Chapitre 5

Les données structurées et leur traitement

Objectifs du chapitre

Thème : Les données structurées et leur traitement.
Identifier les données personnelles et les principaux formats de données, organiser des données en tables (descripteurs et valeurs), les traiter par programme (recherche, filtre, tri, calcul), retrouver des métadonnées et comprendre le stockage dans le nuage et son coût énergétique.
Organisation : programme officiel, cours, méthodes, ateliers Python, exercices corrigés, QCM et fiche de synthèse.

Programme officiel (BO)

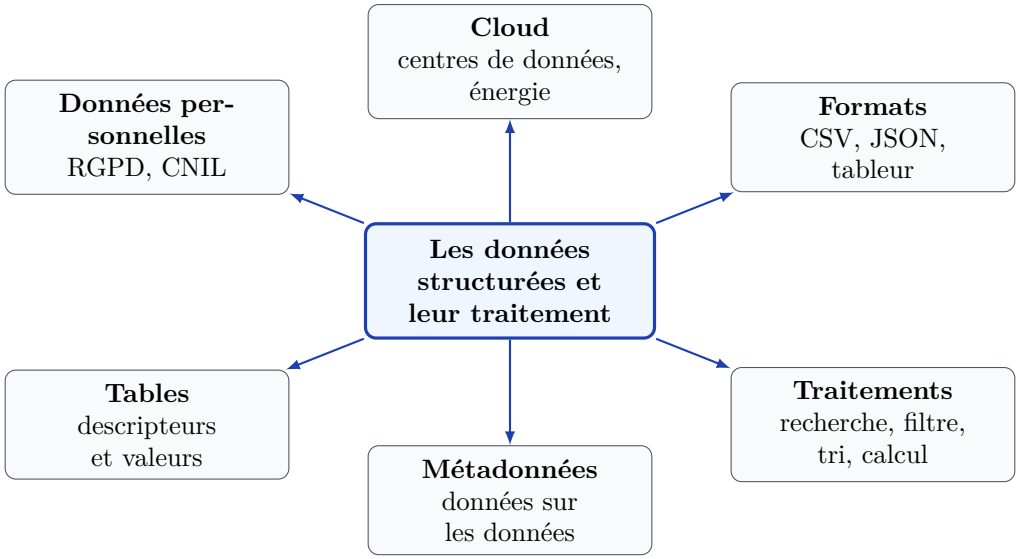
Thème « Les données structurées et leur traitement » – Contenus et capacités attendues

- **Données** : Définir une donnée personnelle. Identifier les principaux formats et représentations de données.
- **Données structurées** : Identifier les différents descripteurs d'un objet. Distinguer la valeur d'une donnée de son descripteur. Utiliser un site de données ouvertes, pour sélectionner et récupérer des données.
- **Traitement de données structurées** : Réaliser des opérations de recherche, filtre, tri ou calcul sur une ou plusieurs tables.
- **Métadonnées** : Retrouver les métadonnées d'un fichier personnel.
- **Données dans le nuage (cloud)** : Utiliser un support de stockage dans le nuage. Partager des fichiers, paramétrer des modes de synchronisation. Identifier les principales causes de la consommation énergétique des centres de données ainsi que leur ordre de grandeur.

Source : BO spécial n°1 du 22 janvier 2019, programme de Sciences numériques et technologie de Seconde générale et technologique.

5.1 Carte mentale du chapitre

Carte mentale



5.2 Les données et leurs formats

Une **donnée** est une valeur qui décrit un objet, une personne ou un événement : un nom, une date, une température, une position. Chaque jour, l'humanité produit des quantités gigantesques de données : messages, photos, achats, capteurs…

Définition – Donnée personnelle

Une **donnée personnelle** est une information qui permet d'identifier une personne physique, **directement** (nom, photo, empreinte) ou **indirectement** (numéro de téléphone, adresse IP, plaque d'immatriculation, ou un croisement de données : date de naissance + commune + lycée). En Europe, le **RGPD** (Règlement général sur la protection des données, 2018) encadre leur collecte : il faut informer la personne, ne collecter que le nécessaire, et elle a le droit d'accéder à ses données, de les rectifier et de les faire effacer. En France, la **CNIL** veille au respect de ces règles.

Définition – Formats de données

— **CSV** (*Comma-Separated Values*) : un fichier texte où chaque ligne est un enregistrement et où les valeurs sont séparées par une virgule ou un point-virgule. Lisible par un tableur, un éditeur de texte ou un programme.
— **JSON** : format texte structuré très utilisé sur le Web, proche des dictionnaires Python : `{"pays": "France", "or": 16}`.
— Formats de tableur (**ODS**, **XLSX**), bases de données, mais aussi formats non structurés : texte libre, image, son, vidéo.

5.3 Données structurées : descripteurs et valeurs

Définition – Table, descripteurs et valeurs

Des données sont **structurées** lorsqu'elles sont organisées en **table** : chaque ligne décrit un **objet** (un enregistrement), chaque colonne correspond à un **descripteur** (une caractéristique). À l'intersection, on lit la **valeur** du descripteur pour cet objet.

Voici le début d'un fichier CSV (Jeux olympiques de Paris 2024, dix premières nations au nombre de médailles d'or) :

```
pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8
```

Exemple – Lire la table

— Les **descripteurs** sont donnés par la première ligne : `pays`, `continent`, `or`, `argent`, `bronze`.
— Chaque autre ligne décrit un objet : un pays. Pour l'objet « France », la **valeur** du descripteur `argent` est `26`.
— Une même valeur (par exemple `12`) peut avoir des significations différentes selon son descripteur : c'est le descripteur qui donne son sens à la valeur.

Définition – Données ouvertes (open data)

Les **données ouvertes** sont des données publiques mises à disposition de tous, librement réutilisables. En France, le site **data.gouv.fr** en propose des dizaines de milliers : horaires de transports, qualité de l'air, résultats d'élections, prénoms donnés aux nouveau-nés… On les télécharge le plus souvent au format CSV, accompagnées de **métadonnées** (producteur, date de mise à jour, licence, description des colonnes).

5.4 Traiter des données structurées

Les quatre opérations de base sur une table sont :

Opération	Exemple sur la table des médailles
Rechercher	combien la France a-t-elle de médailles d'argent ?
Filtrer (sélectionner des lignes)	ne garder que les pays d'Europe
Trier	classer les pays par nombre total de médailles, du plus grand au plus petit
Calculer	nombre total de médailles de chaque pays, moyenne de médailles d'or en Europe

Animation interactive « Rechercher, filtrer et trier une table » : à manipuler sur https://utspe.com/snt/chapitre-5.

Méthode – Traiter une table en Python

On représente la table par une **liste de dictionnaires** : un dictionnaire par ligne, dont les clés sont les descripteurs.

```
table = [
    {"pays": "France", "continent": "Europe", "or": 16, "argent": 26, "bronze": 22},
    {"pays": "Japon", "continent": "Asie", "or": 20, "argent": 12, "bronze": 13},
    {"pays": "Italie", "continent": "Europe", "or": 12, "argent": 13, "bronze": 15},
]

# Filtrer : les pays d'Europe
for p in table:
    if p["continent"] == "Europe":
        print(p["pays"])

# Trier : par nombre de medailles d'or, du plus grand au plus petit
classement = sorted(table, key=lambda p: p["or"], reverse=True)
for p in classement:
    print(p["pays"], p["or"])
```

`sorted(table, key=...)` renvoie une nouvelle liste triée selon la valeur calculée par la fonction `key` ; `reverse=True` trie dans l'ordre décroissant.

Attention

Dans un fichier CSV, toutes les valeurs sont du **texte**. Il faut convertir les nombres avec `int()` ou `float()` avant de calculer ou de trier : sinon `"9"` est classé après `"40"`, car on compare caractère par caractère.

5.5 Les métadonnées

Définition – Métadonnées

Les **métadonnées** sont des **données sur les données** : elles décrivent un fichier sans en faire partie.

— pour tout fichier : nom, taille, type, date de création et de modification, auteur ;
— pour une photo (métadonnées **EXIF**) : date et heure, modèle d'appareil, réglages, et parfois les **coordonnées GPS** du lieu de la prise de vue ;
— pour un document : titre, auteur, logiciel utilisé, nombre de pages.

Certaines sont ajoutées automatiquement par l'appareil (date, GPS), d'autres renseignées par l'utilisateur (titre, mots-clés).

Méthode – Retrouver les métadonnées d'un fichier

Sous Windows : clic droit sur le fichier, **Propriétés**, onglet **Détails**. Sous macOS : **Lire les informations** (ou Aperçu, Outils, Afficher l'inspecteur pour une photo). Sur un téléphone : bouton « i » ou « Détails » dans la galerie.

Attention

Une photo publiée peut révéler, par ses métadonnées GPS, l'adresse de son domicile. Beaucoup de réseaux sociaux suppriment ces données, mais pas tous les sites ni les messageries : on peut désactiver la géolocalisation de l'appareil photo.

5.6 Les données dans le nuage

Définition – Stockage dans le nuage (cloud)

Stocker des fichiers **dans le nuage**, c'est les enregistrer sur les serveurs d'une entreprise, dans des **centres de données**, et y accéder par Internet depuis n'importe quel appareil.

— **Synchronisation** : une modification faite sur un appareil est recopiée sur les autres. On peut choisir les dossiers synchronisés, et garder ou non une copie locale.

— **Partage** : on donne accès à un fichier par un lien, en lecture seule ou en modification, éventuellement avec une date d'expiration.

— Avantages : sauvegarde, travail collaboratif. Inconvénients : dépendance à Internet et au fournisseur, confidentialité, consommation d'énergie.

Propriété – Consommation énergétique des centres de données

Un centre de données contient des milliers de serveurs qui fonctionnent jour et nuit. Les principales causes de consommation sont :

— l'**alimentation des serveurs** et du stockage (calcul, disques) ;

— le **refroidissement** : les serveurs chauffent et il faut climatiser les salles, ce qui peut représenter jusqu'à 40 % de la consommation ;

— la **redondance** : les données sont copiées dans plusieurs centres pour ne jamais être perdues, et les équipements de secours restent alimentés.

Ordre de grandeur : les centres de données consomment environ **400 TWh** d'électricité par an dans le monde (2024), soit environ **1,5 %** de l'électricité mondiale — à peu près la consommation électrique annuelle de la France. Cette part augmente rapidement avec l'intelligence artificielle et la vidéo en ligne.

5.7 Ateliers Python

Les ateliers se font sur ordinateur, dans un éditeur Python ou directement sur https://utspe.com/snt/chapitre-5. Pour chacun : recopier le programme de départ, suivre les étapes dans l'ordre et exécuter le programme après chaque modification.

Atelier 1 – Lire un fichier CSV split(), listes, dictionnaires

Objectif : Transformer le texte d'un fichier CSV en une liste de dictionnaires : un dictionnaire par pays, dont les clés sont les descripteurs.

1. Exécuter le programme : `donnees.split("\n")` découpe le texte en lignes ; la première ligne contient les descripteurs.
2. Compléter la boucle pour que chaque ligne devienne un dictionnaire `{descripteur: valeur}`, ajouté à `table`.
3. Convertir les valeurs `or`, `argent` et `bronze` en entiers avec `int()`.
4. Afficher `table[4]` : quel pays est-ce ? Afficher son nombre de médailles d'argent.

Programme de départ :

```
donnees = """pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
```

```
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8"""

lignes = donnees.split("\n")
descripteurs = lignes[0].split(";")
print("Descripteurs :", descripteurs)

table = []
for ligne in lignes[1:]:
    valeurs = ligne.split(";")
    print(valeurs)
    # a completer : creer le dictionnaire pays et l'ajouter a table

print(len(table), "pays")
```

Coup de pouce : Créer `pays = {}`, puis `for i in range(len(descripteurs)): pays[descripteurs[i]] = valeurs[i]`. Ensuite `pays["or"] = int(pays["or"])`, etc., et enfin `table.append(pays)`.

Corrigé rédigé

```
donnees = """pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8"""

lignes = donnees.split("\n")
descripteurs = lignes[0].split(";")
table = []
for ligne in lignes[1:]:
    valeurs = ligne.split(";")
    pays = {}
    for i in range(len(descripteurs)):
        pays[descripteurs[i]] = valeurs[i]
    for d in ["or", "argent", "bronze"]:
        pays[d] = int(pays[d])
    table.append(pays)

print(len(table), "pays")
print(table[4])
print(table[4]["pays"], ":", table[4]["argent"], "medailles d'argent")
```

Atelier 2 – Rechercher et filtrer

boucle for, conditions, fonctions

Objectif : Répondre à des questions en parcourant la table : rechercher une valeur, sélectionner les lignes qui vérifient une condition.

1. La table est déjà construite (programme de l'atelier 1). Exécuter : le programme affiche les pays d'Asie.
2. Écrire une fonction `recherche(nom)` qui renvoie le dictionnaire du pays `nom`, ou `None` s'il n'est pas dans la table. Tester avec `"Italie"` et `"Bresil"`.
3. Afficher les pays qui ont au moins 15 médailles d'or (condition `p["or"] >= 15`).
4. Afficher les pays européens qui ont plus de médailles d'argent que de médailles d'or.

Programme de départ :

```
donnees = """pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8"""

lignes = donnees.split("\n")
descripteurs = lignes[0].split(";")
table = []
for ligne in lignes[1:]:
    valeurs = ligne.split(";")
    pays = {}
    for i in range(len(descripteurs)):
        pays[descripteurs[i]] = valeurs[i]
    for d in ["or", "argent", "bronze"]:
        pays[d] = int(pays[d])
    table.append(pays)

for p in table:
    if p["continent"] == "Asie":
        print(p["pays"])
```

Coup de pouce : Dans `recherche`, parcourir la table et faire `return p` dès que `p["pays"] == nom` ; après la boucle, `return None`. Deux conditions se combinent avec `and`.

Corrigé rédigé

```
donnees = """pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8"""

lignes = donnees.split("\n")
descripteurs = lignes[0].split(";")
table = []
for ligne in lignes[1:]:
```

```
    valeurs = ligne.split(";")
    pays = {}
    for i in range(len(descripteurs)):
        pays[descripteurs[i]] = valeurs[i]
    for d in ["or", "argent", "bronze"]:
        pays[d] = int(pays[d])
    table.append(pays)

def recherche(nom):
    for p in table:
        if p["pays"] == nom:
            return p
    return None

print(recherche("Italie"))
print(recherche("Bresil"))

print("Au moins 15 medailles d'or :")
for p in table:
    if p["or"] >= 15:
        print("  ", p["pays"], p["or"])

print("Europe, plus d'argent que d'or :")
for p in table:
    if p["continent"] == "Europe" and p["argent"] > p["or"]:
        print("  ", p["pays"])
```

Atelier 3 – Calculer et trier

sorted(), key, lambda

Objectif : Ajouter un descripteur calculé (le total de médailles) puis classer les pays selon ce total.

1. Ajouter à chaque pays une clé `total` égale à or + argent + bronze.
2. Exécuter : le programme trie la table par ordre alphabétique des pays.
3. Modifier le tri pour classer les pays par total décroissant et afficher le classement numéroté : `1. Etats-Unis : 126`.
4. Le classement change-t-il par rapport au classement aux médailles d'or ? Quel pays gagne le plus de places ?

Programme de départ :

```
donnees = """pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8"""

lignes = donnees.split("\n")
descripteurs = lignes[0].split(";")
table = []
for ligne in lignes[1:]:
    valeurs = ligne.split(";")
    pays = {}
```

```
    for i in range(len(descripteurs)):
        pays[descripteurs[i]] = valeurs[i]
    for d in ["or", "argent", "bronze"]:
        pays[d] = int(pays[d])
    table.append(pays)

# a completer : ajouter la cle "total" a chaque pays

classement = sorted(table, key=lambda p: p["pays"])
for p in classement:
    print(p["pays"])
```

Coup de pouce : `for p in table: p["total"] = p["or"] + p["argent"] + p["bronze"]`. Puis `sorted(table, key=lambda p: p["total"], reverse=True)` et un compteur `rang` pour numéroter.

Corrigé rédigé

```
donnees = """pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8"""

lignes = donnees.split("\n")
descripteurs = lignes[0].split(";")
table = []
for ligne in lignes[1:]:
    valeurs = ligne.split(";")
    pays = {}
    for i in range(len(descripteurs)):
        pays[descripteurs[i]] = valeurs[i]
    for d in ["or", "argent", "bronze"]:
        pays[d] = int(pays[d])
    table.append(pays)

for p in table:
    p["total"] = p["or"] + p["argent"] + p["bronze"]

classement = sorted(table, key=lambda p: p["total"], reverse=True)
rang = 1
for p in classement:
    print(f"{rang}. {p['pays']} : {p['total']}")
    rang = rang + 1
```

Atelier 4 – Statistiques par continent

dictionnaires, accumulation

Objectif : Regrouper les lignes par continent pour calculer le nombre de médailles et la moyenne de médailles d'or par pays.

1. Le dictionnaire `somme` doit associer à chaque continent le total de ses médailles d'or. Exécuter le

programme et compléter la boucle.

2. Construire de même le dictionnaire `nombre` : nombre de pays de chaque continent présents dans la table.
3. Afficher, pour chaque continent, la moyenne de médailles d'or par pays, arrondie à 0,1.

Programme de départ :

```
donnees = """pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8"""

lignes = donnees.split("\n")
descripteurs = lignes[0].split(";")
table = []
for ligne in lignes[1:]:
    valeurs = ligne.split(";")
    pays = {}
    for i in range(len(descripteurs)):
        pays[descripteurs[i]] = valeurs[i]
    for d in ["or", "argent", "bronze"]:
        pays[d] = int(pays[d])
    table.append(pays)

somme = {}
for p in table:
    c = p["continent"]
    # a completer
print(somme)
```

Coup de pouce : Si `c` n'est pas encore une clé : `somme[c] = 0`. Puis `somme[c] = somme[c] + p["or"]`. Moyenne : `round(somme[c] / nombre[c], 1)`.

Corrigé rédigé

```
donnees = """pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8"""

lignes = donnees.split("\n")
descripteurs = lignes[0].split(";")
table = []
for ligne in lignes[1:]:
```

```
    valeurs = ligne.split(";")
    pays = {}
    for i in range(len(descripteurs)):
        pays[descripteurs[i]] = valeurs[i]
    for d in ["or", "argent", "bronze"]:
        pays[d] = int(pays[d])
    table.append(pays)

somme = {}
nombre = {}
for p in table:
    c = p["continent"]
    if c not in somme:
        somme[c] = 0
        nombre[c] = 0
    somme[c] = somme[c] + p["or"]
    nombre[c] = nombre[c] + 1

for c in somme:
    print(f"{c} : {somme[c]} medailles d'or, {nombre[c]} pays, moyenne {round(somme[c] /
    nombre[c], 1)}")
```

5.8 Exercices corrigés

Exercice

Exercice 1 – Donnée personnelle ou non ? Dire si chaque donnée est personnelle : 1. le numéro de téléphone d'un élève ; 2. la température à Lyon le 3 mars ; 3. l'adresse IP d'un ordinateur personnel ; 4. le nombre d'élèves d'un lycée ; 5. une photo de classe ; 6. la plaque d'immatriculation d'une voiture.

Corrigé rédigé

Personnelles : 1, 3, 5 et 6 (elles permettent d'identifier une personne directement ou indirectement). Non personnelles : 2 et 4.

Exercice

Exercice 2 – Descripteurs et valeurs. Une bibliothèque décrit ses livres ainsi : `titre;auteur;annee;pages;disponible` puis `Les Miserables;Victor Hugo;1862;1900;oui`.
1. Quels sont les descripteurs ? Combien y a-t-il d'objets décrits ?
2. Quelle est la valeur du descripteur `annee` pour ce livre ?
3. Proposer deux autres descripteurs utiles.

Corrigé rédigé

1. Descripteurs : titre, auteur, annee, pages, disponible. Un seul objet (un livre).
2. 1862.
3. Par exemple : genre, éditeur, langue, numéro ISBN, emplacement en rayon.

Exercice

Exercice 3 – Trier du texte. Un élève trie la liste `["9", "40", "12", "100"]` avec `sorted()` et obtient `['100', '12', '40', '9']`.
1. Expliquer ce résultat.
2. Corriger le programme pour trier les nombres dans l'ordre croissant.

Corrigé rédigé

1. Ce sont des chaînes : elles sont comparées caractère par caractère, comme dans un dictionnaire. « 1 » vient avant « 4 », qui vient avant « 9 ».
2. Il faut convertir en entiers avant de trier :

```
valeurs = ["9", "40", "12", "100"]
print(sorted(valeurs))
nombres = []
for v in valeurs:
    nombres.append(int(v))
print(sorted(nombres))
```

Exercice

Exercice 4 – Filtrer et calculer. À partir de la table des médailles du cours, écrire un programme qui affiche le nombre total de médailles remportées par les pays d'Europe présents dans la table, et le pourcentage de médailles d'or parmi elles.

Corrigé rédigé

Les cinq pays d'Europe totalisent 69 médailles d'or, 81 d'argent et 86 de bronze, soit **236 médailles**, dont environ **29,2 %** d'or.

```
donnees = """pays;continent;or;argent;bronze
Etats-Unis;Amerique;40;44;42
Chine;Asie;40;27;24
Japon;Asie;20;12;13
Australie;Oceanie;18;19;16
France;Europe;16;26;22
Pays-Bas;Europe;15;7;12
Royaume-Uni;Europe;14;22;29
Coree du Sud;Asie;13;9;10
Italie;Europe;12;13;15
Allemagne;Europe;12;13;8"""

lignes = donnees.split("\n")
descripteurs = lignes[0].split(";")
table = []
for ligne in lignes[1:]:
    valeurs = ligne.split(";")
    pays = {}
    for i in range(len(descripteurs)):
        pays[descripteurs[i]] = valeurs[i]
    for d in ["or", "argent", "bronze"]:
        pays[d] = int(pays[d])
    table.append(pays)

total = 0
```

```
ors = 0
for p in table:
    if p["continent"] == "Europe":
        total = total + p["or"] + p["argent"] + p["bronze"]
        ors = ors + p["or"]
print(total, "medailles dont", ors, "en or")
print(round(ors / total * 100, 1), "%")
```

Exercice

Exercice 5 – Métadonnées d'une photo. Les métadonnées EXIF d'une photo indiquent : `2024:07:14 22:47:10` ; `Apple iPhone 13` ; `f/1.6` ; `GPS 48.8584 N, 2.2945 E` ; titre « Feu d'artifice ».

1. Lesquelles ont été enregistrées automatiquement ? Lesquelles par l'utilisateur ?
2. Où et quand la photo a-t-elle probablement été prise ?
3. Quel risque y a-t-il à publier telle quelle une photo prise chez soi ?

Corrigé rédigé

1. Automatiques : date et heure, modèle d'appareil, ouverture, coordonnées GPS. Utilisateur : le titre.
2. Le 14 juillet 2024 vers 22 h 47, près de la tour Eiffel (48,858° N ; 2,294° E) : le feu d'artifice de la fête nationale.
3. Les coordonnées GPS révèlent l'adresse du domicile.

Exercice

Exercice 6 – Le coût énergétique du stockage. On estime grossièrement que stocker 1 Go dans le nuage pendant un an consomme environ 5 kWh (serveurs, refroidissement, copies de sécurité).

1. Un élève stocke 2 000 photos de 4 Mo chacune. Quelle quantité de données cela représente-t-il ? Quelle énergie par an ?
2. Pour un million d'élèves dans la même situation, comparer à la consommation annuelle d'un foyer français (environ 4 500 kWh).
3. Proposer deux gestes pour réduire cette consommation.

Corrigé rédigé

1. 2 000 × 4 Mo = 8 000 Mo = 8 Go, soit environ 40 kWh par an.
2. 40 000 000 kWh, soit la consommation d'environ 8 900 foyers.
3. Supprimer les doublons et les photos ratées, ne pas tout synchroniser automatiquement, compresser les vidéos, préférer un stockage local pour les archives.

```
go = 2000 * 4 / 1000
energie = go * 5
print(go, "Go ->", energie, "kWh par an")
total = energie * 1000000
print(total, "kWh, soit", round(total / 4500), "foyers")
```

5.9 QCM – Teste tes connaissances

Pour chaque question, une seule réponse est correcte.

Q1. Laquelle de ces données est une donnée personnelle ?

a) la population de Paris
b) l'adresse e-mail d'un élève
c) l'altitude du mont Blanc
d) le prix moyen d'un litre d'essence

Q2. Dans un fichier CSV, les valeurs d'une ligne sont…

a) séparées par un caractère comme ; ou ,
b) écrites dans des balises
c) codées en binaire
d) rangées dans des cellules colorées

Q3. Dans la table des médailles, « argent » est…

a) une valeur
b) un descripteur
c) un objet
d) une métadonnée

Q4. Sélectionner uniquement les lignes des pays d'Asie, c'est…

a) trier
b) filtrer
c) calculer
d) indexer

Q5. Que produit ce programme ?

```
p = {"pays": "Japon", "or": 20, "argent": 12}
print(p["or"] + p["argent"])
```

a) 2012
b) 32
c) or + argent
d) une erreur

Q6. Que produit ce programme ?

```
print(sorted(["12", "9", "100"]))
```

a) ['9', '12', '100']
b) ['100', '12', '9']
c) ['12', '100', '9']
d) une erreur

Q7. sorted(table, key=lambda p : p["or"], reverse=True) trie…

a) par ordre alphabétique
b) par médailles d'or, du plus petit au plus grand
c) par médailles d'or, du plus grand au plus petit
d) par continent

Q8. Les métadonnées d'une photo peuvent contenir…

a) uniquement sa taille
b) la position GPS de la prise de vue
c) le mot de passe du téléphone
d) les visages reconnus par la police

Q9. Le site français de données publiques ouvertes est…

a) data.gouv.fr
b) cnil.fr
c) legifrance.gouv.fr
d) service-public.fr

Q10. La synchronisation d'un dossier dans le nuage signifie que…

a) le dossier est supprimé de l'ordinateur
b) une modification sur un appareil est recopiée sur les autres
c) le dossier est chiffré
d) le dossier est public

Q11. Une cause importante de consommation des centres de données est…

a) l'éclairage des salles
b) le refroidissement des serveurs
c) l'impression des données
d) le chauffage des bureaux

Q12. Ordre de grandeur de la part de l'électricité mondiale consommée par les centres de données :

a) 0,01 %
b) 1,5 %
c) 25 %
d) 60 %

Corrigé rédigé

Réponses du QCM.
Q1. b) Une adresse e-mail permet d'identifier une personne.
Q2. a) CSV signifie « valeurs séparées par des virgules ».
Q3. b) « argent » est le nom d'une colonne : un descripteur. 26 est une valeur.
Q4. b) Filtrer, c'est garder les lignes qui vérifient une condition.
Q5. b) Les valeurs sont des entiers : $20 + 12 = 32$.
Q6. b) Des chaînes sont comparées caractère par caractère : « 1 » avant « 9 ».
Q7. c) key indique la valeur utilisée pour trier, reverse=True donne l'ordre décroissant.
Q8. b) Les métadonnées EXIF peuvent contenir date, appareil, réglages et coordonnées GPS.
Q9. a) data.gouv.fr publie les données ouvertes de l'administration.
Q10. b) Les appareils synchronisés gardent les mêmes versions des fichiers.
Q11. b) Les serveurs chauffent : la climatisation peut représenter jusqu'à 40 % de la consommation.
Q12. b) Environ 400 TWh par an, soit 1 à 2 % de l'électricité mondiale, et en forte hausse.

5.10 Fiche de synthèse

Fiche de synthèse

— **Données personnelles :** Elles identifient une personne directement ou indirectement. Protégées par le RGPD ; la CNIL veille en France.

— **Formats :** CSV (texte, valeurs séparées par ; ou ,), JSON, tableur. Les données ouvertes (data.gouv.fr) sont souvent en CSV avec des métadonnées.

— **Tables :** Une ligne = un objet, une colonne = un descripteur, une case = une valeur. En Python : liste de dictionnaires.

— **Traitements :** Rechercher, filtrer (condition), trier (`sorted` avec `key`), calculer. Convertir le texte en nombres avant de calculer ou trier.

— **Métadonnées :** Données sur les données : taille, dates, auteur ; EXIF des photos (appareil, date, GPS).

— **Cloud :** Stockage sur des serveurs distants, synchronisation, partage. Centres de données : environ 1,5 % de l'électricité mondiale (serveurs, refroidissement, redondance).