Vulcano Documentation

AttackIQ

Apr 30, 2020

Contents:

CHAPTER 1

Vulcano

Please, help us to continue improving, become a patreon

1.1 What is Vulcano?

Vulcano is a framework for creating command line utils.

Built on top of prompt_toolkit, it helps you to create human-friendly modern command line utils.

It's simplicity makes it suitable in a lot of scenarios where you just want to run already-created functions in a REPL/ARGS mode.

Note: Important notes Due some design changes we are working on, we recommend you to avoid using this framework on a production environment. We're still looking forward having a more idiomatic module name convention. If you're happy with the current state, just use it ;)

1.2 Key features

- *Autocomplete*: Vulcano will inspect all the functions you register, and will create a list of autocomplete with your command name and it's arguments.
- *Help*: It will create help based on your functions docstrings or the help provided during the registration process.
- *History*: Use up & down arrows to select a command from your history.
- *Register modules*: It can register all the functions inside a module just by calling the register module function. It will help you to prevent modifying the source module.

- *Lexer*: Of course, we use lexer with pygments to colorize your command line ;)
- *Concatenated commands*: You want to execute more than one command at once from the command line arguments? Just use the "and". `python your_script.py my_func arg=\"something\´´ and my_func_2 arg=\"another thing here\"` , such hacker!
- *Context*: If you want to communicate different functions between them, you can use the VulcanoApp.context (it's just a dictionary where you store and read data).
- *Command templating*: You can use whatever is on the context to format your command and generate it with data from the context.
- *Autosuggestion*: Whenever you enter a command that doesn't exists, you'll get the most similar command name. This improves the user experience. You could define your own function to determine which command you should use.

```
>> niu
Command niu not found
Did you mean: "new"?
>>
```

- *Inspect commands source code*: With vulcano, you can inspect a command sourcecode by just typing `?` at the end of the command. For example: `>> bye?` it will print this function source with syntax highlight.

```
>> bye?
@app.command
def bye(name="User"):
    """ Say goodbye to someone """
    return "Bye {}!".format(name)
>>
```

1.3 Installation

Vulcano is automatically delivered through TravisCI, which means that we usually keep the pip package up to date, this will help you to install the vulcano latest version by just executing the: `pip install vulcano`

But in case you're looking for installing a non-delivered version or just a custom branch, you can install it by cloning the repository and executing the: `python setup.py install`

Lets keep things simple.

1.4 Learn by example

The repository usually holds a simple sample ready to execute which brings an example of almost all the features.

In case you don't want to clone it, you can copy paste it:

```
from __future__ import print_function
import random
from vulcano.app import VulcanoApp
from vulcano.app.lexer import MonokaiTheme


app = VulcanoApp()

```

(continues on next page)

(continued from previous page)

```
@app.command("hi", "Salute people given form parameter")
def salute_method_here(name, title="Mr."):
    """Salute to someone

    :param str name: Name of who you want to say hi!
    :param str title: Title of this person
    """
    print("Hi! {} {} :) Glad to see you.".format(title, name))


def has_context_name():
    """Function to hide a command from command line

    This function is to prevent showing help and autocomplete for commands that need␣
↪the name
    to be set up on the context.
    """
    return 'name' in app.context


@app.command
def i_am(name):
    """Set your name

    :param str name: Your name goes here!
    """
    app.context['name'] = name


@app.command(show_if=has_context_name)
def whoami():
    """Returns your name from the context

    This is only shown where you've set your name
    """
    return app.context['name']


@app.command
def bye(name="User"):
    """ Say goodbye to someone """
    return "Bye {}!".format(name)


@app.command
def sum_numbers(*args):
    """ Sums all numbers passed as parameters """
    return sum(args)


@app.command
def multiply(number1, number2):
    """ Just multiply two numbers """
    return number1 * number2


@app.command
```

(continues on next page)

(continued from previous page)

```
def reverse_word(word):
    """ Reverse a word """
    return word[::-1]


@app.command
def random_upper_word(word):
    """ Returns the word with random upper letters """
    return "".join(random.choice([letter.upper(), letter]) for letter in word)


if __name__ == '__main__':
    app.run(theme=MonokaiTheme)
```

This will create next commands: - hi - bye - i_am - whoami - sum_numbers - multiply - reverse_word - random_upper_word

Those commands can `return` data that will be printed (if there's something) and the result will be stored inside the context under the `last_result` node. This helps you to be able to use it on the command line templating.

You can execute from `repl` mode:

```
$ python simple_example.py
>> reverse_word "Hello Baby! This is awesome"
emosewa si sihT !ybaB olleH
>> random_upper_word "{last_result}"
EMosEWa si SiHT !ybAB OlLEH
>> exit
```

And also can be executed from `args` mode:

```
$ python simple_example.py reverse_word \"Hello Baby! This is awesome\" and random_
↪upper_word \"{last_result}\"
emosewa si sihT !ybaB olleH
EMOSEWa Si siHT !YbAB olLeH
```

Nice, right?

1.5 Contribute

If you have an idea, you want to help improving something ... or whatever you think you can help, you're welcome.

All the pull requests will be checked (and also the bugs you report).

1.5.1 Source documentation

vulcano.core

vulcano.app

CHAPTER 2

Indices and tables

- genindex
- modindex
- search