Interrogation d'Informatique n°3B

Mercredi 14/03/2024
Durée : 45 minutes

Consignes

- Les scripts doivent être correctement indentés, en mettant en valeur l'indentation à l'aide d'une barre verticale.
- **Le crayon à papier ne sera pas corrigé.**
- **L'usage de la calculatrice est interdit.**

Nom :		**Prénom :**	

Présentation (écriture, propreté, marqueurs d'indentation du code, ...) : /1

Exercice 1 | Fichiers : Extrait du sujet d'Agro-Véto Modélisation 2021 (sur 11 points)

Une annexe présente en fin d'exercice rappelle certaines commandes utiles pour la résolution de cet exercice.

On considère une maladie infectant une population d'individus. On dispose d'un fichier texte `data.txt`, composé de deux lignes, où figurent des données sous la forme suivante :

- la première ligne du fichier contient le nombre d'individus infectés qui sont morts, pour chaque année ;
- la seconde ligne contient le nombre d'individus infectés qui ont survécu, pour chaque année.

Par exemple, si l'on considère une version simplifiée avec seulement sept années, le fichier data.txt est constitué du texte suivant :

283	133	47	30	21	16	13
1000	855	798	760	732	710	692

L'objectif de cette partie est d'extraire ces informations pour créer la liste `Lnorm` qui contient, pour chaque année, le nombre d'individus décédés divisé par le nombre de survivants : `Lnorm = [283/1000 , 133/855 , 47/798, ...]`

1. /3 Écrire une fonction d'entête `extraction_ch()` qui renvoie une liste composée de deux chaînes de caractères : une correspondant à la première ligne de `data.txt` et l'autre à la seconde. *Cette fonction devra, au moins, ouvrir le fichier* `data.txt`, *le lire puis le fermer. Pour l'ouverture, la lecture et la fermeture d'un fichier, on pourra utiliser les syntaxes décrites dans l'annexe (en fin de l'exerice).* Par exemple, sur le contenu simplifié ci-dessus, `extraction_ch()` renvoie :

```
["283 133 147 30 21 16 13\n","1000  855  798  760  732  710  692"]
```

```
def extraction_ch():
    f = open("data.txt","r")
    ligne1 = f.readline()
    ligne2 = f.readline()
    f.close()
    return [ligne1,ligne2]
```

2. /3 Compléter la fonction suivante d'entête `ch_vers_list(ch)` prenant en entrée une chaîne de caractère (de même forme que celles renvoyées par `extraction_ch`) et qui renvoie la liste de nombres correspondant; ces nombres seront de type flottant. Par exemple, `chVersList('1000 855 798 760 732 710 692')` renvoie : `[1000.0,855.0,798.0,760.0,732.0,710.0,692.0]`
On expliquera, après avoir complété le code, le rôle de la variable `sh`.
Si besoin, on rappelle que `float("6.5\n")` *renvoie* `6.5`.

```
def ch_vers_list(ch):
    L = [ ]
    n = len(ch)
    sh = " "
    for k in range(0,n):
        if ch[k] != " ":
            sh = sh + ch[k]
        else :
            L.append(float(sh))
            sh = " "
    L.append(float(sh))
    return L
```

On crée une liste initialement vide

On ajoute à L le nombre correspondant au mot sh

On efface le contenu de sh pour recommencer avec le nombre suivant

Pour ajouter le dernier nombre (après le dernier espace).

Lycée Camille GUÉRIN – Poitiers

(Explication du rôle de la variable sh)

La variable sh permet de lire au fur et à mesure du parcours de la chaîne de caractères ch chaque « mot » (jusqu'à tomber sur un espace) donc ici chaque nombre à ajouter à la liste L

3. [/3] Écrire une fonction d'entête `division(L1, L2)` prenant en entrée deux listes de nombres flottants. Cette fonction renvoie le booléen **`False`** si les deux listes ne sont pas de la même longueur; sinon, elle renvoie la liste contenant les divisions terme à terme des éléments de `L1` par ceux de `L2` (on suppose qu'il n'y a pas de terme nul dans `L2`). Par exemple, `division([2,10.5,6,4],[2,2,3,1])` renvoie : `[1.0, 5.25, 2.0, 4.0]`.

```
>_
def division (L1, L2) :
    n1 = len (L1)
    n2 = len (L2)
    if n1 != n2 :
        return False
    else :
        D = [ ]
        for i in range (n1) :
            D. append (L1[i] / L2[i])
        return D
```

4. [/2] Proposer alors une suite d'instructions (à rentrer dans la console) qui crée la liste `Lnorm` précédemment définie.

```
>>> L = extraction_ch( )
>>> L1 = ch_vers_list ( L[0] )
>>> L2 = ch_vers_list ( L[1] )
>>> L norm = division (L1, L2)
```

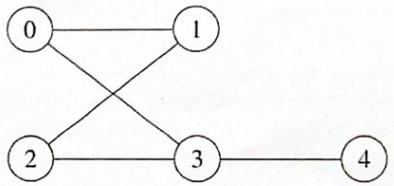
Annexe : rappels de syntaxes Python

- `f = open("monfichier.txt", "r")` ouvre le document intitulé `monfichier.txt` (en mode lecture); le fichier est alors désigné par la variable `f`.
- `f.read()` lit l'intégralité du contenu du fichier `f` et le renvoie sous forme d'une chaine de caractères.
- `f.readline()` lit la première ligne du fichier `f` et la renvoie sous forme d'une chaine de caractères. Ré-exécuter `f.readline()` lit alors la deuxième ligne; et ainsi de suite.
- `f.close()` ferme le fichier `f`.
- Le caractère `\n` (de longueur 1) désigne le passage à la ligne.

Exercice 2 | Graphes, matrices et dictionnaires (sur 8 points)

On suppose, dans tout l'exercice, que la librairie `numpy` *est importée avec la commande* **`import`** `numpy` **`as`** `np`.

On considère le graphe $\mathcal{G}$ suivant et on note A la matrice d'adjacence de $\mathcal{G}$.



1. [/1.5] Implémenter le graphe $\mathcal{G}$ à l'aide d'un dictionnaire `G`.

```
G = { }
G[0] = [1, 3]
G[1] = [0, 2]
G[2] = [1, 3]
G[3] = [0, 2, 4]
G[4] = [3]
```

2. /1.5 Déterminer la matrice A.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

3. (a) /1.5 Par lecture du graphe, donner (en listant leurs sommets) les chaînes de longueur 3 reliant les sommets 2 et 3. Combien y en a-t-il?

2 - 1 - 0 - 3
2 - 1 - 2 - 3
2 - 3 - 0 - 3
2 - 3 - 4 - 3
2 - 3 - 2 - 3

Il y a 5 chemins de longueur 3 reliant les sommets 2 et 3.

(b) /3 Créer une fonction d'en-tête `puissance_mat(M,k)` qui retourne le tableau correspondant à la matrice M^k, étant donné une matrice carrée M et un entier naturel k.

```
def puissance_mat(M,k):
    n,p = M.shape
    P = np.eye(n)
    for _ in range(k):
        P = np.dot(P,M)
    return P
```

(c) /1.5 On suppose que l'on a saisi la matrice A et on considère les instructions :

```
B = puissance_mat(A, 3)
n = B[2 , 3]
print(n)
```

Compléter ces instructions pour qu'elles permettent l'affichage du nombre trouvé à la question **3.a)**.