Concours blanc

Durée : 2H. Calculatrices interdites. Un document annexe listant quelques commandes python classiques est fourni.

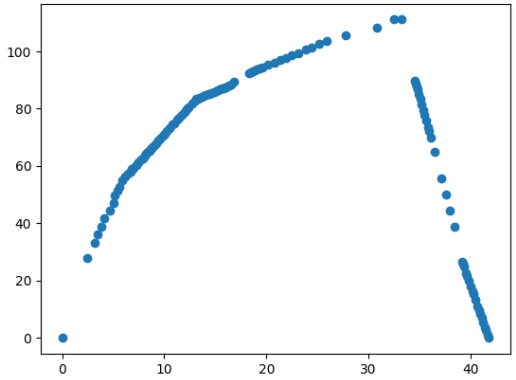
Il est fortement conseillé d'utiliser des noms de variables ne laissant aucune ambiguïté et de commenter le code dès que vous le jugez nécessaire : un algorithme correct, même avec des erreurs de code, peut rapporter des points.

I Récupération des données

Nous disposons d'un fichier csv nommé **vitesses.csv** qui décrit l'accélération puis la décélération d'un véhicule [1]. Chaque ligne du fichier est au format

temps, vitesse

où *temps* et *vitesse* (en m/s) sont des flottants et sont séparés par une virgule (sans espace entre les deux).
On souhaite dans cette première partie obtenir une représentation graphique comme suit :



1. Les points de mesures expérimentaux sont-ils régulièrement espacés dans le temps ?
2. En consultant l'annexe éventuellement, écrire des instructions python qui stockent dans une variable `lignes` la liste des lignes du fichier.
3. En considérant que `lignes` contient la liste des chaînes de caractères qui sont les lignes du fichier **vitesses.csv**, écrire les instructions python permettant de créer les listes `LT` et `LV` contenant respectivement les temps et vitesses des mesures expérimentales.
4. Tracer la courbe de la figure précédente, en utilisant une marque ronde et la couleur bleue (voir l'annexe).

II Analyse statistique des données

Cette partie est indépendante de la première partie, et vos fonctions ne devront pas faire d'hypothèses en plus de celles de l'énoncé concernant les données.

1. Écrire une fonction `indice_maxi` qui prend en argument une liste de nombres `L` et retourne l'indice de la valeur maximale contenue dans `L`. Si ce maximum est atteint plusieurs fois, on retournera l'indice de la première occurrence.
2. Écrire une fonction `moyenne` qui prend en argument une liste de nombres `L` et retourne la moyenne arithmétique de ces nombres.
3. Ecrire une fonction `plus_rapide` prenant une liste `V` contenant des nombres (des vitesses, dans notre exemple), et retournant une liste de booléens de même longueur, chacun indiquant si la vitesse de même indice est strictement supérieure à la moyenne.
4. Écrire une fonction `rapide` d'arguments `T` et `V` deux listes. `T` contient des temps, rangés par ordre strictement croissants et `V` des vitesses mesurées au temps contenus dans `T`. Ainsi ces deux listes ont la même longueur et `V[i]` est la vitesse au temps `T[i]`.

   La fonction `rapide` retourne le temps cumulé pendant lequel le mobile étudié se déplace strictement plus vite que sa vitesse moyenne. On ne cherchera pas à estimer les temps où le mobile dépasse la moyenne (dans un sens ou dans l'autre), autrement dit on ne considère que des temps stockés dans `T`.

   Par exemple, si la vitesse moyenne n'est dépassée qu'une fois, le temps cumulé est nul.

III Cinématique

Dans cette partie, on va étudier plus précisément le mouvement de notre véhicule. En particulier, on s'intéresse aux positions au cours du temps, ainsi qu'aux accélérations. Les listes considérées ici sont supposées construites correctement dans la première partie.

1. On note v_i l'élément d'indice i `LV` et t_i l'élément d'indice i de `LT`. On considère que notre véhicule se déplace le long d'un axe, à la position 0 au temps 0. On note x_i

1. Notre exemple graphique est tiré d'une analyse vidéo d'une Bugatti Chiron, pour les connaisseurs.

la position du véhicule au temps t_i. Exprimer x_{i+1} en fonction de x_i, v_i, t_i et t_{i+1}. Expliquer rapidement l'approximation utilisée.

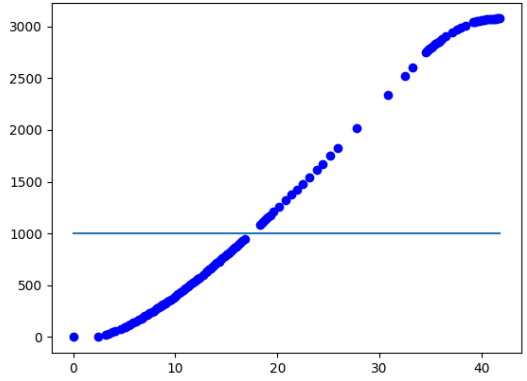
2. Écrire une fonction `positions` prenant comme argument `T` une liste de temps et `V` une liste de vitesses (avec la même interprétation que dans la partie précédente), et retournant la liste des positions estimées (la position vaut 0 au temps 0).

3. Écrire une fonction `mille_metres` prenant comme argument `T` une liste de temps et `X` la liste des positions correspondantes (telle que construite par la fonction de la question précédente) et retournant l'indice p tel que $x_p \leq 1000 < x_{p+1}$. On suppose qu'un tel indice existe toujours.

4. On approxime le mouvement entre les instants t_p et t_{p+1} par un mouvement à la vitesse constante v_p (l'indice p est défini à la question précédente).

   Exprimer le temps t_{mil} où le véhicule a parcouru exactement 1000 mètres en fonction de x_p, t_p et v_p. Un schéma sera apprécié.

5. On considère que les positions ont été calculées et stockées dans la variable `LX`. Donner les instruction permettant d'afficher le graphique des positions ainsi que la droite d'équation $x = 1000$ représentant le parcours du premier kilomètre, sous la forme suivante :



6. Écrire une fonction `accel` d'arguments deux listes `T` et `V` (avec les mêmes signification que précédemment) et retournant la liste des accélérations a_i au temps t_i pour tous les temps sauf le dernier.

   On précisera également l'expression de a_i en fonction de $v_i, v_{i+1}, t_i, t_{i+1}$.

IV Base de données

Les données que nous avons étudiées sont stockées dans une base de données, et pour plusieurs véhicules, suivant le schéma :

vehicules	**tests**
id	id
modele (nombre)	vehicule_id
constructeur (texte)	vit_max (nombre)
annee (nombre)	accel_max (nombre)
...	mille_metre (nombre)
...	date (texte, au format aaaa-mm-jj)

1. Donner la requête SQL permettant d'obtenir tous les tests postérieurs au 1er janvier 2016 (inclus) (on souhaite obtenir la date et l'id_vehicule).
2. Donner la requête permettant de connaître le nombre de tests dans la base.
3. Donner (en écrivant une requête) les modèles des véhicules ayant une accélération maximale strictement supérieure à 5 (exprimée en m/s^2, dans la table).
4. Donner le nombre de véhicule par constructeur pouvant atteindre ou dépasser la vitesse de $70m/s$.
5. Donner le modele et le constructeur de la voiture ayant le plus faible temps pour parcourir 1km (valeur stockée dans la colonnes **mille_metre**).

V Traitement des données stockées

On considère ici que les données présentes dans la base ont été extraite en python et stockées dans des listes.

1. On considère le code suivant :

```
1  def f(L):
2      for i in range(len(L) - 1):
3          a = L[i]
4          b = i
5          for j in range(i + 1, len(L)):
6              if L[j] < a:
7                  a = L[j]
8                  b = j
9          aux = L[i]
10         L[i] = L[b]
11         L[b] = aux
```

   (a) Expliquer l'effet des lignes 9 à 11.

(b) On se place dans le cas $i = 0$ pour cette question (premier passage dans la boucle externe). Expliquer les valeurs des variables `a` et `b` juste avant l'exécution de la ligne 9 (c'est à dire après la fin de la boucle interne).

(c) Justifier, par récurrence, que après l'exécution de la boucle sur i pour la valeur $i = i_0$, les éléments d'indice $0, \ldots, i_0 + 1$ de `L` sont triés dans l'ordre croissant et sont les $i_0 + 1$ plus petits éléments de `L`.

Ainsi, après l'exécution de `f(L)` la liste `L` a été modifiée pour être triée dans l'ordre croissant.

(d) On note N la longueur de L. Donner en justifiant le nombre de fois où la ligne 6 est exécutée.

2. On suppose que les données présentes dans la table **tests** de la partie précédente ont été stockées en python dans une variable `L_tests`. Chaque élément de `L_tests` est lui même une liste de longueur 6 (les données étant dans le même ordre que le tableau de la partie précédente)

Donner la suite d'instruction python permettant de modifier `L_tests` pour que ses 10 premiers éléments soient les tests correspondant aux véhicules ayant les temps les plus faibles pour parcourir 1km, et dans l'ordre croissant.

3. En notant N la longueur de `L_tests`, combien d'opérations de comparaisons doivent être effectuées pour répondre à la question précédente ?