TP de Langages de script n° 2 : Scripts, fichiers et dictionnaires.

Exercice 1 : Premier module.

1. Créez un module `tp2` dans un fichier `tp2.py`, en suivant la structure donnée en cours. Ne pas oublier d'inclure une chaîne de documentation décrivant le module.
2. Définissez une variable `auteur` contenant votre nom, ainsi qu'une fonction `copyright()` affichant à l'écran que vous êtes l'auteur de ce programme et que vous poursuivrez en justice quiconque le copiera sans autorisation. N'oubliez pas d'inclure une chaîne de documentation décrivant la fonction.
3. Faites en sorte que, lorsque le fichier est exécuté en tapant son nom dans le `terminal`, il affiche le nom de son auteur et le message de copyright.
4. Importez le fichier dans l'interpréteur Python. Que se passe-t-il ? Modifiez votre script pour que les commandes de la question précédente ne s'exécutent que si le programme est appelé depuis le *terminal*, et pas quand il est importé dans l'interpréteur.
5. Depuis un interpréteur, consultez la documentation sur votre module grâce à la commande `help` et son environnement local grâce à la commande `dir()`. Importez la variable `auteur` du module dans l'environnement global grâce à la commande `from ... import ...`

Exercice 2 : Traduction automatique

On souhaite écrire un programme de traduction automatique simpliste. Le lexique de traduction est stocké dans un fichier dont chaque ligne est de la forme `mot:traduction` (on suppose que `traduction` est un mot unique sans espace).

1. En utilisant la méthode `split` du type `string`, trouvez la commande permettant de transformer une chaîne de caractère contenant `:` en un tableau contenant la première et la seconde partie de la chaîne séparée par ces deux points.
2. Écrivez une fonction `read_lexic` prenant comme argument le nom d'un fichier et renvoyant un dictionnaire PYTHON dont chaque couple (clé, valeur) correspond à un mot du lexique et à sa traduction.
3. Écrivez une fonction `translate_word` prenant un mot et un dictionnaire et renvoyant sa traduction selon ce lexique.
4. Écrivez une fonction `add_word` prenant deux mots et un dictionnaire et ajoutant le mot et sa traduction au dictionnaire.
5. Écrivez une fonction `write_lexic` réciproque de la fonction `read_lexic`. Améliorez cette fonction en faisant en sorte que les entrées soient classées par ordre alphabétique.
6. Écrivez une fonction principale qui demande à l'utilisateur un mot et lui fournit la traduction si elle existe dans le lexique. Si elle n'existe pas, il demande à l'utilisateur de donner la traduction et la sauvegarde dans le dictionnaire.
7. Écrivez un fichier contenant un lexique de base et utilisez le pour tester votre script.

Exercice 3 : Convertisseur

Le but de cet exercice est de créer un script `convertisseur` qui convertit les températures et les longueurs entre le système anglo-saxon et le système français. On pourra l'utiliser dans un `terminal` de la façon suivante :

```
./convertisseur [-tl] [-s <systeme initial>] [nombres]

-t = temperature
-l = longueur
-s = introduit le systeme initial :
     A = anglo-saxon
     F = francais
```

1. Écrire les fonctions `celsius`, `fahrenheit`, `metre` et `yard` qui permettent de faire les différentes conversions. Les tester.
2. Écrire une fonction `help` qui écrit dans le `terminal` un message aidant à la bonne utilisation du script. La tester.
3. En vous aidant du module `sys` et de la méthode `argv`, écrire la fonction principale du script qui récupère les différents arguments, puis selon les cas, appelle une fonction de conversion sur l'argument adéquat ou appelle la fonction d'aide.

Exercice 4 : Géographie

Le but de cet exercice est de créer un script `capitales` qui permet de tester ses connaissances sur les capitales du monde. On pourra l'utiliser dans un `terminal` de la façon suivante :

```
./capitales -v ville
    -> le script demande a l'utilisateur le nom du pays dont la capitale
    est ville, puis verifie la reponse
./capitales -n
    -> le script  donne un nom de  capitale et le joueur
    doit trouver le noms de pays associé. Le joueur peut
    decider de continuer ou d'arreter, a la fin il obtient une
    statistique sur le nombre de bonne reponse.
```

1. Écrire une fonction qui convertit un fichier contenant des pays et leur capitale en un dictionnaire. Tester cette fonction sur le fichier `capitales.txt` que vous trouverez sur Didel.
2. Écrire une fonction pour chacune des options `-v, -n` (on s'appuiera pour la dernière sur le module `random`).
3. Écrire la partie principale du code qui selon les options appelle la bonne fonction.