TP 4
Analyse d'un profil de randonnée – Sujet

On souhaite réaliser différentes opérations sur un parcours de randonnée effectué lors d'un weekend à la montagne.

Pour s'entraîner nous allons nous entrainer sur un profil fictif. On téléchargera le fichier `DecoupageMontagnes_eleve.py` sur le site de la classe. On enregistrera ce fichier dans vos documents personnel (lecteur `U:`, répertoire `Informatique/TP_04`). On ouvrira ce fichier avec `Pyzo`. Pour exécuter le programme on utilisera la combinison de touche `Ctrl + E`.

Questions préliminaires

La variable `les_x` contient l'abscisse d'un profil. La variable `les_y` contient l'altitude d'un profil. Saisir les instruction suivantes.

```
plt.ylabel("Altitude [m]")
plt.plot(les_x,les_y,label = "Profil")
plt.legend()
plt.grid()
plt.show()
```

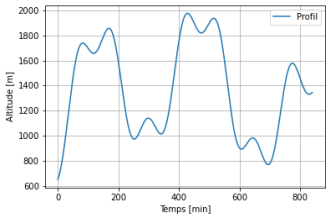

Figure 1 – Profil de la randonnée

Question 1 Vérifier qu'un profil de montagne s'affiche.

Question 2 Écrire la fonction `maximum(L:list) -> int` permettant de déterminer le maximum d'une liste (la fonction `max` sera ici interdite).

Question 3 Vérifier que `maximum(les_y[0:300])` renvoie `1855.99`.

Question 4 Écrire 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 5 Vérifier que `plus_haut_indice(les_y[300:500])` renvoie `199`.

Question 6 Écrire 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.

Question 7 Vérifier que `deniveles(les_y)` renvoie `[3438.100, -2746.747]`.

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.

Question 8 Écrire la fonction `moyenne(alt:list) -> float` permettant de calculer la moyenne des altitudes mesurées par le GPS (l'utilisation de `sum` est interdite).

Question 9 Vérifier que `moyenne(les_y)` renvoie `1376.35`.

Question 10 On note `n` la taille du tableau `les_y`. Créer une liste `les_moy` contenant `n` fois la valeur `moyenne(les_y)`. Tracer sur la même courbe le profil de la randonnée et la valeur moyenne sur tout le profil.

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.

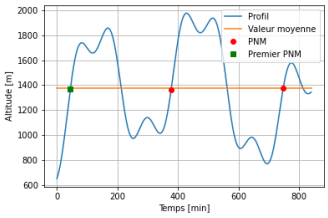

Figure 2 – Points de passage par le niveau moyen en montée [PNM]

Question 11 Écrire 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 12 Vérifier que `indice_premier_PNM(les_y)` renvoie `53` et `indice_premier_PNM(les_y[200:250])` renvoie `-1`.

Question 13 Écrire la fonction `indices_PNM(alt:list) -> list` retournant la liste des indices de tous les PNM.

Question 14 Vérifier que `indices_PNM(les_y)` renvoie `[53, 449, 889]`.

Question 15 Dans le but de séparer les différents profils, nous allons chercher les indices des altitudes minimales entre deux PNM successifs. Écrire la fonction `liste_alt_mini(alt:list) -> list` qui répond à ce besoin. En utilisant le profil donné, cette fonction renverrait la liste `[300,826]`.

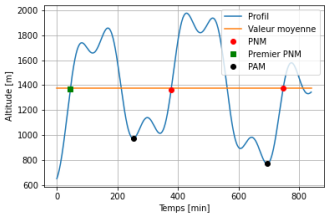

Figure 3 – Points avec altitude minimale [pam]

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

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 16 Écrire la fonction `creer_montagnes(alt) -> list` renvoyant une liste constituée de la liste des montagnes élémentaires.

Question 17 Tracer chacun des montagnes afin d'obtenir le graphique suivant.

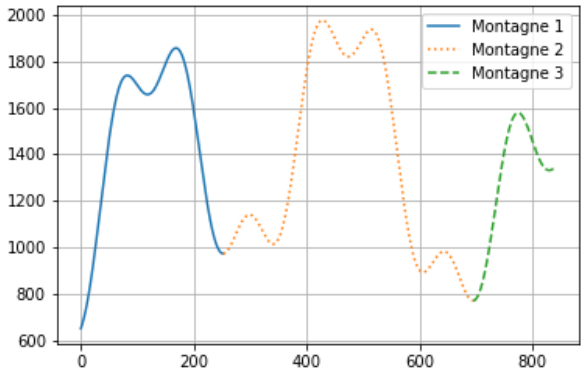

Figure 4 – Découpage en montagnes

TP 4
Analyse d'un profil de randonnée – Corrigé

Questions préliminaires

La variable `les_x` contient l'abscisse d'un profil. La variable `les_y` contient l'altitude d'un profil. Saisir les instruction suivantes.

```
plt.ylabel("Altitude [m]")
plt.plot(les_x,les_y,label = "Profil")
plt.legend()
plt.grid()
plt.show()
```

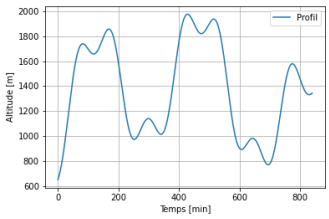

FIGURE 5 – Profil de la randonnée

Question 1 Vérifier qu'un profil de montagne s'affiche.

Question 2 Écrire la fonction `maximum(L:list) -> int` permettant de déterminer le maximum d'une liste (la fonction `max` sera ici interdite).

Correction

```
def plus_haut(L:list) -> float :
    maxi = L[0]
    for el in L :
        if el>maxi :
            maxi = el
    return maxi
```

Question 3 Vérifier que `maximum(les_y[0:300])` renvoie `1855.99`.

Question 4 Écrire la fonction `plus_haut_indice(L:list) -> float` permettant de déterminer l'indice de l'altitude la plus haute atteinte lors de la randonnée.

Correction

```
def plus_haut_indice(L:list) -> float :
    m = 0
    for i in range(len(L)):
        if L[i]>L[m] :
            m = i
    return m
```

Question 5 Vérifier que `plus_haut_indice(les_y[300:500])` renvoie `199`.

Question 6 Écrire 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.

Correction

```
def deniveles(alt:list) -> list:
    pos,neg = 0,0
```

```
    for i in range(1,len(alt)) :
        delta = alt[i]-alt[i-1]
        if  delta > 0:
            pos = pos + delta
        else :
            neg = neg + delta
    return [pos,neg]
```

Question 7 Vérifier que `deniveles(les_y)` renvoie `[3438.100, -2746.747]`.

Découpage du profil

Question 8 Écrire la fonction `moyenne(alt:list) -> float` permettant de calculer la moyenne des altitudes mesurées par le GPS (l'utilisation de `sum` est interdite).

Correction

```
def moyenne(alt:list):
    somme = 0
    for a in alt :
        somme = somme + a
    return somme/len(alt)
```

Question 9 Vérifier que `moyenne(les_y)` renvoie `1376.35`.

Question 10 On note `n` la taille du tableau `les_y`. Créer une liste `les_moy` contenant `n` fois la valeur `moyenne(les_y)`. Tracer sur la même courbe le profil de la randonnée et la valeur moyenne sur tout le profil.

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.

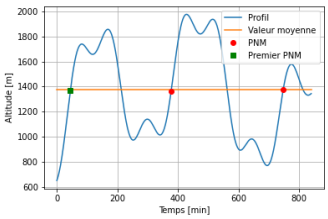

FIGURE 6 – Points de passage par le niveau moyen en montée [PNM]

Question 11 Écrire 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.

Correction

```
def indice_premier_PNM(alt:list):
    m = moyenne(alt)
    indice = -1
    for i in range(len(alt)-1):
        if alt[i]<m and alt[i+1]>m:
            return i
    return indice
```

Question 12 Vérifier que `indice_premier_PNM(les_y)` renvoie `53` et `indice_premier_PNM(les_y[200:250])` renvoie `-1`.

Question 13 Écrire la fonction `indices_PNM(alt:list) -> list` retournant la liste des indices de tous les PNM.

Correction

```
def indices_PNM(alt:list):
    m = moyenne(alt)
    les_PNM = []
    for i in range(len(alt)-1):
        if alt[i]<m and alt[i+1]>m:
            les_PNM.append(i)
    return les_PNM
```

Question 14 Vérifier que `indices_PNM(les_y)` renvoie `[53, 449, 889]`.

Question 15 Dans le but de séparer les différents profils, nous allons chercher les indices des altitudes minimales entre deux PNM successifs. Écrire la fonction `liste_alt_mini(alt:list) -> list` qui répond à ce besoin. En utilisant le profil donné, cette fonction renverrait la liste `[300,826]`.

Correction

```
def liste_alt_min(alt:list):
    les_PNM = indices_PNM(alt)
    les_min = []
    for i in range(len(les_PNM)-1):
        mini = les_PNM[i]
        for j in range(les_PNM[i],les_PNM[i+1]):
            if alt[j]<alt[mini]:
                mini = j
        les_min.append(mini)
    return les_min
```

Question 16 Écrire la fonction `creer_montagnes(alt) -> list` renvoyant une liste constituée de la liste des montagnes élémentaires.

Correction

```
def creer_montagnes(alt):
    pam = liste_alt_min(alt)
    montagnes = []
    mont = []
    for i in range(0,pam[0]):
        mont.append(alt[i])
    montagnes.append(mont)
    for i in range(len(pam)-1):
        mont = []
        for j in range(pam[i],pam[i+1]):
            mont.append(alt[j])
        montagnes.append(mont)
    mont = []
    for i in range(pam[-1],len(alt)):
        mont.append(alt[i])
    montagnes.append(mont)
    return montagnes
```

Question 17 Tracer chacun des montagnes afin d'obtenir le graphique suivant.