CODING ASSIGNMENT

Hello!

In this assignment we'll explore how you approach a development task. We tried making this assignment clear and interesting for you, and we wanted to give you the freedom to come up with your own design and code structure. There are many possible solutions. What's important here is to see your style and thought process. We respect your time, so it's OK to keep it concise.

Instructions

- The assignment should be completed in Python 3.7.
- Use Python's default libraries (you can import anything that comes with Python 3, including 're').
- We are looking for clean, readable code and an emphasis on Object Oriented Programming. Before starting out, please consider carefully which classes should perform which operations.
- Try using a configuration file instead of having hardcoded values in your classes.
- For each part write a small 'Main' function that runs your code, or alternatively – a short unittest.

Guiding questions

In every step, ask yourself:

1. ***Is this code readable enough? (If I see it again in six months, will I understand it easily?)***
2. ***What should be the name(s) of the class(es) for this part?***
3. ***How do I instantiate and operate the classes?***
4. ***What are the appropriate access modifiers for these methods and variables?***
5. ***How do the objects share information and/or resources?***

Submitting your solution

Upload your solution to Google Drive (or similar service) and send back the link.

Feel free to ask

If anything is unclear, please ask for clarifications!

Rafael

***Email**: rafael.benari@gm.com*

Part 1: Keyword Search

Please find the attached file **`keywords.txt`**. In this part of the assignment, we will implement a class for keyword searches: **Given a data string as input, find all instances of the keywords in the input string.**

Instructions

- Find *case insensitive* instances of the keywords.
- Find *whole words* only:
  - *If you're searching for `'sport'`, do not find `'transport'` (false positive).*
  - *If you're searching for `'design'`, do not find `'designated'` (false positive).*
- Keywords can contain spaces (for instance: **`'Cyber Security'`**).
  The search algorithm should be able to find minor alterations, such as a dash (**`'-'`**) instead of space (**`' '`**), or the elimination of the space character. In this case you should also be able to also find **`'cyber-security'`**, **`'cybersecurity'`**.

Output

- The search results should be a list of keywords found without repetitions.

Hint: Use any of the built-in python libraries, especially where it makes your work easier!

Input and Output Examples

[the keywords are read from the file **`keywords.txt`**]

```
Input: 'Welcome to >>GENERAL-motors! We love programming!'

Output: ['general motors', 'programming']
```

```
Input: 'Beside being a team focused on cyber-security, we also do software
engineering. With good communication we might figure out some unsolved
problems in computer-science!'

Output: ['communication', 'cyber security', 'computer science', 'software
engineering', 'unsolved problems in computer science']
```

Part 2: Rate Limiting

Create a multithreaded class with workers that get a random page from Wikipedia and find the keywords from *keywords.txt* anywhere in the response (you can use your search implementation from Part 1).

- The configuration file should have a **parameter** for **number of workers.**
- Each worker runs in its own thread. You must always have this number of workers running.
- You can use additional threads beside workers.

Hint: This URL will get you a random Wikipedia page every time:
https://en.wikipedia.org/wiki/Special:Random

Rate Limiting

Since we don't want to burden Wikipedia's servers, we will use **rate limiting** for our requests.

- The configuration file should have a **parameter** for the maximum number of allowed **requests per second** *(across all workers)*. Make sure the program doesn't exceed this rate of requests.
- Don't stop or start new workers – the "**number of workers**" parameter still applies.
- There might be more workers than allowed requests per second. The rate limit still applies regardless of how many workers are available. (Hint: Workers can wait or become blocked).
- If you can, don't send workers to **`'sleep'`**. Try avoiding 'Busy Waiting' where possible.

Output

The machine should run until you manually stop it. For each result, print:

- The current worker number.
- The URL.
- The keyword matches found in the page.

Output Example

```
Worker: 1
Random URL: https://en.wikipedia.org/wiki/British_Polar_Engines
Matches: ['construction', 'design', 'manufacturing']
----------------------------
Worker: 5
Random URL: https://en.wikipedia.org/wiki/Nuclear_famine
Matches: ['agriculture', 'biotechnology', 'energy', 'medicine', 'artificial intelligence',
'computer model']
----------------------------
Worker: 2
Random URL: https://en.wikipedia.org/wiki/Cannock_Chase_Railways
Matches: ['construction']
```

GOOD LUCK!