

SAÉ 1.01 ALGO

Louis Bec & Azario Cossa

SOMMAIRE:

1. CONCEPTUALISATION DU PROJET ET PROTOTYPAGE
2. PROTOTYPE DU PROJET
3. MISE EN PRATIQUE
4. JEUX D'ESSAIS

INTRODUCTION

Le rapport suivant sera divisé en quatre parties : la conceptualisation, qui présentera la mise en place en amont de la structure du programme ; la présentation des prototypes de l'application ; une partie de mise en pratique pour montrer tous les mécanismes utilisés pour l'écriture du programme, avec toutes les parties auxiliaires ; une partie de jeux d'essais, qui peut aussi être vue comme la partie des tests du programme pour évaluer son niveau de fonctionnement et la qualité du programme.

OBJECTIFS :

Générales :

- Présenter la situation globale de l'étudiant en programmation
- Démontrer le niveau des connaissances dans le domaine de programmation python

Spécifiques :

- Conceptualisation d'un projet
- Réalisation d'un projet
- Création de tests pour s'assurer du bon fonctionnement d'un programme

1. CONCEPTUALISATION DU PROJET ET PROTOTYPAGE

Avant d'aborder la phase de codage, nous avons dû concevoir le prototype de l'application, ainsi que définir les différentes composantes que nous devrions développer pour obtenir un projet aussi bien structuré que possible.

1.1. Les structures complexes

Nous avons opté pour une approche basée sur des structures complexes afin de faciliter le processus de développement du jeu. Chaque structure possède une fonction qui lui est associée et qui nous permet de les initialiser.

```
class Player:
    id : int
    name : str
    playerNumber : int
```

Pour commencer nous avons le type **Player** qui va représenter un joueur connecté.

La composition de ce type:

Nous avons l'identifiant de type entier : qui va permettre de stocker l'identifiant de l'utilisateur dans la base de données et ensuite pouvoir faire les traitements nécessaires.

Name de type chaîne de caractères : qui est le nom du joueur

PlayerNumber de type entier: c'est le numéro du joueur pour les jeux du morpion et du puissance 4

```
class CurrentPlayers:
    player1 : Player
    player2 : Player
```

Ensuite le type **CurrentPlayers** qui sont les joueurs qui sont actuellement connectés et autorisés à jouer

La composition de ce type:

Player1, Player2 sont de type Player : Ce sont les deux joueurs courants

```
class GameRiddle:
    colName : str
    pointWin : int
    pointLoose : int
    numberToGuess : int
    maxAttempts : int
    attempts : int
```

C'est le type pour le jeu des devinettes, cela nous permet de regrouper tout ce que nous avons besoins pour le jeu des devinettes

La composition de ce type:

ColName de type chaîne de caractères : Cela nous permet par la suite d'ajouter des points quand le joueur gagne à ce jeu cet élément se retrouve dans tous les types.

PointWin de type entier : Le nombre de points que rapporte une victoire

PointLoose de type entier : Le nombre de points que rapporte une défaite

NumberToGuess de type entier : C'est le nombre que le joueur deux va devoir trouver

MaxAttempts de type entier : C'est le nombre maximal de tentative qu'a le joueur 2 pour trouver le nombre

Attemps de type entier : c'est le nombre de tentatives actuel

```
class GameMatch:
    colName : str
    pointWin : int
    pointLoose : int
    numberOfMatches : int
```

C'est le type pour le jeu des allumettes

La composition de ce type:

colName de type chaîne de caractères : Cela nous permet par la suite d'ajouter des points quand le joueur gagne à ce jeu

PointWin de type entier : Le nombre de points que rapporte une victoire

PointLoose de type entier : Le nombre de points que rapporte une défaite

NumberOfMatches de type entier : c'est le nombre d'allumettes restantes

```
class GameTicTacToe:
    colName : str
    pointWin : int
    pointLoose : int
    pointDraw: int
    player1Pawn : str
    player2Pawn : str
    sizeX : int
    sizeY : int
    plate : list[list[int]]
```

C'est le type pour le jeu du morpion

La composition de ce type:

colName de type chaîne de caractères : Cela nous permet par la suite d'ajouter des points quand le joueur gagne à ce jeu

PointWin de type entier : Le nombre de points que rapporte une victoire

PointLoose de type entier : Le nombre de points que rapporte une défaite

PointDraw de type entier : Le nombre de points que rapporte une égalité

Player1Pawn de type str : c'est le pion du joueur 1 (X ou O)

Player2Pawn de type str : c'est le pion du joueur 2 (X ou O)

SizeX de type int : c'est la taille en x de la matrice (plateau de jeu)

SizeY de type int : c'est la taille en y de la matrice (plateau de jeu)

Plate de type matrice à deux dimension avec entiers : C'est le plateau de jeu du morpion

```
class GameP4:
    colName : str
    pointWin : int
    pointLoose : int
    pointDraw: int
    player1Pawn : str
    player2Pawn : str
    sizeX : int
    sizeY : int
    plate : list[list[int]]
```

C'est le type pour le jeu du puissance 4

La composition de ce type:

colName de type chaîne de caractères : Cela nous permet par la suite d'ajouter des points quand le joueur gagne à ce jeu

PointWin de type entier : Le nombre de points que rapporte une victoire

PointLoose de type entier : Le nombre de points que rapporte une défaite

PointDraw de type entier : Le nombre de points que rapporte une égalité

Player1Pawn de type str : c'est le pion du joueur 1 (rond rouge ou rond jaune)

Player2Pawn de type str : c'est le pion du joueur 2 (rond rouge ou rond jaune)

SizeX de type int : c'est la taille en x de la matrice (plateau de jeu)

SizeY de type int : c'est la taille en y de la matrice (plateau de jeu)

Plate de type matrice à deux dimension avec entiers : C'est le plateau de jeu du morpion

```
class WinningInformations:
    pointWin : int
    pointDraw : int
    pointLoose : int
    isDraw : int
    colName : str
```

Le type **WinningInformations** nous permet de faire du code réutilisable pour n'importe quel jeu cela nous servira pour la fonction de distribution de points.

La composition de ce type:

PointWin de type entier : Le nombre de points que rapporte une victoire

PointLoose de type entier : Le nombre de points que rapporte une défaite

PointDraw de type entier : Le nombre de points que rapporte une égalité

ColName de type chaîne de caractères : Cela nous permet par la suite d'ajouter des points quand le joueur gagne à ce jeu

IsDraw de type booléen : nous permet de savoir si il y a une égalité

1.2. La base de données

La décision d'utiliser une base de données SQLite pour le projet a été motivée par la prévoyance en vue du travail futur sur le même projet. En l'absence d'informations détaillées sur les prochaines étapes du projet, nous avons pris les devants en choisissant d'implémenter une base de données dès maintenant. En intégrant directement une base de données, nous évitons du temps qui pourrait potentiellement être perdu à la réorganisation du projet. Le choix de SQLite a été fait en raison de sa facilité d'utilisation et du fait qu'il ne nécessite aucune mise en place initiale.

```
CREATE TABLE PLAYER
(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name VARCHAR(150) UNIQUE,
    password VARCHAR(150),
    scoreRiddle SMALLINT NOT NULL,
    scoreTtt SMALLINT NOT NULL,
    scoreMatches SMALLINT NOT NULL,
    scoreP4 SMALLINT NOT NULL
)
```

Nous avons décidé de mettre un système de nom , mot de passe pour permettre à chacun de récupérer sa session de jeu et que personne d'autre ne puisse y accéder.

Id de type entier : c'est l'identifiant unique qui permet de s'assurer de l'unicité des enregistrements de la table Player on appelle cela une clé primaire.

Name de type chaîne de caractères avec taille de 150 caractères max: nom de l'utilisateur qui sera unique obligatoirement deux personnes ne peuvent pas avoir le même nom. On appelle cela un champ unique.

Password de type chaîne de caractères avec taille de 150 caractères max: Ce sera le mot de passe de l'utilisateur, cela lui permettra de se connecter à son compte.

Score ... de type entier : ce sont les scores du joueur pour chacun des jeux.

2. PROTOTYPE DU PROJET

En ce moment il est important de donner les premières idées du projet final, la forme dans laquelle le projet sera présenté et aussi les premières notions de fonctionnement. Il est ici que le projet sera présenté dans sa complexité et aussi certaines parties qui justifient la modularisation du projet seront présentées.

2.1. Le fonctionnement intermodulaire

Pour ce projet, nous avons adopté une approche de programmation modulaire. Le fonctionnement modulaire favorise la collaboration ainsi qu'un développement plus simplifié. Pour ce fonctionnement nous avons décidé de découper l'application en quatre parties distinctes:

2.1.1. Data Services :

Le premier dossier, nommé "Data Services", regroupe tous les modules dédiés au traitement des données en lien avec la base de données.

2.1.2. Entity :

Le second dossier, intitulé "Entity", est destiné à contenir tous les types de données évoqués précédemment.

2.1.3. Helpers:

Le troisième dossier, que l'on a décidé de nommer "Helpers", englobe tous les modules liés à l'assistance pour les autres parties du projet.

2.1.4. Dossier Racine :

Enfin, le quatrième dossier est le dossier racine, comprenant tous les modules dédiés au fonctionnement principal des jeux ainsi que le menu du jeu.

2.2. Gestion d'utilisateurs

Les utilisateurs sont distingués par leur nom et leur identifiant unique, pour cela ils doivent se connecter à un compte qu'ils auront obligatoirement créé précédemment.

2.3. Gestion de scores

La gestion des scores est spécifique à chaque jeu, puisque chaque jeu attribue un nombre de points indépendant en fonction des résultats tels que la victoire, la défaite ou l'égalité, sauf pour le jeu des devinettes où le score change en fonction du nombre de tentatives maximum et du nombre de tentatives que le joueur a mises pour trouver le nombre mystère. Le score est établi au commencement de chaque jeu.

2.4. Déroulement de chaque jeux

2.4.1. Jeu des Devinettes (Riddle) :

Nous allons définir les joueurs participants on les appellera joueur 1 et joueur 2 dans l'explication. Le joueur 1 sera chargé de choisir le nombre à deviner qui se trouvera dans l'intervalle [1,200] ainsi que le nombre de tentatives qu'aura le joueur 2. Une fois le jeu initialisé, le joueur 2 devra trouver le nombre. Une fois qu'il aura fait une tentative le joueur 1 devra dire si le nombre donné est trop petit, trop grand ou c'est gagné en fonction du nombre à trouver.

2.4.2. Jeu des Allumettes (Match):

Il y a 20 allumettes au départ qui seront affichées et les joueurs 1 et 2 devront simultanément retirer de 1 à 3 allumettes. Au final c'est le joueur retirant la dernière allumette qui perd la partie. Les allumettes seront représentés par des caractères ASCII empilés, de couleur rouge et jaune.

2.4.3. Jeu du Morpion (Tic Tac Toe) :

Les joueurs 1 et 2 s'affrontent en jouant simultanément. Chacun doit préciser les coordonnées X et Y, désignant respectivement la ligne et la colonne où ils désirent positionner leur pion. La victoire revient au premier joueur qui parvient à aligner trois de ses pions, symbolisés par la X et le O, sur la grille.

2.4.4. Jeu du Puissance 4 (P4):

Les joueurs 1 et 2 s'engagent dans une partie simultanée du Puissance 4. À tour de rôle, ils sélectionnent une colonne où déposer leur pion. Chaque joueur doit spécifier le numéro de la colonne où ils souhaitent insérer leur pion. Le but du jeu

est d'être le premier à aligner quatre de ses pions de manière consécutive, que ce soit horizontalement, verticalement ou en diagonale.

La grille de jeu est composée de 7 colonnes et 6 lignes. Les pions des joueurs sont représentés par des jetons distinctifs, marqués d'un caractère unicode de couleur jaune pour le joueur 1 et ce même caractère de couleur rouge pour le joueur 2.

Le premier joueur à créer une séquence de quatre pions alignés remporte la partie. Si la grille est remplie sans qu'aucun joueur ne gagne, la partie est dite nulle.

2.4.5. Affichage des scores:

Les joueurs peuvent décider d'afficher les scores de chaque jeu de manière descendante c'est-à-dire du meilleurs joueurs au plus mauvais et cela pour chaque jeu.

3. MISE EN PRATIQUE

Dans cette partie, l'exécution avec une présentation de captures d'écran et aussi du fonctionnement pour chacun des éléments sera présentée. C'est, de manière résumée, la partie de présentation du fonctionnement correct du programme avec son explication, où une exécution de test sera réalisée.

 **Certaines photos de fonctions n'ont pas de pré-post condition afin de faciliter leur lisibilité.**

3.1. Fonctionnement intermodulaire

Le fonctionnement intermodulaire du programme est défini par une intercommunication entre ces modules, par passage de paramètres dans plusieurs cas. Les cas suivants peuvent être pris comme exemple:

3.1.1. Le login des joueurs

Pour une meilleure sécurité, il a été décidé de faire une gestion de joueurs avec l'inclusion du mot de passe, ça veut dire qu'il faudra avoir le *nom d'utilisateur* et aussi le mot de passe qui seront vérifiés dans la base de données du programme avec la partie suivante du programme main :

```

while player1.id == -1:
    print("Connectez-vous à votre compte joueur 1")
    choice = input("vous souhaitez vous " + setColorGreen("connecter(0)") + " ou vous " + setColorGreen("inscrire(1)"))
    while not isDigit(choice) or (int(choice) != 0 and int(choice) != 1):
        choice = input(setColorYellow("vous souhaitez vous " + setColorGreen("connecter(0)") + " ou vous " + setColorGreen("inscrire(1)")))
    username = input("username")
    password = getpass.getpass("password")
    if int(choice) == 0:
        player1 = connect(username,password, con)
        if player1.id == -1:
            print(setColorRed("Mot de passe ou nom d'utilisateur invalide"))
    elif int(choice) == 1:
        player1 = register(username,password,con)
        if player1.id == -1:
            print(setColorRed("⚠ Ce nom d'utilisateur est déjà utilisé"))
    else:
        print(setColorRed("ce choix n'existe pas"))

```

Avec l'appelle de méthode `player1 = connect(username, password, con)`, il devient possible de vérifier les données d'entrée, faire la comparaison avec les données présentes sur la base de données et après présenter une réponse qui peut être positif ou alors négatif, permettant de savoir la présence ou non de l'utilisateur.

```

def changePlayer(currPlayers: CurrentPlayers):
    """
    Permet d'echanger le joueur1 avec le joueur2 pour que le role change en fonction du jeu

    Args :
        currPlayers (CurrentPlayers) cette variable represente les deux joueur qui jouent actuellement
    Return:
        Rien
    """
    tmpPlayer : Player
    tmpValue : int

    #Changer les joueurs
    tmpPlayer = currPlayers.player1
    currPlayers.player1 = currPlayers.player2
    currPlayers.player2 = tmpPlayer
    #Changer valeurs
    tmpValue = currPlayers.player1.playerNumber
    currPlayers.player1.playerNumber = currPlayers.player2.playerNumber
    currPlayers.player2.playerNumber = tmpValue

```

Le module affiché, est un exemple de module qui fait partie du fonctionnement du programme, il prend des données de la classe **CurrentPlayers** venant du module **player** se trouvant lui-même dans le **dossier entity** pour pouvoir changer les joueurs.

3.2. Gestion des utilisateurs

La gestion d'utilisateurs dans ce programme est faite en prenant comme ressource les documents fonctionnantes à base de SQL, dont l'utilisation de requêtes est complètement inévitable, et pour pouvoir avoir une meilleure vue de ce fonctionnement, il faut faire attention à les captures d'écran suivantes :

```

def register(name : str, password : str, conn : Connection)->Player:

    res : Cursor
    cur: Cursor | None
    query : str
    playerElements : list[str] | None

    player : Player
    player = Player()
    player.id = -1
    cur = None
    try:
        #creation du curseur de base de données
        cur = conn.cursor()
        query = f"INSERT INTO PLAYER (name,password,scoreRiddle,scoreTtt,scoreMatches,scoreP4) VALUES (?,?,0,0,0,0)"
        #execute la requete sql
        cur.execute(query,(name,password))
        #rend permanent les changements dans la base de données
        conn.commit()
        query = f"SELECT id,name,scoreRiddle,scoreTtt,scoreMatches,scoreP4 FROM PLAYER WHERE id = ?"
        res = cur.execute(
            query,(
                str(cur.lastrowid),
            ))
        #recuperer un seul enregistrement
        playerElements = res.fetchone()
        if playerElements == None:
            return player
        playerInit(player, int(playerElements[0]), playerElements[1])
        return player
    except sqliteErr:
        player.id = -1
        return player
    finally:
        if cur is not None:
            #on ferme le curseur dans tout les cas
            cur.close()

```

Cette impression d'écran est responsable d'aller chercher les utilisateurs avec avec ses informations pour pouvoir être utilisée pendant l'exécution du programme. Comme il est possible de voir, cette fonction fait utilisation des queries SQL pour lire ces données.

```

def connect(name :str, password : str , conn : Connection) -> Player:
    |
    res : Cursor
    query : str
    cur: Cursor | None
    playerElements : list[str] | None
    player : Player

    player = Player()
    player.id = -1
    cur = None
    try :
        #creation du curseur de base de données
        cur = conn.cursor()
        query = f"SELECT id,name,scoreRiddle,scoreTtt,scoreMatches, scoreP4 FROM PLAYER WHERE name = ? AND password = ?"
        res = cur.execute(
            query,(
                name,
                password
            ))
        #recupere un seul element dans la base de données
        playerElements = res.fetchone()
        if playerElements == None:
            return player
        playerInit(player, int(playerElements[0]), playerElements[1])
        return player
    except sqliteErr:
        player.id = -1
        return player
    finally:
        if cur is not None:
            #on ferme le curseur dans tout les cas
            cur.close()

```

La méthode affichée est responsable de registrer les nouveaux utilisateurs, en prenant comme entrée l'username et le mot de passe, après avoir passé de cette phase, l'utilisateur pourra jouer.

3.3. Gestion de scores

Les points de chaque jeu sont prédéfinis et resteront constants au fil du temps. Lors de l'initialisation des structures, nous attribuons un nombre de points en fonction du résultat du joueur (victoire, défaite ou égalité). Le score varie uniquement dans le jeu de devinettes, en fonction du nombre maximal de tentatives autorisées et du nombre réel de tentatives effectuées par l'utilisateur pour découvrir le nombre mystère. Le nombre de points gagnés minimal est de 1 si le joueur ne parvient pas à trouver le nombre dans un nombre restreint de tentatives, et au maximum le nombre de points défini au départ s'il atteint le nombre de points défini initialement.

```

def calcPoints(attempts: int, maxAttempts: int, pointWin: int) -> int:
    if attempts <= 0 or maxAttempts <= 0:
        return 0 # Éviter la division par zéro

    if attempts == 1:
        return pointWin # Si l'utilisateur trouve en 1 essai, attribuer le maximum de points

    points = 1 - (attempts - 1) / maxAttempts
    points = max(0, points) # Limiter le score à 0 au minimum
    points *= pointWin

    return min(pointWin, int(points)) # Assurer un maximum de 15 points attribués

```

La fonction affichée est responsable de définir le nombre de points du gagnant pour le jeu des Devinettes. Le nombre de points change en fonction du nombre de tentatives du joueur ainsi que du nombre de tentatives maximales qu'il a pour trouver le nombre mystère.

```

def pointsDistribution(
    winningInformations : WinningInformations,
    currentPlayers : CurrentPlayers,
    currentPlayer : Player ,
    conn : Connection
)->None:

    otherPlayer : Player

    otherPlayer = getOtherPlayer(currentPlayers,currentPlayer)
    #s'il n'y a pas d'egalité
    if not winningInformations.isDraw:
        print(setColorGreen("🎉 Bravo c'est " + "(" + currentPlayer.name + ")" + " qui l'emporte vous gagnez " + str(winningInformations.pointWin) + " points"))
        addPoint(currentPlayer.id,winningInformations.pointWin,conn,winningInformations.colName)
        print(setColorGreen("🙏 Merci " + "(" + otherPlayer.name + ")" + " d'avoir participé vous gagnez " + str(winningInformations.pointLoose) + " points"))
        addPoint(otherPlayer.id,winningInformations.pointLoose,conn,winningInformations.colName)
    #s'il y a égalité
    else:
        print(setColorGreen("🎉 Bravo une égalité parfaite " + currentPlayers.player1.name + " et " + currentPlayers.player2.name + "vous remportez " + str(winningInformations.pointDraw) + " points"))
        addPoint(currentPlayers.player1.id,winningInformations.pointDraw,conn,winningInformations.colName)
        addPoint(currentPlayers.player2.id,winningInformations.pointDraw,conn,winningInformations.colName)

```

Cette fonction permet d'attribuer les points aux deux joueurs peu importe le jeu. C'est pour cela que nous avons créé une structure qui fait le lien en regroupant toutes les informations nécessaires.

3.4. Gestion du menu

```
while not end:
    print(setColorGreen("\nChoisissez le jeu:\n"))
    gameChoice = input("1 => Pour jouer aux devinettes \n2 => Pour jouer au jeu des allumettes...")
    while not isDigit(gameChoice):
        print(setColorRed("Ce choix n'existe pas\n"))
        gameChoice = input("1 => Pour jouer aux devinettes \n2...")

    if int(gameChoice) == 1:
        gameRiddle(currentPlayers,con)
    elif int(gameChoice) == 4:
        gameP4(currentPlayers,con)
    elif int(gameChoice) == 3:
        gameTicTacToe(currentPlayers,con)
    elif int(gameChoice) == 2:
        gameMatch(currentPlayers,con)
    elif int(gameChoice) == 5:
        displayLeaderBoards(con)
    elif int(gameChoice) == 6:
        print(setColorGreen("Merci d'avoir joué à bientôt"))
        end = True
    else:
        print(setColorRed("Ce choix n'existe pas"))
        changePlayer(currentPlayers)
```

Nous avons une variable `end` qui permet de savoir le statut du programme `True` si le programme doit s'arrêter `False` sinon. Chaque jeu est choisi par l'utilisateur en fonction d'un chiffre. Par exemple, le joueur doit entrer 2 s'il veut jouer au jeu des allumettes, 6 s'il veut arrêter le jeu. Chaque jeu prend en paramètre les joueurs courants ainsi que la connexion a la base de données pour faire les modifications nécessaires ensuite. A chaque tour de jeu il y a un changement des joueurs pour que ce ne soit pas toujours les mêmes joueurs qui fassent les mêmes actions ou les mêmes pions ou commencent toujours...

3.5 Les bibliothèques utilisés

Getpass : Elle nous permet de saisir des informations sans qu'il n'y est l'affichage sur l'écran cela a été nécessaire dans la **connexion et la création de compte** pour cacher le mot de passe de l'utilisateur mais aussi pour la **saisie du nombre mystère** dans le jeu des devinettes

Sqlite3 : La bibliothèque `sqlite` nous permet de faire la liaison entre notre application et la base de données `sqlite`. Cette bibliothèque nous aura été utile pour la fonction de **connexion, inscription, récupération des scores et ajout des scores**

3.5 Le code couleur ANSI

Les couleurs **ANSI** sont un ensemble de codes utilisés pour définir des couleurs dans les terminaux informatiques. Nous avons intégré des couleurs **ANSI** dans le

projet ce qui nous offre une meilleure lisibilité du texte et permet de signaler des informations importantes de manière visuelle.

Voici les fonctions qui permettent de mettre de la couleur sur un texte.

```
def setColorRed(text : str) -> str:  
    return "\u001b[31m" + text + "\u001b[0m"  
  
def setColorBlue(text: str) -> str:  
    return "\u001b[34m" + text + "\u001b[0m"  
  
def setColorGreen(text : str) -> str:  
    return "\u001b[32m" + text + "\u001b[0m"  
  
def setColorYellow(text : str)->str:  
    return "\u001b[33m" + text + "\u001b[0m"
```

La première couleur **ANSI** dans la chaîne de caractères est la couleur que nous souhaitons mettre et la deuxième nous permet de réinitialiser la couleur afin d'enlever la couleur pour que le reste du texte ne soit pas affecté par le changement de couleur.


4. JEUX D'ESSAIS

Pour la partie des jeux d'essais, le code sera exécuté dans le but de vérifier son fonctionnement dans les différents cas possibles, tout en ayant une notion de sa performance face à plusieurs erreurs éventuelles. Pour une meilleure compréhension, les exécutions seront séparées par le type d'erreur testé lorsque cela est possible.

4.1. Erreur 1 : Saisie utilisateurs

```
/bin/python3 /home/louis/Bureau/sael/index.py
louis@louis-IdeaPad-3-15IML05 ~/Bureau/sael [main] /bin/python3
x.py
Bienvenue à vous! etes vous prêts à jouer?
Chargement...
Je vois que vous n'etes pas connectés
Connectez-vous à votre compte joueur 1
Vous souhaitez vous connecter(0) ou vous inscrire(1)dffffffdfgdfdfdfdf
Vous souhaitez vous connecter(0) ou vous inscrire(1)
```

Dans le premier cas, l'erreur à tester concernait exclusivement la saisie de données par l'utilisateur. La capture d'écran ci-dessus montre les méthodes adoptées par le groupe, dans lesquelles chaque type de saisie est géré par le programme, réduisant ainsi les probabilités d'avoir une erreur susceptible de provoquer le crash du programme. Cette manière de gérer les saisies a été appliquée pour chaque champ utilisateur avec des restrictions de saisie de données. Dans ce cas particulier, l'utilisateur a la possibilité d'entrer un "0" pour une connexion et un "1" pour une inscription, mais l'utilisateur a saisi "dffffffdfgdfdfdf", qui n'est pas contenu dans le champ de choix prévu.

 Cette erreur est traitée de la même manière dans toute la suite du code. C'est pourquoi nous ne ferons pas à chaque fois l'intégralité des explications pour le même type d'erreurs.

Pour avoir une meilleure notion du fonctionnement de ce traitement, voici d'autres exemples sur le menu du jeu:


```
Choisissez le jeu:
```

```
1 => Pour jouer aux devinettes
2 => Pour jouer au jeu des allumettes
3 => Pour jouer au jeu du morpion
4 => Pour jouer au jeu du puissance 4
5 => Pour afficher le classement des joueurs par jeux
6 => Pour quitter le jeu
Mon choix est hhfghfgh
Ce choix n'existe pas
```

Saisie non comprise dans l'intervalle des entrées prévues. L'intervalle des possibles ici est [1,6]

- Nous avons entré une chaîne de caractères quelconque et on voit que le cas d'erreur est bien géré.

```
1 => Pour jouer aux devinettes
2 => Pour jouer au jeu des allumettes
3 => Pour jouer au jeu du morpion
4 => Pour jouer au jeu du puissance 4
5 => Pour afficher le classement des joueurs par jeux
6 => Pour quitter le jeu
Mon choix est 7
Ce choix n'existe pas
```

Saisie (7) non comprise dans l'intervalle des entrées prévues.

```
1 => Pour jouer aux devinettes
2 => Pour jouer au jeu des allumettes
3 => Pour jouer au jeu du morpion
4 => Pour jouer au jeu du puissance 4
5 => Pour afficher le classement des joueurs par jeux
6 => Pour quitter le jeu
Mon choix est 6
Merci d'avoir joué à bientôt
```

Si le joueur entre 6 le jeu s'arrête bien. Si le joueur entre un chiffre compris dans l'intervalle des possibles le jeu se passe correctement et effectue l'action qui est indiqué dans le menu.

4.2. Erreur 2 : Gestion des données

Pour montrer que la communication entre le programme et la base de données est bien fait, exemples de captures d'écran sont affichés, comme c'est le cas de la capture suivant :

Leaderboard pour le jeu Allumettes:

ID	Nom	Score

88	Louis	122
87	Mathieu	111
1	valeur 1	19
71	valeur 5	16
89	Picsou	15
90	AZLE1XV	2
93	Azzario	2
80	valeur78	1
81	valeur5	1
82	valeurf	0

Les scores avant la victoire de mathieu sur le jeu des allumettes montrent qu'il est classé deuxième avec 111 points



Louis à vous de jouer

choisissez entre 1,2 ou 3 allumettes à retirer 3

😊 Bravo c'est (Mathieu) qui l'emporte vous gagnez +15 points

😞 Merci (Louis) d'avoir participé vous gagnez +2 points

Choisissez le jeu:

Le jeu nous indique que 15 points de victoire ont été ajoutés dans le score total de Mathieu.

Leaderboard pour le jeu Allumettes:

ID	Nom	Score

87	Mathieu	126
88	Louis	124
1	valeur 1	19
71	valeur 5	16
89	Picsou	15
90	AZLE1XV	2
93	Azzario	2
80	valeur78	1
81	valeur5	1
82	valeurf	0

Il est visible que le score à Mathieu a bien changé donc la communication avec la base de données pour récupérer les scores fonctionne bien. La fonction d'ajout de points est la même pour tous les jeux donc il est possible de voir que le système de points est bien fonctionnel.

4.4. Erreur 4 : Connexion et inscription

Pendant les parties de connexion et d'inscription, de nombreuses erreurs peuvent être commises par l'utilisateur, c'est pourquoi une méthode très efficace pour traiter tous ces cas possibles, du moins à ce niveau, était nécessaire. Et pour chacun de ces cas, voici un exemple ci-dessous :

```
Vous souhaitez vous connecter(0) ou vous inscrire(1)0
username Louis
password
Connectez-vous à votre compte joueur 2
vous souhaitez vous connecter(0) ou vous inscrire(1)
```

Ici, un utilisateur décide de se connecter à son compte et tout fonctionne bien.

```
Connectez-vous à votre compte joueur 2
vous souhaitez vous connecter(0) ou vous inscrire(1)0
username Louis
password
△ Ce joueur est déjà connecté
Connectez-vous à votre compte joueur 2
```

Le programme a la capacité d'identifier un utilisateur connecté, rendant ainsi impossible toute tentative de se connecter une seconde fois au même compte.

```
vous souhaitez vous connecter(0) ou vous inscrire(1)0
username yjtyttyj
password
Mot de passe ou nom d'utilisateur invalide
Connectez-vous à votre compte joueur 2
vous souhaitez vous connecter(0) ou vous inscrire(1)
```

Lorsque les données de connexion sont incorrectes ou ne correspondent pas à ce qui est enregistré dans la base de données, un message d'erreur sera envoyé.

```
Je vois que vous n'etes pas connectés
Connectez-vous à votre compte joueur 1
Vous souhaitez vous connecter(0) ou vous inscrire(1)1
username Azzario
password
Connectez-vous à votre compte joueur 2
vous souhaitez vous connecter(0) ou vous inscrire(1)
```

Si toutes les étapes sont bien suivies, aucune erreur ne devrait être trouvée. Et l'une des choses les plus importantes ici est le fait que le mot de passe a été masqué pour une meilleure sécurité.

```
vous souhaitez vous connecter(0) ou vous inscrire(1)1
username Azzario
password
⚠ Ce nom d'utilisateur est déjà utilisé
Connectez-vous à votre compte joueur 2
vous souhaitez vous connecter(0) ou vous inscrire(1)
```

Si le nom d'utilisateur a déjà été pris par un autre utilisateur, un message d'erreur sera également envoyé.

4.5. Erreur jeu devinettes :

```
REGLES DU JEU :
1. Le joueur 1 choisi le nombre que le joueur 2 va devoir trouvé se situant
que le nombre de tentative max que le joueur deux a pour trouver (> 1) le n
2. Le joueur deux donne alors un nombre qui pense etre le bon
3. Le joueur 2 indique au joueur un si le nombre donné est (trop petit trop
4. Le jeu s'arrete quand le joeur 2 à trouvé le bon nombre dans le nombre d
son nombre de tentative a depasser le nombre de tentatives maximale

(Azzario) Entrez le nombre à deviner entre 0 et 200
(Azzario)Entrez le nombre à deviner entre 0 et 200
```

Pour ce jeu, le nombre à deviner ne doit pas être visible par le deuxième joueur, c'est la raison pour laquelle le nombre n'est pas affiché. Pour une meilleure

compétitivité, le nombre choisi doit se situer dans un certain intervalle, qui est, dans ce cas, de 0 à 200.

```
(Azzario) Entrez le nombre de tentative qu'aura le joueur >1 0
(Azzario) ΔEntrez le nombre de tentative qu'aura le joueur >1
```

```
(Azzario) ΔEntrez le nombre de tentative qu'aura le joueur >1 -500
(Azzario) ΔEntrez le nombre de tentative qu'aura le joueur >1
```

Le joueur un doit choisir le nombre de tentatives que le joueur 2 aura pour trouver le nombre mystère. Un message d'erreur est envoyé lorsque la valeur ne se trouve pas entre 2 et plus infini.

- Nous avons testé les cas où l'utilisateur entre **(0,-500)** le cas d'erreur est bien géré.

```
(Azzario) ΔEntrez le nombre de tentative qu'aura le joueur >1 2
(Louis) Essayez de trouver le nombre7000
(Azzario) Entrez (trop petit), (trop grand) ou (c'est gagné) en fonction du nombre entre trop petit
(Azzario) ne mentez pas! Entrez (trop petit), (trop grand) ou (c'est gagné) en fonction du nombre e
ntre trop grand
```

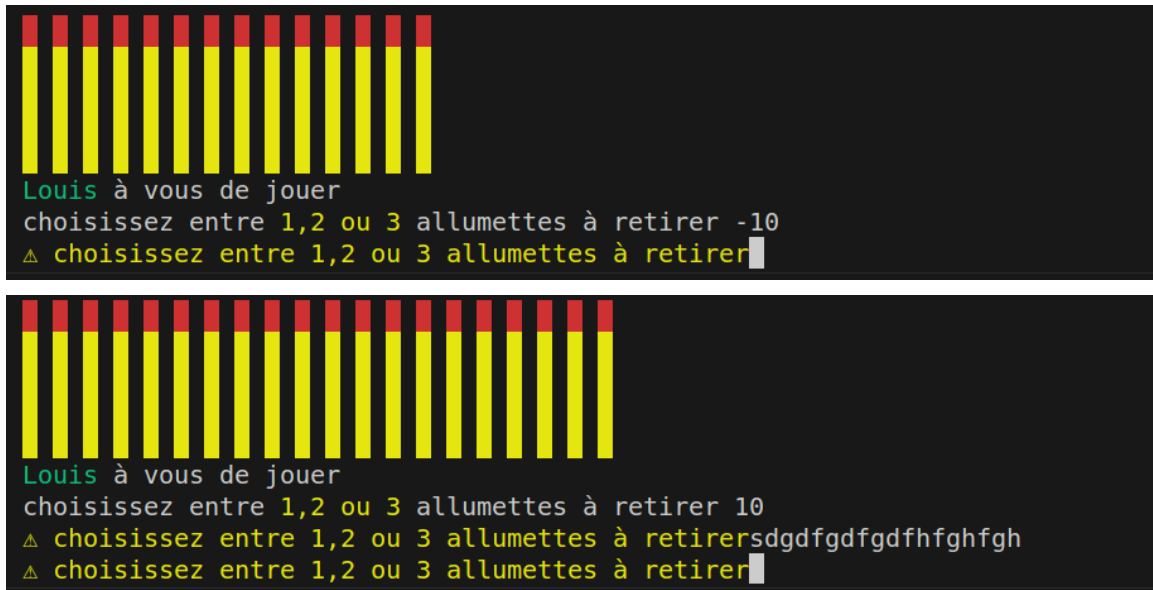
Si le joueur 2 a un nombre trop grand et que le joueur 1 indique un nombre trop petit ou égal, l'affichage du message ne ment pas est envoyé pour indiquer que le joueur 2 essaie de tricher. Ça veut dire que le programme contrôle les entrées des joueurs à chaque fois.

- Dans ce cas nous avons entré le nombre **200** et l'utilisateur 2 a entré **7000**, le joueur 1 indique un nombre **trop petit** ce qui est faux, le message d'erreur est bien affiché.

```
(Azzario) ΔEntrez le nombre de tentative qu'aura le joueur >1 2
(Louis) Essayez de trouver le nombre7000
(Azzario) Entrez (trop petit), (trop grand) ou (c'est gagné) en fonction du nombre entre trop petit
(Azzario) ne mentez pas! Entrez (trop petit), (trop grand) ou (c'est gagné) en fonction du nombre e
ntre trop grand
(Louis) Essayez de trouver le nombre200
(Azzario) Entrez (trop petit), (trop grand) ou (c'est gagné) en fonction du nombre entre trop grand
😊 Bravo c'est (Azzario) qui l'emporte vous gagnez +1 points
😄 Merci (Louis) d'avoir participé vous gagnez +2 points
```

Le joueur 1 a entré précédemment 2 tentatives pour trouver le nombre caché. Le joueur 2 après avoir fait deux tentatives, le jeu s'arrête.

4.6. Erreur jeu des allumettes :



The image contains two screenshots of a terminal window showing a matchstick game. In both, there are 15 matchsticks represented by yellow bars with red tips. The prompt is 'Louis à vous de jouer' and the instruction is 'choisissez entre 1,2 ou 3 allumettes à retirer'. The first screenshot shows an input of '-10' which is rejected with a message '△ choisissez entre 1,2 ou 3 allumettes à retirer'. The second screenshot shows an input of 'sdgdfgdfgdfhghfgh' which is also rejected with the same message.

```
Louis à vous de jouer
choisissez entre 1,2 ou 3 allumettes à retirer -10
△ choisissez entre 1,2 ou 3 allumettes à retirer

Louis à vous de jouer
choisissez entre 1,2 ou 3 allumettes à retirer 10
△ choisissez entre 1,2 ou 3 allumettes à retirersdgdfgdfgdfhghfgh
△ choisissez entre 1,2 ou 3 allumettes à retirer
```

les saisies des nombre d'allumettes ne peut être que dans l'intervalle [1,3], ici nous avons essayé avec d'autres valeurs pour voir que le programme gère bien ces cas d'erreurs

- Nous avons essayé avec une chaîne quelconque (**sdgd...**)
- Nous avons essayé avec des nombres (**-10,10**)

- Nous avons essayé des nombres **(12,-12,0)** qui sont supérieur ou inférieur à l'intervalle des possible

```

      1   2   3
+---+---+---+
1 |   |   |   |
+---+---+---+
2 |   |   |   |
+---+---+---+
3 |   |   | X |
+---+---+---+
False
(Louis) à toi de jouer
choisi ou tu souhaites déposer ton pion sur l'axe x3
choisi ou tu souhaites déposer ton pion l'axe y3
❌(Louis) il ne reste plus d'emplacement libre sur cette colonne

```

On voit que la case **(3,3)** est occupée. Lorsqu'un utilisateur tente de saisir une position déjà occupée, un message d'erreur est affiché pour lui indiquer qu'il ne peut pas utiliser cette position.

```

(Louis) à toi de jouer
choisi ou tu souhaites déposer ton pion sur l'axe x2
choisi ou tu souhaites déposer ton pion l'axe y2
      1   2   3
+---+---+---+
1 |   |   |   |
+---+---+---+
2 |   | 0 |   |
+---+---+---+
3 |   |   | X |
+---+---+---+
False
(Azzario) à toi de jouer
choisi ou tu souhaites déposer ton pion sur l'axe x

```

Quand toutes les règles sont bien respectées, le pion sera mis avec succès. Comme ici avec l'exemple d'un pion placé sur la ligne 2 et la colonne 2.


```

choisi ou tu souhaites déposer ton pion sur l'axe x2
choisi ou tu souhaites déposer ton pion l'axe y1
    1   2   3
+---+---+---+
1 | X | 0 |   |
+---+---+---+
2 |   | 0 | X |
+---+---+---+
3 |   | 0 | X |
+---+---+---+
True
😊 Bravo c'est (Louis) qui l'emporte vous gagnez +5 points
😞 Merci (Azzario) d'avoir participé vous gagnez +1 points

```

Un cas de victoire : le joueur Louis l'emporte car il a aligné 3 pions sur la colonne n°2.

```

(Louis) à toi de jouer
choisi ou tu souhaites déposer ton pion sur l'axe x (ligne) 3
choisi ou tu souhaites déposer ton pion l'axe y (colonne) 1
    1   2   3
+---+---+---+
1 | X | 0 | X |
+---+---+---+
2 | X | 0 | 0 |
+---+---+---+
3 | 0 | X | X |
+---+---+---+
False
😞 Bravo une égalité parfaite Louis et Mathieu vous remportez 2 points

```

Voici un cas d'égalité : aucun joueur n'a aligné 3 pions. On annonce alors une égalité

4.6. Erreur jeu du puissance 4 :

```
(Louis) à toi de jouer  
choisi la colonne ou tu souhaites déposer ton pion7
```

	1	2	3	4	5	6	7	
+	-	-	-	-	-	-	-	+
	●	●	●	●	●	●	●	
+	-	-	-	-	-	-	-	+
	●	●	●	●	●	●	●	
+	-	-	-	-	-	-	-	+
	●	●	●	●	●	●	●	
+	-	-	-	-	-	-	-	+
	●	●	●	●	●	●	●	
+	-	-	-	-	-	-	-	+
	●	●	●	●	●	●	●	
+	-	-	-	-	-	-	-	+
	●	●	●	●	●	●	●	
+	-	-	-	-	-	-	-	+

🤖 Bravo une égalité parfaite Mathieu et Louisvous remportez 2 points

Aucun joueur n'a réussi à aligner 4 pions, c'est donc une égalité. On annonce alors aux joueurs qu'ils ont fait une égalité.

```
vous allez jouer sur cette grille
  1     2     3     4     5     6     7
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
(Mathieu) à toi de jouer
choisi la collonne ou tu souhaites deposer ton pion8
choisi la collonne ou tu souhaites deposer ton pion entre 1 et 7 inclus-5
choisi la collonne ou tu souhaites deposer ton pion entre 1 et 7 inclusfgvbdcbcbv
choisi la collonne ou tu souhaites deposer ton pion entre 1 et 7 inclus
```

La saisie n'est pas prise en compte si elle ne rentre pas dans l'intervalle [1;7]

- Nous avons essayé une chaîne de caractères quelconque (**fgvbdcvbcbvb**)
- Nous avons essayé des nombres (**-5,8**) plus grand et plus petit que l'intervalle autorisée

```
choisi la collonne ou tu souhaites déposer ton pion7
  1   2   3   4   5   6   7
+---+---+---+---+---+---+
|   |   |   |   |   |   | ● |
+---+---+---+---+---+---+
|   |   |   |   |   |   | ● |
+---+---+---+---+---+---+
|   |   |   |   |   |   | ● |
+---+---+---+---+---+---+
|   |   |   |   |   |   | ● |
+---+---+---+---+---+---+
|   |   |   |   |   |   | ● |
+---+---+---+---+---+---+
|   |   |   |   |   |   | ● |
+---+---+---+---+---+---+
|   |   |   |   |   |   | ● |
+---+---+---+---+---+---+
(Mathieu) à toi de jouer
choisi la collonne ou tu souhaites déposer ton pion7
❌ (Mathieu) il ne reste plus d'emplacemement libre sur cette colonne
(Mathieu) à toi de jouer
choisi la collonne ou tu souhaites déposer ton pion
```

Si l'utilisateur essaie d'entrer un pion dans une colonne qui est déjà pleine, un message d'erreur lui est affiché.

```
choisi la collonne ou tu souhaites déposer ton pion56
choisi la collonne ou tu souhaites déposer ton pion entre 1 et 7 inclus6
  1   2   3   4   5   6   7
+---+---+---+---+---+---+
|   |   |   | ● | ● |   | ● |
+---+---+---+---+---+---+
|   |   |   | ● | ● |   | ● |
+---+---+---+---+---+---+
|   |   |   | ● | ● |   | ● |
+---+---+---+---+---+---+
|   |   |   | ● | ● |   | ● |
+---+---+---+---+---+---+
|   |   |   | ● | ● |   | ● |
+---+---+---+---+---+---+
|   |   |   | ● | ● | ● | ● |
+---+---+---+---+---+---+
😊 Bravo c'est (Mathieu) qui l'emporte vous gagnez +5 points
😁 Merci (Louis) d'avoir paricipé vous gagnez +1 points
```

Dans ce cas, l'utilisateur prénommé Mathieu a aligné quatre pions d'affilée, il est le joueur gagnant, un message pour annoncer l'ajout de points est aussi affiché.

CONCLUSION :

En conclusion, ce projet Python nous a permis de développer quatre jeux , allant du Puissance 4 au Jeu de Devinette. En travaillant en groupe, nous avons acquis des compétences cruciales allant de la conceptualisation à la création d'un projet complet. Tous les jeux ont proposé des défis uniques, renforçant nos connaissances en programmation, en résolution de problèmes et en collaboration. La gestion de score nous a aussi appris comment les fichiers binaires fonctionnent ainsi que comment les utiliser de façon correcte.