Mini projet
Implantation d'éoliennes en Eure-et-Loir

Le département de l'Eure-et-Loir décide de développer son parc éolien à l'horizon 2025.
Pour cela un bureau de la préfecture est chargé de définir les lieux potentiellement éligibles pour accueillir un futur parc éolien, et il fait appel à vous pour le faire de façon automatique grâce à un script Python.

Pour être éligible, la commune devra avoir moins de 1000 habitants et être située à une altitude supérieure à 200 mètres[1]. Il y aura maximum 8 parcs de créer. Si le nombre de communes éligibles est supérieur à 8, ce seront celles qui ont la plus grande superficie qui seront retenues.

Pour cela, le bureau décide de référencer toutes les communes du département et d'en extraire les communes éligibles. Il dispose d'un fichier au format csv , récupéré sur data.gouv.fr[2]. Malheureusement ce fichier contient toutes communes de la région Centre val de Loire. Et des renseignements inutiles pour la problématique.

A la fin de ce mini-projet, vous devrez rendre un code fonctionnel, correctement documenté en accompagnés de ses tests unitaires.

Vous rendrez également tous les modules python qui vous auront permis de réaliser vos tests.

1 Ces critères, bien que pertinents, sont complètement fictifs, de même que le projet de développement du parc éolien. Tout ceci n'est qu'un prétexte pour vous faire manipuler des données ;-)

2 https://www.data.gouv.fr/fr/search/?q=contours+g%C3%A9ographiques+des+communes+r%C3%A9gion+centre

I. Importation des données

1) Analyse des données

a) En observant le fichier communes_2016.csv, quelles sont les colonnes qui doivent être conservées ?

b) Quelles structures de données vous semble la plus adaptée pour traiter le problème ?

c) Écrivez ce que doit donner l'importation de la première ligne du tableau communes_2016.csv

2) Import du fichier

La fonction suivante permet d'importer un fichier au format csv sous forme d'un dictionnaire :

```
def import_fichier_csv(nom_fichier):
    """
    TODO
    """

    liste_enregistrements = []

    # Ouverture du fichier
    with open(nom_fichier, 'r', newline='', encoding="utf-8") as fichier_csv:
        lecteur_csv = csv.DictReader(fichier_csv, delimiter=";")

        # Lecture de chaque ligne et conversion en dictionnaire
        # (par défaut, la lecture retourne OrederedDict qu'on ne pourrait pas
        # modifier)
        for ligne in lecteur_csv:
            liste_enregistrements.append(dict(ligne))

    return liste_enregistrements
```

a) Ouvrir les modules python nommés « projet_eolienne_28.py » et « test_projet_eolienne_28.py » qui vous ont été fournis.

b) Compléter la documentation de la fonction et vérifier que les tests ne passent pas.

c) Compléter le code de la fonction et vérifier que les tests passent.

d) À la fin du module, ajouter le code permettant d'importer le fichier « communes_2016.csv » et vérifier le fonctionnement en affichant quelques champs de quelques lignes et en les comparant au contenu du fichier CSV.

Une fois les vérifications effectuées, commenter les « print ».

II. Préparation des données

1) Sélection des champs

Avant de s'intéresser à proprement parler au problème éolien, il faut modifier les données pour ne garder que les champs pertinents déterminés à la question I.1.a.

a) Dans un module de test nommé « test1.py », saisir le code suivant et observer le résultat obtenu :

```
d1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
d2 = {}
for cle, valeur in d1.items():
    if cle == 'a' or cle == 'c':
        d2[cle] = valeur
print(d1)
print(d2)
```

b) Modifier ce code pour que les clés à conserver soient présentées sous forme de liste.

c) Dans le module « projet_eolienne_28.py », documenter le squelette de la fonction qui retourne une liste des communes avec uniquement les champs que vous avez choisis à la question I.1.a.

Le squelette de la fonction est le suivant :

```
def selection_champs(liste_enregistrements, liste_champs):
    """
    TODO
    """

    liste_reduite = []

    # TODO

    return liste_reduite
```

d) Dans le module « test_projet_eolienne_28.py », ajouter les tests associés à cette fonction, et vérifier qu'ils ne passent pas.

e) Compléter le code de la fonction et vérifier que les tests passent.

f) À la fin du module, ajouter le code permettant de ne garder que les champs choisis à la question I.1.a, et vérifier le fonctionnement en affichant quelques lignes.

Une fois les vérifications effectuées, commenter les « print ».

2) Sélection des communes d'Eure-et-Loir

Il faut maintenant ne garder que les communes d'Eure-et-Loir.

a) Dans un module de test nommé « test2.py », saisir le code suivant et observer le résultat obtenu :

```
liste1 = [{'a': 1, 'b': 10, 'c': "40", 'd': "4"},
          {'a': 2, 'b': 10, 'c': "30", 'd': "5"},
          {'a': 3, 'b': 20, 'c': "30", 'd': "6"},
          {'a': 4, 'b': 20, 'c': "30", 'd': "4"}]

liste2 = []

for enregistrement in liste1:
    if enregistrement['c'] == "30":
        liste2.append(enregistrement)

print(liste1)
print(liste2)
```

b) Modifier ce code pour ne garder que les enregistrements pour lesquels 'c' = "40" et 'a' > 2.

c) Modifier ce code pour ne garder que les enregistrements pour lesquels 'c' = "40" et 'd' > 4.

d) Dans le module « projet_eolienne_28.py », documenter le squelette de la fonction qui retourne une liste des communes d'un département choisi.

Le squelette de la fonction est le suivant :

```
def selection_departement(liste_communes, departement):
    """
    TODO
    """

    liste_communes_departement = []

    # TODO

    return liste_communes_departement
```

e) Dans le module « test_projet_eolienne_28.py », ajouter les tests associés à cette fonction, et vérifier qu'ils ne passent pas.

f) Compléter le code de la fonction et vérifier que les tests passent.

f) À la fin du module, ajouter le code permettant de ne garder que les communes d'Eure-et-Loir, et vérifier le fonctionnement en affichant quelques lignes.

Une fois les vérifications effectuées, commenter les « print ».

III. Choix des communes retenues

1) Sélection des communes répondant aux critères de sélection

Comme dit en introduction, il faut que les communes aient une altitude supérieure à 200m et moins de 1000 habitants.

a) Dans le module « projet_eolienne_28.py », documenter le squelette de la fonction qui retourne une liste des communes répondant aux critères.

Le squelette de la fonction est le suivant :

```
def selection_criteres(liste_communes, altitude_min, population_max):
    """
    TODO
    """

    liste_communes_criteres = []

    # TODO

    return liste_communes_criteres
```

b) Dans le module « test_projet_eolienne_28.py », ajouter les tests associés à cette fonction, et vérifier qu'ils ne passent pas.

c) Compléter le code de la fonction et vérifier que les tests passent.

d) À la fin du module, ajouter le code permettant de ne garder que les communes répondant aux critères, et vérifier le fonctionnement en affichant quelques lignes.

Une fois les vérifications effectuées, commenter les « print ».

e) Combien de communes répondent aux critères ?

2) Tri des données

Comme vous venez de le voir, il y a plus de 8 communes éligibles. Il faut donc ne garder que les communes ayant la plus grande superficie, et pour faciliter cette tâche, il convient de trier les données.

a) Dans un module de test nommé « test3.py », saisir le code suivant et observer le résultat obtenu :

```
liste = [{'a': 2, 'b': 10, 'c': "40", 'd': "4"},
         {'a': 1, 'b': 30, 'c': "30", 'd': "51"},
         {'a': 7, 'b': 20, 'c': "30", 'd': "6"},
         {'a': 4, 'b': 20, 'c': "30", 'd': "42"}]

for element in liste:
    print(element)
print()

liste.sort(key = lambda element: element['a'])

for element in liste:
    print(element)
print()
```

b) Modifier ce code pour que le tri se fasse sur le champ ‘b’.

c) Modifier ce code pour que le tri se fasse sur le champ ‘d’.

d) Dans le module « projet_eolienne_28.py », documenter le squelette de la fonction qui trie une liste de dictionnaire sur un critère fourni. À votre choix, le tri peut se faire en place, ou la fonction peut retourner une liste trier.

Le squelette de la fonction est le suivant :

```
def trier_liste_communes(liste_communes, champ, decroissant=True):
    """
    TODO
    """

    # TODO
    pass
```

b) Dans le module « test_projet_eolienne_28.py », ajouter les tests associés à cette fonction, et vérifier qu’ils ne passent pas.

c) Compléter le code de la fonction et vérifier que les tests passent.

d) À la fin du module, ajouter le code permettant d’afficher les communes retenues.

IV. Export du résultat

Vous avez désormais toutes les communes éligibles, il vous reste à exporter dans un fichier CSV les 8 communes retenues.

La fonction suivante permet d'enregistrer dans un fichier au format csv les éléments d'une liste contenant des dictionnaires :

```
def export_fichier_csv(nom_fichier, liste_enregistrements, separateur=";"):
    """
    TODO
    """

    with open(nom_fichier, 'w', newline='', encoding="utf-8") as csvfile:
        writer = csv.DictWriter(csvfile,
                                fieldnames=liste_enregistrements[0].keys(),
                                delimiter=separateur)

        # Ecriture de la première ligne du fichier CSV avec les noms des champs
        writer.writeheader()

        # Export de tous les enregistrements de la liste
        for enregistrement in liste_enregistrements:
            writer.writerow(enregistrement)
```

b) Dans le module « projet_eolienne_28.py », documenter la fonction.

c) À la fin du module, ajouter le code permettant d'exporter un fichier nommé « communes_retenues.csv » contenant uniquement les communes retenues.

Vérifier la validité du traitement en consultant le contenu du fichier généré.