ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

TRAVAUX PRATIQUES 4

À vous de créer entièrement votre document .ipynb de réponse à ce TD, inspirez-vous des précédents TP... *Il pourra être poursuivi ou même fini à la maison...* ***Les exercices 4, 6, 7, 8, 9 et 10 sont facultatifs.***
L'objectif de ce TP est de manipuler les dictionnaires.
Vous pouvez faire ce TP soit en créant quelques fonctions et en utilisant l'interpréteur, soit, pour les plus avancés, en créant un programme complet (avec un menu par exemple) répondant à toutes ces questions.

1 Méthodes pour dictionnaires

Voici un petit mémo sur les méthodes liées au dictionnaire.
- `flag = dico.has_key(clé)` renvoie `True` si le dictionnaire `dico` contient la clé `clé`; préférable avant d'appeler une clé qui n'existerait pas.
- `dico.get(clé)` renvoie la valeur associée à la clé, `None` si la clé n'existe pas.
- `dico.get(clé, exp)` renvoie l'expression (nombre, chaîne...) si la valeur n'existe pas.
- `dico[clé] = valeur` ajoute une valeur associée à une nouvelle clé à un dictionnaire ou modifie la valeur associée à cette clé.
- `dico.pop(clé)` renvoie et supprime la valeur définie par une clé (ainsi que la clé).
- `dico.popitem()` renvoie une paire (clé-valeur) et la supprime du dictionnaire.
- `dic.clear()` supprime tous les éléments du dictionnaire `dic`.
- `dico.update(dic)` ajoute les occurrences du dictionnaire `dic` au dictionnaire `dico`.
- `dico = dic.copy()` copie un dictionnaire dans un autre; les modifications de l'un n'affectent pas l'autre.
- `dico.keys()` crée une liste des clés du dictionnaire `dico`.
- `dico.values()` crée une liste des valeurs du dictionnaire `dico`.
- `dico.items()` crée une liste de tuples (clé, valeur) : [(1,'ek'),(2,'do'),(3,'tin')...]

Exercice 1. Complétez votre connaissance des dictionnaires et entraînez-vous ici en entrant les commandes suivantes :

```
tel = {'jack': 4098, 'joe': 4139}
tel['john'] = 4127
tel
tel['jack']
del tel['joe']
tel['jimmy'] = 4127
tel
tel.keys()
tel.values()
tel.items()
```

Remarque importante : les questions sont dans une large mesure liées entre elles... **On donne les listes suivantes :**

— Liste des mois en français : (# mf pour Mois en Français)
```
mf = ["Janvier", "Février", "Mars", "Avril", "Mai", "Juin", "Juillet",
      "Août", "Septembre", "Octobre", "Novembre", "Décembre"]
```
— Liste des mois en anglais :
```
ma = ["January", "February", "March", "April", "May", "June",
      "July", "August", "September", "October", "November", "December"]
```
— Liste du nombre de jours de chaque mois dans une année non bissextile :
```
nbjN = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```
— Liste des jours de la semaine en français :
```
semf = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"]
```

Définition d'une `date`

Nous considérerons par la suite qu'une `date` est donnée par un quadruplet `(langue, d, m, a)` :

— `langue` est une chaîne de caractères des 3 premières lettres du mot, par exemple : "fra" pour français, "ang" pour anglais, "all" pour allemand, etc.
— `d` : le numéro du jour.
— `m` : le numéro du mois.
— `a` : l'année.

2.1 Dictionnaire de traduction

Exercice 2. –

1. Créez un dictionnaire `DictMois` ayant pour clé une chaîne de caractères de la forme "1er mois", "2e mois", "3e mois", etc., et pour chacune de ces clés une valeur de type liste composée des mois dans les 2 langues (français et anglais).
2. Après avoir créé une 3e liste de mois dans une autre langue (par exemple : espagnol, allemand, chinois, italien, japonais...), donnez l'instruction permettant d'ajouter celle-ci aux valeurs précédentes. Ainsi, par exemple, en utilisant l'espagnol, le début de `DictMois` deviendra : { "1er mois" : ["Janvier", "January", "Enero"], "2e mois" : ["Février", ...]}
3. Créez une fonction `traducm()` qui prend une `date` dans une langue i et la traduit dans une langue j.

Exercice 3. – Réalisez le même travail pour créer `DictJour`, un dictionnaire permettant la traduction des jours de la semaine.

Exercice 4. (Facultatif)

1. Créez avec un traitement de texte un fichier .csv contenant les mois et les jours dans une quatrième langue. Ce fichier aura l'aspect suivant :

   ```
   mois1, mois2, ..., mois12 ; Jour1, jour2, ...
   ```

   mois1 étant la traduction de janvier dans cette quatrième langue... Remarquez les places des virgules et points-virgules.
2. Créez une fonction `ajoutlangue()` qui lira dans un fichier les mois et les jours d'une autre langue afin de les ajouter aux deux dictionnaires. Vous pouvez utiliser les deux fichiers joints : `TableauLangueMois.csv` et `TableauLangueJour` (qui demanderont quelques manipulations...).

2.2 Année bissextile

Pour savoir si une année est bissextile, appliquez le protocole suivant :
— Si l'année est divisible par 4 et non divisible par 100, c'est une année bissextile (elle a alors 366 jours).
— Si l'année est divisible par 400, c'est une année bissextile (elle a alors 366 jours).
— Sinon, l'année n'est pas bissextile (elle a 365 jours).

Exercice 5. – En vous inspirant du protocole de décision ci-dessus, écrivez une fonction `bissextile()` ayant en argument une année et qui renvoie un booléen (`True` ou `False`) selon que l'année donnée est bissextile ou non.

2.3 Nombre de jours séparant 2 dates (pour les plus rapides)

Dans cette sous-partie, nous considérons que `langue` correspond au français.

Exercice 6. – À partir de `nbjN`, créez la liste `nbjB` du nombre de jours de chaque mois dans une année bissextile.

Exercice 7. – Nous aurons besoin par la suite de `nbjNcumul` (c'est-à-dire une liste "d'effectifs cumulés" de la liste `nbjN`). Elle devrait avoir la forme suivante : `nbjNcumul = [0, 31, 59, 90...]` avec `len(nbjN) = len(nbjNcumul)`.

1. Construisez `nbjNcumul` en utilisant une liste en compréhension (si possible).
2. Construisez de même `nbjBcumul` à partir de `nbjB`.

Exercice 8. –

1. Écrivez une fonction `nb_jour_a()` qui, recevant en argument une `date` et l'année `a`, donne le nombre de jours séparant cette date du 1er janvier de l'année `a`. (Étudiez soigneusement les cas de dates placées avant ou après le 29 février pour les années bissextiles...)
2. Écrivez une fonction `nb_jour_entre_2a()` qui, recevant en argument 2 années, donne le nombre de jours séparant les 1er janvier de ces 2 années.
3. Calculez le nombre de jours séparant 2 dates données. (Étudiez soigneusement les cas de date avant ou après le 29 février d'une année bissextile...)

Exercice 9. –

1. Sachant que le 12 novembre 2024 est un mercredi, trouvez quel était le jour de la semaine du 1er janvier 1900.
2. Construisez alors le calendrier de l'année 1789 (jour, mois).
3. Quel jour de la semaine était le 14 juillet de cette année-là ?
4. Quel jour de la semaine était le 15 octobre 1582, date du 1er jour du calendrier grégorien (c'est-à-dire notre calendrier actuel) ?

Exercice 10.
1. Créez une fonction `convertdate()` qui convertit le jour en un quadruplet, par exemple : `convertdate("mi 25 noviembre 2015")` deviendra `(3, 25, 11, 2015)`.
2. Créez une fonction `traduiredate()` qui effectue l'opération inverse. Exemple : `traduiredate((3, 25, 11, 2015))` donnera `miercoles 25 noviembre 2015`.
3. Traduisiez dans votre "3e langue" les dates de l'exercice précédent.