TP Python 8 - Analyse et traitement d'une trace GPS

Les GPS sont de plus en plus utilisés pour faciliter nos déplacements. Ils peuvent être également utilisés afin d'enregistrer un parcours effectué à pied par exemple. Les données sont enregistrées dans un fichier .kml (Keyhole Markup Language). L'objectif de ce TP est d'extraire des données provenant d'un fichier kml puis de les analyser.

1) Ouvrir avec un éditeur de texte le fichier `OuCest.kml` situé dans le dossier de la classe.

Ce fichier est une succession de relevés GPS effectués au cours d'une randonnée. Il est constitué de quelques lignes d'introduction suivies d'une succession de couples de lignes de la forme :

```
<when>2014-01-05T09:40:53.445Z</when>
<gx:coord>6.110211 43.03791 65.49475860595703</gx:coord>
```

Chacun de ces couples de lignes correspond à un relevé GPS. La première donne la date et l'heure du relevé. La seconde donne la position (longitude et latitude en degrés, altitude en mètres) au moment du relevé.

Dans un premier temps on va extraire avec Python les informations contenues dans le fichier.

2) a) Ouvrir le fichier avec Python et créer la liste, qu'on appellera `lignes`, des lignes du fichier (utiliser `readlines`). Le début de cette liste doit être :

```
>>> lignes[:4]
['<?xml version="1.0" encoding="UTF-8"?>\n',
 '<kml xmlns="http://www.opengis.net/kml/2.2"\n',
 'xmlns:gx="http://www.google.com/kml/ext/2.2"\n',
 'xmlns:atom="http://www.w3.org/2005/Atom">\n']
```

b) Que signifient les `'\n'` à la fin de chaque ligne ?

3) Les lignes qui nous intéressent sont celles qui commencent par les balises `<when>` ou `<gx:coord>`.

a) Créer une liste `donnees` qui ne contient que ces lignes (on peut utiliser `in` pour tester si une chaîne de caractères est incluse dans une autre).

```
>>> donnees[:4]
['<when>2014-01-05T09:40:53.445Z</when>\n',
 '<gx:coord>6.110211 43.03791 65.49475860595703</gx:coord>\n',
 '<when>2014-01-05T09:41:01.440Z</when>\n',
 '<gx:coord>6.11024 43.037969 66.05236053466797</gx:coord>\n']
```

b) Combien de relevés ont été effectués ?

4) On souhaite maintenant mettre ces données sous une forme exploitable.

a) Écrire une fonction `heure` qui, recevant une ligne commençant par `<when>`, renvoie une liste de trois flottants correspondant au moment du relevé (heure, minute, seconde).

Pour cela on pourra utiliser la méthode `index` pour repérer la position du `T` et du `Z` dans la ligne :

```
>>> exemple = 'ceci est un exemple'
>>> exemple.index('i')
3
```

On pourra ensuite utiliser un slicing pour récupérer la partie de la ligne comprise entre le `T` et le `Z`, puis utiliser `split` pour la séparer en trois :

```
>>> exemple = "09:40:53.445"
>>> exemple.split(":")
['09', '40', '53.445']
```

et enfin `float` qui convertit une chaîne de caractères en nombre flottant.

```
>>> heure('<when>2014-01-05T09:40:53.445Z</when>\n')
[9.0, 40.0, 53.445]
```

b) De même, écrire une fonction `coord` qui, recevant une ligne commençant par `<gx:coord>`, renvoie une liste de trois flottants correspondant à la position du relevé (longitude, latitude, altitude).

```
>>> coord('<gx:coord>6.110211 43.03791 65.49475860595703</gx:coord>\n')
[6.110211, 43.03791, 65.49475860595703]
```

c) À l'aide des ces fonctions, créer une liste `releves` formée de listes de six flottants correspondant aux relevés du fichier. Chacune de ces listes doit être de la forme `[heure, minute, seconde, longitude, latitude, altitude]`. Le début de la liste sera donc :

```
>>> releves[:4]
[[9.0, 40.0, 53.445, 6.110211, 43.03791, 65.49475860595703],
 [9.0, 41.0, 1.44, 6.11024, 43.037969, 66.05236053466797],
 [9.0, 41.0, 2.749, 6.110254, 43.037998, 66.09146881103516],
 [9.0, 41.0, 6.589, 6.110277, 43.038083, 66.35663604736328]]
```

On peut maintenant exploiter les données extraites du fichier.

5) a) Écrire une fonction `secondes` qui, recevant une liste de longueur 3 représentant un temps (heures, minutes, secondes), renvoie le nombre de secondes correspondantes.

```
>>> secondes([9.0, 40.0, 53.445])
34853.445
```

b) Écrire une fonction `hms` réciproque de la fonction précédente.

```
>>> hms(34853.445)
[9.0, 40.0, 53.44499999999971]
```

c) Combien de temps a duré la randonnée ?

6) Quelles ont été les altitudes minimale et maximale atteintes lors de la randonnée ? À quelle heure ont-elles été atteintes ?

7) On souhaite maintenant obtenir le profil du tracé, c'est-à-dire représenter graphiquement l'altitude en fonction de la distance parcourue.

a) La distance entre deux points situés sur une sphère est appelée **orthodromie**. On note lo_A et la_A la longitude et la latitude du point A et de même lo_B et la_B la longitude et la latitude du point B. L'orthodromie se calcule alors ainsi en kilomètres :

$$2R \operatorname{Arcsin} \sqrt{\sin^2\left(\frac{la_B - la_A}{2}\right) + \cos(la_A)\cos(la_B)\sin^2\left(\frac{lo_B - lo_A}{2}\right)},$$

où $R = 6371$ km représente le rayon de la Terre. Attention : les angles sont en radians.

Écrire une fonction `orthodromie` qui, recevant deux listes contenant la longitude et la latitude de deux points de la Terre, renvoie la distance entre ces deux points. On pourra importer les fonctions nécessaires du module `math` :

```
from math import cos, sin, asin, sqrt, pi
```

Par exemple, la longitude et la latitude du lycée Thuillier sont respectivement 2,29717 et 49,87598 et celles de la gare d'Amiens sont respectivement 2,3082 et 49,89056. La distance entre le lycée et la gare est donc :

```
>>> orthodromie([2.29717, 49.87598], [2.3082, 49.89056])
1.8035801185132567
```

b) À l'aide de cette fonction, construire une liste `distance_parcourue` dont le k-ième élément est la distance parcourue depuis le début au moment du k-ième relevé. Le début de cette liste devrait être :

```
>>> distance_parcourue[:4]
[0, 0.006971023632196368, 0.010390527992269635, 0.02002516909464949]
```

c) Quelle était la longueur de la randonnée ? Quelle a été la vitesse moyenne sur l'ensemble du parcours ?

d) Construire une liste `temps_ecoule` dont le k-ième élément est le temps écoulé depuis le début au moment du k-ième relevé. Le début de cette liste devrait être :

```
>>> temps_ecoule[:4]
[0, 7.995000000002619, 9.304000000003725, 13.144000000000233]
```

e) La fonction `plot` du module `matplotlib` permet de tracer des courbes. Elle attend comme arguments la liste des abscisses et la liste des ordonnées des points à relier. On peut par exemple tracer la distance parcourue en fonction du temps de la manière suivante :

```
import numpy as np
import matplotlib.pyplot as plt
plt.plot(temps_ecoule, distance_parcourue)
plt.show()
```

En procédant de manière analogue, tracer le profil du parcours, c'est-à-dire l'altitude en fonction de la distance parcourue.

f) Pour voir le parcours lui-même, tracer la latitude en fonction de la longitude.

8) Où cette randonnée a-t-elle eu lieu ? On pourra reconnaître le tracé du parcours avec Google Maps par exemple.