Traitement des données en table sur la base de fichiers CSV

I. Ouverture du fichier

1. Créer un fichier **Python table_csv.py**.

2. Avec **Python**, importer le fichier **pokemon_stats_1.csv** grâce au module **csv**.

```
import csv

fichier = open('pokemon_stats_1.csv', 'r', newline='', encoding='utf-8')
objets_pokemon_1 = csv.DictReader(fichier, delimiter=';')

objets_pokemon_1 = [dict(ligne) for ligne in objets_pokemon_1]

fichier.close()
```

3. A quoi correspond cette collection ? Identifier les descripteurs.

4. Combien d'objets la collection contient-elle ? Comment avez-vous obtenu la valeur ?

II. Recherches dans une table

1. Extraire le nom et l'attaque du Pokemon 342.

2. Retrouver l'indice de l'objet de nom **Toxicroak** grâce à la boucle ci-dessous (en une seule ligne **Python**).

```
indices_objets = [i for i in range(len(objets_pokemon_1)) if objets_pokemon_1[i]['name'] == 'Toxicroak']
```

3. Comment la boucle fonctionne-t-elle ?

4. Vérifier que l'indice trouvé correspond bien à **Toxicroak**.

5. Retrouver le Pokemon **Lairon**. Le vérifier . Comment avez-vous fait ?

6. Trouver tous les Pokemons classés **Dragon Pokémon**. Les vérifier. Comment avez-vous fait ?

7. Ecrire une boucle par compréhension qui fournit la liste des objets, nommée **liste_objets**, correspondant à une liste d'indices fournie sous le nom **indices_objets**. La tester sur **[12, 74, 123, 253, 388]**.

8. Trouver tous les Pokémons qui ont une attaque supérieure ou égale à **70**. Combien sont-ils ?

9. Existe-t-il des Pokémons dont l'attaque et la défense sont supérieurs à **100** ? Si oui, combien ?

III. Premiers calculs dans une table

1. A l'aide d'une boucle, montrer que la valeur moyenne de l'attaque de tous les Pokémon est de **77.24**.

2. Quelle est la valeur moyenne de l'attaque de tous les Pokémon ?

3. Ecrire en **Python** l'algorithme suivant qui recherche l'objet d'attaque maximale. Noter le code et les valeurs obtenues. Donner son nom.

indice_max initialisé à 0
valeur_max initialisé à l'attaque de l'objet d'indice 0

pour i allant de 1 à l'indice du dernier objet :
  si l'attaque de l'objet i est strictement supérieure à valeur_max :
    indice_max prend la valeur i
    valeur_max prend la valeur de l'attaque de l'objet i

afficher indice_max et valeur_max

4. Rechercher le Pokémon d'attaque minimale.

5. Trouver les mêmes informations pour le descripteur **défense**.

6. Quel est le Pokémon qui a la meilleure moyenne attaque-défense ? Donner sa valeur. Comment avez-vous fait ?

IV. Tris dans une table

sorted() est la fonction **Python** qui permet de trier des données. Son algorithme de tri est un des plus efficaces.

1. Tester les deux lignes de code ci-dessous.

```
def fonction_cle_de_tri(objet):
    return objet['name']

liste_triee = sorted(objets_pokemon_1, key=fonction_cle_de_tri)
print(liste_triee)
```

2. Ecrire une boucle permettant d'afficher les **10** premiers objets et ainsi vérifier qu'ils ont bien été triés dans l'ordre croissant.

La fonction **fonction_cle_de_tri(objet)** permet de préciser des critères pour le tri et notamment sur le descripteur à utiliser pour l'effectuer. La fonction retourne simplement pour chaque objet, la valeur du descripteur choisi pour le tri. On peut ajouter à **sorted()** l'option **reverse=False** ou **True** pour trier par ordre croissant ou décroissant.

3. Tester cette option.

4. Trier les objets par ordre décroissant de leur attaque. Qu'avez-vous fait ?

5. Quels sont les **3** meilleurs Pokémons défenseurs ? Parmi eux, y en a-t-il des bons en attaque ? Justifier en expliquant votre démarche.

6. Dans la réponse précédente, quels sont les éléments qui relèvent d'une donnée et quels sont les éléments qui relèvent d'une information ? Justifier en expliquant votre démarche.

7. Les Pokémons qui ont la meilleure moyenne attaque-défense font-ils parti des meilleurs attaquants ou des meilleurs défenseurs ? Justifier en expliquant votre démarche.

V. Fusion de tables ayant les mêmes descripteurs, recherche des doublons

Lorsque l'on veut fusionner deux tables, elles auront peut-être des objets en communs. On parle de doublons, qu'il faudra donc supprimer.

1. Dans un fichier **fusion_csv.py**, importer les deux tables **pokemon_stats_2_a.csv** et **pokemon_stats_2_b.csv** dans deux variables **objets_pokemon_2_a** et **objets_pokemon_2_b**.

2. Vérifier que les deux tables ont les mêmes descripteurs. Les donner.

Lorsque deux tables ont les mêmes descripteurs. On peut les fusionner par concaténation..

5. Comment concaténer en **Python** les deux listes d'objets ? Le résultat sera stocké dans la liste **objets_pokemon_2**. Le faire.

6. Vérifier que les longueurs des listes correspondent. Expliquer la démarche.

Une fois fusionnée, il peut y avoir des doublons dans la liste. Pour le vérifier on peut prendre les objets les uns après les autres et les comparer à tous les objets suivants. S'il y a égalité, il y a doublon.

7. Programmer en **Python** l'algorithme ci-dessous et l'appliquer pour vérifier qu'il y a bien des doublons dans la liste.

doublons initialisé à faux
i initialisé à 0
tant que doublons n'est pas vrai et i est inférieur à l'indice du dernier objet:
pour j allant de i+1 au dernier objet :
si l'objet j est égal à l'objet i :
doublons prend la valeur vrai
incrémenter i

afficher doublons

8. Quel est le nom du premier Pokémon en double ? Comment avez-vous fait pour l'obtenir ?

Si des doublons ont été identifiés, il faut nettoyer la liste. La procédure est encore plus simple que celle pour détecter des doublons. On peut même l'appliquer systématiquement sans tester la présence des doublons. Ainsi si la liste n'en comporte pas, il n'y aura aucune modification, sinon, les doublons seront pris en compte. L'algorithme est le suivant.

1. On crée une liste vide, par exemple **objets_pokemon_2_sans_doublons**.
2. On prend les objets de la liste avec doublons **objets_pokemon_2** les uns après les autres.
3. Si l'objet n'est pas dans la nouvelle liste **objets_pokemon_2_sans_doublons**, l'ajouter.

9. Coder cet algorithme et vérifier que vous obtenez une liste de **507** objets.

10. Exporter la table obtenue dans le fichier **pokemon_stats_2.csv** et vérifier que c'est le cas.

VI. Fusion de tables ayant des descripteurs différents

Lorsque que l'on veut fusionner deux tables, en plus d'avoir des doublons, elles peuvent avoir des descripteurs en commun et d'autres non.

1. Ouvrir les deux tables **pokemon_stats_1.csv** et **pokemon_stats_2.csv** dans un tableur. Attention à bien sélectionner le point virgule **uniquement** comme séparateur.

2. Ont-elles des descripteurs en commun ou non ? Détailler la réponse.

3. Quel est le descripteur permettant d'identifier de manière unique un objet dans ces deux collections qui sont en relation ?

Ce descripteur est appelé clé primaire dans une base de donnée relationnelle.

4. Que doit-on faire pour fusionner les deux tables ? Le faire dans le tableur. Quel est le résultat obtenu ?

5. Exporter la table avec comme nom de fichier **pokemon_stats_3.csv**.

VII. Test de cohérence d’une table, domaines de valeurs

Pour obtenir des collections utilisables et faciles à entretenir, il faut fixer un certain nombre de règles qui doivent être respectées par les programmes manipulant les collections et les utilisateurs. On parle de contraintes. Une des premières contraintes est que les valeurs d’un descripteur donné soient conditionnées : caractère, type, bornes, ... On parle de domaine de valeurs. Les autres contraintes évidentes sont qu’une collection ne possède pas de doublons ou que les clés primaires soient uniques.

Par exemple le descripteur **nom** de la collection **Pokémons** doit obligatoirement être une chaîne de caractères alphabétiques, constituée d’un seul mot commençant par une majuscule, les autres caractères étant des minuscules. Deux objets différents ne peuvent pas avoir le même nom. Les descripteurs **attaque** et **défense** sont des entiers non signés positifs. Deux objets différents peuvent avoir la même attaque et/ou la même défense.

Lorsqu’une collection respecte toutes les contraintes fixées, on dit qu’elle est cohérente. Le test de cohérence est une opération nécessaire qui doit être réalisée avec attention par tout système de gestion de base de données.

En fonction des résultats du test de cohérence, la décision doit être prise de supprimer l’objet, ou d’en modifier les valeurs si suffisamment d’informations sont disponibles pour le faire précisément. Dans ce cas, les algorithmes seront complexes à mettre en place pour automatiser les traitements.

1. En examinant les tables **pokemon_stats_1.csv** et **pokemon_stats_2.csv** ou la table **pokemon_stats_3.csv** donner les domaines de valeurs des descripteurs **hauteur**, **type** et **classification**.

2. Importer la table **pokemon_stats_4.csv** sous **Python** dans un fichier **test_coherence.py**.

3. Cette collection n’est pas cohérente. Effectuer en **Python** quelques tests pour le vérifier.

4. Sauriez-vous rendre cohérente la collection ? Au moins sur certains descripteurs ? Expliquer.

VIII. Coloriage