python setup.py install is used to install (typically third party) packages that you're not going to develop/modify/debug yourself.

For your own package, you want to first install your package and then be able to frequently edit the code without having to re-install the package every time — and that is exactly what python setup.py develop does: it installs the package (typically just a source folder) in a way that allows you to conveniently edit your code after it’s installed to the (virtual) environment, and have the changes take effect immediately.

Note that it is highly recommended to use pip install . (install) and pip install -e . (developer install) to install packages, as invoking setup.py directly will do the wrong things for many dependencies, such as pull prereleases and incompatible package versions, or make the package hard to uninstall with pip.

$ pip install -e git+https://github.com/lakshmivyas/hyde.git#egg=hyde

You can’s uninstall package installed using “pip install –e .”. to uninstall, you need to delete the .egg-link file

1. Run “python –m site” to list the site-packages directory
2. In the site-packages directory, find and delete the *yourproject.egg-link* file
3. You might also need to find a file named easy-install.pth in a similar site-packages directory (most likely the same). In this file, delete the line mentioning *yourproject*

**HOW TO USE A GIT REPOSITORY AS A PIP DEPENDENCY**

Did you know you can install a python library using a link to a git repository instead of package index?

It can be handy to use pip to install a project dependency directly from a git repository instead of from a Python package index. I'll show you why you might want to do that and how to do it.

**Why would you want to pip install from a git repository?**

1. You may want to quickly test out some code from a colleague.
2. You may have a library that you want to use in multiple projects, but it doesn't warrant creating a python package.
3. You made a fix to a third party project and can't wait for the pull request to be merged and deployed. You can use your fork and then switch to an official release later.

**How to pip install from a git repository**

If this is a new repo, you'll minimally need a [setup.py](https://github.com/matiascodesal/git-for-pip-example/blob/master/setup.py) so that pip can carry out the install. Other than that, it really just boils down to giving pip a git repository URL.

*# You'll want to do something in the form of*

pip **install** git+<repository\_url>

*# For example*

pip **install** git+https://github.com/matiascodesal/git-for-pip-example.git

*# Check that it worked*

python -c "import greetings;greetings.hello()"

Copy

You can explicitly call out the package name that you're installing with *#egg=*

*# The syntax*

pip **install** git+<repository\_url>*#egg=<package\_name>*

*# Explicit package name example*

pip **install** git+https://github.com/matiascodesal/git-for-pip-example.git*#egg=git-for-pip-example*

Copy

There are also a number of different ways to specify a version of the repository that you want to fetch. It's wise to use one of these methods to lock your dependency so that you get consistent results.

*# Use a commit SHA*

pip **install** git+https://github.com/matiascodesal/git-for-pip-example.git@4045597*#egg=git-for-pip-example*

*# Use a tag*

pip **install** git+https://github.com/matiascodesal/git-for-pip-example.git@v1.0.0*#egg=git-for-pip-example*

*# Use a branch called "GreetingArg"*

pip **install** git+https://github.com/matiascodesal/git-for-pip-example.git@GreetingArg*#egg=git-for-pip-example*

Copy

**How to include the dependency in a requirements.txt**

If a git repository dependency is going to live in a project for more than just testing, you'll likely want to add it to your requirements.txt. There's a bug with "pip freeze" where the git repository dependency wasn't being output by "pip freeze". That was only recently fixed in pip 20.1. I'll show you the workaround for older versions of pip and the new way to list the dependency. The workaround for older versions of pip is to use the *-e* or *--editable* flag:

*# The pip command with "-e"*

pip **install** -e git+https://github.com/matiascodesal/git-for-pip-example.git@v1.0.0*#egg=my-git-package*

*# What that should look like in your requirements.txt*

packageA==1.2.3

-e https://github.com/matiascodesal/git-for-pip-example.git@v1.0.0*#egg=my-git-package*

packageB==4.5.6

Copy

For pip 20.1 or newer, you no longer need the *-e* flag:

*# Just put the pip install argument straight into your requirements.txt*

packageA==1.2.3

git+https://github.com/matiascodesal/git-for-pip-example.git@v1.0.0*#egg=my-git-package*

packageB==4.5.6

*# Or you can use the preferred PEP 440 direct URL syntax*

packageA==1.2.3

git-for-pip-example @ git+https://github.com/matiascodesal/git-for-pip-example.git@v1.0.0

packageB==4.5.6

Copy

I hope this gives you some flexibility in managing your projects while still being mindful of tracking your dependencies and deploying code in a consistent and reliable manner.