Documentation technique

SOMMAIRE

1 Algo

Ecouter de la musique:

- On lance le main.py, on arrive sur un menu
- L'option 1 lance la fonction `search_artist` et lance la fonction `lecteur`
- L'option 2 lance la fonction `search_playlist` et lance la fonction `lecteur`
- L'option 3 fait un calcul et lance directement la fonction `lecteur`
- L'option 4 lance directement la fonction `lecteur`

  En fonction de l'option choisie au préalable le lecteur aura différentes options
- Le lecteur lance la fonction `start_progress_bar`
- La fonction `start_progress_bar` lance la fonction `thread_progress_bar`
- La fonction `thread_progress_bar` lance la fonction `printProgressBar`
- La fonction `printProgressBar` lance les fonctions `print_play_my_playlist` , `print_play_one_song`, `print_play_playlist`

En fonction de l'option choisie au début du cursus la fonction `printProgressBar` appelera une des trois fonctions ci dessus. Elle print des données différentes.

Setting Paylist:

- On lance le main.py, on arrive sur un menu
- La cinquième option du menu lance la fonction `munu_playlist_setting`

  Qui est le premier menu du setting (créer une playlist ou en modifier une)
- Si vous avez choisi l'option `modifier une playlist` le `main.py` lancera la fonction `print_all_playlist`
- Le `main.py` attend le retour de `print_all_playlist` puis lance la fonction `know_rules_playlist`
- Le `main.py` attend le retour de `know_rules_playlist` puis lance la fonction `print_rules`
- Si plus haut vous avez choisi l'option `Créer une nouvelle playlist` le `main.py` lancera la fonction `create_new_playlist`

2 Main.py

---

3 Fonction

3.1 lecteur

```
def lecteur(spotifyObject, playlist=None, deviceID= None,
trackSelectionList=None, type_use=None):
```

spotifyObject = L'object spotify créée a l'initialisation du main.py

playlist = Si on joue une playlist ça sera l'item playlist autrement None

deviceID = l'id du device recolter à l'initialisation du main.py

trackSelectionList = Sert à passer un seul titre

type_use = argument passer en fonction de l'utilisation du lecteur ("my_playlist", "play_playlist", "play_one_song")

return = type_use

3.2 start_progress_bar

```
def start_progress_bar(spotifyObject, playlist=None, kill=None,
running=None, type_use=None):
```

spotifyObject = L'object spotify créée a l'initialisation du main.py

playlist = Si on joue une playlist ça sera l'item playlist autrement None

kill = Sert a tuer le thread (boolean)

running = etat du thread (boolean) global variable

type_use = argument passer en fonction de l'utilisation du lecteur ("my_playlist", "play_playlist", "play_one_song")

3.3 thread_progress_bar

```
class thread_progress_bar (threading.Thread):
```

3.3.1

** Initialise le thread**

```
def __init__(self, total, playlist=None, artist=None, album=None,
total_time1=None, name_song=None, spotifyObject=None, ms_now=None,
deviceID=None, type_use=None):
```

total = temps total en MS

playlist = Si on joue une playlist ça sera l'item playlist autrement None

artist = Nom de l'artist

album = Nom de l'album

total_time1 = Temps total de la chanson en minute seconde

name_song = Nom de la chanson

spotifyObject = L'object spotify créée a l'initialisation du main.py

ms_now = Nb de ms seconde qu'a pris la requette

deviceID = l'id du device recolter à l'initialisation du main.py

type_use = argument passer en fonction de l'utilisation du lecteur ("my_playlist", "play_playlist", "play_one_song")

3.3.2

Run le thread

```
def run(self):
```

3.3.3

Met le thread en pause

```
def paused_x(self, current_track, single_track, trackSelectionList=None):
```

current_track = L'item de la chanson en lecture à l'instant T

single_track boolean pour savoir si c'est une chanson ou une playlist

trackSelectionList Si jamais c'est une chanson, lui dire quelle chanson

---

3.4 printProgressBar

```
def printProgressBar (iteration, total, artist, total_time1,
name_song=None,playlist_name=None, prefix = '', suffix = '', decimals = 1,
length = 100, fill = '█', printEnd = "\r", type=None):
```

iteration = Temps en dixieme de seconde à l'instant T

total = Temps total en minute seconde

artist = Nom de l'artiste

total_time1 = Temps total en MS

name_song = Nom de la chanson

playlist_name = Nom de la playlist

prefix = String avant la bar de progression ("progress")

suffix = String à la fin de la bar de progression ("Complete")

decimals = Nb de décimal après la virgule

length = Longueur de la bar de progression

fill = Caractère a print pour l'avancement de la bar de progression

printEnd = String à la fin du print

type = argument passer en fonction de l'utilisation du lecteur ("my_playlist", "play_playlist", "play_one_song")

3.5 running_false

```
def runing_false():
```

Passe runing à l'état False. (ça tue le thread)

3.6 print_play_my_playlist

```
def print_play_my_playlist(playlist_name, name_song, artist, total_time,
prefix, bar, percent, suffix):
```

playlist_name = Nom de la playlist

name_song = Nom du son

artist = Nom de l'artiste

total_time = Temps total

prefix = String avant la bar de progression ("progress")

bar = La bar de progression

percent = Le temps de la chanson à l'instant T

suffix = String à la fin de la bar de progression ("Complete")

3.7 print_play_one_song

```
def print_play_one_song(name_song, artist, total_time, prefix, bar,
percent, suffix):
```

name_song = Nom du son

artist = Nom de l'artiste

total_time = Temps total

prefix = String avant la bar de progression ("progress")

bar = La bar de progression

percent = Le temps de la chanson à l'instant T

suffix = String à la fin de la bar de progression ("Complete")

3.8 print_play_playlist

```
def print_play_playlist(playlist_name, name_song, artist, total_time,
prefix, bar, percent, suffix):
```

playlist_name = Nom de la playlist

name_song = Nom du son

artist = Nom de l'artiste

total_time = Temps total

prefix = String avant la bar de progression ("progress")

bar = La bar de progression

percent = Le temps de la chanson à l'instant T

suffix = String à la fin de la bar de progression ("Complete")

3.9 search_artist

```
def search_artist(spotifyObject, deviceID):
```

spotifyObject = L'object spotify créée a l'initialisation du main.py

deviceID = l'id du device recolter à l'initialisation du main.py

return = Uri de la track

3.10 search_playlist

```
def search_playlist(spotifyObject, deviceID):
```

spotifyObject = L'object spotify créée a l'initialisation du main.py

deviceID = l'id du device recolter à l'initialisation du main.py

return = Uri de la playlist

3.11 print_all_playlist

```
def print_all_playlist(spotifyObject, user):
```

spotifyObject = L'object spotify créée a l'initialisation du main.py

deviceID = l'id du device recolter à l'initialisation du main.py

return = La playlist (l'item)

3.12 know_rules_playlist

Cette fonction me permet de savoir si la playlist apartient ou non à l'utilisateur

```
def know_rules_playlist(the_playlist, user):
```

the_playlist = l'item de la playlist

user = user

return = True or False

3.13 munu_playlist_setting

```
def munu_playlist_setting():
```

return = la commande de l'utilisateur ("1", "2", "3")

3.14 create_new_playlist

```
def create_new_playlist(spotifyObject, user):
```

spotifyObject = L'object spotify créée a l'initialisation du main.py

user = information sur le user courrant

3.15 print_rules

```
def print_rules(spotifyObject, know_rules,the_playlist,user):
```

(Le menu de modification d'une playlist)

spotifyObject = L'object spotify créée a l'initialisation du main.py

know_rules = True or False. C'est pour savoir si la playlist m'appartient ou non

the_playlist = la playlist

user = user