Python Command Line Tools

Design powerful apps with Click

Noah Gift & Alfredo Deza

Python Command Line Tools

Design powerful apps with Click

Noah Gift and Alfredo Deza

This book is for sale at http://leanpub.com/pythoncli

This version was published on 2020-06-26

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 1: Getting Started with Click

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

What is wrong with the alternatives

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

A helpful Hello World

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Map a function to a command

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 2: Test with Click

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Test a small click app

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 3: Understand IPython

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Using IPython, Jupyter, Colab and Python executable

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

IPython

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Jupyter Notebook

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Hosted Commercial Flavors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Pure Open Source

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Hybrid Solutions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Colab Notebook Key Features

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Magic Commands

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

%timeit

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

%alias

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

%who

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

%writefile

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Bash

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Python2

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Python executable

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 4: Integrating Linux Commands with Click

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Understand subprocess

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Parsing Results

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Simple Parsing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Advanced Parsing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Shell Safety

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 5: Writing pure Bash or ZSH command-line tools

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Understanding environmental variables

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Understand shell profiles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Customizing your shell

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Write Shell functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 6: Use Advanced Click Features

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Subcommands

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Utilities

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Colored Output

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

File handling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 7: Turbocharging Click

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Using The Numba JIT (Just in time Compiler)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Running True Multi-Core Multithreaded Python using Numba

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 8: Integrate Click with the Cloud

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Cloud Developer Workflow

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Using Cloud-based development environments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

AWS Cloud9

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Build a Computer Vision Tool with AWS Boto3

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 9: Case Studies

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Distribute a containerized click application to DockerHub

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Getting started with Docker

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Docker Desktop Overview

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Docker Hub Overview

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Why Docker Containers vs. Virtual Machines?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Real-World Examples of Containers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Developer Shares Local Project

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Data Scientist shares Jupyter Notebook with a Researcher at another University

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

A Machine Learning Engineer Load Tests a Production Machine Learning Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Running Docker Containers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Using "base" images

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Common Issues Running a Docker Container

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Build containerized application from Zero on AWS Cloud9

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Screencast

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Converting a Command-line tool to a Web Service

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Documenting your Project with Sphinx

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythoncli.

Chapter 10: Command-line Rosetta Stone

This chapter serves as a guidebook for users coming from another language. The source code can be found here[1]. The same style, hello world command-line tool is written in many languages.

R Hello World

This step is an example of a hello world command-line tool in R. The source code is here[2].

```
#!/usr/bin/env Rscript
#Hello World R command-line tool
#



suppressPackageStartupMessages(library("optparse"))
parser <- OptionParser()
parser <- add_option(parser, c("-c", "--count"), type = "integer",
                     help = "Number of times to print phrase",
                     metavar = "number")
parser <- add_option(parser, c("-p", "--phrase"),
                    help = "Phrase to print")

args <- parse_args(parser)


# Function to Generate Phrases
phrasegen <- function(arguments){
    for (count in 1:arguments$count) {
        cat(paste(arguments$phrase, "\n"))
    }
}

#Run the program
phrasegen(args)
```

Depends on https://github.com/trevorld/optparse for R

[1] https://github.com/noahgift/cli-rosetta
[2] https://github.com/noahgift/cli-rosetta/tree/master/R/hello-world

Usage

```
$  hello-world git:(master) $ ./hello-world.R --count 5 --phrase "hello world"
hello world
hello world
hello world
hello world
hello world
```

Bash Hello World

This step is a hello world Bash example. The source code is here[3].

```
#!/bin/bash
#output looks like this:
#
#  $hello-world git:(master) $ ./hello-world.sh --count 5 --phrase "hello world"
#hello world
#hello world
#hello world
#hello world
#hello world

#Generate phrase "N" times
phrase_generator() {
    for ((i=0; i<$1;i++)); do
        echo "$2"
    done
}

#Parse Options
while [[ $# -gt 1 ]]
do
key="$1"

case $key in
    -c|--count)
    COUNT="$2"
    shift
    ;;
```

[3] https://github.com/noahgift/cli-rosetta/tree/master/bash/hello-world

```
    -p|--phrase)
    PHRASE="$2"
    shift
    ;;
esac
shift
done

#Run program
phrase_generator "${COUNT}" "${PHRASE}"
```

To lint use `make lint`. And now run it:

```
$  hello-world git:(master) $ ./hello-world.rb --count 5 --phrase "hello world"
hello world
hello world
hello world
hello world
hello world
```

Environment

- Installed https://github.com/koalaman/shellcheck in VSCode
- On Homebrew you can install: `brew install shellcheck`

You may need to also do:

```
$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.zshenv
$ echo 'eval "$(rbenv init -)"' >> ~/.zshenv
$ echo 'source $HOME/.zshenv' >> ~/.zshrc
$ exec $SHELL
```

Go Hello World

The `go` source code is here[4].

[4] https://github.com/noahgift/cli-rosetta/tree/master/go/hello-world

```go
package main

import (
    "fmt"
    "gopkg.in/urfave/cli.v1" // imports as package "cli"
    "os"
)

func main() {
    app := cli.NewApp()
    app.Flags = []cli.Flag{
        cli.StringFlag{
            Name:  "phrase",
            Usage: "Print phrase",
        },
        cli.Int64Flag{
            Name:  "count",
            Usage: "Count to print a phrase",
        },
    }

    app.Action = func(c *cli.Context) error {
        sum := 0
        for i := 0; i < c.Int("count"); i++ {
            sum -= i
            fmt.Println(c.String("phrase"))
        }
        return nil
    }

    app.Run(os.Args)
}
```

Running program

Run `make all`

Then run program:

```
$  hello-world git:(master) $ hello-world --phrase "hello world" --count 5
hello world
hello world
hello world
hello world
hello world
```

Environment

- Setup on mac with homebrew:
  - https://stackoverflow.com/questions/12843063/install-go-with-brew-and-running-the-gotour[5]

Setting these variables:

```
export GOPATH="${HOME}/.go"
export GOROOT="$(brew --prefix golang)/libexec"
export PATH="$PATH:${GOPATH}/bin:${GOROOT}/bin"
export GOBIN=$GOPATH/bin
```

- Write first program with Go
  - https://golang.org/doc/code.html[6]
- Using cli:
  - https://github.com/urfave/cli[7]
- Running lint:
  `make lint`
- Installing dependencies:
  `make install`

Node Hello World

The source code for the `node` examples is here[8]. This project has several components. First, there are the `.js` file.

[5] https://stackoverflow.com/questions/12843063/install-go-with-brew-and-running-the-gotour
[6] https://golang.org/doc/code.html
[7] https://github.com/urfave/cli
[8] https://github.com/noahgift/cli-rosetta/tree/master/node/hello-world

```
#!/usr/bin/env node
"use strict";

/*

Hello World Commandline Tool

node index.js --phrase "hello world" --count 10

hello world hello world hello world

*/

const program = require('commander');
program
  .version('0.0.1')
  .option('-p, --phrase [value]', 'phrase')
  .option('-c, --count <n>', 'Number of Times To Repeat Phrase', parseInt)
  .parse(process.argv);

/**
 * Multiplies string with additional space.
 * @param {string} phrase The phrase.
 * @param {number} count The number of times to repeat
 * @returns {string} The multiplied string
 */
function phraseGenerator (phrase, count) {

    return phrase.concat(" ").repeat(count);

}

// Check to see both options are used
if (typeof program.phrase === 'undefined' ||
    typeof program.count === 'undefined') {

    console.error('ERROR! --phrase and --count options required');
    program.help();
    process.exit(1);

}

// Print Phrase To Standard Out
```

```
console.log(phraseGenerator(
                program.phrase,
                program.count));
```

Next there is a `package.json` file.

```
{
  "name": "nodecli",
  "version": "1.0.0",
  "description": "nodecli",
  "main": "index.js",
  "dependencies": {
    "commander": "^2.10.0"
  },
  "devDependencies": {
    "eslint": "^4.1.1",
    "eslint-config-defaults": "^9.0.0"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/noahgift/nodecli.git"
  },
  "keywords": [
    "cli"
  ],
  "author": "Noah Gift",
  "license": "MIT",
  "bugs": {
    "url": "https://github.com/noahgift/nodecli/issues"
  },
  "homepage": "https://github.com/noahgift/nodecli#readme"
}
```

To run the example, you would do the following.

Steps to run:

```
npm install
./hello-world --phrase "hello world" --count 3
```

The output should be:

```
hello world hello world hello world
```

Multi-paradigm Node.js

Getting Started

This application source code is here[9].

To build this project, you need to:

```
npm install
```

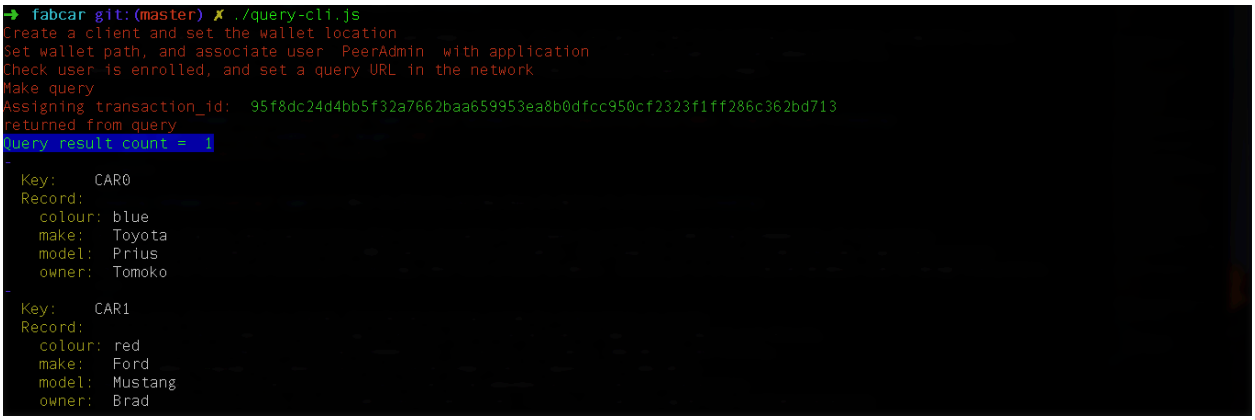
To talk to the blockchain network, please run ./startFabric.sh. Then ./query-cli.js.
Further information and background of this forked version can be found here[10]

Features

- Color Output
- JSON Formatting
- Async Network Operations
- Blockchain Integration

Screenshots

* Screenshot No Options:



Output

[9] https://github.com/noahgift/cli-rosetta/tree/master/node/multi-paradigm
[10] http://hyperledger-fabric.readthedocs.io/en/latest/write_first_app.html

* Screenshot One Option:

```
➜ fabcar git:(master) ✗ ./query-cli.js --car "CAR2"
Create a client and set the wallet location
Set wallet path, and associate user  PeerAdmin  with application
Check user is enrolled, and set a query URL in the network
Make query
Assigning transaction_id:  ab679885c5a4fa8a3148a63c253949102e13a3762fe592b4e94a2dc8d1c49e1f
returned from query
Query result count =  1
colour: green
make:   Hyundai
model:  Tucson
owner:  Jin Soo
```

Output

The source is below. You can see how a more sophisticated `node` tool integrates with both blockchain, and colored output works.

```
#!/usr/bin/env node
"use strict";

/*

Hyperledger Query Commandline Tool

./query-cli.js
__dirname = path.resolve();
*/
const Hfc = require('fabric-client'),
    path = require('path'),
    chalk = require('chalk'),
    prettyjson = require('prettyjson'),
    program = require('commander'),
    options = {
        walletPath: path.join(__dirname, './network/creds'),
        userId: 'PeerAdmin',
        channelId: 'mychannel',
        chaincodeId: 'fabcar',
        networkUrl: 'grpc://localhost:7051'
    };
let channel = {},
    transactionId = null,
    client = null,
    jsonResult = null;
program
    .version('0.0.1')
```

```
    .option('-c, --car [value]', 'car to query')
    .parse(process.argv);

/**
 * Queries Blockchain
 * @param {string} chaincodeFunc The chaincode function to query
 * @param {string} car The individual car to query
 * @returns {string} nothing
 */
function queryHelper (chaincodeFunc = 'queryAllCars', car = '') {

    Promise.resolve().then(() => {

        console.log(chalk.red("Create a client and set the wallet location"));
        client = new Hfc();

        return Hfc.newDefaultKeyValueStore({path: options.walletPath});

    })

    .then((wallet) => {

        console.log(chalk.red("Set wallet path, and associate user ",
            options.userId, " with application"));
        client.setStateStore(wallet);

        return client.getUserContext(options.userId, true);

    })

    .then((user) => {

        console.log(
        chalk.red(
            "Check user is enrolled, and set a query URL in the network"
            ));
        if (typeof user === "undefined" || user.isEnrolled() === false) {

            console.error("User not defined, or not enrolled - error");

        }
        channel = client.newChannel(options.channelId);
        channel.addPeer(client.newPeer(options.networkUrl));
```

```
    })
    .then(() => {

        console.log(chalk.red("Make query"));
        transactionId = client.newTransactionID();
        console.log(chalk.red("Assigning transaction_id: "),
            chalk.green(transactionId._transaction_id));

        // The queryCar - requires 1 argument, ex: args: ['CAR4'],
        // The queryAllCars - requires no arguments , ex: args: [''],
        const request = {
            chaincodeId: options.chaincodeId,
            txId: transactionId,
            fcn: chaincodeFunc,
            args: [car]
        };

        return channel.queryByChaincode(request);

    })
    .then((queryResponses) => {

        console.log(chalk.red("returned from query"));
        if (typeof queryResponses.length === 'undefined') {

            console.log("No payloads were returned from query");

        } else {

            console.log(
                chalk.bgBlue("Query result count = ", queryResponses.length));

        }
        if (queryResponses[0] instanceof Error) {

            console.error("error from query = ", queryResponses[0]);

        }
        jsonResult = JSON.parse(queryResponses[0].toString());
        console.log(prettyjson.render(jsonResult, {
            keysColor: 'yellow',
            dashColor: 'blue',
```

```
            stringColor: 'white'
        }));

    })
    .catch((err) => {

        console.error("Caught Error", err);

    });

}
// Run The Command line Tool
if (typeof program.car === 'undefined') {

    queryHelper('queryAllCars');

} else {

    queryHelper('queryCar', program.car);

}
```

Python Hello World

Recommended environment

Python 3.6.1

Running

Steps to Run:

Install packages:

```
make install
```

Activate Virtual Env:

```
source ~/.hello-world-py-cli/bin/activate
```

Run Tool:

```
./hello-world.py --phrase "hello world" --count 3
```

The output should be:

```
hello world hello world hello world
```

The ‘Makefile looks like:

```makefile
install:
      mkdir -p ~/.hello-world-py-cli &&\
      python3 -m venv ~/.hello-world-py-cli &&\
      pip install -r requirements.txt


source-cmd:
    echo "Virtualenv source command"
    #source ~/.hello-world-py-cli/bin/activate

lint:
    pylint hello-world.py
```

The requirements.txt file looks like:

```
click
pylint
```

Finally, the `python` code is as follows.

```python
#!/usr/bin/env python
import click

@click.version_option("0.1")
@click.group()
def cli():
    """Hello World"""

@cli.command("hello")
@click.option("--phrase", help="phrase to print")
@click.option("--count", help="Number of times to repeat phrase", type=int)
def hello(phrase, count):
    """Hello World Command-line tool"""

    while count:
        count -= 1
        click.echo(phrase)
```

```python
if __name__ == '__main__':
    cli()
```

A full example of lint and run:

```
(.hello-world-py-cli) $  hello-world git:(master) $ ./hello-world.py hello\
    --phrase "hello world" --count 3
hello world
hello world
hello world
(.hello-world-py-cli) $  hello-world git:(master) $ make lint
pylint hello-world.py

------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 7.50/10, +2.50)
```

You can find the latest up to date examples in the Github Repo https://github.com/noahgift/cli-rosetta[11].

[11] https://github.com/noahgift/cli-rosetta