TI 1520 Devoir 5

Instructions

Répondez à la question 1 dans le fichier d5q1.py (qui est partiellement complété), et à la question 2 dans le fichier d5q2.py. Compressez le tout dans D5-NumeroEtudiant.zip.

N'oubliez pas d'ajouter des commentaires dans chaque programme pour expliquer le but du programme, la fonctionnalité de chaque fonction et le type de ses paramètres ainsi que le résultat, et les parties compliquées de vos programmes. Des points vous seront enlevés si les commentaires sont déficients.

Question 1 (12 points)
Jeu de 21 (Blackjack)

On va développer plusieurs classes : Carte (pour simuler une carte à jouer), JeuDeCartes (pour un jeu de cartes composé de 52 cartes), Main (les cartes que possède un joueur pendant le jeu) et Blackjack (pour notre jeu de 21).

Voici les règles du Blackjack (simplifiées) :
Pour commencer le jeu, la banque tire des cartes du paquet. La première carte est remise au joueur, la deuxième à la banque, la troisième au joueur, et la quatrième à la banque. La banque a donc deux cartes, et le joueur aussi. Par la suite, le joueur pourra demander d'autres cartes.
Le but du jeu consiste à détenir une main de cartes dont la valeur totale est la plus proche possible de 21, sans dépasser 21.
La valeur des cartes est établie comme suit :

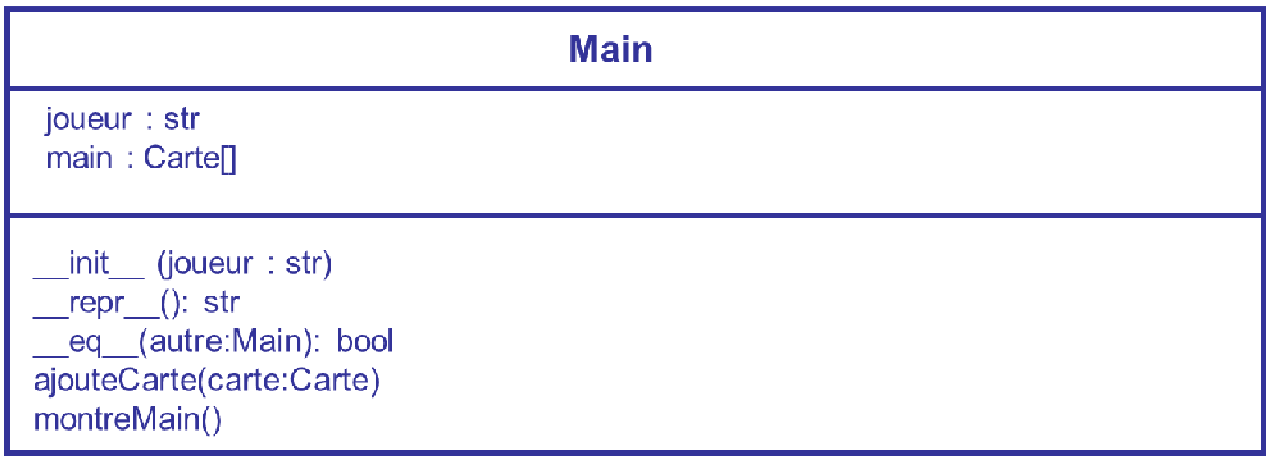
- de 2 à 9 → la valeur nominale de la carte
- chaque figure + le 10 surnommées "bûche" → 10 points
- l'As → 1 point ou 11 points (selon le choix fait par le joueur pour que sa main soit le plus proche possible de 21 sans dépasser 21).

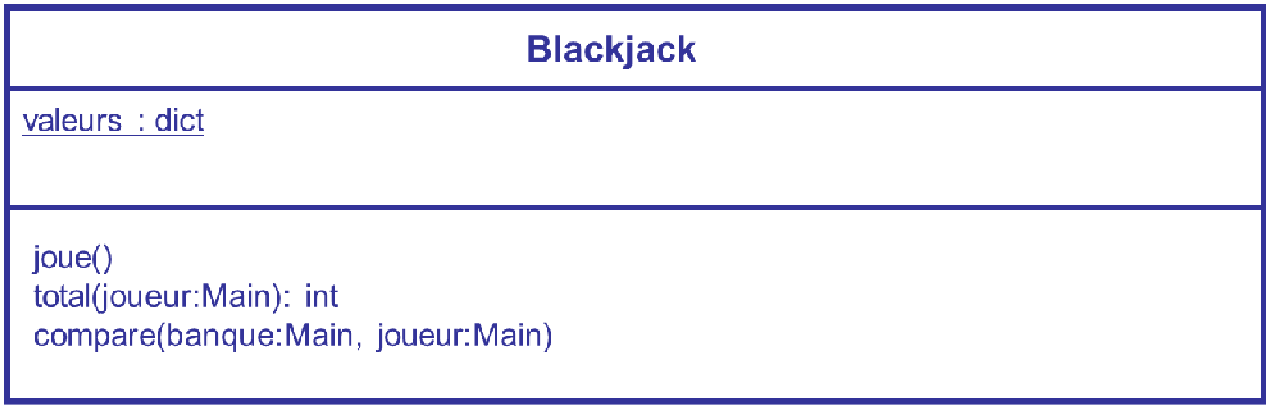
Si le joueur a demandé une carte, et si la valeur totale de sa main dépasse 21, il a perdu. Sinon il peut encore demander des cartes, ou il peut s'arrêter.
Si le joueur s'arrête, la banque doit tirer des cartes en suivant les règles suivantes : si la valeur totale de sa main est inférieure à 17, elle doit continuer à tirer des cartes; si elle dépasse 21, le joueur a gagné; si elle est 17 ou plus (<= 21), la banque s'arrête.
Dans ce cas, il faut comparer la main du joueur avec la main de la banque. Si la valeur de la main du joueur est supérieure à la valeur de la main de la banque, le jouer a gagné. Si elle est inférieure, le joueur a perdu. En cas d'égalité, on affichera Egalité, sauf si la valeur est 21 et que le joueur ou la banque a un blackjack (une bûche/valeur 10 et un As). Si le joueur a le blackjack, il a gagné. Si la banque a le blackjack, le joueur a perdu. (Si les deux ont un blackjack c'est égalité.)

La classe Carte et la classe JeuDeCartes sont fournies. **Vous devez compléter les deux autres classes**.

La classe Carte contient une valeur et une couleur. Chaque carte aura donc une valeur et une couleur. Les valeurs sont des caractères comme '2','3','4','5','6','7','8','9','10','J','Q','K', et 'A'. Les couleurs sont des caractères comme : ♠, ♡, ♣, et ♢. (On utilisera 4 symboles Unicode pour représenter les 4 couleurs: pique, coeur, trèfle, et carreau.)

La classe JeuDeCartes contient deux variables de classe: `valeurs` qui contiennent les valeurs énumérées ci-dessus, et `couleurs` qui contiennent les 4 symboles Unicode pour les couleurs. Il y a aussi une variable d'instance, appelée `paquet`, qui devrait contenir les 52 cartes.
La méthode __init__() initialise la liste `paquet` avec les 52 cartes.
La méthode battre() mélange les cartes dans la liste pour qu'elles soient dans un ordre aléatoire.
La méthode tireCarte() retourne une carte du paquet, la première.





A compléter :

La classe Main va être utilisée pour la main courante d'un joueur. Dans la class Blackjack, on va utiliser seulement deux joueurs: vous (appelé 'joueur') et la banque (appelée 'banque').
La méthode `__init__()` doit initialiser le nom du joueur à une chaine de caractères et la liste `main` avec une liste vide.
La méthode `ajouteCarte()` doit ajouter une carte à la liste `main`
La méthode `montreMain()` doit afficher le nom du joueur et les cartes dans la liste.
La méthode `_repr_()` doit retourner une chaine de caractères avec la même information.
La méthode `_eq_()` doit vérifier si deux mains sont similaires (le même nom du joueur et les mêmes listes de cartes).

La classe Blackjack mène le jeu.
La méthode `joue()` **est déjà complétée**. Elle crée un jeu de cartes et bat les cartes pour qu'elles soient dans un ordre aléatoire. Elle crée une main pour le joueur et une main pour la banque (mais vous devrez compléter le constructeur de la classe Main avant que les mains soient vraiment créées). Ensuite, on donne deux cartes au joueur et deux cartes à la banque. On demande au joueur s'il veut d'autres cartes et, si oui, on les lui donne. Si le joueur dépasse 21, il a perdu. Sinon, la banque prend des cartes, en suivant ses règles (si la valeur totale de sa main est inférieure à 17, elle doit continuer à tirer des cartes ; si elle dépasse 21, le joueur a gagné ; si elle est 17 ou plus (<= 21), la banque s'arrête.). Si 21 n'a pas été dépassé par le joueur ni par la banque, on compare les mains pour désigner le gagnant.

La méthode `total()` permet de calculer la somme des valeurs de toutes les cartes dans la main. Les As ont la valeur 11, mais, s'il est plus avantageux pour le joueur d'accorder à un As la valeur 1, on doit modifier la somme.
La méthode `compare()` permet de comparer la main du joueur avec la main de la banque et d'afficher des messages. Si le total obtenu par la banque est supérieur au total obtenu par le joueur, le message à afficher est 'Vous avez perdu.' S'il est inférieur, le message à afficher est 'Vous avez gagné.' En cas d'égalité, le message est 'Egalité.', sauf si le total est 21 et que la banque ou le joueur (mais pas les deux) a un blackjack (si la banque a le blackjack, le message est 'Vous avez perdu.' ; si le joueur a le blackjack, le message est 'Vous avez gagné.')

Exemples (plusieurs jeux):

```
Banque:   Q ♣   3 ♡
Joueur:   3 ♣   8 ♠
Carte? Oui ou non? (Par défaut oui)
Vous avez:
Carte(4, ♡)
Carte? Oui ou non? (Par défaut oui)
Vous avez:
Carte(9, ♣)
Vous avez dépassé 21. Vous avez perdu.
```

```
Banque:   9 ♡   8 ♣
Joueur:   7 ♠   3 ♠
Carte? Oui ou non? (Par défaut oui) o
Vous avez:
Carte(Q, ♢)
Carte? Oui ou non? (Par défaut oui) n
Vous avez gagné.
```

```
Banque:  10 ♡   2 ♣
Joueur:   2 ♢   K ♢
Carte? Oui ou non? (Par défaut oui)
Vous avez:
Carte(7, ♡)
Carte? Oui ou non? (Par défaut oui) n
La banque a:
Carte(4, ♡)
La banque a:
Carte(3, ♡)
Egalité.
```

Question 2 (8 points)

a. Écrivez une fonction Python **récursive** appelée `etoiles` qui prendra comme paramètre un entier non négatif et qui générera un dessin composé d'étoiles tel qu'affiché ci-dessous. Vous pouvez utiliser une boucle pour générer une ligne d'étoiles, mais pas le dessin en entier.

Exemple :

```
>>> etoiles(4)
****
***
**
*
*
**
***
****
```

b. Écrivez une fonction Python **récursive** appelée `sommeListePos_rec` qui prendra comme paramètre une liste et comme deuxième paramètre le nombre des éléments de la liste, et qui retournera la somme des éléments positifs (> 0).

Exemple :

```
>>> l = [1,-2,5,0,-5]
>>> sommeListePos_rec(l, len(l))
6
```

Note: Si les fonctions pour Q2 a,b ne sont pas récursives, le nombre de points reçus sera **zéro**. (Vous pouvez tester des fonctions équivalentes itératives pour vous-même.)