# Marchons, marchons, marchons...

Ce sujet propose d'appliquer des techniques informatiques à l'étude de trois marches de natures différentes :

— Une marche concrète (partie I - Randonnée)
— Une marche stochastique (partie II - Mouvement brownien d'une petite particule)
— Une marche auto-évitante (partie III - Marches auto-évitantes)

Les trois parties sont indépendantes. L'annexe à la fin du sujet fournit les canevas de code que vous devrez reprendre dans votre copie pour répondre aux questions de programmation Python.

## Partie I. Randonnée

Lors de la préparation d'une randonnée, une accompagnatrice doit prendre en compte les exigences des participants. Elle dispose d'informations rassemblées dans deux tables d'une base de données :

— La table `Rando` décrit les randonnées possibles – la clef primaire entière `rid`, son nom, le niveau de difficulté du parcours (entier entre 1 et 5), le dénivelé en mètres, la durée moyenne en minutes :

| **rid** | **rnom** | **diff** | **deniv** | **duree** |
|---|---|---|---|---|
| 1 | La belle des champs | 1 | 20 | 30 |
| 2 | Lac de Castellagne | 4 | 650 | 150 |
| 3 | Le tour du mont | 2 | 200 | 120 |
| 4 | Les crêtes de la mort | 5 | 1200 | 360 |
| 5 | Yukon Ho ! | 3 | 700 | 210 |
| ... | ... | ... | ... | ... |

— La table `Participant` décrit les randonneurs – la clef primaire entière `pid`, le nom du randonneur, son année de naissance, le niveau de difficulté maximum de ses randonnées :

| **pid** | **pnom** | **ne** | **diff_max** |
|---|---|---|---|
| 1 | Calvin | 2014 | 2 |
| 2 | Hobbes | 2015 | 2 |
| 3 | Susie | 2014 | 2 |
| 4 | Rosalyn | 2001 | 4 |
| ... | ... | ... | ... |

Donner une requête SQL sur cette base pour :

❏ **Q1** – Compter le nombre de participants nés entre 1999 et 2003 inclus.

❏ **Q2** – Calculer la durée moyenne des randonnées pour chaque niveau de difficulté.

❏ **Q3** – Extraire le nom des participants pour lesquels la randonnée n°42 est trop difficile.

❏ **Q4** – Extraire les clés primaires des randonnées qui ont un ou des homonymes (nom identique et clé primaire distincte), sans redondance.

L'accompagnatrice a activé le suivi d'une randonnée par géolocalisation satellitaire et souhaite obtenir quelques propriétés de cette randonnée une fois celle-ci effectuée. Elle a exporté les données au format texte CSV (*comma-separated values* – valeurs séparées par des virgules) dans un fichier nommé `suivi_rando.csv` : la première ligne annonce le format, les suivantes donnent les positions dans l'ordre chronologique.

Voici le début de ce fichier pour une randonnée partant de Valmorel, en Savoie, un bel après-midi d'été :

```
lat(°),long(°),height(m),time(s)
45.461516,6.44461,1315.221,1597496966
45.461448,6.444426,1315.702,1597496970
45.461383,6.444239,1316.182,1597496973
45.461641,6.444035,1316.663,1597496979
45.461534,6.443879,1317.144,1597496984
45.461595,6.4437,1317.634,1597496989
45.461562,6.443521,1318.105,1597496994
...
```

Le module `math` de Python fournit les fonctions `asin`, `sin`, `cos`, `sqrt` et `radians`. Cette dernière convertit des degrés en radians, unité des fonctions trigonométriques. La documentation donne aussi des éléments de manipulation de fichiers textuels :

`fichier = open(NOM_FICHIER, MODE)` ouvre le fichier, en lecture si mode est `"r"`, en écriture si `"w"`.

`ligne = fichier.readline()` récupère la ligne suivante de `fichier` ouvert en lecture avec `open`.

`lignes = fichier.readlines()` donne la liste des lignes suivantes.

`fichier.close()` ferme `fichier`, ouvert avec `open`, après son utilisation.

`ligne.split(SEP)` découpe la chaîne de caractères `ligne` selon le séparateur `SEP` : si `ligne` vaut `"42,43,44"`, alors `ligne.split(",")` renvoie la liste `["42", "43", "44"]`.

On souhaite exploiter le fichier de suivi d'une randonnée – supposé préalablement placé dans le répertoire de travail – pour obtenir une liste `coords` des listes de 4 flottants (latitude, longitude, altitude, temps) représentant les points de passage collectés lors de la randonnée.

À partir du canevas fourni en annexe, et en ajoutant les `import` nécessaires :

❏ **Q5** – Implémenter la fonction `importe_rando` qui réalise cette importation en retournant la liste souhaitée, par exemple en utilisant certaines des fonctions ci-dessus, ou une autre approche de votre choix.

On suppose maintenant l'importation effectuée dans `coords`, avec au moins deux points d'instants distincts.

❏ **Q6** – Implémenter la fonction `plus_haut` qui renvoie la liste (latitude, longitude) du point le plus haut de la randonnée.

❏ **Q7** – Implémenter la fonction `deniveles` qui calcule les denivelés 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.

On souhaite évaluer de manière approchée la distance parcourue lors d'une randonnée. On suppose la Terre parfaitement sphérique de rayon $R_T = 6371$ km au niveau de la mer. On utilise la formule de haversine pour calculer la distance $d$ du grand cercle sur une sphère de rayon $r$ entre deux points de coordonnées (latitude, longitude) $(\varphi_1, \lambda_1)$ et $(\varphi_2, \lambda_2)$ :

$$d = 2\,r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

On prendra en compte l'altitude moyenne de l'arc, que l'on complètera pour la variation d'altitude par la formule de Pythagore, en assimilant la portion de cercle à un segment droit perpendiculaire à la verticale.

❏ **Q8** – Implémenter la fonction `distance` qui calcule avec cette approche la distance en mètres entre deux points de passage. On décomposera obligatoirement les formules pour en améliorer la lisibilité.

❏ **Q9** – Implémenter la fonction `distance_totale` qui calcule la distance en mètres parcourue au cours d'une randonnée.

Annexe : canevas de codes Python

— Partie I : Randonnée

```python
# import Python à compléter...

# importation du fichier d'une randonnée
def importe_rando(nom_fichier):
    # À compléter...

coords = importe_rando("suivi_rando.csv")

# donne le point (latitude, longitude) le plus haut de la randonnée
def plus_haut(coords):
    # À compléter...

print("point le plus haut", plus_haut(coords))
# exemple : point le plus haut [45.461451, 6.443064]

# calcul des dénivelés positif et négatif de la randonnée
def deniveles(coords):
    # À compléter...

print("denivelés", deniveles(coords), "m")
# exemple : denivelés [4.059999999999945, -1.1759999999999309] m

RT = 6371 # rayon moyen volumétrique de la Terre en km

# distance entre deux points
def distance(coord1, coord2):
    # À compléter...

print("premier intervalle", distance(coords[0], coords[1]), "m")
# exemple : premier intervalle 16.230964254992816 m

# distance totale de la randonnée
def distance_totale(coords):
    # À compléter...

print("distance parcourue", distance_totale(coords), "m")
# exemple : distance parcourue 187.9700904658368 m
```