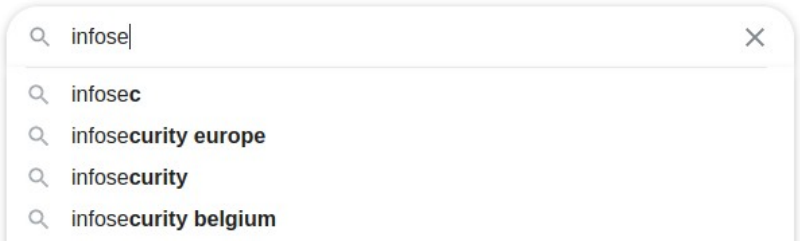
Exercice – Auto-complétion

L'exercice consiste à réaliser un système d'auto-complétion, à la manière de celui de la recherche Google, mais beaucoup plus simple.



Fonctionnement

Il s'agit de développer un service web qui pourra être interrogé pour fournir une auto-complétion.

Auto-complétion

Etant donné un dictionnaire de termes (en annexe), lorsqu'on fournit **une suite de lettres** au système, on veut obtenir la liste ordonnée des termes du dictionnaire qui commencent par ces lettres.

Concernant le résultat de l'auto-complétion :

- La liste de termes retournée est limitée à **4 éléments**.
- Le tri de la liste est effectué selon l'**ordre alphabétique**. Ainsi, les termes retournés sont ceux dont les premières lettres correspondent aux lettres fournies et parmi lesquels les 4 premiers éléments selon l'ordre alphabétique ont été sélectionnés.
- La **casse des lettres (minuscules ou majuscules) n'est pas prise en compte** par le système.

> *Exemple :*
>
> ***crypt*** ⟶ ***crypt****analysis*, ***crypt****ographers*, ***crypt****ographic algorithm*, ***crypt****ography*

Service web

Le système d'auto-complétion est un service web qui expose une API web. L'interaction via l'API se fait de la façon suivante :

- Le service est interrogé via une requête HTTP **GET** sur le point d'entrée **`/autocomplete`**
- Un paramètre nommé **`query`** est passé à la requête **GET** pour indiquer les lettres à partir desquelles l'auto-complétion se fera.
- Le service fournit comme **réponse** une **liste en JSON** contenant les termes pour l'auto-complétion.

> *Exemple :*
>
> Requête :
> GET **`/autocomplete?query=crypt`**
>
> Réponse :
> `["cryptanalysis", "cryptographers", "cryptographic algorithm", cryptography"]`

Résultat attendu

Il est attendu une implémentation minimale de ce système d'auto-complétion mais utilisant néanmoins une structure de données adaptée qui permetterait d'utiliser un dictionnaire plus conséquent que celui fourni. Cette implémentation consistera uniquement en un **service back-end** pour répondre aux requêtes. Il n'est pas attendu d'interface graphique. Le système gérera les fonctionnalités exposées ci-dessus. Un système de gestion des erreurs, en cas d'utilisation incorrect, n'est pas nécessaire.

Un code **claire et modulaire** est attendu. La documentation du code est laissée à l'appréciation du programmeur selon la complexité du code. En revanche, un fichier **README** précisant comment faire fonctionner le logiciel est attendu.

Contraintes

L'exercice doit être réalisé en **Python**.

Le système devra être implémenté avec **la bibliothèque standard uniquement**. **Il n'est pas possible d'utiliser des bibliothèques logicielles tierces. Par ailleurs, il n'est pas possible d'utiliser une base de données.** Toute la logique relative à la recherche des termes pour l'auto-complétion devra être implémentée à la main. Le dictionnaire de termes pourra être mis en dur dans le code pour une question de simplicité.

Améliorations

Le README devra égalemement comporter des éléments de réponse aux questions suivantes :

- Comment améliorer la pertinence des suggestions faites ?
- Comment gérer un dictionnaire de termes de taille plus conséquente ?

Annexe – Dictionnaire des termes

antivirus
application security
asset
attack surface
authorization
business impact assessment
cloud computing
computer network defense analysis
computer network defense infrastructure support
computer security incident
cryptanalysis
cryptographers
cryptographic algorithm
cryptography
cryptology
data breach
data integrity
data leakage
hacker
hash value
hashing
key
keylogger
malicious code
malware
symmetric cryptography
symmetric encryption algorithm