UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Info. TP6: Génération et tracé de données

- https://learn.heeere.com/2020-infospichi-0b73/
- http://matplotlib.org/api/pyplot_api.html (doc pyplot)

Rappel : (pour la fin) rendu sous forme d'archive

Pour faire une **archive zip** de votre TP pour le rendu, exécutez, dans le terminal :

```
cd ~/info-spichi/
zip -r tp6.zip tp6/
```

Important : mise en place

- Suivre les consignes des TP précédents, et travailler dans un dossier tp6.

(Re)Nouveauté (fonctions avec plusieurs valeurs de retour)

Dans ce TP, nous allons utiliser des fonctions qui renvoient des n-uplets de valeurs. On peut par exemple définir une fonction comme ceci :

```
1 def min_et_max(arr):
2     return (np.min(arr), np.max(arr)) # il est accepté d'omettre les parenthèses ici
```

On utilise alors cette fonction comme ceci (si `a` est un tableau existant) :

```
1 (mi, ma) = min_et_max(a) # on pourrait omettre les parenthèses à gauche du =
2 # la ligne précédente est équivalente aux 3 lignes suivantes
3 t = min_et_max(a)
4 mi = t[0]
5 ma = t[1]
```

Exercice 1 – Générateur de données aléatoires

Rappel : pour cet exercice, travaillez dans un dossier `ex1` lui même dans le dossier `tp6`.

Rappel TP5: Bluecime veut compléter son système pour fournir des données météorologiques automatiquement aux stations de ski (température, ensoleillement, épaisseur de neige).

Dans 3 jours, une réunion est prévue dans une station de ski dans le but de présenter l'affichage des courbes et statistiques dérivées des mesures. Comme il n'y a qu'une semaine de données disponibles, les résultats à montrer sont assez limités. Afin de pouvoir montrer l'affichage sur plusieurs semaines tout en ayant des valeurs relativement plausibles, on demande à Régis de créer une générateur de données synthétiques.

Le lendemain, Régis se perd en forêt, pendant que vos collègues partent à sa recherche vous êtes chargé de continuer son travail.

Son programme est disponible dans le fichier `generation_donnees.py` téléchargeable sur le site du cours. Cela peut être réalisé directement depuis le terminal avec les commandes :

```
wget https://learn.heeere.com/2020-infospichi-0b73/raw/generation_donnees.py
```

Q1) Régis fait beaucoup d'erreur d'inattention (comme la plupart des gens) mais il a la (TRÈS TRÈS) mauvaise habitude d'écrire ses programmes sans les tester, exécutez le programme et corrigez ses erreurs. NB : On veut générer 30 jours de données, c'est à dire 720 heures. Important : Répondre aussi à la question suivante.

'Q2) Comment expliquer que l'erreur ligne 58 apparaît avant l'erreur ligne 16 ?

'Q3) À la ligne 18 on a `donnees[:, 0], donnees[:, 1]  = generer_temperatures(nb_jours)`. Est-ce que `donnees[:, :2] = generer_temperatures(nb_jours)` aurait été faux ? Pourquoi ?

Quelqu'un avait suggéré d'ajouter également une mesure de la vitesse du vent. Malheureusement, Régis n'a pas eu le temps de terminer la génération de ces données. Il a commencé mais il s'est laissé à lui même des commentaires « A FAIRE » pour les choses qu'il n'avait pas terminées.

Son idée était de définir la vitesse du vent comme suit:

- le vent n'est pas influencé par le cycle jour/nuit.
- le vent est constant sur des périodes de quelques heures.
- d'une période à l'autre, le vent change un peu.

Q4) Complétez la fonction `generer_vent` en finissant le travail de Régis.

Exercice 2 – Tracés des données

Rappel : pour cet exercice, travaillez dans un dossier `ex2` lui même dans le dossier `tp6`.

Q5) Créez un fichier `plot_donnees.py`, dans lequel vous allez générer un mois de données aléatoires.

Pour créer un groupe de graphiques, à partir d'une « figure » obtenue grâce à `plt.f()igure`, on peut utiliser la fonction `add_suplot` : `ax = fig.add_subplot(lignes, colonnes, id_subplot)`:

```
fig = plt.figure()
ax = fig.add_subplot(1, 2, 1)
ax.plot(tableau1)
ax = fig.add_subplot(1, 2, 2)
ax.plot(tableau2)
plt.show()
```

Ce code donnera 2 graphiques sur un même ligne, le premier contiendra le tracé de `tableau1`, le deuxième celui de `tableau2`.

Q6) Importez les modules nécessaires, dont le module `generation_donnees` qui correspond au fichier écrit à l'exercice précédent, et que vous devez copier à coté du fichier que vous venez de créer. Utilisez la fonction `generer_donnees` pour générer 30 jours de données (on pourra les sauver dans une variable `tableau`). Tracez ensuite sur deux lignes et deux colonnes, 4 graphiques : un graphique contenant les courbes de températures, un graphique contenant la courbe d'ensoleillement, un graphique contenant les courbes d'épaisseur de neige `en mètre`, et un graphique contenant la courbe de la vitesse du vent en `km/h`.

Q7) Comme dans le tp5, n'oubliez pas les légendes. `plt.legend` est disponible pour chaque "subplot" avec `ax.legend`.

Q8) Utilisez `ax.set_title(titre)` pour définir un titre explicite aux différents graphiques.

On peut modifier les valeurs affichées sur les axes de la façon suivante :

```
N = 30
tableau = generation_donnees.generer_donnees(N)
fig = plt.figure(figsize=(18,8))
ax = fig.add_subplot(1, 1, 1)
ax.plot(tableau[:, 0])

ax.set_xticks([0, 15*24, 30*24])
ax.set_xticklabels(["a", "b", "c"])
plt.show()
```

`ax.set_xticks` définit les indices des valeurs à afficher sur l'axe des abscisses, dans l'exemple, seront affichées 3 valeurs, une au début, une au milieu, et une à la fin de l'axe des abscisses. `ax.set_xticklabels` définit les valeurs à afficher sur l'axe des abscisses, dans l'exemple, seront affichées "a" au début, "b" au milieu, et "c" à la fin de l'axe des abscisses.

Note: utilisez `ax.set_yticks` et `ax.set_yticklabels` pour changer l'axe des ordonnées.

Q9) Sur chaque graphique, affichez sur l'axe des abscisses le début de chaque jour (c'est à dire l'heure 0, l'heure 24, etc.). Les jours seront numérotés de 1 à 30.

Les méthodes de la question précédente ne changent que l'affichage de l'axe. pour définir les limites réelles de des axes on utilise `ax.set_ylim(bottom, top)`

```
N = 30
tableau = generation_donnees.generer_donnees(N)
fig = plt.figure(figsize=(18,8))
ax = fig.add_subplot(1, 1, 1)
ax.plot(tableau[:, 0])

ax.set_ylim(0, 5)
plt.show()
```

Dans cet exemple l'axe des ordonnées commencera à 0 et finira à 5, les valeurs inférieures à 0 et celle supérieures à 5 ne seront donc pas affichées.

Note: utilisez `ax.set_xlim` pour changer l'axe des ordonnées.

Q10) Vous connaissez (en regardant comment sont générées les données) les valeurs maximales et minimales de chaque mesures, utilisez `ax.set_ylim` de sorte que les 4 graphiques aient chacun un axe des ordonnées constant qui vous parait adapté (et sans perdre/cacher des données).

Pour créer des diagrammes en "camembert" on utilise la méthode `ax.pie`:

```
N = 30
tableau = generation_donnees.generer_donnees(N)
fig = plt.figure(figsize=(18,8))
ax = fig.add_subplot(1, 1, 1)
ax.pie([50, 15, 25, 10], labels=["50%", "15%", "25%", "10%"])
plt.plot
```

Le premier tableau contient les proportion de chaque part, labels sert à légender les parts.

Q11) Dans une nouvelle figure (avec un seul subplot), utilisez `ax.pie` pour afficher les pourcentages d'heures où la moyenne des épaisseurs de neige est supérieure à 170cm inclue (fort enneigement), entre 169cm et 140cm inclues (enneigement moyen), et inférieure à 139cm inclue (enneigement faible).

Aide: Vous pouvez utiliser `np.bincount(tab)` qui permet de compter le nombre d'occurrences de chaque valeur d'un tableau **d'entiers**. `np.bincount([5,2,5])` par exemple renvoie `[0,0,1,0,0,2]`, aucun 0, aucun 1, un 2, aucun 3, aucun 4, deux 5 (s'arrête à la valeur maximale).

Aide: Étant donné un tableau numpy `t`, on peux transformer toutes ses valeurs en entier avec `t.astype(int)` (qui renvoie un nouveau tableau).

Aide: Extraire les épaisseurs de neige, faire leur moyenne (axis ?), transformer les valeurs en entier, utiliser bincount, puis sommer des parties (tranches) de l'histogramme obtenu, puis tracer le camembert.

Q12) Créez un groupe de graphiques de 1 ligne et 3 colonnes.

Q13) En utilisant la fonction `ax.bar(x, y)`, dans le 1er graphique, tracez les températures minimales au soleil mesurées à 0h, 1h , ..., 23h, dans le 2ème, les températures moyennes, et dans le 3ème, les

températures maximales.

Documentation `ax.bar`: `ax.bar(left, height, width)`

- `left` : tableau de valeur, coordonnées x du côté gauche des barres
- `height` : tableau de valeur, hauteur des barres
- `width` : valeur, largeur des barres

Q14) Grâce à cette documentation au sujet des arguments/paramètres de `ax.bar`, refaites la question précédente sur un seul graphique, de sorte qu'à chaque heure soient affichées 3 barres : la première pour la température minimale, la deuxième pour la température moyenne, et la troisième pour la valeur maximale