cookiecutter Documentation

Release 2.1.2.dev0

Audrey Roy and Cookiecutter community

Jul 06, 2023

CONTENTS

Cookiecutter creates projects from **cookiecutters** (project templates), e.g. Python package projects from Python package templates.

CHAPTER

ONE

BASICS

1.1 cookiecutter

A command-line utility that creates projects from **cookiecutters** (project templates), e.g. creating a Python package project from a Python package project template.

- Documentation: https://cookiecutter.readthedocs.io
- GitHub: https://github.com/cookiecutter/cookiecutter
- PyPI: https://pypi.org/project/cookiecutter/
- Free and open source software: BSD license

1.1.1 Features

- Cross-platform: Windows, Mac, and Linux are officially supported.
- You don't have to know/write Python code to use Cookiecutter.
- Works with Python 3.7, 3.8, 3.9, 3.10, 3.11
- Project templates can be in any programming language or markup format: Python, JavaScript, Ruby, CoffeeScript, RST, Markdown, CSS, HTML, you name it. You can use multiple languages in the same project template.

For users of existing templates

- Simple command line usage:

```
# Create project from the cookiecutter-pypackage.git repo template
# You'll be prompted to enter values.
# Then it'll create your Python package in the current working directory,
# based on those values.
$ cookiecutter https://github.com/audreyfeldroy/cookiecutter-pypackage
# For the sake of brevity, repos on GitHub can just use the 'gh' prefix
$ cookiecutter gh:audreyfeldroy/cookiecutter-pypackage
```

- Use it at the command line with a local template:

```
# Create project in the current working directory, from the local
# cookiecutter-pypackage/ template
$ cookiecutter cookiecutter-pypackage/
```

- Or use it from Python:

```
from cookiecutter.main import cookiecutter

# Create project from the cookiecutter-pypackage/ template
cookiecutter('cookiecutter-pypackage/')

# Create project from the cookiecutter-pypackage.git repo template
cookiecutter('https://github.com/audreyfeldroy/cookiecutter-pypackage.git')
```

- Unless you suppress it with `--no-input`, you are prompted for input:
  - Prompts are the keys in `cookiecutter.json`.
  - Default responses are the values in `cookiecutter.json`.
  - Prompts are shown in order.
- Cross-platform support for `~/.cookiecutterrc` files:

```
default_context:
  full_name: "Audrey Roy Greenfeld"
  email: "audreyr@gmail.com"
  github_username: "audreyfeldroy"
cookiecutters_dir: "~/.cookiecutters/"
```

- Cookiecutters (cloned Cookiecutter project templates) are put into `~/.cookiecutters/` by default, or cookiecutters_dir if specified.
- If you have already cloned a cookiecutter into `~/.cookiecutters/`, you can reference it by directory name:

```
# Clone cookiecutter-pypackage
$ cookiecutter gh:audreyfeldroy/cookiecutter-pypackage
# Now you can use the already cloned cookiecutter by name
$ cookiecutter cookiecutter-pypackage
```

- You can use local cookiecutters, or remote cookiecutters directly from Git repos or Mercurial repos on Bitbucket.
- Default context: specify key/value pairs that you want to be used as defaults whenever you generate a project.
- Inject extra context with command-line arguments:

```
cookiecutter --no-input gh:msabramo/cookiecutter-supervisor program_name=foobar␣
↪startsecs=10
```

- Direct access to the Cookiecutter API allows for the injection of extra context.
- Paths to local projects can be specified as absolute or relative. Projects are generated to your current directory or to the target directory if specified with `-o` option.

For template creators

- Supports unlimited levels of directory nesting.
- 100% of templating is done with Jinja2.
- Both, directory names and filenames can be templated. For example:

```
{{cookiecutter.repo_name}}/{{cookiecutter.repo_name}}/{{cookiecutter.repo_name}}.py
```

- Simply define your template variables in a `cookiecutter.json` file. You can also add human-readable questions that will be prompted to the user for each variable using the `__prompts__` key. For example:

```
{
  "full_name": "Audrey Roy Greenfeld",
  "email": "audreyr@gmail.com",
  "project_name": "Complexity",
  "repo_name": "complexity",
  "project_short_description": "Refreshingly simple static site generator.",
  "release_date": "2013-07-10",
  "year": "2013",
  "version": "0.1.1",
  "__prompts__": {
    "full_name": "Provide your full name",
    "email": "Provide your email"
  }
}
```

- Pre- and post-generate hooks: Python or shell scripts to run before or after generating a project.

1.1.2 Available Cookiecutters

Making great cookies takes a lot of cookiecutters and contributors. We're so pleased that there are many Cookiecutter project templates to choose from. We hope you find a cookiecutter that is just right for your needs.

A Pantry Full of Cookiecutters

The best place to start searching for specific and ready-to-use cookiecutter templates is Github search. Just type `cookiecutter` and you will discover over 4000 related repositories.

We also recommend you check related GitHub topics. For general search use cookiecutter-template. For specific topics try to use `cookiecutter-yourtopic`, like `cookiecutter-python` or `cookiecutter-datascience`. This is a new GitHub feature, so not all active repositories use it at the moment.

If you are a template developer please add related topics with `cookiecutter` prefix to your repository. We believe it will make it more discoverable. You are almost not limited in topic amount, use it!

Cookiecutter Specials

These Cookiecutters are maintained by the cookiecutter team:

- cookiecutter-pypackage: ultimate Python package project template by @audreyfeldroy's.
- cookiecutter-django: a framework for jumpstarting production-ready Django projects quickly. It is bleeding edge with Bootstrap 5, a customizable users app, starter templates, working user registration, celery setup, and much more.
- cookiecutter-pytest-plugin: Minimal Cookiecutter template for authoring pytest plugins that help you to write better programs.

1.1.3 Community

The core committer team can be found in the *authors' section*. We are always welcome and invite you to participate.

Stuck? Try one of the following:

- See the Troubleshooting page.
- Ask for help on Stack Overflow.
- You are strongly encouraged to file an issue about the problem. Do it even if it's just "I can't get it to work on this cookiecutter" with a link to your cookiecutter. Don't worry about naming/pinpointing the issue properly.
- Ask for help on Discord if you must (but please try one of the other options first, so that others can benefit from the discussion).

Development on Cookiecutter is community-driven:

- Huge thanks to all the *contributors* who have pitched in to help make Cookiecutter an even better tool.
- Everyone is invited to contribute. Read the *contributing instructions*, then get started.
- Connect with other Cookiecutter contributors and users on Discord (note: due to work and other commitments, a core committer might not always be available)

Encouragement is unbelievably motivating. If you want more work done on Cookiecutter, show support:

- Thank a core committer for their efforts.
- Star Cookiecutter on GitHub.
- *Support this project*

Got criticism or complaints?

- File an issue so that Cookiecutter can be improved. Be friendly and constructive about what could be better. Make detailed suggestions.
- **Keep us in the loop so that we can help.** For example, if you are discussing problems with Cookiecutter on a mailing list, file an issue where you link to the discussion thread and/or cc at least 1 core committer on the email.
- Be encouraging. A comment like "This function ought to be rewritten like this" is much more likely to result in action than a comment like "Eww, look how bad this function is."

Waiting for a response to an issue/question?

- Be patient and persistent. All issues are on the core committer team's radar and will be considered thoughtfully, but we have a lot of issues to work through. If urgent, it's fine to ping a core committer in the issue with a reminder.
- Ask others to comment, discuss, review, etc.

- Search the Cookiecutter repo for issues related to yours.
- Need a fix/feature/release/help urgently, and can't wait? @audreyfeldroy is available for hire for consultation or custom development.

1.1.4 Support This Project

This project is run by volunteers. Shortly we will be providing means for organizations and individuals to support the project.

1.1.5 Code of Conduct

Everyone interacting in the Cookiecutter project's codebases and documentation is expected to follow the PyPA Code of Conduct. This includes but is not limited to, issue trackers, chat rooms, mailing lists, and other virtual or in real-life communication.

1.1.6 Creator / Leader

This project was created and led by Audrey Roy Greenfeld.

She is supported by a team of maintainers.

1.2 Overview

Cookiecutter takes a template provided as a directory structure with template-files. Templates can be located in the filesystem, as a ZIP-file or on a VCS-Server (Git/Hg) like GitHub.

It reads a settings file and prompts the user interactively whether or not to change the settings.

Then it takes both and generates an output directory structure from it.

Additionally the template can provide code (Python or shell-script) to be executed before and after generation (pre-gen- and post-gen-hooks).

1.2.1 Input

This is a directory structure for a simple cookiecutter:

```
cookiecutter-something/
├── {{ cookiecutter.project_name }}/  <--------- Project template
│   └── ...
├── blah.txt                      <--------- Non-templated files/dirs
│                                            go outside
│
└── cookiecutter.json             <--------- Prompts & default values
```

You must have:

- A `cookiecutter.json` file.
- A `{{ cookiecutter.project_name }}/` directory, where `project_name` is defined in your `cookiecutter.json`.

Beyond that, you can have whatever files/directories you want.

See https://github.com/audreyfeldroy/cookiecutter-pypackage for a real-world example of this.

1.2.2 Output

This is what will be generated locally, in your current directory:

```
mysomething/  <---------- Value corresponding to what you enter at the
│                         project_name prompt
│
└── ...       <-------- Files corresponding to those in your
                        cookiecutter's `{{ cookiecutter.project_name }}/` dir
```

1.3 Installation

1.3.1 Prerequisites

- Python interpreter
- Adjust your path
- Packaging tools

Python interpreter

Install Python for your operating system. On Windows and macOS this is usually necessary. Most Linux distributions come with Python pre-installed. Consult the official Python documentation for details.

You can install the Python binaries from python.org. Alternatively on macOS, you can use the homebrew package manager.

```
brew install python3
```

Adjust your path

Ensure that your `bin` folder is on your path for your platform. Typically `~/.local/` for UNIX and macOS, or `%APPDATA%\Python` on Windows. (See the Python documentation for site.USER_BASE for full details.)

UNIX and macOS

For bash shells, add the following to your `.bash_profile` (adjust for other shells):

```
# Add ~/.local/ to PATH
export PATH=$HOME/.local/bin:$PATH
```

Remember to load changes with `source ~/.bash_profile` or open a new shell session.

Windows

Ensure the directory where cookiecutter will be installed is in your environment's `Path` in order to make it possible to invoke it from a command prompt. To do so, search for "Environment Variables" on your computer (on Windows 10, it is under `System Properties` –> `Advanced`) and add that directory to the `Path` environment variable, using the GUI to edit path segments.

Example segments should look like `%APPDATA%\Python\Python3x\Scripts`, where you have your version of Python instead of `Python3x`.

You may need to restart your command prompt session to load the environment variables.

See also:

See Configuring Python (on Windows) for full details.

Unix on Windows

You may also install Windows Subsystem for Linux or GNU utilities for Win32 to use Unix commands on Windows.

Packaging tools

See the Python Packaging Authority's (PyPA) documentation Requirements for Installing Packages for full details.

1.3.2 Install cookiecutter

At the command line:

```
python3 -m pip install --user cookiecutter
```

Or, if you do not have pip:

```
easy_install --user cookiecutter
```

Though, pip is recommended, easy_install is deprecated.

Or, if you are using conda, first add conda-forge to your channels:

```
conda config --add channels conda-forge
```

Once the conda-forge channel has been enabled, cookiecutter can be installed with:

```
conda install cookiecutter
```

1.3.3 Alternate installations

Homebrew (Mac OS X only):

```
brew install cookiecutter
```

Pipx (Linux, OSX and Windows):

```
pipx install cookiecutter
```

1.3.4 Upgrading

from 0.6.4 to 0.7.0 or greater

First, read *History* in detail. There are a lot of major changes. The big ones are:

- Cookiecutter no longer deletes the cloned repo after generating a project.
- Cloned repos are saved into *~/.cookiecutters/*.
- You can optionally create a *~/.cookiecutterrc* config file.

Or with pip:

```
python3 -m pip install --upgrade cookiecutter
```

Upgrade Cookiecutter either with easy_install (deprecated):

```
easy_install --upgrade cookiecutter
```

Then you should be good to go.

1.4 Usage

1.4.1 Grab a Cookiecutter template

First, clone a Cookiecutter project template:

```
$ git clone https://github.com/audreyfeldroy/cookiecutter-pypackage.git
```

1.4.2 Make your changes

Modify the variables defined in *cookiecutter.json*.

Open up the skeleton project. If you need to change it around a bit, do so.

You probably also want to create a repo, name it differently, and push it as your own new Cookiecutter project template, for handy future use.

1.4.3 Generate your project

Then generate your project from the project template:

```
$ cookiecutter cookiecutter-pypackage/
```

The only argument is the input directory. (The output directory is generated by rendering that, and it can't be the same as the input directory.)

Note: see *Command Line Options* for extra command line arguments

Try it out!

1.4.4 Works directly with git and hg (mercurial) repos too

To create a project from the cookiecutter-pypackage.git repo template:

```
$ cookiecutter gh:audreyfeldroy/cookiecutter-pypackage
```

Cookiecutter knows abbreviations for Github (`gh`), Bitbucket (`bb`), and GitLab (`gl`) projects, but you can also give it the full URL to any repository:

```
$ cookiecutter https://github.com/audreyfeldroy/cookiecutter-pypackage.git
$ cookiecutter git+ssh://git@github.com/audreyfeldroy/cookiecutter-pypackage.git
$ cookiecutter hg+ssh://hg@bitbucket.org/audreyr/cookiecutter-pypackage
```

You will be prompted to enter a bunch of project config values. (These are defined in the project's *cookiecutter.json*.)

Then, Cookiecutter will generate a project from the template, using the values that you entered. It will be placed in your current directory.

And if you want to specify a branch you can do that with:

```
$ cookiecutter https://github.com/audreyfeldroy/cookiecutter-pypackage.git --checkout␣
↪develop
```

1.4.5 Works with private repos

If you want to work with repos that are not hosted in github or bitbucket you can indicate explicitly the type of repo that you want to use prepending *hg+* or *git+* to repo url:

```
$ cookiecutter hg+https://example.com/repo
```

In addition, one can provide a path to the cookiecutter stored on a local server:

```
$ cookiecutter file://server/folder/project.git
```

1.4.6 Works with Zip files

You can also distribute cookiecutter templates as Zip files. To use a Zip file template, point cookiecutter at a Zip file on your local machine:

```
$ cookiecutter /path/to/template.zip
```

Or, if the Zip file is online:

```
$ cookiecutter https://example.com/path/to/template.zip
```

If the template has already been downloaded, or a template with the same name has already been downloaded, you will be prompted to delete the existing template before proceeding.

The Zip file contents should be the same as a git/hg repository for a template - that is, the zipfile should unpack into a top level directory that contains the name of the template. The name of the zipfile doesn't have to match the name of the template - for example, you can label a zipfile with a version number, but omit the version number from the directory inside the Zip file.

If you want to see an example Zipfile, find any Cookiecutter repository on Github and download that repository as a zip file - Github repository downloads are in a valid format for Cookiecutter.

Password-protected Zip files

If your repository Zip file is password protected, Cookiecutter will prompt you for that password whenever the template is used.

Alternatively, if you want to use a password-protected Zip file in an automated environment, you can export the *COOKIECUTTER_REPO_PASSWORD* environment variable; the value of that environment variable will be used whenever a password is required.

1.4.7 Keeping your cookiecutters organized

As of the Cookiecutter 0.7.0 release:

- Whenever you generate a project with a cookiecutter, the resulting project is output to your current directory.
- Your cloned cookiecutters are stored by default in your *~/.cookiecutters/* directory (or Windows equivalent). The location is configurable: see *User Config* for details.

Pre-0.7.0, this is how it worked:

- Whenever you generate a project with a cookiecutter, the resulting project is output to your current directory.
- Cloned cookiecutters were not saved locally.

1.5 Command Line Options

1.5.1 cookiecutter

Create a project from a Cookiecutter project template (TEMPLATE).

Cookiecutter is free and open source software, developed and managed by volunteers. If you would like to help out or fund the project, please get in touch at https://github.com/cookiecutter/cookiecutter.

```
cookiecutter [OPTIONS] [TEMPLATE] [EXTRA_CONTEXT]...
```

Options

-V, --version

Show the version and exit.

--no-input

Do not prompt for parameters and only use cookiecutter.json file content. Defaults to deleting any cached resources and redownloading them. Cannot be combined with the –replay flag.

-c, --checkout <checkout>

branch, tag or commit to checkout after git clone

--directory <directory>

Directory within repo that holds cookiecutter.json file for advanced repositories with multi templates in it

-v, --verbose

Print debug information

--replay

Do not prompt for parameters and only use information entered previously. Cannot be combined with the –no-input flag or with extra configuration passed.

--replay-file <replay_file>

Use this file for replay instead of the default.

-f, --overwrite-if-exists

Overwrite the contents of the output directory if it already exists

-s, --skip-if-file-exists

Skip the files in the corresponding directories if they already exist

-o, --output-dir <output_dir>

Where to output the generated project dir into

--config-file <config_file>

User configuration file

--default-config

Do not load a config file. Use the defaults instead

--debug-file <debug_file>

File to be used as a stream for DEBUG logging

--accept-hooks <accept_hooks>

Accept pre/post hooks

Options

yes | ask | no

-l, --list-installed

List currently installed templates.

--keep-project-on-failure

Do not delete project folder on failure

Arguments

TEMPLATE

Optional argument

EXTRA_CONTEXT

Optional argument(s)

1.6 Tutorials

Tutorials by @audreyfeldroy

1.6.1 Getting to Know Cookiecutter

Note: Before you begin, please install Cookiecutter 0.7.0 or higher. Instructions are in *Installation*.

Cookiecutter is a tool for creating projects from *cookiecutters* (project templates).

What exactly does this mean? Read on!

Case Study: cookiecutter-pypackage

cookiecutter-pypackage is a cookiecutter template that creates the starter boilerplate for a Python package.

Note: There are several variations of it, but for this tutorial we'll use the original version at https://github.com/audreyfeldroy/cookiecutter-pypackage/.

Step 1: Generate a Python Package Project

Open your shell and cd into the directory where you'd like to create a starter Python package project.

At the command line, run the cookiecutter command, passing in the link to cookiecutter-pypackage's HTTPS clone URL like this:

```
$ cookiecutter https://github.com/audreyfeldroy/cookiecutter-pypackage.git
```

Local Cloning of Project Template

First, cookiecutter-pypackage gets cloned to *~/.cookiecutters/* (or equivalent on Windows). Cookiecutter does this for you, so sit back and wait.

Local Generation of Project

When cloning is complete, you will be prompted to enter a bunch of values, such as *full_name*, *email*, and *project_name*. Either enter your info, or simply press return/enter to accept the default values.

This info will be used to fill in the blanks for your project. For example, your name and the year will be placed into the LICENSE file.

Step 2: Explore What Got Generated

In your current directory, you should see that a project got generated:

```
$ ls
boilerplate
```

Looking inside the *boilerplate/* (or directory corresponding to your *project_slug*) directory, you should see something like this:

```
$ ls boilerplate/
AUTHORS.rst      MANIFEST.in      docs             tox.ini
CONTRIBUTING.rst Makefile         requirements.txt
HISTORY.rst      README.rst       setup.py
LICENSE          boilerplate      tests
```

That's your new project!

If you open the AUTHORS.rst file, you should see something like this:

```
=======
Credits
=======

Development Lead
----------------

* Audrey Roy <audreyr@gmail.com>

Contributors
------------

None yet. Why not be the first?
```

Notice how it was auto-populated with your (or my) name and email.

Also take note of the fact that you are looking at a ReStructuredText file. Cookiecutter can generate a project with text files of any type.

Great, you just generated a skeleton Python package. How did that work?

Step 3: Observe How It Was Generated

Let's take a look at cookiecutter-pypackage together. Open https://github.com/audreyfeldroy/cookiecutter-pypackage in a new browser window.

{{ cookiecutter.project_slug }}

Find the directory called *{{ cookiecutter.project_slug }}*. Click on it. Observe the files inside of it. You should see that this directory and its contents corresponds to the project that you just generated.

This happens in *find.py*, where the *find_template()* method looks for the first jinja-like directory name that starts with *cookiecutter*.

AUTHORS.rst

Look at the raw version of *{{ cookiecutter.project_slug }}/AUTHORS.rst*, at https://raw.github.com/audreyfeldroy/cookiecutter-pypackage/master/%7B%7Bcookiecutter.project_slug%7D%7D/AUTHORS.rst.

Observe how it corresponds to the *AUTHORS.rst* file that you generated.

cookiecutter.json

Now navigate back up to *cookiecutter-pypackage/* and look at the *cookiecutter.json* file.

You should see JSON that corresponds to the prompts and default values shown earlier during project generation:

```
{
    "full_name": "Audrey Roy Greenfeld",
    "email": "aroy@alum.mit.edu",
    "github_username": "audreyr",
    "project_name": "Python Boilerplate",
    "project_slug": "{{ cookiecutter.project_name.lower().replace(' ', '_') }}",
    "project_short_description": "Python Boilerplate contains all the boilerplate you␣
↪need to create a Python package.",
    "pypi_username": "{{ cookiecutter.github_username }}",
    "version": "0.1.0",
    "use_pytest": "n",
    "use_pypi_deployment_with_travis": "y",
    "create_author_file": "y",
    "open_source_license": ["MIT", "BSD", "ISCL", "Apache Software License 2.0", "Not␣
↪open source"]
}
```

Questions?

If anything needs better explanation, please take a moment to file an issue at https://github.com/audreyfeldroy/cookiecutter/issues with what could be improved about this tutorial.

Summary

You have learned how to use Cookiecutter to generate your first project from a cookiecutter project template.

In tutorial 2 (*Create a Cookiecutter From Scratch*), you'll see how to create cookiecutters of your own, from scratch.

1.6.2 Create a Cookiecutter From Scratch

In this tutorial, we are creating *cookiecutter-website-simple*, a cookiecutter for generating simple, bare-bones websites.

Step 1: Name Your Cookiecutter

Create the directory for your cookiecutter and cd into it:

```
$ mkdir cookiecutter-website-simple
$ cd cookiecutter-website-simple/
```

Step 2: Create cookiecutter.json

cookiecutter.json is a JSON file that contains fields which can be referenced in the cookiecutter template. For each, default value is defined and user will be prompted for input during cookiecutter execution. Only mandatory field is *project_slug* and it should comply with package naming conventions defined in PEP8 Naming Conventions .

```
{
  "project_name": "Cookiecutter Website Simple",
  "project_slug": "{{ cookiecutter.project_name.lower().replace(' ', '_') }}",
  "author": "Anonymous"
}
```

Step 3: Create project_slug Directory

Create a directory called *{{ cookiecutter.project_slug }}*.

This value will be replaced with the repo name of projects that you generate from this cookiecutter.

Step 4: Create index.html

Inside of *{{ cookiecutter.project_slug }}*, create *index.html* with following content:

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>{{ cookiecutter.project_name }}</title>
    </head>

    <body>
        <h1>{{ cookiecutter.project_name }}</h1>
        <p>by {{ cookiecutter.author }}</p>
    </body>
</html>
```

Step 5: Pack cookiecutter into ZIP

There are many ways to run Cookiecutter templates, and they are described in details in Usage chapter. In this tutorial we are going to ZIP cookiecutter and then run it for testing.

By running following command *cookiecutter.zip* will get generated which can be used to run cookiecutter. Script will generate *cookiecutter.zip* ZIP file and echo full path to the file.

```
$ (SOURCE_DIR=$(basename $PWD) ZIP=cookiecutter.zip && # Set variables
pushd && # Set parent directory as working directory
zip -r $ZIP $SOURCE_DIR --exclude $SOURCE_DIR/$ZIP --quiet && # ZIP cookiecutter
mv $ZIP $SOURCE_DIR/$ZIP && # Move ZIP to original directory
popd && # Restore original work directory
echo  "Cookiecutter full path: $PWD/$ZIP")
```

Step 6: Run cookiecutter

Set your work directory to whatever directory you would like to run cookiecutter at. Use cookiecutter full path and run the following command:

```
$ cookiecutter <replace with Cookiecutter full path>
```

You can expect similar output:

```
$ cookiecutter /Users/admin/cookiecutter-website-simple/cookiecutter.zip
project_name [Cookiecutter Website Simple]: Test web
project_slug [test_web]:
author [Anonymous]: Cookiecutter Developer
```

Resulting directory should be inside your work directory with a name that matches *project_slug* you defined. Inside that directory there should be *index.html* with generated source:

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Test web</title>
    </head>

    <body>
        <h1>Cookiecutter Developer</h1>
        <p>by Test web</p>
    </body>
</html>
```

1.6.3 External Links

- Learn the Basics of Cookiecutter by Creating a Cookiecutter - first steps tutorial with example template by @BruceEckel
- Project Templates Made Easy by @pydanny
- Cookiedozer Tutorials by @hackebrot
  - Part 1: Create your own Cookiecutter template
  - Part 2: Extending our Cookiecutter template
  - Part 3: Wrapping up our Cookiecutter template

1.7 Advanced Usage

Various advanced topics regarding cookiecutter usage.

1.7.1 Using Pre/Post-Generate Hooks

New in cookiecutter 0.7

You can have Python or Shell scripts that run before and/or after your project is generated.

Put them in `hooks/` like this:

```
cookiecutter-something/
├── {{cookiecutter.project_slug}}/
├── hooks
│   ├── pre_gen_project.py
│   └── post_gen_project.py
└── cookiecutter.json
```

Shell scripts work similarly:

```
cookiecutter-something/
├── {{cookiecutter.project_slug}}/
├── hooks
│   ├── pre_gen_project.sh
│   └── post_gen_project.sh
└── cookiecutter.json
```

It shouldn't be too hard to extend Cookiecutter to work with other types of scripts too. Pull requests are welcome.

For portability, you should use Python scripts (with extension *.py*) for your hooks, as these can be run on any platform. However, if you intend for your template to only be run on a single platform, a shell script (or *.bat* file on Windows) can be a quicker alternative.

Writing hooks

Here are some details on how to write pre/post-generate hook scripts.

Exit with an appropriate status

Make sure your hook scripts work in a robust manner. If a hook script fails (that is, if it finishes with a nonzero exit status), the project generation will stop and the generated directory will be cleaned up.

Current working directory

When the hook scripts script are run, their current working directory is the root of the generated project. This makes it easy for a post-generate hook to find generated files using relative paths.

Template variables are rendered in the script

Just like your project template, Cookiecutter also renders Jinja template syntax in your scripts. This lets you incorporate Jinja template variables in your scripts. For example, this line of Python sets `module_name` to the value of the `cookiecutter.module_name` template variable:

```
module_name = '{{ cookiecutter.module_name }}'
```

Example: Validating template variables

Here is an example of a pre-generate hook script, defined at `hooks/pre_gen_project.py`, that validates a template variable before generating the project:

```
import re
import sys


MODULE_REGEX = r'^[_a-zA-Z][_a-zA-Z0-9]+$'

module_name = '{{ cookiecutter.module_name }}'

if not re.match(MODULE_REGEX, module_name):
    print('ERROR: %s is not a valid Python module name!' % module_name)

    # exits with status 1 to indicate failure
    sys.exit(1)
```

Example: Conditional files / directories

Here is an example of a post-generate hook script. The file `hooks/post_gen_project.py` shows how to achieve conditional control of files and directories after generating the project.

The script ensures that the directory structure is as expected by removing unwanted files and directories:

```
import os

REMOVE_PATHS = [
    '{% if cookiecutter.packaging != "pip" %} requirements.txt {% endif %}',
    '{% if cookiecutter.packaging != "poetry" %} poetry.lock {% endif %}',
]

for path in REMOVE_PATHS:
    path = path.strip()
    if path and os.path.exists(path):
        if os.path.isdir(path):
            os.rmdir(path)
        else:
            os.unlink(path)
```

1.7.2 User Config

New in Cookiecutter 0.7

If you use Cookiecutter a lot, you'll find it useful to have a user config file. By default Cookiecutter tries to retrieve settings from a *.cookiecutterrc* file in your home directory.

New in Cookiecutter 1.3

You can also specify a config file on the command line via `--config-file`.

```
cookiecutter --config-file /home/audreyr/my-custom-config.yaml cookiecutter-pypackage
```

Or you can set the `COOKIECUTTER_CONFIG` environment variable:

```
export COOKIECUTTER_CONFIG=/home/audreyr/my-custom-config.yaml
```

If you wish to stick to the built-in config and not load any user config file at all, use the CLI option `--default-config` instead. Preventing Cookiecutter from loading user settings is crucial for writing integration tests in an isolated environment.

Example user config:

```
default_context:
    full_name: "Audrey Roy"
    email: "audreyr@example.com"
    github_username: "audreyr"
cookiecutters_dir: "/home/audreyr/my-custom-cookiecutters-dir/"
replay_dir: "/home/audreyr/my-custom-replay-dir/"
abbreviations:
    pp: https://github.com/audreyfeldroy/cookiecutter-pypackage.git
    gh: https://github.com/{0}.git
    bb: https://bitbucket.org/{0}
```

Possible settings are:

default_context:
A list of key/value pairs that you want injected as context whenever you generate a project with Cookiecutter. These values are treated like the defaults in `cookiecutter.json`, upon generation of any project.

cookiecutters_dir
Directory where your cookiecutters are cloned to when you use Cookiecutter with a repo argument.

replay_dir
Directory where Cookiecutter dumps context data to, which you can fetch later on when using the *replay feature*.

abbreviations
A list of abbreviations for cookiecutters. Abbreviations can be simple aliases for a repo name, or can be used as a prefix, in the form `abbr:suffix`. Any suffix will be inserted into the expansion in place of the text `{0}`, using standard Python string formatting. With the above aliases, you could use the `cookiecutter-pypackage` template simply by saying `cookiecutter pp`, or `cookiecutter gh:audreyr/cookiecutter-pypackage`. The `gh` (GitHub), `bb` (Bitbucket), and `gl` (Gitlab) abbreviations shown above are actually **built in**, and can be used without defining them yourself.

Read also: *Injecting Extra Context*

1.7.3 Calling Cookiecutter Functions From Python

You can use Cookiecutter from Python:

```
from cookiecutter.main import cookiecutter

# Create project from the cookiecutter-pypackage/ template
cookiecutter('cookiecutter-pypackage/')

# Create project from the cookiecutter-pypackage.git repo template
cookiecutter('https://github.com/audreyfeldroy/cookiecutter-pypackage.git')
```

This is useful if, for example, you're writing a web framework and need to provide developers with a tool similar to *django-admin.py startproject* or *npm init*.

See the *API Reference* for more details.

1.7.4 Injecting Extra Context

You can specify an `extra_context` dictionary that will override values from `cookiecutter.json` or `.cookiecutterrc`:

```
cookiecutter(
    'cookiecutter-pypackage/',
    extra_context={'project_name': 'TheGreatest'},
)
```

This works as command-line parameters as well:

```
cookiecutter --no-input cookiecutter-pypackage/ project_name=TheGreatest
```

You will also need to add these keys to the `cookiecutter.json` or `.cookiecutterrc`.

Example: Injecting a Timestamp

If you have `cookiecutter.json` that has the following keys:

```
{
    "timestamp": "{{ cookiecutter.timestamp }}"
}
```

This Python script will dynamically inject a timestamp value as the project is generated:

```
from cookiecutter.main import cookiecutter

from datetime import datetime

cookiecutter(
    'cookiecutter-django',
    extra_context={'timestamp': datetime.utcnow().isoformat()}
)
```

How this works:

1. The script uses `datetime` to get the current UTC time in ISO format.
2. To generate the project, `cookiecutter()` is called, passing the timestamp in as context via the `extra_context` dict.

1.7.5 Suppressing Command-Line Prompts

To suppress the prompts asking for input, use `no_input`.

Note: this option will force a refresh of cached resources.

Basic Example: Using the Defaults

Cookiecutter will pick a default value if used with `no_input`:

```
from cookiecutter.main import cookiecutter
cookiecutter(
    'cookiecutter-django',
    no_input=True,
)
```

In this case it will be using the default defined in `cookiecutter.json` or `.cookiecutterrc`.

Note: values from `cookiecutter.json` will be overridden by values from `.cookiecutterrc`

Advanced Example: Defaults + Extra Context

If you combine an `extra_context` dict with the `no_input` argument, you can programmatically create the project with a set list of context parameters and without any command line prompts:

```
cookiecutter('cookiecutter-pypackage/',
             no_input=True,
             extra_context={'project_name': 'TheGreatest'})
```

See also *Injecting Extra Context* and the *API Reference* for more details.

1.7.6 Templates in Context Values

The values (but not the keys!) of *cookiecutter.json* are also Jinja2 templates. Values from user prompts are added to the context immediately, such that one context value can be derived from previous values. This approach can potentially save your user a lot of keystrokes by providing more sensible defaults.

Basic Example: Templates in Context

Python packages show some patterns for their naming conventions:

- a human-readable project name
- a lowercase, dashed repository name
- an importable, dash-less package name

Here is a *cookiecutter.json* with templated values for this pattern:

```
{
  "project_name": "My New Project",
  "project_slug": "{{ cookiecutter.project_name|lower|replace(' ', '-') }}",
  "pkg_name": "{{ cookiecutter.project_slug|replace('-', '') }}"
}
```

If the user takes the defaults, or uses *no_input*, the templated values will be:

- *my-new-project*
- *mynewproject*

Or, if the user gives *Yet Another New Project*, the values will be:

- `yet-another-new-project`
- `yetanothernewproject`

1.7.7 Private Variables

Cookiecutter allows the definition private variables by prepending an underscore to the variable name. The user will not be required to fill those variables in. These can either be not rendered, by using a prepending underscore, or rendered, prepending a double underscore. For example, the `cookiecutter.json`:

```
{
    "project_name": "Really cool project",
    "_not_rendered": "{{ cookiecutter.project_name|lower }}",
    "__rendered": "{{ cookiecutter.project_name|lower }}"
}
```

Will be rendered as:

```
{
    "project_name": "Really cool project",
    "_not_rendered": "{{ cookiecutter.project_name|lower }}",
    "__rendered": "really cool project"
}
```

The user will only be asked for `project_name`.

Non-rendered private variables can be used for defining constants. An example of where you may wish to use private **rendered** variables is creating a Python package repository and want to enforce naming consistency. To ensure the repository and package name are based on the project name, you could create a `cookiecutter.json` such as:

```
{
    "project_name": "Project Name",
    "__project_slug": "{{ cookiecutter.project_name|lower|replace(' ', '-') }}",
    "__package_name": "{{ cookiecutter.project_name|lower|replace(' ', '_') }}",
}
```

Which could create a structure like this:

```
project-name
├── Makefile
├── README.md
├── requirements.txt
└── src
    ├── project_name
    │   └── __init__.py
    ├── setup.py
    └── tests
        └── __init__.py
```

The `README.md` can then have a plain English project title.

1.7.8 Copy without Render

New in Cookiecutter 1.1

To avoid rendering directories and files of a cookiecutter, the `_copy_without_render` key can be used in the `cookiecutter.json`. The value of this key accepts a list of Unix shell-style wildcards:

```
{
    "project_slug": "sample",
    "_copy_without_render": [
        "*.html",
        "*not_rendered_dir",
        "rendered_dir/not_rendered_file.ini"
    ]
}
```

Note: Only the content of the files will be copied without being rendered. The paths are subject to rendering. This allows you to write:

```
{
    "project_slug": "sample",
    "_copy_without_render": [
        "{{cookiecutter.repo_name}}/templates/*.html",
    ]
}
```

In this example, `{{cookiecutter.repo_name}}` will be rendered as expected but the html file content will be copied without rendering.

1.7.9 Replay Project Generation

New in Cookiecutter 1.1

On invocation **Cookiecutter** dumps a json file to `~/.cookiecutter_replay/` which enables you to *replay* later on.

In other words, it persists your **input** for a template and fetches it when you run the same template again.

Example for a replay file (which was created via `cookiecutter gh:hackebrot/cookiedozer`):

```
{
    "cookiecutter": {
        "app_class_name": "FooBarApp",
        "app_title": "Foo Bar",
        "email": "raphael@example.com",
        "full_name": "Raphael Pierzina",
        "github_username": "hackebrot",
        "kivy_version": "1.8.0",
        "project_slug": "foobar",
        "short_description": "A sleek slideshow app that supports swipe gestures.",
        "version": "0.1.0",
        "year": "2015"
    }
}
```

To fetch this context data without being prompted on the command line you can use either of the following methods.

Pass the according option on the CLI:

```
cookiecutter --replay gh:hackebrot/cookiedozer
```

Or use the Python API:

```
from cookiecutter.main import cookiecutter
cookiecutter('gh:hackebrot/cookiedozer', replay=True)
```

This feature comes in handy if, for instance, you want to create a new project from an updated template.

Custom replay file

New in Cookiecutter 2.0

To specify a custom filename, you can use the `--replay-file` option:

```
cookiecutter --replay-file ./cookiedozer.json gh:hackebrot/cookiedozer
```

This may be useful to run the same replay file over several machines, in tests or when a user of the template reports a problem.

1.7.10 Choice Variables

New in Cookiecutter 1.1

Choice variables provide different choices when creating a project. Depending on a user's choice the template renders things differently.

Basic Usage

Choice variables are regular key / value pairs, but with the value being a list of strings.

For example, if you provide the following choice variable in your `cookiecutter.json`:

```
{
    "license": ["MIT", "BSD-3", "GNU GPL v3.0", "Apache Software License 2.0"]
}
```

you'd get the following choices when running Cookiecutter:

```
Select license:
1 - MIT
2 - BSD-3
3 - GNU GPL v3.0
4 - Apache Software License 2.0
Choose from 1, 2, 3, 4 [1]:
```

Depending on an user's choice, a different license is rendered by Cookiecutter.

The above `license` choice variable creates `cookiecutter.license`, which can be used like this:

```
{%- if cookiecutter.license == "MIT" -%}
# Possible license content here

{%- elif cookiecutter.license == "BSD-3" -%}
# More possible license content here

{% endif %}
```

Cookiecutter is using Jinja2's if conditional expression to determine the correct license.

The created choice variable is still a regular Cookiecutter variable and can be used like this:

```
License
-------

Distributed under the terms of the `{{cookiecutter.license}}`_ license,
```

Overwriting Default Choice Values

Choice Variables are overwritable using a *User Config* file.

For example, a choice variable can be created in `cookiecutter.json` by using a list as value:

```
{
    "license": ["MIT", "BSD-3", "GNU GPL v3.0", "Apache Software License 2.0"]
}
```

By default, the first entry in the values list serves as default value in the prompt.

Setting the default `license` agreement to *Apache Software License 2.0* can be done using:

```
default_context:
    license: "Apache Software License 2.0"
```

in the *User Config* file.

The resulting prompt changes and looks like:

```
Select license:
1 - Apache Software License 2.0
2 - MIT
3 - BSD-3
4 - GNU GPL v3.0
Choose from 1, 2, 3, 4 [1]:
```

Note: As you can see the order of the options changed from `1 - MIT` to `1 - Apache Software License 2.0`. **Cookiecutter** takes the first value in the list as the default.

1.7.11 Boolean Variables

New in version 2.2.0.

Boolean variables are used for answering True/False questions.

Basic Usage

Boolean variables are regular key / value pairs, but with the value being `True`/`False`.

For example, if you provide the following boolean variable in your `cookiecutter.json`:

```
{
    "run_as_docker": true
}
```

you will get the following user input when running Cookiecutter:

```
run_as_docker [True]:
```

User input will be parsed by *read_user_yes_no()*. The following values are considered as valid user input:

- `True` values: "1", "true", "t", "yes", "y", "on"
- `False` values: "0", "false", "f", "no", "n", "off"

The above `run_as_docker` boolean variable creates `cookiecutter.run_as_docker`, which can be used like this:

```
{%- if cookiecutter.run_as_docker -%}
# In case of True add your content here

{%- else -%}
# In case of False add your content here

{% endif %}
```

Cookiecutter is using Jinja2's if conditional expression to determine the correct `run_as_docker`.

Input Validation

If a non valid value is inserted to a boolean field, the following error will be printed:

```
run_as_docker [True]: docker
Error: docker is not a valid boolean
```

1.7.12 Dictionary Variables

New in Cookiecutter 1.5

Dictionary variables provide a way to define deep structured information when rendering a template.

Basic Usage

Dictionary variables are, as the name suggests, dictionaries of key-value pairs. The dictionary values can, themselves, be other dictionaries and lists - the data structure can be as deep as you need.

For example, you could provide the following dictionary variable in your `cookiecutter.json`:

```
{
    "project_slug": "new_project",
    "file_types": {
        "png": {
            "name": "Portable Network Graphic",
            "library": "libpng",
            "apps": [
                "GIMP"
            ]
        },
        "bmp": {
            "name": "Bitmap",
            "library": "libbmp",
            "apps": [
                "Paint",
                "GIMP"
            ]
        }
    }
}
```

The above `file_types` dictionary variable creates `cookiecutter.file_types`, which can be used like this:

```
{% for extension, details in cookiecutter.file_types|dictsort %}
<dl>
  <dt>Format name:</dt>
  <dd>{{ details.name }}</dd>

  <dt>Extension:</dt>
  <dd>{{ extension }}</dd>

  <dt>Applications:</dt>
  <dd>
      <ul>
      {% for app in details.apps -%}
          <li>{{ app }}</li>
      {% endfor -%}
      </ul>
  </dd>
</dl>
{% endfor %}
```

Cookiecutter is using Jinja2's for expression to iterate over the items in the dictionary.

1.7.13 Templates inheritance (2.2+)

New in Cookiecutter 2.2+

Sometimes you need to extend a base template with a different configuration to avoid nested blocks.

Cookiecutter introduces the ability to use common templates using the power of jinja: *extends*, *include* and *super*.

Here's an example repository:

```
https://github.com/user/repo-name.git
├── {{cookiecutter.project_slug}}/
|   └── file.txt
├── templates/
|   └── base.txt
└── cookiecutter.json
```

every file in the *templates* directory will become referable inside the project itself, and the path should be relative from the *templates* folder like

```
# file.txt
{% extends "base.txt" %}

... or ...

# file.txt
{% include "base.txt" %}
```

see more on https://jinja.palletsprojects.com/en/2.11.x/templates/

1.7.14 Template Extensions

New in Cookiecutter 1.4

A template may extend the Cookiecutter environment with custom Jinja2 extensions. It can add extra filters, tests, globals or even extend the parser.

To do so, a template author must specify the required extensions in `cookiecutter.json` as follows:

```
{
    "project_slug": "Foobar",
    "year": "{% now 'utc', '%Y' %}",
    "_extensions": ["jinja2_time.TimeExtension"]
}
```

On invocation Cookiecutter tries to import the extensions and add them to its environment respectively.

In the above example, Cookiecutter provides the additional tag now, after installing the jinja2_time.TimeExtension and enabling it in `cookiecutter.json`.

Please note that Cookiecutter will **not** install any dependencies on its own! As a user you need to make sure you have all the extensions installed, before running Cookiecutter on a template that requires custom Jinja2 extensions.

By default Cookiecutter includes the following extensions:

- `cookiecutter.extensions.JsonifyExtension`
- `cookiecutter.extensions.RandomStringExtension`

- `cookiecutter.extensions.SlugifyExtension`
- `cookiecutter.extensions.TimeExtension`
- `cookiecutter.extensions.UUIDExtension`

Jsonify extension

The `cookiecutter.extensions.JsonifyExtension` extension provides a `jsonify` filter in templates that converts a Python object to JSON:

```
{% {'a': True} | jsonify %}
```

Would output:

```
{"a": true}
```

Random string extension

New in Cookiecutter 1.7

The `cookiecutter.extensions.RandomStringExtension` extension provides a `random_ascii_string` method in templates that generates a random fixed-length string, optionally with punctuation.

Generate a random n-size character string. Example for n=12:

```
{{ random_ascii_string(12) }}
```

Outputs:

```
bIIUczoNvswh
```

The second argument controls if punctuation and special characters `!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~` should be present in the result:

```
{{ random_ascii_string(12, punctuation=True) }}
```

Outputs:

```
fQupUkY}W!)!
```

Slugify extension

The `cookiecutter.extensions.SlugifyExtension` extension provides a `slugify` filter in templates that converts string into its dashed ("slugified") version:

```
{% "It's a random version" | slugify %}
```

Would output:

```
it-s-a-random-version
```

It is different from a mere replace of spaces since it also treats some special characters differently such as ' in the example above. The function accepts all arguments that can be passed to the `slugify` function of python-slugify. For example to change the output from `it-s-a-random-version` to `it_s_a_random_version`, the `separator` parameter would be passed: `slugify(separator='_')`.

UUID4 extension

New in Cookiecutter 1.x

The `cookiecutter.extensions.UUIDExtension` extension provides a `uuid4()` method in templates that generates a uuid4.

Generate a uuid4 string:

```
{{ uuid4() }}
```

Outputs:

```
83b5de62-31b4-4a1e-83fa-8c548de65a11
```

1.7.15 Organizing cookiecutters in directories

New in Cookiecutter 1.7

Cookiecutter introduces the ability to organize several templates in one repository or zip file, separating them by directories. This allows using symlinks for general files. Here's an example repository demonstrating this feature:

```
https://github.com/user/repo-name.git
    ├── directory1-name/
    |   ├── {{cookiecutter.project_slug}}/
    |   └── cookiecutter.json
    └── directory2-name/
        ├── {{cookiecutter.project_slug}}/
        └── cookiecutter.json
```

To activate one of templates within a subdirectory, use the `--directory` option:

```
cookiecutter https://github.com/user/repo-name.git --directory="directory1-name"
```

1.7.16 Customizing the Jinja2 environment

The special template variable `_jinja2_env_vars` can be used to customize the [Jinja2 environment](https://jinja.palletsprojects.com/en/3.1.x/api/#jinja2.Environment).

This example shows how to control whitespace with `lstrip_blocks` and `trim_blocks`:

```
{
    "project_slug": "sample",
    "_jinja2_env_vars": {"lstrip_blocks": true, "trim_blocks": true}
}
```

1.7.17 Working with line-ends special symbols LF/CRLF

New in Cookiecutter 2.0

Note: Before version 2.0 Cookiecutter silently used system line end character. LF for POSIX and CRLF for Windows. Since version 2.0 this behaviour changed and now can be forced at template level.

By default Cookiecutter checks every file at render stage and uses the same line end as in source. This allow template developers to have both types of files in the same template. Developers should correctly configure their `.gitattributes` file to avoid line-end character overwrite by git.

The special template variable `_new_lines` enforces a specific line ending. Acceptable variables: `'\r\n'` for CRLF and `'\n'` for POSIX.

Here is example how to force line endings to CRLF on any deployment:

```
{
    "project_slug": "sample",
    "_new_lines": "\r\n"
}
```

1.7.18 Local Extensions

New in Cookiecutter 2.1

A template may extend the Cookiecutter environment with local extensions. These can be part of the template itself, providing it with more sophisticated custom tags and filters.

To do so, a template author must specify the required extensions in `cookiecutter.json` as follows:

```
{
    "project_slug": "Foobar",
    "year": "{% now 'utc', '%Y' %}",
    "_extensions": ["local_extensions.FoobarExtension"]
}
```

This example uses a simple module `local_extensions.py` which exists in the template root, containing the following (for instance):

```
from jinja2.ext import Extension


class FoobarExtension(Extension):
    def __init__(self, environment):
        super(FoobarExtension, self).__init__(environment)
        environment.filters['foobar'] = lambda v: v * 2
```

This will register the `foobar` filter for the template.

For many cases, this will be unnecessarily complicated. It's likely that we'd only want to register a single function as a filter. For this, we can use the `simple_filter` decorator:

```
{
    "project_slug": "Foobar",
```

(continues on next page)

(continued from previous page)

```
    "year": "{% now 'utc', '%Y' %}",
    "_extensions": ["local_extensions.simplefilterextension"]
}
```

```
from cookiecutter.utils import simple_filter


@simple_filter
def simplefilterextension(v):
    return v * 2
```

This snippet will achieve the exact same result as the previous one.

For complex use cases, a python module `local_extensions` (a folder with an `__init__.py`) can also be created in the template root. Here, for example, a module `main.py` would have to export all extensions with `from .main import FoobarExtension, simplefilterextension` or `from .main import *` in the `__init__.py`.

1.7.19 Nested configuration files

If you wish to create a hierarchy of templates and use cookiecutter to choose among them, you need just to specify the key `template` in the main configuration file to reach the other ones.

Let's imagine to have the following structure:

```
main-directory/
    ├── project-1
    │   ├── cookiecutter.json
    │   ├── {{cookiecutter.project_slug}}
    |       |   ├── ...
    ├── project-2
    │   ├── cookiecutter.json
    │   ├── {{cookiecutter.project_slug}}
    |       |   ├── ...
    └── cookiecutter.json
```

It is possible to specify in the main `cookiecutter.json` how to reach the other config files as follows:

```
{
    "template": [
        "Project 1 (./project-1)",
        "Project 2 (./project-2)"
    ]
}
```

Then, when `cookiecutter` is launched in the main directory it will ask to choice among the possible templates:

```
Select template:
1 - Project 1 (./project-1)
2 - Project 2 (./project-2)
Choose from 1, 2 [1]:
```

Once a template is chosen, for example 1, it will continue to ask the info required by `cookiecutter.json` in the `project-1` folder, such as `project-slug`

1.7.20 Human readable prompts

You can add human-readable prompts that will be shown to the user for each variable using the `__prompts__` key:

```
{
    "package_name": "my-package",
    "module_name": "{{ cookiecutter.package_name.replace('-', '_') }}",
    "package_name_stylized": "{{ cookiecutter.module_name.replace('_', ' ').capitalize()␣
↪}}",
    "short_description": "A nice python package",
    "github_username": "your-org-or-username",
    "full_name": "Firstname Lastname",
    "email": "email@example.com",
    "command_line_interface": ["yes", "no"],
    "init_git": ["yes", "no"],
    "enable_pre_commit": ["yes", "no"],
    "documentation_website": ["yes", "no"],
    "black_formatting": ["yes", "no"],
    "__prompts__": {
        "package_name": "Select your package name:",
        "module_name": "Select your module name:",
        "package_name_stylized": "Stylized package name:",
        "short_description": "Short description:",
        "github_username": "GitHub username or organization:",
        "full_name": "Author full name:",
        "email": "Author email:",
        "command_line_interface": "Add CLI:",
        "init_git": "Initialize a git repository:",
        "enable_pre_commit": "Enable pre-commit:",
        "documentation_website": "Add a documentation website:",
        "black_formatting": "Enable black formatting:"
    }
}
```

1.8 Troubleshooting

1.8.1 I created a cookiecutter, but it doesn't work, and I can't figure out why

- Try upgrading to Cookiecutter 0.8.0, which prints better error messages and has fixes for several common bugs.

1.8.2 I'm having trouble generating Jinja templates from Jinja templates

Make sure you escape things properly, like this:

```
{{ "{{" }}
```

Or this:

```
{% raw %}
<p>Go <a href="{{ url_for('home') }}">Home</a></p>
{% endraw %}
```

Or this:

```
{{ {{ url_for('home') }} }}
```

See https://jinja.palletsprojects.com/en/latest/templates/#escaping for more info.

You can also use the _copy_without_render key in your *cookiecutter.json* file to escape entire files and directories.

1.8.3 Other common issues

TODO: add a bunch of common new user issues here.

This document is incomplete. If you have knowledge that could help other users, adding a section or filing an issue with details would be greatly appreciated.

CHAPTER

TWO

API REFERENCE

2.1 API

This is the Cookiecutter modules API documentation.

2.1.1 cookiecutter.cli module

Main *cookiecutter* CLI.

`cookiecutter.cli.`**`list_installed_templates`**(*default_config*, *passed_config_file*)

List installed (locally cloned) templates. Use cookiecutter –list-installed.

`cookiecutter.cli.`**`validate_extra_context`**(*ctx*, *param*, *value*)

Validate extra context.

`cookiecutter.cli.`**`version_msg`**()

Return the Cookiecutter version, location and Python powering it.

2.1.2 cookiecutter.config module

Global configuration handling.

`cookiecutter.config.`**`get_config`**(*config_path*)

Retrieve the config from the specified path, returning a config dict.

`cookiecutter.config.`**`get_user_config`**(*config_file=None*, *default_config=False*)

Return the user config as a dict.

If `default_config` is True, ignore `config_file` and return default values for the config parameters.

If a path to a `config_file` is given, that is different from the default location, load the user config from that.

Otherwise look up the config file path in the `COOKIECUTTER_CONFIG` environment variable. If set, load the config from this path. This will raise an error if the specified path is not valid.

If the environment variable is not set, try the default config file path before falling back to the default config values.

`cookiecutter.config.`**`merge_configs`**(*default*, *overwrite*)

Recursively update a dict with the key/value pair of another.

Dict values that are dictionaries themselves will be updated, whilst preserving existing keys.

2.1.3 cookiecutter.environment module

Jinja2 environment and extensions loading.

class cookiecutter.environment.**ExtensionLoaderMixin**(***kwargs*)

Bases: object

Mixin providing sane loading of extensions specified in a given context.

The context is being extracted from the keyword arguments before calling the next parent class in line of the child.

class cookiecutter.environment.**StrictEnvironment**(***kwargs*)

Bases: *ExtensionLoaderMixin*, Environment

Create strict Jinja2 environment.

Jinja2 environment will raise error on undefined variable in template- rendering context.

2.1.4 cookiecutter.exceptions module

All exceptions used in the Cookiecutter code base are defined here.

exception cookiecutter.exceptions.**ConfigDoesNotExistException**

Bases: *CookiecutterException*

Exception for missing config file.

Raised when get_config() is passed a path to a config file, but no file is found at that path.

exception cookiecutter.exceptions.**ContextDecodingException**

Bases: *CookiecutterException*

Exception for failed JSON decoding.

Raised when a project's JSON context file can not be decoded.

exception cookiecutter.exceptions.**CookiecutterException**

Bases: Exception

Base exception class.

All Cookiecutter-specific exceptions should subclass this class.

exception cookiecutter.exceptions.**FailedHookException**

Bases: *CookiecutterException*

Exception for hook failures.

Raised when a hook script fails.

exception cookiecutter.exceptions.**InvalidConfiguration**

Bases: *CookiecutterException*

Exception for invalid configuration file.

Raised if the global configuration file is not valid YAML or is badly constructed.

exception cookiecutter.exceptions.**InvalidModeException**

Bases: *CookiecutterException*

Exception for incompatible modes.

Raised when cookiecutter is called with both *no_input==True* and *replay==True* at the same time.

exception cookiecutter.exceptions.**InvalidZipRepository**

Bases: *CookiecutterException*

Exception for bad zip repo.

Raised when the specified cookiecutter repository isn't a valid Zip archive.

exception cookiecutter.exceptions.**MissingProjectDir**

Bases: *CookiecutterException*

Exception for missing generated project directory.

Raised during cleanup when remove_repo() can't find a generated project directory inside of a repo.

exception cookiecutter.exceptions.**NonTemplatedInputDirException**

Bases: *CookiecutterException*

Exception for when a project's input dir is not templated.

The name of the input directory should always contain a string that is rendered to something else, so that input_dir != output_dir.

exception cookiecutter.exceptions.**OutputDirExistsException**

Bases: *CookiecutterException*

Exception for existing output directory.

Raised when the output directory of the project exists already.

exception cookiecutter.exceptions.**RepositoryCloneFailed**

Bases: *CookiecutterException*

Exception for un-cloneable repo.

Raised when a cookiecutter template can't be cloned.

exception cookiecutter.exceptions.**RepositoryNotFound**

Bases: *CookiecutterException*

Exception for missing repo.

Raised when the specified cookiecutter repository doesn't exist.

exception cookiecutter.exceptions.**UndefinedVariableInTemplate**(*message*, *error*, *context*)

Bases: *CookiecutterException*

Exception for out-of-scope variables.

Raised when a template uses a variable which is not defined in the context.

exception cookiecutter.exceptions.**UnknownExtension**

Bases: *CookiecutterException*

Exception for un-importable extension.

Raised when an environment is unable to import a required extension.

exception cookiecutter.exceptions.**UnknownRepoType**

Bases: *CookiecutterException*

Exception for unknown repo types.

Raised if a repo's type cannot be determined.

exception cookiecutter.exceptions.**UnknownTemplateDirException**

Bases: *CookiecutterException*

Exception for ambiguous project template directory.

Raised when Cookiecutter cannot determine which directory is the project template, e.g. more than one dir appears to be a template dir.

exception cookiecutter.exceptions.**VCSNotInstalled**

Bases: *CookiecutterException*

Exception when version control is unavailable.

Raised if the version control system (git or hg) is not installed.

2.1.5 cookiecutter.extensions module

Jinja2 extensions.

class cookiecutter.extensions.**JsonifyExtension**(*environment*)

Bases: Extension

Jinja2 extension to convert a Python object to JSON.

identifier = 'cookiecutter.extensions.JsonifyExtension'

class cookiecutter.extensions.**RandomStringExtension**(*environment*)

Bases: Extension

Jinja2 extension to create a random string.

identifier = 'cookiecutter.extensions.RandomStringExtension'

class cookiecutter.extensions.**SlugifyExtension**(*environment*)

Bases: Extension

Jinja2 Extension to slugify string.

identifier = 'cookiecutter.extensions.SlugifyExtension'

class cookiecutter.extensions.**TimeExtension**(*environment*)

Bases: Extension

Jinja2 Extension for dates and times.

parse(*parser*)

Parse datetime template and add datetime value.

identifier = 'cookiecutter.extensions.TimeExtension'

tags = {'now'}

if this extension parses this is the list of tags it's listening to.

class cookiecutter.extensions.**UUIDExtension**(*environment*)

Bases: Extension

Jinja2 Extension to generate uuid4 string.

identifier = 'cookiecutter.extensions.UUIDExtension'

2.1.6 cookiecutter.find module

Functions for finding Cookiecutter templates and other components.

cookiecutter.find.**find_template**(*repo_dir*)

Determine which child directory of repo_dir is the project template.

Parameters
repo_dir (PathLike[str]) – Local directory of newly cloned repo.

Return type
Path

Returns
Relative path to project template.

2.1.7 cookiecutter.generate module

Functions for generating a project from a project template.

cookiecutter.generate.**apply_overwrites_to_context**(*context*, *overwrite_context*)

Modify the given context in place based on the overwrite_context.

cookiecutter.generate.**ensure_dir_is_templated**(*dirname*)

Ensure that dirname is a templated directory name.

cookiecutter.generate.**generate_context**(*context_file='cookiecutter.json'*, *default_context=None*, *extra_context=None*)

Generate the context for a Cookiecutter project template.

Loads the JSON file as a Python object, with key being the JSON filename.

Parameters

- **context_file** – JSON file containing key/value pairs for populating the cookiecutter's variables.
- **default_context** – Dictionary containing config to take into account.
- **extra_context** – Dictionary containing configuration overrides

cookiecutter.generate.**generate_file**(*project_dir*, *infile*, *context*, *env*, *skip_if_file_exists=False*)

Render filename of infile as name of outfile, handle infile correctly.

Dealing with infile appropriately:

a. If infile is a binary file, copy it over without rendering.

b. If infile is a text file, render its contents and write the rendered infile to outfile.

Precondition:

When calling *generate_file()*, the root template dir must be the current working directory. Using *utils.work_in()* is the recommended way to perform this directory change.

Parameters

- **project_dir** – Absolute path to the resulting generated project.
- **infile** – Input file to generate the file from. Relative to the root template dir.
- **context** – Dict for populating the cookiecutter's variables.
- **env** – Jinja2 template execution environment.

cookiecutter.generate.**generate_files**(*repo_dir*, *context=None*, *output_dir='.'*, *overwrite_if_exists=False*, *skip_if_file_exists=False*, *accept_hooks=True*, *keep_project_on_failure=False*)

Render the templates and saves them to files.

Parameters

- **repo_dir** – Project template input directory.
- **context** – Dict for populating the template's variables.
- **output_dir** – Where to output the generated project dir into.
- **overwrite_if_exists** – Overwrite the contents of the output directory if it exists.
- **skip_if_file_exists** – Skip the files in the corresponding directories if they already exist
- **accept_hooks** – Accept pre and post hooks if set to *True*.
- **keep_project_on_failure** – If *True* keep generated project directory even when generation fails

cookiecutter.generate.**is_copy_only_path**(*path*, *context*)

Check whether the given *path* should only be copied and not rendered.

Returns True if *path* matches a pattern in the given *context* dict, otherwise False.

Parameters

- **path** – A file-system path referring to a file or dir that should be rendered or just copied.
- **context** – cookiecutter context.

cookiecutter.generate.**render_and_create_dir**(*dirname*, *context*, *output_dir*, *environment*, *overwrite_if_exists=False*)

Render name of a directory, create the directory, return its path.

2.1.8 cookiecutter.hooks module

Functions for discovering and executing various cookiecutter hooks.

cookiecutter.hooks.**find_hook**(*hook_name*, *hooks_dir='hooks'*)

Return a dict of all hook scripts provided.

Must be called with the project template as the current working directory. Dict's key will be the hook/script's name, without extension, while values will be the absolute path to the script. Missing scripts will not be included in the returned dict.

Parameters

- **hook_name** – The hook to find
- **hooks_dir** – The hook directory in the template

Returns
The absolute path to the hook script or None

cookiecutter.hooks.**run_hook**(*hook_name*, *project_dir*, *context*)

Try to find and execute a hook from the specified project directory.

Parameters

- **hook_name** – The hook to execute.
- **project_dir** – The directory to execute the script from.
- **context** – Cookiecutter project context.

cookiecutter.hooks.**run_script**(*script_path*, *cwd='.'*)

Execute a script from a working directory.

Parameters

- **script_path** – Absolute path to the script to run.
- **cwd** – The directory to run the script from.

cookiecutter.hooks.**run_script_with_context**(*script_path*, *cwd*, *context*)

Execute a script after rendering it with Jinja.

Parameters

- **script_path** – Absolute path to the script to run.
- **cwd** – The directory to run the script from.
- **context** – Cookiecutter project template context.

cookiecutter.hooks.**valid_hook**(*hook_file*, *hook_name*)

Determine if a hook file is valid.

Parameters

- **hook_file** – The hook file to consider for validity
- **hook_name** – The hook to find

Returns
The hook file validity

2.1.9 cookiecutter.log module

Module for setting up logging.

cookiecutter.log.**configure_logger**(*stream_level='DEBUG'*, *debug_file=None*)

Configure logging for cookiecutter.

Set up logging to stdout with given level. If `debug_file` is given set up logging to file with DEBUG level.

2.1.10 cookiecutter.main module

Main entry point for the *cookiecutter* command.

The code in this module is also a good example of how to use Cookiecutter as a library rather than a script.

cookiecutter.main.**cookiecutter**(*template*, *checkout=None*, *no_input=False*, *extra_context=None*, *replay=None*, *overwrite_if_exists=False*, *output_dir='.'*, *config_file=None*, *default_config=False*, *password=None*, *directory=None*, *skip_if_file_exists=False*, *accept_hooks=True*, *keep_project_on_failure=False*)

Run Cookiecutter just as if using it from the command line.

Parameters

- **template** – A directory containing a project template directory, or a URL to a git repository.
- **checkout** – The branch, tag or commit ID to checkout after clone.
- **no_input** – Do not prompt for user input. Use default values for template parameters taken from *cookiecutter.json*, user config and *extra_dict*. Force a refresh of cached resources.
- **extra_context** – A dictionary of context that overrides default and user configuration.
- **replay** – Do not prompt for input, instead read from saved json. If `True` read from the `replay_dir`. if it exists
- **output_dir** – Where to output the generated project dir into.
- **config_file** – User configuration file path.
- **default_config** – Use default values rather than a config file.
- **password** – The password to use when extracting the repository.
- **directory** – Relative path to a cookiecutter template in a repository.
- **accept_hooks** – Accept pre and post hooks if set to *True*.
- **keep_project_on_failure** – If *True* keep generated project directory even when generation fails

2.1.11 cookiecutter.prompt module

Functions for prompting the user for project info.

cookiecutter.prompt.**process_json**(*user_value*, *default_value=None*)

Load user-supplied value as a JSON dict.

Parameters

user_value (*str*) – User-supplied value to load as a JSON dict

cookiecutter.prompt.**prompt_choice_for_config**(*cookiecutter_dict*, *env*, *key*, *options*, *no_input*, *prompts=None*)

Prompt user with a set of options to choose from.

Parameters

no_input – Do not prompt for user input and return the first available option.

cookiecutter.prompt.**prompt_for_config**(*context*, *no_input=False*)

Prompt user to enter a new config.

Parameters

- **context** (*dict*) – Source for field names and sample values.
- **no_input** – Do not prompt for user input and use only values from context.

cookiecutter.prompt.**read_repo_password**(*question*)

Prompt the user to enter a password.

Parameters

question (*str*) – Question to the user

cookiecutter.prompt.**read_user_choice**(*var_name*, *options*, *prompts=None*)

Prompt the user to choose from several options for the given variable.

The first item will be returned if no input happens.

Parameters

- **var_name** (*str*) – Variable as specified in the context
- **options** (*list*) – Sequence of options that are available to select from

Returns

Exactly one item of `options` that has been chosen by the user

cookiecutter.prompt.**read_user_dict**(*var_name*, *default_value*, *prompts=None*)

Prompt the user to provide a dictionary of data.

Parameters

- **var_name** (*str*) – Variable as specified in the context
- **default_value** – Value that will be returned if no input is provided

Returns

A Python dictionary to use in the context.

cookiecutter.prompt.**read_user_variable**(*var_name*, *default_value*, *prompts=None*)

Prompt user for variable and return the entered value or given default.

Parameters

- **var_name** (*str*) – Variable of the context to query the user
- **default_value** – Value that will be returned if no input happens

cookiecutter.prompt.**read_user_yes_no**(*var_name*, *default_value*, *prompts=None*)

Prompt the user to reply with 'yes' or 'no' (or equivalent values).

- These input values will be converted to `True`: "1", "true", "t", "yes", "y", "on"
- These input values will be converted to `False`: "0", "false", "f", "no", "n", "off"

Actual parsing done by `click.prompt()`; Check this function codebase change in case of unexpected behaviour.

Parameters

- **question** (*str*) – Question to the user
- **default_value** – Value that will be returned if no input happens

cookiecutter.prompt.**render_variable**(*env*, *raw*, *cookiecutter_dict*)

Render the next variable to be displayed in the user prompt.

Inside the prompting taken from the cookiecutter.json file, this renders the next variable. For example, if a project_name is "Peanut Butter Cookie", the repo_name could be be rendered with:

{{ cookiecutter.project_name.replace(" ", "_") }}.

This is then presented to the user as the default.

Parameters

- **env** (*Environment*) – A Jinja2 Environment object.
- **raw** – The next value to be prompted for by the user.
- **cookiecutter_dict** (*dict*) – The current context as it's gradually being populated with variables.

Returns

The rendered value for the default variable.

2.1.12 cookiecutter.replay module

cookiecutter.replay.

---

cookiecutter.replay.**dump**(*replay_dir*, *template_name*, *context*)

Write json data to file.

cookiecutter.replay.**get_file_name**(*replay_dir*, *template_name*)

Get the name of file.

cookiecutter.replay.**load**(*replay_dir*, *template_name*)

Read json data from file.

2.1.13 cookiecutter.repository module

Cookiecutter repository functions.

cookiecutter.repository.**determine_repo_dir**(*template*, *abbreviations*, *clone_to_dir*, *checkout*, *no_input*, *password=None*, *directory=None*)

Locate the repository directory from a template reference.

Applies repository abbreviations to the template reference. If the template refers to a repository URL, clone it. If the template is a path to a local repository, use it.

Parameters

- **template** – A directory containing a project template directory, or a URL to a git repository.
- **abbreviations** – A dictionary of repository abbreviation definitions.
- **clone_to_dir** – The directory to clone the repository into.
- **checkout** – The branch, tag or commit ID to checkout after clone.
- **no_input** – Do not prompt for user input and eventually force a refresh of cached resources.
- **password** – The password to use when extracting the repository.

- **directory** – Directory within repo where cookiecutter.json lives.

Returns
A tuple containing the cookiecutter template directory, and a boolean describing whether that directory should be cleaned up after the template has been instantiated.

Raises
RepositoryNotFound if a repository directory could not be found.

cookiecutter.repository.**expand_abbreviations**(*template*, *abbreviations*)

Expand abbreviations in a template name.

Parameters

- **template** – The project template name.
- **abbreviations** – Abbreviation definitions.

cookiecutter.repository.**is_repo_url**(*value*)

Return True if value is a repository URL.

cookiecutter.repository.**is_zip_file**(*value*)

Return True if value is a zip file.

cookiecutter.repository.**repository_has_cookiecutter_json**(*repo_directory*)

Determine if *repo_directory* contains a *cookiecutter.json* file.

Parameters
repo_directory – The candidate repository directory.

Returns
True if the *repo_directory* is valid, else False.

2.1.14 cookiecutter.utils module

Helper functions used throughout Cookiecutter.

cookiecutter.utils.**force_delete**(*func*, *path*, *exc_info*)

Error handler for *shutil.rmtree()* equivalent to *rm -rf*.

Usage: *shutil.rmtree(path, onerror=force_delete)* From https://docs.python.org/3/library/shutil.html#rmtree-example

cookiecutter.utils.**make_executable**(*script_path*)

Make *script_path* executable.

Parameters
script_path – The file to change

cookiecutter.utils.**make_sure_path_exists**(*path*)

Ensure that a directory exists.

Parameters
path (PathLike[str]) – A directory tree path for creation.

Return type
None

cookiecutter.utils.**prompt_and_delete**(*path*, *no_input=False*)

Ask user if it's okay to delete the previously-downloaded file/directory.

If yes, delete it. If no, checks to see if the old version should be reused. If yes, it's reused; otherwise, Cookiecutter exits.

Parameters

- **path** – Previously downloaded zipfile.
- **no_input** – Suppress prompt to delete repo and just delete it.

Returns

True if the content was deleted

cookiecutter.utils.**rmtree**(*path*)

Remove a directory and all its contents. Like rm -rf on Unix.

Parameters

path – A directory path.

cookiecutter.utils.**simple_filter**(*filter_function*)

Decorate a function to wrap it in a simplified jinja2 extension.

cookiecutter.utils.**work_in**(*dirname=None*)

Context manager version of os.chdir.

When exited, returns to the working directory prior to entering.

2.1.15 cookiecutter.vcs module

Helper functions for working with version control systems.

cookiecutter.vcs.**clone**(*repo_url*, *checkout=None*, *clone_to_dir='.'*, *no_input=False*)

Clone a repo to the current directory.

Parameters

- **repo_url** (str) – Repo URL of unknown type.
- **checkout** (Optional[str]) – The branch, tag or commit ID to checkout after clone.
- **clone_to_dir** (PathLike[str]) – The directory to clone to. Defaults to the current directory.
- **no_input** (bool) – Do not prompt for user input and eventually force a refresh of cached resources.

Returns

str with path to the new directory of the repository.

cookiecutter.vcs.**identify_repo**(*repo_url*)

Determine if *repo_url* should be treated as a URL to a git or hg repo.

Repos can be identified by prepending "hg+" or "git+" to the repo URL.

Parameters

repo_url – Repo URL of unknown type.

Returns

('git', repo_url), ('hg', repo_url), or None.

cookiecutter.vcs.**is_vcs_installed**(*repo_type*)

Check if the version control system for a repo type is installed.

Parameters

repo_type –

2.1.16 cookiecutter.zipfile module

Utility functions for handling and fetching repo archives in zip format.

cookiecutter.zipfile.**unzip**(*zip_uri*, *is_url*, *clone_to_dir='.'*, *no_input=False*, *password=None*)

Download and unpack a zipfile at a given URI.

This will download the zipfile to the cookiecutter repository, and unpack into a temporary directory.

Parameters

- **zip_uri** (str) – The URI for the zipfile.
- **is_url** (bool) – Is the zip URI a URL or a file?
- **clone_to_dir** (PathLike[str]) – The cookiecutter repository directory to put the archive into.
- **no_input** (bool) – Do not prompt for user input and eventually force a refresh of cached resources.
- **password** (Optional[str]) – The password to use when unpacking the repository.

2.1.17 Module contents

Main package for Cookiecutter.

CHAPTER

THREE

PROJECT INFO

3.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

3.1.1 Types of Contributions

You can contribute in many ways:

Report Bugs

Report bugs at https://github.com/cookiecutter/cookiecutter/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- If you can, provide detailed steps to reproduce the bug.
- If you don't have steps to reproduce the bug, just note your observations in as much detail as you can. Questions to start a discussion about the issue are welcome.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "please-help" is open to whoever wants to implement it.

Please do not combine multiple feature enhancements into a single pull request.

Note: this project is very conservative, so new features that aren't tagged with "please-help" might not get into core. We're trying to keep the code base small, extensible, and streamlined. Whenever possible, it's best to try and implement feature ideas as separate projects outside of the core codebase.

Write Documentation

Cookiecutter could always use more documentation, whether as part of the official Cookiecutter docs, in docstrings, or even on the web in blog posts, articles, and such.

If you want to review your changes on the documentation locally, you can do:

```
pip install -r docs/requirements.txt
make servedocs
```

This will compile the documentation, open it in your browser and start watching the files for changes, recompiling as you save.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/cookiecutter/cookiecutter/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.1.2 Setting Up the Code for Local Development

Here's how to set up `cookiecutter` for local development.

1. Fork the `cookiecutter` repo on GitHub.
2. Clone your fork locally:

   ```
   git clone git@github.com:your_name_here/cookiecutter.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   cd cookiecutter/
   pip install -e .
   ```

4. Create a branch for local development:

   ```
   git checkout -b name-of-your-bugfix-or-feature
   ```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass the tests and lint check:

```
pip install tox
tox
```

6. Ensure that your feature or commit is fully covered by tests. Check report after regular `tox` run. You can also run coverage only report and get html report with statement by statement highlighting:

```
make coverage
```

   You report will be placed to `htmlcov` directory. Please do not include this directory to your commits. By default this directory in our `.gitignore` file.

7. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

3.1.3 Contributor Guidelines

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. The pull request should be contained: if it's too big consider splitting it into smaller pull requests.
3. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.md.
4. The pull request must pass all CI/CD jobs before being ready for review.
5. If one CI/CD job is failing for unrelated reasons you may want to create another PR to fix that first.

Coding Standards

- PEP8
- Functions over classes except in tests
- Quotes via http://stackoverflow.com/a/56190/5549
  - Use double quotes around strings that are used for interpolation or that are natural language messages
  - Use single quotes for small symbol-like strings (but break the rules if the strings contain quotes)
  - Use triple double quotes for docstrings and raw string literals for regular expressions even if they aren't needed.
  - Example:

```
LIGHT_MESSAGES = {
    'English': "There are %(number_of_lights)s lights.",
    'Pirate':  "Arr! Thar be %(number_of_lights)s lights."
```

(continues on next page)

(continued from previous page)

```
}
def lights_message(language, number_of_lights):
    """Return a language-appropriate string reporting the light count."""
    return LIGHT_MESSAGES[language] % locals()
def is_pirate(message):
    """Return True if the given message sounds piratical."""
    return re.search(r"(?i)(arr|avast|yohoho)!", message) is not None
```

3.1.4 Testing with tox

`tox` uses `pytest` under the hood, hence it supports the same syntax for selecting tests.

For further information please consult the pytest usage docs.

To run a particular test class with `tox`:

```
tox -e py310 -- '-k TestFindHooks'
```

To run some tests with names matching a string expression:

```
tox -e py310 -- '-k generate'
```

Will run all tests matching "generate", test_generate_files for example.

To run just one method:

```
tox -e py310 -- '-k "TestFindHooks and test_find_hook"'
```

To run all tests using various versions of Python, just run `tox`:

```
tox
```

This configuration file setup the pytest-cov plugin and it is an additional dependency. It generate a coverage report after the tests.

It is possible to test with specific versions of Python. To do this, the command is:

```
tox -e py37,py38
```

This will run `py.test` with the `python3.7` and `python3.8` interpreters.

3.1.5 Core Committer Guide

Vision and Scope

Core committers, use this section to:

- Guide your instinct and decisions as a core committer
- Limit the codebase from growing infinitely

Command-Line Accessible

- Provides a command-line utility that creates projects from cookiecutters
- Extremely easy to use without having to think too hard
- Flexible for more complex use via optional arguments

API Accessible

- Entirely function-based and stateless (Class-free by intentional design)
- Usable in pieces for developers of template generation tools

Being Jinja2-specific

- Sets a standard baseline for project template creators, facilitating reuse
- Minimizes the learning curve for those who already use Flask or Django
- Minimizes scope of Cookiecutter codebase

Extensible

Being extendable by people with different ideas for Jinja2-based project template tools.

- Entirely function-based
- Aim for statelessness
- Lets anyone write more opinionated tools

Freedom for Cookiecutter users to build and extend.

- No officially-maintained cookiecutter templates, only ones by individuals
- Commercial project-friendly licensing, allowing for private cookiecutters and private Cookiecutter-based tools

Fast and Focused

Cookiecutter is designed to do one thing, and do that one thing very well.

- Cover the use cases that the core committers need, and as little as possible beyond that :)
- Generates project templates from the command-line or API, nothing more
- Minimize internal line of code (LOC) count
- Ultra-fast project generation for high performance downstream tools

Inclusive

- Cross-platform and cross-version support are more important than features/functionality
- Fixing Windows bugs even if it's a pain, to allow for use by more beginner coders

Stable

- Aim for 100% test coverage and covering corner cases
- No pull requests will be accepted that drop test coverage on any platform, including Windows
- Conservative decisions patterned after CPython's conservative decisions with stability in mind
- Stable APIs that tool builders can rely on
- New features require a +1 from 3 core committers

VCS-Hosted Templates

Cookiecutter project templates are intentionally hosted VCS repos as-is.

- They are easily forkable
- It's easy for users to browse forks and files
- They are searchable via standard Github/Bitbucket/other search interface
- Minimizes the need for packaging-related cruft files
- Easy to create a public project template and host it for free
- Easy to collaborate

Process: Pull Requests

If a pull request is untriaged:

- Look at the roadmap
- Set it for the milestone where it makes the most sense
- Add it to the roadmap

How to prioritize pull requests, from most to least important:

- Fixes for broken tests. Broken means broken on any supported platform or Python version.
- Extra tests to cover corner cases.
- Minor edits to docs.
- Bug fixes.
- Major edits to docs.
- Features.

Pull Requests Review Guidelines

- Think carefully about the long-term implications of the change. How will it affect existing projects that are dependent on this? If this is complicated, do we really want to maintain it forever?
- Take the time to get things right, PRs almost always require additional improvements to meet the bar for quality. **Be very strict about quality.**
- When you merge a pull request take care of closing/updating every related issue explaining how they were affected by those changes. Also, remember to add the author to `AUTHORS.md`.

Process: Issues

If an issue is a bug that needs an urgent fix, mark it for the next patch release. Then either fix it or mark as please-help.

For other issues: encourage friendly discussion, moderate debate, offer your thoughts.

New features require a +1 from 2 other core committers (besides yourself).

Process: Roadmap

The roadmap located here

Due dates are flexible. Core committers can change them as needed. Note that GitHub sort on them is buggy.

How to number milestones:

- Follow semantic versioning. Look at: http://semver.org

Milestone size:

- If a milestone contains too much, move some to the next milestone.
- Err on the side of more frequent patch releases.

Process: Your own code changes

All code changes, regardless of who does them, need to be reviewed and merged by someone else. This rule applies to all the core committers.

Exceptions:

- Minor corrections and fixes to pull requests submitted by others.
- While making a formal release, the release manager can make necessary, appropriate changes.
- Small documentation changes that reinforce existing subject matter. Most commonly being, but not limited to spelling and grammar corrections.

Responsibilities

- Ensure cross-platform compatibility for every change that's accepted. Windows, macOS and Linux.
- Create issues for any major changes and enhancements that you wish to make. Discuss things transparently and get community feedback.
- Don't add any classes to the codebase unless absolutely needed. Err on the side of using functions.
- Keep feature versions as small as possible, preferably one new feature per version.
- Be welcoming to newcomers and encourage diverse new contributors from all backgrounds. Look at *Code of Conduct*.

Becoming a Core Committer

Contributors may be given core commit privileges. Preference will be given to those with:

1. Past contributions to Cookiecutter and other open-source projects. Contributions to Cookiecutter include both code (both accepted and pending) and friendly participation in the issue tracker. Quantity and quality are considered.
2. A coding style that the other core committers find simple, minimal, and clean.
3. Access to resources for cross-platform development and testing.
4. Time to devote to the project regularly.

3.2 Credits

3.2.1 Development Leads

- Audrey Roy Greenfeld (@audreyfeldroy)
- Daniel Roy Greenfeld (@pydanny)
- Raphael Pierzina (@hackebrot)

3.2.2 Core Committers

- Michael Joseph (@michaeljoseph)
- Paul Moore (@pfmoore)
- Andrey Shpak (@insspb)
- Sorin Sbarnea (@ssbarnea)
- Fábio C. Barrionuevo da Luz (@luzfcb)
- Simone Basso (@simobasso)
- Jens Klein (@jensens)
- Érico Andrei (@ericof)

3.2.3 Contributors

- Steven Loria (@sloria)
- Goran Peretin (@gperetin)
- Hamish Downer (@foobacca)
- Thomas Orozco (@krallin)
- Jindrich Smitka (@s-m-i-t-a)
- Benjamin Schwarze (@benjixx)
- Raphi (@raphigaziano)
- Thomas Chiroux (@ThomasChiroux)
- Sergi Almacellas Abellana (@pokoli)
- Alex Gaynor (@alex)
- Rolo (@rolo)
- Pablo (@oubiga)
- Bruno Rocha (@rochacbruno)
- Alexander Artemenko (@svetlyak40wt)
- Mahmoud Abdelkader (@mahmoudimus)
- Leonardo Borges Avelino (@lborgav)
- Chris Trotman (@solarnz)
- Rolf (@relekang)
- Noah Kantrowitz (@coderanger)
- Vincent Bernat (@vincentbernat)
- Germán Moya (@pbacterio)
- Ned Batchelder (@nedbat)
- Dave Dash (@davedash)
- Johan Charpentier (@cyberj)
- Éric Araujo (@merwok)
- saxix (@saxix)
- Tzu-ping Chung (@uranusjr)
- Caleb Hattingh (@cjrh)
- Flavio Curella (@fcurella)
- Adam Venturella (@aventurella)
- Monty Taylor (@emonty)
- schacki (@schacki)
- Ryan Olson (@ryanolson)
- Trey Hunner (@treyhunner)
- Russell Keith-Magee (@freakboy3742)

- Mishbah Razzaque (@mishbahr)
- Robin Andeer (@robinandeer)
- Rachel Sanders (@trustrachel)
- Rémy Hubscher (@Natim)
- Dino Petron3 (@dinopetrone)
- Peter Inglesby (@inglesp)
- Ramiro Batista da Luz (@ramiroluz)
- Omer Katz (@thedrow)
- lord63 (@lord63)
- Randy Syring (@rsyring)
- Mark Jones (@mark0978)
- Marc Abramowitz (@msabramo)
- Lucian Ursu (@LucianU)
- Osvaldo Santana Neto (@osantana)
- Matthias84 (@Matthias84)
- Simeon Visser (@svisser)
- Guruprasad (@lgp171188)
- Charles-Axel Dein (@charlax)
- Diego Garcia (@drgarcia1986)
- maiksensi (@maiksensi)
- Andrew Conti (@agconti)
- Valentin Lab (@vaab)
- Ilja Bauer (@iljabauer)
- Elias Dorneles (@eliasdorneles)
- Matias Saguir (@mativs)
- Johannes (@johtso)
- macrotim (@macrotim)
- Will McGinnis (@wdm0006)
- Cédric Krier (@cedk)
- Tim Osborn (@ptim)
- Aaron Gallagher (@habnabit)
- mozillazg (@mozillazg)
- Joachim Jablon (@ewjoachim)
- Andrew Ittner (@tephyr)
- Diane DeMers Chen (@purplediane)
- zzzirk (@zzzirk)

- Carol Willing (@willingc)
- phoebebauer (@phoebebauer)
- Adam Chainz (@adamchainz)
- Sulé (@suledev)
- Evan Palmer (@palmerev)
- Bruce Eckel (@BruceEckel)
- Robert Lyon (@ivanlyon)
- Terry Bates (@terryjbates)
- Brett Cannon (@brettcannon)
- Michael Warkentin (@mwarkentin)
- Bartłomiej Kurzeja (@B3QL)
- Thomas O'Donnell (@andytom)
- Jeremy Carbaugh (@jcarbaugh)
- Nathan Cheung (@cheungnj)
- Abdó Roig-Maranges (@aroig)
- Steve Piercy (@stevepiercy)
- Corey (@coreysnyder04)
- Dmitry Evstratov (@devstrat)
- Eyal Levin (@eyalev)
- mathagician (@mathagician)
- Guillaume Gelin (@ramnes)
- @delirious-lettuce (@delirious-lettuce)
- Gasper Vozel (@karantan)
- Joshua Carp (@jmcarp)
- @meahow (@meahow)
- Andrea Grandi (@andreagrandi)
- Issa Jubril (@jubrilissa)
- Nytiennzo Madooray (@Nythiennzo)
- Erik Bachorski (@dornheimer)
- cclauss (@cclauss)
- Andy Craze (@accraze)
- Anthony Sottile (@asottile)
- Jonathan Sick (@jonathansick)
- Hugo (@hugovk)
- Min ho Kim (@minho42)
- Ryan Ly (@rly)

- Akintola Rahmat (@mihrab34)
- Jai Ram Rideout (@jairideout)
- Diego Carrasco Gubernatis (@dacog)
- Wagner Negrão (@wagnernegrao)
- Josh Barnes (@jcb91)
- Nikita Sobolev (@sobolevn)
- Matt Stibbs (@mattstibbs)
- MinchinWeb (@MinchinWeb)
- kishan (@kishan)
- tonytheleg (@tonytheleg)
- Roman Hartmann (@RomHartmann)
- DSEnvel (@DSEnvel)
- kishan (@kishan)
- Bruno Alla (@browniebroke)
- nicain (@nicain)
- Carsten Rösnick-Neugebauer (@croesnick)
- igorbasko01 (@igorbasko01)
- Dan Booth Dev (@DanBoothDev)
- Pablo Panero (@ppanero)
- Chuan-Heng Hsiao (@chhsiao1981)
- Mohammad Hossein Sekhavat (@mhsekhavat)
- Amey Joshi (@amey589)
- Paul Harrison (@smoothml)
- Fabio Todaro (@SharpEdgeMarshall)
- Nicholas Bollweg (@bollwyvl)
- Jace Browning (@jacebrowning)
- Ionel Cristian Mărieș (@ionelmc)
- Kishan Mehta (@kishan3)
- Wieland Hoffmann (@mineo)
- Antony Lee (@anntzer)
- Aurélien Gâteau (@agateau)
- Axel H. (@noirbizarre)
- Chris (@chrisbrake)
- Chris Streeter (@streeter)
- Gábor Lipták (@gliptak)
- Javier Sánchez Portero (@javiersanp)

- Nimrod Milo (@milonimrod)
- Philipp Kats (@Casyfill)
- Reinout van Rees (@reinout)
- Rémy Greinhofer (@rgreinho)
- Sebastian (@sebix)
- Stuart Mumford (@Cadair)
- Tom Forbes (@orf)
- Xie Yanbo (@xyb)
- Maxim Ivanov (@ivanovmg)

3.2.4 Backers

We would like to thank the following people for supporting us in our efforts to maintain and improve Cookiecutter:

- Alex DeBrie
- Alexandre Y. Harano
- Bruno Alla
- Carol Willing
- Russell Keith-Magee

3.2.5 Sprint Contributors

PyCon 2016 Sprint

The following people made contributions to the cookiecutter project at the PyCon sprints in Portland, OR from June 2-5 2016. Contributions include user testing, debugging, improving documentation, reviewing issues, writing tutorials, creating and updating project templates, and teaching each other.

- Adam Chainz (@adamchainz)
- Andrew Ittner (@tephyr)
- Audrey Roy Greenfeld (@audreyfeldroy)
- Carol Willing (@willingc)
- Christopher Clarke (@chrisdev)
- Citlalli Murillo (@citmusa)
- Daniel Roy Greenfeld (@pydanny)
- Diane DeMers Chen (@purplediane)
- Elaine Wong (@elainewong)
- Elias Dorneles (@eliasdorneles)
- Emily Cain (@emcain)
- John Roa (@jhonjairoroa87)
- Jonan Scheffler (@1337807)

- Phoebe Bauer (@phoebebauer)
- Kartik Sundararajan (@skarbot)
- Katia Lira (@katialira)
- Leonardo Jimenez (@xpostudio4)
- Lindsay Slazakowski (@lslaz1)
- Meghan Heintz (@dot2dotseurat)
- Raphael Pierzina (@hackebrot)
- Umair Ashraf (@umrashrf)
- Valdir Stumm Junior (@stummjr)
- Vivian Guillen (@viviangb)
- Zaro (@zaro0508)

3.3 History

History is important, but our current roadmap can be found here

3.3.1 2.2.0 (2023-07-06)

Changes

- Added timeout on request.get() for ensuring that if a recipient serve… (#1772) @openrefactory
- Fixing Carriage Return Line Feed (CRLF) order in docs #1792 (#1793) @Lahiry
- Reduce I/O (#1877) @kurtmckee
- Remove a pre-commit hook special case (#1875) @kurtmckee
- Remove universal bdist_wheel option; use "python -m build" (#1739) @mwtoews
- Remove unused import from post-generate hook script example (#1795) @KAZYPinkSaurus
- Standardize newlines for all platforms (#1870) @kurtmckee
- feat: Add resolved template repository path as _repo_dir to the context (#1771) @tmeckel

Minor Changes

- Added support for providing human-readable prompts to the different variables (#1881) @vemonet
- Added: Boolean variable support in JSON (#1626) @liortct
- Added: CLI option to keep project files on failure. (#1669) @MaciejPatro
- Added: Support partially overwrite keys in nested dict (#1692) @cksac
- Added: Templates inheritance (#1485) @simobasso
- Code quality: Tests upgrade: Use pathlib for files read/write (#1718) @insspb
- Inline jinja2-time extension code (#1779) @tranzystorek-io

- Support Python 3.11 (#1850) @kurtmckee
- Support nested config files (#1770) @dariocurr
- preserves original options in `_cookiecutter` (#1874) @kjaymiller

CI/CD and QA changes

- Add a Dependabot config to autoupdate GitHub workflow actions (#1851) @kurtmckee
- Added: Readthedocs build config (#1707) @insspb
- Bump actions/setup-python from 3 to 4 (#1854) @dependabot
- Bump paambaati/codeclimate-action from 3.0.0 to 4.0.0 (#1853) @dependabot
- CI/CD: Tox -> Nox: Added nox configuration (#1706) @insspb
- CI/CD: Tox -> Nox: Github actions definition minimized + Sync nox and github actions (#1714) @insspb
- CI/CD: Tox -> Nox: Makefile update: Removed watchmedo and sed dependency, tox replaced with nox (#1713) @insspb
- CI/CD: Updated .pre-commit-config.yaml to use latest hooks versions (#1712) @insspb
- Code quality: Core files: Added exception reason reraise when exception class changed (PEP 3134) (#1719) @insspb
- Code quality: Tests upgrade: Use pathlib for files read/write (#1718) @insspb
- Code quality: core files: Format replaced with f-strings (#1716) @insspb
- Code quality: find.py refactored and type annotated (#1721) @insspb
- Code quality: tests files: Simplify statements fixes (#1717) @insspb
- Code quality: utils.make_sure_path_exists refactored and type annotated (#1722) @insspb
- Fixed: recommonmark replaced with myst, as recommonmark is deprecated (#1709) @insspb
- Pretty-format JSON files (#1864) @kurtmckee
- Rename `master` to `main` so CI runs correctly on merge (#1852) @kurtmckee
- Standardize EOF newlines (#1876) @kurtmckee
- Update `.gitignore` and cite where it was copied from (#1879) @kurtmckee
- Update base docs, remove tox (#1858) @ericof
- Update pre-commit hook versions (#1849) @kurtmckee
- Updated: Release drafter configuration (#1704) @insspb
- Use tox (#1866) @kurtmckee
- Verify an expected warning is raised (#1869) @kurtmckee
- fixed failing lint ci action by updating repo of flake8 (#1838) @Tamronimus

Documentation updates

- Add jinja env docs (#1872) @pamelafox
- Documentation extension: Create a Cookiecutter From Scratch tutorial (#1592) @miro-jelaska
- Easy PR! Fix typos and add minor doc updates (#1741) @Alex0Blackwell
- Expand cli documentation relating to the no-input flag (#1543) (#1587) @jeremyswerdlow
- Fix @audreyr to @audreyfeldroy github account rename (#1604) @ri0t
- Fixed broken links to jinja docs (#1691) @insspb
- Fixed minor typos in docs (#1753) @segunb
- Fixed: Python code block in the replay documentation (#1715) @juhannc
- Fixed: recommonmark replaced with myst, as recommonmark is deprecated (#1709) @insspb
- Improve Docs Readability (#1690) @ryanrussell
- Update base docs, remove tox (#1858) @ericof
- Updated: Boolean Variables documentation and docstrings (#1705) @italomaia
- docs: fix simple typo, shat -> that (#1749) @timgates42
- fixing badge display problem (#1798) @Paulokim1

Bugfixes

- Fixed the override not working with copy only dir #1650 (#1651) @zhongdai
- Fixed: Removed mention of packages versions, to exclude dependabot warnings alerts (#1711) @insspb
- cleanup files if panics during hooks - bugfix (#1760) @liortct

This release is made by wonderful contributors:

@Alex0Blackwell, @KAZYPinkSaurus, @Lahiry, @MaciejPatro, @Paulokim1, @Tamronimus, @cksac, @cookies-xor-cream, @dariocurr, @dependabot, @dependabot[bot], @ericof, @insspb, @italomaia, @jeremyswerdlow, @juhannc, @kjaymiller, @kurtmckee, @liortct, @miro-jelaska, @mwtoews, @openrefactory, @pamelafox, @ri0t, @ryanrussell, @segunb, @simobasso, @timgates42, @tmeckel, @tranzystorek-io, @vemonet and @zhongdai

3.3.2 2.1.1 (2022-06-01)

Documentation updates

- Fix local extensions documentation (#1686) @alkatar21

Bugfixes

- Sanitize Mercurial branch information before checkout. (#1689) @ericof

This release is made by wonderfull contributors:

@alkatar21, @ericof and @jensens

3.3.3 2.1.0 (2022-05-30)

Changes

- Move contributors and backers to credits section (#1599) @doobrie
- test_generate_file_verbose_template_syntax_error fixed (#1671) @MaciejPatro
- Removed changes related to setuptools_scm (#1629) @ozer550
- Feature/local extensions (#1240) @mwesterhof

CI/CD and QA changes

- Check manifest: pre-commit, fixes, cleaning (#1683) @jensens
- Follow PyPA guide to release package using GitHub Actions. (#1682) @ericof

Documentation updates

- Fix typo in dict_variables.rst (#1680) @ericof
- Documentation overhaul (#1677) @jensens
- Fixed incorrect link on docs. (#1649) @luzfcb

Bugfixes

- Restore accidentally deleted support for click 8.x (#1643) @jaklan

This release was made possible by our wonderful contributors:

@doobrie, @jensens, @ericof, @luzfcb

3.3.4 2.0.2 (2021-12-27)

Remark: This release never made it to official PyPI

- Fix Python version number in cookiecutter –version and test on Python 3.10 (#1621) @ozer550
- Removed changes related to setuptools_scm (#1629) @audreyfeldroy @ozer550

3.3.5 2.0.1 (2021-12-11)

Remark: This release never made it to official PyPI

Breaking Changes

- Release preparation for 2.0.1rc1 (#1608) @audreyfeldroy
- Replace poyo with pyyaml. (#1489) @dHannasch
- Added: Path templates will be rendered when copy_without_render used (#839) @noirbizarre
- Added: End of line detection and configuration. (#1407) @insspb
- Remove support for python2.7 (#1386) @ssbarnea

Minor Changes

- Adopt setuptools-scm packaging (#1577) @ssbarnea
- Log the error message when git clone fails, not just the return code (#1505) @logworthy
- allow jinja 3.0.0 (#1548) @wouterdb
- Added uuid extension to be able to generate uuids (#1493) @jonaswre
- Alert user if choice is invalid (#1496) @dHannasch
- Replace poyo with pyyaml. (#1489) @dHannasch
- update AUTHOR lead (#1532) @HosamAlmoghraby
- Add Python 3.9 (#1478) @gliptak
- Added: –list-installed cli option, listing already downloaded cookiecutter packages (#1096) @chrisbrake
- Added: Jinja2 Environment extension on files generation stage (#1419) @insspb
- Added: –replay-file cli option, for replay file distributing (#906) @Cadair
- Added: _output_dir to cookiecutter context (#1034) @Casyfill
- Added: CLI option to ignore hooks (#992) @rgreinho
- Changed: Generated projects can use multiple type hooks at same time. (sh + py) (#974) @milonimrod
- Added: Path templates will be rendered when copy_without_render used (#839) @noirbizarre
- Added: End of line detection and configuration. (#1407) @insspb
- Making code python 3 only: Remove python2 u’ sign, fix some strings (#1402) @insspb
- py3: remove futures, six and encoding (#1401) @insspb
- Render variables starting with an underscore. (#1339) @smoothml
- Tests refactoring: test_utils write issues fixed #1405 (#1406) @insspb

CI/CD and QA changes

- enable branch coverage (#1542) @simobasso
- Make release-drafter diff only between master releases (#1568) @SharpEdgeMarshall
- ensure filesystem isolation during tests execution (#1564) @simobasso
- add safety ci step (#1560) @simobasso
- pre-commit: add bandit hook (#1559) @simobasso
- Replace tmpdir in favour of tmp_path (#1545) @SharpEdgeMarshall
- Fix linting in CI (#1546) @SharpEdgeMarshall
- Coverage 100% (#1526) @SharpEdgeMarshall
- Run coverage with matrix (#1521) @SharpEdgeMarshall
- Lint rst files (#1443) @ssbarnea
- Python3: Changed io.open to build-in open (PEP3116) (#1408) @insspb
- Making code python 3 only: Remove python2 u' sign, fix some strings (#1402) @insspb
- py3: remove futures, six and encoding (#1401) @insspb
- Removed: Bumpversion, setup.py arguments. (#1404) @insspb
- Tests refactoring: test_utils write issues fixed #1405 (#1406) @insspb
- Added: Automatic PyPI deploy on tag creation (#1400) @insspb
- Changed: Restored coverage reporter (#1399) @insspb

Documentation updates

- Fix pull requests checklist reference (#1537) @glumia
- Fix author name (#1544) @HosamAlmoghraby
- Add missing contributors (#1535) @glumia
- Update CONTRIBUTING.md (#1529) @glumia
- Update LICENSE (#1519) @simobasso
- docs: rewrite the conditional files / directories example description. (#1437) @lyz-code
- Fix incorrect years in release history (#1473) @graue70
- Add slugify in the default extensions list (#1470) @oncleben31
- Renamed cookiecutter.package to API (#1442) @grrlic
- Fixed wording detail (#1427) @steltenpower
- Changed: CLI Commands documentation engine (#1418) @insspb
- Added: Example for conditional files / directories in hooks (#1397) @xyb
- Changed: README.md PyPI URLs changed to the modern PyPI last version (#1391) @brettcannon
- Fixed: Comma in README.md (#1390) @Cy-dev-tex
- Fixed: Replaced no longer maintained pipsi by pipx (#1395) @ndclt

Bugfixes

- Add support for click 8.x (#1569) @cjolowicz
- Force click<8.0.0 (#1562) @SharpEdgeMarshall
- Remove direct dependency on markupsafe (#1549) @ssbarnea
- fixes prompting private rendered dicts (#1504) @juhuebner
- User's JSON parse error causes ugly Python exception #809 (#1468) @noone234
- config: set default on missing default_context key (#1516) @simobasso
- Fixed: Values encoding on Windows (#1414) @agateau
- Fixed: Fail with gitolite repositories (#1144) @javiersanp
- MANIFEST: Fix file name extensions (#1387) @sebix

Deprecations

- Removed: Bumpversion, setup.py arguments. (#1404) @insspb
- Removed support for Python 3.6 and PyPy (#1608) @audreyfeldroy

This release was made possible by our wonderful contributors:

@Cadair, @Casyfill, @Cy-dev-tex, @HosamAlmoghraby, @SharpEdgeMarshall, @agateau, @audreyfeldroy, @brettcannon, @chrisbrake, @cjolowicz, @dHannasch, @gliptak, @glumia, @graue70, @grrlic, @insspb, @javiersanp, @jonaswre, @jsoref, @Jthevos, @juhuebner, @logworthy, @lyz-code, @milonimrod, @ndclt, @noirbizarre, @noone234, @oncleben31, @ozer550, @rgreinho, @sebix, @Sahil-101, @simobasso, @smoothml, @ssbarnea, @steltenpower, @wouterdb, @xyb, Christopher Wolfe and Hosam Almoghraby (RIAG Digital)

3.3.6 1.7.2 (2020-04-21)

- Fixed: Jinja2&Six version limits causing build errors with ansible project @insspb (#1385)

3.3.7 1.7.1 (2020-04-21)

This release was focused on internal code and CI/CD changes. During this release all code was verified to match pep8, pep257 and other code-styling guides. Project CI/CD was significantly changed, Windows platform checks based on Appveyor engine was replaced by GitHub actions tests. Appveyor was removed. Also our CI/CD was extended with Mac builds, to verify project builds on Apple devices.

Important Changes:

- Added: Added debug messages for get_user_config @ssbarnea (#1357)
- Multiple templates per one repository feature added. @RomHartmann (#1224, #1063)
- Update replay.py json.dump indent for easy viewing @nicain (#1293)
- 'future' library replaced with 'six' as a more lightweight python porting library @asottile (#941)
- Added extension: Slugify template filter @ppanero (#1336)
- Added command line option: `--skip-if-file-exists`, allow to skip the existing files when doing `overwrite_if_exists`. @chhsiao1981 (#1076)

- Some packages versions limited to be compatible with python2.7 and python 3.5 @insspb (#1349)

Internal CI/CD and tests changes:

- Coverage comment in future merge requests disabled @ssbarnea (#1279)
- Fixed Python 3.8 travis tests and setup.py message @insspb (#1295, #1297)
- Travis builds extended with Windows setup for all supported python versions @insspb (#1300, #1301)
- Update .travis.yml to be compatible with latest travis cfg specs @luzfcb (#1346)
- Added new test to improve tests coverage @amey589 (#1023)
- Added missed coverage lines highlight to pytest-coverage report @insspb (#1352)
- pytest-catchlog package removed from test_requirements, as now it is included in pytest @insspb (#1347)
- Fixed `cov-report` tox invocation environment @insspb (#1350)
- Added: Release drafter support and configuration to exclude changelog update work and focus on development @ssbarnea @insspb (#1356, #1362)
- Added: CI/CD steps for Github actions to speedup CI/CD @insspb (#1360)
- Removed: Appveyor CI/CD completely removed @insspb @ssbarnea @insspb (#1363, #1367)

Code style and docs changes:

- Added black formatting verification on lint stage + project files reformatting @ssbarnea @insspb (#1368)
- Added pep257 docstring for tests/* files @insspb (#1369, #1370, #1371, #1372, #1373, #1374, #1375, #1376, #1377, #1378, #1380, #1381)
- Added pep257 docstring for tests/conftests.py @kishan (#1272, #1263)
- Added pep257 docstring for tests/replay/conftest.py @kishan (#1270, #1268)
- Added pep257 docstring for docs/**init**.py @kishan (#1273, #1265)
- Added missing docstring headers to all files @croesnick (#1269, #1283)
- Gitter links replaced by Slack in README @browniebroke (#1282)
- flake8-docstrings tests added to CI/CD @ssbarnea (#1284)
- Activated pydocstyle rule: D401 - First line should be in imperative mood @ssbarnea (#1285)
- Activated pydocstyle rule: D200 - One-line docstring should fit on one line with quotes @ssbarnea (#1288)
- Activated pydocstyle rule: D202 - No blank lines allowed after function docstring @ssbarnea (#1288)
- Activated pydocstyle rule: D205 - 1 blank line required between summary line and description @ssbarnea (#1286, #1287)
- Activated pydocstyle rule: ABS101 @ssbarnea (#1288)
- Replaced click documentation links to point to version 7 @igorbasko01 (#1303)
- Updated submodule link to latest version with documentation links fix @DanBoothDev (#1388)
- Fixed links in main README file. @insspb (#1342)
- Fix indentation of .cookiecutterrc in README.md @mhsekhavat (#1322)
- Changed format of loggers invocation @insspb (#1307)

3.3.8 1.7.0 (2019-12-22) Old friend

Important Changes:

- Drop support for EOL Python 3.4, thanks to @jamescurtin and @insspb (#1024)
- Drop support for EOL Python 3.3, thanks to @hugovk (#1024)
- Increase the minimum click version to 7.0, thanks to @rly and @luzfcb (#1168)

Other Changes:

- PEP257 fixing docstrings in exceptions.py. Thanks to @MinchinWeb (#1237)
- PEP257 fixing docstrings in replay.py. Thanks to @kishan (#1234)
- PEP257 fixing docstrings in test_unzip.py. Thanks to @tonytheleg and @insspb (#1236, #1262)
- Fixed tests sequence for appveyor, to exclude file not found bug. Thanks to @insspb (#1257)
- Updates REAMDE.md with svg badge for appveyor. Thanks to @sobolevn (#1254)
- Add missing {% endif %} to Choice Variables example. Thanks to @mattstibbs (#1249)
- Core documentation converted to Markdown format thanks to @wagnernegrao, @insspb (#1216)
- Tests update: use sys.executable when invoking python in python 3 only environment thanks to @vincentbernat (#1221)
- Prevent `click` API v7.0 from showing choices when already shown, thanks to @rly and @luzfcb (#1168)
- Test the codebase with python3.8 beta on tox and travis-ci (#1206), thanks to @mihrab34
- Add a CODE_OF_CONDUCT.md file to the project, thanks to @andreagrandi (#1009)
- Update docstrings in `cookiecutter/main.py`, `cookiecutter/__init__.py`, and `cookiecutter/log.py` to follow the PEP 257 style guide, thanks to @meahow (#998, #999, #1000)
- Update docstrings in `cookiecutter/utils.py` to follow the PEP 257 style guide, thanks to @dornheimer(#1026)
- Fix grammar in *Choice Variables* documentation, thanks to @jubrilissa (#1011)
- Update installation docs with links to the Windows Subsystem and GNU utilities, thanks to @Nythiennzo for the PR and @BruceEckel for the review (#1016)
- Upgrade flake8 to version 3.5.0, thanks to @cclauss (#1038)
- Update tutorial with explanation for how cookiecutter finds the template file, thanks to @accraze(#1025)
- Update CI config files to use `TOXENV` environment variable, thanks to @asottile (#1019)
- Improve user documentation for writing hooks, thanks to @jonathansick (#1057)
- Make sure to preserve the order of items in the generated cookiecutter context, thanks to @hackebrot (#1074)
- Fixed DeprecationWarning for a regular expression on python 3.6, thanks to @reinout (#1124)
- Document use of cookiecutter-template topic on GitHub, thanks to @ssbarnea (#1189)
- Update README badge links, thanks to @luzfcb (#1207)
- Update prompt.py to match pep257 guidelines, thanks to @jairideout (#1105)
- Update link to Jinja2 extensions documentation, thanks to @dacog (#1193)
- Require pip 9.0.0 or newer for tox environments, thanks to @hackebrot (#1215)
- Use io.open contextmanager when reading hook files, thanks to @jcb91 (#1147)

- Add more cookiecutter templates to the mix:
  - cookiecutter-python-cli by @xuanluong (#1003)
  - cookiecutter-docker-science by @takahi-i (#1040)
  - cookiecutter-flask-skeleton by @mjhea0 (#1052)
  - cookiecutter-awesome by @Pawamoy (#1051)
  - cookiecutter-flask-ask by @machinekoder (#1056)
  - cookiecutter-data-driven-journalism by @JAStark (#1020)
  - cookiecutter-tox-plugin by @obestwalter (#1103)
  - cookiecutter-django-dokku by @mashrikt (#1093)

3.3.9 1.6.0 (2017-10-15) Tim Tam

New Features:

- Include template path or template URL in cookiecutter context under `_template`, thanks to @aroig (#774)
- Add a URL abbreviation for GitLab template projects, thanks to @hackebrot (#963)
- Add option to use templates from Zip files or Zip URLs, thanks to @freakboy3742 (#961)

Bug Fixes:

- Fix an issue with missing default template abbreviations for when a user defined custom abbreviations, thanks to @noirbizarre for the issue report and @hackebrot for the fix (#966, #967)
- Preserve existing output directory on project generation failure, thanks to @ionelmc for the report and @michaeljoseph for the fix (#629, #964)
- Fix Python 3.x error handling for `git` operation failures, thanks to @jmcarp (#905)

Other Changes:

- Fix broken link to *Copy without Render* docs, thanks to @coreysnyder04 (#912)
- Improve debug log message for when a hook is not found, thanks to @raphigaziano (#160)
- Fix module summary and `expand_abbreviations()` doc string as per pep257, thanks to @terryjbates (#772)
- Update doc strings in `cookiecutter/cli.py` and `cookiecutter/config.py` according to pep257, thanks to @terryjbates (#922, #931)
- Update doc string for `is_copy_only_path()` according to pep257, thanks to @mathagician and @terryjbates (#935, #949)
- Update doc strings in `cookiecutter/extensions.py` according to pep257, thanks to @meahow (#996)
- Fix miscellaneous issues with building docs, thanks to @stevepiercy (#889)
- Re-implement Makefile and update several make rules, thanks to @hackebrot (#930)
- Fix broken link to pytest docs, thanks to @eyalev for the issue report and @devstrat for the fix (#939, #940)
- Add `test_requirements.txt` file for easier testing outside of tox, thanks to @ramnes (#945)
- Improve wording in *copy without render* docs, thanks to @eyalev (#938)
- Fix a number of typos, thanks to @delirious-lettuce (#968)
- Improved *extra context* docs by noting that extra context keys must be present in the template's `cookiecutter.json`, thanks to @karantan for the report and fix (#863, #864)

- Added more cookiecutter templates to the mix:
  - cookiecutter-kata-cpputest by @13coders (#901)
  - cookiecutter-kata-gtest by @13coders (#901)
  - cookiecutter-pyramid-talk-python-starter by @mikeckennedy (#915)
  - cookiecutter-android by @alexfu (#890)
  - cookiecutter-lux-python by @alexkey (#895)
  - cookiecutter-git by @tuxredux (#921)
  - cookiecutter-ansible-role-ci by @ferrarimarco (#903)
  - cookiecutter_dotfile by @bdcaf (#925)
  - painless-continuous-delivery by @painless-software (#927)
  - cookiecutter-molecule by @retr0h (#954)
  - sublime-snippet-package-template by @agenoria (#956)
  - cookiecutter-conda-python by @conda (#969)
  - cookiecutter-flask-minimal by @candidtim (#977)
  - cookiecutter-pypackage-rust-cross-platform-publish by @mckaymatt (#957)
  - cookie-cookie by @tuxredux (#951)
  - cookiecutter-telegram-bot by @Ars2014 (#984)
  - python-project-template by @Kwpolska (#986)
  - wemake-django-template by @wemake-services (#990)
  - cookiecutter-raml by @genzj (#994)
  - cookiecutter-anyblok-project by @AnyBlok (#988)
  - cookiecutter-devenv by @greenguavalabs (#991)

3.3.10 1.5.1 (2017-02-04) Alfajor

New Features:

- Major update to installation documentation, thanks to @stevepiercy (#880)

Bug Fixes:

- Resolve an issue around default values for dict variables, thanks to @e-kolpakov for raising the issue and @hackebrot for the PR (#882, #884)

Other Changes:

- Contributor documentation reST fixes, thanks to @stevepiercy (#878)
- Added more cookiecutter templates to the mix:
  - widget-cookiecutter by @willingc (#781)
  - cookiecutter-django-foundation by @Parbhat (#804)
  - cookiecutter-tornado by @hkage (#807)
  - cookiecutter-django-ansible by @Ivaylo-Bachvarov(#816)

- CICADA by @elenimijalis (#840)
- cookiecutter-tf-module by @VDuda (#843)
- cookiecutter-pyqt4 by @aeroaks (#847)
- cookiecutter-golang by @mjhea0 and @lacion (#872, #873)
- cookiecutter-elm, cookiecutter-java and cookiecutter-spring-boot by @m-x-k (#879)

3.3.11 1.5.0 (2016-12-18) Alfajor

The primary goal of this release was to add command-line support for passing extra context, address minor bugs and make a number of improvements.

New Features:

- Inject extra context with command-line arguments, thanks to @msabramo and @michaeljoseph (#666).
- Updated conda installation instructions to work with the new conda-forge distribution of Cookiecutter, thanks to @pydanny and especially @bollwyvl (#232, #705).
- Refactor code responsible for interaction with version control systems and raise better error messages, thanks to @michaeljoseph (#778).
- Add support for executing cookiecutter using `python -m cookiecutter` or from a checkout/zip file, thanks to @brettcannon (#788).
- New CLI option `--debug-file PATH` to store a log file on disk. By default no log file is written. Entries for `DEBUG` level and higher. Thanks to @hackebrot(#792).
- Existing templates in a user's `cookiecutters_dir` (default is `~/.cookiecutters/`) can now be referenced by directory name, thanks to @michaeljoseph (#825).
- Add support for dict values in `cookiecutter.json`, thanks to @freakboy3742 and @hackebrot (#815, #858).
- Add a `jsonify` filter to default jinja2 extensions that json.dumps a Python object into a string, thanks to @aroig (#791).

Bug Fixes:

- Fix typo in the error logging text for when a hook did not exit successfully, thanks to @luzfcb (#656)
- Fix an issue around **replay** file names when **cookiecutter** is used with a relative path to a template, thanks to @eliasdorneles for raising the issue and @hackebrot for the PR (#752, #753)
- Ignore hook files with tilde-suffixes, thanks to @hackebrot (#768)
- Fix a minor issue with the code that generates a name for a template, thanks to @hackebrot(#798)
- Handle empty hook file or other OS errors, thanks to @christianmlong for raising this bug and @jcarbaugh and @hackebrot for the fix (#632, #729, #862)
- Resolve an issue with custom extensions not being loaded for `pre_gen_project` and `post_gen_project` hooks, thanks to @cheungnj (#860)

Other Changes:

- Remove external dependencies from tests, so that tests can be run w/o network connection, thanks to @hackebrot (#603)
- Remove execute permissions on Python files, thanks to @mozillazg (#650)
- Report code coverage info from AppVeyor build to codecov, thanks to @ewjoachim (#670)
- Documented functions and methods lacking documentation, thanks to @pydanny (#673)

- Documented `__init__` methods for Environment objects, thanks to @pydanny (#677)
- Updated whichcraft to 0.4.0, thanks to @pydanny.
- Updated documentation link to Read the Docs, thanks to @natim (#687)
- Moved cookiecutter templates and added category links, thanks to @willingc (#674)
- Added Github Issue Template, thanks to @luzfcb (#700)
- Added `ssh` repository examples, thanks to @pokoli (#702)
- Fix links to the cookiecutter-data-science template and its documentation, thanks to @tephyr for the PR and @willingc for the review (#711, #714)
- Update link to docs for Django's `--template` command line option, thanks to @purplediane (#754)
- Create *hook backup files* during the tests as opposed to having them as static files in the repository, thanks to @hackebrot (#789)
- Applied PEP 257 docstring conventions to:
  - `environment.py`, thanks to @terryjbates (#759)
  - `find.py`, thanks to @terryjbates (#761)
  - `generate.py`, thanks to @terryjbates (#764)
  - `hooks.py`, thanks to @terryjbates (#766)
  - `repository.py`, thanks to @terryjbates (#833)
  - `vcs.py`, thanks to @terryjbates (#831)
- Fix link to the Tryton cookiecutter, thanks to @cedk and @nicoe (#697, #698)
- Added PyCon US 2016 sponsorship to README, thanks to @purplediane (#720)
- Added a sprint contributor doc, thanks to @phoebebauer (#727)
- Converted readthedocs links (.org -> .io), thanks to @adamchainz (#718)
- Added Python 3.6 support, thanks to @suledev (#728)
- Update occurrences of `repo_name` in documentation, thanks to @palmerev (#734)
- Added case studies document, thanks to @pydanny (#735)
- Added first steps cookiecutter creation tutorial, thanks to @BruceEckel (#736)
- Reorganised tutorials and setup git submodule to external tutorial, thanks to @dot2dotseurat (#740)
- Debian installation instructions, thanks to @ivanlyon (#738)
- Usage documentation typo fix., thanks to @terryjbates (#739)
- Updated documentation copyright date, thanks to @zzzirk (#747)
- Add a make rule to update git submodules, thanks to @hackebrot (#746)
- Split up advanced usage docs, thanks to @zzzirk (#749)
- Documentation for the `no_input` option, thanks to @pokoli (#701)
- Remove unnecessary shebangs from python files, thanks to @michaeljoseph (#763)
- Refactor cookiecutter template identification, thanks to @michaeljoseph (#777)
- Add a `cli_runner` test fixture to simplify CLI tests, thanks to @hackebrot (#790)
- Add a check to ensure cookiecutter repositories have JSON context, thanks to @michaeljoseph(#782)

- Rename the internal function that determines whether a file should be rendered, thanks to @audreyfeldroy for raising the issue and @hackebrotfor the PR (#741, #802)
- Fix typo in docs, thanks to @mwarkentin (#828)
- Fix broken link to *Invoke* docs, thanks to @B3QL (#820)
- Add documentation to `render_variable` function in `prompt.py`, thanks to @pydanny (#678)
- Fix python3.6 travis-ci and tox configuration, thanks to @luzfcb (#844)
- Add missing encoding declarations to python files, thanks to @andytom (#852)
- Disable poyo logging for tests, thanks to @hackebrot (#855)
- Remove pycache directories in make clean-pyc, thanks to @hackebrot (#849)
- Refactor hook system to only find the requested hook, thanks to @michaeljoseph (#834)
- Add tests for custom extensions in `pre_gen_project` and `post_gen_project` hooks, thanks to @hackebrot (#856)
- Make the build reproducible by avoiding nondeterministic keyword arguments, thanks to @lamby and @hackebrot (#800, #861)
- Extend CLI help message and point users to the github project to engage with the community, thanks to @hackebrot (#859)
- Added more cookiecutter templates to the mix:
  - cookiecutter-funkload-friendly by @tokibito (#657)
  - cookiecutter-reveal.js by @keimlink (#660)
  - cookiecutter-python-app by @mdklatt (#659)
  - morepath-cookiecutter by @href (#672)
  - hovercraft-slides by @jhermann (#665)
  - cookiecutter-es6-package by @ratson (#667)
  - cookiecutter-webpack by @hzdg (#668)
  - cookiecutter-django-herokuapp by @dulaccc (#374)
  - cookiecutter-django-aws-eb by @peterlauri (#626)
  - wagtail-starter-kit by @tkjone (#658)
  - cookiecutter-dpf-effect by @SpotlightKid (#663)
  - cookiecutter-dpf-audiotk by @SpotlightKid (#663)
  - cookiecutter-template by @eviweb (#664)
  - cookiecutter-angular2 by @matheuspoleza (#675)
  - cookiecutter-data-science by @pjbull (#680)
  - cc_django_ember_app by @nanuxbe (#686)
  - cc_project_app_drf by @nanuxbe (#686)
  - cc_project_app_full_with_hooks by @nanuxbe (#686)
  - beat-generator by @ruflin (#695)
  - cookiecutter-scala by @Plippe (#751)

- cookiecutter-snakemake-analysis-pipeline by @xguse (#692)
- cookiecutter-py3tkinter by @ivanlyon (#730)
- pyramid-cookiecutter-alchemy by @stevepiercy (#745)
- pyramid-cookiecutter-starter by @stevepiercy (#745)
- pyramid-cookiecutter-zodb by @stevepiercy (#745)
- substanced-cookiecutter by @stevepiercy (#745)
- cookiecutter-simple-django-cn by @shenyushun (#765)
- cookiecutter-pyqt5 by @mandeepbhutani (#797)
- cookiecutter-xontrib by @laerus (#817)
- cookiecutter-reproducible-science by @mkrapp (#826)
- cc-automated-drf-template by @elenimijalis (#832)

3.3.12 1.4.0 (2016-03-20) Shortbread

The goal of this release is changing to a strict Jinja2 environment, paving the way to more awesome in the future, as well as adding support for Jinja2 extensions.

New Features:

- Added support for Jinja2 extension support, thanks to @hackebrot (#617).
- Now raises an error if Cookiecutter tries to render a template that contains an undefined variable. Makes generation more robust and secure (#586). Work done by @hackebrot (#111, #586, #592)
- Uses strict Jinja2 env in prompt, thanks to @hackebrot (#598, #613)
- Switched from pyyaml/ruamel.yaml libraries that were problematic across platforms to the pure Python poyo library, thanks to @hackebrot (#557, #569, #621)
- User config values for `cookiecutters_dir` and `replay_dir` now support environment variable and user home expansion, thanks to @nfarrar for the suggestion and @hackebrot for the PR (#640,#642)
- Add jinja2-time as default extension for dates and times in templates via `{% now 'utc' %}`,thanks to @hackebrot (#653)

Bug Fixes:

- Provided way to define options that have no defaults, thanks to @johtso (#587, #588)
- Make sure that `replay.dump()` and `replay.load()` use the correct user config, thanks to @hackebrot (#590, #594)
- Added correct CA bundle for Git on Appveyor, thanks to @maiksensi (#599, #602)
- Open `HISTORY.rst` with `utf-8` encoding when reading the changelog, thanks to @0-wiz-0 for submitting the issue and @hackebrot for the fix (#638, #639)
- Fix repository indicators for privaterepository urls, thanks to @habnabit for the fix (#595) and @hackebrot for the tests (#655)

Other Changes:

- Set path before running tox, thanks to @maiksensi (#615, #620)
- Removed xfail in test_cookiecutters, thanks to @hackebrot (#618)
- Removed django-cms-plugin on account of 404 error, thanks to @mativs and @pydanny (#593)

- Fixed docs/usage.rst, thanks to @macrotim (#604)
- Update .gitignore to latest Python.gitignore and ignore PyCharm files, thanks to @audreyfeldroy
- Use open context manager to read context_file in generate() function, thanks to @hackebrot (#607, #608)
- Added documentation for choice variables, thanks to @maiksensi (#611)
- Set up Scrutinizer to check code quality, thanks to @audreyfeldroy
- Drop distutils support in setup.py, thanks to @hackebrot (#606, #609)
- Change cookiecutter-pypackage-minimal link, thanks to @kragniz (#614)
- Fix typo in one of the template's description, thanks to @ryanfreckleton (#643)
- Fix broken link to _copy_without_render in *troubleshooting.rst*, thanks to @ptim (#647)
- Added more cookiecutter templates to the mix:
  - cookiecutter-pipproject by @wdm0006 (#624)
  - cookiecutter-flask-2 by @wdm0006 (#624)
  - cookiecutter-kotlin-gradle by @thomaslee (#622)
  - cookiecutter-tryton-fulfilio by @cedk (#631)
  - django-starter by @tkjone (#635)
  - django-docker-bootstrap by @legios89 (#636)
  - cookiecutter-mediawiki-extension by @JonasGroeger (#645)
  - cookiecutter-django-gulp by @valerymelou (#648)

3.3.13 1.3.0 (2015-11-10) Pumpkin Spice

The goal of this release is to extend the user config feature and to make hook execution more robust.

New Features:

- Abort project generation if `pre_gen_project` or `post_gen_project` hook scripts fail, thanks to @eliasdorneles (#464, #549)
- Extend user config capabilities with additional cli options `--config-file` and `--default-config` and environment variable `COOKIECUTTER_CONFIG`, thanks to @jhermann, @pfmoore, and @hackebrot (#258, #424, #565)

Bug Fixes:

- Fixed conditional dependencies for wheels in setup.py, thanks to @hackebrot (#557, #568)
- Reverted skipif markers to use correct reasons (bug fixed in pytest), thanks to @hackebrot (#574)

Other Changes:

- Improved path and documentation for rendering the Sphinx documentation, thanks to @eliasdorneles and @hackebrot (#562, #583)
- Added additional help entrypoints, thanks to @michaeljoseph (#563, #492)
- Added Two Scoops Academy to the README, thanks to @hackebrot (#576)
- Now handling trailing slash on URL, thanks to @ramiroluz (#573, #546)
- Support for testing x86 and x86-64 architectures on appveyor, thanks to @maiksensi (#567)

- Made tests work without installing Cookiecutter, thanks to @vincentbernat (#550)
- Encoded the result of the hook template to utf8, thanks to @ionelmc (#577. #578)
- Added test for _run_hook_from_repo_dir, thanks to @hackebrot (#579, #580)
- Implemented bumpversion, thanks to @hackebrot (#582)
- Added more cookiecutter templates to the mix:
  - cookiecutter-octoprint-plugin by @foosel (#560)
  - wagtail-cookiecutter-foundation by @chrisdev, et al. (#566)

3.3.14 1.2.1 (2015-10-18) Zimtsterne

Zimtsterne are cinnamon star cookies.

New Feature:

- Returns rendered project dir, thanks to @hackebrot (#553)

Bug Fixes:

- Factor in *choice* variables (as introduced in 1.1.0) when using a user config or extra context, thanks to @ionelmc and @hackebrot (#536, #542).

Other Changes:

- Enable py35 support on Travis by using Python 3.5 as base Python (@maiksensi / #540)
- If a filename is empty, do not generate. Log instead (@iljabauer / #444)
- Fix tests as per last changes in cookiecutter-pypackage, thanks to @eliasdorneles(#555).
- Removed deprecated cookiecutter-pylibrary-minimal from the list, thanks to @ionelmc (#556)
- Moved to using rualmel.yaml instead of PyYAML, except for Windows users on Python 2.7, thanks to @pydanny (#557)

Why 1.2.1 instead of 1.2.0? There was a problem in the distribution that we pushed to PyPI. Since you can't replace previous files uploaded to PyPI, we deleted the files on PyPI and released 1.2.1.

3.3.15 1.1.0 (2015-09-26) Snickerdoodle

The goals of this release were `copy without render` and a few additional command-line options such as `--overwrite-if-exists`, `---replay`, and `output-dir`.

Features:

- Added copy without render feature, making it much easier for developers of Ansible, Salt Stack, and other recipe-based tools to work with Cookiecutter. Thanks to @osantana and @LucianU for their innovation, as well as @hackebrot for fixing the Windows problems (#132, #184, #425).
- Added specify output directory, thanks to @tony and @hackebrot (#531, #452).
- Abort template rendering if the project output directory already exists, thanks to @lgp171188 (#470, #471).
- Add a flag to overwrite existing output directory, thanks to @lgp171188 for the implementation (#495) and @schacki, @ionelmc, @pydanny and @hackebrot for submitting issues and code reviews (#475, #493).
- Remove test command in favor of tox, thanks to @hackebrot (#480).

- Allow cookiecutter invocation, even without installing it, via `python -m cookiecutter.cli`, thanks to @vincentbernat and @hackebrot (#449, #487).
- Improve the type detection handler for online and offline repositories, thanks to @charlax (#490).
- Add replay feature, thanks to @hackebrot (#501).
- Be more precise when raising an error for an invalid user config file, thanks to @vaab and @hackebrot (#378, #528).
- Added official Python 3.5 support, thanks to @pydanny and @hackebrot (#522).
- Added support for *choice* variables and switch to click style prompts, thanks to @hackebrot (#441, #455).

Other Changes:

- Updated click requirement to < 6.0, thanks to @pydanny (#473).
- Added landscape.io flair, thanks to @michaeljoseph (#439).
- Descriptions of PEP8 specifications and milestone management, thanks to @michaeljoseph (#440).
  - Added alternate installation options in the documentation, thanks to @pydanny (#117, #315).
- The test of the which() function now tests against the date command, thanks to @vincentbernat (#446)
- Ensure file handles in setup.py are closed using with statement, thanks to @svisser (#280).
- Removed deprecated and fully extraneous compat.is_exe() function, thanks to @hackebrot (#485).
- Disabled sudo in .travis, thanks to @hackebrot (#482).
- Switched to shields.io for problematic badges, thanks to @pydanny (#491).
- Added whichcraft and removed `compat.which()`, thanks to @pydanny (#511).
- Changed to export tox environment variables to codecov, thanks to @maiksensi. (#508).
- Moved to using click version command, thanks to @hackebrot (#489).
- Don't use unicode_literals to please click, thanks to @vincentbernat (#503).
- Remove warning for Python 2.6 from __init__.py, thanks to @hackebrot.
- Removed compat.py module, thanks to @hackebrot.
- Added future to requirements, thanks to @hackebrot.
- Fixed problem where expanduser does not resolve "~" correctly on windows 10 using tox, thanks to @maiksensi. (#527)
- Added more cookiecutter templates to the mix:
  - cookiecutter-beamer by @luismartingil (#307)
  - cookiecutter-pytest-plugin by @pytest-dev and @hackebrot (#481)
  - cookiecutter-csharp-objc-binding by @SandyChapman (#460)
  - cookiecutter-flask-foundation by @JackStouffer (#457)
  - cookiecutter-tryton-fulfilio by @fulfilio (#465)
  - cookiecutter-tapioca by @vintasoftware (#496)
  - cookiecutter-sublime-text-3-plugin by @kkujawinski (#500)
  - cookiecutter-muffin by @drgarcia1986 (#494)
  - cookiecutter-django-rest by @agconti (#520)

  - cookiecutter-es6-boilerplate by @agconti (#521)
  - cookiecutter-tampermonkey by @christabor (#516)
  - cookiecutter-wagtail by @torchbox (#533)

3.3.16 1.0.0 (2015-03-13) Chocolate Chip

The goals of this release was to formally remove support for Python 2.6 and continue the move to using py.test.

Features:

- Convert the unittest suite to py.test for the sake of comprehensibility, thanks to @hackebrot (#322, #332, #334, #336, #337, #338, #340, #341, #343, #345, #347, #351, #412, #413, #414).
- Generate pytest coverage, thanks to @michaeljoseph (#326).
- Documenting of Pull Request merging and HISTORY.rst maintenance, thanks to @michaeljoseph (#330).
- Large expansions to the tutorials thanks to @hackebrot (#384)
- Switch to using Click for command-line options, thanks to @michaeljoseph (#391, #393).
- Added support for working with private repos, thanks to @marctc (#265).
- Wheel configuration thanks to @michaeljoseph (#118).

Other Changes:

- Formally removed support for 2.6, thanks to @pydanny (#201).
- Moved to codecov for continuous integration test coverage and badges, thanks to @michaeljoseph (#71, #369).
- Made JSON parsing errors easier to debug, thanks to @rsyring and @mark0978 (#355, #358, #388).
- Updated to Jinja 2.7 or higher in order to control trailing new lines in templates, thanks to @sfermigier (#356).
- Tweaked flake8 to ignore e731, thanks to @michaeljoseph (#390).
- Fixed failing Windows tests and corrected AppVeyor badge link thanks to @msabramo (#403).
- Added more Cookiecutters to the list:
  - cookiecutter-scala-spark by @jpzk
  - cookiecutter-atari2600 by @joeyjoejoejr
  - cookiecutter-bottle by @avelino
  - cookiecutter-latex-article by @Kreger51
  - cookiecutter-django-rest-framework by @jpadilla
  - cookiedozer by @hackebrot

3.3.17 0.9.0 (2015-01-13)

The goals of this release were to add the ability to Jinja2ify the cookiecutter.json default values, and formally launch support for Python 3.4.

Features:

- Python 3.4 is now a first class citizen, thanks to everyone.
- cookiecutter.json values are now rendered Jinja2 templates, thanks to @bollwyvl (#291).
- Move to py.test, thanks to @pfmoore (#319) and @ramiroluz (#310).
- Add PendingDeprecation warning for users of Python 2.6, as support for it is gone in Python 2.7, thanks to @michaeljoseph (#201).

Bug Fixes:

- Corrected typo in Makefile, thanks to @inglesp (#297).
- Raise an exception when users don't have git or hg installed, thanks to @pydanny (#303).

Other changes:

- Creation of gitter account for logged chat, thanks to @michaeljoseph.
- Added ReadTheDocs badge, thanks to @michaeljoseph.
- Added AppVeyor badge, thanks to @pydanny
- Documentation and PyPI trove classifier updates, thanks to @thedrow (#323 and #324)

3.3.18 0.8.0 (2014-10-30)

The goal of this release was to allow for injection of extra context via the Cookiecutter API, and to fix minor bugs.

Features:

- cookiecutter() now takes an optional extra_context parameter, thanks to @michaeljoseph, @fcurella, @aventurella, @emonty, @schacki, @ryanolson, @pfmoore, @pydanny, @audreyfeldroy (#260).
- Context is now injected into hooks, thanks to @michaeljoseph and @dinopetrone.
- Moved all Python 2/3 compatibility code into cookiecutter.compat, making the eventual move to six easier, thanks to @michaeljoseph (#60, #102).
- Added cookiecutterrc defined aliases for cookiecutters, thanks to @pfmoore (#246)
- Added flake8 to tox to check for pep8 violations, thanks to @natim.

Bug Fixes:

- Newlines at the end of files are no longer stripped, thanks to @treyhunner (#183).
- Cloning prompt suppressed by respecting the `no\_input` flag, thanks to @trustrachel (#285)
- With Python 3, input is no longer converted to bytes, thanks to @uranusjr (#98).

Other Changes:

- Added more Cookiecutters to the list:
  - Python-iOS-template by @freakboy3742
  - Python-Android-template by @freakboy3742
  - cookiecutter-djangocms-plugin by @mishbahr

- cookiecutter-pyvanguard by @robinandeer

3.3.19 0.7.2 (2014-08-05)

The goal of this release was to fix cross-platform compatibility, primarily Windows bugs that had crept in during the addition of new features. As of this release, Windows is a first-class citizen again, now complete with continuous integration.

Bug Fixes:

- Fixed the contributing file so it displays nicely in Github, thanks to @pydanny.
- Updates 2.6 requirements to include simplejson, thanks to @saxix.
- Avoid unwanted extra spaces in string literal, thanks to @merwok.
- Fix @unittest.skipIf error on Python 2.6.
- Let sphinx parse :param: properly by inserting newlines #213, thanks to @mineo.
- Fixed Windows test prompt failure by replacing stdin per @cjrh in #195.
- Made rmtree remove readonly files, thanks to @pfmoore.
- Now using tox to run tests on Appveyor, thanks to @pfmoore (#241).
- Fixed tests that assumed the system encoding was utf-8, thanks to @pfmoore (#242, #244).
- Added a tox ini file that uses py.test, thanks to @pfmoore (#245).

Other Changes:

- @audreyfeldroy formally accepted position as **BDFL of cookiecutter**.
- Elevated @pydanny, @michaeljoseph, and @pfmoore to core committer status.
- Added Core Committer guide, by @audreyfeldroy.
- Generated apidocs from make docs, by @audreyfeldroy.
- Added contributing command to the makedocs function, by @pydanny.
- Refactored contributing documentation, included adding core committer instructions, by @pydanny and @audreyfeldroy.
- Do not convert input prompt to bytes, thanks to @uranusjr (#192).
- Added troubleshooting info about Python 3.3 tests and tox.
- Added documentation about command line arguments, thanks to @saxix.
- Style cleanups.
- Added environment variable to disable network tests for environments without networking, thanks to @vincentbernat.
- Added Appveyor support to aid Windows integrations, thanks to @pydanny (#215).
- CONTRIBUTING.rst is now generated via make contributing, thanks to @pydanny (#220).
- Removed unnecessary endoing argument to json.load, thanks to @pfmoore (#234).
- Now generating shell hooks dynamically for Unix/Windows portability, thanks to @pfmoore (#236).
- Removed non-portable assumptions about directory structure, thanks to @pfmoore (#238).
- Added a note on portability to the hooks documentation, thanks to @pfmoore (#239).

- Replaced unicode_open with direct use of io.open, thanks to @pfmoore (#229).
- Added more Cookiecutters to the list:
  - cookiecutter-kivy by @hackebrot
  - BoilerplatePP by @Paspartout
  - cookiecutter-pypackage-minimal by @borntyping
  - cookiecutter-ansible-role by @iknite
  - cookiecutter-pylibrary by @ionelmc
  - cookiecutter-pylibrary-minimal by @ionelmc

3.3.20 0.7.1 (2014-04-26)

Bug fixes:

- Use the current Python interpreter to run Python hooks, thanks to @coderanger.
- Include tests and documentation in source distribution, thanks to @vincentbernat.
- Fix various warnings and missing things in the docs (#129, #130), thanks to @nedbat.
- Add command line option to get version (#89), thanks to @davedash and @cyberj.

Other changes:

- Add more Cookiecutters to the list:
  - cookiecutter-avr by @solarnz
  - cookiecutter-tumblr-theme by @relekang
  - cookiecutter-django-paas by @pbacterio

3.3.21 0.7.0 (2013-11-09)

This is a release with significant improvements and changes. Please read through this list before you upgrade.

New features:

- Support for –checkout argument, thanks to @foobacca.
- Support for pre-generate and post-generate hooks, thanks to @raphigaziano. Hooks are Python or shell scripts that run before and/or after your project is generated.
- Support for absolute paths to cookiecutters, thanks to @krallin.
- Support for Mercurial version control system, thanks to @pokoli.
- When a cookiecutter contains invalid Jinja2 syntax, you get a better message that shows the location of the TemplateSyntaxError. Thanks to @benjixx.
- Can now prompt the user to enter values during generation from a local cookiecutter, thanks to @ThomasChiroux. This is now always the default behavior. Prompts can also be suppressed with `--no-input`.
- Your cloned cookiecutters are stored by default in your ~/.cookiecutters/ directory (or Windows equivalent). The location is configurable. (This is a major change from the pre-0.7.0 behavior, where cloned cookiecutters were deleted at the end of project generation.) Thanks @raphigaziano.
- User config in a ~/.cookiecutterrc file, thanks to @raphigaziano. Configurable settings are cookiecutters_dir and default_context.

- File permissions are now preserved during project generation, thanks to @benjixx.

Bug fixes:

- Unicode issues with prompts and answers are fixed, thanks to @s-m-i-t-a.
- The test suite now runs on Windows, which was a major effort. Thanks to @pydanny, who collaborated on this with me.

Other changes:

- Quite a bit of refactoring and API changes.
- Lots of documentation improvements. Thanks @sloria, @alex, @pydanny, @freakboy3742, @es128, @rolo.
- Better naming and organization of test suite.
- A CookiecutterCleanSystemTestCase to use for unit tests affected by the user's config and cookiecutters directory.
- Improvements to the project's Makefile.
- Improvements to tests. Thanks @gperetin, @s-m-i-t-a.
- Removal of subprocess32 dependency. Now using non-context manager version of subprocess.Popen for Python 2 compatibility.
- Removal of cookiecutter's cleanup module.
- A bit of setup.py cleanup, thanks to @oubiga.
- Now depends on binaryornot 0.2.0.

3.3.22 0.6.4 (2013-08-21)

- Windows support officially added.
- Fix TemplateNotFound Exception on Windows (#37).

3.3.23 0.6.3 (2013-08-20)

- Fix copying of binary files in nested paths (#41), thanks to @sloria.

3.3.24 0.6.2 (2013-08-19)

- Depend on Jinja2>=2.4 instead of Jinja2==2.7.
- Fix errors on attempt to render binary files. Copy them over from the project template without rendering.
- Fix Python 2.6/2.7 UnicodeDecodeError when values containing Unicode chars are in cookiecutter.json.
- Set encoding in Python 3 unicode_open() to always be utf-8.

3.3.25 0.6.1 (2013-08-12)

- Improved project template finding. Now looks for the occurrence of {{,cookiecutter, and }} in a directory name.
- Fix help message for input_dir arg at command prompt.
- Minor edge cases found and corrected, as a result of improved test coverage.

3.3.26 0.6.0 (2013-08-08)

- Config is now in a single `cookiecutter.json` instead of in `json/`.
- When you create a project from a git repo template, Cookiecutter prompts you to enter custom values for the fields defined in `cookiecutter.json`.

3.3.27 0.5 (2013-07-28)

- Friendlier, more simplified command line usage:

```
# Create project from the cookiecutter-pypackage/ template
$ cookiecutter cookiecutter-pypackage/
# Create project from the cookiecutter-pypackage.git repo template
$ cookiecutter https://github.com/audreyfeldroy/cookiecutter-pypackage.git
```

- Can now use Cookiecutter from Python as a package:

```
from cookiecutter.main import cookiecutter

# Create project from the cookiecutter-pypackage/ template
cookiecutter('cookiecutter-pypackage/')

# Create project from the cookiecutter-pypackage.git repo template
cookiecutter('https://github.com/audreyfeldroy/cookiecutter-pypackage.git')
```

- Internal refactor to remove any code that changes the working directory.

3.3.28 0.4 (2013-07-22)

- Only takes in one argument now: the input directory. The output directory is generated by rendering the name of the input directory.
- Output directory cannot be the same as input directory.

3.3.29 0.3 (2013-07-17)

- Takes in command line args for the input and output directories.

3.3.30 0.2.1 (2013-07-17)

- Minor cleanup.

3.3.31 0.2 (2013-07-17)

Bumped to "Development Status :: 3 - Alpha".

- Works with any type of text file.
- Directory names and filenames can be templated.

3.3.32 0.1.0 (2013-07-11)

- First release on PyPI.

3.4 Case Studies

This showcase is where organizations can describe how they are using Cookiecutter.

3.4.1 BeeWare

Building Python tools for platforms like mobile phones and set top boxes requires a lot of boilerplate code just to get the project running. Cookiecutter has enabled us to very quickly stub out a starter project in which running Python code can be placed, and makes maintaining those templates very easy. With Cookiecutter we've been able to deliver support Android devices, iOS devices, tvOS boxes, and we're planning to add native support for iOS and Windows devices in the future.

BeeWare is an organization building open source libraries for Python support on all platforms.

3.4.2 ChrisDev

Anytime we start a new project we begin with a Cookiecutter template that generates a Django/Wagtail project Our developers like it for maintainability and our designers enjoy being able to spin up new sites using our tool chain very quickly. Cookiecutter is very useful for because it supports both Mac OSX and Windows users.

ChrisDev is a Trinidad-based consulting agency.

3.4.3 OpenStack

OpenStack uses several Cookiecutter templates to generate:

- Openstack compliant puppet-modules
- Install guides
- New tempest plugins

OpenStack is open source software for creating private and public clouds.

3.5 Code of Conduct

Everyone interacting in the Cookiecutter project's codebases and documentation is expected to follow the PyPA Code of Conduct. This includes, but is not limited to, issue trackers, chat rooms, mailing lists, and other virtual or in real life communication.

CHAPTER

FOUR

INDEX

- genindex
- modindex

PYTHON MODULE INDEX

C

INDEX

Symbols

A

C

D

E

F

G

I

J

L

M

N

O

P

R

S

T

U

V

W