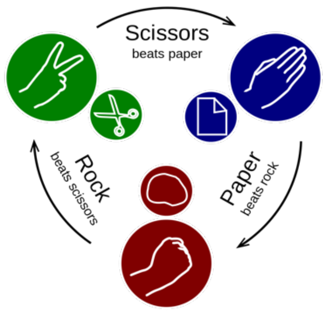
Data Science with Python – Assignment #1

Assignment description

This assignment contains two tasks: (1) the rock-paper-scissors (אבן נייר ומספריים) game solution, and (2) deciphering a secret code (פענוח צופן סודי) in English.

Task #1: the rock-paper-scissors game



The well know game (originating from China) follows the rules in the attached image. Several students are playing the game and recording their choices in a file. The goal of the code you will write is to read the file and decide who is the winner of the game, or if it's a tie (multiple players have the same score). At each turn two students out of the group play, and their result is recorded in a file, in the following manner:

```
player1      player1-choice     player2      player2-choice
```

As an example, for the below result of four rounds, Moshe will be announced as a winner, as he has the highest score of 2/3 (two wins out of his three games). Note that some players may have more rounds than others. We announce a winner according to the proportion of rounds they win.

```
Moshe        rock        Ayelet      scissors
David        scissors    Noam        paper
David        rock        Moshe       paper
Moshe        rock        Noam        paper
```

You are given two files: `config-rps.json` and `rps_game.py`, where `config-rps.json` contains the input file location. You can change these locations in `config-rps.json`, but do not make any changes to the `main()` code. Given additional file with a game results – `rps-results.txt` – implement the `game()` function in `rps_game.py`. The function should return one of the string values: <player name> (a single winner) or "tie" (tie between two or more players, without mentioning their names).

```
def game(results_filename):
    # todo: implement this function
    ...
```

<u>Comments:</u>

(1) Do not assume that the required input file exists in the specified location – check that.
If the file does not exist, print error message, and exit the program.

(2) Assume the input file has a valid format, containing a header and at least one game result.

(3) In the case of the same choice at a single round (e.g., paper and paper) – no points are assigned.

Task #2: deciphering a secret code

The goal of this task is to decipher a secret code using the partially-known rules of the cipher. As an example, the below phrase is encoded using the cipher:

```
hexe wgmirgi mw jyr
```

We know that the ciphering process works as follows: given an English phrase and a secret number K, replace each character (except spaces) with another one that is located K characters further away in the English ABC. In the above example, we will found that K is 4 and the deciphered phrase is:

```
data science is fun
```

You are given multiple files: `config-decipher.json` and `decipher.py`, where `config-decipher.json` contains a secret phrase, and additional input locations: a serialized file with top-10K most frequent words in the English lexicon and a file with the English ABC. Do not make any changes to the `main()` code. Given these input files, implement the `decipher()` function in `decipher.py`. The function should print the value of K and the deciphered (original) phrase:

```
def decipher_phrase(phrase, lexicon_filename, abc_filename):
    # todo: implement this function
    ...
```

Note that if adding K characters results in “overflow” (as an example the current character is “x” and K is 7) -- the character for replacement will be computed in a cycled manner: reaching the end and starting from the beginning. In the given example, “x” will be replaced with “e”.

The function returns a dictionary with the result: (a) status, (b) deciphered phrase and (c) K.

- In case a given phrase can be deciphered, the status should be equal to 1.
- In case a given phrase cannot be deciphered, the status should be equal to -1.
- In case a given phrase is empty, the status should be equal to 0.

Comments:

(1) Do not assume that the required input file exists in the specified location – check that.
If the file does not exist, print error message, and exit the program.

(2) Assume we are only using the lowercase English letters (as in the attached `abc.txt` file).

(3) The ciphering procedure is only applied on non-whitespace characters: spaces are left as they are.

(4) The attached English lexicon consists of all possible words that original phrase can contain.

(5) Assume we are not dealing with punctuation in this assignment.

(6) Assume a positive K in the [0, 25] range, including.

Submission

Grading criteria include: correctness (the major part), code design, readability and documentation.

Good documentation implies a comment for every function, comments for constants (if exist) and comments for code blocks that may not be clear when reading the code.

Good design will result in several functions, where each function is responsible for a single action (or small number of related actions). You can make use also of Python functions we didn't cover in class, after reading documentation and understanding them properly.

Submit a single zip file – assignment1_ xxxxxxxxx_xxxxxxxxx.zip , where "xxxxxxxxx" stands for a student id. Please specify two student ids (your and your partner's). It should include two files:

1. Your implementation for task #1 – `rps_game.py`.
2. Your implementation for task #2 – `decipher.py`.

Good Luck!