

Rapport de projet

Les tours d’Hanoi

Réalisé par :

ALSABR Ibrahim

Cours :

INF101

# Introduction

Dans ce rapport, on présente le travail effectué dans le cadre du projet de TP. Son objectif c’est de clarifier la méthodologie adoptée pour les différentes parties du projet, en illustrant des parties de codes Python de projet.

# Partie A

Dans cette partie, on a commencé par la fonction ***init(n)*** qui permet de renvoyer la configuration initiale de plateau. A partir de nombre de disques reçu en paramètre, cette fonction crée une liste des entiers classés par ordre décroissant pour la tour de gauche (indice 0 de la liste plateau).

La deuxième fonction ***nombre\_disque(plateau, numtour)*** affiche le nombre de disques qui se trouvent dans la tour renseignée. Le principe c’est de mesurer la taille de la liste correspondante à la tour en utilisant la fonction *len().*

Ensuite, les fonctions ***disque\_superieur(plateau, numtour)*** et ***position\_disque(plateau, numdisque)*** permettent de récupérer le numéro du disque supérieur de la tour (en affichant le dernier élément de la lise (plateau[numtour][-1])) pour la première et le numéro de la tour où se positionne le disque pour la deuxième.

Dans la fonction ***verifier\_deplacement(plateau, nt1, nt2),*** on a utilisé la fonction ***disque\_superieur(plateau, numtour)*** pour vérifier si le déplacement qui a demandé l’utilisateur est possible ou pas, en utilisant la fonction disque\_superieur définie avant.

# Question N°5

def verifier\_deplacement(plateau,nt1,nt2):

  if plateau[nt1]!=[] and plateau[nt2]==[]:

    return True

  elif plateau[nt1]!=[] and plateau[nt2]!=[]:

    if disque\_superieur(plateau,nt2) > disque\_superieur(plateau,nt1):

      return True

    else:

      return False

La dernière fonction vérifie si les disques sont tous bien positionnés dans la troisième tour, ce qui correspond à la situation de la victoire des tours de Hanoi.

# Partie B

Cette partie concerne le graphisme avec Turtle. La première fonction dessine le plateau en fonction de nombre de disques renseignés par l’utilisateur. On a commencé par introduire les mesures de la figure pour le cas d’un seul disque. Ces mesures augmentent en lien avec le nombre n de disques.

La fonction ***dessine\_disque(nd, plateau, n)*** permet de dessiner le disque en se basant sur sa position sur le plateau, donnée par la configuration de ce dernier. ***efface\_disque(nd, plateau, n)*** est basée sur le même principe en dessinant cette fois en blanc par-dessus.

La fonction ***dessine\_config(plateau, n)*** fait appel à la fonction dessine\_disque() pour dessiner chaque disque de plateau afin d’avoir la configuration finale de plateau.

def dissque\_config(plateau,n):

    i=0

    dessine\_plateau(n)

    while i<3:

        a=list(plateau[i])

        j=0

        while j<len(a):

            nd=a[j]

            dessine\_disque(nd,plateau,n)

            j+=1

        i+=1

    return

Pareil pour la fonction ***efface\_tout(plateau, n)*** :

def efface\_tout(plateau,n):

    i=0

    while i<3:

        a=list(plateau[i])

        j=0

        while j<len(a):

            nd=a[j]

            efface\_disque(nd,plateau,n)

            j+=1

        i+=1

    return

# Partie C

Cette partie consiste à interagir avec le joueur. Elle commence par la fonction ***lire\_coords(plateau)*** qui permet au joueur d’entrer le déplacement qu’il souhaite effectuer sur le plateau et vérifier au même temps s’il est possible. Dans le cas contraire, il redemande à l’utilisateur d’entrer des coordonnées valides.

La fonction jouer\_un\_coup(plateau, n) effectue ce déplacement, en utilisant les fonction dessine et efface disque définies dans la partie B, en plus de la fonction disque\_configuration pour tracer le plateau et les autres disques :

def jouer\_un\_coup(plateau,n):

  dissque\_config(plateau,n)

  ls= lire\_coords(plateau)

  if(ls[0]!=ls[1]) :

        a=plateau[ls[0]]

        nd = a[-1]

        efface\_disque(nd,plateau,n)

        plateau[ls[1]].append(a[-1])

        plateau[ls[0]].remove(a[-1])

        b = plateau[ls[1]]

        nd = b[-1]

        dessine\_disque(nd,plateau,n)

  else:

         print("Vous avez quitté le jeu")

         plateau = [[],[],[]]

  return plateau

La fonction ***boucle\_jeu(plateau, n)*** regroupe les deux premières fonction pour permettre à l’utilisateur de jouer jusqu’à ce qu’il gagne, en vérifiant à chaque fois s’il atteint la configuration finale avec la fonction verifier\_victoire(). On a également fixé un nombre maximal de coups à jouer, et on a donné à l’utilisateur le droit de quitter la partie en saisissant la valeur -1 comme tour de départ.

En fin, dans le programme principale on demande au joueur d’entrer le nombre de disques, pour lancer après la boucle de jeu avec la fonction boucle jeu.

# Partie D

L’objectif de cette partie c’est de permettre au joueur d’annuler son dernier coup et de revenir en arrière. Pour cela on a utilisé un dictionnaire qui enregistre à chaque fois la configuration du plateau après chaque coup.

Le problème qu’on a rencontré dans cette partie c’est par rapport à l’ajout des éléments dans le dictionnaire. En fait, quand on rajoute une nouvelle valeur on perd les anciennes parce qu’ils se remplacent toutes par la nouvelle valeur.

Après chaque coup, on demande au joueur s’il voulait annuler son dernier coup. Pour cela, on a modifié la fonction boucle\_jeu pour qu’il autorise l’annulation du dernier coup.

def annuler\_dernier\_coup(coups , n\_dernier) :

    ls = dernier\_coup(coups , n\_dernier)

    plateau = [coups[n\_dernier]['Tour\_1'], coups[n\_dernier]['Tour\_2'], coups[n\_dernier]['Tour\_3']]

    a=plateau[ls[1]]

    n = len(plateau[0]) + len(plateau[1]) + len(plateau[2])

    nd = a[-1]

    efface\_disque(nd,plateau,n)

    plateau[ls[0]].append(a[-1])

    plateau[ls[1]].remove(a[-1])

    b = plateau[ls[0]]

    nd = b[-1]

    dessine\_disque(nd,plateau,n)

    del coups[n\_dernier]['Tour\_1']

    del coups[n\_dernier]['Tour\_2']

    del coups[n\_dernier]['Tour\_3']

    return coups

# Partie E

L’objectif de cette partie est d’enregistrer les scores des joueurs dans un fichier, pour pouvoir les comparer par la suite.

Tout d’abord on a commencé par écrire une fonction qui enregistre le score du joueur en tenant compte de 3 informations dans un fichier de scores :

* Nom du joueur
* Nombre de disques
* Nombre de coups

On a choisi de sauvegarder ces données dans un fichier binaire de type Pickle. Donc, au niveau du programme principal ***Programme.py***, on a importé le module Pickle :

import pickle

A la fin de la partie du joueur, on met à jour le dictionnaire des scores par les informations du joueur.

        nom = str(input("Ecrire votre nom :"))

        dictionnaire\_score[nom] = {}

        dictionnaire\_score[nom]["nb\_disque"] = n

        dictionnaire\_score[nom]["nb\_coups"] = donnees[0]

        dictionnaire\_score[nom]["time"] = time.time() - start\_time

On crée le fichier binaire et on appelle la méthode dump pour écrire dans le fichier Scores\_Des\_Joueurs.pickle les informations du joueur

 # creation du fichier Pickle

        pickle\_out = open("Scores\_Des\_Joueurs.pickle","wb")

        pickle.dump(dictionnaire\_score, pickle\_out)

        pickle\_out.close()

Une fois qu’on a recueilli les informations de tous les joueurs, on lit le fichier binaire de type Pickle et on appelle la fonction afficher\_meilleurs\_scores.

#Lecture du fichier

pickle\_in = open("Scores\_Des\_Joueurs.pickle","rb")

example\_dictio = pickle.load(pickle\_in)

# Meilleurs scores (nombre de coups)

afficher\_meilleurs\_scores(example\_dictio)

Voici l’implémentation de cette fonction.

def afficher\_meilleurs\_scores(dict\_des\_resultats) :

    #print (dict\_des\_resultats)

    dic\_meilleurs\_scores = {}

    for key, value in dict\_des\_resultats.items():

        for cles,valeurs in value.items():

            if(cles == "nb\_coups") :

                dic\_meilleurs\_scores[key] = valeurs

    print(sorted(dic\_meilleurs\_scores.items(), key = lambda kv:(kv[1], kv[0])))

    return

On crée un dictionnaire des meilleurs scores contenant les noms des participants (Clé) et leurs nombre de coups respectifs (Valeur) et on utilise la méthode sorted pour trier ce dictionnaire dans l’ordre croissant.

# Partie F

L’objectif de cette est de créer une fonction capable de trouver la liste optimale des mouvements à jouer pour déplacer n disques de la tour de départ à la tour d’arrivée.

On a créé une fonction récursive Tour Hanoi ayant comme paramètres le nombre de disques, la tour de départ, la tour d’arrivée et la tour auxiliaire.

def Tour\_Hanoi(n , tour\_depart, tour\_arrive, tour\_auxiliare):

    if n == 1:

        print ("Deplacer le disque 1 de la tour",tour\_depart,"a la tour",tour\_arrive)

return

    Tour\_Hanoi(n-1, tour\_depart, tour\_auxiliare, tour\_arrive)

    print ("Deplacer le disque",n,"de la tour",tour\_depart,"a la tour",tour\_arrive )

    Tour\_Hanoi(n-1, tour\_auxiliare, tour\_arrive, tour\_depart)