Jeux de Cartes

J. Devalette

1 Régles du jeu

1.1 Déroulement de la Partie

1. On dispose d'un jeu de 24 cartes de couleur vertes, violettes ou orange sur lesquelles est portées un nombre.
2. Distribuer aléatoirement 12 cartes à chaque joueur.
3. Simultanément chaque joueur joue une carte, la carte la plus forte emporte le le pli.
4. Quand toutes les cartes sont jouées, chaque joueur fait la somme des nombres portés par les cartes, le plus fort total l'emporte.

1.2 Valeur des cartes

— Les cartes oranges l'emportent sur les cartes vertes et violettes.
— Les cartes vertes l'emportent sur les cartes violettes.
— Si deux cartes violettes sont jouées en même temps la plus grande l'emporte.
— Si deux cartes vertes sont jouées en même temps la plus grande l'emporte
— Si deux cartes identiques sont jouées en même temps, les cartes sont écartées et ne compteront pas dans le total final.

2 Création du jeu de cartes

2.1 Fonction de création du jeu

Le jeu de cartes est généré par une fonction `crea_jeu()` qui génère un jeu de 24 cartes grâce au code ci-dessous :

```
def crea_jeu() :
    jeu = []
    oranges = [0 for i in range(4)]
    vertes = [i for i in range(1,11)]
    violettes = [ i*10 for i in vertes if i%2 != 0]
    violettes = violettes * 2
    for i in vertes :
        jeu.append([1,i])
    for i in violettes :
        jeu.append([0,i])
    for i in oranges :
        jeu.append([2,i])
    return jeu
```

2.2 Etude du jeu de cartes

Afin de mieux comprendre comment est construit le jeu de cartes, essayer de répondre aux questions suivantes :

1. Combien y a-t-il de cartes de chacune des couleurs dans le jeu ainsi généré ?
2. Quelle est la valeur des cartes Oranges ?
3. Par quoi est représentée la couleur verte ?
4. Quels sont les valeurs des cartes vertes ? des cartes violettes ?
5. Comment est représenté le 5 Vert dans la liste `jeu` ?
6. Donner le début et la fin de la liste générée par la fonction `crea_jeu()`

3 La Partie de Cartes

3.1 Distribution des cartes.

Le jeu de cartes est distribué aléatoirement entre les deux joueurs. On constitue ainsi deux jeux de cartes qui sont les listes `jeu1` et `jeu2` qui correspondent aux jeux de chacun des joueurs. La fonction `distribue(jeu)` renvoie un tuple contenant les deux listes `jeu1` et `jeu2` :

```
def distribue(jeu) :
    jeu1,jeu2 = [],[]
    while len(jeu) != 0 :
        a = randint(0,len(jeu)-1)
        jeu1.append(jeu[a])
        del jeu[a]
        b = randint(0,len(jeu)-1)
        jeu2.append(jeu[b])
        del jeu[b]
    return(jeu1,jeu2)
```

Les deux liste renvoyées ne sont donc pas classées et contiennent 12 cartes chacune, par exemple on peut avoir :

```
[[0, 70], [1, 8], [1, 7], [2, 0], [0, 30], ...]
```

3.2 Qui gagne le pli ?

Écrire en langage Python une fonction `pli(carte1,carte2)` qui renvoie la carte qui gagne le pli.

— Données : `carte1` et `carte2` sont des listes de la forme `[couleur,valeur]`.
— Résultat : retourne sous forme de liste `[couleur,valeur]` la carte gagnant le pli. Retourne `None` en cas d'égalité

```
def pli(carte1,carte2) :
    """ carte1 et carte2 : listes de la forme [Couleur,Valeur] """
    """ retourne une liste : carte ayant remporte le pli """
    """ retourne None en cas d'egalite """


    return(gagnant)
```

3.3 Elaborer la stratégie de jeu

3.3.1 La stratégie du robot

Vous allez jouer dans un premier temps contre un robot, ce robot ne joue pas ses cartes au hasard mais suit une logique définie par un algorithme, qui est traduit en code python ci-dessous :

```
def jouer_carte2(main) :
""" main est une liste de cartes que peut jouer le robot"""
  liste1 = [i[1] for i in main if i[0]==1]
  liste2 = [i[1] for i in main if i[0]==0]
  for i in main :
    if i[0] == 2 :
      return i
  for i in main :
    if i[0] == 1 and i[1] == max(liste1) :
      return i
  for i in main :
    if i[1] == min(liste2) :
      return i
  return(main[0])
```

3.3.2 Algorithme en langage naturel

Vous allez devoir maintenant écrire une fonction qui choisi la carte qu'il faut jouer en fonction de la main du joueur, c'est à dire en fonction de la liste de cartes que le joueur a en main. **Votre objectif est de battre le robot a tous les coup en sachant que la partie se joue en 1000 donnes**

La stratégie la plus simple consisterai à tirer une carte au hasard dans sa main et à la jouer, dans ce cas l'algorithme s'écrirai de la façon suivante :

Fonction : Jouer une carte Niveau 0

def JouerUneCarte(ListeCartes):

```
/* ListeCarte : liste contenant des cartes de la forme [couleur,valeur] */
```

tirer carte dans ListeCartes au hasard

retourner carte

Mais cette stratégie n'est pas satisfaisante, elle ne permet de battre le robot que une fois sur deux en moyenne (cf. code en annexe). Il existe sûrement une meilleure stratégie que de tirer une carte au hasard, essayer d'imaginer une stratégie,donc un algorithme, qui choisisse une carte dans votre jeu, mais pas au hasard. Écrivez l'algorithme en langage naturel :

Fonction : Jouer une carte Niveau 1

def JouerUneCarte(ListeCartes):

```
/* ListeCarte : liste contenant des cartes de la forme [couleur,valeur] */
/* Décrivez ici la façon dont vous choisissez la carte jouée.           */
.
.
.
```

retourner carte

On peut même aller plus loin : En effet comme l'on connait le jeu de l'adversaire on peut choisir la carte à jouer en fonction du jeu de l'adversaire.

Fonction : Jouer une carte Niveau 2

```
def JouerUneCarte(ListeCartes1,ListeCartes2):
    /* ListeCartes1 : liste de vos cartes de la forme [couleur,valeur]      */
    /* ListeCartes2 : Liste des cartes de votre adversaire                  */
    /* Décrivez ici la façon dont vous choisissez la carte jouée.           */
    .
    .
    .
    retourner carte
```

Attention une règle importante :

Il est interdit dans vos fonctions `JouerUneCarte` de modifier les tableaux passés en paramètre, on doit considérer ces tableaux comme des objet non-mutables, en tout cas dans la fonction, sinon modifier le jeu de son adversaire, c'est de la triche ! ! !

3.3.3 Fonction en Python

Programmer maintenant votre algorithme sous la forme d'une fonction en langage python. Tester votre fonction avec différentes mains que vous générerez vous même.

Tester votre programme sur `Thonny` pour éliminer toute trace d'erreur.. et vérifiez à l'aide d'un jeu de tests que vous élaborerez que fonction renvoie bien la carte que vous désirez jouer en fonction de la main du joueur... Quand tout cela sera terminé, vous serez prêt pour le grand tournoi !

4 Le grand tournoi

4.1 Qualifications : battre le robot

Pour participer au **Grand Tournoi**, vous allez devoir passer par une phase de qualification : pour cela il va falloir que votre stratégie de jeu que vous avez codé dans la fonction `JouerUneCarte` batte le robot à tous les coups.

Implanter votre fonction dans le code donnée en annexe, et faites valider votre qualification par le professeur.

4.2 Phases Finales : Le Grand Tournoi

Vous allez maintenant pouvoir faire jouer votre algorithme de jeu de cartes contre un autre algorithme élaboré par un autre groupe.

Pour que éviter les mauvais tirages, nous allons faire jouer les deux algorithmes 10 fois 1000 fois ce qui constituera dix manches. Le vainqueur sera celui qui remportera le plus de manches..

ANNEXE : le code du jeu de cartes

Il appartient aux deux joueurs de remplacer les fonctions `jouer_carte1` et `jouer_carte2` par leur propres codes. La fonction `JouerUneCarte1` joue une carte de façon aléatoire (la première de la main)

```
from random import randint
#Creation du jeu de cartes
def crea_jeu() :
  jeu = []
  oranges = [0 for i in range(4)]
  vertes = [i for i in range(1,11)]
  violettes = [ i*10 for i in vertes if i%2 != 0]
  violettes = violettes * 2
  for i in vertes :
    jeu.append([1, i])
  for i in violettes :
    jeu.append([0, i])
  for i in oranges :
    jeu.append([2, i])
  return jeu

# Distribution aleatoire du jeu de cartes aux deux joueurs
def distribue(jeu) :
  jeu1,jeu2 = [],[]
  while len(jeu) != 0 :
    a = randint(0,len(jeu)-1)
    jeu1.append(jeu[a])
    del jeu[a]
    b = randint(0,len(jeu)-1)
    jeu2.append(jeu[b])
    del jeu[b]
  return(jeu1,jeu2)

# Retourne la carte gagnante parmi les deux passees en parametres
def pli(carte1,carte2) :
  """ carte1 et carte2 : listes de la forme [Couleur,Valeur] """
  """ retourne une liste : carte ayant remporte le pli """
  """ retourne None en cas d'egalite """
  .
  .
  .
  .
  .
  .
  return (gagnant)

# Fonction de jeu d'une carte par le joueur 1
def jouer_carte1(main) :
  return(main[0])
```

```
# Fonction de jeu d'une carte par le joueur 2
def jouer_carte2(main) :
  liste1 = [i[1] for i in main if i[0]==1]
  liste2 = [i[1] for i in main if i[0]==0]
  for i in main :
    if i[0] == 2 :
      return i
  for i in main :
    if i[0] == 1 and i[1] == max(liste1) :
      return i
  for i in main :
    if i[1] == min(liste2) :
    return i
  return(main[0])

# la partie, renvoie le gagnant.
def partie() :
  joueur1,joueur2 = distribue(crea_jeu())
  score1,score2 = 0,0
  for i in range(12):
    carte_j1 = jouer_carte1(joueur1)
    joueur1.remove(carte_j1)
    carte_j2 = jouer_carte2(joueur2)
    joueur2.remove(carte_j2)
    gagnant = pli(carte_j1,carte_j2)
    if gagnant == carte_j1:
      score1 = score1 + carte_j1[1] + carte_j2[1]
    elif gagnant == carte_j2 :
      score2 = score2 + carte_j1[1] + carte_j2[1]
  if score1 > score2 :
    return 1
  elif score1 < score2 :
    return 2
  else :
    return 0

# programme principal, on joue 1000 parties
g1,g2 = 0,0
for x in range(1000) :
r = partie()
if r == 1 :
g1 += 1
if r == 2 :
g2 += 1
print (g1,' - ',g2)
```