TP 1 API : Exploration et Manipulation des API avec Python

Objectif :

L'objectif de ce TP est de vous familiariser avec l'utilisation des API en Python. Vous apprendrez à interagir avec des services web externes, à envoyer des requêtes et à traiter les réponses.

Prérequis :

- Connaissance de base de Python
- Connaissance de base des requêtes HTTP (don’t worry si vous connaissez pas HTTP)

Matériel/Apps :

- Un éditeur de texte (VSCode, PyCharm, etc.)

Ps : J’ai utilisé PyCharm pour ce TP

Modalités pédagogiques :

Travail individuel

Durée : 1 jour

Contenu du TP :

Partie 1 : Introduction aux API et aux requêtes HTTP

1. **Introduction théorique :**
    - Qu'est-ce qu'une API ?
    - Les différents types de requêtes HTTP : GET, POST, PUT, DELETE ?
    - Les codes de statut HTTP les plus courants ?
2. **Installation de bibliothèques nécessaires**

- Utilisation de la bibliothèque `requests` en Python pour effectuer des requêtes HTTP.

```
pip install requests
```

Questions :

1. **Qu'est-ce qu'une API RESTful et comment diffère-t-elle des autres types d'API ?**
2. **Pourquoi les API utilisent-elles différents types de requêtes HTTP (GET, POST, PUT, DELETE) ? Donnez un exemple pour chaque type de requête.**
3. **Expliquez comment le code de statut HTTP 404 diffère du code 500.**

Partie 2 : Utilisation d'une API publique

1. **Configuration initiale**
   - Créer un nouveau fichier Python nommé `api_example.py`.
2. **Faire une requête GET à une API publique**
   - Utiliser l'API OpenWeatherMap pour obtenir des informations météorologiques (Trouvez comment avoir votre propre clé API sur le site : https://openweathermap.org, un exemple de code est donné ci-dessous, à adapter à vos besoin) :

```
import requests

# Remplacer 'votre_cle_api' par votre clé API obtenue de
puis OpenWeatherMap
api_key = 'votre_cle_api'
city = 'Paris'
url = f'http://api.openweathermap.org/data/2.5/weather?q
={city}&appid={api_key}'

response = requests.get(url)
if response.status_code == 200:
    data = response.json()
    print(data)
```

```
else:
    print('Erreur:', response.status_code)
```

- Exécutez le code ci-dessus et interprétez les résultats obtenus.

3. **Afficher les informations météorologiques de manière lisible :**

Utilisez le code ci-dessous pour afficher les informations météorologiques d'une ville de votre choix :

```
if response.status_code == 200:
    data = response.json()
    main = data['main']
    wind = data['wind']
    weather = data['weather'][0]

    print(f"Ville: {city}")
    print(f"Température: {main['temp']}°K")
    print(f"Pression: {main['pressure']} hPa")
    print(f"Humidité: {main['humidity']}%")
    print(f"Vitesse du vent: {wind['speed']} m/s")
    print(f"Description: {weather['description']}")
else:
    print('Erreur:', response.status_code)
```

Questions :

- **Dans cette ligne : response.status_code == 200 , que représente le 200 ?**
- **Faites des recherches et donnez les autres codes de statut HTTP qui existent pour les API.**
- **Quelles sont les informations obligatoires pour faire une requête à l'API OpenWeatherMap ?**
- **Comment géreriez-vous les erreurs si l'API OpenWeatherMap retourne un code de statut 401 (Unauthorized) ?**
- **Modifiez le code pour convertir la température de Kelvin en Celsius avant de l'afficher.**

For you :

> Les codes de statut HTTP permettent aux clients et aux serveurs de communiquer de manière standardisée sur l'état des requêtes et des réponses, facilitant ainsi la gestion des erreurs et des redirections dans les applications utilisant des API.

Partie 3 : Envoi de données avec une requête POST

1. **Introduction théorique**
   - La différence entre GET et POST ?
   - Comment utiliser des requêtes POST pour envoyer des données à un serveur ?
2. **Exemple pratique avec une API de test**
   - Utiliser l'API jsonplaceholder pour envoyer une requête POST

```
url = 'https://jsonplaceholder.typicode.com/posts'
payload = {
    "title": "foo",
    "body": "bar",
    "userId": 1
}

response = requests.post(url, json=payload)
if response.status_code == 201:
    print('Données envoyées avec succès:', response.json
())
else:
    print('Erreur:', response.status_code)
```

Questions :

- **Expliquez le code ci-dessous et que signifie API jsonplaceholder ?**

- **Pourquoi serait-il important de vérifier le code de statut de la réponse après une requête POST ?**
- **Que signifie un payload ?**
- **Que se passe-t-il si vous envoyez une requête POST sans le payload approprié à l'API jsonplaceholder ?**

Partie 4 : Manipulation des données et gestion des erreurs

1. **Vérification et traitement des erreurs :**

- Dans cette partie on va gérer les erreurs lors de l'exécution des requêtes HTTP avec la bibliothèque `requests` en Python. Le but est de vous assurer que votre programme peut gérer proprement les erreurs qui peuvent survenir lors de l'envoi de requêtes HTTP.

Questions :

- **Pour le code ci-dessous, rajoutez des commentaires pour chaque ligne, testez le code et analysez les résultats obtenus.**
- **Pourquoi utilise-t-on ce code ?**

```
try:
    response = requests.get(url)
    response.raise_for_status()
    data = response.json()
except requests.exceptions.HTTPError as err:
    print(f'Erreur HTTP: {err}')
except requests.exceptions.RequestException as err:
    print(f'Erreur de requête: {err}')
```

- **Comment pourriez-vous améliorer le code pour gérer des données JSON mal formées dans la réponse ?**
- **Écrivez une fonction qui prend en paramètre une ville et retourne les informations météorologiques de cette ville.**

2. **Manipulation des données**
    - Vous êtes invité à manipuler les données obtenues d'une requête HTTP en filtrant et en affichant uniquement certaines informations

spécifiques, testez et expliquez le rôle du code :

```
if response.status_code == 200:
    data = response.json()
    main = data.get('main', {})
    temperature = main.get('temp')
    humidity = main.get('humidity')

    print(f"Température: {temperature}°K")
    print(f"Humidité: {humidity}%")
else:
    print('Erreur:', response.status_code)
```

Questions de réflexion et de recherche :

1. **Recherchez une autre API publique et écrivez un script Python pour interagir avec elle. Décrivez les étapes suivies.**
2. **Pourquoi est-il important de respecter les limites de taux (rate limits) imposées par les API ? Comment pourriez-vous gérer cela dans votre code ?**

Livrables :

Réponses aux questions.