

For convenience, the full path to a file can often be used instead of the module path on unix systems. For example we can run all the tests by using

```shell

iptest IPython/core/tests/test\_alias.py

```