Requête avec authentification : Twitter

Définitions

La majorité des API demandent au client (vous) de s'authentifier grâce à une **clé d'API** (= mot de passe, parfois appelé "token") - sans quoi l'API renverrait une erreur d'authentification.

Pour utiliser une clé d'API, il est nécessaire de :

- Créer un **compte** chez le fournisseur de l'API (ce compte peut être payant, avec une facturation en fonction de l'utilisation de l'API)
- Inclure la **clé d'API** qui vous a été fournie dans les paramètres (ou headers) de chaque requête

La clé est généralement une longue chaîne de caractères alphanumériques (ex : `5468de-973sg0-97t-06d79657`).

Requête GET avec authentification : la recherche de tweets

Nous utiliserons ici l'**API Twitter** (`'https://api.twitter.com/'`), dont la documentation est disponible ici (https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets). On y trouve des explications sur chaque type d'objet avec lesquels il est possible d'interagir et les endpoints + paramètres correspondant : poster des tweets, retweeter, obtenir des informations sur les utilisateurs, rechercher des tweets, etc.

Ici, nous utiliserons l'endpoint `'1.1/search/tweets.json'` , pour **rechercher des tweets**. Nous voulons par exemple extraire et analyser les tweets les plus récents à propos d'Emmanuel Macron.

Entrée [1]:
```
import requests
```

Entrée [2]:
```
# Clé d'API (personnelle)
BEARER_TOKEN = 'AAAAAAAAAAAAAAAAAAAAAO%2FqOgEAAAAArkiiQQmmxHqWAbuMWQga4yAly
```

Entrée [3]:
```
base_url = 'https://api.twitter.com/'
```

Pour cette recherche de tweets, quel type de requête allons-nous effectuer?

C'est une **requête GET**. Il faut y inclure les **paramètres** et le **header** (informations sur le contexte de la requête, notamment l'API key, pour aider le serveur à adapter sa réponse) requis dans la documentation, dans un format dictionnaire (json).

Entrée [4]:
```
headers = {'Authorization': f'Bearer {BEARER_TOKEN}'}

parameters = {
    'q': 'macron',
    'result_type': 'recent',
    'count': 2
}

url = base_url + '1.1/search/tweets.json'
```

Entrée [5]:
```
response = requests.get(url, headers=headers, params=parameters)

# ou (syntaxe bis)
#response = requests.request("GET", url, headers=headers, params=parameters
```

Entrée [6]:
```
response.status_code
```

Out[6]:
```
200
```

Entrée [7]:
```
response.json()
```

Out[7]:
```
{'statuses': [{'created_at': 'Fri Sep 10 12:13:54 +0000 2021',
   'id': 1436301908382191618,
   'id_str': '1436301908382191618',
   'text': 'RT @EnModeMacaron: On fait le buzz⁉\n\nRT si tu ne voteras
jamais pour Macron et sa bande ❗ https://t.co/moIlbxltdt', (https://t.
co/moIlbxltdt',)
   'truncated': False,
   'entities': {'hashtags': [],
    'symbols': [],
    'user_mentions': [{'screen_name': 'EnModeMacaron',
      'name': ' ❗ EnModeMacaron™ ❗ ',
      'id': 893758854294589440,
      'id_str': '893758854294589440',
      'indices': [3, 17]}],
    'urls': [],
    'media': [{'id': 1436189142992437250,
      'id_str': '1436189142992437250',
      'indices': [91, 114],
      'media_url': 'http://pbs.twimg.com/media/E-5eapJWQAISVf3.jpg',
```

Entrée [10]:
```
# Récupérons le texte du premier tweet
response.json()['statuses'][0]['text']
```

Out[10]:
```
'RT @EnModeMacaron: On fait le buzz⁉\n\nRT si tu ne voteras jamais pour M
acron et sa bande ❗ https://t.co/moIlbxltdt' (https://t.co/moIlbxltdt')
```

BONUS: Requête POST

Si nous voulions maintenant **poster un tweet**, quel type de requête ferions-nous ?

Une **requête POST**, dont les paramètres compredront cette fois le contenu du tweet à poster (toujours au format JSON / dictionnaire). Ces données postées sont parfois appelées **payload** (charge utile).

Notez qu'une seconde clé d'API est necessaire pour effectuer un requête POST sur l'API Twitter, les cellules ci-dessous ne fonctionneront donc pas en l'état, mais la syntaxe est

Entrée [11]:
```
headers = {'Authorization': f'Bearer {BEARER_TOKEN}'}

parameters = {'status': 'Hi Databirdies!'}

url = base_url + '1.1/statuses/update.json'
```

Entrée [12]:
```
response = requests.post(url, headers=headers, params=parameters)

# ou (syntaxe bis):
#response = requests.request("POST", url, headers=headers, params=parameter

response.json()
```

Out[12]:
```
{'errors': [{'code': 220,
   'message': 'Your credentials do not allow access to this resource.'}]}
```

La documentation (https://developer.twitter.com/en/docs/authentication/oauth-2-0/application-only) détaille les autres types de requêtes possibles.

En pratique, nous écrirons **des scripts (boucles, etc) pour envoyer automatiquement des requêtes**. Nous ferons alors attention de ne pas surcharger le serveur pour éviter une surfacturation ou une erreur - c'est ce qu'on appelle le **rate limiting**.

Rate limiting

Le rate liming consiste à ralentir le code pour **limiter le nombre de demandes par seconde envoyées à une API**.

L'API Twitter, par exemple, fixe une limite à 250 recherches de tweets (de 100 tweets max par recherche) toutes les 15 minutes.

La façon la plus simple de ralentir le code est d'utiliser `time.sleep()` . Cette fonction prend en argument un nombre de secondes à attendre avant d'éxécuter la suite du code. Nous pourrons (si besoin) insérer cette commande dans les boucles qui effectueront nos requêtes.

Entrée [14]:
```
import time

print("one")
time.sleep(0.5)
print("two")
```

```
one
two
```

Entrée [ ]:
```
```