HelCLI Documentation

Release 0.1.0

Alex Edwards

November 11, 2016

Contents

CHAPTER 1

Getting Started

This guide will walk you through setting up an example CLI project using HelCLI.

1.1 Project Layout

```
project-src/
    project/
        __init__.py
        commands/
            __init__.py
            example-command.py
    setup.py
```

1.1.1 project/__init__.py

This is where most of core CLI code will go, we need to instantiate the Class and run it inside of a function. When we go to reference this with an entry point in setup.py we'll need to remember the name we use. We will call it with the `main` function as an example. Our entry point will then be `project:main`.

```
from helcli import HelCLI


def main():
    cli = HelCLI(sub_commands='commands', description='A simple CLI')
    cli.run()
```

Great! Now when project.main() is called it will execute our CLI. Take note that sub_commands is set to 'commands' this is the directory relative to our package (project) where our sub-commands for this CLI will go.

1.1.2 project/commands/__init__.py

Python requires this file to know that the commands directory is a submodule that we can import from. It's fine to leave this file blank, it just needs to exist.

1.1.3 project/commands/example-command.py

HelCLI will be looking for commands inside of the 'commands' directory which we set above. It will go through every file in this directory when the `run` method is called on the HelCLI object and execute `setup`. This is where

you should setup any command line arguments you expect for this command. Inside of the main method is what should happen if someone is actually calling this command. The arguments you requested above will be passed to `main` as a dictionary you can reference.

For this example let's have example-command take one argument, name, and have the command print out 'Hello `name`!'. Note that it uses the argparse module to set this up.

```
def setup(parser, subparsers):
    parser_example_command = subparsers.add_parser(
        'example-command',
        help='Prints "Hello <name>!"')

    parser_example_command.add_argument(
        'name',
        help='The name to say hello to.')

def main(parser_d):
    name = parser_d['name']  # retrieved from parser name argument
    print 'Hello {}!'.format(name)
```

1.1.4 setup.py

Lastly is our setup file. This syntax isn't specific to HelCLI, but is included for completeness.

```
from setuptools import setup, find_packages


setup(name='project',
      version=0.0.1,
      author='You',
      author_email='you@mail.com',
      packages=find_packages(),
      install_requires=['helcli'],
      entry_points={'console_scripts': [
          'project = project:main',
      ]})
```

1.2 Wrapping up

Now you may `pip install -e ./project-src` and run the `project` command to have a fully functional CLI.

CHAPTER 2

Additional Arguments

You will likely need to pass additional variables to your commands in order for them to run properly. An example would be a config object with all of your config options. This is accomplished by passing additional arguments to the HelCLI object when instantiated.

In practice it looks like this:

```
from helcli import HelCLI


var_one = "test1"
var_two = "test2"
cli = HelCLI(sub_commands='command_dir',
             description='A simple CLI',
             var_one, var_two)
cli.run()
```

Then when it comes time to use this variable in the `main` function of all our commands these are available as an additional variable.

```
def main(parser_d, additional_args):
    # unpack additional_args
    var_one, var_two = additional_args
    print(var_one)  # prints "test1"
    print(var_two)  # prints "test2"
```

Note: If you pass additional arguments to the HelCLI object you will need to have the `additional_args` variable passed in the `main()` function in *all* of your commands, or else you will get errors.

HelCLI is an opinionated way to create and organize your command line programs, and aims to help bootstrap the process and provide a foundation that works for most use cases.

In the most simplest form, HelCLI requires three lines of code to have a functional command line program.

```
from helcli import HelCLI

cli = HelCLI(sub_commands='command_dir', description='A simple CLI')
cli.run()
```

To see some examples of HelCLI in use see the example project, or Getting Started.