DS n° 02 – Algorithmes gloutons

- Faire tous les exercices dans un fichier NomPrenom.py à sauvegarder,
- mettre en commentaire l'exercice et la question traités (ex : # Exercice 1),
- ne pas oublier pas de commenter ce qui est fait dans votre code (ex : # Je crée une fonction pour calculer la racine d'un nombre),
- il est possible de demander un déblocage pour une question, mais celle-ci sera notée 0,
- il faut vérifier avant de partir que le code peut s'exécuter et qu'il affiche les résultats que vous attendez. Les lignes de code qui doivent s'exécuter sont décommentées.

Contexte

L'objectif de cette étude est de déterminer l'itinéraire optimal permettant à un camion de livraison de faire sa tournée entre les villes Amiens, Bayonne, Caen, Dax et Évry.

L'objectif est de réaliser un programme en Python qui génère cet itinéraire.

1 Importer les distances entre les villes

Un fichier `distances.csv` est donné dans le dossier `Ressources`, il contient les distances entre les villes au format `csv`.

	Amiens	Bayonne	Caen	Dax	Évry
Amiens	0	910	250	877	269
Bayonne	910	0	791	53	788
Caen	250	791	0	723	350
Dax	877	53	723	0	748
Évry	269	788	350	748	0

Question 1 **Proposer** un script python qui permet de lire le fichier `csv` et de créer une liste `distances` contenant les distances entre les villes. **Afficher** `distances[1][0]` et `distances[0][1]` qui doivent renvoyer 910.

Remarque, si vous n'avez pas réussi cette question, vous pouvez utiliser la liste (simplifiée) suivante :

```
distances=[[0,10,2,9,3],[10,0,8,1,7],[2,8,0,5,4],[9,1,5,0,6],[3,7,4,6,0]]
```

Question 2 **Créer** une liste `villes` contenant le nom des 5 villes dans l'ordre alphabétique. Puis **coder** une fonction `calcul_distance(ville1,ville2)` qui renvoie la distance entre deux villes. Ainsi, `calcul_distance('Bayonne','Dax')` renvoi 53km (ou 1 avec la liste simplifiée).

Le code suivant permet de déterminer parmi les villes de la liste `sous_groupe_villes` la plus proche de la ville `ville_base`.

```
def plus_proche(ville_base,sous_groupe_villes):
    plus_proche=sous_groupe_villes[0]
    dist_plus_proche=calcul_distance(ville_base,plus_proche)
    for ville in sous_groupe_villes:
        if ..................................................:
            dist_plus_proche=.........................
            plus_proche=.........................
    return plus_proche

sous_groupe_villes=['Amiens','Bayonne','Caen']

print(plus_proche('Dax',sous_groupe_villes))
```

Question 3 **Recopier** et **compléter** ce code à la suite de votre script.
`plus_proche('Dax',sous_groupe_villes)` affiche `Bayonne`.

Le code suivant permet de générer la tournée optimale qui parcours toutes les villes de la liste `villes_a_visiter` en partant de la ville `depart`.

```
def generer_tournee(depart,villes_a_visiter):
    visitees = [depart]
    courant = depart

    # tant qu'il reste des villes non visitées
    while len(visitees) < len(villes_a_visiter):
        non_visitees = [v for v in villes_a_visiter if v not in visitees]

        # choix glouton : prendre la ville la plus proche
        prochain = plus_proche(courant,non_visitees)

        # la prochaine ville va être visitée
        ..................................
        ..................................

    # retour au point de départ
    visitees.append(depart)
    return visitees

tournee = generer_tournee('Amiens',villes)

print(tournee)
```

Question 4 **Recopier** et **compléter** ce code à la suite de votre script.

Question 5 **Coder** une fonction `longueur_tournee(tournee)` qui à partir de la tournée générée par la question précédente, calcule le nombre total de kilomètres parcourus.

FIN

Correction

Question 1

```
file=open('distances.csv','r')
contenu=file.read()
file.close()

lignes=contenu.split('\n')
distances=[]
for ligne in lignes[:-1]:
    data=ligne.split('&')
    data=[int(i) for i in data]
    distances.append(data)

#distances=[[0,10,2,9,3],[10,0,8,1,7],[2,8,0,5,4],[9,1,5,0,6],[3,7,4,6,0]]

print(distances[0][1])
```

Question 2

```
villes=['Amiens','Bayonne','Caen','Dax','Evry']

def calcul_distance(ville1,ville2):
    return distances[villes.index(ville1)][villes.index(ville2)]

print(calcul_distance('Bayonne','Dax'))
```

Question 3

```
def plus_proche(ville_base,sous_groupe_villes):
    plus_proche=sous_groupe_villes[0]
    dist_plus_proche=calcul_distance(ville_base,plus_proche)
    for ville in sous_groupe_villes:
        if calcul_distance(ville_base,ville)<dist_plus_proche:
            dist_plus_proche=calcul_distance(ville_base,ville)
            plus_proche=ville
    return plus_proche

sous_groupe_villes=['Amiens','Bayonne','Caen']

print(plus_proche('Dax',sous_groupe_villes))
```

Question 4

```python
def generer_tournee(depart,villes_a_visiter):
    visitees = [depart]
    courant = depart

    # tant qu'il reste des villes non visitées
    while len(visitees) < len(villes_a_visiter):
        non_visitees = [v for v in villes_a_visiter if v not in visitees]

        # choix glouton : prendre la ville la plus proche
        prochain = plus_proche(courant,non_visitees)

        visitees.append(prochain)
        courant = prochain

    # retour au point de départ
    visitees.append(depart)
    return visitees

tournee = generer_tournee('Amiens',villes)

print(tournee)
```

Question 5

```python
def longueur_tournee(tournee):
    total = 0
    for i in range(len(tournee) - 1):
        total += calcul_distance(tournee[i],tournee[i+1])
    return total

longueur = longueur_tournee(tournee)

print(longueur)
```