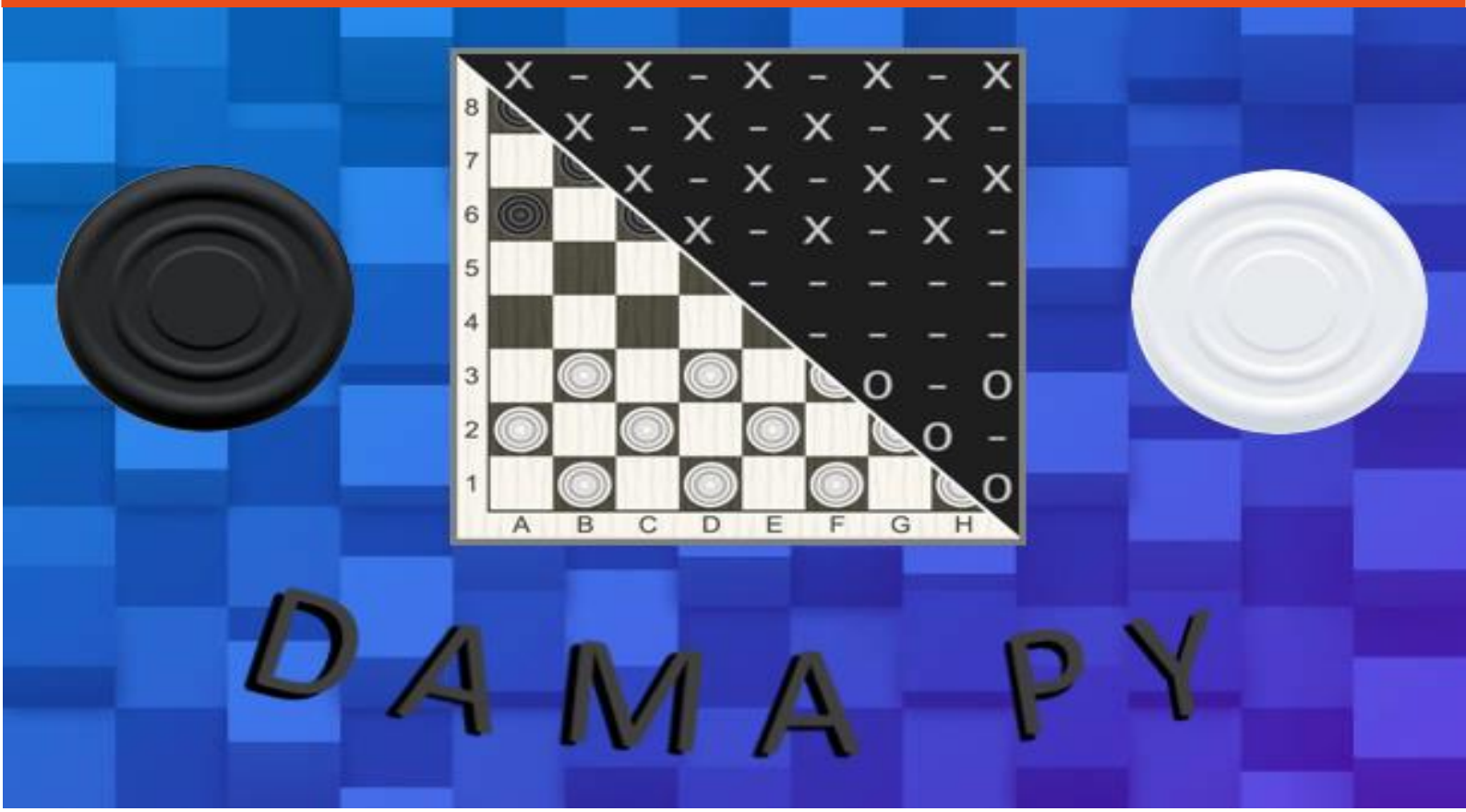
LES TROPHÉES NSI

Édition 2023

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET



NOM DU PROJET : DamaPy

- une version modernisée du jeu de dames en Python

> PRÉSENTATION GÉNÉRALE :

Tout d'abord, l'idée de créer un jeu de dames nous est venue du fait que nous voulions un projet qui nous permette de nous améliorer le plus possible en programmation. En effet, lorsque nous avons eu l'idée d'un jeu de dames, nous n'avions pas vraiment d'idée de la manière dont nous pouvions le réaliser. C'est justement le fait de devoir expérimenter et effectuer des recherches notamment sur les règles des dames dans le but de voir le projet grandir petit à petit qui nous a motivés. Une autre chose qui était importante pour nous dans le choix du projet était que nous voulions qu'il puisse être amélioré en fonction de notre avancement. L'avantage du jeu de dames est qu'il possède de nombreuses règles à inclure ou non en fonction de l'avancement du projet, ce qui a permis de garantir une grande modularité.

Par le choix de ce projet, nous avons voulu recréer en Python un jeu que nous aimons particulièrement. Nous avions un résultat en tête et le but que nous nous sommes fixés était de s'en rapprocher au maximum. Nous voulions un jeu qui puisse mettre en relation et faire s'affronter deux joueurs l'un contre l'autre. Ce qui nous a fait nous diriger vers le jeu de dames, c'est aussi le fait qu'il est très complet : il nous a permis notamment de beaucoup travailler sur les listes et leur manipulation. Nous avons pris beaucoup de plaisirs à travailler sur ce projet et à le voir évoluer jusqu'à son aboutissement.

> ORGANISATION DU TRAVAIL :

Lorsque nous programmions quelque chose de nouveau chez nous, chacun écrivait son code que nous mettions en commun une fois en groupe dans le but de prendre le meilleur de chacun pour avoir un programme le plus complet, propre et optimisé possible. Malgré le fait que nous avons tous endossé chacun des rôles dans le projet (code, recherche ...), nous nous sommes chacun spécialisés dans les tâches qui nous plaisaient le plus et où nous sommes les plus efficaces.

TOMASETTO Pierre : Programmation des parties compliquées du projet + vérification du bon fonctionnement du programme et correction des différents bugs et dysfonctionnements

STEINER Éric : Recherches sur la manière d'optimiser le programme (raccourcir et simplifier certaines fonctions, gestion de l'interface homme-machine ...)

RIEU-CARMONA Michel : Recherche de nouvelles idées à implémenter et rédaction de la documentation du programme.

Lorsque nous travaillons en classe, nous séparions notre groupe de trois en deux parties. D'un côté, deux d'entre nous travaillaient sur le programme (avancement du code par exemple) pendant que le troisième s'occupait la plupart du temps d'effectuer des recherches sur certaines règles ou fonctionnalités des dames ou encore par exemple sur certaines bibliothèques Python à implémenter. Grâce à cela, nous avons pu intégrer certaines fonctionnalités auxquelles nous n'aurions pas pensé aux premiers abords. Nous avons effectué cette séparation car nous avons vite remarqué lors des séances que nous étions moins efficaces en travaillant à trois en même temps sur le code lorsque nous étions réunis.

En plus des deux heures par semaine dans lesquelles nous avancions sur le projet en classe, nous avons beaucoup travaillé sur ce dernier chez nous ou dans d'autres endroits où nous pouvions nous retrouver pour mettre en commun ce que nous avions fait chacun de notre côté (code, documentation du projet …). Pour faciliter et rendre plus efficaces les séances où travaillons ensemble, nous échangions beaucoup par mail. Nous déposions également à chaque fois les morceaux de code que nous faisions sur une plateforme de stockage en ligne pour que chacun y ait accès facilement en plus du programme entier dont chacun avait un exemplaire sur clef USB.

LES ÉTAPES DU PROJET :

Nous avons réussi à réaliser l'entièreté du projet nous-mêmes, c'est-à-dire que nous avons rédigé l'entièreté du code du début jusqu'à la fin (en passant par la documentation du projet ainsi que la réalisation d'une interface homme-machine la plus agréable possible par exemple) et nous en sommes très fiers puisque c'était très important pour nous de créer un projet qui nous corresponde vraiment jusqu'au bout. Puisque nous n'avions au départ aucune idée de la manière dont nous allions créer ce jeu de dame, nous avons procédé par étapes, pour améliorer le projet petit à petit :

Etape du projet	Description	Précisions techniques
Création du plateau de jeu	Création d'une liste qui comporte dix autres listes. Ces dernières comportent dix cases qui sont représentées par des tirets	Voir l'annexe 1.1 pour l'illustration
Ajout des pions des joueurs	On remplit une case sur deux avec les pions de chaque joueur un en haut (pions « X ») et un en bas (pions « O »)	Voir l'annexe 1.2 pour l'illustration
Possibilité de se déplacer et de manger des pions	Comme dans un vrai jeu de dames les déplacements des pions sont limités (impossibilité de se déplacer en arrière). Cependant, le joueur peut manger un pion adverse dans n'importe quelle direction tant qu'il y a un pion adverse sur une case mitoyenne.	Pour tout changement de position d'un pion, on commence par supprimer le pion de sa position initiale, avant de le faire apparaître sur la case d'arrivée
Prise en compte des erreurs de l'utilisateur et potentielles erreurs	Le but est d'empêcher les dysfonctionnements et les bugs du programme lié à une mauvaise utilisation de l'utilisateur (sélection d'une case vide, collisions entre les pions, définition des limites du plateau …).	Chaque type de vérification est contenue dans une fonction prévue à cet effet. Ces vérifications représentent une grande partie du code et ont été l'une des parties les plus longues à réaliser.
Détection de la fin d'une partie	Le programme détecte tout seul si la partie en cours est terminée. Lorsqu'un des joueurs n'a plus de pions, le programme s'arrête automatiquement.	Le programme vérifie sur chaque case du plateau à la fin du tour de chaque joueur si ce dernier a au moins un pion restant.
Intégration des dames	Les dames ont un fonctionnement identique à celles du jeu traditionnel : elles peuvent manger et se déplacer comme elles le souhaitent. Lorsqu'il les déplace, le joueur peut donc choisir librement la direction et la distance du déplacement	Le programme détecte tout seul si un pion atteint la dernière ligne dans le camp adverse puis transforme automatiquement le pion en dame

Ajout de la prise obligatoire	Pour inclure cette règle du jeu traditionnel, nous avons fait en sorte que le programme détecte tout seul s'il y a un pion adverse à manger sur le plateau. Il va donc manger tous les pions qu'il trouve sur son chemin jusqu'à ce qu'il n'en trouve plus. Cela nous a permis de rendre le jeu plus fluide et agréable à prendre en main	Au début de chaque tour, le programme va regarder pour chaque pion dans toutes les directions s'il y a un pion à manger. Le joueur pourra donc se déplacer uniquement si le programme ne trouve aucun pion à manger
Amélioration de la prise obligatoire des dames	Dans le but de rajouter un peu de complexité au programme, nous avons fait en sorte que les dames mangent automatiquement le pion le plus proche lorsque le programme détecte une prise obligatoire et que deux pions différents peuvent être mangés	Si le programme détecte deux pions (qui peuvent être mangés) sur la trajectoire d'une même dame, il va comparer la distance qui sépare chacun d'eux de la dame et va décider tout seul lequel manger
Création d'un mode d'apprentissage	Nous voulions créer un mode de jeu qui serve de démonstration rapide de notre projet. Ce mode de jeu à un joueur permet (en plus d'introduire très brièvement au jeu de dames) de montrer rapidement et facilement certaines fonctionnalités que nous avons incluses	Les coups de l'ordinateur ainsi que du joueur sont programmés à l'avance. Le joueur joue quand même lors de son tour, mais son coup ne sera comptabilité uniquement s'il respecte la saisie attendue (sinon il rejoue son tour).

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

Pour vérifier l'absence de bugs dans notre programme, nous avons créé un plateau qui avait pour but de tester les nouvelles fonctionnalités. Pour cela, nous devions juste modifier le plateau (ce que contiennent les listes) en plaçant les différents pions de sorte à pouvoir créer une situation nous permettant de tester la fonctionnalité que nous avions réalisé. Par exemple, pour tester les collisions entre les pions et les bordures du plateau, nous placions directement des pions à côté des bordures. Bien sûr, pour nous assurer du bon fonctionnement du programme, nous avons fait régulièrement des parties complètes pour essayer de trouver puis de corriger les problèmes qui auraient pu nous échapper des vérifications avec le plateau prévu à cet effet.

Évidemment, comme dans tout projet, nous avons rencontré des difficultés lors de l'ajout de certaines fonctionnalités. En plus de la difficulté à corriger les dysfonctionnements que nous avons rencontrés, nous avions souvent du mal à les détecter puisque certaines erreurs ne se produisaient que dans certains cas particuliers et très rares ce qui les rendaient assez difficiles à repérer.

Certaines fonctionnalités ont aussi été particulièrement longues à mettre en place puisqu'elles ont demandé beaucoup de réflexion sur la manière dont nous pouvions les mettre en place. Par exemple, la fonctionnalité qui nous a demandé le plus de temps et de travail a été la mise en place de la prise obligatoire et automatique des pions. En effet, pour la prise obligatoire, il a fallu faire en sorte que le programme détecte automatiquement les pions à manger. Nous avons donc mis en place une fonction qui regarde pour chaque case du plateau s'il y a un pion du joueur dont c'est le tour et un pion adverse à manger. Une fois cela fait, le plus dur était à venir puisque nous avons dû faire en sorte d'actualiser les informations du pion du joueur (coordonnées, nombre de pions qui ont déjà été mangés ...) à chaque prise. Pour cela, à chaque fois qu'on utilise la fonction pour manger, on renvoie les nouvelles coordonnées du pion (la ligne et la colonne de départ additionnées ou soustraites à la valeur du déplacement en fonction de la direction du déplacement : cf. Annexe 2).

Pour finir, il a fallu faire en sorte que le programme puisse donner la possibilité au joueur de se déplacer si aucun pion n'a été mangé durant la phase de détection par le programme. Nous avons donc eu l'idée d'ajouter un compteur qui s'incrémente à chaque fois qu'on mange un pion adverse (que l'on utilise aussi pour indiquer au joueur le nombre de pions qu'il a mangé lors de son tour) qui nous sert à vérifier si on donne la possibilité de se déplacer ou non (cf. annexe 3). Pour cela, le programme regarde la valeur de la variable « nbre_pions_manges » et ne donne le tour au joueur uniquement si elle n'est pas de 0.

> OUVERTURE :

Toujours dans l'idée de se rapprocher le plus possible d'un vrai jeu de dames et de rendre le jeu plus dynamique, nous aimerions intégrer quelques règles supplémentaires pour finir le projet. Sans rentrer dans les détails, nous pensons particulièrement à mettre en place en fonction qui, dans la mesure où un pion a plusieurs directions où manger, choisit automatiquement sur la diagonale qui entraîne le plus de prises à la fin du tour.

Enfin, même si cela nous parait assez long et périlleux à mettre en place, nous aimerions mettre en place une sorte de petite IA (très simple et sommaire) qui pourrait jouer au jeu et servir de deuxième joueur. Cette idée nous plait et nous motive beaucoup puisqu'elle nous permettrait de grandement nous améliorer en programmation sur un sujet qui nous passionne et que nous avons encore peu abordé.

Il est important pour nous de mettre en avant notre projet ! En effet, dans le but de récolter des retours et des critiques sur notre projet en vue de l'améliorer ainsi que de présenter notre travail au plus grand nombre, nous avons essayé de trouver le plus de moyens possibles de partager notre travail au plus grand nombre. Par exemple, nous l'avons présenté lors de la semaine des sciences de notre lycée et nous avons particulièrement aimé en faire la démonstration et expliquer son fonctionnement (pour peut-être essayer de donner envie à certains de se mettre à la programmation en partageant notre passion pour l'informatique).

Pour finir, si nous devions refaire ce projet nous changerions certaines choses. En effet, nous trouvons que l'interface d'utilisation du programme est assez floue et un peu compliquée à prendre en main (malgré le fait que nous avons beaucoup travaillé sur l'interface homme-machine). De plus, nous trouvons qu'il serait intéressant de revoir la fluidité du jeu car le rythme de jeu (surtout dans les parties classiques) est assez lent et il est donc assez long et lassant de terminer une partie. Nous serions intéressés de refaire ce projet en corrigeant cette partie, par exemple en créant une interface graphique qui permettrait de pousser jusqu'au bout notre projet afin de créer le jeu le plus agréable et le plus réaliste possible.

En dehors de la prise en main de notre jeu, si nous avions à recommencer ce projet depuis le début, nous ferions attention de bien tester le projet au fur et à mesure lors de la mise en place de certaines fonctionnalités importantes pour gagner du temps en évitant les longues phases de débogage (avec des dysfonctionnements qui se fondent dans la masse du code).

DOCUMENTATION

Avant toute chose, l'utilisation de notre programme ne nécessite aucun pré-requis de lancement. C'est-à-dire qu'il peut être lancé sans installer de bibliothèques (puisque nous avons seulement utilisé « time » qui est inclue dans python).

Notre programme comporte cinq fichiers : « main », « action joueur », « pions », « dames » et « plateau » qui sont contenus dans le dossier « source » à l'intérieur du dossier technique.

Pour utiliser notre projet, il faut tout simplement lancer le fichier « main ». Les autres fichiers sont importés dans le fichier principal et contiennent toutes les fonctions du programme qui sont organisées en fonction de leur utilité (pour les pions, pour les dames, pour le plateau et pour les joueurs en général). Il faut donc faire attention à ce que tous les fichiers soient présents sans quoi le programme ne pourra pas fonctionner correctement.

Il y a trois modes de jeux :

- Le mode « classique » qui permet de jouer aux dames comme dans le jeu original.
- Le mode « apprentissage » qui permet de réaliser une courte partie avec des coups restreints pour le joueur et prévus à l'avance pour l'ordinateur dans le but de tester notre programme de manière plus simple et rapide.
- Le mode « test » qui permet (en modifiant le plateau de test à l'intérieur de la fonction « plateau_test » dans le fichier « plateau ») de créer des parties personnalisées pour par exemple tester certaines fonctionnalités ou réaliser des parties plus courtes que dans le mode classique.

Pour essayer notre programme, nous conseillons d'abord de faire le mode apprentissage (qui est un bon moyen de découvrir notre projet comme dans la démonstration vidéo). Ensuite, pour éventuellement essayer le programme de manière plus poussée, il est possible de créer des parties personnalisées avec le mode test ou d'utiliser le mode classique, même si nous le trouvons moins pratique puisqu'il est assez long et fastidieux (surtout pour accéder aux parties intéressantes du programme).

Lors de la modification du plateau de test (annexe 4), il est important de respecter certaines choses pour éviter d'entraîner des erreurs lors de la partie. Il faut en effet faire attention à utiliser les bon symboles (qui représentent les pions). Le premier joueur a les pions « X » et les dames « V » et le deuxième a les pions « O » et les dames « L » sachant que les lettres doivent être en majuscule. Il faut les insérer à la place des tirets sur le plateau et il faut impérativement utiliser les bons signes sinon le programme ne les reconnait pas et refuse de les utiliser.

Au lancement du programme, il faut choisir le mode de jeu entre les trois disponibles. Une fois cela fait, le programme va lancer le mode sélectionné (grâce à deux fonctions contiennent chacune un mode de jeu : cf. annexes 5.1 et 5.2). Le programme va commencer par afficher les règles du jeu et des précisions sur la manière dont le programme fonctionne (qui sont importantes de lire).

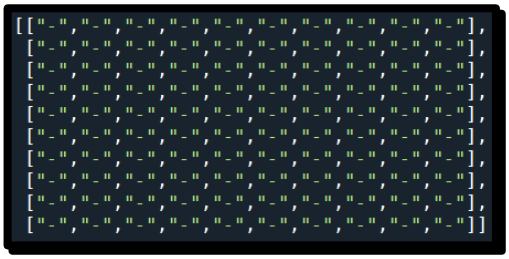
Pour le mode apprentissage, le programme va demander au joueur de jouer (avec un coup prédéfini annoncé par le programme). Pour jouer, il suffit de jouer le coup prévu par le programme (qui est affiché dans la console avant de jouer). Cependant, si le coup entré par le joueur n'est pas le bon, le programme va redemander au joueur de jouer son coup jusqu'à ce qu'il soit correct (grâce à une fonction qui vérifie la validité des coups : cf. annexe 6). Les coups alternent entre le joueur et l'ordinateur avec des explications (à lire) à chaque tour jusqu'à la fin de la partie. A savoir que même si les coups sont restreints, le fonctionnement reste le même que dans le mode classique, c'est-à-dire que les fonctions utilisées sont identiques : on utilise simplement une fonction en plus pour s'assurer de la validité des coups).

Dans le mode classique, les coups ne sont bien sûr pas limités. Chaque joueur joue à son tour (en sélectionnant la ligne et la colonne du pion qu'il veut jouer ainsi que la diagonale sur laquelle il souhaite se déplacer et le nombre de cases à franchir si le joueur sélectionne une dame). Chacun joue donc son tour jusqu'à ce qu'il ne reste plus de pions sur le plateau.

Pour finir, le mode test utilise tout simplement la même fonction (et donc la même boucle) que pour le mode classique (annexe 5.1) avec un plateau différent qui est donc choisi par l'utilisateur au préalable.

Notre programme a été réalisé à l'aide du langage Python. De plus nous avons utilisé une bibliothèque : « time » qui nous a permis de marquer des pauses dans l'exécution du programme pour le rendre plus compréhensible. Elle nous a servi à ajouter une pause entre chaque affichage de plateau lorsque le programme procède à une prise automatique.

Annexe :



Annexe 1.1 : Plateau de jeu vide

Annexe 1.2 : Plateau de jeu rempli

```python
def deplacement_pion(joueur: str, ligne: int, colonne: int, diagonale: int) :
    plateau[ligne][colonne] = "-"
    if diagonale == 1 :
        plateau[ligne-1][colonne-1] = joueur
        afficher()
        return ligne-1, colonne-1
    elif diagonale == 2 :
        plateau[ligne-1][colonne+1] = joueur
        afficher()
        return ligne-1, colonne+1
    elif diagonale == 3 :
        plateau[ligne+1][colonne-1] = joueur
        afficher()
        return ligne+1, colonne-1
    else :
        plateau[ligne+1][colonne+1] = joueur
        afficher()
        return ligne+1, colonne+1
```

Annexe 2 : Exemple de fonction de déplacement qui actualise les coordonnées du pion

```python
def action_joueur_0() :
    nbre_pions_manges = prise_obligatoire("O", "L", "X", "V")
    if nbre_pions_manges == 0 :
        deplacement_joueur_0()
    else :
        print("durant votre tour, vous avez mangé", nbre_pions_manges, "pions\n")
```

Annexe 3 : Zoom sur la variable « nbre_pions_manges »

```python
def plateau_test() -> list :
    return [["-","-","-","-","-","-","-","-","-","-"],
            ["-","-","-","-","-","-","-","-","-","-"],
            ["-","-","-","-","-","-","-","-","-","-"],
            ["-","-","-","-","-","-","-","-","-","-"],
            ["-","-","-","X","V","-","O","L","-","-"],
            ["-","-","-","-","-","-","-","-","-","-"],
            ["-","-","-","-","-","-","-","-","-","-"],
            ["-","-","-","-","-","-","-","-","-","-"],
            ["-","-","-","-","-","-","-","-","-","-"],
            ["-","-","-","-","-","-","-","-","-","-"]]
```

Annexe 4 : fonction à modifier avec les différentes lettres qui représentent les pions

```
def mode_classique() :
    while True :
        # Tour du joueur X
        if fin_de_jeu("X") == True :
            print("La partie est terminÃ©e : victoire du joueur 0")
            break
        else :
            print("Au tour du joueur X (V) \n")
            action_joueur_X()
            # On vérifie si une dame peut-être créée et si oui, on la crée
            creation_dames("X")
        # Tour du joueur 0
        if fin_de_jeu("O") == True :
            print("La partie est terminÃ©e : victoire du joueur X")
            break
        else :
            print("Au tour du joueur O (L) \n")
            action_joueur_O()
            creation_dames("O")
```

Annexe 5.1 : Mode de jeu classique et test

```
def mode_apprentissage() :
    print("Le contenu des print a été supprimé par soucil de compréhension")
    input()
    print()
    action_joueur_apprentissage(1, 1, 4)
    print()
    input()
    action_ordinateur_apprentissage(4, 2, 2)
    print()
    action_joueur_apprentissage(2,2,4)
    print()
    action_ordinateur_apprentissage(6, 6, 1)
    action_joueur_apprentissage(4, 4, 4)
```

Annexe 5.2 : Début du mode de jeu apprentissage

```
def action_joueur_apprentissage(ligne_attendue, colonne_attendue, diagonale_attendue) :
    # Dans le mode apprentissage, le joueur possède les pions X
    signe, ligne, colonne = appartenance_pion("X", "V")
    diagonale = choix_diagonale()
    if ligne == ligne_attendue and colonne == colonne_attendue and diagonale == diagonale_attendue :
        #On stocke le nombre de pions mangés par la prise obligatoire dans une variable
        nbre_pions_manges = prise_oligatoire_pions("X", "O", "L", ligne, colonne, 1, 0)
        if nbre_pions_manges == 0 :
            deplacement_pion("X", ligne, colonne, diagonale)
    else :
        return action_joueur_apprentissage(ligne_attendue, colonne_attendue, diagonale_attendue)
```

Annexe 6 : Fonction permettant de vérifier la saisie du joueur dans le mode apprentissage