

# Présentation de l'outil Podcasts Tracker

Lyna BENYAHIA, Jason MOREL

2023-03-07

# Introduction

- ▶ Constat : il n'existe pas de moyens pour filtrer les résultats par durée sur Spotify
- ▶ Création d'un outil pour chercher des épisodes/émissions par durée
- ▶ Démonstration de l'outil

# Mise en place du projet

## 1. Utilisation de Spotify pour notre projet à l'aide du package Python Spotipy

Fonction principale que nous avons utilisé :

---

```
search(q, limit=10, offset=0, type='track', market=None)
```

searches for an item

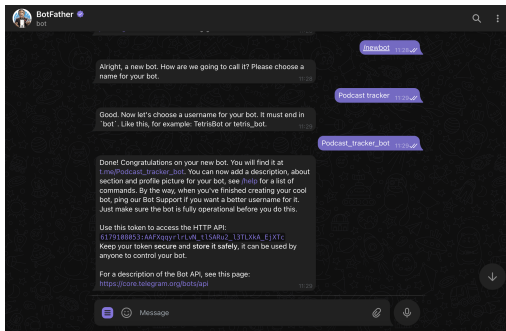
Parameters:

- **q** - the search query (see how to write a query in the official documentation <https://developer.spotify.com/documentation/web-api/reference/search/>) # noqa
- **limit** - the number of items to return (min = 1, default = 10, max = 50). The limit is applied within each type, not on the total response.
- **offset** - the index of the first item to return
- **type** - the types of items to return. One or more of 'artist', 'album', 'track', 'playlist', 'show', and 'episode'. If multiple types are desired, pass in a comma separated string; e.g., 'track,album,episode'.
- **market** - An ISO 3166-1 alpha-2 country code or the string from\_token.

NB : distinction entre émission et épisode, impossible de faire des recherches par durée même depuis l'API

# Mise en place du projet

## 2. Création d'un Bot Telegram pour envoyer les liens directement sur le téléphone de l'utilisateur



- Récupération du Token afin de connecter le code Python au Bot Telegram

# Premier problème rencontré

- Documentation complexe sur l'API Telegram, manque de temps pour créer un bot complètement automatisé  
—> Solution : inputs seront saisis depuis l'ordinateur et seuls les résultats seront envoyés sur Telegram

```
12 # info sur telegram
13 TOKEN_telegram = "6179108053:AAFxqqyrlrLvN_tISARu2_l3TLXka_EjXTc" # obtenu en créant notre bot avec le telegram BotFather
14 #input arbitraire
15 chat_id = 0
16
17 # fonction pour envoyer des messages sur Telegram à un ID donné
18 def send_telegram_message(message, chat_id, TOKEN_telegram):
19     url = f"https://api.telegram.org/bot{TOKEN_telegram}/sendMessage?chat_id={chat_id}&text={message}"
20     response = requests.get(url)
21
22     return response
```

- Fonction pour envoyer un message sur Telegram  
(send\_message.py sur Github)

# Interface graphique

- Création d'une interface graphique à l'aide du package Tkinter (podcast\_finder.py sur Github) :

```
43 MessageVersion.py
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

def submit_form():
    # Récupération des valeurs sélectionnées par l'utilisateur
    type_choice = type_var.get()
    time_choice = time_var.get()
    search_word = search_entry.get()
    chat_id = chat_entry.get()

    ##### PARTIE EPISODE #####
    if type_choice == 1:
        find_episode(search_word, time_choice, chat_id, TOKEN_telegram)

    ##### PARTIE SHOW #####
    if type_choice == 2:
        find_shows(search_word, time_choice, chat_id, TOKEN_telegram)

# Création de la fenêtre et des widgets
root = tk.Tk()
root.title("Choix de podcasts") # CRÉATION DU TITRE DE LA FENÊTRE

chat_label = tk.Label(root, text="Désarrez une conversation avec 'userinfobot' sur Telegram.\nEntrez votre ID ici :").pack(pady=10) # DEMANDE DE L'ID DE L'UTILISATEUR
chat_entry = tk.Entry(root)
chat_entry.pack()

type_label = tk.Label(root, text="Que préférez-vous ?").pack(pady=10) # DEMANDE DE LA PRÉFÉRENCE ENTRE SHOW OU EPISODE
type_var = tk.IntVar()
time1_radio = tk.Radiobutton(root, text="Recevoir une liste d'épisodes uniques", variable=time_var, value=1).pack(anchor="w")
time2_radio = tk.Radiobutton(root, text="Recevoir une liste d'émissions (une émission contient plusieurs épisodes)", variable=time_var, value=2).pack(anchor="w")

time_label = tk.Label(root, text="Quel est le temps d'écoute que vous souhaitez ?").pack(pady=10) # DEMANDE DU TEMPS D'ÉCOUTE
time_var = tk.IntVar()
time1_radio = tk.Radiobutton(root, text="Moins de 5 minutes", variable=time_var, value=1).pack(anchor="w")
time2_radio = tk.Radiobutton(root, text="De 5 à 15 minutes", variable=time_var, value=2).pack(anchor="w")
time3_radio = tk.Radiobutton(root, text="De 15 à 30 minutes", variable=time_var, value=3).pack(anchor="w")
time4_radio = tk.Radiobutton(root, text="De 30 à 45 minutes", variable=time_var, value=4).pack(anchor="w")
time5_radio = tk.Radiobutton(root, text="Plus de 45 minutes", variable=time_var, value=5).pack(anchor="w")

search_label = tk.Label(root, text="Quel type de podcasts souhaitez-vous écouter ?\nEntrez le thème de votre choix :").pack(pady=10) # DEMANDE DU THÈME D'ÉCOUTE
search_entry = tk.Entry(root)
search_entry.pack()

submit_button = tk.Button(root, text="Valider", command=submit_form).pack(pady=10) # CRÉATION DU BOUTON "VALIDER"

root.mainloop() # MISE EN ROUTE DE L'INTERFACE
```

# Traitement des émissions

- ▶ Voir fichier Show\_treatment sur Github
- ▶ Pas de variable indiquant la durée des émissions : il faut l'approximer !
- ▶ A partir des 50 premiers épisodes de l'émission
- ▶ Estimer bornes mini et maxi pour durée de chaque émission
- ▶ Eliminer les intervalles trop importants
- ▶ Uniformiser les intervalles restants
- ▶ Eviter les bugs

# Traitement des épisodes

## ► Episode\_treatment sur Github :

```
# episode_treatment.py
20 # Définit la durée minimale demandée par l'utilisateur
21 def min_for_episode(time_choice):
22     if time_choice == 1:
23         min_duration = 0
24     elif time_choice == 2:
25         min_duration = 300000
26     elif time_choice == 3:
27         min_duration = 900000
28     elif time_choice == 4:
29         min_duration = 1800000
30     elif time_choice == 5:
31         min_duration = 2700000
32
33     return min_duration
34
35 # Définit la durée maximale demandée par l'utilisateur
36 def max_for_episode(time_choice):
37     if time_choice == 1:
38         max_duration = 300000
39     elif time_choice == 2:
40         max_duration = 900000
41     elif time_choice == 3:
42         max_duration = 1800000
43     elif time_choice == 4:
44         max_duration = 2700000
45     elif time_choice == 5:
46         max_duration = 10*1000
47
48     return max_duration
49
50 # trouve une liste d'épisodes selon le thème inscrit, dont la durée est comprise entre la durée minimale et la durée maximale et en français
51 # envoie les noms et liens des épisodes à l'utilisateur sur Telegram
52 def find_episode(search_word, time_choice, chat_id, TOKEN_telegram):
53     min_duration = min_for_episode(time_choice)
54     max_duration = max_for_episode(time_choice)
55
56     super_episode = sp.search(q=search_word, limit=50, type='episode', market='FR')
57     selected_episodes = [episode for episode in super_episode['episodes'] if min_duration <= episode['duration_ms'] <= max_duration and episode['language'] == 'fr']
58     offset = 50
59     while len(selected_episodes) < 4 and offset < super_episode['episodes']['total']:
60         results = sp.search(q=search_word, limit=50, type='episode', market='FR', offset=offset)
61         episodes = results['episodes']['items']
62         selected_episodes += [episode for episode in episodes if min_duration <= episode['duration_ms'] <= max_duration and episode['language'] == 'fr']
63         offset += 50
64
65     if not selected_episodes:
66         messagefinal = "Oops ! Aucun podcast ne correspond à vos critères. Veuillez entrer un autre thème."
67     else:
68         messagefinal = "Voici une liste de plusieurs podcasts correspondant à votre recherche :\n\n"
69         for episode in selected_episodes[:3]:
70             messagefinal += f"{episode['name']}\n{episode['external_urls']['spotify']}\n\n"
71
72     send_telegram_message(messagefinal, chat_id, TOKEN_telegram)
73
74     return messagefinal
```



# Conclusion

- ▶ Pratique que chacun travaille sur une partie différente du code
- ▶ MAIS, cela complique l'assemblage des différentes parties
- ▶ Extensions possibles :
  - ▶ Bot Telegram complètement autonome
  - ▶ Donner plus de résultats sur un même sujet
  - ▶ Garder en mémoire chat\_id
  - ▶ Créer .log des recherches et résultats de l'utilisateur