Université de Lille 1 SCIENCES ET TECHNOLOGIES

ufr d'IEEA
Formations en
Informatique de
Lille 1

L2 Informatique - 2015–2016

Algorithmes et Programmation 2

DS1 - 16/10/2015

durée 2h - documents non autorisés

Ce devoir contient trois exercices indépendants. Sauf mention du contraire, on ne vous demande pas la documentation des fonctions que vous réaliserez.

Exercice 1 *Modules au service de l'organisation d'un marathon*

Vous êtes chargé de la gestion informatique des participants à un marathon. Ces participants sont caractérisés par leur nom, un numéro de dossard et la durée mise pour parcourir les 42,195 km de l'épreuve.

Pour cela vous allez réaliser deux modules :

1. l'un pour la structure de données décrivant un participant (module `competitor`) ;
2. et l'autre pour la représentation des durées de parcours (module `time`).

Module `time`

Question 1 Réalisez ce module avec
- un constructeur `create`, dont vous donnerez la documentation, prenant une durée décrite par trois nombres entiers : le nombre d'heures, de minutes et de secondes, étant entendu que ce constructeur déclenche une exception `InvalidTimeError` (que vous supposerez définie) si les données ne peuvent pas représenter une durée ;
- les sélecteurs associés `get_hours`, `get_minutes` et `get_seconds` ;
- et un prédicat `before` qui prend deux durées en paramètre, et répond `True` si la première est plus courte que la seconde, `False` sinon.

Module `competitor`

Question 2 Réalisez ce module avec
- un constructeur `create` construisant la fiche d'un participant au marathon en y portant son nom, son numéro de dossard et une durée de parcours qui par défaut est `None` (à l'inscription, la durée de parcours n'est pas encore connue !) et qui doit bien évidemment pouvoir être modifiée par la suite ;
- les sélecteurs associés ;
- et un modificateur pour la durée de parcours.

Question 3 Expliquez en quoi il n'est pas gênant de travailler avec deux constructeurs portant le même nom.

À la fin de l'épreuve : le juge de course vous apporte la liste (de Python) des résultats de la course. Cette liste contient les performances réalisées par chacun des candidats ayant terminé l'épreuve sous la forme d'un quadruplet `(num_dossard,heures,minutes,secondes)`. Certains candidats peuvent ne pas figurer dans cette liste car ils ont abandonné.

Question 4 Écrivez une fonction qui, à partir d'une liste de candidats inscrits, dans laquelle aucune durée ne figure encore, et une liste des performances, reporte les durées de parcours dans toutes les fiches des candidats qui ont accompli l'épreuve.

Exercice 2 *Récursivité*

Les questions de cet exercice sont indépendantes. Pour chacune d'elles vous préciserez si la fonction que vous réaliserez est récursive terminale ou non en expliquant pourquoi.

Question 1 Réalisez une fonction récursive qui calcule le nombre de chiffres de l'écriture décimale d'un nombre entier passé en paramètre.

```
>>> size (0)
1
>>> size (1234)
4
```

Question 2 Réalisez une fonction nommée `shuffle` paramétrée par deux listes `l1` et `l2` qui renvoie une liste dont les éléments sont ceux de ces deux listes dans un ordre alterné.

```
>>> shuffle ([1,3,5,7,9],[2,4,6])
[1, 2, 3, 4, 5, 6, 7, 9]
```

On rappelle que deux mots sont *anagrammes* l'un de l'autre si on peut former l'un des mots avec les lettres de l'autre. Ainsi, `égratigna` et `gagnerait` sont des anagrammes.

Question 3 Proposez une version récursive d'un prédicat nommé `are_anagrams` qui détermine si deux mots sont des anagrammes (vous pouvez négliger les différences entre lettres capitales ou non, accentuées ou non).

Indication : utilisez la méthode `s.index(c)` des chaînes de caractères qui renvoie l'indice de la première occurrence du caractère `c` dans la chaîne `s` s'il y en a une, et déclenche l'exception `ValueError` sinon, ainsi que les constructions indicielles de sous-chaînes.

```
>>> are_anagrams ('egratigna','gagnerait')
True
>>> are_anagrams ('ensabler','ebranlees')
False
```

Exercice 3 *Autour du tri rapide*

Les listes de cet exercice sont des listes de nombres que l'on pourra supposer tous différents.

Question 1 Sans le programmer, rappelez le principe du tri rapide d'une liste vu en cours. Précisez les étapes récursives de ce tri. Et illustrez-le sur la liste `[20, 22, 7, 44, 1, 19, 17, 21]`.

Question 2 Quel est le nombre de comparaisons entre éléments de la liste à trier lorsqu'à chaque étape le pivot est le plus petit élément ?

Dans le tri rapide, il est préférable que le pivot choisi à chaque étape soit l'*élément médian* de la liste à trier.

On rappelle que l'*élément médian* d'une liste est un élément de la liste tel que la moitié (arrondie à l'entier inférieur) des autres éléments lui sont inférieurs et l'autre moitié (arrondie à l'entier supérieur) supérieurs. Ainsi, l'élément médian de la liste de la première question est 19.

Question 3 Réalisez une fonction nommée `median` qui renvoie l'élément médian d'une liste.

```
>>> median ([20, 22, 7, 44, 1, 19, 17, 21])
19
```