TP N° 5

Gestion des fichiers, Les exceptions et la gestion des erreurs

Exercice 1 :

1. Écrivez un programme qui ouvre un fichier texte pour afficher son contenu. Le programme modifie les lignes affichées afin qu'elles commencent par une majuscule (vous pouvez utiliser la méthode `capitalize()`). Le chemin du fichier peut être demandé à l'utilisateur ou passé en paramètre du script.

2. Modifiez le programme précédent afin que le contenu modifié du fichier ne soit plus affiché mais sauvé dans un autre fichier. Le chemin du fichier de destination peut être demandé à l'utilisateur ou passé en paramètre du script.

Exercice 2 :

Nous voulons charger les données d'un QCM à partir d'un document **JSON**. Pour charger un document **JSON**, vous devrez utiliser le module `json`. Ce dernier fournit notamment la méthode `load` qui permet de charger un document **JSON** sous la forme d'un dictionnaire Python.

Un fichier JSON pour un QCM :

```
[
    {
        "libelle": "Comment s'appelle le chien de Tintin ?",
        "choix": ["Félix", "Gustave", "Milou"],
        "reponse": 2
    },
    {
        "libelle": "Combien y a-t-il de points cardinaux ?",
        "choix": ["2", "4", "360"],
        "reponse": 1
    }
]
```

Exercice 3 :

Le fichier CSV (Comma-Separated Values) est un format texte très simple utilisé pour stocker des tables de données. Le module csv offre des méthodes pour lire et écrire des fichiers CSV. Pour lire le fichier fish.csv et afficher les données qu'il contient, vous pouvez utiliser le code suivant :

```
import csv

with open("fish.csv") as f:

    lecteur = csv.reader(f)

    for ligne in lecteur:

       print(ligne)
```

1. Gérer les exceptions lors de l'ouverture du fichier fish.csv. Si le fichier n'est pas trouvé (FileNotFoundError), afficher un message spécifique. En outre, gérer toute autre exception inattendue, puis afficher un message générique indiquant qu'une erreur s'est produite.

2. Adapter cet affichage et rendre les données plus compréhensibles, vous pouvez utiliser des libellés appropriés pour chaque colonne et formater l'affichage des valeurs. Voici un exemple d'affichage :

```
Age   Température Poids (g)    Longueur (cm)
-----------------------------------------
14    25          100          620
28    25          40           1315
41    25          78           2120
55    25          99           2600
69    25          163          3110
83    25          243          3535
97    25          202          3935
111   25          241          4465
125   25          290          4530
```

Exercice 4 :

Écrivez un programme qui demande à l'utilisateur de saisir un nombre. Si l'utilisateur ne saisit pas un nombre, le programme doit indiquer à l'utilisateur qu'il a fait une erreur et lui redemander de saisir un nombre. Puis le programme affiche le nombre saisi.

Exercice 5 :

On définit un QCM comme une liste de questions. La fonction `poser_question(question)` prend en paramètre un dictionnaire question représentant une question du QCM. Ce dictionnaire doit contenir les clés suivantes :

- libelle, qui donne le libellé de la question.
- choix, qui est un tableau des différents choix de réponse.
- reponse, qui est un nombre correspondant à l'index du choix qui correspond à la bonne réponse.

La fonction `poser_question(question)` pour poser des questions à l'utilisateur dans le cadre d'un QCM. Chaque fois que vous appelez cette fonction avec un dictionnaire représentant une question, elle affiche la question avec les options de réponse, récupère la réponse de l'utilisateur, valide cette réponse et retourne un booléen indiquant si la réponse est correcte ou non.

Utilisez le mécanisme des exceptions pour :

- Vérifier que l'utilisateur saisit bien un nombre pour indiquer son choix. Si ce n'est pas le cas, il faut afficher à nouveau les choix de réponse et lui redemander son choix.
- Vérifier que l'utilisateur saisit bien un numéro qui correspond à un choix. Sinon, il faut lui indiquer que ce choix n'existe pas et lui redemander son choix.