Devoir d'informatique

Consignes

- Le devoir se fera sur copie double uniquement.
- Le numéro de chaque exercice et de chaque question devra être indiqué sur votre copie.
- Les indentations devront correctement figurer sur votre copie. Vous pourrez par exemple tracer une barre verticale.
- Pour chaque fonction vous donnerez au plus une ligne de commentaire permettant de spécifier votre fonction.

1 Randonnée

Lors d'une randonnée, un promeneur active son GPS pour retracer son parcours. Chaque point de mesure est caractérisé par quatre informations : la latitude (en degrés), la longitude (en degrés), l'altitude (en mètres) et le temps (en secondes). Ces informations sont stockées dans 4 tableaux :

```
lat = [45.461516, 45.461448, 45.461383, 45.461641, 45.461534, 45.461595, 45.461562, ...]
long = [6.44461, 6.444426, 6.444239, 6.444035, 6.443879, 6.4437, 6.443521, ...]
alt = [1315.221, 1315.702, 1316.182, 1316.663, 1317.144, 1317.634, 1318.105, ...]
tps = [1597496965, 1597496980, 1597496995, 1597496710, 1597496725, 1597496740, 1597496755,
...].
```

1.1 Analyse du profil global

Question 1
Donner l'instruction permettant de connaître la taille du tableau `lat`.

Le module `math` fournit la fonction `radians` qui convertit des degrés en radians.

Question 2 *Implémenter la fonction* `conversion(L:list)-> list` *permettant de convertir chacune des valeurs d'un tableau de flottants en radians. Cette fonction agira sans effet de bord.*

Question 3 *Donner les instructions permettant de créer les tableaux* `latr` *et* `longr`*, résultats de la conversion en radians des tableaux* `lat` *et* `long`*.*

Question 4 *Implémenter la fonction* `plus_haut(L:list) -> int` *permettant de déterminer l'altitude la plus haute atteinte lors de la randonnée (la fonction* `max` *sera ici interdite).*

Question 5 *Implémenter la fonction* `plus_haut_indice(L:list) -> float` *permettant de déterminer l'indice de l'altitude la plus haute atteinte lors de la randonnée.*

Question 6 *En utilisant la fonction précédente, implémenter la fonction* `coords_plus_haut(alt:list, long:list, lat:list)-> list` *permettant de renvoyer la liste* `[latitude, longitude]` *des coordonnées du point le plus haut de la randonnée.*

Question 7 *Implémenter la fonction* `deniveles(alt:list) -> list` *qui calcule les dénivelés cumulés positif et négatif (en mètres) de la randonnée, sous forme d'une liste de deux flottants. Le dénivelé positif est la somme des variations d'altitude positives sur le chemin, et inversement pour le dénivelé négatif.*

Nous avons à notre disposition la fonction `distance(list:c1, list:c2) -> float` permettant de déterminer la distance entre deux points successifs en tenant compte de la latitude, de longitude, de l'altitude d'un point 1 de coordonnées `c1` et d'un point 2 de coordonnées `c2`, du rayon de la Terre *etc.* `c1` est constitué de l'altitude, la latitude et la longitude d'un seul point.

Question 8 *Implémenter la fonction* `distance_totale(alt:list, long:list, lat:list) -> float` *renvoyant la distance parcourue (en mètres) au cours de la randonnée.*

1.2 Découpage du profil

Dans cette partie, nous allons chercher à découper le profil de la randonnée en tentant de retrouver les différentes montagnes franchies par le randonneur.

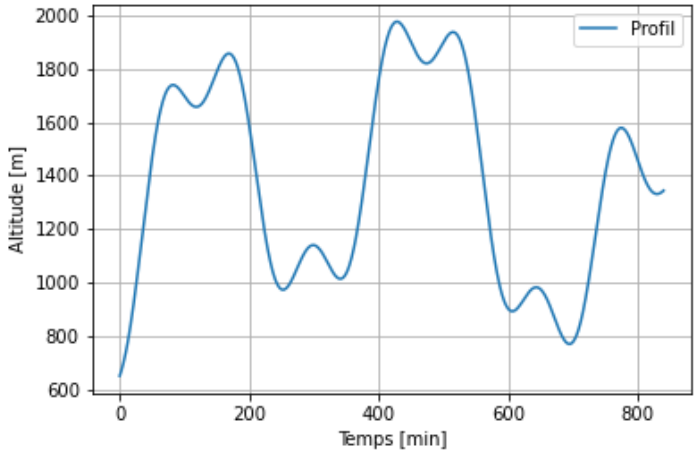

FIGURE 1 – Profil de la randonnée

Question 9 *Implémenter la fonction* `moyenne(alt:list) -> float` *permettant de calculer la moyenne des altitudes mesurées par le GPS.*

On note PND les points de passages par le niveau moyen en descente et PNM les points de passage par le niveau moyen en montée.

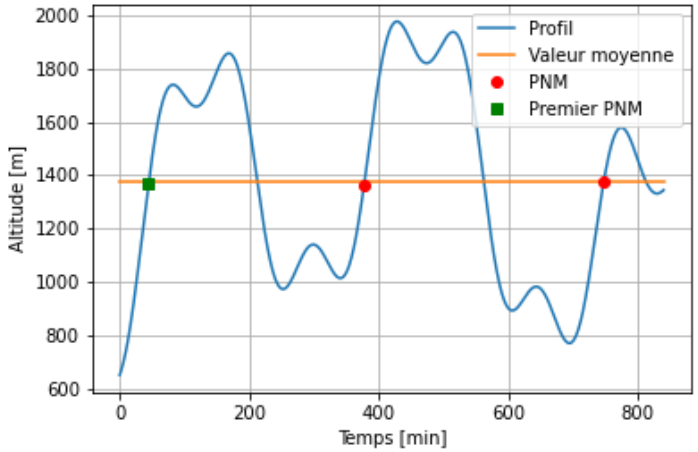

FIGURE 2 – Points de passage par le niveau moyen en montée [PNM]

Question 10 *Implémenter la fonction* `indice_premier_PNM(alt:list) -> int` *renvoyant, s'il existe, l'indice* `i` *du premier élément de la liste tel que cet élément soit inférieur à la moyenne et l'élément suivant soit supérieur*

à la moyenne. Cette fonction devra renvoyer `-1` *si aucun élément vérifiant cette condition n'existe.*

Question 11 *Implémenter la fonction* `indices_PNM(alt:list) -> list` *retournant la liste des indices de tous les PNM.*

Question 12 *Dans le but de séparer les différents profils, nous allons chercher les indices des altitudes minimales entre deux PNM successifs. Implémenter la fonction* `liste_alt_mini(alt:list) -> list` *qui répond à ce besoin.*

On appelle `pam` la liste des indices des points ayant une altitude minimale.

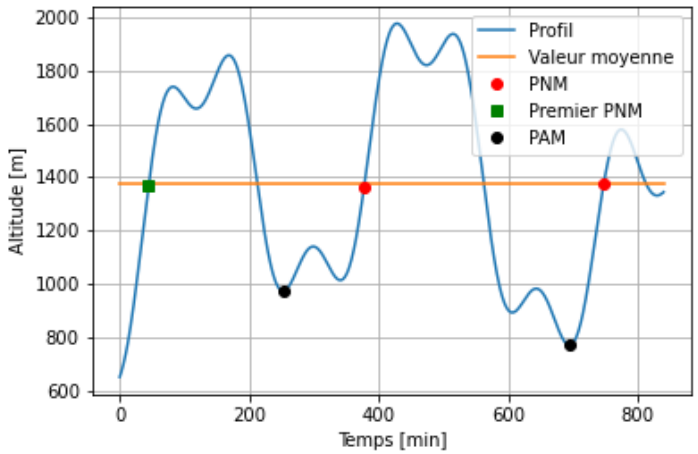

FIGURE 3 – Points avec altitude minimale [pam]

On souhaite maintenant décomposer le profil mesuré en plusieurs « montagnes ». Une montagne est une liste constituée d'altitudes successives. La première montagne ira de la première altitude mesurée au premier `pam`. On aura ensuite une montagne entre chaque `pam`. La dernière montagne ira du dernier `pam` à la dernière altitude mesurée.

Question 13 *Implémenter la fonction* `creer_montagnes(alt) -> list` *renvoyant une liste constituée de la liste des montagnes élémentaires.*

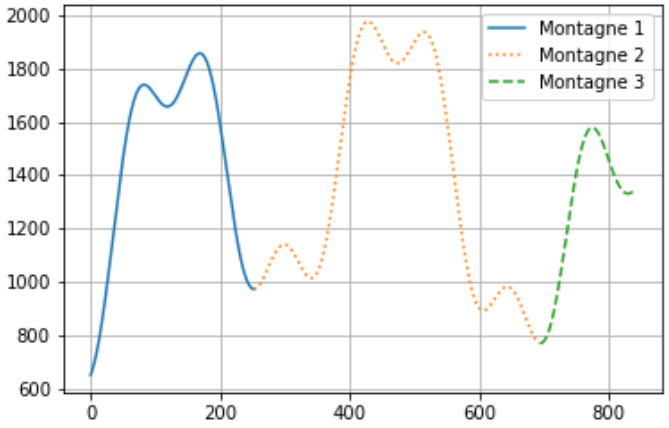

FIGURE 4 – Découpage en montagnes