## Contributing to CVXPY

This document is a guide to contributing to CVXPY.

We welcome all contributions. You don't need to be an expert in optimization

to help out.

## Checklist

Contributions are made through

[pull requests](https://help.github.com/articles/using-pull-requests/).

Before sending a pull request, make sure you do the following:

- Check that your code adheres to our [coding style](#code-style)

- Add our [license](#license) to new files

- [Write unit tests](#writing-unit-tests)

- Run the [unit tests](#running-unit-tests) and check that they're passing

- Run the [benchmarks](#benchmarks) to make sure your change does not introduce a regression

## Building CVXPY from source

You'll need to build CVXPY locally in order to start editing code. We recommend

that you do this in a fresh [virtual

environment](https://virtualenv.pypa.io/en/latest/).

To install CVXPY from source, clone the Github repository, navigate to the

repository root, and run the following command:

```

python setup.py develop

```

## Contributing code

To contribute to CVXPY, send us pull requests. For those new to contributing,

check out Github's

[guide](https://help.github.com/articles/using-pull-requests/).

Once you've made your pull request, a member of the CVXPY development team

will assign themselves to review it. You might have a few back-and-forths

with your reviewer before it is accepted, which is completely normal. Your

pull request will trigger continuous integration tests for many different

Python versions and different platforms. If these tests start failing, please

fix your code and send another commit, which will re-trigger the tests.

If you'd like to add a new feature to CVXPY, or a new example to our

[library](https://www.cvxpy.org/examples/index.html), please do propose your

change on a Github issue, to make sure that your priorities align with ours.

If you'd like to contribute code but don't know where to start, try one of the

following:

\* Read the CVXPY source and enhance the documentation, or address TODOs

\* Browse the [issue tracker](https://github.com/cvxgrp/cvxpy/issues), and

look for the issues tagged "help wanted".

\* Polish the [example library](https://www.cvxpy.org/examples/index.html) or add new examples

\* Add a [benchmark](https://github.com/cvxgrp/cvxpy/tree/master/cvxpy/tests/test\_benchmarks.py)

## License

Please add the following license to new files:

```

"""

Copyright, the CVXPY authors

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

"""

```

## Code style

We use [flake8](http://flake8.pycqa.org/en/latest/) to enforce our Python coding

style. Before sending us a pull request, navigate to the project root

and run

```

flake8 cvxpy/

```

to make sure that your changes abide by our style conventions. Please fix any

errors that flake8 reports before sending the pull request.

## Writing unit tests

Most code changes will require new unit tests. (Even bug fixes require unit tests,

since the presence of bugs usually indicates insufficient tests.) CVXPY tests

live in the directory `cvxpy/tests`, which contains many files, each of which

contains many unit tests. When adding tests, try to find a file in which your

tests should belong; if you're testing a new feature, you might want to create

a new test file.

We use the standard Python [`unittest`](https://docs.python.org/3/library/unittest.html)

framework for our tests. Tests are organized into classes, which inherit from

`BaseTest` (see `cvxpy/tests/base\_test.py`). Every method beginning with `test\_` is a unit

test.

## Running unit tests

We use `nose` to run our unit tests, which you can install with `pip install nose`.

To run all unit tests, `cd` into `cvxpy/tests` and run the following command:

```

nosetests

````

To run tests in a specific file (e.g., `test\_dgp.py`), use

```

nosetests test\_dgp.py

```

To run a specific test method (e.g., `TestDgp.test\_product`), use

```

nosetests test\_dgp.py:TestDgp.test\_product

```

Please make sure that your change doesn't cause any of the unit tests to fail.

`nosetests` suppresses stdout by default. To see stdout, pass the `-s` flag

to `nosetests`.

## Benchmarks

CVXPY has a few benchmarks in `cvxpy/tests/test\_benchmarks.py`, which test

the time to canonicalize problems. Please run

```

nosetests -s test\_benchmarks.py

```

with and without your change, to make sure no performance regressions are

introduced. If you are making a code contribution, please include the output of

the above command (with and without your change) in your pull request.