Langages de script (LS4)

Examen du 20 juin 2011 (session 2)

Durée: 3 heures, sans documents

- Vous devez éteindre et ranger vos téléphones.
- Les programmes sont à faire en PYTHON 3.
- Chaque exercice doit être rédigé sur une copie distincte.

Exercice 1 – *Compréhension*

Donnez le résultat des commandes suivantes : étant donné que le ; représente l'enchaînement de deux commandes et que :

```
l = [1,2,3,4,5,6,7,8,9]
d = {1:'a',2:'b',3:'c'}
s = {1,2,3}
```

1. `l[0]`
2. `l[10]`
3. `l[1:5:2]`
4. `l[-2:-1]`
5. `l[-1:-2]`
6. `s.add(2) ; s`
7. `[(i*i,j+j) for i,j in d.items()]`
8. `{d[a]:a for a in d}`

Exercice 2 – *Évènements historiques*

Le but de cet exercice est de faire un script permettant de gérer une petite "chronique" des évènements historiques .

On donnera la chaîne représentant un évènement[1] :

```
Nom: Édit de Nantes
Type: Loi
Personne: Henri IV
Année: 1598
Religion: Protestantisme
Pays: France
Descriptif: un édit de tolérance par lequel le roi reconnaît la liberté de culte aux protesta
```

Le nom et l'année sont toujours présents, tous les autres attributs et leur ordre sont arbitraires (et peuvent varier selon les évènements). À chaque attribut correspond une valeur représentée par une chaîne de caractères (sauf l'année qui est un entier).

Chaque évènement sera représenté par un dictionnaire PYTHON(pour l'exemple ci-dessus, à la clé `"Personne"` correspondra la valeur `"Henri IV"`). Décidez vous-même si la clef `"Nom"` (et sa valeur) sont présents dans le dictionnaire.

1. Source : Wikipedia.

1. Écrivez une fonction appelée `lire_evenement` prenant en entrée l'évènement sous la forme d'une chaîne de caractères et renvoyant le dictionnaire correspondant à l'évènement.
2. Une chronique est un fichier regroupant différents évènements, séparés par des lignes blanches. La chronique sera représentée par un dictionnaire dont les clefs sont les noms des évènements et les valeurs sont les dictionnaires associés à chacun des évènements.
   Écrivez une fonction appelée `lire_chronique` prenant en entrée le nof de fichier contenant une chronique et renvoyant le dictionnaire représentant la chronique.
3. Écrivez une fonction appelée `sauve_chronique` prenant en entrée un dictionnaire et le nom de fichier dans lequel on veut écrire les évènements au format décrit à la question précédente.
4. Écrivez une fonction `liste_chrono` prenant en entrée un dictionnaire chronique et renvoyant la liste de noms de tous ses évènements dans l'ordre chronologique (c-à-d dans l'ordre croissant des années).
5. Écrivez une fonction `liste_attr` prenant en entrée un dictionnaire chronique et renvoyant la liste de tous les attributs de tous ses évènements (sans doublons).
   Indication : l'utilisation des moyens "PYTHONesques" (mutation des listes ...) est recommandée (et sera récompensée).
6. Écrivez une fonction `supprime_evenement` prenant en entrée un dictionnaire `chronique` et une chaîne `pays`. Cette fonction doit supprimer de la chronique tous les évènements concernant le pays donné et garder tous les autres. **Indication :** les évènements sans attribut "Pays" doivent rester aussi dans la chronique.
7. Écrivez un script `histoire.py`, utilisable en ligne de commande avec la spécification suivante :
   - `histoire.py -l FICHIER` doit lister les noms de tous les évènements d'une chronique stockée dans le FICHIER.
   - `histoire.py -a FICHIER` doit lister les attributs de tous les évènements d'une chronique stockée dans le FICHIER.
   - `histoire.py -x FICHIER PAYS` doit lire la chronique stockée dans le FICHIER, supprimer tous les évènements concernant le PAYS, et sauvegarder le résultat dans le même fichier.

   Indication : vous pouvez utiliser les fonctions des questions 1 – 6 même si vous ne les avez pas programmées.

Exercice 3 – *Simulations de compteurs*

Les compteurs sont un outil permettant de compter plus facilement que les dictionnaires :

```
>>> cnt = Counter()
>>> for word in ['red', 'blue', 'red', 'green', 'blue', 'blue']:
...     cnt[word] += 1
>>> cnt
Counter({'blue': 3, 'red': 2, 'green': 1})
```

Plusieurs opérations mathématiques usuelles sont définies sur ces objets :

```
>>> c = Counter(a=3, b=1)
>>> d = Counter(a=1, b=2)
>>> c + d
Counter({'a': 4, 'b': 3})
>>> c - d                       # conserve uniquement les occurrences positives
Counter({'a': 2})
>>> c & d                       # intersection:  min(c[x], d[x])
Counter({'a': 1, 'b': 1})
>>> c | d                       # union:  max(c[x], d[x])
Counter({'a': 3, 'b': 2})
```

On se propose dans cet exercice de définir les opérations +,-,&,| directement sur les dictionnaires pour en faciliter les manipulations.

1. Écrivez pour cela quatre fonctions `plus`, `moins`, `et`, `ou` prenant en argument deux dictionnaires pour lesquels les valeurs sont des entiers. Elles doivent se comporter de la même façon que les opérations sur les compteurs. et renvoyer le dictionnaire correspondant. Attention, les dictionnaires ne contiennent pas forcément les mêmes clés. Vous devez faire comme pour les compteurs dans ces cas-là :

```
>>> c = Counter(a=3, b=1)
>>> d = Counter(e=4, f=5)
>>> c | d
Counter({'f': 5, 'e': 4, 'a': 3, 'b': 1})
```

   Rappel : si on essaie d'afficher la valeur d'une clef qui n'existe pas un dictionnaire, on obtient une erreur `KeyError`. Évidemment, vous devez faire en sorte que cela n'arrive pas.

2. Les fonctions définies à la question 1 présentent le défaut de planter si on leur passe en argument un dictionnaire qui a autre chose que des entiers comme valeurs. Pour éviter cela, on va décider de faire les opérations lorsque c'est possible et laisser tel quel sinon. Par exemple, on veut avoir :

```
>>> d = {'a':2, 'b':[]}
>>> e = {'c':'k', 'a':5}
>>> plus(d, e)
{'a':7, 'b':[], 'c':'k'}
```

   Pour faciliter les choses, on va utiliser de façon rudimentaire le mécanisme des exceptions. Voici un exemple d'utilisation :

```
>>> d = {'j':[]}
>>> e = {'f':4}
>>> try:
...     d['j'] + e['f']
... except:
...     print('erreur')
...
erreur
```

   Réécrivez **uniquement** la fonction plus pour qu'elle ait ce comportement à l'aide d'exceptions.