Programmer en Python
SVE Licence 1, Mathématiques
TP2 - Graphiques et importation de données

Installer des modules sous Thonny

Python est un langage composé de modules spécialisés. Nous allons maintenant installer des modules de Python qui ne sont pas chargés par défaut dans la version de base que nous avons utilisée jusqu'à présent. Il s'agit des modules

— `matplotlib` pour les outils graphiques

— `numpy` pour manipuler des vecteurs et des matrices

Dans Thonny, choisir le menu `Tools` puis `Manage Packages` ; taper le nom du module, le chercher on cliquant sur le bouton `search` puis l'installer. Si vous devez choisir un répertoire d'installer, choisir le répertoire `h:` sans quoi l'installation sera locale et vous ne la retrouverez pas si vous changez de machine au prochains TP....

Remarque : vous pourrez désormais utiliser ces modules sans avoir à les réinstaller.

1. Module matplotlib

A chaque session, il est nécessaire de charger le module. Par exemple, les commandes

```
import matplotlib.pyplot as plt
import numpy as np
```

permettent de charger le sous module `pyplot` du module `matplotlib` et le module `numpy`. Le module `numpy` permet de manipuler des vecteurs et des tableaux.

```
debut = -10
fin = 10
pas = 3
x = np.arange(debut,fin,pas)
f = 1+np.exp(x)
```

Vous remarquez que nous avons accès aux fonctions du module `numpy` via le prefixe `"np."`.

Exercice :

1. Taper les commandes ci-dessus puis afficher `x` et `f`. Quelle est la séquence créée par l'instruction `np.arange(debut,fin,pas)` ?

Le module `matplotlib` permet de générer des graphes depuis Python. On peut alors tracer des figures. Par exemple, nous pouvons tracer des fonctions classiques.

```
plt.plot(x,f) # par défaut python trace une courbe bleue
plt.xlabel("x")
plt.ylabel("f(x) = 1+exp(x)")
plt.title("Fonction exponentielle translatée de 1")
plt.show()
```

Vous remarquez que nous avons accès aux fonctions du module `pyplot` via le prefixe `"plt."`.
L'instruction `plt.show()` indique que la liste des commandes à exécuter pour créer le graphique est terminée et que Python peut afficher la figure. La liste est initialisée par la commande `plt.plot` ici.

Exercice :

2. La courbe devrait représenter la fonction continue et de dérivée continue $f : x \mapsto 1 + e^x$. Une fonction continue et de dérivée continue est lisse par nature. Or nous observons que la courbe tracée présente des "cassures" (ie que sa dérivée n'est pas continue). Quelle ligne de code doit-on modifier, parmi les 2 blocs d'instructions précédents, pour tracer une courbe plus lisse ?

On peut changer la couleur de la courbe. Il est inutile de charger à nouveau les modules si vous ajoutez les lignes au script précédent.

```
plt.plot(x,f,"k") #l'argument 'k' trace une courbe noire
plt.xlabel("x")
plt.ylabel("f(x) = 1+exp(x)")
plt.title("Fonction exponentielle translatée de 1")
plt.show()
```

Python est codé en anglais, les couleurs sont en général accessaible via leur initiale (en anglais). Par exemple `"r"` pour rouge, `"g"` pour vert, etc.

On peut aussi tracer des points issus de données.
On s'intéresse ici au lion du parc national GIR
`https://fr.wikipedia.org/wiki/Parc_national_et_sanctuaire_faunique_de_Gir`
et notamment à la dynamique de sa population dans le parc national. On créé deux listes

```
annees = [1969, 1974, 1979, 1985,1990, 1995, 2001, 2005, 2010, 2015]
effectifs = [177, 180, 205, 239, 284, 304,   329, 359, 411, 523]
```

On peut alors tracer les points avec les années en abscisse et les effectifs en ordonnées.

```
import matplotlib.pyplot as plt # inutile si le module est déjà chargé.
plt.plot(annees,effectifs,"r+") # trace un "+" rouge ("r")
plt.xlabel("Annee")
plt.ylabel("Effectif")
```

```
plt.title("Population de lions du parc GIR")
plt.grid()
plt.show()
```

⚠*Attention- Les commandes Python sont lues et exécutées par bloc. Il faut éviter de laisser des lignes vides dans les blocs d'instructions.*

On peut superposer plusieurs graphiques. Par exemple ici, on superpose les observations issues des recensements. , un modèle quadratique qui décrit l'évolution de la population.

```
a = 458810.401
b = -4671.840
c = 11.897
t = np.asarray(annees)/10
y = a+np.dot(b,t)+ np.dot(c,t**2)
# np.dot permet de multiplier
# chaque élément de la liste t par la constante b
plt.plot(annees,effectifs,"r+",label="Observations")
# trace un "+" rouge ("r")
plt.xlabel("Annee")
plt.ylabel("Effectif")
plt.title("Population de lions du parc GIR")
plt.grid()
plt.plot(annees,y,"b",label="Modèle") # trace la droite en bleu
plt.legend()
plt.show()
```

Les "label" permettent de créer une légende affichée par `plt.legend()`. On peut alors extrapoler (ie prolonger) la courbe du modèle pour prédire le nombre de lions en 2020.

```
a = 458810.401
b = -4671.840
c = 11.897
t = np.asarray(annees)/10
y = a+np.dot(b,t)+ np.dot(c,t**2)
plt.plot(annees,effectifs,"r+",label="Observations")
plt.xlabel("Annee")
plt.ylabel("Effectif")
plt.title("Population de lions du parc GIR")
plt.grid()
plt.plot(annees,y,"b",label="Modèle")
plt.plot(2020,a+b*2020/10+c*(2020/10)**2,"go",label="Extrapolation")
plt.legend()
plt.show()
```

Exercice :

3. Tracer en vert la fonction qui à x associe $3 * \log(x) - 1$ pour x variant de 0.1 à 10 avec un pas de 0.1. Utiliser la fonction `np.log()`.
   Ajouter les légendes et le titre appropriés.
4. Tracer en rouge la fonction qui à x associe $3 * \sqrt{x} - 1$ pour x variant de 0.1 à 10 avec un pas de 0.1. Utiliser la fonction `np.sqrt()`. Ajouter les légendes et le titre appropriés.
5. D'après la théorie, quelle est la fonction qui croit le plus vite en $+\infty$? qui décroit le plus vite en 0 ?
   Vérifier en superposant les 2 courbes.
6. Tracer `plt.plot(x,np.exp(np.log(x)))`. Quelle est la forme de la courbe obtenue ? Pourquoi obtient-on cette forme ?
7. Donner le domaine de définition et dériver les fonctions suivantes. Pour chaque fonction tracer sa dérivée (vous calculerez les dérivées à la main sur une feuille, comme en cours de math) et ajouter sur le même graphe la fonction $\tilde{f}'(x) = \frac{f(x+h)-f(x)}{h}$ pour $h = 0.0001$.
   (a) $f(x) = \sqrt{x^2+1}$
   (b) $f(x) = xe^{-x}$
8. Chercher une équation de la tangente de la courbe au point indiqué. Tracer la courbe et sa tangente sur un même graphique.

   (a) $y = x^4 + 2x^2 - x$ au point d'abscisse 1
   (b) $y = e^x - \sin x$ au point d'abscisse 0

En biologie, il est souvent intéressant d'utiliser les échelles logarithmiques.
Exercice

9. Représentation en échelle logarithmique (I)
   Tracer une représentation de la fonction qui a t associe $f(t) = exp(-t/5.0)$ pour t variant de 0.01 à 20 avec un pas de 0.01.
   Nous allons maintenant utiliser une représentation de la même courbe en échelle logarithmique. Pour cela, nous pouvons utiliser la fonction `plt.semilogy`. Cette fonction prend en entrée les mêmes arguments que la fonction `plot`. Quelle est la forme de la courbe obtenue ? Pourquoi ?
10. Représentation en échelle logarithmique (II)
    Tracer tout d'abord le graphique correspondant aux instructions ci-dessous

```
plt.plot(t, np.sin(2*np.pi*t))
plt.grid(True)
plt.show()
```

    Quelle est la fonction tracée ?
    Comparer le premier graphique à celui ci

```
plt.plot(np.log(t), np.sin(2*np.pi*t))
plt.title("semilogx")
plt.grid(True)
plt.show()
```

Qu'observe t'on ?
On trace maintenant

```
plt.semilogx(t, np.sin(2*np.pi*t))
plt.title("semilogx")
plt.grid(True)
plt.show()
```

Y a t'il des différences entre ce graphique et le précédent ?

Le tableau ci-dessous donne des abondances en individu par ml en fonction du temps en jours. Ces données pourraient par exemple décrire la croissance d'une population de *Paramecium aurelia* à 10 degrés celsius dans une solution stérile de peptone protéolysée (1% masse/vol).

Temps (jours)	0	3	6	9	12
Abondance (ind/ml)	10	102	1221	2435	5184

Exercice :

11. Tracer les abondances en fonction du temps. On utilisera une croix pour chaque donnée. Ajouter les légendes.
12. Utiliser la fonction `semilogy` pour représenter les mêmes données.
    Les arguments de cette fonction sont les mêmes que ceux de la fonction `plot`.
13. Commenter les graphiques obtenus.

2. Importation de données

Il est souvent utile pour analyser des données d'importer des tables `.csv`. En Python, ceci est facile en utilisant le module `pandas`.

Par exemple, la table suivante contient des mesures météorologiques ainsi que des mesures de concentration en particules fines PM2.5 dans l'atmosphère de Shanghai.

```
import pandas as pd
filename = "https://perso.univ-rennes1.fr/valerie.monbet/
    DATA/ShanghaiPM20100101_20151231.csv"
PM = pd.read_csv(filename)
print(PM)
```

On observe que PM est une table à 17 colonnes et 52583 lignes. En effet, testez la commande

```
print("Dimension de la table")
print(PM.shape)
```

Vous pouvez lister les noms de colonnes

```
print("Liste des variables")
print(list(PM))
```

Exercice :

14. Donner la liste des noms de colonnes de la table `PM`.
15. Tracer la série temporelle des températures :

    ```
    import datetime as date
    rng = pd.date_range("1/1/2010", periods=len(PM["TEMP"]), freq='H')
    plt.plot(rng,PM["TEMP"])
    plt.show()
    ```

    On remarque que
    - la commande `pd.date_range("1/1/2010", periods=len(PM["TEMP"]), freq='H')` permet de créer une suite de dates démarrant le 1er janvier 2010 à 00h00, de longeur `len(PM["TEMP"])` et de fréquence horaire ;
    - la commande `PM["TEMP"]` donne accès aux valeurs de température de la table `PM`.
    Ajouter les légendes appropriées et le titre. On remarque que la commande `PM["TEMP"]` donne accès aux valeurs de température de la table `PM`.
16. Tracer la série temporelle des concentrations en PM2.5 à Jingan.
17. Tracer un nuage de points des concentrations en PM2.5 à Jingan en fonction des pressions (PRES). Pour quelles pressions sont observés les pics de pollution ?
18. Faire un graphique qui permette de répondre à la question suivante : pour quelles températures sont observés les pics de pollution ?