TRAIN MEXICAIN – PROJET ALGO3 2022

Faculté des Sciences Jean Perrin
Auteur : TEBNI WAJIH

UNIVERSITÉ D'ARTOIS

Les différentes classes :

Dans ce projet, j'ai utilisé 4 classes différentes que j'ai importé dans le document principal "Domino" :

1. **Class Domino**
2. **Class Pile**
3. **Class Jeu mexicain**
4. **Class Joueur**

Tout d'abord, j'ai commencé par écrire le code de la classe "Domino" où j'ai implémenté :

Class Domino :

1. Le constructeur __init__ :

-> le constructeur prend en paramètre deux entiers qui représentent les valeurs des points présents sur les deux faces du domino

2. La méthode str :

-> Cette méthode ne nécessite aucun paramètre, elle affiche le domino et les points présents sur ses deux faces (initialisés précédemment)

3. La méthode valeur :

-> Cette méthode ne nécessite aucun paramètre, elle permet de calculer la somme des deux faces du domino

4. La méthode estDouble :

-> Cette méthode ne nécessite aucun paramètre, elle teste si un domino est double.
Un domino est dit double si les deux faces sont égales.

5. La méthode estVide :

-> Cette méthode ne nécessite aucun paramètre, elle teste si le domino est vide.
Un domino est dit vide si les deux faces sont vides (égales à -1)

6. La méthode peutEtrePlaceApres :

-> Cette méthode prend en paramètre un deuxième domino, et teste si ce domino peut être placé après celui qui le précède.
Un domino peut être placé après un autre si l'une de ses faces est égale à la dernière face du dernier domino.

7. La méthode permute :

-> Cette méthode ne nécessite aucun paramètre, elle permet d'inverser les deux faces d'un domino.

8. La méthode egal :

-> Cette méthode prend en paramètre un deuxième domino, elle teste si les deux dominos sont égaux.
Deux dominos sont égaux si et seulement si leurs deux faces sont égales.

9. La méthode estPlusGrand :

-> Cette méthode prend en paramètre un deuxième domino et fait appel à la méthode valeur() pour tester si un domino est plus grand qu'un autre.
Un domino est plus grand qu'un autre si sa valeur (la somme de ses deux faces) est supérieure à celle du deuxième.

Ensuite, j'ai créé la classe "Pile" étudiée en cours Piles & Files :

Class Pile :

1. Le constructeur __init__ :

-> Ce constructeur permet de créer une pile et initialiser le nombre des éléments à nul.

Troisièmement, j'ai créé la classe Jeu mexicain où j'ai importé les deux classes (Pile et Domino) :

Class Jeumexicain :

1. Le constructeur __init__ :

-> On initialise :
_Un domino de départ qui est un domino vide (-1, -1).
_Un train mexicain qui est une pile.
_Une pioche qui est une liste vide et qui sera remplie au fur et à mesure par tous les dominos possibles en créant une boucle qui fait appel à constructeur de la classe Domino à chaque itération.
_Les joueurs qui est une liste vide.
_Le joueur actuel qui est initialisé à zéro.

2. **La méthode ajouterjoueur :**

-> Un joueur est passé en paramètre dans cette méthode, elle permet d'ajouter ce joueur dans la liste des joueurs.

3. **La méthode joueursuivant :**

-> Cette méthode n'exige aucun paramètre, elle permet de changer le joueur actuel à chaque tour de jeu en évitant le débordement hors de la liste des joueurs. Par exemple, si le dernier joueur qui a joué est le quatrième (d'indice 3 dans la liste des joueurs), le tour suivant sera au premier joueur de la liste (qui a pour indice 0).

4. **La méthode joueurprecedent :**

-> Cette méthode n'exige aucun paramètre, elle permet de déterminer le dernier joueur qui a joué avant le joueur actuel.

5. **La méthode choisirjoueur :**

-> Cette méthode ne demande aucun paramètre, elle permet de choisir un joueur aléatoire parmi les joueurs dans la liste. Grâce à la méthode random.choice(param) on peut choisir aléatoirement un élément d'un objet passé en paramètre de type liste, tuple etc...

6. **La méthode piocheVide :**

-> Cette méthode permet de vérifier si la pioche est vide.

Dernièrement les deux méthodes jouer et jouersimple qui en faisant appel à différentes méthodes précédemment définies et bien expliquées font dérouler le jeu entre les différents joueurs. A chaque tour fini, on passe au joueur suivant et on lui demande de piocher un domino.

Finalement, j'ai créé la classe Joueur où j'ai importé les autres classes (Pile, Domino et Jeu mexicain) :

Class Joueur :

1. Le constructeur __init__ :

-> Cette méthode permet de créer la réserve des joueurs et le train mexicain. Elle prend en paramètre une pioche.

2. La méthode plusGrandDominoDouble :

-> Aucun paramètre n'est demandé par cette méthode, elle parcourt la réserve des dominos des joueurs et retourne le plus grand domino double dedans. Elle fait appel aux deux méthodes de la classe "Domino" : estDouble et estPlusGrand .
On crée un domino vide (ses deux faces valent -1) puis on parcourt la réserve, s'il existe un domino double ET plus grand que lui on l'attribue ses valeurs.

3. La méthode premierDominoSurLeT :

-> Cette méthode exige le domino de départ en paramètre. Elle vérifie si le train n'est pas vide (si la pile n'est pas vide) elle retourne le sommet de la pile qui est le dernier domino du train. Sinon, elle parcourt la réserve en testant si le domino passé en paramètre peut être posé sur le train du joueur.
Comment le domino peut être posé est expliqué avant.

4. La méthode premierDominoSurLeTM :

-> Même principe du fonctionnement de la méthode précédente sauf qu'elle exige deux paramètres : le domino de départ et le train mexicain.

5. La méthode nombreDominosReserve :

-> Cette méthode n'exige aucun paramètre, elle retourne le nombre des dominos de la réserve du Joueur. La réserve est définie de type liste dans le constructeur.

6. La méthode valeurReserve :

-> Aucun paramètre n'est demandé par cette méthode, elle parcourt tous les dominos, elle fait appel à la méthode "valeur" de la classe Domino pour récupérer la

valeur de chaque domino et les additionne au fur et à mesure pour retourner au final la valeur de tous les dominos de la réserve.

7. **La méthode piocherDomino :**

-> Une pioche est passée en paramètre dans cette méthode, elle permet de piocher un domino et le rajouter à la réserve.

Dans cette partie, j'explique le programme principal qui est "Domino" :

Le "main" s'occupe de l'interface graphique du jeu à l'aide de "tkinter".
En premier temps, j'ai importé la librairie "tkinter", ensuite les classes précédemment construites et la librairie "random".
Ensuite, dans la partie d'initialisation et création du jeu, j'ai créé les joueurs et j'ai initialisé les fonctions. J'ai créé aussi deux variables : "partiefinie" et "gagnant" qui seront initialisés à False et 0.

On passe à la fonction principale du jeu "jouer" qui permet de dérouler le jeu, ce que le programme exécute à chaque clic sur un bouton "GO !".
Le principe de cette fonction se base sur le test : si la partie est finie ou non, si c'est vrai on appelle la fonction "donothing()" qui affiche une

nouvelle fenêtre en mentionnant la fin de la partie et le joueur gagnant sinon en déroule le jeu en appelant les fonctions définies après.

Les fonctions d'affichage :

1. **La fonction "affichedom" :**

Cette fonction permet l'affichage du domino de départ. Avec deux variables globales :

a. **PhotoD** : la photo du domino de départ. Puisqu'il sera affiché pendant le déroulement du jeu, on importe son image du fichier image fournie avec le projet sur Moodle.
b. **jMexicain** : qui sert à définir le domino dans le code de cette fonction.

2. **La fonction "affichetr" :**

Cette fonction permet d'afficher les trains des joueurs et le train mexicain (Le premier frame dans la fenêtre Tkinter).
Je déclare deux variables globales :

a. Photo : la liste des photos des trains.
b. jMexicain : les trains des joueurs.

3. **La fonction "afficheres" :**

Cette fonction est responsable à l'affichage de la réserve des dominos (dans le deuxième frame dans la fenêtre Tkinter).
Je déclare deux variables globales que je vais utiliser plus tard

a. photoR : la liste des photos de la réserve
b. Maurice : la réserve des trains.

4. **La fonction "arretelejeu" :**

Cette fonction ouvre la fenêtre de fin du jeu quand la partie est finie. Elle annonce le vainqueur du jeu et avec un bouton “Quitter !”, elle permet de détruire les deux fenêtres et sortir du jeu.

5. **La fonction “partiefinie” :**

Cette fonction indique si la partie est finie ou non.
La définition de la fin de la partie dans le projet sur Moodle **: La partie prend fin lorsqu’un joueur dépose son dernier domino ou lorsque le jeu est bloqué parce que la pioche est vide, et que plus aucun joueur ne peut poser de dominos. Dans ce cas le gagnant est le joueur dont la valeur des wagons restants est la moins élevée (points figurant sur les dominos).**

6. **La fonction “queljoueur” :**

Elle prend en paramètre le joueur et le retourne en chaîne de caractères. J’ai essayé la méthode prédéfinie “str(param)” qui rend en chaîne de caractères mais elle ne permet pas de retourner le nom du joueur.

La construction des frames et des Canvas :

L’initialisation des images des domino :

1. **Photo** : la liste des images sur les trains.
2. **PhotoR** : la liste des images des dominos de la réserve.
3. **PhotoD** : L’image du domino de départ.

Frame 1 : Le domino de départ, les noms des joueurs et leurs trains.
Le domino de départ est un Canvas placé sur la 1ère ligne et la 3éme colonne, de fond blanc.

La partie des trains est un Canvas placé sur la 2ème ligne et la 2éme colonne avec un ensemble de 5 lignes groupées, de fond blanc.

Frame 2 : La réserve
Cette dernière est un Canvas placé sur la 4éme ligne et la 3éme colonne avec un fond blanc.

Frame 3 : Bouton de jeu GO !
Le bouton exécute la fonction “jouer” à chaque clic de souris.

Tous les Frames sont placés grâce à : “frame#.pack(side=TOP)”, en suivant l’ordre les frames sont l’un au-dessus de l’autre.

Finalement, l’initialisation du jeu en appelant les trois fonctions d’affichage : “afficheres”, ”affichetr” et “affichedom”, et, le lancement de la fenêtre tkinter.