Projet 5 Openclassrooms
Utiliser les données avec l'API d'Open Food Facts

lien github : https://github.com/Blankxx420/Project_5-OC
lien trello : https://trello.com/b/uBhCgrdO/projet-5

1.Introduction du projet

1.1 Contexte :

La startup Pur Beurre travaille connait bien les habitudes alimentaires françaises. Leur restaurant, Ratatouille, remporte un succès croissant et attire toujours plus de visiteurs sur la butte de Montmartre. L'équipe a remarqué que leurs utilisateurs voulaient bien changer leur alimentation mais ne savaient pas bien par quoi commencer. Remplacer le Nutella par une pâte aux noisettes, oui, mais laquelle ? Et dans quel magasin l'acheter ? Leur idée est donc de créer un programme qui interagit avec la base Open Food Facts pour en récupérer les aliments, les comparer et proposer à l'utilisateur un substitut plus sain à l'aliment qui lui fait envie.

1.2 Configuration minimal :

Pour ce projet deux modules ont étés nécessaire à la réalisation il s'agit de ***requests*** et de ***mysql-connector-python***

1.3 Architecture:

- L'architecture de l'application est simple, un fichier ***constant.py*** qui contient les catégories à rechercher et les identifiants de connexion à la base de données.

- Le fichier ***api.py*** il utilise le module ***requests*** recherche les catégories contenue dans ***constant.py*** tri les données à recupérer et nettoie les clés vide avant des les envoyer à l'insertion.

- Le fichier ***db_managment.py*** regroupre toutes les fonctions liée à la création de la base de données à l'ajout des produits et leurs affichages.

- Le fichier ***setup.py*** quand à lui fonctionne avec les classes ***Db_managment*** et ***Api*** contenu dans ***db_management.py*** et ***api.py*** qui initialise la base de données et insert les produits à l'intérieur grâce aux requêtes faites à l'api dans la classe ***Api.***

- Le fichier ***openfoodfact.py*** contient la classe Openfoodfact qui regroupe les fonctions d'interface et de menu en ligne de commande

- le fichier ***main.py*** utilise la classe Openfoofact et sert de lancement de l'application.

2. Les Classes

- **Api** : pour cette classe j'avais besoin du module ***requests*** et des catégories contenu dans ***constant.py*** :

  - ***get_product()*** : Je commence donc par créer une liste(**product**) je parcours les catégories contenue dans **CATEGORIES_LIST** et ensuite je formule ma requête à l'api avec les paramètres nécessaire je passe par un try et except pour éviter les erreurs de connexion ou autre et j'ajoute une liste vide si aucune erreur je reçois une réponse sous forme de dictionnaire json que j'ajoute à ma liste(**product**) et je fini par return ma liste

  - **filter_data()** : je récupère ma liste product dans la variable **product_by_category** et parcours les deux liste(**product_by_category**, **CATEGORIES_LIST** grâce à la fonction **zip()** puis chaque produit dans **product_by_category,** je créer un dictionnaire **attributes** et je récupère les informations donc j'ai besoin avant d'ajouter le dictionnaire à ma liste **self.product_list**

  - **clean_product()** : fonction qui parcourt la liste **self.product_list** vérifie chaque clé du dictionnaire de chaque produit et si une clef et vide supprime le produit entier de la liste

- **Db_managment** : cette classe utilise le module **mysql-python-connector** des identifiant à ma base de données contenue dans **constant.py** la classe **Api** afin de récupérer les données des produit:

  -**__init__()** : Je remplie les paramètres de connexion à ma base de données afin que dès que j'initialise ma classe il se connecte à ma base.

  - **create_database()** : Ouvre le fichier **Database_script.sql** et exécute les requêtes mysql afin de créer la base de donnée.

  - **insert_category**() : Je parcours la liste(**CATEGORIES_LIST**), récupère chaque catégorie et les inserts avec un id autoincrement.

  - **return_category() :** Affiche les catégories et leurs ID de la table **category.**

  -**insert_product()** : Je fais appel à la fonction **clean_product()** pour récupérer les produits une fois nettoyer avant d'insérer les valeurs associé à la table Product et d'y associer les catégories sur leurs ID.

  - **insert_sub()** : Récupère les IDs du produit sélectionné dans **choice_prod** et de l'id du substitut retourné dans la table **substitute**.

- **return_product() :** Affiche dix produits aléatoire de la table **Product** associé à la catégorie sélectionné par l'utilisateur dans **choice_cat**.

- select_product() : Affiche le produit sélectionné par l'utilisateur dans **choice_p**.

- **return_sub()** : Sélectionne le nutri-score du produit sélectionné dans **choice_p**, puis retourne un produit aléatoire de la même catégorie avec un nutri-score supérieur.

- **insert_sub()** : Récupère les IDs du produit sélectionné dans **choice_prod** et de l'id du substitut retourné et les insères dans la table **substitute**.

- **show_favorites():** Affiche les produits enregistrés grâce à la jointure dans la table substitute

- **delete_favorite():** Efface le favoris sélectionné

- **delete_all_fav()** : Efface tout les favoris

- **Openfoodfact** : Cette classe à pour but principale de réaliser l'interface en ligne de commande.

- **home_menu()**: fonction à choix pour l'utilisateur du menu qu'il veut choisir

- **home()** : Fonction du menu principale avec tous les appels de fonction liée à la base de donnée.

- **menu_fav()** : Fonction du menu des favoris

- **choice_sub()** : Menu du choix du substitut suivant la réponse :
    - Oui : Appelle la fonction **return_sub()**
    - Non : Appelle la fonction **home()**

- **keep_substitute()** : Menu du choix de garder ou non le substitut retourné :
    - Oui : Appelle la fonction **insert_sub()** et indique que le produit est bien sauvegardé
    - Non : Appelle la fonction **home()**

- show_fav(): menu pour accéder au favoris dans la table **substitute** :
    - demande à l'utilisateur s'il souhaite accéder au favoris :
        - oui : Appelle la fonction **show_favorites()**
        - non : Appelle la fonction **home()**

- deletes_favorites() : fonction qui permet à l'utilisateur de choisir s'il souhaite supprimer tous les favoris