Bash Code Risk assignment

Randy Tran
Clayton Wright

Research Gap

Overview

Contents of Bash are currently handwritten. Our goal was to see if it is possible to randomly generate computer readable bash commands. Generating computer readable commands would be beneficial for being able to understand the risk of our code as well as protect against the greatest risk of them all - human error.

Goals

1. Compile bash man page data into a CSV file to generate the bash commands
2. Generate random bash commands on a docker image. Docker is used in order to mitigate potential unintended consequences of random commands

Progress

Steps

I. Data extract

- Using Beautiful Soup, we were able to scrape the bash man page's HTML code and extract relevant data (command name, syntax, options, keys) and write them to our own .txt file

II. Search each section of the text file

- Scanning through the .txt file, each command name is separated by '****' and the options begin with '-'. Still not computer readable because of the descriptions for each option/key.

III. CSV creation

- Parse down the .txt file to a CSV to easily navigate and interact with the data. The CSV has 3 columns: command, syntax, and options/key.

IIII. Generate commands

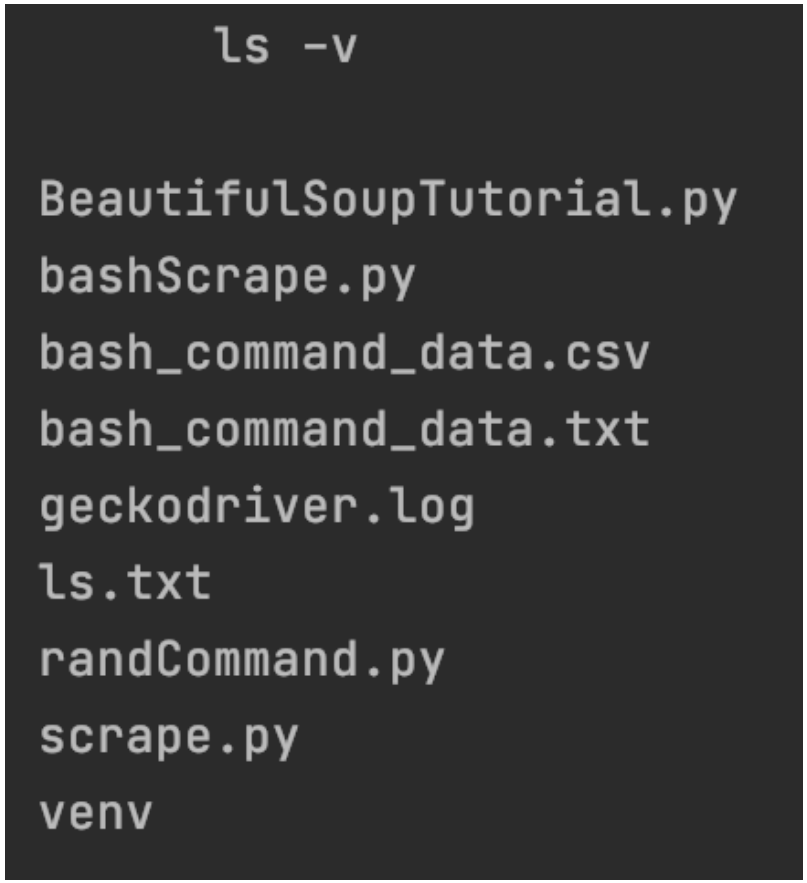
- Using the data from the CSV, generate random yet functional commands by adhering to each command's syntax and options. Would be optimal to be able to generate for the 15 most popular commands.

V. Gather data

- Affirming the commands and demonstrating the system's functionality, compile a list of commands to further demonstrate its function

```
********************

ls
Syntax
      ls [Options]... [File]...

Key
  -a, --all                  List all entries including those starting with a dot .

  -A, --almost-all           List all entries including those starting with a dot .
                             Except for . and .. (implied).

  -b, --escape               Print octal escapes for nongraphic characters.

      --block-size=SIZE      Use SIZE-byte blocks.

  -B, --ignore-backups       Do not list implied entries ending with ~

  -c                         Sort by change time; with -l: show ctime.

  -C                         List entries by columns.

      --color[=WHEN]         Control whether color is used to distinguish file
                             types. WHEN can be 'never', 'always', or 'auto'

  -d, --directory            List directory entries instead of contents.

  -D, --dired                Generate output designed for Emacs' dired mode.

  -f                         Do not sort, enable -aU, disable -lst

  -F, --classify             Append indicator (one of */=@|) to entries.
```

ls bash_command_data.txt example



```
ls -v

BeautifulSoupTutorial.py
bashScrape.py
bash_command_data.csv
bash_command_data.txt
geckodriver.log
ls.txt
randCommand.py
scrape.py
venv
```

Random command example

Takeaways

- Lessons learned
  - Research requires asking questions and there are no dumb questions
- Challenges faced
  - Learning on the fly and adapting to the next challenge
- What went well
  - Developing skills in software engineering and learning a lot about bash and docker