homework_02

This homework will have you write shell scripts that that use unix utilities and python utilities that you build. This is done in the name of analyzing (an altered version of) the SF 311 Dataset. This altered version is available here

Due: Monday Feb 18, 6pm.

To receive full credit, you must commit and push code that passes all unit tests, and shell scripts that give the correct output.

Setup

Clone the repo and save it in a local directory called `homework_02` by typing

```
git clone https://github.com/columbia-applied-data-science/homework_02_team_XX.git \
homework_02
```

Utilities

Note: To use the pytyhon utilities, your PYTHONPATH must be modified. In your `~/.bashrc` (or `~/.bash_profile` on macs), put

```
export PYTHONPATH=path-to-directory-above-homework_02:$PYTHONPATH
```

Then source it with `source ~/.bashrc` or open a new terminal.

To see how the utilities *should* work:

- Create a comma delimited file with a header and run the utilities on it. Set a breakpoint and step through, guessing reading the comments and code fragments provided. You can view the documentation for each utility by typing `python utilityname -h`.
- Go to `test/` and view the unit tests in `test/testutils.py`.
- Look at the comments in the utilities. These are only hints. Any utility that passes tests is acceptable.

body

Note: This utility will not be tested, it is just given to you.

In your `.bashrc`, put

```
body() {
    IFS= read -r header
    printf '%s\n' "$header"
    "$@"
}
export -f body
```

then source the bashrc.

This allows you to run a command on the body of the function, skipping the header (but still printing the header). For example,

```
cat filewithheader | body sort -k1,1
```

will sort `filewithheader`, using the first field, but leave the header at the top of the file.

cut.py

Acts like the unix cut utility, except...

- Takes field names rather than numbers
- Uses the python csv module for more automatic handling of stuff like quoted delimiters

reformat.py

Reformats stuff like delimiters and capitalization

common.py

Common files for all utilities

averager.py

Gets the average of different groups of a sorted file

timeopen.py

Reads a SF 311 case file, appends a 'timeopen' column giving the time (in minutes) a case was open.

subsample.py

Subsamples in the space of rows.

Shell Scripts

These are simple shell scripts. They simply define variables and pipe together some commands. The input file is written into the script. The script writes to stdout and stderr. An example of a script like this (that counts words) would be:

```
DATA=../data

cat $DATA/infile.csv \
  | sort \
  | uniq -c \
  > outfile.csv
```

Use the hints inside of these shell scripts to complete them. "Complete" means that they reproduce the sample input/output inside `data/`. For example,

```
cd scripts
./count_categories.sh > /tmp/stdout 2> /tmp/stderr
diff /tmp/stderr ../data/count_categories_stderr
diff /tmp/stdout ../data/count_categories_stdout
```

will produce two files, `/tmp/stdout` and `/tmp/stderr` and then compare them to the files in `data`. If everything is working, then `diff` should print nothing.

count_categories.sh

Count the number of tickets in each category

count_categories_openclosed.sh

Count the number of tickets in each category that are Open or Closed

compute_averages.sh

Compute the average time tickets in different categories remain open.

- For closed tickets, compute the average time it was open before being closed.
- For open tickets, compute the time it has been left open.

Unit Tests

To run tests, cd to *tests/* and do

```
python -m unittest -v testutils
```

Once you are done, you will get notification that all tests passed.