ECOLE SUPERIEURE D'INFORMATIQUE DE GESTION

Epreuve Regroupée

Session de novembre 2022

BRANCHE : **ALGORITHMES ET LANGAGE DE PROGRAMMATION (ALP)**

CLASSE : **ESIG1 Classes A et B (Groupes 1, 2 et 3)**

DATE : 31 octobre 2022

NOM : ……

PRÉNOM : ……

PROFESSEUR : Eric BATARD / Mirko STEINLE / Clément VOGT

N° du poste de travail : ESIG-PB.....

N° de clé USB : ……

Modalités

- Durée : 240 minutes.
- Travail individuel.
- Documentation personnelle (livres, papiers) : Autorisée.
  Documentation électronique (clé USB, …) : Autorisée si recopiée avant le début de l'épreuve.
- Tout partage de ressources de votre poste de travail avec le réseau ou toute autre tentative de communication seront considérés comme de la fraude et sanctionnés par la note minimale.
  Cela comprend la présence d'un téléphone portable, montre connectée, lunettes connectées, clé USB ou tout autre dispositif de stockage ou de communication à proximité immédiate de votre place de travail : tout ceci doit être posé à l'endroit indiqué par le/la surveillant-e.

Pour démarrer

- ***Avant*** le début de l'épreuve, votre documentation sur disque externe, clé USB, … doit être recopiée sur **C:\ESIGUsers\Doc**. Pour cela, connectez-vous au réseau sur le poste de travail qui vous a été attribué.
- ***Une fois l'épreuve commencée,*** copiez dans **C:\ESIGUsers** le contenu du dossier réseau qui vous sera indiqué au début de l'épreuve de façon à avoir un répertoire **C:\ESIGUsers\Eléments ER ALP 31-10-22**.
- Ouvrez ce dossier **C:\ESIGUsers\Eléments ER ALP 31-10-22**.
- Ouvrez Thonny puis ouvrez les fichiers indiqués dans l'énoncé. ***N'oubliez pas d'indiquer vos nom et prénom au début des fichiers modifiés.***

Consignes générales

Sur l'organisation

- Lisez tous les documents fournis.
- Complétez les procédures/fonctions demandées de manière à ce qu'elles répondent aux spécifications de l'énoncé.
- Vous rendrez les fichiers correspondants sur la clé USB qui vous sera remise quand vous serez prêt à rendre.

Sur la programmation en Python

- **Interdiction** d'utiliser fonctions prédéfinies `max, min, index, count, sum, mean` ou équivalent dans des modules fournis par python.

- Les éléments suivants sont explicitement autorisés : la fonction `len`, le mot clé `in` dans toutes ses formes, les « compréhensions ». En cas de doute, demandez au responsable d'énoncé.

- Pas de résultats ou de données codés « en dur » : votre code doit pouvoir s'adapter à d'autres jeux de données.

- Les seules variables globales autorisées sont les constantes.

C'est *à vous* de vérifier que les fichiers enregistrés et rendus contiennent bien
la dernière version de votre travail

Quand les deux cas se présentent, le masculin est utilisé dans cet énoncé de façon générique.

Observer les étoiles de la marche dans *TrekStar* !

Contexte

Une compétition de *trekking* (randonnée de plusieurs jours consécutifs) appelée **TrekStar**, regroupe les plus célèbres trekkeurs (c'est le mot officiel en français). Elle fait appel à vous pour comparer les résultats de ces champions.

Eléments fournis dans le fichier `TrekStar.py` [A NE PAS MODIFIER !]

LES DONNEES

Il s'agit de 3 listes, chacune correspondant à un compétiteur et portant le nom de ce compétiteur.

Chacune de ces listes contient des nombres. Il s'agit du nombre de kilomètres parcourus pendant une journée, dans l'ordre chronologique bien sûr. Le premier nombre correspond à la distance parcourue le jour n°0, le second le jour n°1, le troisième le jour n°2, etc.

On conserve la numérotation démarrant à 0 pour plus de simplicité. Notez cependant que, s'il y a 3 nombres dans une liste, cela signifie bien qu'il y a eu 3 jours de compétition même si les numéros de jours vont de 0 à 2.

Importante exception : pour les compétiteurs qui ont abandonné, on *ajoute à la fin* de la liste ***un 0***. Ce nombre supplémentaire est un code qui signifie que le compétiteur a abandonné, qu'il n'est donc pas arrivé à la fin et qu'il n'est plus en course. Dans ce cas particulier, **quand il y a un 0 à la fin, le nombre de jours de compétition active est un de moins que le nombre d'éléments de la liste**. Cela a de l'importance dans certaines questions (mais pas toutes) où figure l'expression « Attention aux abandons ! » vous le rappelera.

CODE FOURNI

Une constante `DIST_TOT_TREK` donne le nombre de kilomètres total du trek (200 km).

Deux fonctions sont fournies (en dehors du `main()` et des fonctions à compléter bien sûr) :

- `arrondi_dixieme()` qui, comme son nom l'indique arrondit son unique argument au dixième
- `arrondi_superieur()` qui, quant à elle, arrondit son unique argument à l'entier supérieur.

On indiquera dans l'énoncé quand utiliser ces deux fonctions.

Remarques générales

Vous compléterez le fichier fourni `TrekStar.py`. ***N'oubliez pas d'indiquer vos nom et prénom au début de ce fichier***.

Il va vous être demandé, à plusieurs reprises dans la Partie A, de programmer certains calculs *dans une fonction (ou procédure) à part*. L'existence d'une telle fonction, définie par vous, fera partie de l'évaluation, même si elle est :

- incomplète = les instructions de la fonction ne font pas tout ce qui est demandé
- voire vide
  - l'en-tête de la fonction doit alors être complet = nom et liste de paramètres pertinents
  - utilisez alors l'instruction `pass` comme unique instruction,
  - ou un `return` adapté pour une fonction, notamment booléenne,
  - et, si possible, appelez cette fonction dans un endroit pertinent.

De plus vous ne recevrez qu'une partie des points si la fonctionnalité est présente dans la fonction `recap()` (cf. Partie B) mais pas dans une fonction (ou procédure) à part comme demandé en Partie A.

Partie A – Préparation d'un récapitulatif

Le but ultime est d'obtenir (afficher) un récapitulatif de diverses caractéristiques des performances des compétiteurs. Pour plus de clarté, cette partie A demande la réalisation de plusieurs fonctions ou procédures en vue de les intégrer au futur récapitulatif (dans la partie B).

Vous pouvez donc programmer ces fonctions et procédures indépendamment les unes des autres, et indépendamment de la partie B.

Pour ***tester*** ces fonctions et procédures, vous pouvez :

- soit introduire un code de test (par exemple un appel) juste après votre fonction
- soit écrire ce code de test dans la fonction `recap()` à compléter (en partie B).

Dans tous les cas, il est impératif que votre code de test soit mis en commentaire dans le fichier que vous rendrez. Ce sera évalué aussi.

QUESTION A1 – distance parcourue

Définir une fonction à part qui calcule la distance totale parcourue à partir d'une liste de distances donnée en paramètre, c'est-à-dire la somme des distances parcourues par jour.

Cf l'annexe pour des exemples de résultat, listes `PICARD` et `SULU`.

QUESTION A2 – toujours en course ou pas

Définir une fonction booléenne à part qui indique si le compétiteur est, oui ou non, toujours en course à partir d'une liste de distances donnée en paramètre.

Un compétiteur est toujours en course s'il n'a pas atteint l'objectif du trek et s'il n'a pas abandonné.

Cf l'annexe pour des exemples de résultat, listes `PICARD` et `SULU`.

QUESTION A3 – vitesse moyenne globale

Définir une fonction à part qui calcule la vitesse moyenne globale à partir d'une liste de distances donnée en paramètre.

La vitesse moyenne globale est simplement la distance totale parcourue (question A1) divisée par le nombre de jours de trek. Attention aux abandons !

Le résultat sera en kilomètres par jour. Il faudra appliquer à ce résultat la fonction `arrondi_dixieme()` fournie pour obtenir une vitesse arrondie au dixième.

Cf l'annexe pour des exemples de résultat, listes `KIRK`, `PICARD` et `SULU`.

QUESTION A4 – distance restant à parcourir

Définir une fonction à part qui calcule la distance restant à parcourir pour une distance donnée (en paramètre) déjà parcourue.

C'est juste la différence entre la distance totale de la compétition et le paramètre !

Cf l'annexe pour des exemples de résultat, liste `PICARD`.

QUESTION A5 – mi-parcours

Définir une fonction à part qui calcule au bout de combien de jours le compétiteur a atteint ou dépassé la moitié de la distance totale, à partir d'une liste de distances donnée en paramètre.

Vous pouvez supposer que le compétiteur a toujours atteint ou dépassé ces 100 km car cette fonction ne devra être appelée que pour des compétiteurs ayant atteint l'objectif de 200 km

Cf l'annexe pour des exemples de résultat, liste `KIRK`.

QUESTION A6 – jour de la meilleure performance

Définir une fonction à part qui détermine le numéro de jour de la meilleure performance d'un compétiteur à partir d'une liste de distances donnée en paramètre.

La meilleure performance est la distance la plus longue parcourue en un jour. Le numéro de jour correspondant est la position de la valeur dans la liste.

Cf l'annexe pour des exemples de résultat, listes `KIRK`, `PICARD` et `SULU`.

QUESTION A7 – évolution de la moyenne

Définir une procédure à part qui affiche l'évolution de la moyenne à partir d'une liste de distances donnée en paramètre.

Pour chaque jour du trek, il faut calculer la distance totale parcourue depuis le début et la diviser par le nombre de jours. Attention aux abandons !

Cf l'annexe pour des exemples de résultat, listes `KIRK`, `PICARD` et `SULU`.

Exemple avec les données de la liste SULU

Tout d'abord la sortie à produire

```
Voici l'évolution de sa moyenne au jour le jour (km/jour) :
au jour 0 : 21.0
au jour 1 : 21.5
au jour 2 : 19.7
au jour 3 : 19.2
au jour 4 : 21.0
```

Pour chaque jour du trek, on a calculé la distance totale parcourue depuis le début et on l'a divisé par le nombre de jours.

Il faudra appliquer au résultat de cette formule brute la fonction `arrondi_dixieme()` fournie pour obtenir une vitesse arrondie au dixième.

Détail des calculs pour les données de `SULU` :

numéro de jour (indice)	distance (valeur)	distance totale (somme partielle)	nombre de jours de trek	moyenne
0	21	21	1	21/1= 21,0
1	22	21+22= 43	2	43/2= 21,5
2	16	43+16= 59	3	59/3= 19,7
3	18	59+18= 77	4	77/4= 19,2
4	28	77+28= 105	5	105/5= 21,0
5	0	*pas pris en compte car code abandon*		

QUESTION A8 – nombre approximatif de jours restants

Définir une fonction à part qui, à partir d'une liste de distances donnée en paramètre, indique au bout de combien de jours un compétiteur toujours en course devrait terminer son trek s'il maintient sa vitesse moyenne globale.

Pour cela il faut calculer la vitesse moyenne globale (question A3) et la distance restant à parcourir (question A4), et ensuite appliquer la formule de calcul brut suivante :

la distance restant à parcourir divisée par la vitesse globale moyenne.

Enfin il faut appliquer au résultat de cette formule brute la fonction `arrondi_superieur()` fournie pour obtenir un nombre de jours entiers.

Cf l'annexe pour des exemples de résultat, liste `PICARD`.

Partie B – Création du récapitulatif

La fonction `recap()` à compléter prend en paramètre une liste et un nom (chaîne de caractères). On va détailler ici la présentation demandée mais vous pouvez consulter les sorties à produire complètes en annexe pour une vue générale.

Grâce aux fonctions demandées en Partie A, mais vous pouvez en définir d'autres en plus, une bonne partie de travail est déjà effectuée !
(Pour rappel : des appels corrects de fonctions restées incomplètes ou vides rapporteront des points.)

Voici les spécifications accompagnées d'un exemple de sortie `en police Courier`. Le récapitulatif doit répondre aux spécifications suivantes :

- Afficher le nom.

  ```
  Parcours de Kirk
  ```

- *Si* la distance totale parcourue (question A1) est égale à la distance totale du trek (`DIST_TOT_TREK`), indiquer en combien de jours elle a été réalisée.
  Indiquer aussi au bout de combien de jours le compétiteur a atteint ou dépassé la moitié de la distance de la compétition (question A5).

  ```
  Kirk a atteint l'objectif en 7 jours
  Kirk a réalisé la moitié de la distance en 4 jours
  ```

- *Sinon*, selon que le compétiteur est toujours en course ou pas (question A2),
  - s'il est toujours en course, indiquer le nombre de kilomètres qu'il a déjà parcourus (question A1) et afficher une approximation du nombre de jours restants (question A8) ainsi que sa vitesse moyenne globale en kilomètre par jour (question A3).

    ```
    Picard est toujours en course après 185 km
    Il reste 15 km qui devraient être parcourus en 1 jour(s) s'il maintient
    sa moyenne qui est de 30.8 km par jour
    ```

  - s'il a abandonné, indiquer le nombre de kilomètres qu'il a parcourus avant d'abandonner (question A1) ainsi que le nombre de jours de trek effectifs.

    ```
    Sulu a abandonné au bout de 105 km et 5 jours
    ```

- Indiquer dans les autres cas (hors course = objectif atteint ou abandon) (question A2) sa vitesse moyenne globale en kilomètre par jour (question A3).

  Exemple avec les données de `KIRK`

  ```
  Ce qui représente une moyenne globale de 28.6 km par jour
  ```

- Dans tous les cas, afficher, au moyen d'une procédure à part, l'évolution de la moyenne à partir d'une liste de distances (question A7).

- Dans tous les cas, afficher le numéro de jour de la meilleure performance et le nombre de kilomètres parcourus ce jour-là (question A6). Notez les deux textes possibles selon le cas.

  (A partir du numéro de jour, vous obtiendrez facilement la distance parcourue ce jour-là.)

  Exemple avec les données de `KIRK`

  ```
  Au final sa meilleure performance est de 30 km réalisée le jour n°2
  ```

  Exemple avec les données de `PICARD`

  ```
  Jusqu'à maintenant sa meilleure performance est de 37 km réalisée le jour n°4
  ```

  Exemple avec les données de `SULU`

  ```
  Au final sa meilleure performance est de 28 km réalisée le jour n°4
  ```

Partie C – Comparaison des avancements

On souhaite ici comparer l'avancement de deux trekkeurs.

Complétez la procédure `depassement()` qui prend en paramètre deux listes et deux noms.

Elle affichera le numéro du jour où le premier trekker a dépassé le second ou le fait qu'il n'y est jamais arrivé. Inutile de poursuivre les comparaisons une fois qu'on a trouvé un dépassement.

Les calculs en grisé (bleu) dans l'exemple 1 ne devraient pas être effectués par votre procédure !

On comparera ce qui est comparable : le plus petit nombre de jours effectifs entre les deux parcours (listes). Les listes n'ont donc pas forcément la même longueur et un des deux trekkeurs (voire les deux) peut avoir abandonné. Cf. les indications ci-dessous dans les deux exemples.

EXEMPLE 1 DE SORTIES A PRODUIRE

```
Comparaison entre Picard et Kirk
Picard a dépassé pour la première fois Kirk au jour n°1
```

La comparaison se fait (au maximum) sur les 6 premières valeurs car une liste contient 6 valeurs (sans abandon) mais l'autre en contient 7 (sans abandon). Donc la dernière valeur de la seconde liste ne sera pas prise en compte car on ne comparera au maximum que les 6 premières valeurs des deux listes. Avec les données de ce cas précis, on s'arrête à la comparaison des valeurs en position 1 (deuxième jour) comme le montre le détail du calcul :

numéro de jour (indice)	distance PICARD (valeur)	distance KIRK (valeur)	distance totale PICARD (somme partielle)	distance totale KIRK (somme partielle)	comparaison
0	29	29	29	29	29 = 29
1	33	26	29+33= 62	29+26= 55	62 > 55
2	27	30	62+27= 89	55+30= 85	**Conclusion : Picard dépasse Kirk !**
3	29	29	89+29= 118	85+29= 114	
4	37	28	118+37= 155	114+28= 142	
5	30	30	155+30= 185	142+30= 172	
6		~~28~~			

dernière valeur pas prise en compte car comparaison possible sur 6 valeurs seulement (au maximum)

EXEMPLE 2 DE SORTIES A PRODUIRE

```
Comparaison entre Sulu et Kirk
Sulu n'a jamais dépassé Kirk
```

Dans ce cas, non seulement la plus courte des listes ne contient que 6 valeurs (alors que l'autre en contient 7) mais la 6ème valeur est le code d'abandon (0). Donc la comparaison entre les deux listes se fera au maximum sur les 5 premières valeurs des deux listes :

numéro de jour (indice)	distance SULU (valeur)	distance KIRK (valeur)	distance totale SULU (somme partielle)	distance totale KIRK (somme partielle)	comparaison
0	21	29	21	29	21 < 29
1	22	26	21+22= 43	29+26= 55	43 < 55
2	16	30	43+16= 59	55+30= 85	59 < 85
3	18	29	59+18= 77	85+29= 114	77 < 114
4	28	28	77+28= 105	114+28= 142	105 < 142
5	~~0~~	~~30~~			
		~~28~~			

Conclusion :
il n'y a aucun jour où "Sulu" > "Kirk"

deux dernières valeurs pas prises en compte car comparaison possible sur 5 valeurs seulement

Annexe aux parties A et B

Sorties à produire complètes pour la liste `KIRK = [29, 26, 30, 29, 28, 30, 28]`

```
Parcours de Kirk
Kirk a atteint l'objectif en 7 jours
Kirk a réalisé la moitié de la distance en 4 jours
Ce qui représente une moyenne globale de 28.6 km par jour
Voici l'évolution de sa moyenne au jour le jour (km/jour) :
au jour 0 : 29.0
au jour 1 : 27.5
au jour 2 : 28.3
au jour 3 : 28.5
au jour 4 : 28.4
au jour 5 : 28.7
au jour 6 : 28.6
Au final sa meilleure performance est de 30 km réalisée le jour n°2
```

Sorties à produire complètes pour la liste `PICARD = [29, 33, 27, 29, 37, 30]`

```
Parcours de Picard
Picard est toujours en course après 185 km
Il reste 15 km qui devraient être parcourus en 1 jour(s) s'il maintient sa
moyenne qui est de 30.8 km par jour
Voici l'évolution de sa moyenne au jour le jour (km/jour) :
au jour 0 : 29.0
au jour 1 : 31.0
au jour 2 : 29.7
au jour 3 : 29.5
au jour 4 : 31.0
au jour 5 : 30.8
Jusqu'à maintenant sa meilleure performance est de 37 km réalisée le jour n°4
```

Sorties à produire complètes pour la liste `SULU = [21, 22, 16, 18, 28, 0]`

```
Parcours de Sulu
Sulu a abandonné au bout de 105 km et 5 jours
Ce qui représente une moyenne globale de 21.0 km par jour
Voici l'évolution de sa moyenne au jour le jour (km/jour) :
au jour 0 : 21.0
au jour 1 : 21.5
au jour 2 : 19.7
au jour 3 : 19.2
au jour 4 : 21.0
Au final sa meilleure performance est de 28 km réalisée le jour n°4
```