Pyg Documentation

Release 1.0

Michele Lacchia

December 01, 2013

Contents

Thanks to: Free Domain Name for the pyg-installer.co.nr hook.

Authors Michele Lacchia <michelelacchia@gmail.com>, Fabien Devaux

Version 1.0

Source code http://github.com/rubik/pyg

Download http://pypi.python.org/pypi/pyg

Homepage http://pyg-installer.co.nr

Keywords python, easy_install, packages, pypi, command line, cli

Date December 01, 2013

Note: These are development docs. Documentation for Pyg last stable release is here: http://pyg.readthedocs.org/en/latest

---

CHAPTER 1

Quickstart

1.1 Installing Pyg

1.1.1 The `get-pyg.py` file

The preferred way to install Pyg (and the easiest one) is to download the `get-pyg.py` file from Github and to launch it:

```
$ curl -O https://raw.github.com/rubik/pyg/master/get-pyg.py
$ python get-pyg.py
```

This will install the latest stable release, but if you want to install the development version, just do:

```
$ python get-pyg.py --dev
```

1.1.2 Using `Pip` or `easy_install`

If you have `easy_install` or `Pip` installed, to install Pyg it only takes one simple command:

```
$ pip install pyg
```

or if you must:

```
$ easy_install pyg
```

And then you should no longer need them!

1.2 Getting the source

You can also install Pyg from source. The lastest release is avaiable on GitHub:

- tarball
- zipball

Once you have unpacked the archive you can install it easily:

```
$ python setup.py install
```

1.3 Building the documentation

In order to build Pyg's documentation locally you have to install Sphinx:

```
$ pyg install sphinx
```

Download the source (see *Getting the source*), go to `docs/` and run **make**:

```
$ cd docs/
$ make html
```

Now Pyg's documentation is in `_build/html`.

CHAPTER 2

Overview

2.1 Pyg's features

Pyg can:

- install packages from `.tar.gz`, `.tar.bz2`, `.zip` archives, as well as from `.egg` files and `.pybundle` files.
- Uninstall packages.
- Define fixed sets of requirement.
- Perform searches on PyPI.
- Install from binaries (e.g. from `.exe` or `.msi`) on Windows.
- Install packages in editable mode from VCS (Git, Mercurial, Bazaar, Svn).

Currently Pyg don't:

- understand Setuptools extras (like `package[extra]`). This is planned for Pyg 1.0.

2.1.1 Pyg compared to easy_install

Pyg is meant to be a replacement for easy_install, and tries to improve it. In particular:

- It can install packages from a name, URL, local directory or file.
- It supports installation from requirements files.
- It can install packages from Pip's bundels.
- Easy yet very powerful uninstallation of packages.
- It can perform searches on PyPI.
- It offers the possibility to download a package without installing it.
- Source code concise and cohesive and easily extensible.
- Pyg can used either as a command-line tool or a Python library.
- The output on the console should be useful.

But at the moment Pyg does not do everything that easy_install does:

- It does not support Setuptools extras (like `package[extra]`). This is planned for Pyg 1.0.

2.1.2 Pyg compared to Pip

Pyg is very similar to Pip but tries to improve something. Specifically:

- Pyg uses the same installation method as Pip but a different package discovery system that should be faster.
- Pyg supports Python Eggs installation, while Pip doesn't.
- A better uninstallation of packages (Pip cannot install packages installed with `python setup.py install`).

In addition to that, Pyg is completely useable under vitualenvs.

2.1.3 Uninstall

Pyg can uninstall most installed packages with:

```
$ pyg uninstall packname
```

It tries to detect the directory where the packages have been installed and delete them. Pyg can uninstall all packages, except those that have been installed in editable mode.

See also: *Uninstalling*.

2.1.4 Package upgrading

This is a feature unique to Pyg: by running `pyg update` you can check all your installed packages and upgrade those for which there is a newer release. Pyg collects all packages that can upgrade and then check for updates.

See also: *Upgrading installed packages*.

2.1.5 Pyg and virtualenv

New in version 0.5. From Pyg 0.5 onwards, virtualenv is completely supported. You can easily manage packages from inside it. A little example:

```
$ virtualenv env -p /usr/bin/python2.6 --no-site-packages
Running virtualenv with interpreter /usr/bin/python2.6
New python executable in env/bin/python2.6
Also creating executable in env/bin/python
Installing setuptools.............................done.
$ cd env
$ source bin/activate
(env)$ curl -O https://github.com/rubik/pyg/raw/master/get_pyg.py
(env)$ python get_pyg.py
(env)$ pyg install sphinx
Looking for sphinx releases on PyPI
Best match: Sphinx==1.0.7
Downloading Sphinx
Checking md5 sum
Running setup.py egg_info for Sphinx
Running setup.py install for Sphinx
Installing dependencies...
Installing Jinja2>=2.2 (from Sphinx==1.0.7)
```

```
        Looking for Jinja2 releases on PyPI
        Best match: Jinja2==2.5.5
        Downloading Jinja2
        Checking md5 sum
        Running setup.py egg_info for Jinja2
        Running setup.py install for Jinja2
        Installing dependencies...
Installing Babel>=0.8 (from Jinja2==2.2)
        Looking for Babel releases on PyPI
        Best match: Babel==0.9.6
        Downloading Babel
        Checking md5 sum
        Running setup.py egg_info for Babel
        Running setup.py install for Babel
        Babel installed successfully
Finished installing dependencies
Jinja2 installed successfully
Installing docutils>=0.5 (from Sphinx==1.0.7)
        Looking for docutils releases on PyPI
        Best match: docutils==0.7
        Downloading docutils
        Checking md5 sum
        Running setup.py egg_info for docutils
        Running setup.py install for docutils
        docutils installed successfully
Installing Pygments>=0.8 (from Sphinx==1.0.7)
        Looking for Pygments releases on PyPI
        Best match: Pygments==1.4
        Downloading Pygments
        Checking md5 sum
        Running setup.py egg_info for Pygments
        Running setup.py install for Pygments
        Pygments installed successfully
Finished installing dependencies
Sphinx installed successfully
(env)$ python
Python 2.6.6 (r266:84292, Mar 25 2011, 19:24:58)
[GCC 4.5.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import sphinx
>>> sphinx.__version__
'1.0.7'
>>>
(env)$ pyg remove sphinx
Uninstalling sphinx
        env/lib/python2.6/site-packages/Sphinx-1.0.7-py2.6.egg-info
        env/bin/sphinx-quickstart
        env/lib/python2.6/site-packages/sphinx
        env/bin/sphinx-build
        env/bin/sphinx-autogen
Proceed? (y/[n]) y
Removing egg path from easy_install.pth...
sphinx uninstalled succesfully
```

2.1.6 Pyg Shell

You can launch Pyg Shell with:

```
$ pyg shell
```

and it will open a shell where you can use all Pyg's command. This is particularly useful on when you need root privileges to installs packages (e.g. Unix): if you need to execute many commands you can fire up the shell and then use Pyg without worrying about root privileges.

See also: *Pyg Shell*.

2.1.7 Bundles

Pyg supports Pip's bundles. The bundle format is specific to Pip (see Pip documentation). Once you have one you can install it like:

```
$ pyg install yourbundle.pyb
```

The internet access is not necessary. In addition to that, you can easily create bundles with Pyg. For example, if you want to create a bundle of Pyg, you would do:

```
$ pyg bundle pyg-bundle.pyb pyg
```

See also: *Bundles*.

2.1.8 Packs

New in version 0.7. Packs are very similar to bundles, except that they can contain Python executables too. Packs were invented by Fabien Devaux for the Zicbee project (check it at PyPI: Zicbee).

A pack contains a folder in which there is an egg (with all necessary packages) and some Python executable files (`run_name.py`). You can unpack the pack and then run the executables without touching the egg! Like a bundle, a pack does not require an internet connection to work: all required package are inside the zipped egg. The advantage of packs over bundles is that you can run included Python executables without installing the library, because everything necessary is included in the egg!

See also: *Packs*.

CHAPTER 3

Using Pyg

3.1 Using Pyg from the command line

Pyg should be used mostly from the command line, as it requires root's privileges to install and remove packages.

On some systems (e.g. on Unix systems), you may need root privileges to execute some commands such as **install**, **uninstall**, or **upgrade**. In this case you should use the **sudo** command.

Main commands:

3.1.1 Installing and removing packages

Installing

To install a package, simply run:

```
$ pyg install package
```

package can be a number of things:

- the name of the package you want to install (e.g. `pyg` or `sphinx`)
- a package URL (e.g. `http://www.example.org/path/to/mypackage-0.1.tar.gz`)
- a local file (like `path/to/mypackage-0.42-py2.7.egg`)
- a local directory containing a `setup.py` file
- a repository URL (like `git+git@github.com/rubik/pyg`)
- a gist URL (i.e. `gist+928471`)

Pyg supports these file-types:

- `.tar.gz`
- `.tgz`
- `.tar.bz2`
- `.zip`
- `.egg`

- `.exe`
- `.msi`
- `.pybundle`
- `.pyb` (an abbreviation of Pip's bundle files)

-e <URL>, **-editable** <URL>
New in version 0.3.Changed in version 0.7: Allow paths as arguments as well. Install a package in editable mode (`python setup.py develop`) from an online repository. Supported VCS are:

•Git (prefix `git+`)

•Mercurial (prefix `hg+`)

•Bazaar (prefix `bzr+`)

•Subversion (prefix `svn+`)

The URL syntax is as follows:

```
<prefix><repo_url>#egg=<package_name>
```

All fields are required. The last part (`#egg=<package_name>`) specifies the package name.

You can run it with a path as well. Actually, this:

```
$ pyg install -e path/to/a/directory
...
```

is equivalent to:

```
$ cd path/to/a/directory
$ python setup.py develop
```

-no-script
New in version 0.3. Do not install packages' scripts.

-no-data
New in version 0.3. Do not install packages' data files.

-r <path>, **-req-file** <path>
Install packages from the specified requirement file:

```
$ pyg install -r requirements.txt
```

See also: *Freezing requirements*

-U, -upgrade
New in version 0.2. If the package is already installed, install it again. For example, if you have installed `pypol_ v0.4`:

```
$ pyg install pypol_==0.4
Best match: pypol_==0.4
Downloading pypol_
Checking md5 sum
Running setup.py egg_info for pypol_
Running setup.py install for pypol_
pypol_ installed successfully
```

Later you may want to re-install the package. Instead of running **remove** and then **install**, you can use the `-U` option:

```
$ pyg install -U pypol_
Best match: pypol_==0.5
Downloading pypol_
Checking md5 sum
Installing pypol_ egg file
pypol_ installed successfully
```

This command **does not** upgrade dependencies (see *install -A*).

-A, -upgrade-all
New in version 0.5. Like, *install --upgrade*, but upgrade dependencies too.

-n, -no-deps
Do not install package's dependencies.

-i <url>, **-index-url** <url>
Specify the base URL of Python Package Index (default to `http://pypi.python.org/pypi`).

-d <path>, **-install-dir** <path>
The base installation directory for all packages.

-u, -user
Install the package in the user site-packages.

Uninstalling

Changed in version 0.5. Removing a package is dead simple:

```
$ pyg remove packname
```

Pyg tries to detect the package's folder and delete it:

```
$ pyg remove sphinx
Uninstalling sphinx
        /usr/bin/sphinx-build
        /usr/local/lib/python2.7/dist-packages/Sphinx-1.0.7-py2.7.egg
        /usr/bin/sphinx-quickstart
        /usr/bin/sphinx-autogen
Proceed? (y/[n])
```

If *packname* is a module and not a package, Pyg will automatically detect it:

```
$ pyg remove roman
Uninstalling roman
        /usr/local/lib/python2.7/dist-packages/roman.pyc
        /usr/local/lib/python2.7/dist-packages/roman.py
Proceed? (y/[n])
```

If your answer is *yes* the files will be deleted. This operation is **not undoable**:

```
$ pyg remove itertools_recipes
Uninstalling itertools_recipes
        /usr/local/lib/python2.7/dist-packages/itertools_recipes-0.1-py2.7.egg
Proceed? (y/[n]) y
Deleting: /usr/local/lib/python2.7/dist-packages/itertools_recipes-0.1-py2.7.egg...
Removing egg path from easy_install.pth...
itertools_recipes uninstalled succesfully
```

-y, -yes
Do not ask confirmation of uninstall deletions:

```
$ pyg remove -y iterutils
Uninstalling iterutils
        /usr/local/lib/python2.7/dist-packages/iterutils.py
        /usr/local/lib/python2.7/dist-packages/iterutils-0.1.6.egg-info
        /usr/local/lib/python2.7/dist-packages/iterutils.pyc
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils.py...
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils-0.1.6.egg-info...
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils.pyc...
Removing egg path from easy_install.pth...
iterutils uninstalled succesfully
```

-r <path>, **-req-file** <path>
Uninstall all the packages listed in the given requirement file.

```
$ echo -e 'itertools_recipes\niterutils' > reqfile.txt
$ cat reqfile.txt
itertools_recipes
iterutils

$ pyg remove -r reqfile.txt
Uninstalling itertools_recipes
        /usr/local/lib/python2.7/dist-packages/itertools_recipes.py
        /usr/local/lib/python2.7/dist-packages/itertools_recipes.pyc
        /usr/local/lib/python2.7/dist-packages/itertools_recipes-0.1.egg-info
Proceed? (y/[n]) y
Deleting: /usr/local/lib/python2.7/dist-packages/itertools_recipes.py...
Deleting: /usr/local/lib/python2.7/dist-packages/itertools_recipes.pyc...
Deleting: /usr/local/lib/python2.7/dist-packages/itertools_recipes-0.1.egg-info...
Removing egg path from easy_install.pth...
itertools_recipes uninstalled succesfully
Uninstalling iterutils
        /usr/local/lib/python2.7/dist-packages/iterutils.py
        /usr/local/lib/python2.7/dist-packages/iterutils-0.1.6.egg-info
        /usr/local/lib/python2.7/dist-packages/iterutils.pyc
Proceed? (y/[n]) y
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils.py...
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils-0.1.6.egg-info...
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils.pyc...
Removing egg path from easy_install.pth...
iterutils uninstalled succesfully
```

You can supply both `packname` (one or more) and requirement files:

```
$ pyg remove -r reqfile.txt docutils
Uninstalling itertools_recipes
        /usr/local/lib/python2.7/dist-packages/itertools_recipes.py
        /usr/local/lib/python2.7/dist-packages/itertools_recipes.pyc
        /usr/local/lib/python2.7/dist-packages/itertools_recipes-0.1.egg-info
Proceed? (y/[n]) y
Deleting: /usr/local/lib/python2.7/dist-packages/itertools_recipes.py
Deleting: /usr/local/lib/python2.7/dist-packages/itertools_recipes.pyc
Deleting: /usr/local/lib/python2.7/dist-packages/itertools_recipes-0.1.egg-info
Removing egg path from easy_install.pth...
itertools_recipes uninstalled succesfully
Uninstalling iterutils
        /usr/local/lib/python2.7/dist-packages/iterutils.py
        /usr/local/lib/python2.7/dist-packages/iterutils-0.1.6.egg-info
        /usr/local/lib/python2.7/dist-packages/iterutils.pyc
Proceed? (y/[n]) y
```

```
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils.py
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils-0.1.6.egg-info
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils.pyc
Removing egg path from easy_install.pth...
iterutils uninstalled succesfully
Uninstalling docutils
        /usr/local/lib/python2.7/dist-packages/docutils
        /usr/local/lib/python2.7/dist-packages/docutils-0.7.egg-info
Proceed? (y/[n]) y
Deleting: /usr/local/lib/python2.7/dist-packages/docutils
Deleting: /usr/local/lib/python2.7/dist-packages/docutils-0.7.egg-info
Removing egg path from easy_install.pth...
docutils uninstalled succesfully
```

Note: You can remove Pyg either with `pyg remove pyg` or `pyg remove yourself`. New in version 0.5.

3.1.2 Requirement files and bundles

Freezing requirements

Changed in version 0.7: From Pyg 0.7 onwards, this command has been renamed `pyg site`. When you launch:

```
$ pyg site
```

Pyg tries to detect all installed packages and prints requirements on Standard Output:

```
# Python version: '2.7.1+ (r271:86832, Apr 11 2011, 18:05:24) \n[GCC 4.5.2]'
# Python version info: '2.7.1.final.0'
# Python Prefix: '/usr'
# Platform: 'linux-i686'
# Pyg version: '0.6'

Brlapi==0.5.5
BzrTools==2.3.1
Cython==0.14.1
...
wadllib==1.1.8
wsgi-intercept==0.4
wsgiref==0.1.2
xkit==0.0.0
zope.interface==3.6.1
```

Note that the first lines – information about the site – are commented, so that if they're written into a requirement file, they will be ignored.

-f <path>, **-file** <path>
: Write requirements into the specified file. Equivalent to:

  ```
  $ pyg site > reqfile.txt
  ```

-c, -count
: Return the number of installed packages:

  ```
  $ pyg site -c
  55
  ```

-n, -no-info
Do not add site information:

```
$ pyg site -n
Brlapi==0.5.5
BzrTools==2.3.1
Cython==0.14.1
...
wadllib==1.1.8
wsgi-intercept==0.4
wsgiref==0.1.2
xkit==0.0.0
zope.interface==3.6.1
```

Bundles

The bundle format is specific to Pip (see Pip documentation). To create a bundle do:

```
$ pyg bundle app.pyb package_name
```

This will download all packages (including dependencies) and put them in a bundle file. Install packages from a bundle is dead simple, and you don't need internet access:

```
$ pyg install app.pyb
```

For example, here is `Pyg` bundle:

```
$ pyg bundle pyg.pyb pyg
pyg:
        Retrieving data for pyg [100% - 472.3 Kb / 472.3 Kb]
        Writing data into pyg-0.6.tar.gz
        pyg downloaded successfully
        Looking for pyg dependencies
                Found: setuptools
                Found: pkgtools>=0.4
                Found: argh>=0.14
argh>=0.14:
        Retrieving data for argh [100% - 11.4 Kb / 11.4 Kb]
        Writing data into argh-0.14.0.tar.gz
        argh downloaded successfully
        Looking for argh>=0.14 dependencies
pkgtools>=0.4:
        Retrieving data for pkgtools [100% - 28.7 Kb / 28.7 Kb]
        Writing data into pkgtools-0.6.tar.gz
        pkgtools downloaded successfully
        Looking for pkgtools>=0.4 dependencies
setuptools:
        Retrieving data for setuptools [100% - 250.8 Kb / 250.8 Kb]
        Writing data into setuptools-0.6c11.tar.gz
        setuptools downloaded successfully
        Looking for setuptools dependencies
Finished processing dependencies
Adding packages to the bundle
Adding the manifest file
```

You can download the generated example bundle from here (direct link to download).

-r <path>, **-req-file** <path>
New in version 0.5. Specify requirement files containing packages to add. This option can be repeated many

times:

```
$ pyg bundle bundlename.pybundle -r myreqs.txt -r other_reqs ...
```

-e <requirement>, **-exclude** <requirement>
New in version 0.5. Specify packages to exclude from the bundle (can be repeated many times):

```
$ pyg bundle pyg.pyb pyg -e argh -e "pkgtools<=0.3"
```

-d, -use-develop
New in version 0.7. If specified, for every package look for a local (*develop*) package. If it does not find it, it will download it from PyPI:

```
$ pyg bundle pyg pyg -d
pyg:
        Looking for a local package...
        Looking for pyg dependencies
                Found: setuptools
                Found: pkgtools>=0.6.1
                Found: argh
argh:
        Looking for a local package...
        Cannot find the location of argh
        Retrieving data for argh [100% - 11.4 Kb / 11.4 Kb]
        Writing data into argh-0.14.0.tar.gz
        argh downloaded successfully
        Looking for argh dependencies
pkgtools>=0.6.1:
        Looking for a local package...
        Looking for pkgtools>=0.6.1 dependencies
setuptools:
        Looking for a local package...
        Cannot find the location of setuptools
        Retrieving data for setuptools [100% - 250.8 Kb / 250.8 Kb]
        Writing data into setuptools-0.6c11.tar.gz
        setuptools downloaded successfully
        Looking for setuptools dependencies
Finished processing dependencies
Adding packages to the bundle
Adding the manifest file
```

Packs

New in version 0.7. Packs are zip files containing an egg (which includes all necessary packages) and some Python executable files (`run_name.py`). The **pack** command has the following syntax:

```
pyg pack {packname} {package} [{options}, ...]
```

Its name can either have the `.zip` extension or not, and can be a path.

You can create a pack with the following command:

```
$ pyg pack pyg.zip pyg
Generating the bundle...
pyg:
        Retrieving data for pyg [100% - 472.3 Kb / 472.3 Kb]
        Writing data into pyg-0.6.tar.gz
        pyg downloaded successfully
```

```
        Looking for pyg dependencies
                Found: setuptools
                Found: pkgtools>=0.4
                Found: argh>=0.14
argh>=0.14:
        Retrieving data for argh [100% - 11.4 Kb / 11.4 Kb]
        Writing data into argh-0.14.0.tar.gz
        argh downloaded successfully
        Looking for argh>=0.14 dependencies
pkgtools>=0.4:
        Retrieving data for pkgtools [100% - 27.2 Kb / 27.2 Kb]
        Writing data into pkgtools-0.6.1.tar.gz
        pkgtools downloaded successfully
        Looking for pkgtools>=0.4 dependencies
setuptools:
        Retrieving data for setuptools [100% - 250.8 Kb / 250.8 Kb]
        Writing data into setuptools-0.6c11.tar.gz
        setuptools downloaded successfully
        Looking for setuptools dependencies
Finished processing dependencies
Adding packages to the bundle
Generating EGG-INFO...
```

For example, Pyg Pack has the following structure:

```
pyg-0.6
    -- pyg.egg
    -- run_easy_install-2.3.py
    -- run_easy_install.py
    -- run_pyg.py
```

As you can see, there are already some executable files (Pyg looks for them in the packages' `entry_points.txt` file) and you can run them without installing Pyg: everything necessary is in `pyg.egg`. Amazing!

If you want to try it, download it from here (direct link to download), unpack it and run the `run_pyg.py` file. You will be able to use Pyg without installing it!

`-e` `<requirement>`**, `-exclude`** `<requirement>`
: Specify packages to exclude from the pack (can be repeated many times):

  ```
  $ pyg pack pyg.zip pyg -e argh -e "pkgtools<=0.3"
  ```

  Warning: If you exclude some packages from the pack it is very likely that its executables will not work, without some dependencies.

`-d, -use-develop`
: If specified, for every package look for a local (*develop*) distribution. If it does not find it, it will download it from PyPI.

  On certain systems (probably Unix-like ones) the **pack** command with this option enabled may require root privileges, because Pyg will run the **sdist** command (`python setup.py sdist`) for every local distribution.

  (See also *`bundle -d`*.)

3.1.3 Upgrading, downloading and Pyg Shell

Upgrading installed packages

New in version 0.3. When you use the `update` command, Pyg searches through all installed packages and checks for updates. If there are some, Pyg installs them.

Before loading the entire list of installed packages, Pyg checks the `~/.pyg/installed_packages.txt` file. If it exists Pyg will update only packages in that file:

```
$ pyg update
Cache file not found: $HOME/.pyg/installed_packages.txt
Loading list of installed packages...
15 packages loaded
Searching for updates
A new release is avaiable for simplejson: 2.1.5 (old 2.1.3)
Do you want to upgrade? (y/[n]) y
Upgrading simplejson to 2.1.5
        Installing simplejson-2.1.5.tar.gz...
            Installing simplejson-2.1.5.tar.gz
            Running setup.py egg_info for simplejson
            Running setup.py install for simplejson
            simplejson installed successfully

...

Updating finished successfully

$ pyg update
Loading list of installed packages...
Reading cache...
15 packages loaded
Searching for updates
A new release is avaiable for wadllib: 1.2.0 (old 1.1.8)
Do you want to upgrade? (y/[n]) n
wadllib has not been upgraded
A new release is avaiable for launchpadlib: 1.9.8 (old 1.9.7)
Do you want to upgrade? (y/[n]) n
launchpadlib has not been upgraded
Updating finished successfully
```

Downloading packages

New in version 0.2. If you only need to download a package you can use the `download` command:

```
$ pyg download packname
```

If the requirement is not satisfied Pyg won't download anything:

```
$ pyg download pyg==1024
E: Did not find files to download
```

-u, -unpack
After downloading a package, Pyg unpacks it:

```
$ pyg download -u pypol_
Found egg file for another Python version: 2.6. Continue searching...
Retrieving data for pypol_
```

```
Writing data into pypol_-0.5-py2.7.egg
pypol_ downloaded successfully
Unpacking pypol_-0.5-py2.7.egg to ./pypol_-0.5-py2.7
$ l
pypol_-0.5-py2.7/  pypol_-0.5-py2.7.egg
```

-d <path>, **-download-dir** <path>
Where to download the package, default to `.` (current working directory):

```
$ pyg download -d /downloads/python_downloads/ pyg
```

If the path does not exist, Pyg will create it.

-p <ext>, **-prefer** <ext>
The preferred file type for the download. Pyg looks for that file type and, if it does not exists, will try another extension:

```
$ pyg download -p .tar.gz pyg
Retrieving data for pyg
Writing data into pyg-0.1.tar.gz
pyg downloaded successfully

$ pyg download -p .egg pyg
Retrieving data for pyg
Writing data into pyg-0.1-py2.7.egg
pyg downloaded successfully

$ pyg download -p .myawesomeext pyg
Retrieving data for pyg
Writing data into pyg-0.1-py2.7.egg
pyg downloaded successfully
```

Pyg Shell

New in version 0.4. If you need to execute many Pyg commands and you need root privileges (for example on Unix systems), you can fire up Pyg Shell and you are done:

```
$ pyg shell
```

Now you can use all Pyg's commands plus 3 shell commands: **cd**, **pwd**, and **ls**:

```
pyg:/home/user$ check pyg
True
pyg:/home/user$ check pyg==0.3.2
True
pyg:/home/user$ ls
pkgtools  pyg
pyg:/home/user$ pwd
/home/user
pyg:/home/user$ cd pyg
pyg:/home/user/pyg$ pwd
/home/user/pyg
pyg:/home/user/pyg$ install sphinx
sphinx is already installed
pyg:/home/user/pyg$ install -U sphinx
sphinx is already installed, upgrading...
Looking for sphinx releases on PyPI
Best match: Sphinx==1.0.7
```

```
Downloading Sphinx
Checking md5 sum
Running setup.py egg_info for Sphinx
Running setup.py install for Sphinx
Installing dependencies...
    Jinja2>=2.2 is already installed
    docutils>=0.5 is already installed
    Pygments>=0.8 is already installed
Finished installing dependencies
Sphinx installed successfully
pyg:/home/user/pyg$ cd
pyg:/home/user$ exit
```

Minor commands:

3.1.4 Some minor stuff

The `list` command

You can use this command to list all package's avaiable versions:

```
$ pyg list pypol_
0.5 installed
0.4
0.3
0.2

$ pyg list itertools_recipes
0.1
```

If that package is installed, Pyg will add `installed` after the current version.

Checking installed packages

If you want to check if a package is installed, you can use the `check` command:

```
$ pyg check packname
```

Some examples:

```
$ pyg check pyg
True
$ pyg check pyg==42
False
$ pyg check pyg==0.1.2
True
$ pyg check pyg==0.1.3
False
```

Searching PyPI

Pyg can perform searches on PyPI with the `search` command:

```
$ pyg search pypol_
pypol_  0.5 - Python polynomial library
pypolkit  0.1 - Python bindings for polkit-grant

$ pyg search distribute
distribute  0.6.15 - Easily download, build, install, upgrade, and uninstall Python packages
virtualenv-distribute  1.3.4.4 - Virtual Python Environment builder
```

Linking directories

If you want to add a directory to `PYTHONPATH` permanently the `link` command is what you need:

```
$ pyg link dirname
```

When you link a directory Pyg add in a `.pth` file the dir's path.

Unlinking

If you want to remove a directory from `PYTHONPATH` you can use the `unlink` command. Pyg can remove a directory from `PYTHONPATH` only if that directory has been added previously.

`-a, -all`
: Remove all links in the `.pth` file.

Other stuff:

3.1.5 Pyg's config files

New in version 0.4.Changed in version 0.8: From version 0.8 onwards, Pyg supports a `PYG_CONFIG_FILE` environment variable. Config files allow you to override command-line options, and to save them somewhere. During the initialization process, Pyg looks for configurations file, it this order:

- `PYG_CONFIG_FILE`: an environment variable pointing to the config file. This variable can contain multiple paths (separated by a color, `:`). (Introduced in Pyg 0.8)
- `./pyg.conf`: a configuration file in the current working directory:
- `~/pyg.conf`: where ~ stands for your `HOME` directory;
- `~/.pyg/pyg.conf`.

A config file has this structure:

```
[section_name]
option = value

[section_name2]
option1 = value1
option2 = value2

...
```

In addition to this, Pyg supports a nonstandard syntax, allowing multiple section names in a single header:

```
[section1 & section2 & section6]
option = value
```

It will set that option in all specified sections.

Note: If you set an option to `False`, `false`, or `0`, Pyg will consider it as false.

The `global` section

New in version 0.7. This section is a special one: there is no `global` command, and it refers to Pyg's global options. At the moment there is only one global option: *`--no-colors`*. You can set it as follows:

```
[global]
no_colors = True
```

Example usage

Example 1

`~/pyg.conf`:

```
[remove]
yes = True

[install]
upgrade_all = True
```

Pyg:

```
$ pyg install iterutils
Loading options from ~/pyg.conf
iterutils is already installed, upgrading...
Looking for iterutils releases on PyPI
Best match: iterutils==0.1.6
Downloading iterutils [100% - 2.9 Kb]
Checking md5 sum
Running setup.py egg_info for iterutils
Running setup.py install for iterutils
iterutils installed successfully
$ pyg remove iterutils
Loading options from ~/pyg.conf
Uninstalling iterutils
        /usr/local/lib/python2.7/dist-packages/iterutils.py
        /usr/local/lib/python2.7/dist-packages/iterutils.pyc
        /usr/local/lib/python2.7/dist-packages/iterutils-0.1.6-py2.7.egg-info
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils.py
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils.pyc
Deleting: /usr/local/lib/python2.7/dist-packages/iterutils-0.1.6-py2.7.egg-info
Removing egg path from easy_install.pth...
iterutils uninstalled succesfully
```

Example 2

`~/pyg.conf`:

```
[freeze]
count = True
```

Pyg:

```
$ pyg freeze
Loading options from ~/pyg.conf
84
```

Example 3

You can also override saved options from the command line. `pyg.conf`:

```
[install]
index_url = http://pypi.python.org/pypi
```

Pyg:

```
$ pyg install itertools_recipes -U --index-url = http://pypi.python.org/simple
itertools_recipes is already installed, upgrading...
Looking for links on http://pypi.python.org/simple
        Found: itertools_recipes-0.1.tar.gz
        Downloading itertools_recipes [100% - 2.3 Kb]
        Running setup.py egg_info for itertools_recipes
        Running setup.py install for itertools_recipes
itertools_recipes installed successfully
```

instead of:

```
$ pyg install -U itertools_recipes
itertools_recipes is already installed, upgrading...
Looking for itertools_recipes releases on PyPI
Best match: itertools_recipes==0.1
Downloading itertools_recipes [100% - 2.3 Kb]
Checking md5 sum
Running setup.py egg_info for itertools_recipes
Running setup.py install for itertools_recipes
itertools_recipes installed successfully
```

Option tree

Here is a list of all sections and their default options:

global:

- *no_colors* = False

install:

- *upgrade* = False
- *upgrade_all* = False
- *no_deps* = False
- *index_url* = `http://pypi.python.org/pypi`
- *install_dir* = `pyg.locations.INSTALL_DIR`
- *user* = False
- *no_scripts* = False
- *ignore* = False

remove:

- *yes* = False

bundle:

- *exclude* = None

update:

- *yes* = False

download:

- *unpack* = False
- *download_dir* = .
- *prefer* = None

freeze:

- *count* = False
- *file* = None

unlink:

- *all* = False

Development

4.1 CHANGELOG

4.1.1 1.0 (planning)

- #69: Write tests. We haven't chosen the test framework to use yet.
- #70: Upgrade `pyg.req.Requirement` to honor setuptools extras.

4.1.2 0.8 (in development)

Bugs fixed

Currently none.

Features added

- #79: Allow requirements with multiple version specification: `pyg==0.4,==0.8`.
- #80: Add a new method: `pyg.log.Logger.ask()`.
- #82: Allow comments in Pyg's configuration files.
- #83: Added support for environment variables like `PYG_CONFIG_FILE` and `PYG_HOME`.
- #96: Added an `argv=` keyword argument to `pyg.main()` function.

TODO

- #78: Allow installing packages from *dev* version: `pyg install "pyg==dev"`.
- #81: Read `~/.pypirc` file.
- #95: Add TAB autocompletion for Unix shells.
- #97: Optimize uninstaller file deletion.

4.1.3 0.7.1 (July 25, 2011)

- Fixed a serious bug in the **pack** command.

4.1.4 0.7 (July 16, 2011)

Bugs fixed

- #59: Solved `pyg.web.LinkFinder` issue once and for all.
- #62: `pyg bundle` failed when it did not find links on PyPI. Thanks to Fabien Devaux.
- #63: Pyg could not install packages which use distutils instead of setuptools. Thanks to Fabien Devaux.
- #64: Fixed `pyg install --user`.
- #66: `pyg install -d {dir}` could be ineffective.
- #73: Sometimes the **search** did not find anything (even if the package really exists).
- #74: Fixed VCS install error.
- #94: Error when bundling same packages.

Features added

- #57: Created an installer, like Pip's `get-pip.py`. You can grab it here: https://raw.github.com/rubik/pyg/master/get-pyg.py
- #58: Added a `pygx.y` program (where `x.y` is Python current version), in addition to `pyg`.
- #68: Now you can install local directories in development mode with the *`install -e`* option.
- #72: Default argparse help is incomplete: wrote Pyg's own HelpFormatter.
- #72b: Added colored output.
- #76: Allow installing eggs which requires a different Python installation (added an *`install --force-egg-install`* option).
- #77: Improved uninstaller's file-detection.
- #90: Now it is possible to create Packs: see *Packs*.
- #91: Replaced **pyg freeze** with **pyg site**.
- #92: Now it is possible to bundle local packages.

4.1.5 0.6 (May 15, 2011)

- #40: Now Pyg can install packages from URL even if the URL does not end with a file-extension.
- #44: Added a *`install --ignore`* option.
- #45: Replaced `pkgtools.WorkingSet` with `pkg_resources.working_set` in `pyg.inst.Updater`.
- #46: Added two new global command-line option: *`-d, --debug`* and *`--verbose`*.
- #47: The Updater is now faster and searches links on `http://pypi.python.org/simple` too.
- #48: Implement a special uninstallation system for `pyg.inst.Updater`.

- #53: Show download progress.
- #55: You can install packages from Github Gists!

4.1.6 0.5.2 (May 05, 2011)

- Fixed package installation from VCS.

4.1.7 0.5.1 (May 05, 2011)

- Fixed `setup.py`: added the `zip-safe` flag.

4.1.8 0.5 (May 05, 2011)

- #29: Create bundles from requirements files.
- #30: Keep track of why requirements are needed.
- #31: Replace **uninstall** and **rm** with a new **remove** command.
- #32: Follow links in `dependency_links.txt` file.
- #33: Fix option *`--index-url`*.
- #34: `pyg remove yourself`.
- #36: Add `virtualenv` support.
- #38: Add *`-A, --upgrade-all`* option for the **install** command.
- #39: Add *`-e, --exclude`* option for the **bundle** command.

4.1.9 0.4.1 (May 01, 2011)

- Fixed an issue with the `subprocess` module.

4.1.10 0.4 (May 01, 2011)

- #19: Added `pyg.inst.Bundler`: now Pyg can create bundles!
- #20: Installation from binaries (on Windows).
- #22: Support a config file somewhere.
- #25: Link following: when a package does not have any file on PyPI, Pyg have to follow links (e.g. package's home page, etc.) to find download links.
- #26: Added Pyg Shell.

4.1.11 0.3.2 (Apr 29, 2011)

- Fixed `setup.py`: didn't create Eggs properly.

4.1.12 0.3.1 (Apr 22, 2011)

- Fixed `setup.py`: Setuptools didn't save requirements correctly.

4.1.13 0.3 (Apr 18, 2011)

- #9: Added the **update** command.
- #11: Added VCS support.
- #12: Command-line options in requirement file are allowed.
- #16: Added the *`--no-scripts`* and *`--no-data`* options to the *install* command.
- #17: Added the possibility to install packages from directories.
- #23: Comments in requirement file are allowed.
- Added `pyg.inst.Updater`.
- Added a new file for the command-line options parser.

4.1.14 0.1.2 (Mar 26, 2011)

- #6: Added a **download** command.
- Added several options to the command-line parser.
- Fixed some strange behavior of `pyg.req.Requirement.install()`.

4.1.15 0.1.1

- #2: `pyg.inst.Installer` now download dependencies.
- #4: Make `pyg.types.Egg` installing entry points.
- #5: Fixed `pyg.inst.Uninstaller`.

4.1.16 0.1 (Mar 24, 2011)

- First release.