NSI - Première - Données en table - rappels

Partie I : généralités sur le fichier CSV et les tables en Python

Les pays

On a téléchargé un fichier CSV contenant une liste de pays avec différentes informations

Voici un extrait des premières lignes de ce fichier :

```
ISO;Country;Capital;Area(in sq km);Population;Continent;CurrencyCode;CurrencyName
AD;Andorra;Andorra la Vella;468;84000;EU;EUR;Euro
AE;United Arab Emirates;Abu Dhabi;82880;4975593;AS;AED;Dirham
AF;Afghanistan;Kabul;647500;29121286;AS;AFN;Afghani
AG;Antigua and Barbuda;St. John's;443;86754;NA;XCD;Dollar
AI;Anguilla;The Valley;102;13254;NA;XCD;Dollar
```

1. Que signifie l'acronyme CSV ?
2. Quels sont les différents champs de ce fichier ? Quel caractère les sépare ?
3. Écrire la ligne correspondant à la France. La population française s'élève à 64.768.389 habitants et sa surface est de 547.030 km^2
4. En Python, un dictionnaire est une série de paires clé, valeurs de la forme `d = {'a': 1, 'b': 2, 'c': 3}`
   a. Que produit l'instruction `d['a']` ?
   b. Et `d['e']` ?
5. En Python une *table* est généralement représenté par une liste de dictionnaires ayant les mêmes clés :

   ```
   table = [
       {'a': 2, 'b': 3},
       {'a': 4, 'b': 4},
       {'a': 5, 'b': 4}
   ]
   ```

   a. Comment accéder à la longueur de la table ?
   b. Comment accéder à son premier élément ?
   c. Écrire une instruction qui renvoie une table ne contenant que les enregistrements pour lequel 'b' vaut 4.

Partie II : utiliser un fichier CSV avec un script Python

Voici un script Python permettant de travailler sur ce fichier :

Données en tables : rappels

```python
import csv

def charger(fichier: str, delimiter=';') -> list:
    '''charge un fichier csv et renvoie une table'''
    with open(fichier) as fichiercsv:
        lecteur_csv = csv.DictReader(fichiercsv, delimiter=delimiter)
        return [dict(enregistrement) for enregistrement in lecteur_csv]

def caster_pays(table: list):
```

```
    '''renvoie une copie de la table avec les bons types'''
    return [{
        'ISO': pays['ISO'],
        'Country': pays['Country'],
        'Capital': pays['Capital'],
        'Area': float(pays['Area(in sq km)']),
        'Population': int(pays['Population']),
        'CurrencyCode': pays['CurrencyCode'],
        'CurrencyName': pays['CurrencyName'],
    } for pays in table]

def recup_pays(table: list, iso: str) -> dict:
    '''renvoie l'enregistrement d'un pays de la table depuis son code iso'''
    return [enregistrement for enregistrement in table
            if enregistrement['ISO'] == iso][0]

def project(table: list, liste_champs: list):
    '''projette la table et ne garde que les colonnes de liste_champs'''
    return [{champ: enregistrement[champ] for champ in liste_champs}
            for enregistrement in table]

def trier_critere(table: list, champ: str, reverse=False):
    '''renvoie une copie triée de la table selon un critère'''
    return sorted(table, key=lambda pays: pays[champ], reverse=reverse)
```

On exécute ce script en mode intéractif dans le dossier contenant le fichier `countries.csv`.

1. Décrire la valeur de `table_pays` après l'instruction `table_pays = charger("countries.csv")`.
2. Quel est l'intérêt de la fonction `caster_pays` ?

On exécute `table_pays = caster_pays(table_pays)`

3. Décrire le premier enregistrement de cette table.
4. Décrire le paramètre de sortie de `recup_pays('FR')`.
5. On exécute `recup_pays('ZZ')`, sachant que `ZZ` n'est pas un code ISO présent dans la table, que produit cette instruction ?
6. Écrire une fonction `recup_capitale` qui prend deux paramètres, une table et un code ISO et renvoie la capitale d'un Pays.
7. Que produit l'instruction `project(table_pays, ['ISO', 'Country', 'Capital'])` ?

La fonction `sorted` prend au moins un argument et peut en prendre trois. Elle renvoie une copie triée de son premier argument.

- le premier est une `list` Python,
- le second `key` est le critère du tri,
- le troisième `reverse` est une booléen. Si `reverse=False`, la liste est triée par ordre croissant, sinon elle est triée par ordre décroissant.

Examinez bien les exemples suivants :

```
>>> sorted([3, 1, 2])
[1, 2, 3]
>>> sorted([3, 1, 2], reverse=True)
[3, 2, 1]
>>> sorted([{'a': 1, 'b': 20}, {'a': 2, 'b': 10}], key=lambda l: l['a'])
[{'a': 1, 'b': 20}, {'a': 2, 'b': 10}]
>>> sorted([{'a': 1, 'b': 20}, {'a': 2, 'b': 10}], key=lambda l: l['b'])
[{'a': 2, 'b': 10}, {'a': 1, 'b': 20}]
```

8. Étudier le code de `trier_critere`.
    a. Que produit `trier_critere(table_pays, 'Area')` ?
    b. Proposer une instruction renvoyant les pays triés par population décroissante.