Ce projet est disponible sur Github à l'adresse:

https://github.com/mehdi-monfort/-P5-OpenFoodFacts

1. Ma démarche:

L'idée était d'apprendre à faire les requêtes avec la bases de données de l'OpenFoodFact, de les utilisées dans un code python pour les afficher sur la console. J'ai commencé par cela.

Puis est crée la database en python (qui utilise un fichier SQL disposant des différentes tables), avec l'aide de la documentation. J'ai, par la suite, rempli les tables à l'aide de la commande « .get », testé les différentes requêtes d'affichage nécessaire au programme.

J'ai terminé par la « main » où est crée le menu et d'où les différents appels sont fait pour l'affichage console.

1.1. Constitution de l'API :

Le programme est constitué de :

Un fichier config.py avec la configuration de la base de données (modifiable par l'utilisateur)
Un fichier main.py contenant le menu de l'application et ses différent choix.
Un dossier Database contenant les fichiers :
  create.py pour la suppression et création de la base de données
  filling.py permettant de remplir la base de donées
Un dossier SQL contenant les fichiers :
  table.py avec le code SQL des différentes tables
  request.py avec les différentes requêtes nécessaire à l'application

2. Difficultés :

Ma difficultés principale est de mettre le code en orienté objet, il me resterai la main à améliorer.

Mon mentor m'a aider à débloquer la situation grâce à ses explications ce qui m'a facilité la tâche par la suite et devrait se ressentir sur la suite de la formation.

J'ai donc dû refaire mon code pour l'orienté objet et également pour y insérer les requêtes directement en python sans passé par un fichier SQL comme c'est le cas pour mes tables.

j'ai également dû travailler sur mes « cursor.execute » qui ne se trouvait pas tous dans le même fichier et qui était faites, en partie, dans la main.

2.1. Améliorations possibles :

Modifier la main pour l'orienté objet avec une classe disposant d'une méthode pour chaque menu.
Un fichier avec les différents messages d'erreur (ex : la réponse n'est pas un integer).
Utilisation de Flask pour en faire une application web, cela permettrait de se rapprocher d'une demande client. Interface graphique ?

3. PEP8 et vérification :

Après utilisation de « flake8 » à la racine de l'api, la principale erreur ressorti depuis flake8 est la longueur de ligne qui dépasse des 79 caractères.