S3H - PROG - Contrôle de TD
lundi 14 novembre 2016
durée :1h30 - calculatrices/documents non autorisés (sauf mémento Python 3)

Université de Lille 1 SCIENCES ET TECHNOLOGIES — UFR IEEA Formations en Informatique de Lille 1

[le Barème donné est indicatif (long mais sur 22 points, donc 2 points de bonus)]

Exercice 1 (6 pt) : *LISTES*

On définit à titre d'exemple la liste suivante `ls = [131,42,9,42,131]`

Q1-1 (1.0 pt) : Qu'affiche ce code python 3 ? Combien de lignes seront affichées ?

```
for e in ls:
    print(e, end=' ')
```

Q1-2 (1.5 pt) : Donnez le code python 3 de la fonction `palindrome(ls)` qui retourne `True` *si et seulement* si la liste `ls` passé en paramètre est un « palindrome » lorsque l'on compare ses éléments respectifs : vous devez par exemple obtenir `True` si `palindrome(ls)` est appliqué sur la liste proposée en exemple. Notez qui vous **ne devez pas** construire de **nouvelle liste**.

Q1-3 (1.0 pt) : En quoi le code que vous avez proposé à la question précédente *peut* /ou/ *ne peut pas* être appliqué à des chaînes de caractères plutôt qu'à des listes (`type(ls) == string`)? Justifiez clairement votre réponse.

Q1-4 (2.5 pt) : On souhaite, à partir de deux listes `ls1` et `ls2` (non vides mais de taille potentiellement différentes), réaliser la création d'une *liste de couples d'éléments* communs aux indices *i* des deux listes. Donnez le code python 3 de la fonction `combine(ls1,ls2)` qui réalise cela **sans utiliser la fonction « zip »**.

exemple : si `ls1 = [1,2,3]` et `ls2 = ['a','b']` alors `combine(ls1,ls2)` retourne `[(1,'a'),(2,'b')]`

Exercice 2 (5 pt) : *TRANCHES DE LISTES et RANGE*

On définit la liste suivante `l = [1,2,3,4,5]`

Q2-1 (0.5 pt): Qu'affiche cette suite d'instructions dans l'interpréteur python 3 ?

```
>>>l[2:]
>>>l[:2]
>>>l[:2] + [23] + l[2:]
```

Q2-2 (1.0 pt): Qu'affiche cette suite d'instructions dans l'interpréteur python 3 ?

```
>>>l[1:2] = [22] ; l
>>>l[2:] = [] ; l
```

Q2-3 (1.0 pt): Qu'affiche cette suite de 4 blocs d'instructions dans l'interpréteur python ?

```
>>> for i in range(5):
>>>         print(i,end=' ')
>>> for i in range(1,5):
>>>         print(i,end=' ')
>>> for i in range(1,5,-1):
>>>         print(i,end=' ')
>>> for i in range(5,1,-1):
>>>         print(i,end=' ')
```

Q2-4 (2.5 pt): Donnez le code python 3 pour tester si un entier `e` est à la fois :
- compris entre 1 et 42,

- est pair, mais n'est pas multiple de 3 :

(a) avec un seule expression booléenne courte (composition au choix : les « % », « and », « not », « <= », « >= », « == » ou tests bit à bit « & » et « | » vu en codage sont autorisés) ?

(b) à l'aide de deux tests d'appartenance à deux *ranges* (que vous définirez auparavant), et une expression booléenne utilisant seulement un « and » et un « not »

Exercice 3 (5 pt) : *CATALAN*

On donne cette définition des nombres de Catalan C_n

$$C_0 = 1 \quad \text{et} \quad C_{n+1} = \sum_{i=0}^{n} C_i \, C_{n-i} \quad \text{pour } n \geq 0.$$

On souhaite calculer la liste des nombres de Catalan jusqu'au rang « n-1 », pour un « n » strictement positif passé en paramètre.

exemple : si **`n = 6`** alors **`liste_catalan(n)`** retourne **`[1,1,2,5,14,42]`**

Q3-1 (4 pt): Proposez un implémentation itérative rapide de cette fonction **`liste_catalan(n)`**

Q3-2 (1 pt): Justifiez son efficacité par rapport à une version récursive standard (non terminale)

Exercice 4 (6 pt) : *DICTIONNAIRES*

On donne une longue chaine de caractères **s = "la cigale ayant chanté tout l été se trouva" ...**

On souhaite d'abord compter la fréquence des mots afin de retrouver les plus utilisés, ainsi que leurs positions dans un second temps.

Q4-1 (0.5 pt) : En évoquant vos souvenirs du *TP anagrammes (**et** en vous aidant du mémento python)*, pouvez-vous transformer cette phrase sans signe de ponctuation en liste de mots **ls** sans *blancs (espaces* ou *tabulations)*

Q4-2 (1.5 pt) : Proposez une fonction efficace **`compte_occurences(ls)`** qui compte le nombre d'occurrences de chaque mot présent au moins une fois dans la liste de mots **ls.** Cette fonction retournera directement cette « collection de comptes » via la structure de données la plus adaptée pour réaliser ce compte.

Q4-3 (1.0 pt) : Proposer un procédure qui affiche le mot ou les mots qui atteignent le nombre maximal d'occurrences

Q4-4 (2.0 pt) : Proposez une fonction efficace **`donne_occurences(ls)`** qui est capable de donner les indices des occurrences des mots présents au moins une fois dans la liste **`ls`**.

Q4-5 (1.0 pt) : On suppose qu'il y a « toujours » un seul blanc consécutif dans la chaîne de caractères **s et pas** de blanc avant le premier mot. Comment, dans ce cas, modifier la fonction précédente pour connaître l'indice original des occurrences des mots dans la chaîne **s** et non dans la liste **ls** *(indiquez la modification avec des flèches sans nécessairement tout réécrire)*