

---

# **Carp Documentation**

***Release 0.0.3***

**W. Matthew Wilson**

July 24, 2013



# CONTENTS

|          |                                  |           |
|----------|----------------------------------|-----------|
| <b>1</b> | <b>Summary</b>                   | <b>1</b>  |
| <b>2</b> | <b>Isn't this a bad idea?</b>    | <b>3</b>  |
| <b>3</b> | <b>Installation</b>              | <b>5</b>  |
| <b>4</b> | <b>Usage</b>                     | <b>7</b>  |
| <b>5</b> | <b>Templates made of folders</b> | <b>9</b>  |
| <b>6</b> | <b>Future plans</b>              | <b>11</b> |
| <b>7</b> | <b>Boring stuff to do</b>        | <b>13</b> |



# SUMMARY

Carp cuts down on typing by letting you store code templates (using `jinja2`) and then render them.

I named this project carp because this project will save me from CARP-al tunnel. Harharhar.



# ISN'T THIS A BAD IDEA?

Yes. Terrible idea. Good programmers use libraries and inheritance and dynamic code generation and run-time message passing to shrink redundancy.

Shitty programmers use copy-paste. And only the worst programmer imaginable would ever design and use a copy-paste framework.

This project is an abomination.

Hurry back to [hacker news](#)!





# INSTALLATION

You can use `easy_install`:

```
$ easy_install carp
```

Or you can get the source from github [here](#):

```
$ git clone https://github.com/mw44118/carp.git
$ cd carp
$ python setup.py install
```



# USAGE

The first thing to do is make a directory named “carpdir” at the top of your project:

```
$ cd ~/src/my-awesome-project
$ mkdir carpdir
```

That’s where carp will store templates.

Now you can ask carp what templates are available. Since you haven’t stored any templates yet, you won’t get anything back though:

```
$ carp-list
no carpdir found!
```

```
$ mkdir carpdir
```

```
$ carp-list
```

Next, pick a file that you want to use as a template and copy it to some out-of-the-way place:

```
$ cp script.py /tmp/base_script.py
```

Now edit that copy and replace the text that you want to be passed in as a parameter, by using double-curly braces, like this:

```
{{project_name}}
```

Actually, you can use any [jinja2](#) tricks, because the files will be run as [jinja2](#) templates. For example, you can use a filter to make sure something comes out as lower case like this:

```
{{ project_name | lower }}
```

Once you have replaced the text that you want to use as parameters, add this template like this:

```
$ carp-add /tmp/base_script.py
```

You can get a list of required variables to pass in:

```
$ carp-info base_script.py
base_script.py required variables

*   project_name
```

That tells you that you have to pass in a `project_name` variable when you render the `base_script.py` template.

And now you can render that template:

```
$ carp-render base_script.py --define project_name=bogus
```

That will write a bunch of stuff to standard output, so use redirect to put it in a file:

```
$ carp-render base_script.py --define project_name=bogus > bogus_script.py
```

# TEMPLATES MADE OF FOLDERS

You can have single-file templates, but you can also store trees of folders and files as templates too.

Imagine you have a folder `proj` with a bunch of stuff inside and you want it to be a template:

```
$ find proj
proj/
proj/setup.py
proj/src
proj/src/proj
proj/src/proj/__init__.py
```

You add folder templates just the same:

```
$ carp-add proj
```

And `carp-info` works the same:

```
$ carp-info proj
proj required variables

* projname
```

But when you render a template that is a folder, you have to give me a folder to put it in, like this:

```
$ carp-render proj /tmp --define projname=scratch
```

Now you'll see a folder `/tmp/proj` with all your stuff inside:

```
$ find /tmp/proj
/tmp/proj
/tmp/proj/setup.py
/tmp/proj/src
/tmp/proj/src/proj
/tmp/proj/src/proj/__init__.py
```



# FUTURE PLANS

- Include a few built-in templates when you install carp.
- Add descriptions and other metadata to templates.
- Support command-line tab completion for template names.





## BORING STUFF TO DO

- Add a bootstrap + backbone.js template to the list of built-in templates.
- Add a `--version` command-line option to all the scripts.
- Pretty-up the error messaging when somebody forgets to define a variable during `carp-render`. Right now, it just blows up.
- Move dependencies out from `setup.py` into `requirements.txt`, because that's how the world is going, apparently.
- Show docstrings from modules in docs.