0.1 Introduction

Une API (Application Programming Interface), ou interface de programmation, est un ensemble normalisé de briques logicielles grâce auxquelles un système informatique offre des services à d'autres systèmes.

Quand nous consultons la météo sur notre smartphone, il utilise l'API du service météo en question. La plupart des applications mobiles sont conçues autour d'une ou plusieurs API. L'objectif d'une API est de fournir des fonctionnalités, sans révéler le fonctionnement interne de l'application qui fournit les données. C'est en cela qu'on dit qu'une API est en façade : on ne voit pas l'intérieur du bâtiment.

Concrètement, une API est constituée d'une bibliothèque logicielle, d'un service web et d'une description qui spécifie comment les clients peuvent interagir avec la plateforme logicielle qui fournit les données, appelée aussi fournisseur

0.1.1 Comment procéder?

Généralement, pour utiliser une API, il faut effectuer une requête sur un serveur web associé au fournisseur. Les éléments nécessaires à la requête sont précisés dans la description de l'API. Afin de limiter le nombre d'accès ou d'identifier des accès malveillants au serveur, il est souvent demandé une clé d'authentification. Cette clé s'obtient sur le site web du fournisseur. Elle peut être gratuite ou payante, selon les cas.

0.1.2 intérêt

Pourquoi utiliser une API pour s'interfacer à une base de données plutôt que de télécharger simplement un ou plusieurs fichier(s) CSV de l'ensemble des données ?

Quelques avantages:

- les données changent très rapidement. Ce qui est vrai aujourd'hui peut-être faux demain. Une API permet un lien permanent avec la base de données ;
- on peut filtrer les données que l'on récupère, en fonction de multiples critères, et ne pas télécharger la totalité d'une base de données à chaque fois ;
- une API permet d'automatiser les requêtes et de les intégrer dans d'autres applications.

0.1.3 En Python

De nombreuses API disposent d'un wrapper Python : c'est un module Python qui permet d'interroger la base de données du fournisseur de façon simplifiée, sans avoir à écrire les requêtes Web, ni à interpréter les réponses. Celles-ci sont généralement écrites au format JSON, qui est un format d'échange de données, au même titre que le XML. À partir de ces réponses, le wrapper Python génère des listes ou des dictionnaires, facilement manipulables.

Dans cette activité, nous allons utiliser l'API de la NASA :

0.2 API Asteroid NeoWs(Near Earth Object Web Service)

0.2.1 Installation

La description de l'API Asteroid NeoWs est disponible à cette adresse :

https://api.nasa.gov/

Premièrement, il faut générer une clé d'authentification, indispensable pour accéder à l'API, à cette même adresse. Il est possible d'utiliser une clé de démo, nommé DEMO_ KEY. Cette clé est plus

limitée en terme de nombre de requêtes, et cela pourrait ne pas suffire à mener cette activité à son terme.

utilisation de l'api :
Cette api permet de récupérer des données sur des astéroïdes ayant approché la planète Terre

1. Récupérer une clé d'authentification sur le site de la NASA Ici
2. Dans la rubrique Browse APIS , lire la documentation sur cette api.

   Utilisez ce que vous avez appris dans ce qui précède pour compléter le programme suivant qui permet de récupérer les données sur les astéroïdes ayant approché la planète Terre.

```
1
2       import requests
3       import json
4       api_key = " .. " # à remplacer par la clé d'authentification
5       url = "http://api.nasa.gov/neo/rest/v1/feed?/"
6       date_début = ' .. '
7       # à compléter
8       date_fin = ' .. '
9       # à comlpléter
10      params = ......................
11      response = requests.get (url , params = params)
12      data = json.loads(response.text )
13      print (data)
```

   ou ce code :

```
1       import requests
2
3       url= 'http://api.nasa.gov/neo/rest/v1/feed?/'
4
5       params= { 'start_date': '2024-09-01','end_date': '2024-09-02'
6           ,'api_key':'DEMO_KEY'}
7       response = requests.get(url, params=params)
8       data = response.json()
```

3. Quel est le format de la structure de données `data`.
4. Déterminer les clés de data.
5. Quel est le nombre d'astéroides observés?
6. astero['2024-09-02'] est une liste de dictionnaires.Vérifier-le.
7. Afficher les clés du premier dictionnaire de la liste précédente.
8. Ecrire une fonction plus_gros_astro (date) qui prend pour paramètre une date donnée (au format année-mois-jour) et qui renvoie le nom et le diamètre (max) du plus gros astéroïde ayant approché la Terre à date.
9. De même , écrire une fonction astero_plus_proche(date) qui renvoie le nom et la distance (min_dist) de l'astéroïde ayant approché le plus près la planète Terre à date