Examen Python SRI 2023-2024

Durée 1h30, Documents et machines autorisées, barême indicatif

Les parties sont indépendantes

A l'intérieur d'une partie, vous pouvez utiliser les fonctions définies précédemment, même si vous ne les avez pas codées (mais bien sur, ce sera plus dur à tester dans le notebook).

Partie 1

On va considérer des étudiants qui ont des notes à des matières, et chaque matière a un coefficient dans la moyenne finale.

Par exemple: (en listant les coeffs et les noms des matières dans le même ordre que les notes)

[1]:
```
all_etudiants = ['superdupont', 'asterix', 'tyrion']
coeffs = [5,2,2]
notes_par_matière = [[15, 2, 5], [8, 5, 20], [19, 15, 13]]
matières = ["poterie","anglais","c++"]
```

1.1 Ecrire une fonction **notes_par_etud** qui prend en entrée les notes par matières et renvoie une liste des listes de notes par étudiant.

Sur l'exemple on doit avoir:

```
notes_par_etud(notes_par_matière)
[[15, 8, 19], [2, 5, 15], [5, 20, 13]]
```

1.2 Ecrire une fonction **make_dict** qui fait un dictionnaire des notes par étudiants avec leur nom, à partir de la liste d'étudiants et des notes par matière. Sur l'exemple, elle renverrait

```
{'superdupont': [15, 8, 19], 'asterix': [2, 5, 15], 'tyrion': [5, 20, 13]}
```

1.3 Ecrire une fonction **moyenne_1** qui fait la moyenne d'une liste de notes (pour 1 etudiant seulement) pondérée par une liste de coefficients

Ecrire une fonction **moyenne** qui prend en entreé un dictionnaire comme la sortie de la fonction make_dict et calcule un dictionnaire des moyennes par étudiant.

Ecrire une fonction **ajout_moyenne** qui prend en entrée aussi un dictionnaire de notes par étudiants et ajoute la moyenne comme dernière valeur.

[1]:
```
# sur l'exemple:
{'superdupont': [15, 8, 19, 14.333333333333334],
 'asterix': [2, 5, 15, 5.555555555555555],
 'tyrion': [5, 20, 13, 10.11111111111111]}
```

[1]:
```
{'superdupont': [15, 8, 19, 14.333333333333334],
 'asterix': [2, 5, 15, 5.555555555555555],
 'tyrion': [5, 20, 13, 10.11111111111111]}
```

1.4 Ecrire une fonction **valide** qui prend en entrée un dictionnaire de notes + moyennes comme au 1.3, et renvoie la liste des étudiants qui remplissent les conditions suivantes:

- au moins 6 à chaque matière
- la moyenne supérieure ou égale à 10

Sur l'exemple, seul superdupont remplit les conditions

1.5 Généraliser la fonction précédente en une fonction **passe_custom** pour qu'elle prenne en argument en plus une liste de fonctions, chacune des fonctions testant une condition différente sur la liste des notes d'un étudiant seul.

Recalculer le résultat précédent avec cette fonction

Partie 2

2.1 Définir une classe Jury qui est initialisé à partir d'une liste de noms d'étudiants, de noms de matières, et de coefficients par matière.

2.2 Définir les méthodes suivantes (ici vous pouvez utiliser les fonctions de la partie 1 à l'intérieur de ces méthodes:

- une méthode add_notes qui prend en paramètre une liste de notes par matière et va stocker dans l'objet un dictionnaire de notes similaires au résultat de la partie 1
- une méthode add_moyenne qui va ajouter la moyenne dans la liste des notes de chaque étudiant
- une méthode qui_passe, qui renvoie la liste des étudiants acceptés (comme partie 1)

[14]:
```
j = Jury(all_etudiants,coeffs,matières)
j.add_notes(notes_par_matière)
print(j.note_dict)
j.add_moyenne()
print(j.qui_passe())
print(j.note_dict)
```

```
{'superdupont': [15, 8, 19], 'asterix': [2, 5, 15], 'tyrion': [5, 20, 13]}
['superdupont']
{'superdupont': [15, 8, 19, 14.333333333333334], 'asterix': [2, 5, 15,
5.555555555555555], 'tyrion': [5, 20, 13, 10.11111111111111]}
```

Partie 3

On va faire en partie la même chose que la partie 1 en partant de zéro avec numpy.

Attention: on ne peut pas réutiliser les fonctions des parties précédentes, et il ne doit y avoir aucune boucle, sinon les questions valent 0

Vous pouvez par contre utiliser des fonctions de la partie 3 pour faire des fonctions qui viennent après dans la partie.

Rappel de quelques méthodes numpy, sur un vecteur a, dont certaines peuvent servir ici :

- a.mean() fait la moyenne de toutes les valeurs; a.mean(axis=0) fait la moyenne selon l'axe 0
- idem pour sum, min, max
- a.T est la transposée de a
- list(a) transforme a en une liste selon la première dimension de a
- a.reshape(new_shape): permet de redéfinir une matrice avec de nouvelles dimensions (si le nb d'éléments correspond), eg un vecteur de 20 valeurs peut etre "reshaped" en une matrice de 5x4 valeurs
- np.concatenate(liste_de_matrices): met bout a bout en ligne des vecteurs/matrices de même nb de colonnes

Pour l'exemple on peut juste convertir l'exemple à base de liste en array

```
par_matière = np.array(notes_par_matière)
coeffs = np.array(coeffs)
```

3.1 Comment peut-on avoir la matrice avec les notes de chaque étudiant par ligne ?

3.2 La fonction suivante va servir à créer une matrice en dupliquant coeffs avec autant de lignes qu'il y a d'étudiants

```
def coeffs_mat(coeffs,nb_etud):
    return np.array(list(coeffs)*nb_etud).reshape(...,...)
```

Quels sont les paramètres que l'on doit donner à reshape pour que cela remplisse l'objectif ?

Sur l'exemple:

```
array([[5, 2, 2],
       [5, 2, 2],
       [5, 2, 2]])
```

3.3 Ecrire une fonction qui renvoie le vecteur des moyennes par étudiant à partir de la matrice de la question 3.1

3.4 (attention : question peu facile) Ecrire une fonction qui renvoie le vecteur des moyennes pondérées par les coefficients, en utilisant le résultat de la fonction 3.3

Sur l'exemple cela donnerait:

```
array([14.33333333,  5.55555556, 10.11111111])
```

3.5 On ne demande pas d'ajouter les moyennes à la matrice, on pourrait le faire par exemple avec

```
def add_moyenne(m):
    mpe = moy_par_etud(m)
    return np.concatenate((m,mpe.reshape(1,3))).T
```

qui renverrait la matrice des notes par étudiants et leur moyenne, sur l'exemple

```
array([[15.        ,  8.        , 19.        , 14.33333333],
       [ 2.        ,  5.        , 15.        ,  5.55555556],
       [ 5.        , 20.        , 13.        , 10.11111111]])
```

A partir de cette matrice:

(a) Comment peut-on exprimer la condition qu'un étudiant doit avoir au moins 5 en poterie ?

(b) Comment peut-on exprimer la condition qu'un étudiant doit avoir au moins 10 de moyenne ?

(c) Comment peut-on exprimer la condition qu'un étudiant doit avoir au moins 6 à toutes les matières ?

Ecrire une fonction qui prend la matrice de notes+moyenne, et qui retourner les lignes de la matrice de note correspondant aux étudiants qui "passent" selon les critères (b) et (c) (comme dans la partie 1)

Ici on doit donc avoir au final

```
array([[15.        ,  8.        , 19.        , 14.33333333]])
```

[NB, pas demandé: on pourrait aussi utiliser np.argwhere pour avoir les indices des étudiants concernés à la place]