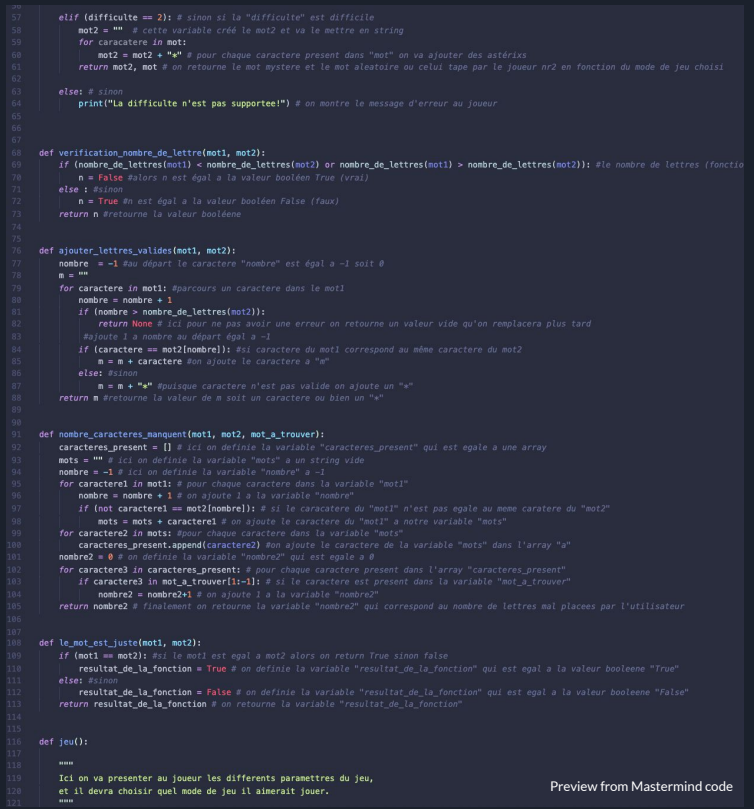
Projet Mastermind

1G-NSI-a

Océane, Mathias & Eduard

Sommaire - Projet Mastermind

- Présentation du projet
- Cahier des charges
- Les fonctions de Mathias
- Les fonctions d'Océane
- Les fonctions d'Eduard



```
    elif (difficulte == 2): # sinon si la "difficulte" est difficile
        mot2 = ""  # cette variable créé le mot2 et va le mettre en string
        for caractere in mot:
            mot2 = mot2 + "*" # pour chaque caractere present dans "mot" on va ajouter des astérixs
        return mot2, mot # on retourne le mot mystere et le mot aleatoire ou celui tape par le joueur nr2 en fonction du mode de jeu choisi

    else: # sinon
        print("La difficulte n'est pas supportee!") # on montre le message d'erreur au joueur


def verification_nombre_de_lettre(mot1, mot2):
    if (nombre_de_lettres(mot1) < nombre_de_lettres(mot2) or nombre_de_lettres(mot1) > nombre_de_lettres(mot2)): #le nombre de lettres (fonctio
        n = False #alors n est égal a la valeur booléen True (vrai)
    else : #sinon
        n = True #n est égal a la valeur booléen False (faux)
    return n #retourne la valeur booléene


def ajouter_lettres_valides(mot1, mot2):
    nombre  = -1 #au départ le caractere "nombre" est égal a -1 soit 0
    m = ""
    for caractere in mot1: #parcours un caractere dans le mot1
        nombre = nombre + 1
        if (nombre > nombre_de_lettres(mot2)):
            return None # ici pour ne pas avoir une erreur on retourne un valeur vide qu'on remplacera plus tard
        #ajoute 1 a nombre au départ égal a -1
        if (caractere == mot2[nombre]): #si caractere du mot1 correspond au même caractere du mot2
            m = m + caractere #on ajoute le caractere a "m"
        else: #sinon
            m = m + "*" #puisque caractere n'est pas valide on ajoute un "*"
    return m #retourne la valeur de m soit un caractere ou bien un "*"


def nombre_caracteres_manquent(mot1, mot2, mot_a_trouver):
    caracteres_present = [] # ici on definie la variable "caracteres_present" qui est egale a une array
    mots = "" # ici on definie la variable "mots" a un string vide
    nombre = -1 # ici on definie la variable "nombre" a -1
    for caractere1 in mot1: # pour chaque caractere dans la variable "mot1"
        nombre = nombre + 1 # on ajoute 1 a la variable "nombre"
        if (not caractere1 == mot2[nombre]): # si le caracatere du "mot1" n'est pas egale au meme caratere du "mot2"
            mots = mots + caractere1 # on ajoute le caractere du "mot1" a notre variable "mots"
    for caractere2 in mots: #pour chaque caractere dans la variable "mots"
        caracteres_present.append(caractere2) #on ajoute le caractere de la variable "mots" dans l'array "a"
    nombre2 = 0 # on definie la variable "nombre2" qui est egale a 0
    for caractere3 in caracteres_present: # pour chaque caractere present dans l'array "caracteres_present"
        if caractere3 in mot_a_trouver[1:-1]: # si le caractere est present dans la variable "mot_a_trouver"
            nombre2 = nombre2+1 # on ajoute 1 a la variable "nombre2"
    return nombre2 # finalement on retourne la variable "nombre2" qui correspond au nombre de lettres mal placees par l'utilisateur


def le_mot_est_juste(mot1, mot2):
    if (mot1 == mot2): #si le mot1 est egal a mot2 alors on return True sinon false
        resultat_de_la_fonction = True # on definie la variable "resultat_de_la_fonction" qui est egal a la valeur booleene "True"
    else: #sinon
        resultat_de_la_fonction = False # on definie la variable "resultat_de_la_fonction" qui est egal a la valeur booleene "False"
    return resultat_de_la_fonction # on retourne la variable "resultat_de_la_fonction"


def jeu():

    """
    Ici on va presenter au joueur les differents parametres du jeu,
    et il devra choisir quel mode de jeu il aimerait jouer.
    """
```

Preview from Mastermind code

Présentation du projet Mastermind

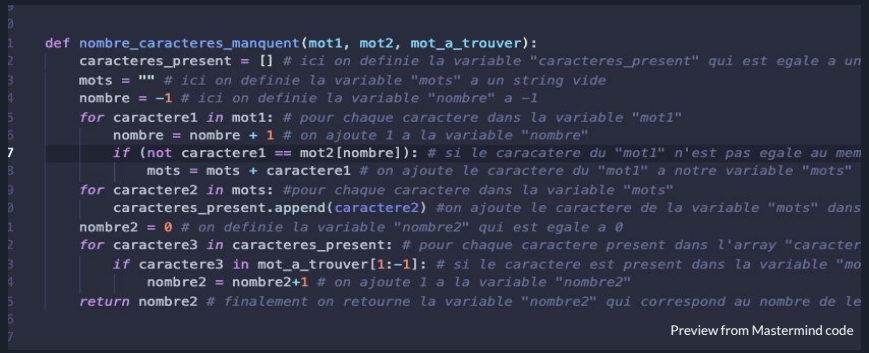
Pour ce projet on a dû implémenter une variante du jeu Mastermind avec des chaînes de caractères en Python.
Pour cela nous avons dû suivre quelques étapes:

- L'ordinateur choisit un mot aléatoirement parmi une liste de mots lus dans un fichier.
- Il affiche à l'utilisateur le mot mystère avec seulement la première et la dernière lettre du mot.
- L'utilisateur propose un mot. Le mot doit contenir le bon nombre de lettres, et les lettres déjà validées. Sinon l'ordinateur refuse l'essai.
- Ensuite l'ordinateur ajoute au mot mystère chaque lettre valide au bon endroit et indique le nombre de lettres présentes dans le mot, mais mal placées.
- L'utilisateur propose encore un mot jusqu'à trouver le bon.
- L'utilisateur a au maximum 10 tentatives.

Présentation du projet Mastermind

Puis on a eu des évolutions à faire sur le projet comme par exemple :

- Ajouter un mode multijoueur (en plus du mode solo) permettant à un second joueur de proposer le mot à deviner.
- Ajouter un mode difficile ou aucune lettre n'est montrée dès le départ, le joueur est autorisé à jouer des mots non vladies (tailles différentes et lettres invalides) et il a moins de tentatives possibles.



```
def nombre_caracteres_manquent(mot1, mot2, mot_a_trouver):
    caracteres_present = [] # ici on definie la variable "caracteres_present" qui est egale a un
    mots = "" # ici on definie la variable "mots" a un string vide
    nombre = -1 # ici on definie la variable "nombre" a -1
    for caractere1 in mot1: # pour chaque caractere dans la variable "mot1"
        nombre = nombre + 1 # on ajoute 1 a la variable "nombre"
        if (not caractere1 == mot2[nombre]): # si le caracatere du "mot1" n'est pas egale au mem
            mots = mots + caractere1 # on ajoute le caractere du "mot1" a notre variable "mots"
    for caractere2 in mots: #pour chaque caractere dans la variable "mots"
        caracteres_present.append(caractere2) #on ajoute le caractere de la variable "mots" dans
    nombre2 = 0 # on definie la variable "nombre2" qui est egale a 0
    for caractere3 in caracteres_present: # pour chaque caractere present dans l'array "caracter
        if caractere3 in mot_a_trouver[1:-1]: # si le caractere est present dans la variable "mo
            nombre2 = nombre2+1 # on ajoute 1 a la variable "nombre2"
    return nombre2 # finalement on retourne la variable "nombre2" qui correspond au nombre de le
```

Preview from Mastermind code

Cahier des charges - Projet Mastermind

Liste des fonctionnalités prévues dans le programme:

➥ L'ordinateur doit choisir un mot aléatoirement parmi une liste de mots lus dans un fichier.

- Une fonction intitulée **"mot_aleatoire"**: le rôle de cette fonction est de parcourir le fichier avec les mots, et en retourner un parmi eux aléatoirement.

➥ Il affiche à l'utilisateur un mot mystère avec seulement la première et la dernière lettre du mot révélée.

- Une fonction intitulée "**nombre_de_lettres**" avec comme paramètre "**mot**": le rôle de cette fonction est de compter toutes les lettres présentes dans un mot, le mot devra être indiqué au tant que paramètre quand on fera appel à cette fonction, elle retourne le nombre de lettres présentes dans "**mot**".
- Une fonction intitulée "**mot_mystere**" avec comme paramètre **"difficulte"** et **"mode"**: le rôle de cette fonction est de prendre le mot retourner par la fonction "**mot_aleatoire**" ou introduit par le **"joueur nr2"** en **fonction du mode de jeu choisi** et devra transformer toutes les lettres sauf la première et la dernière en "*" si le paramètre **"difficulte"** est égal à "facile", sinon si le paramètre **"difficulte"** est égal à **"difficile"** toutes les lettres seront transformés en "*", elle retourne le mot mystère et le mot aléatoire obtenu de la fonction **"mot_aleatoire"**.

➥ Le mot que l'utilisateur a proposé doit contenir le bon nombre de lettres, et les lettres déjà validées (la première et la dernière). Sinon l'ordinateur refuse l'essai.

- Une fonction intitulée "**verfication_nombre_de_lettre**" avec comme paramètre "**mot1** et **mot2**": le rôle de cette fonction est de vérifier si le nombres de lettres du "**mot1**" est supérieur ou inférieur à celui du "**mot2**", si c'est le cas la fonction retourne une valeur booléen **vrai** sinon une valeur booleen **fausse**.

➥ L'ordinateur ajoute au mot mystère chaque lettre valide au bon endroit.

- Une fonction intitulée "**ajouter_lettres_valides**" avec comme paramètre "**mot1** et **mot2**": le rôle de cette fonction est d'ajouter les lettres valides du "**mot1**" au "**mot2**" et de retourner un nouveau mot mystère.

Cahier des charges - Projet Mastermind

➥ Indiquer le nombre de lettres présentes dans le mot, mais mal placées.
- Une fonction intitulée “**le_mot_est_juste**” avec comme paramètre “**mot1** et **mot2**” : le rôle de cette fonction est de vérifier si le “**mot1**” est égal au “**mot2**” si c'est le cas la fonction retourne une valeur booléen **vrai** sinon une valeur booleen **fausse**.
- Une fonction intitulée “**nombre_caracteres_manquent**” avec comme paramètre “**mot_utilisateur**, **nouveau_mot** et **mot_original**”: le rôle de cette fonction est de compter le nombre de lettres valides mais mal placées et le retourner.

➥ L'exécution du jeu en lui-même.
- Une fonction intitulée “**jeu**”: le rôle de cette fonction est de faire appel aux autres fonctions, à exécuter le jeu mastermind, a envoyer des informations aux joueurs et à traiter toutes les informations.

La répartitions des tâches :

Océane - fonction
Mathias - fonction
Eduard - fonction

Fonctions de Mathias - Projet Mastermind

main.py > ...

```python
import random


def mot_aleatoire(): # creation de la fonction "mot_aleatoire" pour recuperer le mot aleatoire
    fichier=open("dictionnaire.txt","r") # creation de la variable "fichier" qui ouvre le ficher en mode lecture
    mot = "" # creation de la variable "mot" egale a un string
    mots=[] # creation d'une array "mots"
    nombre_de_mots=-1 # creation de la variable "nombre_de_mots" egale a -1 car l'array commence a 0
    for ligne in fichier.readlines(): # cette fonction va lire tout les lignes du fichier
        nombre_de_mots=nombre_de_mots+1 # il va compter toute les lignes
        mots.append(ligne[:-1])#il va prendre tout les lettres du mots sauf la derniere
    fichier.close() # ferme le fichier
    nombre_aleatoire = random.randint(0, nombre_de_mots) # on va choisir un nombre aléatoire entre 0 et le nombre de lignes
    mot = mot + str(mots[nombre_aleatoire])#il va mettre le mot en str
    return mot # il va retourner le mot
```

Le rôle de cette fonction est de compter toutes les lettres présentes dans un mot, le mot devra être indiqué au tant que paramètre quand on fera appel à cette fonction, elle retourne le nombre de lettres présentes dans “**mot**”.

Fonctions de Mathias - Projet Mastermind

main.py > ...

```
"""
Quand le mode = 1 signifie que le jeu va etre solo.
Quand le mode = 2 signigie que le jeu va etre en multijoueur.
Quand la difficulte = 1 signifie que le jeu va etre facile.
Quand la difficulte = 2 signigie que le jeu va etre difficile.
"""
def mot_mystere(difficulte, mode):

    """
    Dans un premiere partie on va verifier
    si le jeu est en solo ou si il est en multijoueur.
    """
    if (mode == 1): #si le mode est solo alors
        mot =  mot_aleatoire() #on fait appel a la function mot_aleatoire
    elif (mode == 2): #sinon si le mode est multijoueur on demande le mot au joueur nr2
        mot = input("Joueur nr 2, veuillez introduire le mot a deviner en majuscule:") #on demande le mot au joueur 2
    else: #sinon message d'erreur
        print("Ce mode de jeu n'est pas supporter: Solo/Multijoueur")
```

Le rôle de cette fonction est de prendre le mot retourner par la fonction “**mot_aleatoire**” ou introduit par le **“joueur nr2**” en **fonction du mode de jeu choisi** et devra transformer toutes les lettres sauf la première et la dernière en “*” si le paramètre "**difficulte**" est égal à “facile”, sinon si le paramètre "**difficulte**" est égal à "**difficile**" toutes les lettres seront transformés en “*”, elle retourne le mot mystère et le mot aléatoire obtenu de la fonction "**mot_aleatoire**”.

Fonctions de Mathias - Projet Mastermind

main.py > ...

```
    """
    Dans cette deuxieme partie on va verifier
    si le jeu est facile ou difficile.
    """
    if (difficulte == 1): # si la "difficulte" est facile
        mot2 = " " # cette variable créé le mot2 et va le mettre en string
        for caractere in mot: # parcourir chaque caractere dans la variable "mot"
            if caractere in mot[1 : -1]: # si la lettre est comprise entre la 1ere lettre exclue et la derniere exclue on ajoute une *
            genre x appartient a ]1;-1[
                mot2 = mot2 + "*" # va prendre le mot 2 + des astérixs sauf pour la premiere et derniere lettre
            else: # sinon
                mot2 = mot2+caractere # on ajoute la lettre ici representer par la variable "i" au mot
        return mot2, mot # on retourne le mot mystere et le mot de la fonction mot_aleatoire

    elif (difficulte == 2): # sinon si la "difficulte" est difficile
        mot2 = ""  # cette variable créé le mot2 et va le mettre en string
        for caracatere in mot:
            mot2 = mot2 + "*" # pour chaque caractere present dans "mot" on va ajouter des astérixs
        return mot2, mot # on retourne le mot mystere et le mot aleatoire ou celui tape par le joueur nr2 en fonction du mode de jeu
        choisi

    else: # sinon
        print("La difficulte n'est pas supportee!") # on montre le message d'erreur au joueur
```

Fonctions de Océane - Projet Mastermind

main.py > ...

```python
def nombre_de_lettres(mot):
    m=-1 #au départ le nombre de caractere "m" est égal a 0
    for caractere in mot: #parcours un caractere dans le mot
        m=m+1 #ajoute 1 au nombre de lettre
    return m #retourne le nombre de lettre (résultat)

```

Le rôle de cette fonction est de compter toutes les lettres présentes dans un mot, le mot devra être indiqué au tant que paramètre quand on fera appel à cette fonction, elle retourne le nombre de lettres présentes dans “**mot**”.

Fonctions de Océane - Projet Mastermind

main.py > ...

```
68  def verification_nombre_de_lettre(mot1, mot2):
69      if (nombre_de_lettres(mot1) < nombre_de_lettres(mot2) or nombre_de_lettres(mot1) > nombre_de_lettres(mot2)): #le nombre de lettres
        (fonction1) du mot1 < ou > au nombre de lettres du mot2
70          n = False #alors n est égal a la valeur booléen True (vrai)
71      else : #sinon
72          n = True #n est égal a la valeur booléen False (faux)
73      return n #retourne la valeur booléene
```

Le rôle de cette fonction est de vérifier si le nombres de lettres du “**mot1**” est supérieur ou inférieur à celui du “**mot2**”, si c’est le cas la fonction retourne une valeur booléen **vrai** sinon une valeur booleen **fausse**.

Fonctions de Océane - Projet Mastermind

main.py > ...

```python
def ajouter_lettres_valides(mot1, mot2):
    nombre  = -1 #au départ le caractere "nombre" est égal a -1 soit 0
    m = ""
    for caractere in mot1: #parcours un caractere dans le mot1
        nombre = nombre + 1
        if (nombre > nombre_de_lettres(mot2)):
            return None # ici pour ne pas avoir une erreur on retourne un valeur vide qu'on remplacera plus tard
         #ajoute 1 a nombre au départ égal a -1
        if (caractere == mot2[nombre]): #si caractere du mot1 correspond au même caractere du mot2
            m = m + caractere #on ajoute le caractere a "m"
        else: #sinon
            m = m + "*" #puisque caractere n'est pas valide on ajoute un "*"
    return m #retourne la valeur de m soit un caractere ou bien un "*"
```

Le rôle de cette fonction est d'ajouter les lettres valides du “**mot1**” au “**mot2**” et de retourner un nouveau mot mystère.

Fonctions de Eduard - Projet Mastermind

main.py > ...

```python

def le_mot_est_juste(mot1, mot2):
    if (mot1 == mot2): #si le mot1 est egal a mot2 alors on return True sinon false
        resultat_de_la_fonction = True # on definie la variable "resultat_de_la_fonction" qui est egal a la valeur booleene "True"
    else: #sinon
        resultat_de_la_fonction = False # on definie la variable "resultat_de_la_fonction" qui est egal a la valeur booleene "False"
    return resultat_de_la_fonction # on retourne la variable "resultat_de_la_fonction"

```

Le rôle de cette fonction est de vérifier si le “**mot1**” est égal au “**mot2**” si c’est le cas la fonction retourne une valeur booléen **vrai** sinon une valeur booleen **fausse**.

Fonctions de Eduard - Projet Mastermind

main.py > ...

```python
def nombre_caracteres_manquent(mot1, mot2, mot_a_trouver):
    caracteres_present = [] # ici on definie la variable "caracteres_present" qui est egale a une array
    mots = "" # ici on definie la variable "mots" a un string vide
    nombre = -1 # ici on definie la variable "nombre" a -1
    for caractere1 in mot1: # pour chaque caractere dans la variable "mot1"
        nombre = nombre + 1 # on ajoute 1 a la variable "nombre"
        if (not caractere1 == mot2[nombre]): # si le caracatere du "mot1" n'est pas egale au meme caratere du "mot2"
            mots = mots + caractere1 # on ajoute le caractere du "mot1" a notre variable "mots"
    for caractere2 in mots: #pour chaque caractere dans la variable "mots"
        caracteres_present.append(caractere2) #on ajoute le caractere de la variable "mots" dans l'array "a"
    nombre2 = 0 # on definie la variable "nombre2" qui est egale a 0
    for caractere3 in caracteres_present: # pour chaque caractere present dans l'array "caracteres_present"
        if caractere3 in mot_a_trouver[1:-1]: # si le caractere est present dans la variable "mot_a_trouver"
            nombre2 = nombre2+1 # on ajoute 1 a la variable "nombre2"
    return nombre2 # finalement on retourne la variable "nombre2" qui correspond au nombre de lettres mal placees par l'utilisateur

```

Le rôle de cette fonction est de compter le nombre de lettres valides mais mal placées et le retourner.

Fonctions de Eduard - Projet Mastermind

main.py > ...

```python
def jeu():

    """
    Ici on va presenter au joueur les differents parametres du jeu,
    et il devra choisir quel mode de jeu il aimerait jouer.
    """
    menu_selection = True
    while menu_selection == True:
        print("Modes de jeu disponibles:\n 1 - Solo | Facile\n 2 - Solo | Difficile\n 3 - Multijoueur | Facile\n 4 - Multijoueur | Difficile")
        mode_de_jeu = int(input("Veuillez choisir le nombre correspondant au mode du jeu auquel vous voulez jouer:"))
        if (mode_de_jeu == 1):
            mode = 1
            difficulte = 1
            menu_selection = False
        elif (mode_de_jeu == 2):
            mode = 1
            difficulte = 2
            menu_selection = False
        elif (mode_de_jeu == 3):
            mode = 2
            difficulte = 1
            menu_selection = False
        elif (mode_de_jeu == 4):
            mode = 2
            difficulte = 2
            menu_selection = False
        else:
            print("Le mode de jeu choisi est n'existe pas.")
```

Le rôle de cette fonction est de faire appel aux autres fonctions, à exécuter le jeu mastermind, a envoyer des informations aux joueurs et à traiter toutes les informations.

Fonctions de Eduard - Projet Mastermind

main.py > ...

```
145     resultat_de_la_fonction_m_myst = mot_mystere(difficulte, mode) # on enregistre les 2 resultats de la fonction "mot_mystere" dans
        la variable "resultat_de_la_fonction_m_myst"
146     mot_myst = resultat_de_la_fonction_m_myst[1] # on enregistre le deuxieme resultat de la fonction "mot_mystere" dans la variable
        "mot_myst"
147     nombre_de_tentatives = 0 # on definie la variable "nombre_de_tentatives" qui est egale  0
148     print("TEST MOT MYST", mot_myst)
149
150     """
151     Ici on choisi les boucles que le programme va utiliser et le nombre
152     de tentatives en fonction du choix de mode de jeu du joueur.
153     """
154     if (difficulte == 1):
155         boucle1 = True
156         boucle2 = False
157         boucle3 = False
158         boucle4 = False
159         tentatives = 10
160     elif (difficulte == 2):
161         boucle1 = False
162         boucle2 = False
163         boucle3 = False
164         boucle4 = True
165         tentatives = 7
166
```

Fonctions de Eduard - Projet Mastermind

```
    """
    Ici on montre pour la premiere fois au joueur le mot mystere
    et on lui demande d'entrer un mot pour rentrer dans la boucle 1
    ou dans la boucle 4 en fonction du mode de jeu choisi.
    """
    print("Mot mystere :", resultat_de_la_fonction_m_myst[0])
    mot_utilisateur = input("Proposez un mot :")

    if (nombre_de_tentatives == tentatives): # si le nombre de tentatives est egal a la limite de tentatives
        print("Vous avez perdu perdu.") # ici on averti le joueur qu'il a perdu
        # on desactive toutes les boucles
        boucle1 = False
        boucle2 = False
        boucle3 = False
        boucle4 = False
    else: # sinon
        nombre_de_tentatives = nombre_de_tentatives + 1 # on ajoute une tentative de plus au nombre de tentatives
```

Fonctions de Eduard - Projet Mastermind

main.py > ...

```
186     """
187     Dans cette premiere boucle nous allons verifier le nombre de lettres que l'utlisateur
188     a entre, correspond bien au nombre de lettres du mot mystere a trouver, sinon on lui redemande
189     de rentrer le mot jusqu'a ce que le nombre de lettres corresponde.
190     """
191     while boucle1 == True:
192
193         if (verification_nombre_de_lettre(mot_utilisateur, mot_myst) == False): # si notre fonction "verification_nombre_de_lettres"
        retourne une valuer booleen "False"
194             print("Votre mot contient", nombre_de_lettres(mot_utilisateur) + 1, "lettres alors qu'il en faut", nombre_de_lettres
            (mot_myst)+1) # on montre l'erreur au joueur
195             mot_utilisateur = input("Proposez un mot :") #on redemande au joueur de resaisir le mot
196
197             """
198             Ici nous verifions que le joueur ne
199             depasse pas le nombre de tentatives.
200             """
201             if (nombre_de_tentatives == tentatives): # si le nombre de tentatives est egal a la limite de tentatives
202                 print("Vous avez perdu.") # ici on averti le joueur qu'il a perdu
203                 break # ici on arrete toute la fonction ici.
204             else:  # sinon
205                 nombre_de_tentatives = nombre_de_tentatives + 1 # on ajoute une tentative de plus au nombre de tentatives
206
207         elif (verification_nombre_de_lettre(mot_utilisateur, mot_myst) == True): # sinon si notre fonction
        "verification_nombre_de_lettres" retourne une valeur booleen "True"
208             boucle1 = False # on sort de la boucle 1
209             boucle2 = True # on rentre dans la boucle 2
210
211
```

Fonctions de Eduard - Projet Mastermind

main.py > jeu

```
    """
    Dans cette deuxieme boucle nous allons verifier que la premiere lettre que l'utlisateur
    a entre, correspond bien a la premiere lettre du mot mystere a trouver, sinon on lui redemande
    de rentrer le mot jusqu'a ce que le nombre de lettres corresponde et de meme pour la derniere lettre.
    """
    while boucle2 == True:

        if (not mot_utilisateur[0] == mot_myst[0]):  #si la premiere lettre du "mot_utilisateur" n'es pas egale a la premiere lettre du "mot_myst"
            print("La 1ere lettre de votre mot est un", mot_utilisateur[0], "alors que ca devrait etre un", mot_myst[0]) # on montre l'erreur au joueur
            mot_utilisateur = input("Proposez un mot :")

            if (nombre_de_tentatives == tentatives):
                print("Vous avez perdu.")
                break
            else:
                nombre_de_tentatives = nombre_de_tentatives + 1

        if (not mot_utilisateur[nombre_de_lettres(mot_utilisateur)] == mot_myst[nombre_de_lettres(mot_myst)]): #si la derniere lettre du "mot_utilisateur" n'es pas egale a la derniere lettre du "mot_myst"
            print("La derniere lettre de votre mot est un", mot_utilisateur[nombre_de_lettres(mot_utilisateur)], "alors que ca devrait etre", mot_myst[nombre_de_lettres(mot_myst)])
            mot_utilisateur = input("Proposez un mot :")

            if (nombre_de_tentatives == tentatives):
                print("Vous avez perdu.")
                break
            else:
                nombre_de_tentatives = nombre_de_tentatives + 1

        elif (mot_utilisateur[0] == mot_myst[0] and mot_utilisateur[nombre_de_lettres(mot_utilisateur)] == mot_myst[nombre_de_lettres(mot_myst)]): #sinon si la premiere et la derniere lettre du "mot_utilisateur" correspondent a la premiere et la derniere lettre du "mot_myst"
            boucle2 = False # on sort de la boucle 2
            boucle3 = True # on rentre dans la boucle 3
```

Fonctions de Eduard - Projet Mastermind

main.py > jeu

```
    """
    Dans cette deuxieme boucle nous allons verifier que la premiere lettre que l'utlisateur
    a entre, correspond bien a la premiere lettre du mot mystere a trouver, sinon on lui redemande
    de rentrer le mot jusqu'a ce que le nombre de lettres corresponde et de meme pour la derniere lettre.
    """
    while boucle3 == True:

        nouveau_mot = ajouter_lettres_valides(mot_myst, mot_utilisateur) # on definie la variable "nouveau_mot" qui va contenir le mot mystere avec les lettres deja trouves par le joueur
        nombre_caracteres = nombre_caracteres_manquent(mot_utilisateur, nouveau_mot, mot_myst) # on definie la variable "nombre_caracteres" qui va contenir le resultat de la fonction "nombre_caracteres_manquent"

        if (le_mot_est_juste(mot_utilisateur, mot_myst) == False): # si la fonction "le_mot_est_juste" nous retourne une valeur booleen "False"
            if (nombre_de_tentatives == tentatives):
                print("Vous avez perdu.")
                break
            else:
                nombre_de_tentatives = nombre_de_tentatives + 1

            print("Mot mystere :", nouveau_mot) # on montrer le mot mystere au joueur avec les nouvelles lettres
            print("Lettres mal place :", nombre_caracteres) # on lui montre le nombre de lettres qu'il a mal placee
            mot_utilisateur = input("Proposez un mot :")


        elif(le_mot_est_juste(mot_utilisateur, mot_myst) == True): # sinon si la fonction "le_mot_est_juste" nous retourne une valeur booleen "True"
            boucle3=False # on sort de la boucle 3
            print('Bravo tu as trouver le mot, nombre de tenatives:', nombre_de_tentatives) # on montre au joueur qu'il a gagner
```

Fonctions de Eduard - Projet Mastermind

main.py > ...

```
    """
    Cette 4eme boucle est pour le mode difficile ou on n'a plus
    besoin de la 1er, 2eme et de la 3eme boucle pour faire les verifications et les ajouts de lettres.
    """
    mot_mystere_afficher = resultat_de_la_fonction_m_myst[0] # on definie la variable "mot_mystere_afficher" qui contient le mot_mystere
    while boucle4 == True:
        nouveau_mot = ajouter_lettres_valides(mot_myst, mot_utilisateur) # on definie la variable "nouveau_mot" qui va contenir le mot mystere avec les lettres deja trouves par le joueur

        if (le_mot_est_juste(mot_utilisateur, mot_myst) == False): # si la fonction "le_mot_est_juste" nous retourne une valeur booleen "False"

            if (nombre_de_tentatives == tentatives):
                print("Vous avez perdu.")
                break
            else:
                nombre_de_tentatives = nombre_de_tentatives + 1

            if (nouveau_mot == None): # si le joueur ne respecte pas le nombre de "*" qui correspond au nombre de lettres du mot a trouver la fonction "nouveau_mot" va retourner "None" qui est un valeur vide
                print("Mot mystere :", mot_mystere_afficher) # on affiche le mot mystere qui va etre le resultat de la fonction "mot_mystere"
            else: #sinon si le joueur a trouver des lettres valides
                print("Mot mystere :", nouveau_mot) # on affiche la variable "nouveau_mot"
                mot_mystere_afficher = nouveau_mot # on definie la variable "mot_mystere_afficher" a la variable "nouveau_mot"
            mot_utilisateur = input("Proposez un mot :")
        elif (le_mot_est_juste(mot_utilisateur, mot_myst) == True): # sinon si la fonction "le_mot_est_juste" nous retourne une valeu booleen "True"
            print('Bravo tu as trouver le mot, nombre de tenatives:', nombre_de_tentatives) # on montre au joueur qu'il a gagner
            boucle4 = False #on sort de la boucle 4

jeu()
```

FIN

```
    return n #retourne la valeur booleene


def ajouter_lettres_valides(mot1, mot2):
    nombre  = -1 #au départ le caractere "nombre" est égal a -1 soit 0
    m = ""
    for caractere in mot1: #parcours un caractere dans le mot1
        nombre = nombre + 1
        if (nombre > nombre_de_lettres(mot2)):
            return None # ici pour ne pas avoir une erreur on retourne un valeur vide qu'on remplacera plus tard
         #ajoute 1 a nombre au départ égal a -1
        if (caractere == mot2[nombre]): #si caractere du mot1 correspond au même caractere du mot2
            m = m + caractere #on ajoute le caractere a "m"
        else: #sinon
            m = m + "*" #puisque caractere n'est pas valide on ajoute un "*"
    return m #retourne la valeur de m soit un caractere ou bien un "*"


def nombre_caracteres_manquent(mot1, mot2, mot_a_trouver):
    caracteres_present = [] # ici on definie la variable "caracteres_present" qui est egale a une array
    mots = "" # ici on definie la variable "mots" a un string vide
    nombre = -1 # ici on definie la variable "nombre" a -1
    for caractere1 in mot1: # pour chaque caractere dans la variable "mot1"
        nombre = nombre + 1 # on ajoute 1 a la variable "nombre"
        if (not caractere1 == mot2[nombre]): # si le caracatere du "mot1" n'est pas egale au meme caratere du "mot2"
            mots = mots + caractere1 # on ajoute le caractere du "mot1" a notre variable "mots"
    for caractere2 in mots: #pour chaque caractere dans la variable "mots"
        caracteres_present.append(caractere2) #on ajoute le caractere de la variable "mots" dans l'array "a"
    nombre2 = 0 # on definie la variable "nombre2" qui est egale a 0
    for caractere3 in caracteres_present: # pour chaque caractere present dans l'array "caracteres_present"
        if caractere3 in mot_a_trouver[1:-1]: # si le caractere est present dans la variable "mot_a_trouver"
            nombre2 = nombre2+1 # on ajoute 1 a la variable "nombre2"
    return nombre2 # finalement on retourne la variable "nombre2" qui correspond au nombre de lettres mal placees par l'utilisateur
```

Preview from Mastermind code