## Contributing to Pyston

### Pull Requests

Before a pull request can be merged, you need to to sign the [Dropbox Contributor License Agreement](https://opensource.dropbox.com/cla/).

Please make sure to run at least the basic tests (`make quick\_check`) on your changes. Travis CI will run the full test suite (`make check`) on each pull request.

##### Formatting

Please make sure `make check\_format` passes on your commits. If it reports any issues, you can run `make format` to auto-apply the project formatting rules. Note that this will format your files in-place with no built-in undo, so you may want to create a temporary commit if you have any unstaged changes.

Adding a pre-commit hook can be useful to catch formatting errors earlier. i.e. have in `~/pyston/.git/hooks/pre-commit`:

```

#!/bin/sh

exec make -C ~/pyston check\_format

```

### Getting started with the codebase

The easiest way to contribute to Pyston is to help us improve our compatibility with CPython. There are many small tasks to do such as built-in functions that are not yet implemented, edge cases that are not being handled by Pyston or where our output is slightly different than CPython, etc. The fix will often involve a local change, giving a smooth start to learning the codebase. One of Python's greatest strengths is that it comes ["batteries included"](https://xkcd.com/353/), but this means that there is a long long tail of these little tasks that needs to be driven through - your help is immensely valuable there!

The command `make quick\_check` will first run our Pyston tests (great way to make sure everything is in order) and then the default CPython tests. You will get an output that looks like this:

```

...

test\_bastion.py Correct output (125.7ms)

test\_unittest.py (skipped: expected to fail)

test\_json.py (skipped: expected to fail)

test\_future3.py Correct output (952.8ms)

...

```

Notice that a large number tests are currently marked as failing (marked with an `# expected: fail` comment at the top of the file). Just pick any that you think is interesting and get it to pass! Remove the `#expected: fail` flag and run `make check\_TESTNAME` (without `.py`) to compare the result to CPython's (the command will search for TESTNAME in the `test/` directory). If the test is crashing, the easiest way to start debugging is to use `make dbg\_TESTNAME` which is essentially `make check\_TESTNAME` inside GDB.

This kind of work will often happen where native libraries are defined (e.g. `src/runtime/builtin\_modules/builtins.cpp`), implementation of types (e.g. `src/runtime/str.cpp`) and debugging may involve tracing through the interpreter (`src/codegen/ast\_interpreter.cpp`). The code in those files should be relatively straightforward. Code that involve the JIT (function rewriting, assembly generation, etc) is more intricate and confusing in the beginning (e.g. `src/asm\_writing/rewriter.cpp`). Keep in mind, it's perfectly fine to ask for help!

To save you some time, the cause of failures for some of the tests [may have already been identified](test/CPYTHON\_TEST\_NOTES.md). Do note, however, that not all of CPython's behavior can be matched exactly. For example, by nature of having a garbage collector over reference counting, the freeing of objects is non-deterministic and we can't necessarily call object finalizers in the same order as CPython does.

[Some tips on challenges you might run into and debugging tips](docs/TIPS.md).

You can also check out our [Github issues list](https://github.com/dropbox/pyston/issues), especially those marked ["probably easy"](https://github.com/dropbox/pyston/labels/probably%20easy).

### Communicating

- We use a [gitter chat room](https://gitter.im/dropbox/pyston) for most of our discussions. If you need help with figuring out where to start or with the codebase, you should get a response there fairly quickly. If you found a small project to work on already and are eager to start, by all means get started! It is still a good idea to drop us a note - we might some suggestions or we can think of an edge case or two.

- Email the [pyston-dev mailing list](http://lists.pyston.org/cgi-bin/mailman/listinfo/pyston-dev), or [browse the archives](http://lists.pyston.org/pipermail/pyston-dev/)

### Bigger projects

There are many big areas where Pyston can use improvements. This includes, for example, a better garbage collector, better performance profiling tools (including finding more benchmarks), finding new libraries to add to our test suite, etc. These can be very involved - if you are interested bigger projects (e.g. as part of graduate studies), please contact us directly.