Division des enseignements en informatique

TP Gestion d'association

L'objectif de ce TP est de développer une application de gestion des adhésions à un club multisports. L'association propose différentes activités, avec pour chacune un tarif. La cotisation payée par un adhérent est égale à la somme des tarifs des activités auxquels il s'est inscrit. A partir de la troisième activité, une réduction de 20% est accordée.

Les données de l'association sont sotckées dans deux fichiers texte :

— un premier contenant les informations relatives aux activités ;
— un second avec les données sur les adhérents.

Afin de tracer toutes les opérations effectuées avec l'application, nous developperons en fin de TP un module simple d'écriture de fichiers de logs.

Il est rappelé que le code devra être documenté et que les tests unitaires, lorsqu'il seront possibles, devront être prévus. Nous utiliserons pour cela les `doctrings` qui pourront être utilisées par le module `doctest`.

1 Utilisation du fichier des activités

1. Une activité est décrite par un nom et un tarif (par exemple : `"Natation", 100`). Ecrivez une méthode `ajouter_activite(nom, cotisation, chemin_fichier)` permettant d'ajouter une activité (nom au format texte et cotisation de type nombre à virgule) dans le fichier les décrivant. Utilisez correctement le mécanisme des exceptions de manière à ce que l'accès au fichier soit libéré lorsque la méthodes est terminée.
2. Ajoutez une fonction `existe_activite(nom_activite, chemin_fichier)` testant si une activité est présente dans le fichier des activités.
3. Modifiez la méthode de la question 1 de manière à ce qu'elle n'ajoute une activité que si elle n'existe pas déjà.
4. Complétez votre programme avec une méthode de lecture du fichier des activités. Cette méthode retournera un dictionnaire dont les clés seront les noms des activités et les valeurs associées les montants des cotisations.

2 Utilisation du fichier des adhérents

5. De manière similaire, écrire une méthode d'ajout d'un adhérent dans le fichier des adhérents. La liste des activités auxquels l'adhérent est inscrit pourra être écrite sous forme d'une chaîne de caractères en séparant les activités par des " ;".

6. Prévoyez des méthodes de suppression d'un adhérent et de modification de la liste des activités auxquels il est inscrit.
7. Ajoutez une méthode `lire_activites_adherent(nom_ahderent, chemin_fichier)` retournant la liste des activités auxquels l'adhérent en entrée de la fonction est inscrit.

3 Calcul des cotisations

Dans cette partie, nous nous attaquons au calcul des cotisations dues par les adhérents. Dans un premier temps, nous appliquerons une formule simple : cotisation totale = somme des tarifs des activités. Nous tiendrons compte des réductions dans un second temps.

8. Ecrivez une méthode de calcul de la cotisation due par un adhérent à partir du dictionnaire des activités et de la liste des associations auxquelles un adhérent est inscrit.
9. Améliorez la fonction précédente pour appliquer une réduction de 20% aux activités les plus couteuses à partir de la troisième inscription d'un adhérent à une activité.
10. Prévoyez des tests unitaires si ce n'est pas déjà fait.

4 Gestion du journal

Nous souhaitons tracer dans un fichier texte toutes les opérations effectuées avec l'application (ajout d'une activité, ouverture d'un fichier, appel d'une méthode, etc.).

11. Dans un nouveau module (ie. nouveau fichier `.py`), écrivez une méthode `log(nom_fichier, nature , message)` permettant d'écrire dans un fichier log (fichier texte) le message correspondant à l'événement survenu, en précisant sa nature qui peut être :

— `INFO` pour un simple message d'information (fichier ouvert, traitement effectué, etc.) ;
— `DEBUG` pour un message utile au debogage par le développeur ;
— `WARNING` pour la survenu d'un évémenent non attendu mais traité correctement par l'application ;
— `ERROR` pour une erreur entraînant l'arrêt de l'application.

5 Documentation

Cette partie n'est à traiter que si vous avez déjà complété intégralement les parties précédentes.

12. Si vous ne l'avez toujours pas fait, documentez votre code !
13. En vous aidant de la documentation en ligne (http ://www.sphinx-doc.org/en/stable/tutorial.html), installez la librairie `Sphinx` et servez-vous en pour générer la documentation de votre application au format HTML.