Fichiers

Tu vas apprendre à lire et à écrire des données dans des fichiers.

Vidéo ■ Fichiers - partie 1 - Écrire un fichier
Vidéo ■ Fichiers - partie 2 - Lire un fichier
Vidéo ■ Fichiers - partie 3 - Images

Cours 1 (Écrire dans un fichier)**.**
Écrire dans un fichier est presque aussi facile que d'afficher une phrase à l'écran. Voici à gauche un programme qui écrit deux lignes dans un fichier appelé `mon_fichier.txt` ; à droite le fichier résultant qui s'affiche dans un éditeur de texte.

```
fic = open("mon_fichier.txt","w")

fic.write("Bonjour le monde\n")

ligne = "Coucou\n"
fic.write(ligne)

fic.close()
```

```
Bonjour le monde
Coucou
```

Explications.

- La commande `open` permet d'ouvrir un fichier. Le premier argument est le nom du fichier. Le second argument est ici `"w"` pour dire que l'on veut écrire dans le fichier (*write* en anglais).
- On ne travaille pas avec le nom du fichier, mais avec la valeur renvoyée par la fonction `open`. Ici nous avons nommé `fic` ce fichier-objet. C'est avec cette variable `fic` que l'on travaille désormais.
- On écrit maintenant dans le fichier presque comme on afficherait une phrase à l'écran. L'instruction est `fic.write()` où l'argument est une chaîne de caractères.
- Pour passer à la ligne, il faut ajouter le caractère de fin de ligne `"\n"`.
- Il est important de fermer son fichier quand on a fini d'écrire. La commande est `fic.close()`.
- Les données à écrire sont des chaînes, donc pour écrire un nombre, il faut d'abord le transformer par `str(nombre)`.

Cours 2 (Lire un fichier)**.**
C'est tout aussi facile de lire un fichier. Voici comment faire (à gauche) et l'affichage par `Python` à l'écran (à droite).

```
fic = open("mon_fichier.txt","r")

for ligne in fic:
    print(ligne)

fic.close()
```

```
Bonjour le monde

Coucou
```

Explications.

- La commande `open` est cette fois appelée avec l'argument `"r"` (pour *read*), elle ouvre le fichier en lecture.
- On travaille de nouveau avec un fichier-objet nommé ici `fic`.
- Une boucle parcourt tout le fichier ligne par ligne. Ici on demande juste l'affichage de chaque ligne.
- On ferme le fichier avec `fic.close()`.
- Les données lues sont des chaînes, donc pour obtenir un nombre, il faut d'abord le transformer par `int(chaine)` (pour un entier) ou `float(chaine)` (pour un nombre à virgule).

Activité 1 (Lire et écrire un fichier)**.**

Objectifs : écrire un fichier de notes, puis le lire pour calculer les moyennes.

1. Génère au hasard un fichier de notes, nommé `notes.txt`, qui est composé de lignes ayant la structure :

   `Prenom Nom note1 note2 note3`

   Par exemple :

   ```
   Robin Dubois 5.0 16.5 19.5
   Tintin Dubois 8.5 15.5 16.5
   Gargamel Lupin 15.0 18.0 5.5
   Tintin Tchoupi 18.0 11.0 13.0
   Hermione Lupin 19.5 10.5 9.0
   James Tchoupi 10.0 11.0 6.0
   Alice Voldemort 13.0 16.5 20.0
   ```

   Indications.
   - Construis un liste de prénoms `liste_prenoms = ["Tintin","Harry","Alice",...]`. Puis choisis un prénom au hasard par la commande `prenom = choice(liste_prenoms)` (il faut importer le module `random`).
   - Même chose pour les noms !
   - Pour une note, tu peux choisir un nombre au hasard avec la commande `randint(a,b)`.
   - **Attention !** N'oublie pas de convertir les nombres en une chaîne de caractères pour l'écrire dans le fichier : `str(note)`.
2. Lis le fichier `notes.txt` que tu as produit. Calcule la moyenne de chaque personne et écrit le résultat dans un fichier `moyennes.txt` où chaque ligne est de la forme :

   `Prenom Nom moyenne`

   Par exemple :

```
Robin Dubois 13.67
Tintin Dubois 13.50
Gargamel Lupin 12.83
Tintin Tchoupi 14.00
Hermione Lupin 13.00
James Tchoupi 9.00
Alice Voldemort 16.50
```

Indications.

- Pour chaque ligne lue du fichier `notes.txt`, tu récupères les données dans une liste par la commande `ligne.split()`.
- **Attention !** Les données lues sont des chaînes de caractères. Tu peux convertir une chaîne `"12.5"` en le nombre `12.5` par la commande `float(chaine)`.
- Pour convertir un nombre en une chaîne avec seulement deux décimales après la virgule, tu peux utiliser la commande `'{0:.2f}'.format(moyenne)`.
- N'oublie pas de fermer tous tes fichiers.

Cours 3 (Fichiers au format *csv*)**.**
Le format *csv* (pour *comma-separated values*) est un format très simple de fichier texte contenant des données. Chaque ligne du fichier contient des données (des nombres ou du texte). Sur une même ligne les données sont séparées par une virgule (d'où le nom du format, même si d'autres séparateurs sont possibles).
Exemple. Voici un fichier qui contient les noms, prénoms, années de naissance, la taille ainsi que le nombre de prix Nobel reçus :

```
CURIE,Marie,1867,1.55,2
EINSTEIN,Albert,1879,1.75,1
NOBEL,Alfred,1833,1.70,0
```

Activité 2 (Format csv)**.**

Objectifs : écrire un fichier de données au format csv*, puis le lire pour un affichage graphique.*

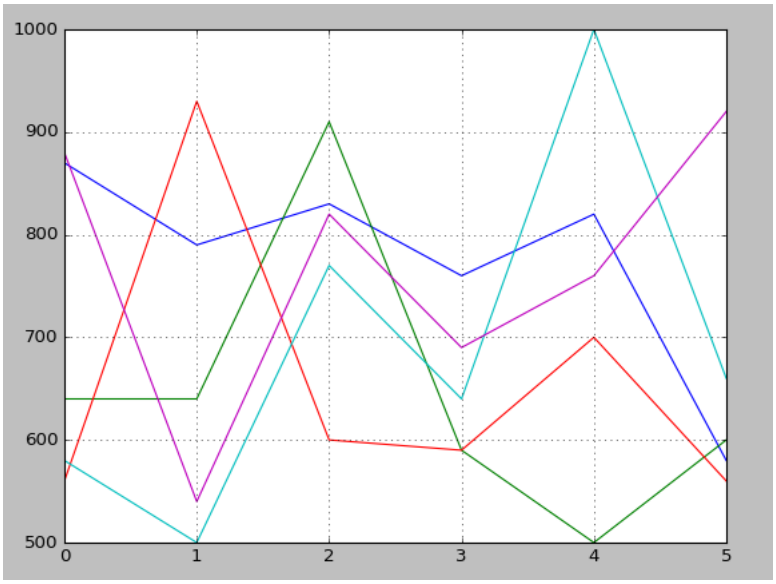
1. Génère un fichier `ventes.csv` des chiffres de ventes (tirés au hasard) d'une enseigne de sport. Voici un exemple :

```
Meilleures ventes de la société 'Pentathlon'

,2013,2014,2015,2016,2017,2018

Vélo VTT,870,790,830,760,820,580
Planche de surf,640,640,910,590,500,600
Chaussures de courses,560,930,600,590,700,560
Raquette de badminton,580,500,770,640,1000,660
Ballon de volley,880,540,820,690,760,920
```

- Les données débutent à partir de la cinquième ligne.
- Le fichier produit respecte le format *csv* et doit pouvoir être lu par *LibreOffice Calc* par exemple.

	A	B	C	D	E	F	G	H
1	Meilleures ventes de la société 'Pentathlon'							
2								
3		2013	2014	2015	2016	2017	2018	
4								
5	Vélo VTT	870	790	830	760	820	580	
6	Planche de surf	640	640	910	590	500	600	
7	Chaussures de courses	560	930	600	590	700	560	
8	Raquette de badminton	580	500	770	640	1000	660	
9	Ballon de volley	880	540	820	690	760	920	
10								

2. Lit le fichier `ventes.csv` pour afficher les courbes de ventes.



Indications.

- Le package `matplotlib` permet d'afficher facilement des graphiques, il s'appelle souvent avec l'instruction :

```
import matplotlib.pyplot as plt
```

- Voici comment visualiser deux listes de données `liste1` et `liste2` :

```
plt.plot(liste1)
plt.plot(liste2)
plt.grid()
plt.show()
```

Cours 4 (Images *bitmap*)**.**

Il existe un format simple de fichier, appelé format *bitmap*, qui décrit pixel par pixel une image. Ce format se décline en trois variantes selon que l'image est en noir et blanc, en niveaux de gris ou bien en couleurs.

Image noir et blanc, le format « pbm ».

L'image est décrite par des 0 et des 1.

Voici un exemple : le fichier `image_nb.pbm` à gauche (lu comme un fichier texte) et à droite sa visualisation (à l'aide d'un lecteur d'images, ici très agrandi).

```
P1
4 5
1 1 1 1
1 0 0 0
1 1 1 0
1 0 0 0
1 1 1 1
```

Voici la description du format :

- Première ligne : l'identifiant `P1`.
- Deuxième ligne : le nombre de colonnes puis le nombre de lignes (ici 4 colonnes et 5 lignes).
- Puis la couleur de chaque pixel ligne par ligne : `1` pour un pixel noir, `0` pour un pixel blanc. Attention : c'est contraire à la convention habituelle !

Image en niveaux de gris, le format « pgm ».
L'image est décrite par différentes valeurs pour différents niveaux de gris. Voici un exemple : le fichier `image_gris.pbm` à gauche et à droite sa visualisation.

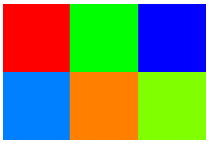
```
P2
4 5
255
  0   0   0   0
192 192 192 192
192 255 128 128
192 255  64  64
192   0   0   0
```

Voici la description du format :

- Première ligne : l'identifiant est cette fois `P2`.
- Deuxième ligne : le nombre de colonnes puis le nombre de lignes.
- Troisième ligne : la valeur maximale du niveau de gris (ici 255).
- Puis le niveau de gris de chaque pixel ligne par ligne : cette fois `0` pour un pixel noir, la valeur maximale pour un pixel blanc et les valeurs intermédiaires donnent des gris.

Image en couleurs, le format « ppm ».
L'image est décrite par trois valeurs par pixel : une pour le rouge, une pour le vert, une pour le bleu. Voici un exemple : le fichier `image_coul.ppm` à gauche et à droite sa visualisation.

```
P3
3 2
255
255   0   0     0 255 0     0   0 255
  0 128 255   255 128 0   128 255   0
```



Voici la description du format :

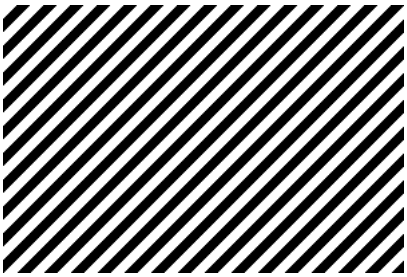
- Première ligne : l'identifiant est maintenant `P3`.
- Deuxième ligne : le nombre de colonnes puis le nombre de lignes.
- Troisième ligne : la valeur maximale des niveaux de couleur (ici 255).

- Puis chaque pixel est décrit par 3 nombres : le niveau de rouge, celui de vert puis celui de bleu (système RVB, *RGB* en anglais). Par exemple le premier pixel est codé par $(255, 0, 0)$ c'est donc un pixel rouge.

Activité 3 (Images *bitmap*)**.**

Objectifs : définir tes propres images pixel par pixel.

1. Génère un fichier `image_nb.pbm` qui représente une image en noir et blanc (par exemple de taille 300×200) selon le motif suivant :

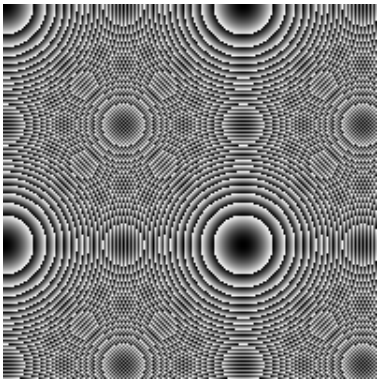


Indications. Si i désigne le numéro de ligne et j le numéro de colonne (en partant du haut à gauche), alors le pixel en position (i, j) est blanc si $i + j$ est compris entre 0 et 9, ou compris entre 20 et 29, ou entre 40 et 49,... Ce qui s'obtient par la formule :

```
coul = (i+j)//10 % 2
```

qui renvoie 0 ou 1 comme désiré.

2. Génère un fichier `image_gris.pgm` qui représente une image en niveaux de gris (par exemple de taille 200×200 avec 256 niveaux de gris) selon le motif suivant :

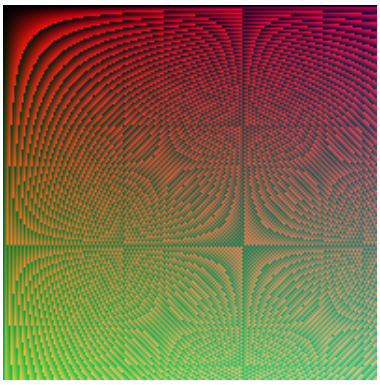


Indications. Cette fois la formule est :

```
coul = (i**2 + j**2) % 256
```

qui renvoie un entier entre 0 et 255.

3. Génère un fichier `image_coul.ppm` qui représente une image en couleurs (par exemple de taille 200×200 avec 256 niveaux de rouge, vert et bleu) selon le motif suivant :



Indications. Cette fois la formule est :

```
R = (i*j) % 256
V = i % 256
B = (i + j)//3 % 256
```

qui donne les niveaux de rouge, vert et bleu du pixel (i, j).

4. Écris une fonction `inverser_couleurs_nb(fichier)` qui lit un fichier image noir et blanc `.pbm` et crée un nouveau fichier dans lequel les pixels blancs sont devenus noirs et inversement.

   Exemple : à gauche l'image de départ, à droite l'image d'arrivée.

5. Écris une fonction `couleurs_vers_gris(fichier)` qui lit un fichier image couleur au format `.ppm` et crée un nouveau fichier au format `.pgm` dans lequel les pixels couleurs sont transformés en niveaux de gris.

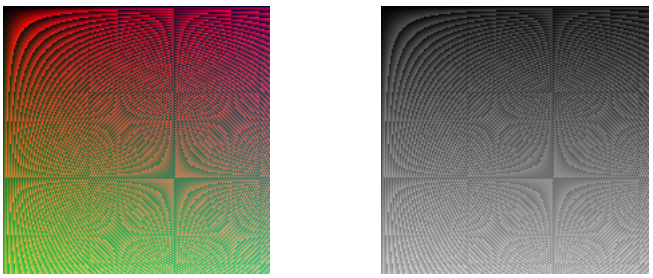
   Tu peux utiliser la formule :

$$G = 0,21 \times R \ + \ 0,72 \times V \ + \ 0,07 \times B$$

   où

   - R, V, B sont les niveaux de rouge, vert et bleu du pixel coloré,
   - G est le niveau de gris du pixel transformé.

   Exemple : à gauche l'image de départ en couleur, à droite l'image d'arrivée en niveau de gris.



Activité 4 (Distance entre deux villes)**.**

Objectifs : lire les coordonnées des villes et écrire les distances entre elles.

1. **Distance dans le plan.**

Écris un programme qui lit un fichier contenant les coordonnées (x, y) de villes, puis qui calcule et écrit dans un autre fichier les distances (dans le plan) entre deux villes.

La formule pour la distance entre deux points (x_1, y_1) et (x_2, y_2) du plan est :

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Exemple. Voici un exemple de fichier en entrée :

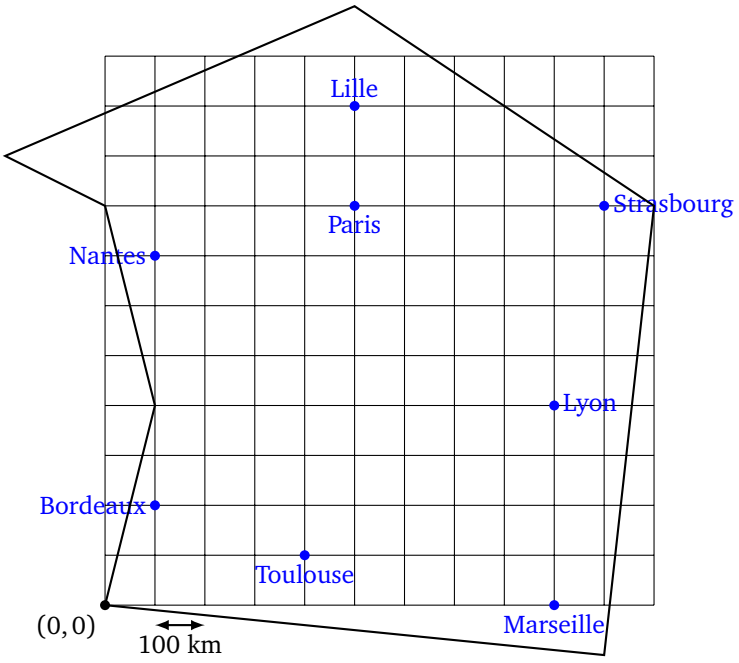
```
Paris 500 800
Lille 500 1000
Nantes 100 700
Marseille 900 0
```

Et voici le fichier de sortie produit par le programme :

```
                Paris      Lille     Nantes  Marseille
Paris               0        200        412        894
Lille             200          0        500       1077
Nantes            412        500          0       1063
Marseille         894       1077       1063          0
```

On lit sur ce fichier que la distance entre Lille et Marseille est de 1077 kilomètres.

Ci-dessous la carte de France qui a fourni des données (très approximatives) pour le fichier d'entrée. L'origine est en bas à gauche, chaque côté d'un carré représente 100 km. Par exemple, dans ce repère, Paris a pour coordonnées $(500, 800)$.



2. **Distance sur la sphère.**

   Sur la Terre, la distance entre deux villes correspond à un trajet suivant un *grand cercle* à la surface de la sphère et pas selon une ligne droite. C'est la distance que parcourt un avion pour relier deux villes.

   Écris un programme qui lit les latitudes et longitudes des villes, puis qui calcule et écrit dans un autre fichier les distances (à la surface de la Terre) entre deux villes.

   Exemple. Voici un exemple de fichier en entrée :

```
Paris 48.853 2.350
New-York 40.713 -74.006
Vancouver 49.250 -123.119
Lima -12.043 -77.0282
Hong-Kong 22.286 114.158
Addis-Abeba 9.0250 38.747
```

Et voici le fichier de sortie produit par le programme :

	Paris	New-York	Vancouver	Lima	Hong-Kong	Addis-Abeba
Paris	0	5837	7924	10253	9629	5573
New-York	5837	0	3905	5874	12959	11207
Vancouver	7924	3905	0	8168	10257	13298
Lima	10253	5874	8168	0	18371	13001
Hong-Kong	9629	12959	10257	18371	0	8135
Addis-Abeba	5573	11207	13298	13001	8135	0

Mise en œuvre et explications.

- Le fichier d'entrée contient la latitude (notée φ) et la longitude (notée λ) en degrés de chaque ville. Par exemple Paris a pour latitude $\varphi = 48.853$ degrés et pour longitude $\lambda = 2.350$ degrés.
- Pour les formules, il faudra utiliser les angles en radians. La formule de conversion de degrés vers radians est :
$$\text{angle en radians} = \frac{2\pi}{360} \times \text{angle en degrés}$$
- **Formule de la distance approchée.**
Il existe une formule simple qui donne une bonne estimation pour la distance la plus courte entre deux points d'une sphère de rayon R. Poser d'abord :
$$x = (\lambda_2 - \lambda_1) \cdot \cos\left(\frac{\varphi_1 + \varphi_2}{2}\right) \quad \text{et} \quad y = \varphi_2 - \varphi_1$$
La distance approchée est alors
$$\tilde{d} = R\sqrt{x^2 + y^2}$$
(φ_1, λ_1) et (φ_2, λ_2) sont les latitudes/longitudes de deux villes exprimées en radians.
- **Formule de la distance exacte.**
Les plus courageux peuvent utiliser la formule exacte pour calculer la distance. Poser d'abord :
$$a = \left(\sin\left(\frac{\varphi_1 + \varphi_2}{2}\right)\right)^2 + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \left(\sin\left(\frac{\lambda_2 - \lambda_1}{2}\right)\right)^2$$
La distance exacte est alors :
$$d = 2 \cdot R \cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$
où $\text{atan2}(y, x)$ est la fonction « arctangente » qui s'obtient par la commande `atan2(y,x)` du module `math`.
- Pour le rayon de la Terre on prendra $R = 6371$ km.