Project Big Data

Assignment 1

The goal of this assignment is to first make you familiar with the command line and next with Pythons syntax, data types, and file i/o. This assignment is divided into twelve exercises.

The files `hue_upload.csv` and `hue_upload2.csv`, bundled in "`data week 1.zip`", are two files in the same format, that have been collected over different time periods. Each line contains a line number, a user id, an event string, and a data field, separated by semicolons (;) and enclosed by double quotes (").

Command-line exercises

Most *nix utilities provide information on their options and behaviour when they are passed the command-line option `--help`. Besides, each command comes with a manual (man) page, invoked by typing "`man <utility>`" for *nix or Mac. For Windows, the man pages are best found on the web, e.g. at https://man.cx.

Regular expressions come in different flavours. We advise to use the flavour known as 'extended regular expressions'. `grep -E` and `sed -r` are options that select this flavour. Python and gawk support extended regular expressions by default.

Some characters have a special meaning for the terminal/CommandPrompt, like ", |, ; and then some. If they need to appear on your command line normally, i.e. not in their special meaning, they must be *escaped*. Alas, this works differently under Windows and Mac/*nix.

Windows: to pass e.g. a " or ; in a parameter to your command-line program, escape it by prefixing it with ^\^

Mac: to pass e.g. a ; in a parameter to your command-line program, escape it by prefixing it with a \

Both: if a special character occurs inside a double-quoted string, that is (usually) enough to escape its special meaning.

Exercise 1. (2 points) Write a command that combines `hue_upload.csv` and `hue_upload2.csv` into a single file `hue_combined.csv`.

Exercise 2. (5 points) Write a command that reads `hue_combined.csv`, removes all double quote (") characters and the first field of each row, and stores the result as `hue_combined_cleaned.csv`.

Exercise 3. (5 points) Write a command that removes duplicate lines from `hue_combined_cleaned.csv` and stores the result as `hue.csv`.

The following commands should take `hue.csv` as input.

Exercise 4. (3 points) Write a command that shows the number of lamp_change events.

Exercise 5. (5 points) Write a command that shows the unique values of adherence_importance.

Exercise 6. (5 points) Write a command that shows the number of data points (lines) for each user id.

Exercise 7. (10 points) Write a command that shows all unique strings. A string consists of one or more words joined by an underscore, a word being one or more alphabetic characters: lamp_change, rise_reason, mei, but not 15_mei or 2015_risetime.

Exercise 8. (10 points) Write a command that shows the number of lamp_change events for each relevant day.

Python exercises

The following assignments use Python 3. To get hands-on familiarity with Python, you are strongly advised to work through a Python tutorial (up to some point: e.g., our course "Project Big Data" stops short of Python objects). We recommend one of these two web tutorials:

1. The "official" Python site: Python3 tutorial;

2. Google's Python Class, with the caveat that it uses Python2.7. However, we think it contains mostly version-agnostic code so it will also serve to learn Python3. One notable difference between Python2 and Python3 is the `print` function: in Python3, braces are required around the function arguments.

Python has an excellent reference web site, with documentation for all its language constructs and library APIs. Use this for all your Python work!

Exercise 9. (10 points) A number is composed of digits. For example, 512 is composed of a 5, a 1 and a 2. Write a Python script that accepts keyboard input (stdin). If the user does not enter a number or if the number is smaller than 10, the script has to keep asking for input. When a number is detected, the script should compute and output the number of digits, the number of distinct digits, the largest sum of two consecutive digits (Dutch: opeenvolgende cijfers), and the sum of its distinct prime factors. For the number 5112, the program should give the following output:

```
4
3
6
76
```

Your implementation should print the solution to the screen. The function does not have to return anything.

Exercise 10. (10 points) Write a function that reads two text files, each of which has one word per line in lowercase, checks which words occur in the first file but not in the second file, and writes those words to a third file in alphabetical order separated by a newline character. You may assume that both text files comfortably fit in memory, and that the files do not contain duplicates.

Your implementation should output the solution to a file. The function does not have to return anything or print anything to the screen.

Exercise 11. (10 points) Write a function that reads an integer distance matrix from a text file. An example is the following input file:

```
0 1 -
2 0 4
- 4 0
```

which contains, e.g., the information that the distance from point 0 to point 1 is 1, while the distance from point 1 to point 0 is 2. A dash means that no direct connection exists (which can be implemented as float('inf'), so that the connection will not get used). Implement Dijkstra's Algorithm to find the shortest path from point 0 to the last point (2 in this case). You may not use a package that features Dijkstras Algorithm. For the matrix shown above, the shortest distance is 5.

Your implementation should print the length of the shortest path and the shortest path itself to the screen.

Exercise 12. (15 points) International draughts (Dutch: dammen) is a board game played on a 10x10 field with a chessboard pattern. For this assignment, you have to read a file that contains the state of a game as it is being played, and provide a graphical representation. For this, only a small subset of the rules is relevant:

- There are two players, white and black.
- A 10x10 coordinate system is imposed on the board such that (1,1) is the left lower square for the white player and (10,1) is the right lower square for the white player.
- The board is oriented such that (1,1) is a dark square.
- Pieces can only be placed on dark squares, i.e., on (x,y) for which `|x-y| = 0 (mod 2)`.
- There are two types of pieces: regular pieces and promoted pieces.

The starting position will be supplied in an ASCII text file. Its lines contain the positions of (crowned) pieces. A valid line contains a coordinate, followed by a tab, followed by the type (w, W, b or B for a white piece, white promoted piece, black piece or black promoted piece, respectively). Invalid lines should be ignored, as should any characters after the type. The newline character can be CR+LF (Windows), CR (Mac, up to OS 9) or LF ("*"nix). An example is shown below:

```
w
(5,5) w the predator
(6,6) b the prey
(this is an example)
```

Write a function that takes the name and path of a text file as its argument, reads the text file, and prints a graphical representation of the board on the screen. The representation should use underscores and vertical bars as follows (notice that this example is a 3x3 board in stead of a full board):

```
 -----------
|   |   |   |
|   |   |   |
|___|___|___|
|   |   |   |
|   |   |   |
|___|___|___|
|   |   |   |
|   |   |   |
|___|___|___|
```

Note that each side consists of either three underscores or three vertical bars. The center of each square should contain the character w, W, b or B to indicate the piece (if any).

Your implementation should print the output to the screen. The function does not need to return anything. For this assignment, you need to verify whether `|x-y| = 0 (mod 2)`, in other words `|x-y|` is an even number. The template provides some invalid lines as examples. The line with coordinate (3,1) is invalid since it starts with a space, and the line with coordinate (3,5) is invalid since there is no tab that separates the coordinate from the color.

Notes

Another 10 points are awarded for the cleanliness of your code and the use of idiomatic Python (such as list comprehensions, sets, dicts, etc.). You can obtain 100 points in total.

The assignment should be done in groups of two students, and must be handed in via Canvas on June 8 by 23:59h. Your solution should use the provided template (`solution.py`), which is the (only) file that you should submit. It is *crucial* that you do not alter the names and arguments of the existing functions, although adding additional functions is recommended. The template contains the file `run_solution.py` that can be used to verify that your `solution.py` is in the correct format; `run_solution.py` can be run from the command line via `python run_solution.py` (Windows) or `python3.6 run_solution.py` (Mac), or from Spyder.

Each group should hand in only one solution. If both students submit a solution, only the first submission will be graded. Feedback will be provided to the e-mail addresses you provide in `solution.py`.

Tips

- If your shell command does not run, check whether you accidentally use ” (curly quote) in stead of " (straight quote) or – (endash) in stead of - (minus sign). These are common copy/paste errors.
- We will run your solutions with different (possibly longer or more complex) input files than provided in the template.
- Use the Python reference on the web!