CODING ASSIGNMENT

Hello!

In this assignment we'll explore how you would approach a development task at work. We tried making this assignment clear and interesting for you. Additionally, we wanted to give you the freedom to come up with your own project structure, classes and OOP design.

Please take your time, relax and work comfortably. Remember, there is no single right answer, it's an exploration of your own style - and there are many possible solutions.

Instructions

- The assignment should be completed in Python 3.
- Use Python's default libraries (you can import anything that comes with Python 3).
- Using the default libraries is encouraged! Especially where it saves you work!
- 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.

Guiding questions

In every step, ask yourself:

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

Submitting your solution

Just put everything in a zip file and email it back.

Feel free to ask

If anything is unclear, please ask for clarifications!

Rafael

***Email**: rafael.benari@gm.com*
***WhatsApp**: 0525304377 (Please mention who you are if you message me!)*

Part 1: Keyword Search

Please find the attached file **`keywords.txt`**. In this part of the assignment, we need to create these capabilities:

- **Given a data string as input, find all instances of the keywords in the input string.**

Instructions

- The object should be able to 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.

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: Multithreading

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 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.
- Do not use **`'sleep'`**, timeouts or equivalents in a worker. If you use **`'sleep'`**, use it in one thread only.

Output

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

- The current worker name (or id).
- The random URL.
- The keywords 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/Manitoba_Highway_16A
Matches: []
----------------------------
Worker: 1
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!