Want to contribute? Great! First, read this page (including the small print at

the end).

### Before you contribute

Before we can use your code, you must sign the

[Google Individual Contributor License Agreement](https://developers.google.com/open-source/cla/individual?csw=1)

(CLA), which you can do online. The CLA is necessary mainly because you own the

copyright to your changes, even after your contribution becomes part of our

codebase, so we need your permission to use and distribute your code. We also

need to be sure of various other things -- for instance that you'll tell us if you

know that your code infringes on other people's patents. You don't have to sign

the CLA until after you've submitted your code for review and a member has

approved it, but you must do it before we can put your code into our codebase.

Before you start working on a larger contribution, you should get in touch with

us first through the issue tracker with your idea so that we can help out and

possibly guide you. Coordinating up front makes it much easier to avoid

frustration later on.

### Code reviews

All submissions, including submissions by project members, require review. We

use Github pull requests for this purpose.

### Pytype dependencies

Before you can build and test Pytype, you will have to install a few

dependencies.

1. \_\_A C++11 compiler for your platform\_\_: Pytype uses extension modules.

A C++11 compiler is required to build these extension modules.

2. \_\_[CMake](https://cmake.org) version 2.8 or higher\_\_: To build the extension

modules and to run tests in parallel, Pytype makes use of a CMake based

build system.

3. \_\_[Bison](https://www.gnu.org/software/bison/) version 3.0.2 or higher\_\_:

Pytype uses a custom parser to parse PYI files. This parser is implemented

using Flex and Bison.

4. \_\_[Flex](https://www.gnu.org/software/flex/) version 2.5.35 or higher\_\_

5. \_\_[ninja build](https://ninja-build.org/)\_\_: Pytype's test utility scripts

make use of ninja as the CMake generated build system.

6. \_\_Python2.7 and Python3.x Interpreters\_\_: A large subset of Pytype's

functional tests analyse the target (the Python source code that is being

analyzed by Pytype) twice: once as if it were Python2.7 code, and another

time as if it were in the Python 3.x version that pytype is running under.

Hence, to run these tests, you will need Python 2.7 and Python 3.x

(preferably 3.6+) interpreters installed on your system.

The Pytype Git repository also contains few Git submodules. Before building

the `pytype` executable or running tests, one has to ensure that the submodules

are up to date. This can be done with the following command:

```

$> git submodule update --init

```

### Building `pytype` and other executables

The executables like `pytype` etc. are built using a convenience script as

follows:

```

$> python build\_scripts/build.py

```

`build.py` will build the executables in the `out/bin` directory.

### Logging

One can pass the logging verbosity level option to `pytype-single` to see the

logs:

```

$> out/bin/pytype-single -v<N> <other command like arguments>

```

For information about the logging levels, run `pytype-single --help`.

#### Logging from extension modules

The `pytype-single` executable makes use of few a C extension modules. Logging

from these extension modules is enabled only in debug builds. One can build

`pytype-single` in debug mode by passing the `--debug` option to the build

script as follows:

```

$> python build\_scripts/build.py --debug

```

In a debug build of `pytype-single`, logging from extension modules follows the

same verbosity levels as the rest of the Python modules.

### Adding tests to your Changes

Ideally, every change should include a test. Depending on the type of your

change, you should either be adding a functional test or a unit test (some

changes might warrant both). Functional tests should be added in the

`pytype/tests` directory. Unit tests should be added in a test module next to

the module that is being tested.

Since Pytype already has exhaustive tests, a change will most likely need to

add a test method to an existing test module. In such a case, there is

nothing special required, other than just adding a new test method. If adding a

new test module is more meaningful, apart from adding the new test module, your

change should also add a `py\_test` target to the `CMakeLists.txt` file in the

directory in which the test module lives. See existing `py\_test` targets (in

`CMakeLists.txt` files) for examples on how to do this.

NOTE: Please see `pytype/tests/README.md` for more rules pertaining to adding

new functional tests.

### Running tests

There exists a convenience script to run Pytype tests. A typical usage of this

script is as follows:

```

$> python build\_scripts/run\_tests.py <TARGET>

```

`TARGET` is the fully qualified name of the test target within the root Pytype

source tree. If a target name is not specified, the script runs all `py\_test`

and `cc\_test` targets in the Pytype source tree.

For more information about `run\_tests.py` options, run `run\_test.py --help`.

### The small print

Contributions made by corporations are covered by a different agreement than

the one mentioned above; they're covered by the Software Grant and

Corporate Contributor License Agreement.