# sphinxgen Documentation

***Release 1.0 (v1.0.0.0-x-dev)***

**Brian Mearns**

November 20, 2014

Contents

*sphinxgen* is used to generate files from python packages and modules found in a specified set of directories. It is intended for generating sphinx autodoc stub files, but can be used for other purposes. Files are generated using jinja templates to provide maximum flexbility. You can use a set of built-in templates, or provide your own.

CHAPTER 1

# Documentation Contents:

## 1.1 README

**Page Contents**



### 1.1.1 tl;dr

#### What?

**sphinxgen** is used to generate files from python packages and modules found in a specified set of directories. It is intended for generating sphinx autodoc stub files, but can be used for other purposes. Files are generated using jinja templates to provide maximum flexbility. You can use a set of built-in templates, or provide your own.

#### Install?

Install with pip:

```
$ pip install sphinxgen
```

Or, from source:

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

#### Examples?

```
> sphinxgen -o sphinx/source src/python/my_package src/python/my_other_package
```

You can also use it as a setuptools command:

```
#setup.cfg

[sphinxgen]
package_dirs = src/python/my_package,src/python/my_other_package
output = sphinx/source
```

```
> setup.py sphinxgen
```

#### Dependencies?

sphinxgen is developed against python version 2.7.

Other dependencies are handled by pip.

For building the docs from source with sphinx, you will need the packages listed under `sphinx/requirements.pip`.

#### Docs?

- Read The Docs (.org)
- Python Hosted (.org)

### 1.1.2 Misc.

#### Contact Information

This project is currently hosted on bitbucket, at https://bitbucket.org/bmearns/sphinxgen. The primary author is Brian Mearns, whom you can contact through bitbucket at https://bitbucket.org/bmearns.

#### Copyright and License

## 1.2 Command Line Interface

### 1.2.1 sphinxgen

Generate sphinx stub files for all modules in a python package.

**package_dir**
The package directories to process.

**-h, --help**
show this help message and exit

**-o** <output>**, --output** <output>
The directory where output will be written. The default is the current directory.

**--prefix** <prefix>
A prefix to use for every generated file name. If –index is used, the prefix will not be used for the generated index file.

**--overwrite**
Overwrite any existing files.

**-n, --dry-run**
Do a dry run, do not actually generate any files, just print what would happen if we did.

**--index** <path>
The path to the index file to generate (without extension). The default is “index”, with appropriate prefix as specified by the –prefix option. if you explicitly use this option, the prefix will not be added.

**--no-index**
Do not generate an index file.

**--no-modules**
Do not generate separate files for modules, only packages.

**--package-template** <path>
The path to the jinja2 template file to use for generating package files. If not given, a built-in template will be used.

**--module-template** <path>
The path to the jinja2 template file to use for generating module files. See –package-template for more details. The default is “module.rst”.

**--index-template** <path>
The path to the jinja2 template file to use for generating the index file. See –package-template for more details. The default is “index.rst”.

**--dump-templates** <template_dir>
Dump the built-in template files to the specified directory. This is useful as a starting point for creating your own template files. The template files are named package.rst, module.rst, and index.rst.

**--debug**
Print detailed logs for debugging.

**--version**
show program’s version number and exit

## 1.3 Jinja Template Contexts

*sphinxgen* uses jinja templates for generating the output files.

There are three types of files generated, each has its own template: *package* files, *module* files, and *index* files. The following sections describe the template contexts available for each type of file:

### 1.3.1 The Package Template

The *package* files are used for python packages, which are directories that contain an `__init__.py` file. Packages may contain *modules* as well as subpackages. The purpose of the package file is generally to document the top-level contents of the package (i.e., objects in the packages `__init__.py` module) and link to the documentation for the contained modules and sub packages. Alternatively, you can use the *`--no-modules`* option to supress generation of individual module files, and document the modules directly in the package documentation file.

The context for the package template is a dictionary describing the package itself. The following variables are defined for use in the template:

**`name`**
: The base name of the package, without any hierarchical information. This is simply the name of the package directory. For instance, the `name` of package `foo.bar.baz` is simply `'baz'`.

**`doc_name`**
: The name of the document being generated. This is the basename of the file, without extension (`.rst`) or directory. This includes any prefix specified by the *`--prefix`* option.

**`package`**
: The fully qualified name of the parent package, if there is any, or `None` if this appears to be a top-level package. For instance, for package `foo.bar.baz`, this value with be `'foo.bar'`, where as for package `foo`, it would be `None`.

  Note that we don't actually search the file system for parent packages, packages that were specified as a `PACKAGE_DIR` on the command line are assumed to be top-level packages.

**`fullname`**
: This is simply the fully qualified name of the package, which includes the package's base `name` and it's parent `package`. For `foo.bar.baz`, this would be `'foo.bar.baz'`.

**`path`**
: The filesystem path for where this directory is defined. These are derived from the `PACKAGE_DIR` values specified on the command line, and are not resolved to absolute or normalized paths.

**`modules`**
: A list of dictionaries describing each of the modules contained in this package. These are the same objects that will be used as the context for the *module template*; see that section for a description of these objects.

  Note that the order of the list is arbitrary, so you may want to sort it. Also note that this only contains immediate children of the package, no deeper descendants.

**`sub_packages`**
: A list of dictionaries describing each of the sub-packages contained in this package. These are the same objects that provide the context for the package template to generate the files for those packages, so they have the same structure as this dictionary itself.

  Note that the order of the list is arbitrary, so you may want to sort it. Also note that this only contains immediate children of the package, no deeper descendants.

**`children`**
: For convenience, this field is a concatenation of `sub_packages` and `modules`.

### 1.3.2 The Module Template

The *module* files are used to document individual python modules, each of which corresponds to a single python source file.

Basic information about each module is placed in a dictionary which provides the context for this template. The same dictionary is also used to represent the module in the `modules` member of the `package` template context.

The following variables are defined for use in the template (note that these are all essentially duplicates of the fields in the `package context`):

**name**
: The base name of the module, without any hierarchical information. This is simply the base name of the module's file. For instance, the `name` of module `foo.bar.baz` is simply `'baz'`.

**doc_name**
: The name of the document being generated. This is the basename of the file, without extension (`.rst`) or directory. This includes any prefix specified by the *`--prefix`* option.

**package**
: The fully qualified name of the parent package. For instance, for module `foo.bar.baz`, this value with be `'foo.bar'`.

**fullname**
: This is simply the fully qualified name of the module, which includes the modules's base `name` and it's parent `package`. For `foo.bar.baz`, this would be `'foo.bar.baz'`.

**path**
: The filesystem path to this module's source file. These are derived from the `PACKAGE_DIR` values specified on the command line, and are not resolved to absolute or normalized paths.

### 1.3.3 The Index Template

A single *index* file is generated by an invocation of **sphinxgen**. It's intended purpose is to create a `toctree` for all of the top-level packages specified on the command line. It can also be used to provide high-level information about the project if appropriate.

There is only a single variable defined for this template:

**packages**
: A list of all the top-level packages processed by **sphinxgen**. Each element is a dictionary describing the package, as documented under the `package` section, above. These appear in the same order as they were specified on the command line.

## 1.4 `sphinxgen` python package

### 1.4.1 `sphinxgen.version` module

The `version` module provides version numbering for the entire package.

**Page Contents**



## Versioning

This packages uses a five part version number, plus an incremental release number. Either the version number or the release number can be used to identify a released version of the code.

### Version Number

The version number is a four part dotted number, with an optional suffix on the end. Formally, a version number looks like:

```
version number  ::=   <Major>.<minor>[.<patch>[.<semantic>]][-[x-]<suffix>]
```

With each new released version of the code, exactly one of the four numbers will increase, and any numbers to its right will reset to 0.

The easiest way to understand version numbers is from the perspective of someone who has written *client code*: i.e., code that makes use of a particular version of the library. From this perspective, the version number indicates whether or not your client code can be expected to work with different versions of this package.

**Major Version** The `<Major>` component is the **major version number**, and it describes *backward compatibility*. Going to a *newer* version of the package, your code should continue to work as long as the major version doesn't change.

The major version is changed only when something is removed from the public interface. For instance, if a function is no longer supported, the major version number would have to increase, because client code which relied on that function would no longer work.

The major version number can be accessed through the `MAJOR` member of this module.

**Minor Version** The `<minor>` component is the **minor version number**, and it describes *forward compatibility*: Going to an *older* version of the package, your code will continue to work as long as the minor version doesn't change. (As before, your code will also work for *newer* versions, as long as the major version number hasn't changed).

The minor version number is changed only when something is added to the public interface, for instance a new function is added. Such a change maintains *backward* compatibility (as described above), but *loses forward compatibility*, because any client code written again this new version may not work with an older version.

The minor version number can be accessed through the `MINOR` member of this module.

**Patch Version** The `<patch>` component is the **patch number**, and it describes changes that *do not affect compatibility*, either forwards or backwards. Your client code will continue to work with an older or newer version of the package as long as the major and minor version numbers are the same, regardless of the patch number.

Patch changes are code changes that do not effect the interface, for instance bug-fixes or performance enhancements. (although some bugs effect the interface and may therefore cause a higher version number to change).

The patch number can be accessed through the `PATCH` member of this module.

**Semantic Version** The `<semantic>` component is the **semantic version number**, and it describes changes that do not affect how the code runs at all. Ths generally means that documentation or other auxilliary files included in the package have changed.

The semantic version number can be accessed through the `SEMANTIC` member of this module.

**Compatibility Summary** The following table summarizes compatibility for a hypothetical client application built against released version `M.n.p.s`:

| Component | Compatibile (all) | Incompatible (any) |
|---|---|---|
| Major | M | != M |
| minor | >= n | < n |
| patch | any | |
| semantic | any | |

**Version Suffix** The `<suffix>` component is the **version suffix**, which is used only for non-released code. The suffix has one of the following forms:

```
version suffix  ::=   << empty >>
                      dev[-<rev>]
                      blood-<branch>[-<rev>]
```

The first form is an empty suffix, and is reserved for released (tagged) code only.

The second form, `"dev"`, is for non-released code in the *trunk*. This is the main line of development. Dev code may not be completely functional, and may even break the existing interface.

The third form, `"blood-..."`, if for non-released code on a *branch*. The `<branch>` component of this form should be the name of the branch. This is considered *bleeding-edge* code and may be highly unstable.

The optional `<rev>` component on both the second and third forms can be used to specify a specific revision for comitted development code. This must be an globally unambiguous identifier for the revision, for instance the change set id.

**Development code** A non-empty version suffix indicates a *development version* of the code. In this case, the four version numbers remain *unchanged* until the code is released (in which case it is no longer development code, and the suffix is changed to empty).

In other words, anytime you see a non-empty version suffix, the version numbers shown refer to version from which the development code is derived. This is done because it is not generally known until release what the next released version number will be, since it is not known what types of changes will be included in it.

**Specifying a version number** When specifying a version number, the major and minor version numbers should always be included. Additionally, all non-zero version numbers should be included, and any version number to the left of a non-zero version number should be included.

The suffix should always be included in the version number, with the indicated hyphen separating the semantic version number and the suffix. The only exception is for released code, in which case the suffix is empty and should be omitted, along with the joining hyphen.

The optional `"x-"` shown preceding the suffix in the version number is for compatibility with setup-tools so that versions compare correctly.

The above rules will unambiguously describe any released version of the package.

**Interface Version** Because any change to the public interface requires a change to either the major or minor version numbers, the interface can be specified by a shortened two part version:

```
interface version  ::=  <Major>.<minor>
```

Note that this only applies for released versions: development versions may modify the public interface prior to changing the version numbers.

#### Release Number

The release number is a simple integer which increments by one for every public release of the code. It does not convey any information about compatibility with other versions, but it does provide a simple alternative to identifying released versions.

The release number should be written with a leading `"r"` or `"rel"`. For instance, the first release was `"r1"`.

For release code, the release number may be used in place of the suffix in the version number. This is optional because the version number and the release number are synonymous. However, including them both in the version string is a useful way to provide both pieces of information.

This alternative form of the version number is:

```
alt.  version number  ::=  <Major>.<minor>[.<patch>[.<semantic>]]-r<release>
```

### Module Contents

`sphinxgen.version.`**`RELEASE = 1`**
The current Release Number.

`sphinxgen.version.`**`MAJOR = 1`**
The current *major version number*.

`sphinxgen.version.`**`MINOR = 0`**
The current *minor version number*.

`sphinxgen.version.`**`PATCH = 0`**
The current *patch version number*.

`sphinxgen.version.`**`SEMANTIC = 0`**
The current *semantic version number*.

sphinxgen.version.**SUFFIX = 'dev'**
The current Version Suffix.

Suffix options are None, "dev", and "blood-"

- None means this is a released/tagged version.
- "dev" means this is a development version from the trunk/mainline.
- "blood-" means it's on a branch. After the dash, fill in the name of the branch.

Dev and blood versions are still numbered for the *previous* version, because we may not know what the next version will be until we're finished.

sphinxgen.version.**COPYRIGHT = 2014**
The copyright year for the code.

sphinxgen.version.**YEAR = 2014**
The year in which the code was released.

**See also:**

- MONTH
- DAY
- datestr

sphinxgen.version.**MONTH = 11**
The month in which the code was released. This is 1 indexed, in [1, 12].

**See also:**

- YEAR
- DAY
- datestr
- MONTH_NAMES

sphinxgen.version.**DAY = 20**
The day of the month on which the code was released.

**See also:**

- YEAR
- MONTH
- datestr

sphinxgen.version.**MONTH_NAMES = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']**
A sequence giving the names of months, for use by datestr. Standard values are three-letter English-language abbreviations for the months of the Gregorian calendar.

sphinxgen.version.**setuptools_string**()
Returns the version string used by setuptools. This takes one of two forms:

```
setuptools_string  ::=   <Major>.<minor>.<patch>.<semantic>-x-<suffix>
                         <Major>.<minor>.<patch>.<semantic>-r<release>
```

The first form is used for development code (i.e., when SUFFIX is not None), and the second it used for released code.

This is similar to string, except for the additional x- for development versions, which is used to ensure that setuptools sorts versions correctly. (specifically, so that released versions are earler than development versions which are derived from them).

sphinxgen.version.**tag_name**()
Returns the tag name for the most recent release.

sphinxgen.version.**short_string**()
Returns a string describing the Interface Version (i.e., <Major>.<minor>).

sphinxgen.version.**string**()
Like setuptools_string, except leaves out the x- for development versions.

sphinxgen.version.**datestr**()
Returns a simple string giving the date of release. Format of this string is unspecified, it intended to be human readable, not machine parsed. For machine processing, use the individual variables, as listed below.

**See also:**

- YEAR
- MONTH
- DAY
- MONTH_NAMES

A utility module for generating basic sphinx rest files for each module and package specified, recursively.

---

**Note:** Please forgive me for this crappy code and crappier documentation. I hacked this together while working on another project. It's worth refactoring, if you'd like to do so, please feel free and send me a pull request: https://bitbucket.org/bmearns/sphinxgen

---

**class** sphinxgen.**TemplateLoader**(*args*)
Bases: jinja2.loaders.BaseLoader

A jinja2 template loaded that loads templates needed by SphinxGen.

**get_source**(*environment*, *template*)
Returns the source for the named template.

**class** sphinxgen.**SphinxGen**(*parsed_args*, *log=None*)
Bases: object

The class that actually does the work.

Initializing an instance runs the operation as well.

**Parameters parsed_args** – This should be the namespace object returned by the argument_parser after parsing the command line options.

**generate_output**(*template*, *context*, *opath*)
Writes output to the specified file. Given a jinja2 template object and the context for the template, it renders the template and writes the results to the specified path.

If the path exists, it is not modified unless overwrite is set. If dry_run is set, no output is actually generated.

**build_package**(*package_name*, *package_dir*, *output_dir*)
Does all the work (recursively) for a particular package, which is a directory with an `__init__.py` file in it. Returns `None` if the specified directory (`package_dir`) is not actually a package (has not `__init__.py` file). Otherwise returns a dictionary representing the some basic information about the package, it's subpackages, and its submodules (python files in the package directory).

Generates the file for the package as well, plus all it's submodules, and all its sub packages. This is done through `generate_output`, so if `dry_run` is set, nothing will actually be written out.

sphinxgen.**argument_parser = ArgumentParser(prog='sphinxgen', usage='\n %(prog)s [options] PACKAGE_DIR [PACI**
An `ArgumentParser` instance that can be used to parse the command line options for the command line program. It is populated when the module is constructed.

sphinxgen.**main**(*args=None*)
The command line program. If this module is invoked as the main module, this function is called.

This simply parses the `args` with `argument_parser`, and passes them on to the `SphinxGen` factory method.

**Parameters** **args** – A sequence of command line arguments (like `sys.argv`) or `None` (in which case `sys.argv` is used).

**class** sphinxgen.**sphinxgen**(*dist*, ***kw*)
Bases: `setuptools.Command`

A setuptools `setuptools.Command` for running `SphinxGen`.

**description = 'Generate base sphinx ReST files for python packages and modules.'**

**user_options = [('output=', 'o', 'The directory where output will be written. The default is the current directory.'), ('p**

**initialize_options**()

**finalize_options**()

**run**()

## 1.5 LICENSE (GPLv3)

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 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.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is 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.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.
```

```
  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.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

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

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU 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.

  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.

  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.

  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. 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.

  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.

  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.

  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.

  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.

  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.

  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.

  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.

  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.

  13. Use with the GNU Affero General Public License.

  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 Affero 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 special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU 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 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 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 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.

  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.

  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.

  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

            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 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 General Public License for more details.

    You should have received a copy of the GNU 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 the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
```

```
might be different; for a GUI interface, you would use an "about box".

  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 GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

CHAPTER 2

# Indices and tables

- *genindex*
- *modindex*
- *search*

CHAPTER 3

# Version

This documentation is for sphinxgen 1.0 (v1.0.0.0-x-dev).

CHAPTER 4

# Project Resources

- sphinxgen project homepage (bitbucket)
- sphinxgen on pypi
- **Online documentation:**
  - Read The Docs (.org)
  - Python Hosted (.org)

## S

## Symbols



## A



## B



## C



## D



## F



## G



## I



## M



## N



## P



## R

## S



## T



## U



## Y