How to contribute to Flask

Thank you for considering contributing to Flask!

Support questions

Please, don't use the issue tracker for this. Use one of the following resources for questions about your own code:

- The `#get-help` channel on our Discord chat: https://discordapp.com/invite/t6rrQZH
    - The IRC channel `#pocoo` on FreeNode is linked to Discord, but Discord is preferred.
- The mailing list flask@python.org for long term discussion or larger issues.
- Ask on Stack Overflow. Search with Google first using: `site:stackoverflow.com flask {search term, exception message, etc.}`

Reporting issues

- Describe what you expected to happen.
- If possible, include a minimal reproducible example to help us identify the issue. This also helps check that the issue is not with your own code.
- Describe what actually happened. Include the full traceback if there was an exception.
- List your Python, Flask, and Werkzeug versions. If possible, check if this issue is already fixed in the repository.

Submitting patches

- Use Black to autoformat your code. This should be done for you as a git pre-commit hook, which gets installed when you run `pip install -e .[dev]`. You may also wish to use Black's Editor integration.
- Include tests if your patch is supposed to solve a bug, and explain clearly under which circumstances the bug happens. Make sure the test fails without your patch.
- Include a string like "Fixes #123" in your commit message (where 123 is the issue you fixed). See Closing issues using keywords.

First time setup

- Download and install the latest version of git.
- Configure git with your username and email:

```
git config --global user.name 'your name'
git config --global user.email 'your email'
```

- Make sure you have a GitHub account.
- Fork Flask to your GitHub account by clicking the Fork button.
- Clone your GitHub fork locally:

```
git clone https://github.com/{username}/flask
cd flask
```

- Add the main repository as a remote to update later:

```
git remote add pallets https://github.com/pallets/flask
git fetch pallets
```

- Create a virtualenv:

```
python3 -m venv env
. env/bin/activate
# or "env\Scripts\activate" on Windows
```

- Install Flask in editable mode with development dependencies:

```
pip install -e ".[dev]"
```

- Install the pre-commit framework.
- Install the pre-commit hooks:

```
pre-commit install --install-hooks
```

Start coding

- Create a branch to identify the issue you would like to work on. If you're submitting a bug or documentation fix, branch off of the latest ".x" branch:

```
git checkout -b your-branch-name origin/1.0.x
```

  If you're submitting a feature addition or change, branch off of the "master" branch:

```
git checkout -b your-branch-name origin/master
```


- Using your favorite editor, make your changes, committing as you go.

- Include tests that cover any code changes you make. Make sure the test fails without your patch. Run the tests..
- Push your commits to GitHub and create a pull request by using:

```
git push --set-upstream origin your-branch-name
```

- Celebrate 🎉

Running the tests

Run the basic test suite with:

```
pytest
```

This only runs the tests for the current environment. Whether this is relevant depends on which part of Flask you're working on. Travis-CI will run the full suite when you submit your pull request.

The full test suite takes a long time to run because it tests multiple combinations of Python and dependencies. You need to have Python 2.7, 3.4, 3.5 3.6, and PyPy 2.7 installed to run all of the environments. Then run:

```
tox
```

Running test coverage

Generating a report of lines that do not have test coverage can indicate where to start contributing. Run `pytest` using `coverage` and generate a report on the terminal and as an interactive HTML document:

```
coverage run -m pytest
coverage report
coverage html
# then open htmlcov/index.html
```

Read more about coverage.

Running the full test suite with `tox` will combine the coverage reports from all runs.

Building the docs

Build the docs in the `docs` directory using Sphinx:

```
cd docs
pip install -r requirements.txt
make html
```

Open `_build/html/index.html` in your browser to view the docs.

Read more about Sphinx.

Caution: zero-padded file modes

This repository contains several zero-padded file modes that may cause issues when pushing this repository to git hosts other than GitHub. Fixing this is destructive to the commit history, so we suggest ignoring these warnings. If it fails to push and you're using a self-hosted git service like GitLab, you can turn off repository checks in the admin panel.

These files can also cause issues while cloning. If you have

```
[fetch]
fsckobjects = true
```

or

```
[receive]
fsckObjects = true
```

set in your git configuration file, cloning this repository will fail. The only solution is to set both of the above settings to false while cloning, and then setting them back to true after the cloning is finished.