université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

vendredi 6 décembre 2024

Introduction à l'informatique
L1

TP nº 10-2

Consignes les exercices ou questions marqués d'un ⋆ devront être rédigés sur papier (afin de se préparer aux épreuves écrites du partiel et de l'examen). En particulier, il est recommandé d'être dans les même conditions qu'en examen : pas de document ni de calculatrice. Les questions marquées d'un ⋄ sont des questions supplémentaires permettant d'aller plus loin (et ne seront pas forcément corrigées en TP). Tous les TPs se font sous Linux.

But du TP

Le but de cette feuille d'exercice est :
— manipuler les compréhension de tableaux
— charger des fichiers CSV
— dessiner des graphes simples avec matplotlib

Exercice 0

Ouvrir un terminal :
— créer un répertoire `IntroInfo` et un répertoire `TP10` à l'intérieur. Note : les fichiers des TPs précédents ne sont pas accessibles directement depuis cet environnement.
— se placer à l'intérieur du répertoire `TP10`
— télécharger les fichiers `tweets_s.csv` et `tweets.py` sur la page du cours, puis les déposer dans votre répertoire `TP10` (en utilisant le bouton Upload).

On suppose pour les autres exercices que le répertoire `TP10` est le répertoire courant.

Exercice 1

Le but de cet exercice est d'analyser un fichier CSV contenant des messages du réseau social Twitter [1] [2]. Le fichier contient plusieurs colonnes :

`id` : l'identifiant unique du tweet

`user_id` : l'identifiant unique de l'auteur

`user_name` : le nom d'utilisateur de l'auteur

`user_screen_name` : le nom affiché pour l'utilisateur

`user_followers_count` : le nombre de « *followers* » de l'utilisateur (les *followers* sont les gens abonnés au flux de tweets de l'utilisateur).

`text` : le texte du tweet

`place_name` : le nom du lieu où a été posté le tweet

`place_country` : le nom du pays où a été posté le tweet

`place_country_code` : le code sur deux lettre du pays où a été posté le tweet

`longitude` : la longitude du lieu du tweet. Entre -180 et +180 degrés.

`latitude` : la latitude du lieu. Entre -90 et +90 degrés.

`lang` : la langue du tweet (sur 2 lettres)

`date` : la date du tweet au format ISO `AAAA-MM-JJThh:mm:ss+00:00`

`timestamp` : la date du tweet en millisecondes depuis le 1er janvier 1970

1. Ces messages sont des tweets publics qui ont été archivés sur Web Archive
2. Twitter s'appelle maintenant X

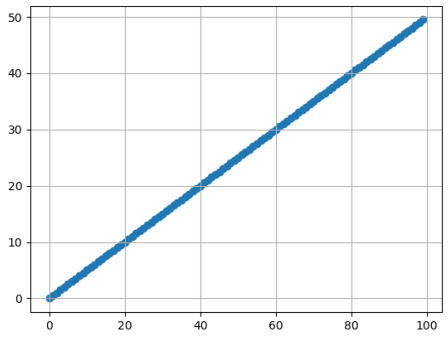
`hashtag_0` : le premier hash tag du tweet ou `""` si aucun hashtag

`hashtag_1` : le deuxième hash tag du tweet ou `""` si aucun hashtag

`hashtag_2` : le troisième hash tag du tweet ou `""` si aucun hashtag

Dans un premier temps, on va mettre en place des fonctions permettant de créer des graphiques. On édite pour cela le fichier `tweets.py`.

1. Écrire une fonction **`dessine_points(lx, ly, img)`** qui :
   — efface le dessin courant (**`pyplot.clf()`**)
   — dessine les points dont les abscisses sont dans le tableau `lx` et les ordonnées dans le tableau `ly` (**`pyplot.scatter(...)`**)
   — dessine une grille (**`pyplot.grid()`**)
   — sauvegarde le dessin dans le fichier dont le nom est donné par la variable `img`
   Tester votre fonction sur la liste de point $\{(x, 0.5 \times x) \mid x \in \mathbb{N}, 0 \leq x \leq 100\}$. Vous devriez obtenir ce dessin :



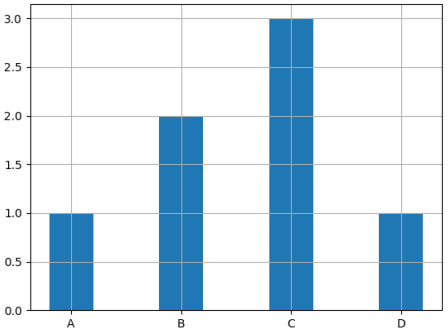
2. Écrire une fonction **`dessine_barres(t, img)`**. Cette fonction prend en argument un tableau de couples `t` et un nom de fichier d'image `img`. Les couples du tableau `t` sont composés d'une chaîne de caractères et d'un nombre. La fonction doit :
   — créer un tableau `lx` des entiers $0, 2, 4, 6, \ldots, 2 \times$ `len(t) - 1`
   — créer un tableau `ly` des secondes composantes de `t`
   — créer un tableau `lt` des premières composantes de `t`
   Une fois ces trois tableaux créés, dessiner le graphique en barres avec

```
1 pyplot.clf()
2 pyplot.bar(lx, ly, tick_label=lt)
3 pyplot.grid()
```

   puis sauver le dessin dans le fichier dont le nom est donné par `img`.
   Tester votre fonction sur le tableau :

```
1 [ ("A", 1), ("B", 2), ("C", 3), ("D", 1) ]
```

   Vous devriez obtenir ce dessin :



3. Écrire du code (en dehors d'une fonction) chargeant le fichier `tweets_s.csv`. Le code doit :
   — ouvrir le fichier en lecture

— charger le fichier comme un tableau de dictionnaires (`csv.DictReader(...)`)
— stocker ce tableau dans une variable globale `TWEETS`
— modifier chaque dictionnaire du tableau pour que les valeurs associée aux clés `user_followers_count` et `timestamp` soient converties en entier et celles des clés `latitude` et `longitude` soient converties en flottants.

Dans la suite, on appalle *un tableau de tweets* un tableau de dictionnaires similaire à celui contenu dans le tableau `TWEETS`.

4. Écrire une fonction `nb_tweets(t)` qui renvoie le nombre de tweets dans le tableau de tweets `t`. Tester cette fonction et afficher le nombre de tweets contenus dans `TWEETS`.
5. Écrire une fonction `long_tweet_temps(t)` qui calcule dans un tableau `lx` le jour de chaque tweet à partir du début de l'année (tous les tweets datent de 2019). On pourra procéder de la façon suivante. Pour chaque dictionnaire `d` dans `t`
   — récupérer la valeur de `d["timestamp"]`
   — lui soustraire 1546300800000 (le nombre de secondes entre le 1er janvier 1970 et le 1er janvier 2019)
   — diviser ensuite le nombre par $24 \times 3600 \times 1000$ pour obtenir le nombre de jours.

   La fonction calcule ensuite dans un tableau `ly` de toutes les longueurs des textes des tweets (clé `"text"`). Enfin la fonction dessine les points contenus dans les tableaux `lx` et `ly`.
6. `nb_tweets_langue(t)` qui dessine un graphique en barres indiquant pour chaque langue de message, le nombre de tweets, uniquement pour les langues ayant plus de 100 tweets. On procèdera de la façon suivante :
   — créer un dictionnaire vide `res`
   — pour chaque dictionnaire `d` dans `t` (en utilisant une boucle `for`)
      — récupérer la valeur de la clé `d["lang"]` (le code de pays du tweet)
      — si ce code est absente du dictionnaire `res`, lui associer la valeur 1
      — si ce code est présent, incrémenter la valeur associée

      En d'autre termes, `res` est un dictionnaire qui associe à chaque code de langue le nombre de `tweets`.
   — Créer ensuite un tableau `res100` contenant les paires `(code, num)` des codes de pays et nombres de tweets pour tous les `num` plus grand que 100 dans `res`. On pourra utiliser l'opération `res.items()` renvoyant le tableau des paires (clé, valeur) d'un dictionnaire dans une compréhension de tableaux permettant de filtrer ces paires.
   — Enfin, on pourra utiliser `dessine_barres` pour dessiner le graphe du tableau `res100`.
7. ◇ Pour chaque Emoji apparaissant dans les messages, afficher le graphique du nombre de tweets utilisant cet Emoji, pour les Emoji apparaissant plus de 50 fois. Les Emojis sont les caractères dont le code (que l'on peut obtenir avec la fonction Python `ord(c)`) est compris entre 126980 et 128592 (c'est une simplification, il y a d'autres codes de caractères).
8. ◇ Modifier la fonction `dessine_barres` pour qu'elle normalise les données et les affiche comme des pourcentages (i.e. chaque valeur est normalisée comme un pourcentage du total des valeurs).
9. ◇ Tracer les points correspondant aux coordonnées GPS (latitudes et longitudes).