S3H - PROG – Examen de fin de module
mercredi 13 décembre – 16h00-18h00 (Examens A5)
durée :2h - 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 : sur 24 points donc 4 points de bonus

Ex 1 et Ex 3 : nous rappelons que `random.randint(a,b)` ***retourne une valeur entière*** `r` ***aléatoire uniforme dans l'intervalle*** `a <= r <= b`

Exercice 1 (7 pt) : Questions de TP et de Cours

TP Bon parenthesage

Q1-1 (0.5 pt): À quoi sert la pile dans le TP "bon parenthésage" ?

Q1-2 (1.0 pt): Quelles différentes erreurs avez-vous traitées dans le TP "bon parenthésage" ? A quoi correspondent-elles dans l'état de la pile ?

Cours

Q1-3 (1.0 pt): Que signifie l'opérateur "=>" pour des ensembles ? Pouvez vous donner un exemple valant "True" ainsi que deux exemples (chacun étant de *nature différente*) valant "False" ?

Q1-4 (1.5 pt): Expliquez le principe, et si possible, donnez un code python 3, pour mélanger une liste **sans utiliser** `random.shuffle` ?

Q1-5 (1.0 pt): Qu'est ce qu'un "hash" ? Dans quelle(s) structure(s) de données python 3 apparaît-il ? Qu'est ce qu'une "collision sur un hash" ?

Q1-6 (2.0 pt): Donnez un **vrai nom**, et expliquez rapidement le comportement du *code suivant* lors de l'appel `my_what(l,0,len(l)-1)` sur la liste `l = [1,2,3,4,5]`. Statistiquement, comment, améliorer ce comportement en a*joutant quelques lignes* juste avant la ligne critique **`p = l[u]`** ?

```
def my_what(l,u,v):
    if u < v:
        # (a) qui a pu écrire tout ce code sans un seul commentaire?
        p = l[u]    # p ??
        i = u + 1   # first index
        j = v       # last index
        while i <= j:
            if l[i] < p :
                i += 1
            elif l[j] >= p:
                j -= 1
            else:
                l[i],l[j] = l[j],l[i]
                i += 1
                j -= 1
        # (b) ceci est un échange, semblerait-il?
        l[i-1],l[u] = l[u],l[i-1]
        # (c) ceci est un double appel récursif, semblerait-il?
        my_what(l,u,i-2)
        my_what(l,i,v)
```

Exercice 2 (9 pt) : Classe `List`

Dans cet exercice, ne vous focalisez pas sur la "syntaxe exacte" : je serais tolérant sur ce point, mais indiquez juste vos doutes ...

Q2-1 (0.5 pt): Comme vu en cours *AP2 - List*, proposez une "Vrai" **classe `List`**

Q2-2-a (1.0 pt): Elle doit disposer d'un constructeur d'objet avec deux attributs : une valeur (*value*) et un suivant (*next*), les deux étant passés en paramètre. Si possible, lors de l'appel au constructeur, le second attribut doit être mis à `None` s'il n'est pas fourni ; autrement, il doit être de type `List`. Les deux attributs associés doivent être *privés*.

Q2-2-b (0.5 pt): Ajoutez une assertion sur le précédent constructeur, qui vérifie la cohérence du type du second attribut (le type doit être, soit `NoneType`, soit `List`)

Q2-3 (1.0 pt): Spécifiez et implémenter deux accesseurs `get_value` et `get_next`

Q2-4 (1.5 pt): Spécifiez et implémenter deux modificateurs `set_value` et `set_next` disposant, pour le second au moins, d'une assertion de type

– questions à réaliser hors de la classe List, pour faciliter l'écriture –

Q2-5 (1.0 pt): Spécifiez et implémenter une fonction qui insère un élément *en tête de liste.*

Q2-6 (1.5 pt): Spécifiez et implémenter une fonction qui insère un élément *en fin de liste*. Indiquez sa complexité.

Q2-7 (2.0 pt): Spécifiez et implémenter une fonction qui insère un élément *à un indice donné dans la liste.* Vous pouvez lever des exceptions d'un type supposé prédéfini `ListIndexError` au besoin.

Exercice 3 (8 pt) : Mélanges de cartes (d'après, heu ... les pauses de certains S3H cette année :-p)

Nous allons programmer dans cet exercice deux mélanges de cartes classiques. Tous ces mélanges seront réalisés sur une liste python 3 qui représente le paquet (`pop` et `append` accèdent ici au sommet du paquet). Le fait que l'on stocke, dans cette liste, des cartes, ou toute autre objet, ne change absolument pas le problème qui est de ce fait générique : n'utilisez donc pas ici le module « card » ici, ce n'est pas nécéssaire.

Stack shuffle

*Le paquet initial de **n** cartes est coupé en deux sous-paquets (non-vides) : le premier est d'une taille donné par une variable aléatoire **r** comprise entre **1** et **n-1** (le second est donc de taille complémentaire); Ces deux sous-paquets sont ensuite remis l'un sur l'autre, dans le **seul ordre** qui **ne recrée pas le paquet avant coupe.***

Q3-1 (1.5 pt): Réaliser une fonction ou procédure `stack_shuffle_once(l)` qui réalise la coupe/replacement une seule fois, pour un jeu de *n* cartes donné dans une liste `l`, en choisissant la manière de retourner la liste symbolisant le paquet après coupe ; Expliquez les deux choix possibles de retour de votre fonction ou procédure.

Q3-2 (1.0 pt): Proposez une documentation pour cette fonction/procédure.

Q3-3 (1.0 pt): Expliquez et proposer un test unitaire pour vérifier qu'un seul point de coupe a été réalisé (vous pouvez vous aider d'une fonction/procédure supplémentaire que vous définirez).

On effectue ensuite n fois la précédente étape de coupe pour "bien mélanger"

Q3-4 (1.0 pt): Appliquez *n* fois le principe de la (**Q3-1**) afin d'avoir un mélange *suffisant* (format de fonction/procédure libre, mais à documenter)

Q3-5 (1.0 pt): Sur un jeu réduit de *n = 3* cartes (ou plus), pouvez-vous montrer que le mélange réalisé n'est pas *homogène* ? (*nb : au moins deux méthodes sont possibles*)

Riffle shuffle

*On suppose que l'on mélange désormais les **r** et **n-r** cartes par entrelacement **une à une** pour les **min(r,n-r)** première cartes du bas des deux sous-paquets, suivi du reste du sous paquet le plus grand, en prenant la même contrainte que les piles (la carte du dessous du paquet initial **ne doit jamais** se retrouver en dessous après ce mélange) : il est bon de noter que cela ne donne **qu'un seul mélange possible**, une fois la coupe choisie.*

Q3-6 (1.5 pt): Réaliser une fonction/procédure `riffle_shuffle_once(l)` équivalente à la (**Q3-1**) pour ce type de mélange.

Q3-7 (1.0 pt): Appliqué *n* fois sur le paquet initial, le mélange est-il *homogène*, à partir de *n = 3* cartes ?