Examen - Programmation (MASCO 1)
Durée : 2h
Documents autorisés : le polycopié du cours

A lire avant de commencer

— Les questions de cet examen portent sur un même problème, mais elles sont indépendantes, vous pouvez y répondre dans l'ordre que vous voulez.
— Lorsque vous répondez à une question vous pouvez utiliser les questions précédentes, même si vous n'y avez pas répondu.
— Ecrivez de manière la plus L I S I B L E possible, les copies illisibles ne seront pas lues.

1 Trésor volé

Cette question s'inspire d'un jeu d'enfants fondé sur la déduction appelé *trésor volé*. Il est composé d'un ensemble de cartes et se joue à deux joueurs : A et B.

Les cartes sont composées de trois cases. La première comporte un chiffre, la seconde une lettre et la troisième une couleur.

Exemple : $(2, B, Bleu)$, $(3, C, Rouge)$

Une carte R est choisie au hasard par le joueur A, le reste des cartes constitue la pioche. Le but du jeu, pour le joueur B, est de deviner R. Pour cela, il choisit une carte H de la pioche et la montre à A. A lui répond alors 0, 1 ou 2, selon que H et R possèdent 0, 1 ou 2 éléments en commun.

Exemple : si $H = (2, B, Bleu)$ et $R = (1, B, Rouge)$ alors la réponse est 1.

Le jeu continue ainsi jusqu'à ce que B devine R.

Q1.1 Ecrire une fonction `genereCartes(chiffres, lettres, couleurs)` qui renvoie l'ensemble de toutes les cartes que l'on peut générer à partir des trois tuples `chiffres`, `lettres` et `couleurs`.

```
>>> print(genereCartes(('1', '2'), ('A','B'),('Rouge',)))
[('1', 'A', 'Rouge'), ('1', 'B', 'Rouge'), ('2', 'A', 'Rouge'), ('2', 'B', 'Rouge')]
```

Q1.2 Ecrire une fonction `melangeCartes(paquet)` qui prend en argument un paquet de cartes et renvoie un autre paquet, composé des mêmes cartes dans un ordre aléatoire. Pour cela vous n'utiliserez pas de fonctions permettant mélanger une liste, tel que `shuffle`, mais vous utiliserez la fonction `random.randint(a,b)` et ma méthode `l.pop(i)`. La première renvoie un nombre entier aléatoire i, tel que $a \leq i \leq b$, la seconde élimine l'élément d'indice i de la liste l.

```
>>> print(melangeCartes(genereCartes(('1', '2'), ('A','B'),('Rouge',))))
[('2', 'A', 'Rouge'), ('1', 'A', 'Rouge'), ('2', 'B', 'Rouge'), ('1', 'B', 'Rouge')]
```

Q1.3 Ecrire la fonction `compare(c1, c2)` qui prend en arguments deux cartes, représentées par des tuples et renvoie 0, 1, 2 ou 3 selon que `c1` et `c2` ont 0, 1, 2 ou 3 cartes en commun

```
>>> print(compare(('1', 'A', 'Rouge'), ('2', 'A', 'Vert')))
1
```

Q1.4 Ecrire la fonction `arg_compare(c1, c2)` qui prend en arguments deux cartes et renvoie un tuple composé de `0` et de `1`. Si les i-èmes composantes de `c1` et `c2` sont égales, alors la i-ème composante du tuple renvoyé est égal à `1` sinon elle est égale à `0`.

```
>>> print(arg_compare(('1', 'A', 'Rouge'), ('2', 'A', 'Vert')))
(0,1,0)
```

On définit la classe `tv` qui permet de représenter une partie de trésor volé. Cette classe possède une liste appelée `pioche`, une variable `carteCachee` et une liste `cartesTirees` qui contient les cartes tirées jusque là, ainsi que leur score. Le score d'une carte étant le nombre d'éléments qu'elle a en commun avec la carte cachée.

Q1.5 Ecrire un constructeur pour la classe `tv` qui prend en argument trois tuples, correspondant aux chiffres, lettres et couleurs. Ce constructeur crée le jeu de cartes, le mélange, enlève une carte qui constituera la carte cachée.

Q1.6 Ecrire la méthode `affiche` qui affiche à l'écran l'état du jeu sous la forme suivante :

```
>>> t = tv(('A', 'B'), ('C','D'),('E'))
>>> t.affiche()
chiffres =  ('A', 'B')
lettres =  ('C', 'D')
couleurs =  ('E')
carte cachée =  ('A', 'D', 'E')
pioche =  [('B', 'C', 'E'), ('A', 'C', 'E'), ('B', 'D', 'E')]
cartes tirées =  []
```

Q1.7 Ecrire la méthode `tireCarte()` qui enlève une carte de la pioche, la compare à la carte cachée et l'ajoute à la liste `cartesTirees` avec son score.

```
>>> t.tireCarte()
>>> t.affiche()
chiffres =  ('A', 'B')
lettres =  ('C', 'D')
couleurs =  ('E')
carte cachée =  ('B', 'D', 'E')
pioche =  [('B', 'C', 'E'), ('A', 'C', 'E')]
cartes tirées =  [(('A', 'D', 'E'), 2)]
```

Q1.8 Ecrire la méthode `carteValide(c)` qui renvoie `True` si la carte `c` est valide et `False` sinon. Une carte est valide si elle possède un chiffre, une lettre et une couleur et que le chiffre appartient à l'ensemble des chiffres de l'objet, de même pour les lettres et les couleurs.

Q1.9 Ecrire la méthode `solution(c)` qui prend en argument la carte `c` et renvoie `True` si `c` est la carte cachée et `False` sinon.