Introduction aux listes en Python - Corrigé

NUMWORKS

Cette fiche a été rédigée par **Philippe Moutou**. Il enseigne au lycée Henri IV à Paris.

1 Exercice

Écrire une fonction `elimine(L)` qui supprime les doublons d'une liste : `elimine([1,2,2,1,2,1])` renvoie `[1,2]`.

On peut parcourir la liste `L` de départ et ajouter dans une liste `P` les éléments de `L` s'ils n'y sont pas déjà.

```
def elimine1(L):
  P=[]
  for e in L:
    if e not in P:
      P.append(e)
  return P
```

Une seconde option serait, pour chaque élément de `L`, de se demander s'il est présent à un rang supérieur et dans ce cas supprimer les occurrences redondantes (on n'a pas alors besoin d'une liste `P`).

```
def elimine2(L):
  for e in L:
    if L.count(e)>1:
      for i in range(L.count(e)-1):
        L.remove(e)
  return L
```

2 Autre exercice

Écrire une fonction `cartes(n)` qui renvoie une main de N cartes prises au hasard dans un jeu de 32 cartes. Par exemple, avec `cartes(2)`, on devrait pouvoir obtenir `[As de Trèfle, Dix de Pique]`.

Une première approche est la fonction `cartes(n)` qui choisit n fois une valeur et une couleur dans les listes correspondantes avec la fonction `choice(liste)`.

```
from random import *
def cartes(n):
  couleur=['Trefle','Carreau','Coeur','Pique']
  valeur=['7','8','9','10','Valet','Dame','Roi','As']
  main=[]
  for i in range(n):
    carte="{} de {}".format(choice(valeur),choice(couleur))
```

```
    main.append(carte)
  return main
```

Pour éviter d'avoir deux cartes identiques, on peut remplacer la boucle **for** par une boucle **while**. La condition d'arrêt du **while** peut porter sur la longueur de la liste **main** créée. Avant d'ajouter une nouvelle carte, on vérifie que l'élément n'est pas déjà présent dans la liste.

```
from random import *
def cartes(n):
  couleur=['Trefle','Carreau','Coeur','Pique']
  valeur=['7','8','9','10','Valet','Dame','Roi','As']
  main=[]
  while len(main)<n:
    carte="{} de {}".format(choice(valeur),choice(couleur))
    if carte not in main:
      main.append(carte)
  return main
```

Pour prolonger ce travail, on pourrait écrire un programme qui renvoie P mains différentes de N cartes, les P mains ne pouvant contenir les mêmes cartes. On pourrait ainsi distribuer le jeu entre P joueurs, par exemple `cartes(8,4)` distribuerait 8 cartes à 4 joueurs.

Pour réaliser ce projet, il serait peut-être plus judicieux de générer une liste contenant l'ensemble des cartes et de tirer ensuite dans cette liste un élément au hasard que l'on supprimerait de la liste (pour éviter l'inconvénient d'une boucle **while** qui tire très souvent et pour rien des cartes déjà tirées). Le programme qui suit réalise cela, d'une part en parcourant les deux listes (**couleur** et **valeur**) en entier pour constituer la liste **jeu**, et d'autre part en tirant un élément de la liste jeu au hasard et en l'y supprimant avec cette instruction à tiroirs : `main.append(jeu.pop(randrange(len(jeu))))`.

```
from random import *
def cartes(n,p):
  couleur=['Trefle','Carreau','Coeur','Pique']
  valeur=['7','8','9','10','Valet','Dame','Roi','As']
  jeu,mains=[],[]
  for c in couleur:
    for v in valeur:
      jeu.append("{} de {}".format(v,c))
  for i in range(p):
    main=[]
    while len(main)<n:
      main.append(jeu.pop(randrange(len(jeu))))
      mains.append(main)
  return mains
```

Pour distribuer 8 cartes à chacun des 4 joueurs, on peut écrire :

```
nb_cartes,nb_joueurs=8,4
jeux=cartes(nb_cartes,nb_joueurs)
for i in range(nb_joueurs):
  print("Joueur {}:".format(i+1))
  print(" - ".join(jeux[i]))
```