Exercice 1 : **Formatage de donnée 1 : une liste de noms**

Le fichier `communes.txt` contient la liste des communes de France métropolitaine.
Utiliser le programme `lecture.py` pour charger la chaine de caractère en mémoire.
Le programme et le fichier doivent être copiés dans le répertoire de travail. On pourra vérifier que le chargement de le tableau a bien été effectuée en consultant l'explorateur de variables.

Q1 : Indiquer ce que réalise le programme pour chaque ligne.

On cherche à extraire des données de ce tableau :

Q2 : Combien y a-t-il de communes en France ?
Q3 : Afficher le nom de la commune ayant le plus de caractère.
Q4 : Afficher le nombre de noms de commune ayant ce nombre de caractère.
Q5 : Afficher le nom de la commune qui contient le plus grand nombre d'occurrences de la lettre « s ».

Exercice 2 : **Formatage de donnée 2 : deux listes formant un tableau**

Le fichier `tableau.txt` contient deux colonnes : le nom des communes de France, et leur population.

Ecrire le programme qui charge les données en mémoire vive sous la forme d'un tableau à deux colonnes.
Convertir la deuxième colonne en nombre entier.

Q1 : Combien y a-t-il de ville en France ?
Q2 : Afficher la plus grande population
Q3 : Afficher la plus petite population
Q4 : Afficher le nom de la ville ayant la plus grande population
Q5 : Afficher le nom de la ville ayant la plus petite population
Q6 : Afficher la population de la commune d'Aubagne
Q7 : Y a-t-il des communes dont le nombre d'habitants est inférieur au nombre de lettre dans son nom ?

Exercice 3 : **Traitement de données expérimentales**

Dans le dossier « traitement_donnée_exp », se trouve des relevés d'expérimentation réalisées aux laboratoires par des logiciels dédiés.

Ouvrir le fichier texte « maxpid.txt ». Observer son contenu :

Q1 : Quelles sont les informations contenues dans ce fichier ? Sous quelles formes ?
Q2 : Combien de lignes correspondent aux entêtes ?
Q3 : Comment distinguer les colonnes ?
Q4 : Quels est le séparateur entre les unités et les dizièmes ?

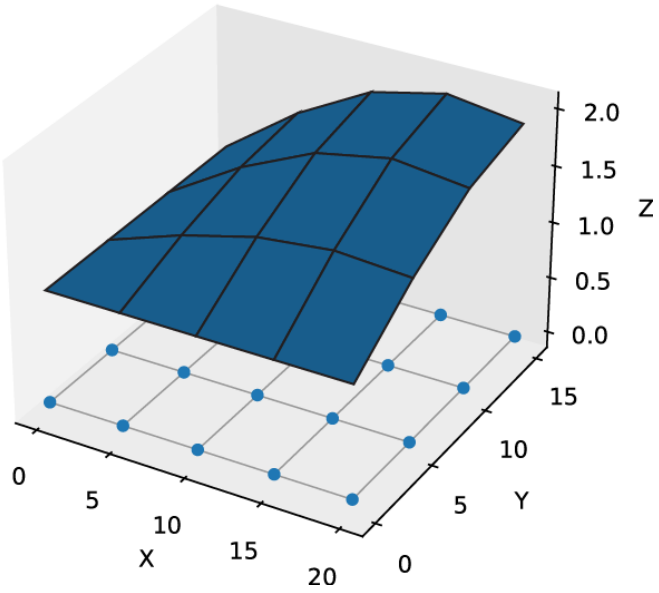
Créer un programme qui :
- charger ce fichier ;
- remplace les virgules par des points ;
- supprimer les deux premières lignes ;
- crée une liste des instants de mesures ;
- crée une liste des valeurs mesurées de la position ;
- crée un tableau de 3 colonnes & 55 lignes contenant toutes les valeurs.

Tracer le graphe de la position en fonction du temps.

Exercice 4 : **Echantillonnage du plan**

Objectif : tracer une surface :

$$f(x,y) = \sin\left(\frac{x}{10}\right) \cdot \sin\left(\frac{y}{10}\right)$$



On cherche à tracer une surface définie par une fonction $z = f(x,y)$ où $x \in [0;20]$ et $y \in [0;15]$.

Le programme ne pouvant pas traiter toutes les valeurs de x et y, il est nécessaire de discrétiser le plan (x,y). Nous ne garderons qu'un nombre $n \times m$ fini de points du plan π, régulièrement espacés sur l'ensemble de définition.

(exemple pour $5 \times 4 = 20$ points)

Q1: Ecrire un programme qui crée une liste Lx de $n = 5$ nombres décimaux également répartis entre 0 et 20.

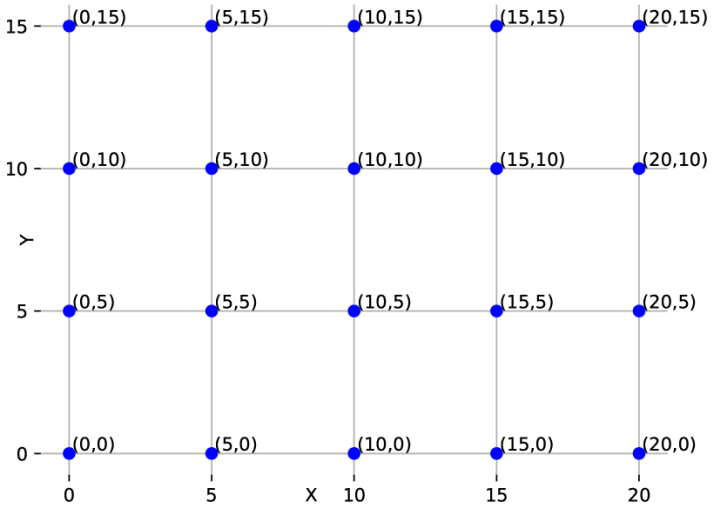
Pour la suite, python aura besoin de deux tableaux `X` et `Y` :

- `X` : tableau formé des abscisses de chaque point ;
- `Y` : tableau formé des ordonnées de chaque point.

Par exemple :

```
X= [[0, 5, 10, 15, 20],
    [0, 5, 10, 15, 20],
    [0, 5, 10, 15, 20],
    [0, 5, 10, 15, 20]]

Y= [[0, 0, 0, 0, 0],
    [5, 5, 5, 5, 5],
    [10, 10, 10, 10, 10],
    [15, 15, 15, 15, 15]]
```



Q2: Ecrire un programme qui crée les tableaux `X` et `Y` de $n \times m$ valeurs.

Q3: Ecrire un programme qui crée le tableau `Z` correspondant à l'altitude de chacun des points.

On peut alors tracer la surface avec les trois instructions :

```
import numpy as np
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig1.add_subplot(111, projection='3d')
ax.plot_surface(np.array(X),np.array(Y),np.array(Z))
```

Exercice 5 : **Codage d'une image en noir et blanc**

Le fichier `felix.txt` contient une suite de caractères correspondant à une image carrée en noir et blanc.

Q1 : Combien de pixels contient cette image ? Quelles sont ses dimensions ?
Q2 : Convertir la chaine de caractère en un tableau de type `array`.
Q3 : Afficher l'image correspondante. Qui est le personnage représenté ?

Exercice 6 : **Travailler avec des images**

Partie 1 : Codage d'une image

Les fichiers fournis par le professeurs contiennent :

- `Gall-Peters_projection.jpg` : une image de la terre en projection selon Gall-Peters
- `image.py` : un programme à compléter pour réaliser les fonctions demandées

Dans spyder, ouvrir le fichier `image.py`

Q1 : Expliquer ce qui y est fait lors des 8 premières lignes du programme.
Q2 : Quelles sont les dimensions de l'image (horizontalement et verticalement) ? Quel est le type colorimétrique de l'image ? Quelles sont les dimensions du tableau correspondant ?

Afficher dans la console `terretab`.

Q3 : Justifier le type de valeur utilisé dans le tableau.
Q4 : Déterminer le profil colorimétrique du pixel situé en bas à droite sur l'image.

Partie 2 : Portion d'image

Q5 : Ecrire un programme qui affiche uniquement la moitié gauche de l'image.

Partie 3 : Rotation d'une image

On cherche à faire tourner cette image de +90°.
On va donc créer un nouveau tableau `terretabnew` qui sera converti en image par les deux dernières lignes du programme.

Q6 : Supprimer les marques de commentaire # du programme. A la vingtième ligne, indiquer entre crochet les dimensions du nouveau tableau.
Q7 : En utilisant des boucles, construire un nouveau tableau correspondant à l'image rechercher.

Faire valider par le professeur.

Partie 4 : Tracer d'une ligne

Q8 : Proposer un programme qui trace une ligne noire dans l'image précédente, entre deux pixels qui seront définis par l'utilisateur.

Exercice 7 : **Image : manipulation des couleurs**

Pour cet exercice, il faut utiliser l'image `lena.png`, disponible sur le site www.touron.tech.

Partie 1 : Convertir une image : couleur vers nuance de gris

Pour transformer une image en couleur en image en niveaux de gris, nous allons nous contenter de calculer la moyenne pixel par pixel des trois composantes du triplet (R,V,B).

Q1. Ecrire une fonction `niveau_gris` ayant pour argument une chaîne de caractères étant le nom du fichier de l'image initiale. Cette fonction doit créer et afficher une nouvelle image en niveau de gris.

Partie 2 : Histogramme de luminosité : comptage rapide

Un *histogramme* représente le mode de répartition des pixels dans une image en traçant le graphe du nombre de pixels correspondant à chaque niveau d'intensité lumineuse en fonction de cette intensité (0 à 255).

Q1. Ecrire un programme qui compte le nombre de pixels de chaque niveau d'intensité lumineuse de l'image en nuance de gris.

Q2. Ecrire une fonction qui trace l'histogramme associé à une image en nuance de gris.

Partie 3 : Négatif d'une image

Pour créer une image, on peut procéder par addition de lumières colorées sur un fond noir (principe du vidéo projecteur numérique) ou par soustraction de couleur à partir une lumière blanche (principe des cartouches d'encre).

Couleurs par addition

Couleurs par soustraction

(voir la version numérique du sujet en couleur)

Une image convertie d'un format dans l'autre est transformée par une « image en négatif ».

Une image négative est une image dont les couleurs ont été inversée par rapport à l'originale ; par exemple le rouge devient cyan, le vert devient magenta, le bleu devient jaune et inversement. Les régions sombres deviennent claires, le noir devient blanc.
Les couleurs primaires étant codées sur un octet (256 nuances), la couleur opposée à x se calcule par : $255 - x$

Q1. Ecrire une fonction `inverse` ayant pour argument une chaîne de caractères étant le nom du fichier de l'image initiale. Cette fonction doit créer et afficher une nouvelle image négative de la première.

Exercice 8 : **Image : Détection de bord**

Pour cet exercice, il faut utiliser l'image `shapes.bmp`, disponible sur le site www.touron.tech.

Le bord d'une figure particulière, insérée dans une photo, est reconnaissable par une forte variation de l'intensité lumineuse. Cette variation est quantifiable en calculant la dérivée de l'intensité lumineuse $I(x, y)$, où x et y sont les coordonnées verticale et horizontale du pixel dans l'image.

Une approximation de la dérivée dans la direction x s'écrit :

$$D_x(x, y) = \frac{I(x+1, y) - I(x-1, y)}{2\Delta x}$$

($\Delta x = 1$ pixel)
En termes de traitement informatique, cela correspond à la convolution de la matrice équivalente à l'image par la matrice :

$$\frac{1}{2}[-1,0,1]$$

Une approximation plus fine est obtenue par l'opérateur de Sobel selon x :

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Et selon y :

$$S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Q1. Ecrire un programme qui réalise et qui affiche les dérivées selon x et y de l'image originale par convolution par l'opérateur de Sobel (les dimensions de l'image produite auront 2 pixels de moins que celles de l'originale, horizontalement et verticalement).

Q2. On souhaite combiner les dérivées D_x et D_y en calculant pour chaque pixel la norme du vecteur formé par ces deux valeurs : $N(x, y) = \sqrt{D_x(x, y)^2 + D_y(x, y)^2}$. Ecrire un programme qui réalise et affiche l'image formée par la norme des deux dérivées.