Percolate

Coding challenge

If you're reading this, that means that Percolate is really interested in your engineering abilities. Congratulations!

This is intended to be a small coding exercise meant to gauge your programming style. Now's your chance to show off your design skills (simple but extensible), your testing habits (hey, hey unittest), and your keen attention to detail.

Included is an input file that you will write a Python command-line utility to process. After you've engineered a solution to your satisfaction, you'll upload your solution and the result of running the input file through your program. Please package up your response in a tar.gz file. The URL for upload will be included in the email you received.

Your solution should be documented and tested to production standard. It should follow good Python conventions, and should be easy to get running.

After you send it off to us, and we'll get back to you shortly.

The problem

You're tasked with taking entries of personal information in multiple formats and normalizing each entry into a standard JSON format.

Python version

The utility should be compatible with Python 2.7. If you can't meet this requirement, let us know. Your utility should also only import modules that are available in the standard library. This makes the utility easy to test for us here at Percolate.

Input

Your program will be fed an input file of n lines. Each line contains "entry" information, which consists of a first name, last name, phone number, color, and zip code.

The order and format of these lines vary in three separate ways. The three acceptable formats are as follows:

Lastname, Firstname, (703)-742-0996, Blue, 10013
Firstname Lastname, Red, 11237, 703 955 0373
Firstname, Lastname, 10013, 646 111 0101, Green

A line is defined as invalid if it does not comply with one of the formats shown above. Invalid lines should not interfere with the processing of subsequent valid lines. A zip code is considered valid if it has 5 digits, and no other characters. A phone number is considered valid if it has 10, and only 10, digits. A phone number may have other non-numerical characters however, as shown shown above.

Output

The program should write a valid, formatted JSON object out to result.out. The JSON representation should be indented with two spaces. Within the JSON object should be a list named "entries". The "entries" list should be sorted in ascending alphabetical order by (last name, first name).

Successfully processed lines should result in a normalized addition to the list associated with the “entries” key. For lines that were unable to be processed, a line number i (where $0 \le i < n$) for each faulty line should be appended to the list associated with the “errors” key. The first line in the file is numbered 0.

Sample

For the input

```
Booker T., Washington, 87360, 373 781 7380, yellow
Chandler, Kerri, (623)-668-9293, pink, 123123121
James Murphy, yellow, 83880, 018 154 6474
asdfawefawea
```

We should receive the output

```
{
  "entries": [
    {
      "color": "yellow",
      "firstname": "James",
      "lastname": "Murphy",
      "phonenumber": "018-154-6474",
      "zipcode": "83880"
    },
    {
      "color": "yellow",
      "firstname": "Booker T.",
      "lastname": "Washington",
      "phonenumber": "373-781-7380",
      "zipcode": "87360"
    }
  ],
  "errors": [
    1,
    3
  ]
}
```

Questions?

Feel free to email.

Good luck!
Your friends at Percolate