TP 10 – Fichiers et bibliothèques

1 Lecture de valeurs dans un fichiers

1. Enregistrer les fichiers `donnees.txt` et `regLinData.txt` dans le même dossier que le programme
2. Taper le code suivant puis commenter

```
x,y = [],[]
with open("donnees.txt", "r") as inp:
    for line in inp:
        xi, yi = line.split()
        x.append(float(xi))
        y.append(float(yi))
print(x)
print(y)
```

3. Tapez les commandes suivantes dans la console puis commentez.

```
Fichier = open("donnees.txt", "r")
L = Fichier.readline()
print(L)
Lbis = Fichier.readlines()
print(Lbis)
Fichier.close()
Fichier.readlines()
```

4. La ligne L est elle présente dans Lbis ?
5. Chaque ligne du fichier `donnees.txt` contient les deux coordonnées d'un point du plan. Modifier les commandes précédentes pour écrire un programme qui calcule la moyenne des abscisses et la moyenne des ordonnées des points du fichier `donnees.txt`.

2 Affichage des valeurs : le module matplotlib

Les fonctions de base de python sont générales. Pour des travaux plus spécialisés (par exemple le calcul scientifique), des auteurs indépendants ont créé de nouvelles fonctions, stockées dans des programme appelées modules ou bibliothèques de fonctions.
Pour accéder aux fonctions disponibles dans un module on doit le charger à l'aide de la commande import. Par exemple, pour utiliser la fonction cosinus disponible dans le module math on utilise les commandes

```
import math
c = math.cos(5*math.pi / 6)
print(c)
```

Pour importer seulement la fonction cos on écrit :

```
from math import cos
c = cos(1)
print(c)
```

On peut aussi importer toutes les commandes du module math via la commande :

```
from math import *
```

Pour obtenir de l'aide concernant un module (par exemple le module math) on utilise la commande

```
import math
help(math)
```

Voici des modules usuels :

- math : fonctions et constantes mathématiques de base (sin, cos, exp, pi...).
- numpy : calcul scientifique / matriciel.
- matplotlib : affichage de graphiques.
- random : génération de nombres aléatoires.
- time : accès à l'heure de l'ordinateur et aux fonctions gérant le temps.
- urllib : récupération de données sur internet depuis Python.
- sys : interaction avec l'interpréteur Python, passage d'arguments.
- os : dialogue avec le système d'exploitation.

Questions :

6. **Dans une console,** entrez les commandes suivantes et déduisez en ce que fait la commande `xlim`

   ```
   import matplotlib.pyplot as plt
   help(plt.xlim)
   ```

7. Tapez les commandes suivantes et commentez les :

```
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = 8, 8 # ajuste la taille des figures
plt.plot(x, y, "bo", label="donnees")
plt.xlabel("x")
plt.ylabel("y")
plt.xlim(0, 11)
plt.legend()
plt.title("Regression Lineaire")
plt.show()
```

8. Que peut-on remarquer sur les points dans le fichier `donnees.txt` ?
Cela vous rappelle-t-il quelque chose concernant le cours de chimie ? (indice : regardez le titre suivant)

3 Régression linéaire

En statistique, on cherche souvent à repérer des tendances dans des suites de données. Un des modèles les plus simples est le modèle linéaire (ou dit de proportionnalité). Dans le cadre d'une suite de points du plan, il s'agit intuitivement de vérifier si les points sont "presque alignés".
Si on fait l'hypothèse qu'une suite de points $P_k = (x_k; y_k)$ sont alignés sur la droite d'équation $y = ax + b$, alors l'erreur dans notre hypothèse pour un point k donné est :

$$e_k = |y_k - ax_k - b|$$

et l'erreur globale d'approximation est définie (au sens des moindres carrés) par

$$Q(a; b) = \sum_{k=1}^{N} (y_k - ax_k - b)^2$$

(*le choix du passage au carré est admis ici, et sera justifié dans le cours de mathématiques sur les espaces euclidiens*).

En régression linéaire, on part d'une situation ou l'on ne connait ni a ni b. Pour trouver une droite "proche" des points $P_1, \ldots, P_N$ on veut trouver des valeurs de a et b qui donnent la plus petite valeur de $Q(a; b)$.
Le cours de mathématique de fin d'année vous permettra d'étudier des fonctions en deux variables (par calcul de dérivées) et ainsi de trouver les valeurs de a et b qui minimisent $Q(a; b)$.
En attendant vous admettrez que les valeurs de a et b qui minimisent $Q(a; b)$ sont données par les formules suivantes :

$$a = \frac{\left(N \times \sum_{k=1}^{N} x_k y_k\right) - \left(\sum_{k=1}^{N} x_k\right) \times \left(\sum_{k=1}^{N} y_k\right)}{\left(N \times \sum_{k=1}^{N} x_k^2\right) - \left(\sum_{k=1}^{N} x_k\right)^2}$$

et

$$b = \frac{\left(\sum_{k=1}^{N} x_k^2\right) \times \left(\sum_{k=1}^{N} y_k\right) - \left(\sum_{k=1}^{N} x_k\right) \times \left(\sum_{k=1}^{N} x_k y_k\right)}{\left(N \times \sum_{k=1}^{N} x_k^2\right) - \left(\sum_{k=1}^{N} x_k\right)^2}$$

4 Utilisation du module numpy

Le module `numpy` permet entre autres de faire du calcul matriciel. On pourra se référer à l'aide mémoire du concours central concernant le calcul matriciel.

1. écrivez un code qui

- ouvre le fichier `regLinData.txt`
- créé les vecteurs $x = (x_1; \dots x_N)$ et $y = (y_1; \dots; y_N)$.

2. étant donné deux vecteurs $x = (x_1; \dots x_N)$ et $y = (y_1; \dots; y_N)$ les sommes apparaissant dans les formules pour a et b peuvent s'écrire matriciellement sous la forme :

$$\sum_{k=1}^{N} x_k y_k = x.y^T$$

Cette somme s'appelle produit scalaire usuel de x et y.

a. Quelle commande numpy permet de calculer cette somme ?

b. Comment calcule-t-on la somme $\sum_{k=1}^{N} x_k^2$?

c. En déduire un programme permettant de calculer les valeurs de a et b.

3. Sur un même graphique, tracez

- les points du fichier `regLinData.txt` (représentés par des points bleus comme à la section 2)
- La droite d'équation $y = ax + b$ (en rouge)

$\sim$