

SAE 1.02

-

Implémentation d'un besoin client

Sommaire :

1 - Introduction :	3
Contexte.....	3
Objectifs.....	3
Portée et Limites.....	3
2 - Architecture du projet :	4
2.1- Organisation générale.....	4
3 - Description des modifications apportées.....	5
3.1. Ajout des bots.....	5
Conception des bots.....	5
3.2 - Intégration des bots.....	11
3.3 - Changement dans l'organisation des fichiers.....	11
Avant :	11
Après :	11
Impact :	11
4 - Problèmes rencontrés et solutions.....	12
4.1 - Gestion des joueurs en jeu.....	12
Problème :	12
Solution :	12
Impact :	12
4.2 - Réussir à trouver une stratégie gagnante.....	13
Problème :	13
Solution :	13
Impact :	13
5 - Test des nouvelles fonctionnalités.....	14
5.1- Gestion des joueurs en jeu.....	14
5.2- Devinette.....	15
Jouer contre une machine :	15
Machine contre machine :	17
5.3- Allumette.....	18
Jouer contre une machine.....	18
Machine vs machine.....	20
5.4- Morpion.....	21
Jouer contre une machine.....	21
Machine vs machine.....	22
5.5- Puissance 4.....	24
Jouer contre une machine.....	24
Machine vs machine.....	25
5.6 Résumé global des tests.....	26
6 - Conclusion.....	27

1 - Introduction :

Contexte

Pour la S1.02, nous avons amélioré un projet en Python visant à concevoir un ensemble de minis-jeux. Ce projet permet de mettre en pratique nos connaissances en programmation et d'adaptation.

Objectifs

Ce programme propose une expérience ludique pour un ou deux joueurs. Il inclut :

1. Un **menu principal** permettant de naviguer entre les différentes sections.
2. Une **gestion des joueurs**, avec la possibilité de créer, choisir, ou supprimer des profils.
3. Quatre jeux :
 - Le jeu des **allumettes**.
 - Le jeu de **devinette**.
 - Le **morpion**.
 - Le **puissance 4**.
4. Un système de **sauvegarde et chargement** des scores pour permettre la continuité des parties.

Portée et Limites

Ce projet est conçu pour fonctionner exclusivement en mode console. Certaines fonctionnalités avancées, telles que l'interface graphique ou l'intégration en ligne, n'ont pas été implémentées dans le cadre des contraintes du projet

2 - Architecture du projet :

2.1- Organisation générale

Le projet est organisé avec des fichiers et dossiers ayant des rôles spécifiques. Cette organisation facilite la lisibilité du code et sa gestion d'erreur.

```

projet/
|-- main.py                # Point d'entrée du programme
|-- base/
|   |-- interface_joueurs.py # Gestion des joueurs
|   |-- sauvegarde.py        # Sauvegarde des données
|   |-- menuJoueurs.py      # Menu de gestion des joueurs
|   |-- menuScoreSave.py    # Gestion des scores
|   |-- outils.py           # Fonctions utilitaires
|-- jeux/
|   |-- menuJeux.py         # Menu de sélection des jeux
|   |-- allumette/
|       |-- __init__.py     # Initialisation du module allumette
|       |-- bot_allumette.py # IA pour le jeu des allumettes
|       |-- jeu_allumette.py # Logique du jeu des allumettes
|       |-- joueur_allumette.py # Gestion des joueurs pour le jeu des allumettes
|   |-- devinette/
|       |-- __init__.py     # Initialisation du module devinette
|       |-- bot_devinette.py # IA pour le jeu de devinette
|       |-- jeu_devinette.py # Logique du jeu de devinette
|       |-- joueur_devinette.py # Gestion des joueurs pour le jeu de devinette
|   |-- morpion/
|       |-- __init__.py     # Initialisation du module morpion
|       |-- bot_morpion.py  # IA pour le jeu de morpion
|       |-- jeu_morpion.py  # Logique du jeu de morpion
|       |-- joueur_morpion.py # Gestion des joueurs pour le jeu de morpion
|   |-- puissance4/
|       |-- __init__.py     # Initialisation du module puissance 4
|       |-- bot_puissance4.py # IA pour le jeu de puissance 4
|       |-- jeu_puissance4.py # Logique du jeu de puissance 4
|       |-- joueur_puissance4.py # Gestion des joueurs pour le jeu de puissance 4

```

3 - Description des modifications apportées

3.1. Ajout des bots

Conception des bots

Les bots sont conçus avec trois niveaux de difficulté, chaque niveau apportant une stratégie différente :

Jeu des Allumettes

Niveau 1 : Aléatoire

Le bot retire un nombre d'allumettes choisi au hasard (entre 1 et 3). Ce niveau est idéal pour les débutants, car le bot ne suit aucune stratégie.

Code Python :

```
def machine_n1(nombre_allumettes: int) -> int:
    return random.randint(1, min(3, nombre_allumettes))
```

Niveau 2 : Semi-stratégique

Le bot a une chance sur deux de jouer aléatoirement ou de jouer de façon optimale.

Code Python :

```
def machine_n2(nombre_allumettes: int) -> int:
    return random.choice([machine_n1(nombre_allumettes),
                          machine_n3(nombre_allumettes)])
```

Niveau 3 : Optimal

Ce niveau utilise une stratégie mathématique optimale. Le bot cherche toujours à placer l'adversaire dans une situation perdante. En cas de position désavantageuse pour le bot (exemple : nombre d'allumettes - 1 divisible par 4), il joue aléatoirement.

Code Python :

```
def machine_n3(nombre_allumettes: int) -> int:
    if (nombre_allumettes - 1) % 4 == 0:
        return random.randint(1, min(3, nombre_allumettes))
```

Là, le bot vient de vérifier si il est dans une position désavantageuse ou pas, c'est à dire si le nombre d'allumette restante est $(\text{nombre_allumettes} - 1) \% 4 == 0$ (5,9,13, etc.). Cela signifie que quel que soit le coup du bot (1, 2 ou 3 allumettes), l'adversaire peut toujours ramener la situation à une position favorable pour lui. Dans ce cas il joue aléatoirement.

```
    coup = (nombre_allumettes - 1) % 4
    if coup == 0:
        coup = random.randint(1, min(3, nombre_allumettes))
```

Là, le bot n'est pas dans une position désavantageuse et met l'adversaire dans une position désavantageuse

```
    return min(max(1, coup), 3)
```

Jeu de Devinette

Niveau 1 : logique- début

Dans ce niveau le bot joue soit la moitié soit un quart de la plage donnée (pour 100 il joue soit 50 soit 25 soit 75 qui sont les premiers nombres auxquels on pense.)

Code Python :

```
def choix_machine_n1(maximum: int) -> int:
    choix: int
    choix = choice([maximum//2, maximum//4, maximum//4+maximum//2])
    return choix
```

Niveau 1 : Aléatoire - en jeu

La machine définit son choix de manière aléatoire.

Code Python :

```
def choix_machine_n2(maximum: int) -> int:
    return randint(1, maximum)
```

Niveau 2 : Aleatoire, optimal - début

Ici le bot choisi un nombre au hasard

Code Python :

```
def choix_machine_n2(maximum: int) -> int:  
    return randint(1, maximum)
```

Niveau 2 : Ajustement - en jeu

Ici le bot retient ce qu'il a joué et choisi un nombre aléatoire en le plus petit et le plus grand

Code Python :

```
def coup_suivant_n2(debut: int, fin: int) -> int:  
    return randint(debut, fin)
```

Niveau 3 : Optimal - en jeu

Code Python :

Le bot choisi la moitié de la plage qu'il test.

```
def coup_suivant_n3(debut: int, fin: int) -> int:  
    return (fin - debut) // 2 + debut
```

Jeu de Morpion

Niveau 1 : Aléatoire

Le bot place un symbole sur une case libre choisie au hasard.

Code Python :

```
def coup_morpion_n1(grille: list[list[str]], symbole_bot: str) -> list[list[str]]:
    # Récupérer toutes les cases vides.
    cases_vides = coups_possibles(grille)
    if cases_vides:
        # Choisir une case vide aléatoire.
        i, j = random.choice(cases_vides)
        grille[i][j] = symbole_bot
    return grille
```

Niveau 2 : Défensif ou aléatoire

50% de chance jouer comme le niveau 1 et 50% de chance de jouer comme le niveau 3

Code Python :

```
def coup_morpion_n2(grille: list[list[int]], symbole_bot: str,
                    symbole_adversaire: str) -> list[list[int]]:
    return random.choice[coup_morpion_n1(grille, symbole_bot),
                          coup_morpion_n3(grille, symbole_bot, symbole_adversaire)]
```


Niveau 3 : Défensif

Le bot essaye de gagner sur le moment ou bloque les menaces directes de l'adversaire si possible.

Code Python :

```
def coup_morpion_n3(grille: list[list[str]], symbole_bot: str,
                    symbole_adversaire: str) -> list[list[str]]:
    # Vérifier si le bot peut gagner au prochain coup.
    for i in range(3):
        for j in range(3):
            if grille[i][j] == " ":
                grille[i][j] = symbole_bot
                if verif_victoire_cible(grille, symbole_bot):
                    return grille
                grille[i][j] = " " # Annuler le coup si pas gagnant.

    # Bloquer un coup gagnant de l'adversaire.
    for i in range(3):
        for j in range(3):
            if grille[i][j] == " ":
                grille[i][j] = symbole_adversaire
                if verif_victoire_cible(grille, symbole_adversaire):
                    grille[i][j] = symbole_bot
                    return grille
                grille[i][j] = " " # Annuler le coup si pas menaçant.

    # Sinon, jouer comme le bot niveau 1 (coup aléatoire).
    return coup_morpion_n1(grille, symbole_bot)
```

Jeu de Puissance 4

Niveau 1 : Aléatoire

Le bot choisit une colonne libre au hasard.

Code Python :

```
def coup_machine_n1(grille: list[list[str]]) -> int:
    colonnes_valides = [col for col in range(7) if not
                        verif_colonne_pleine(grille, col)]
    return random.choice(colonnes_valides)
```

Niveau 2 : Défensif

Le bot bloque une victoire imminente de l'adversaire.

Code Python :

```
def coup_machine_n2(grille: list[list[str]]) -> int:
    colonnes_valides = [col for col in range(7) if not
        verifier_colonne_pleine(grille, col)]
    if 3 in colonnes_valides:
        return 3
    return random.choice(colonnes_valides)
```

Niveau 3 : Meilleur stratégie

Le bot vérifie si il peut gagner puis regarde si l'adversaire peut gagner sinon il joue aléatoirement

Code Python :

```
def coup_machine_n3(grille: list[list[str]]) -> int:
    # Vérifier si le bot peut gagner au prochain coup
    for col in range(7):
        if not verifier_colonne_pleine(grille, col):
            ligne = trouver_ligne_vide(grille, col)
            grille[ligne][col] = '●'
            if verifier_victoire(grille):
                grille[ligne][col] = ' '
                return col
            grille[ligne][col] = ' '

    # Vérifier si l'adversaire peut gagner au prochain coup et bloquer
    for col in range(7):
        if not verifier_colonne_pleine(grille, col):
            ligne = trouver_ligne_vide(grille, col)
            grille[ligne][col] = '●'
            if verifier_victoire(grille):
                grille[ligne][col] = ' '
                return col
            grille[ligne][col] = ' '

    # Sinon, choisir une colonne aléatoire
    colonnes_valides = [col for col in range(7) if not
        verifier_colonne_pleine(grille, col)]
    return random.choice(colonnes_valides)
```

3.2 - Intégration des bots

Architecture : Les bots sont implémentés dans des modules séparés (par exemple, *bot_allumette.py* pour le jeu des allumettes). Cela permet une maintenance facile et la réutilisation du code.

Interface utilisateur :

```
=====
                          Puissance 4
=====

1. Jouer contre un joueur
2. Jouer contre une machine
3. Faire jouer deux machines l'une contre l'autre
4. Règles
5. Quitter

=====
Veuillez entrer 1, 2, 3, 4 ou 5
-> |
```

3.3 - Changement dans l'organisation des fichiers

Avant :

- Tous les jeux (allumette, devinette, morpion, puissance 4) étaient regroupés directement dans le dossier **jeux/** sous forme de fichiers individuels.
- Chaque jeu partageait une structure commune dans un seul fichier (par exemple *allumette.py*, *devinette.py*).

Après :

- Chaque jeu a maintenant son propre sous-dossier (par exemple, **jeux/allumette/**, **jeux/devinette/**).
- Chaque sous-dossier contient plusieurs fichiers séparés pour différents aspects du jeu, tels que :
 - o *bot_*.py* : Contient la logique d'intelligence artificielle pour le jeu.
 - o *jeu_*.py* : Contient la logique principale du jeu.
 - o *joueur_*.py* : Gère la gestion des joueurs dans le contexte spécifique au jeu.

Impact :

- La modularité a été renforcée : chaque jeu est maintenant organisé dans son propre espace, ce qui rend le projet plus lisible, évolutif et maintenable.
- Les fonctionnalités spécifiques (IA, logique du jeu, gestion des joueurs) sont mieux isolées, ce qui facilite leur extension ou modification.

4 - Problèmes rencontrés et solutions

4.1 - Gestion des joueurs en jeu

Problème :

Dans la version initiale du projet (S1.01), il était obligatoire de créer deux profils de joueurs avant de pouvoir lancer une partie. Cela compliquait l'expérience utilisateur, notamment dans les cas où aucun joueur n'était préalablement créé. De plus, le menu des jeux ne pouvait pas être accédé sans la présence de deux joueurs actifs, ce qui était limitant.

Solution :

Lorsqu'aucun joueur n'est présent dans la base de données, le système propose directement la création d'un profil de joueur.

Si un joueur actif est requis pour accéder au menu des jeux, une notification s'affiche pour inviter à en sélectionner un ou en créer un nouveau.

Une vérification supplémentaire a été ajoutée dans le menu de lancement des parties pour empêcher les erreurs liées à l'absence de joueur actif.

Impact :

Ces améliorations ont simplifié l'utilisation du programme, en rendant le processus plus intuitif et fluide pour les utilisateurs. De plus, elles réduisent les risques d'erreurs ou d'arrêts inattendus du programme.

4.2 - Réussir à trouver une stratégie gagnante

Problème :

Concevoir une intelligence artificielle (IA) efficace pour les jeux proposés (Puissance 4, Morpion, Devinette, et Allumettes) s'est avéré complexe. Bien que des bases stratégiques aient été implémentées, certaines limitations subsistent :

- L'absence d'algorithmes avancés, comme Minimax, pour prévoir les coups futurs.
- Une difficulté à élaborer des stratégies adaptatives dans le temps imparti.
- Des niveaux d'IA inégaux en termes de performances pour certains jeux, notamment Puissance 4 et Morpion.

Solution :

1.Simplification des algorithmes :

Pour compenser le manque de temps, des algorithmes rudimentaires mais fonctionnels ont été privilégiés. Par exemple :

- **Puissance 4** : Priorisation des coups gagnants ou bloquants.
- **Morpion** : Détection des opportunités de victoire ou de défense.

2.Améliorations futures identifiées

- Implémentation d'algorithmes avancés comme Minimax pour le Morpion et Puissance 4.
- Exploration de l'apprentissage automatique pour améliorer les stratégies adaptatives.
- Optimisation des performances des niveaux supérieurs pour garantir une expérience de jeu plus stimulante.

Impact :

Bien que les IA actuelles présentent des limites, elles offrent une variété d'expériences de jeu, à la fois pour les joueurs novices et avancés. Le projet dispose maintenant d'une base solide pour des évolutions futures.

5 - Test des nouvelles fonctionnalités

5.1- Gestion des joueurs en jeu

Sélection des joueurs si la base de donnée est vide

```
=====
Choix des Joueurs Actifs
=====

Veillez créer au minimum 1 joueur avant de pouvoir en choisir.
Veillez entrer le nom du joueur a créer (15 caractères max)
-> |
```

Menu des jeux sans aucun joueur actif

```
=====
Choix des Joueurs Actifs
=====

  Nom du joueur | Allumette | Devinette | Morpion | Puissance 4
1-      Alice |      0pts |      0pts |      0pts |      0pts
2-      Bob  |      0pts |      0pts |      0pts |      0pts

=====

Veillez entrer l'indice du joueur 1
-> |
```

Jouer contre un adversaire non définie

```
=====
Jeu de Devinette
=====

1. Jouer a deux joueurs
2. Jouer contre une machine
3. Faire jouer deux machines l'une contre l'autre
4. Règles
5. Option
6. Quitter

=====

Veillez entrer 1, 2, 3, 4, 5 ou 6
-> 1

Il n'y a pas deux joueurs actifs.
|
```

5.2- Devinette

Jouer contre une machine :

Le joueur commence

```
=====
Jeu de Devinette
=====

Qui doit deviner le nombre ?
  1. La machine
  2. Vous
Votre choix
-> 1

Veuillez choisir le niveau de la machine :
  1. Niveau 1
  2. Niveau 2
  3. Niveau 3
Votre choix
-> 3|
```

```
=====
Jeu de Devinette
=====

Alice, veuillez entrer un nombre entre 1 et 500 compris : 127

Vous avez choisi le nombre 127 (Message affiché 5s).
```

```
=====
Jeu de Devinette
=====

La machine à 10 essais pour trouver le nombre compris entre 1 et 500 compris.

Tentative n°1 :
  La machine propose le nombre 250.
Tentative n°2 :
  La machine propose le nombre 125.
Tentative n°3 :
  La machine propose le nombre 187.
Tentative n°4 :
  La machine propose le nombre 156.
Tentative n°5 :
  La machine propose le nombre 140.
Tentative n°6 :
  La machine propose le nombre 132.
Tentative n°7 :
  La machine propose le nombre 128.
Tentative n°8 :
  La machine propose le nombre 126.
Tentative n°9 :
  La machine propose le nombre 127.

Dommage Alice ! la machine avait choisi le nombre 127 et l'a trouvé en 9 essais.

Appuyez sur la touche 'Entrée' pour continuer...|
```

Le bot commence

```
=====
                                Jeu de Devinette
=====

Qui doit deviner le nombre ?
    1. La machine
    2. Vous
Votre choix
-> 2

Veuillez choisir le niveau de la machine :
    1. Niveau 1
    2. Niveau 2
Votre choix
-> 2|
```

```
=====
                                Jeu de Devinette
=====

    Alice, c'est à vous de jouer !
    Vous avez 10 essais pour trouver le nombre compris entre 1 et 500 compris.

-----

    Veuillez entrer un nombre : 250

Le nombre est plus grand que 250.
Il vous reste 9 essais.

-----
```

...

```
-----

    Veuillez entrer un nombre : 470

Le nombre est plus petit que 470.
Il vous reste 3 essais.

-----

    Veuillez entrer un nombre : 467
ah ?!
Il vous reste 2 essais.

-----

    Bravo Alice ! Vous avez trouvé le nombre 467 en 8 essais.

=====

Appuyez sur la touche 'Entrée' pour continuer...|
```


Machine contre machine :

Niveau 1 vs Niveau 1 (100 tours)

```
Score de la machine 1 (fait deviner le nombre) : 95  
Score de la machine 2 (devine le nombre) : 5  
Temps de l'exécution du programme : 0.23 secondes
```

Niveau 1 vs Niveau 2 (100 tours)

```
Score de la machine 1 (fait deviner le nombre) : 58  
Score de la machine 2 (devine le nombre) : 42  
Temps de l'exécution du programme : 0.22 secondes
```

Niveau 1 vs Niveau 3 (100 tours)

```
Score de la machine 1 (fait deviner le nombre) : 0  
Score de la machine 2 (devine le nombre) : 100  
Temps de l'exécution du programme : 0.11 secondes
```

Niveau 2 vs Niveau 1 (100 tours)

```
Score de la machine 1 (fait deviner le nombre) : 99  
Score de la machine 2 (devine le nombre) : 1  
Temps de l'exécution du programme : 0.22 secondes
```

Niveau 2 vs Niveau 2 (100 tours)

```
Score de la machine 1 (fait deviner le nombre) : 53  
Score de la machine 2 (devine le nombre) : 47  
Temps de l'exécution du programme : 0.21 secondes
```

Niveau 2 vs Niveau 3 (100 tours)

```
Score de la machine 1 (fait deviner le nombre) : 0  
Score de la machine 2 (devine le nombre) : 100  
Temps de l'exécution du programme : 0.2 secondes
```

5.3- Allumette

Jouer contre une machine

```
=====
Jeu de l'allumette
=====

Choisissez le niveau de la machine (1, 2 ou 3)
-> 3

=====
Jeu de l'allumette
=====

Alice commence la partie !

-----

Il reste 20 allumettes.
Alice, combien d'allumettes voulez-vous retirer ? (1, 2 ou 3)
-> 3

Alice retire 3 allumettes.

-----

Il reste 17 allumettes.
Le bot de niveau 3 retire 3 allumettes.

-----

Il reste 14 allumettes.
Alice, combien d'allumettes voulez-vous retirer ? (1, 2 ou 3)
-> 3

Alice retire 3 allumettes.

-----

Il reste 11 allumettes.
Le bot de niveau 3 retire 2 allumettes.

-----

Il reste 9 allumettes.
Alice, combien d'allumettes voulez-vous retirer ? (1, 2 ou 3)
-> 2

Alice retire 2 allumettes.

=====
```

```
Il reste 7 allumettes.
Le bot de niveau 3 retire 2 allumettes.

-----

Il reste 5 allumettes.
Alice, combien d'allumettes voulez-vous retirer ? (1, 2 ou 3)
-> 1
Alice retire 1 allumettes.

-----

Il reste 4 allumettes.
Le bot de niveau 3 retire 3 allumettes.

-----

Il reste 1 allumettes.
Alice, vous devez retirer la dernière allumette.
Alice retire 1 allumettes.

-----

La machine de niveau 3 à gagné !

=====
Appuyez sur la touche 'Entrée' pour continuer...|
```

Machine vs machine

Niveau 1 vs Niveau 1 (100 tours)

```
Score final :  
Machine de niveau 1 : 42 victoires  
Machine de niveau 1 : 58 victoires  
Temps de l'exécution du programme : 0.53 secondes
```

Niveau 1 vs Niveau 2 (100 tours)

```
Score final :  
Machine de niveau 1 : 26 victoires  
Machine de niveau 2 : 74 victoires  
Temps de l'exécution du programme : 0.47 secondes
```

Niveau 1 vs Niveau 3 (100 tours)

```
Score final :  
Machine de niveau 1 : 0 victoires  
Machine de niveau 3 : 100 victoires  
Temps de l'exécution du programme : 0.5 secondes
```

Niveau 2 vs Niveau 2 (100 tours)

```
Score final :  
Machine de niveau 2 : 53 victoires  
Machine de niveau 2 : 47 victoires  
Temps de l'exécution du programme : 0.53 secondes
```

Niveau 2 vs Niveau 3 (100 tours)

```
Score final :  
Machine de niveau 2 : 7 victoires  
Machine de niveau 3 : 93 victoires  
Temps de l'exécution du programme : 0.52 secondes
```

Niveau 3 vs Niveau 3 (100 tours)

```
Score final :  
Machine de niveau 3 : 46 victoires  
Machine de niveau 3 : 54 victoires  
Temps de l'exécution du programme : 0.5 secondes
```

5.4- Morpion

Jouer contre une machine

```
=====
Jeu du Morpion
=====

Veuillez choisir le niveau de la machine :
1. Niveau 1
2. Niveau 2
3. Niveau 3
Votre choix
-> 3|

=====
Jeu du Morpion
=====

La machine commence

=====
| | | |
| | | |
| | | |
=====

=====
| | | |
| | | 0 |
| | | |
=====

Alice (X), c'est à vous de jouer !
Les lignes et colonnes sont numérotées de 1 à 3.

- Veuillez entrer la ligne
-> 1
- Veuillez entrer la colonne
-> 1

=====
| X | | |
| | | 0 |
| | | |
=====
```

...

```

-----
|  X  | 0  | 0  |
|-----|
|  0  |   | 0  |
|-----|
|  X  |   |  X  |
|-----|

Alice (X), c'est à vous de jouer !
Les lignes et colonnes sont numérotées de 1 à 3.

- Veuillez entrer la ligne
-> 1
- Veuillez entrer la colonne
-> 2

La case est déjà occupée.

-----

|  X  | 0  | 0  |
|-----|
|  0  |   | 0  |
|-----|
|  X  |   |  X  |
|-----|

=====

|  X  | 0  | 0  |
|-----|
|  0  | 0  | 0  |
|-----|
|  X  |   |  X  |
|-----|

Le bot de niveau 3 a gagné !
Appuyez sur la touche 'Entrée' pour continuer...|

```

Machine vs machine

Niveau 1 vs Niveau 1 (100 tours)

```

Score de la machine 1 (niveau : 1) : 48
Score de la machine 2 (niveau : 1) : 41
Match nul(s) : 11
Temps d'exécution du programme : 0.45

```

Niveau 1 vs Niveau 2 (100 tours)

```

Score de la machine 1 (niveau : 1) : 24
Score de la machine 2 (niveau : 2) : 55
Match nul(s) : 21
Temps d'exécution du programme : 0.44

```

Niveau 1 vs Niveau 3 (100 tours)

```
Score de la machine 1 (niveau : 1) : 4  
Score de la machine 2 (niveau : 3) : 85  
Match nul(s) : 11  
Temps d'exécution du programme : 0.41
```

Niveau 2 vs Niveau 2 (100 tours)

```
Score de la machine 1 (niveau : 2) : 52  
Score de la machine 2 (niveau : 2) : 34  
Match nul(s) : 14  
Temps d'exécution du programme : 0.4
```

Niveau 2 vs Niveau 3 (100 tours)

```
Score de la machine 1 (niveau : 2) : 9  
Score de la machine 2 (niveau : 3) : 62  
Match nul(s) : 29  
Temps d'exécution du programme : 0.43
```

Niveau 3 vs Niveau 3 (100 tours)

```
Score de la machine 1 (niveau : 3) : 32  
Score de la machine 2 (niveau : 3) : 23  
Match nul(s) : 45  
Temps d'exécution du programme : 0.47
```

5.5- Puissance 4

Jouer contre une machine

```
=====
Puissance 4
=====

Veuillez choisir le niveau de la machine :
    1. Niveau 1
    2. Niveau 2
    3. Niveau 3
Votre choix
-> 3|
```

Les jetons sont animés pour ce mode de jeu.

```
=====
Puissance 4
=====

|01|02|03|04|05|06|07|
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
=====

C'est a Alice ● de jouer.

Veuillez entrer le numéro de la colonne où vous voulez placer votre jeton (1-7)
-> 3|
```

```
=====
Puissance 4
=====

|01|02|03|04|05|06|07|
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
=====

C'est a Alice ● de jouer.

Veuillez entrer le numéro de la colonne où vous voulez placer votre jeton (1-7)
-> |
```

Le bot bloque les coup et a l'air de jouer aléatoirement.



S'il peut gagner il le fait.

Machine vs machine

Ici pas d'animation pour plus de rapidité

Niveau 1 vs Niveau 1 (100 tours)

```
Score de la machine 1 (niveau : 1) : 50
Score de la machine 2 (niveau : 1) : 50
Match nul(s) : 0
Temps d'exécution du programme : 1.1
```

Niveau 1 vs Niveau 2 (100 tours)

```
Score de la machine 1 (niveau : 1) : 19
Score de la machine 2 (niveau : 2) : 81
Match nul(s) : 0
Temps d'exécution du programme : 0.65
```

Niveau 1 vs Niveau 3 (100 tours)

```
Score de la machine 1 (niveau : 1) : 5
Score de la machine 2 (niveau : 3) : 95
Match nul(s) : 0
Temps d'exécution du programme : 1.22
```

Niveau 2 vs Niveau 2 (100 tours)

```
Score de la machine 1 (niveau : 2) : 47  
Score de la machine 2 (niveau : 2) : 53  
Match nul(s) : 0  
Temps d'exécution du programme : 1.23
```

Niveau 2 vs Niveau 3 (100 tours)

```
Score de la machine 1 (niveau : 2) : 9  
Score de la machine 2 (niveau : 3) : 91  
Match nul(s) : 0  
Temps d'exécution du programme : 1.27
```

Niveau 3 vs Niveau 3 (100 tours)

```
Score de la machine 1 (niveau : 3) : 46  
Score de la machine 2 (niveau : 3) : 52  
Match nul(s) : 2  
Temps d'exécution du programme : 1.67
```

5.6 Résumé global des tests

Gestion des joueurs : L'ajout de vérifications et de messages interactifs rend la navigation plus intuitive, tout en garantissant qu'au moins un joueur actif soit disponible avant de commencer une partie.

Intelligence artificielle des bots : Les différents niveaux de difficulté offrent une progression adaptée, avec des stratégies allant de l'aléatoire à des approches plus optimisées. Cependant, des algorithmes avancés comme Minimax pourraient renforcer l'intelligence des bots pour les jeux comme le Morpion ou le Puissance 4.

Évaluation comparative : Les tests machine vs machine ont permis de valider la cohérence des niveaux de difficulté et de mettre en lumière leurs forces et limites, notamment sur la prédictibilité des stratégies utilisées.

Interface et animations : Les animations ajoutées, comme celles des jetons dans Puissance 4, enrichissent l'expérience visuelle et augmentent l'immersion dans le jeu.

6 - Conclusion

En somme, le projet réalisé dans le cadre de la SAE 1.02 a permis d'explorer plusieurs aspects fondamentaux du développement en Python, tout en répondant à un besoin client fictif : concevoir une collection de mini-jeux jouables en console. Les tests ont validé la robustesse et l'efficacité des fonctionnalités développées. Toutefois, certaines limitations subsistent. Ce projet offre plusieurs pistes d'amélioration. Cette SAE a permis d'acquérir des compétences pratiques en programmation et gestion de projet, tout en soulignant l'importance de concevoir des solutions modulaires, maintenables et adaptées aux besoins des utilisateurs.