# On Github Issues and Pull Requests

Found a bug? Have a new feature to suggest? Want to contribute changes to the codebase? Make sure to read this first.

## Bug reporting

Your code doesn't work, and you have determined that the issue lies with Keras? Follow these steps to report a bug.

1. Your bug may already be fixed. Make sure to update to the current Keras master branch, as well as the latest Theano/TensorFlow/CNTK master branch.

To easily update Theano: `pip install git+git://github.com/Theano/Theano.git --upgrade`

2. Search for similar issues. Make sure to delete `is:open` on the issue search to find solved tickets as well. It's possible somebody has encountered this bug already. Also remember to check out Keras' [FAQ](http://keras.io/faq/). Still having a problem? Open an issue on Github to let us know.

3. Make sure you provide us with useful information about your configuration: what OS are you using? What Keras backend are you using? Are you running on GPU? If so, what is your version of Cuda, of cuDNN? What is your GPU?

4. Provide us with a script to reproduce the issue. This script should be runnable as-is and should not require external data download (use randomly generated data if you need to run a model on some test data). We recommend that you use Github Gists to post your code. Any issue that cannot be reproduced is likely to be closed.

5. If possible, take a stab at fixing the bug yourself --if you can!

The more information you provide, the easier it is for us to validate that there is a bug and the faster we'll be able to take action. If you want your issue to be resolved quickly, following the steps above is crucial.

---

## Requesting a Feature

You can also use [Tensorflow Github issues](https://github.com/tensorflow/tensorflow/issues) to request features you would like to see in Keras, or changes in the Keras API.

1. Provide a clear and detailed explanation of the feature you want and why it's important to add. Keep in mind that we want features that will be useful to the majority of our users and not just a small subset. If you're just targeting a minority of users, consider writing an add-on library for Keras. It is crucial for Keras to avoid bloating the API and codebase.

2. Provide code snippets demonstrating the API you have in mind and illustrating the use cases of your feature. Of course, you don't need to write any real code at this point!

3. After discussing the feature you may choose to attempt a Pull Request on tf.keras. If you're at all able, start writing some code. We always have more work to do than time to do it. If you can write some code then that will speed the process along.

---

## Requests for Contributions

[This is the board](https://github.com/keras-team/keras/projects/1) where we list current outstanding issues and features to be added. If you want to start contributing to Keras, this is the place to start.

---

## Pull Requests

\*\*Where should I submit my pull request?\*\*

#### Note:

We are no longer adding new features to multi-backend Keras (we only fix bugs), as we are refocusing development efforts on tf.keras. If you are still interested in submitting a feature pull request, please direct it to tf.keras in the TensorFlow repository instead.

1. \*\*Keras improvements and bugfixes\*\* go to the [Keras `master` branch](https://github.com/keras-team/keras/tree/master).

2. \*\*Experimental new features\*\* such as layers and datasets go to [keras-contrib](https://github.com/farizrahman4u/keras-contrib). Unless it is a new feature listed in [Requests for Contributions](https://github.com/keras-team/keras/projects/1), in which case it belongs in core Keras. If you think your feature belongs in core Keras, you can submit a design doc to explain your feature and argue for it (see explanations below).

Please note that PRs that are primarily about \*\*code style\*\* (as opposed to fixing bugs, improving docs, or adding new functionality) will likely be rejected.

Here's a quick guide to submitting your improvements:

1. If your PR introduces a change in functionality, make sure you start by writing a design doc and sending it to the Keras mailing list to discuss whether the change should be made, and how to handle it. This will save you from having your PR closed down the road! Of course, if your PR is a simple bug fix, you don't need to do that. The process for writing and submitting design docs is as follow:

- Start from [this Google Doc template](https://docs.google.com/document/d/1ZXNfce77LDW9tFAj6U5ctaJmI5mT7CQXOFMEAZo-mAA/edit#), and copy it to new Google doc.

- Fill in the content. Note that you will need to insert code examples. To insert code, use a Google Doc extension such as [CodePretty](https://chrome.google.com/webstore/detail/code-pretty/igjbncgfgnfpbnifnnlcmjfbnidkndnh?hl=en) (there are several such extensions available).

- Set sharing settings to "everyone with the link is allowed to comment"

- Send the document to `keras-users@googlegroups.com` with a subject that starts with `[API DESIGN REVIEW]` (all caps) so that we notice it.

- Wait for comments, and answer them as they come. Edit the proposal as necessary.

- The proposal will finally be approved or rejected. Once approved, you can send out Pull Requests or ask others to write Pull Requests.

2. Write the code (or get others to write it). This is the hard part!

3. Make sure any new function or class you introduce has proper docstrings. Make sure any code you touch still has up-to-date docstrings and documentation. \*\*Docstring style should be respected.\*\* In particular, they should be formatted in MarkDown, and there should be sections for `Arguments`, `Returns`, `Raises` (if applicable). Look at other docstrings in the codebase for examples.

4. Write tests. Your code should have full unit test coverage. If you want to see your PR merged promptly, this is crucial.

5. Run our test suite locally. It's easy: from the Keras folder, simply run: `py.test tests/`.

- You will need to install the test requirements as well: `pip install -e .[tests]`.

6. Make sure all tests are passing:

- with the Theano backend, on Python 2.7 and Python 3.6. Make sure you have the development version of Theano.

- with the TensorFlow backend, on Python 2.7 and Python 3.6. Make sure you have the development version of TensorFlow.

- with the CNTK backend, on Python 2.7 and Python 3.6. Make sure you have the development version of CNTK.

7. We use PEP8 syntax conventions, but we aren't dogmatic when it comes to line length. Make sure your lines stay reasonably sized, though. To make your life easier, we recommend running a PEP8 linter:

- Install PEP8 packages: `pip install pep8 pytest-pep8 autopep8`

- Run a standalone PEP8 check: `py.test --pep8 -m pep8`

- You can automatically fix some PEP8 error by running: `autopep8 -i --select <errors> <FILENAME>` for example: `autopep8 -i --select E128 tests/keras/backend/test\_backends.py`

8. When committing, use appropriate, descriptive commit messages.

9. Update the documentation. If introducing new functionality, make sure you include code snippets demonstrating the usage of your new feature.

10. Submit your PR. If your changes have been approved in a previous discussion, and if you have complete (and passing) unit tests as well as proper docstrings/documentation, your PR is likely to be merged promptly.

---

## Adding new examples

Even if you don't contribute to the Keras source code, if you have an application of Keras that is concise and powerful, please consider adding it to our collection of examples. [Existing examples](https://github.com/keras-team/keras/tree/master/examples) show idiomatic Keras code: make sure to keep your own script in the same spirit.