Yaclog

Release 1.2.1.dev3+g3094776

Andrew Cassidy

Apr 16, 2024

CONTENTS

Yaclog is a python library and command line tool to make it easier to keep track of changes to your projects. It includes commands for appending new changes to a markdown changelog file, as well as releasing new versions for deployment via git tags.

CHAPTER

ONE

HANDBOOK

1.1 Getting Started

1.1.1 Installation

Install and update using pip:

```
$ pip install -U yaclog
```

1.1.2 Usage

For detailed documentation on the **`yaclog`** command and its subcommands see the *Command Reference*.

Example workflow

Create a new changelog in the current directory:

```
$ yaclog init
```

Add some new entries to the "Added" section of the current unreleased version:

```
$ yaclog entry -b 'Introduced some more bugs'
$ yaclog entry -b 'Introduced some more features'
```

Show the current version:

```
$ yaclog show
```

```
Unreleased

- Introduced some more bugs
- Introduced some more features
```

Release the current version and make a git tag for it

```
$ yaclog release 0.0.1 -c
```

```
Renamed version "Unreleased" to "0.0.1".
Commit and create tag for version 0.0.1? [y/N]: y
Created commit a7b6789
Created tag "0.0.1".
```

1.2 Changelog Files

Yaclog works on Markdown changelog files, using a machine-readable format based on what is proposed by Keep a Changelog. Changelog files can be created using the **`yaclog init`** command.

1.2.1 Preamble

The preamble is the text at the top of the file before any version information. It can contain the title, an explanation of the file's purpose, as well as any general machine-readable information you may want to include for use with other tools. Yaclog does not provide any ways to manipulate the front matter from the command line due to its open-ended nature.

1.2.2 Versions

Version information begins with a header, which is an H2 containing the version's name, as well as optionally the date in ISO-8601 form, and any tag metadata. Some example version headers:

```
## 1.0.0
```

```
## 3.2.0 "Columbia" - 1981-07-20
```

```
## Version 8.0.0rc1 1988-11-15 [PRERELEASE]
```

Version names should (but are not required to) include a version number in **PEP 440** format, which is a superset of semantic versioning. Versions can be incremented or renamed using the **`yaclog release`** command.

1.2.3 Entries

Entries are individual changes made since the previous version. They can be paragraphs, list items, or any markdown block element. Entries can be either uncategorized, or organized into sections using H3 headers. Entries can be added using the **`yaclog entry`** command.

1.2.4 Tags

Tags are additional metadata added to a version header, denoted by all-caps text surrounded in square brackets. Tags can be used to mark that a version is a prerelease, that it has been yanked for security reasons, or for marking compatibility with some other piece of software. Tags can be added and removed using the **`yaclog tag`** command.

1.2.5 Example

```
# Changelog

All notable changes to this project will be documented in this file.

## 0.13.0 "Aquarius" - 1970-04-11 [YANKED]

Yanked due to issues with oxygen tanks, currently investigating

### Added

- Extra propellant in preparation for future versions

### Changed

- Replaced Ken Mattingly
- Stirred oxygen tanks

## 0.12.0 "Intrepid" - 1969-11-14

### Added

- New ALSEP package for surface science
- Color cameras
- Surface rendezvous with Surveyor 3

### Fixed

- 1201/1202 alarm distracting crew during landing

### Known Issues

- Lightning strike during launch: No effect on performance

## 0.11.0 "Eagle" - 1969-07-20

Initial stable release

### Changed

- Fully fueled lander to allow landing on the lunar surface
```

1.3 Command Reference

1.3.1 yaclog

Manipulate markdown changelog files.

```
yaclog [OPTIONS] COMMAND [ARGS]...
```

Options

--path <FILE>

Location of the changelog file.

Default
CHANGELOG.md

--version

Show the version and exit.

Environment variables

YACLOG_PATH

Provide a default for *--path*

entry

Add entries to SECTION in VERSION

SECTION is the name of the section to append to. If not given, entries will be uncategorized.

VERSION is the name of the version to append to. If not given, the most recent version will be used, or a new 'Unreleased' version will be added if the most recent version has been released.

```
yaclog entry [OPTIONS] SECTION VERSION
```

Options

-b, --bullet <TEXT>

Add a bullet point.

-p, --paragraph <TEXT>

Add a paragraph

Arguments

SECTION
Optional argument

VERSION
Optional argument

format

Reformat the changelog file.

```
yaclog format [OPTIONS]
```

init

Create a new changelog file.

```
yaclog init [OPTIONS]
```

release

Release VERSION, or a version incremented from the last release.

VERSION is the name of the version to release. If VERSION is not provided but increment options are, then the most recent valid PEP440 version number is used instead.

The most recent version in the log will be renamed (except by the –commit option) by using the VERSION as well as any increment options. Increment options will always reset the later segments, and prerelease increments will clear other kinds of prerelease.

```
yaclog release [OPTIONS] VERSION
```

Options

-M, --major
Increment major version number.

-m, --minor
Increment minor version number.

-p, --patch
Increment patch number.

-s, --segment <rel_seg>
Increment nth segment of the version. For example, `--segment 2` is equivalent to `--patch`

-a, --alpha
Increment alpha version number.

-b, --beta
Increment beta version number.

-r, --rc
: Increment release candidate version number.

-f, --full
: Clear the prerelease value creating a full release.

-c, --commit
: Create a git commit tagged with the new version number. If there are no changes to commit, the current commit will be tagged instead.

-C, -, --cargo
: Update the version in a Rust cargo.toml manifest file.

-y, --yes
: Answer "yes" to all confirmation dialogs

-n, --new
: Create a new version instead of renaming an existing one

Arguments

VERSION
: Optional argument

show

Show the changes for VERSIONS.

VERSIONS is a list of versions to print. If not given, the most recent version is used.

```
yaclog show [OPTIONS] VERSIONS
```

Options

-a, --all
: Show the entire changelog.

-m, --markdown, -t, --txt
: Display as markdown or plain text.

-f, --full
: Show version header and body.

-n, --name
: Show only the version name

-b, --body
: Show only the version body.

-h, --header
: Show only the version header.

Arguments

VERSIONS
Optional argument(s)

tag

Modify TAG on VERSION.

VERSION is the name of a version to add tags to. If not given, the most recent version is used.

```
yaclog tag [OPTIONS] TAG VERSION
```

Options

-a, --add, -d, --delete
Add or delete tags

Arguments

TAG
Required argument

VERSION
Optional argument

CHAPTER

TWO

API REFERENCE

2.1 changelog Module

Contains the *Changelog* class that represents a parsed changelog file that can be read from and written to disk as markdown, as well as the *VersionEntry* class that represents a single version within that changelog.

class VersionEntry(*name: str = 'Unreleased'*, *date: date | None = None*, *tags: List[str] | None = None*, *link: str | None = None*, *link_id: str | None = None*, *line_no: int | None = None*)

A serialized representation of a single version entry in a *Changelog*, containing the changes made since the previous version

Parameters

- **name** (*str*) – The version's name
- **date** (*Optional[datetime.date]*) – When the version was released
- **tags** – The version's tags
- **link** – The version's URL
- **link_id** – The version's link ID
- **line_no** – What line in the original file the version starts on

name: str

The version's name

date: date | None

When the version was released

tags: List[str]

The version's tags

link: str | None

The version's URL

link_id: str | None

The version's link ID, uses the version name by default when writing

line_no: int | None

What line the version occurs at in the file, or None if the version was not read from a file. This is not guaranteed to be correct after the changelog has been modified, and it has no effect on the written file

sections: `Dict[str, List[str]]`

The dictionary of change entries in the version, organized by section. Uncategorized changes have a section of an empty string.

classmethod from_header(*header: str*, *line_no: int | None = None*) → *VersionEntry*

Create a new version entry from a markdown header

Parameters

- **header** – A markdown header to parse
- **line_no** – Line number the header is on

Returns

a new VersionEntry with the header's information

add_entry(*contents: str*, *section: str = ''*) → None

Add a new entry to the version

Parameters

- **contents** – The contents string to add
- **section** – Which section to add to.

body(*md: bool = True*, *color: bool = False*) → str

Get the version's body as a string

Parameters

- **md** – Format headings as markdown
- **color** – Add color codes to the string for display in a terminal

Returns

The formatted version body, without the version header

header(*md: bool = True*, *color: bool = False*) → str

Get the version's header as a string

Parameters

- **md** – Format headings as markdown
- **color** – Add color codes to the string for display in a terminal

Returns

The formatted version header

text(*md: bool = True*, *color: bool = False*) → str

Get the version's contents as a string

Parameters

- **md** – Format headings as markdown
- **color** – Add color codes to the string for display in a terminal

Returns

The formatted version header and body

property released: `bool`

Returns true if a PEP440 version number is present in the version name, and has no prerelease segments

property version

Returns the PEP440 version number from the version name, or None if none is found

class Changelog(*path=None*, *preamble: str = '# Changelog\n\nAll notable changes to this project will be documented in this file'*)

A serialized representation of a Markdown changelog made up of a preamble, multiple versions, and a link table.

Contents will be automatically read from disk if the file exists

Parameters

- **path** – The changelog's path on disk.
- **preamble** (*str*) – The changelog preamble to use if the file does not exist.

path

The path of the changelog's file on disk

preamble: str

Any text at the top of the changelog before any version information, including the title. It can contain the title, an explanation of the file's purpose, as well as any general machine-readable information for use with other tools.

versions: List[*VersionEntry*]

A list of versions in the changelog, with the most recent version first

links: Dict[str, str]

Link definitions at the end of the changelog, as a dictionary of `{id: url}`

read(*path=None*) → None

Read a markdown changelog file from disk. The object's contents will be overwritten by the file contents if reading is successful.

Parameters
path – The changelog's path on disk. By default, *path* is used

write(*path=None*) → None

Write a changelog to a Markdown file.

Parameters
path – The changelog's path on disk. By default, *path* is used.

add_version(*index: int = 0*, **args*, ***kwargs*) → *VersionEntry*

Add a new version to the changelog

Parameters

- **index** – Where to add the new version in the log. Defaults to the top
- **args** – args to forward to the *VersionEntry* constructor
- **kwargs** – kwargs to forward to the *VersionEntry* constructor

Returns
The created entry

current_version(*released: bool | None = None*, *new_version: bool = False*, *new_version_name: str = 'Unreleased'*) → *VersionEntry*

Get the current version from the changelog

Parameters

- **released** – if the returned version should be a released version, an unreleased version, or None to return the most recent
- **new_version** – if a new version should be created if none exist.
- **new_version_name** – The name of the version to create if there are no matches and new_version is True.

Returns
The current version matching the criteria, or None if new_version is disabled and none are found.

get_version(*name: str | None = None*) → *VersionEntry*

Get a version from the changelog by name.

Parameters
name – The name of the version to get, or None to return the most recent. The first version with this value in its name is returned.

Returns
The first version with the selected name

2.2 markdown Module

Tools for parsing and manipulating markdown, including a very basic tokenizer.

strip_link(*text*)

Parses and removes any links from the input string

Parameters
text – An input string which may be a markdown link, either literal or an ID

Returns
A tuple of (name, url, id). If the input is not a link, it is returned verbatim as the name.

join(*segments: List[str]*) → str

Joins multiple lines of markdown by adding double newlines between them, or a single newline between list items

Parameters
segments – A list of strings to join

Returns
A joined markdown string

class Token(*line_no: int*, *lines: List[str]*, *kind: str*)

A single tokenized block of markdown, consisting of one or more lines of text.

line_no

Which line this block appears on in the original file

lines

The lines of text making up this block

kind

What kind of token this is. One of h[1-6], p, li or code

tokenize(*text: str*)

Tokenize a markdown string

The tokenizer is very basic, and only cares about the highest-level blocks (Headers, top-level list items, links, code blocks, paragraphs).

Parameters
text – input text to tokenize

Returns
A list of tokens and a dictionary of links

2.3 version Module

Various helper functions for analyzing and manipulating **PEP 440** version numbers, meant to augment the packaging.version module.

extract_version(*version_str: str*) → Tuple[Version | None, int, int]

Extracts a **PEP 440** version object from a string which may have other text

Parameters
version_str – The input string to extract from

Returns
A tuple of (version, start, end), where start and end are the span of the version in the original string

increment_version(*version_str: str*, *rel_seg: int = None*, *pre_seg: str = None*) → str

Increment the **PEP 440** version number in a string

Parameters

- **version_str** – The input string to manipulate
- **rel_seg** – Which segment of the "release" value to increment, if any
- **pre_seg** – Which kind of prerelease to use, if any. An empty string clears the prerelease field.

Returns
The original string with the version number incremented

join_version(*epoch*, *release*, *pre*, *post*, *dev*, *local*) → str

Join multiple segments of a **PEP 440** version

is_release(*version_str: str*) → bool

Check if a version string is a release version

Parameters
version_str – the input string to check

Returns
True if the input contains a released **PEP 440** version, or False if a prerelease version or no version is found

CHAPTER

THREE

CHANGELOG

All notable changes to this project will be documented in this file

3.1 Version 1.2.0 - 2024-04-16

3.1.1 Added

- added the `-s` option to `yaclog release` to increment arbitrary version segments
- added the `-n` option to `yaclog release` to create a new release instead of releasing a new one
- added the `-y` option to `yaclog release` to answer "yes" to all confirmation dialogs. Use with caution!

3.2 Version 1.1.2 - 2022-12-29

3.2.1 Changed

- yaclog now only tries to use git when invoked with a command that needs it, meaning most sub commands can now be used on systems without git

3.3 Version 1.1.1 - 2022-08-15

3.3.1 Fixed

- Fixed `yaclog release -C -c` not committing changes to cargo.toml

3.4 Version 1.1.0 - 2022-08-14

3.4.1 Added

- Added a flag to update Rust Cargo.toml files when releasing a new version

3.5 Version 1.0.4 - 2022-04-08

3.5.1 Fixed

- Fixed tests folder being installed as a package

3.6 Version 1.0.3 - 2021-05-12

3.6.1 Fixed

- Fixed `show` command not working with Click version 8
- Fixed release message incorrectly stating if a commit will be created or not

3.7 Version 1.0.2 - 2021-05-12

3.7.1 Changed

- Updated to support Click version 8
- Modified module documentation page titles to include a module role

3.7.2 Fixed

- Fixed tag names with spaces in versions

3.8 Version 1.0.1 - 2021-05-10

3.8.1 Fixed

- Fixed broken header in new changelogs
- Improved consistency in command documentation metavars

3.9 Version 1.0.0 - 2021-05-07

3.9.1 Changed

- API changes:
  - `header` attribute renamed to `preamble` to avoid confusion.
- improved version header parsing to be more robust and handle multi-word version names.
- improved version number incrementing in `release`.
  - can now handle other text surrounding a pep440-compliant version number, which will not be modified

  - can now handle pre-releases correctly. The version to increment is the most recent version in the log with a valid pep440 version number in it.
  - Release increment and prerelease increments can be mixed, allowing e.g: `yaclog release -mr` to create a release candidate with in incremented minor version number.
- `release` base version is now an argument instead of an option, for consistency with other commands.

3.9.2 Removed

- `entry` with multiple `-b` options no longer add sub bullet points, instead adding each bullet as its own line.

3.9.3 Added

- Terminal output has color to distinguish version names/headers, sections, and git information.
- Extra newlines are added between versions to improve readability of the raw markdown file.

3.10 Version 0.3.3 - 2021-04-27

3.10.1 Added

- Unit tests in the `tests` folder

3.10.2 Fixed

- Default links and dates in VersionEntry are now consistently `None`
- Changelog links dict now contains version links. Modified version links will overwrite those in the table when writing to a file
- Changelog object no longer errors when creating without a path.
- `release` now resets lesser version values when incrementing
- `release` now works with logs that have only unreleased changes

3.11 Version 0.3.2 - 2021-04-24

3.11.1 Added

- Readme file now has installation and usage instructions.
- yaclog command entry point added to setup.cfg.

3.11.2 Changed

- `release -c` will no longer create empty commits, and will use the current commit instead.

3.11.3 Fixed

- `release` and `entry` commands now work using empty changelogs.

3.12 Version 0.3.1 - 2021-04-24

3.12.1 Added

- `yaclog` tool for manipulating changelogs from the command line
  - `init` command to make a new changelog
  - `format` command to reformat the changelog
  - `show` command to show changes from the changelog
  - `entry` command for manipulating entries in the changelog
  - `tag` command for manipulating tags in the changelog
  - `release` command for creating releases

3.13 Version 0.2.0 - 2021-04-19

3.13.1 Added

- New yak log logo drawn by my sister

3.13.2 Changed

- Updated package metadata
- Rewrote parser to use a 2-step method that is more flexible.
  - Parser can now handle code blocks.
  - Parser can now handle setext-style headers and H2s not conforming to the schema.

3.14 Version 0.1.0 - 2021-04-16

First release

3.14.1 Added

- `yaclog.read()` method to parse changelog files

CHAPTER

FOUR

GNU AFFERO GENERAL PUBLIC LICENSE

Version 3, 19 November 2007 Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

4.1 Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: **(1)** assert copyright on the software, and **(2)** offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

4.2 TERMS AND CONDITIONS

4.2.1 0. Definitions

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that **(1)** displays an appropriate copyright notice, and **(2)** tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

4.2.2 1. Source Code

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that **(a)** is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and **(b)** serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

4.2.3 2. Basic Permissions

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

4.2.4 3. Protecting Users' Legal Rights From Anti-Circumvention Law

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4.2.5 4. Conveying Verbatim Copies

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

4.2.6 5. Conveying Modified Source Versions

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- **a)** The work must carry prominent notices stating that you modified it, and giving a relevant date.
- **b)** The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- **c)** You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

- **d)** If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

4.2.7 6. Conveying Non-Source Forms

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- **a)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- **b)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either **(1)** a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or **(2)** access to copy the Corresponding Source from a network server at no charge.
- **c)** Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- **d)** Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- **e)** Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either **(1)** a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or **(2)** anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

4.2.8 7. Additional Terms

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- **a)** Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- **b)** Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- **c)** Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- **d)** Limiting the use for publicity purposes of names of licensors or authors of the material; or
- **e)** Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- **f)** Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

4.2.9 8. Termination

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated **(a)** provisionally, unless and until the copyright holder explicitly and finally terminates your license, and **(b)** permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

4.2.10 9. Acceptance Not Required for Having Copies

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

4.2.11 10. Automatic Licensing of Downstream Recipients

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

4.2.12 11. Patents

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either **(1)** cause the Corresponding Source to be so available, or **(2)** arrange to deprive yourself of the benefit of the patent license for this particular work, or **(3)** arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license **(a)** in connection with copies of the covered work conveyed by you (or copies made from those copies), or **(b)** primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

4.2.13 12. No Surrender of Others' Freedom

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

4.2.14 13. Remote Network Interaction; Use with the GNU General Public License

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the

resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

4.2.15 14. Revised Versions of this License

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

4.2.16 15. Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

4.2.17 16. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4.2.18 17. Interpretation of Sections 15 and 16

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

4.3 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see <http://www.gnu.org/licenses/>.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

y

INDEX

Symbols