



# 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

```
57 elif (difficulte == 2): # sinon si la "difficulte" est difficile
58     mot2 = "" # cette variable crée le mot2 et va le mettre en string
59     for caractere in mot1:
60         mot2 = mot2 + "*" # pour chaque caractere present dans "mot1" on va ajouter des astérisques
61     return mot2, mot # on retourne le mot mystere et le mot aleatoire au celui tape par le joueur nr2 en fonction du mode de jeu choisi
62
63 else: # sinon
64     print("La difficulté n'est pas supportee!") # on montre le message d'erreur au joueur
65
66
67
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 (fonction)
70         n = False # alors n est égal a la valeur boolean True (vrai)
71     else: # sinon
72         n = True # n est égal a la valeur boolean False (faux)
73     return n # retourne la valeur booleane
74
75
76 def ajouter_lettres_valides(mot1, mot2):
77     nombre = -1 # au depart le caractere "nombre" est égal a -1 soit 0
78     m = ""
79     for caractere in mot1: # parcours un caractere dans le mot1
80         nombre = nombre + 1
81         if (nombre > nombre_de_lettres(mot2)):
82             return None # ici pour ne pas avoir une erreur on retourne un valeur vide qu'on remplacera plus tard
83             # ajoute 1 a nombre au depart égal a -1
84         if (caractere == mot2[nombre]): # si caractere du mot1 correspond au même caractere du mot2
85             m = m + caractere # on ajoute le caractere a "m"
86         else: # sinon
87             m = m + "*" # puisque caractere n'est pas valide on ajoute un "*"
88     return m # retourne la valeur de m soit un caractere ou bien un "*"
89
90
91 def nombre_caracteres_manquant(mot1, mot2, mot_a_trouver):
92     caracteres_present = [] # ici on definit la variable "caracteres_present" qui est egale a une array
93     mots = "" # ici on definit la variable "mots" a un string vide
94     nombre = -1 # ici on definit la variable "nombre" a -1
95     for caractere1 in mot1: # pour chaque caractere dans la variable "mot1"
96         nombre = nombre + 1 # on ajoute 1 a la variable "nombre"
97         if (not caractere1 == mot2[nombre]): # si le caractere du "mot1" n'est pas egale au même caractere du "mot2"
98             mots = mots + caractere1 # on ajoute le caractere du "mot1" a notre variable "mots"
99     for caractere2 in mots: # pour chaque caractere dans la variable "mots"
100         caracteres_present.append(caractere2) # on ajoute le caractere de la variable "mots" dans l'array "a"
101     nombre2 = 0 # on definit la variable "nombre2" qui est egale a 0
102     for caractere3 in caracteres_present: # pour chaque caractere present dans l'array "caracteres_present"
103         if caractere3 in mot_a_trouver[1:-1]: # si le caractere est present dans la variable "mot_a_trouver"
104             nombre2 = nombre2 + 1 # on ajoute 1 a la variable "nombre2"
105     return nombre2 # finalement on retourne la variable "nombre2" qui correspond au nombre de lettres mal placees par l'utilisateur
106
107
108 def le_mot_est_juste(mot1, mot2):
109     if (mot1 == mot2): # si le mot1 est egal a mot2 alors on return True sinon False
110         resultat_de_la_fonction = True # on definit la variable "resultat_de_la_fonction" qui est egal a la valeur booleane "True"
111     else: # sinon
112         resultat_de_la_fonction = False # on definit la variable "resultat_de_la_fonction" qui est egal a la valeur booleane "False"
113     return resultat_de_la_fonction # on retourne la variable "resultat_de_la_fonction"
114
115
116 def jeu():
117     """
118     Ici on va presenter au joueur les differents parametres du jeu,
119     et il devra choisir quel mode de jeu il aimerait jouer.
120     """
```

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 valides (tailles différentes et lettres invalides) et il a moins de tentatives possibles.

```
1 def nombre_caracteres_manquant(mot1, mot2, mot_a_trouver):
2     caracteres_present = [] # ici on definie la variable "caracteres_present" qui est egale a un
3     mots = "" # ici on definie la variable "mots" a un string vide
4     nombre = -1 # ici on definie la variable "nombre" a -1
5     for caractere1 in mot1: # pour chaque caractere dans la variable "mot1"
6         nombre = nombre + 1 # on ajoute 1 a la variable "nombre"
7         if (not caractere1 == mot2[nombre]): # si le caractere du "mot1" n'est pas egale au mem
8             mots = mots + caractere1 # on ajoute le caractere du "mot1" a notre variable "mots"
9         caracteres_present.append(caractere1) #on ajoute le caractere de la variable "mots" dans
10    nombre2 = 0 # on definie la variable "nombre2" qui est egale a 0
11    for caractere3 in caracteres_present: # pour chaque caractere present dans l'array "caracter
12        if caractere3 in mot_a_trouver[1:-1]: # si le caractere est present dans la variable "mo
13            nombre2 = nombre2+1 # on ajoute 1 a la variable "nombre2"
14    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 transformées 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 "verification\_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 boolean 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 répartition des tâches :

Océane - fonction

Mathias - fonction

Eduard - fonction

# Fonctions de Mathias - Projet Mastermind

main.py > ...

```
1  import random
2
3
4  def mot_aleatoire(): # creation de la fonction "mot_aleatoire" pour recuperer le mot aleatoire
5      fichier=open("dictionnaire.txt","r") # creation de la variable "fichier" qui ouvre le fichier en mode lecture
6      mot = "" # creation de la variable "mot" egale a un string
7      mots=[] # creation d'une array "mots"
8      nombre_de_mots=-1 # creation de la variable "nombre_de_mots" egale a -1 car l'array commence a 0
9      for ligne in fichier.readlines(): # cette fonction va lire tout les lignes du fichier
10         nombre_de_mots=nombre_de_mots+1 # il va compter toute les lignes
11         mots.append(ligne[:-1])#il va prendre tout les lettres du mots sauf la derniere
12     fichier.close() # ferme le fichier
13     nombre_aleatoire = random.randint(0, nombre_de_mots) # on va choisir un nombre aléatoire entre 0 et le nombre de lignes
14     mot = mot + str(mots[nombre_aleatoire])#il va mettre le mot en str
15     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 > ...
25  """
26  Quand le mode = 1 signifie que le jeu va etre solo.
27  Quand le mode = 2 signifie que le jeu va etre en multijoueur.
28  Quand la difficulte = 1 signifie que le jeu va etre facile.
29  Quand la difficulte = 2 signifie que le jeu va etre difficile.
30  """
31  def mot_mystere(difficulte, mode):
32
33      """
34      Dans un premiere partie on va verifier
35      si le jeu est en solo ou si il est en multijoueur.
36      """
37      if (mode == 1): #si le mode est solo alors
38          mot = mot_aleatoire() #on fait appel a la fonction mot_aleatoire
39      elif (mode == 2): #sinon si le mode est multijoueur on demande le mot au joueur nr2
40          mot = input("Joueur nr 2, veuillez introduire le mot a deviner en majuscule:") #on demande le mot au joueur 2
41      else: #sinon message d'erreur
42          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 > ...
44     """
45     Dans cette deuxieme partie on va verifier
46     si le jeu est facile ou difficile.
47     """
48     if (difficulte == 1): # si la "difficulte" est facile
49         mot2 = " " # cette variable créé le mot2 et va le mettre en string
50         for caractere in mot: # parcourir chaque caractere dans la variable "mot"
51             if caractere in mot[1 : -1]: # si la lettre est comprise entre la 1ere lettre exclue et la derniere exclue on ajoute une *
52                 mot2 = mot2 + "*" # va prendre le mot 2 + des astérixs sauf pour la premiere et derniere lettre
53             else: # sinon
54                 mot2 = mot2+caractere # on ajoute la lettre ici represente par la variable "i" au mot
55         return mot2, mot # on retourne le mot mystere et le mot de la fonction mot_aleatoire
56
57     elif (difficulte == 2): # sinon si la "difficulte" est difficile
58         mot2 = "" # cette variable créé le mot2 et va le mettre en string
59         for caractere in mot:
60             mot2 = mot2 + "*" # pour chaque caractere present dans "mot" on va ajouter des astérixs
61         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
62         choisi
63
64     else: # sinon
65         print("La difficulte n'est pas supportee!") # on montre le message d'erreur au joueur
```

# Fonctions de Océane - Projet Mastermind



main.py > ...

```
18  def nombre_de_lettres(mot):  
19      m=-1 #au départ le nombre de caractere "m" est égal a 0  
20      for caractere in mot: #parcours un caractere dans le mot  
21          m=m+1 #ajoute 1 au nombre de lettre  
22      return m #retourne le nombre de lettre (résultat)  
23
```

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 > ...

```
76 def ajouter_lettres_valides(mot1, mot2):
77     nombre = -1 #au départ le caractere "nombre" est égal a -1 soit 0
78     m = ""
79     for caractere in mot1: #parcours un caractere dans le mot1
80         nombre = nombre + 1
81         if (nombre > nombre_de_lettres(mot2)):
82             return None # ici pour ne pas avoir une erreur on retourne un valeur vide qu'on remplacera plus tard
83             #ajoute 1 a nombre au départ égal a -1
84         if (caractere == mot2[nombre]): #si caractere du mot1 correspond au même caractere du mot2
85             m = m + caractere #on ajoute le caractere a "m"
86         else: #sinon
87             m = m + "*" #puisque caractere n'est pas valide on ajoute un "*"
88     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 > ...

```
107
108 def le_mot_est_juste(mot1, mot2):
109     if (mot1 == mot2): #si le mot1 est egal a mot2 alors on return True sinon false
110         resultat_de_la_fonction = True # on definie la variable "resultat_de_la_fonction" qui est egal a la valeur booleene "True"
111     else: #sinon
112         resultat_de_la_fonction = False # on definie la variable "resultat_de_la_fonction" qui est egal a la valeur booleene "False"
113     return resultat_de_la_fonction # on retourne la variable "resultat_de_la_fonction"
114
```

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 > ...
91 def nombre_caracteres_manquant(mot1, mot2, mot_a_trouver):
92     caracteres_present = [] # ici on definie la variable "caracteres_present" qui est egale a une array
93     mots = "" # ici on definie la variable "mots" a un string vide
94     nombre = -1 # ici on definie la variable "nombre" a -1
95     for caractere1 in mot1: # pour chaque caractere dans la variable "mot1"
96         nombre = nombre + 1 # on ajoute 1 a la variable "nombre"
97         if (not caractere1 == mot2[nombre]): # si le caractere du "mot1" n'est pas egale au meme caractere du "mot2"
98             mots = mots + caractere1 # on ajoute le caractere du "mot1" a notre variable "mots"
99     for caractere2 in mots: #pour chaque caractere dans la variable "mots"
100         caracteres_present.append(caractere2) #on ajoute le caractere de la variable "mots" dans l'array "a"
101     nombre2 = 0 # on definie la variable "nombre2" qui est egale a 0
102     for caractere3 in caracteres_present: # pour chaque caractere present dans l'array "caracteres_present"
103         if caractere3 in mot_a_trouver[1:-1]: # si le caractere est present dans la variable "mot_a_trouver"
104             nombre2 = nombre2+1 # on ajoute 1 a la variable "nombre2"
105     return nombre2 # finalement on retourne la variable "nombre2" qui correspond au nombre de lettres mal placees par l'utilisateur
106
```

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 > jeu
116
117 def jeu():
118
119     """
120     Ici on va presenter au joueur les differents parametres du jeu,
121     et il devra choisir quel mode de jeu il aimerait jouer.
122     """
123     menu_selection = True
124     while menu_selection == True:
125         print("Modes de jeu disponibles:\n 1 - Solo | Facile\n 2 - Solo | Difficile\n 3 - Multijoueur | Facile\n 4 - Multijoueur | Difficile")
126         mode_de_jeu = int(input("Veuillez choisir le nombre correspondant au mode de jeu auquel vous voulez jouer:"))
127         if (mode_de_jeu == 1):
128             mode = 1
129             difficulte = 1
130             menu_selection = False
131         elif (mode_de_jeu == 2):
132             mode = 1
133             difficulte = 2
134             menu_selection = False
135         elif (mode_de_jeu == 3):
136             mode = 2
137             difficulte = 1
138             menu_selection = False
139         elif (mode_de_jeu == 4):
140             mode = 2
141             difficulte = 2
142             menu_selection = False
143         else:
144             print("Le mode de jeu choisi est n'existe pas.")
145
146     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"
147     mot_myst = resultat_de_la_fonction_m_myst[1] # on enregistre le deuxieme resultat de la fonction "mot_mystere" dans la variable "mot_myst"
148     nombre_de_tentatives = 0 # on definit la variable "nombre_de_tentatives" qui est egale 0
```

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 > jeu
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     boucle_difficile = False
157     tentatives = 10
158 elif (difficulte == 2):
159     boucle1 = False
160     boucle_difficile = True
161     tentatives = 7
162
163 """
164 Dans cette premiere boucle nous allons verifier le nombre de lettres que l'utilisateur
165 a entre, correspond bien au nombre de lettres du mot mystere a trouver, sinon on lui redemande
166 de rentrer le mot jusqu'a ce que le nombre de lettres corresponde, pareille pour la premiere et
167 derniere lettres, ensuite si l'utilisateur a bien rentre le bon format du mot.
168 """
169
170 premiere_demande = True # on definie la variable "premiere_demande" sur "True"
171 partie_ajout_letters = False # on definie la variable "partie_ajout_letters" sur "False"
172 while boucle1 == True:
173
174     """
175     Ici on montre pour la premiere fois au joueur le mot mystere
176     et on lui demande d'entrer un mot pour rentrer dans la boucle 1
177     ou dans la boucle 4 en fonction du mode de jeu choisi.
178     """
179     if premiere_demande == True: # si la variable "premiere_demande" est egale a "True"
180         print("Mot mystere :", resultat_de_la_fonction_m_myst[0])
181         mot_utilisateur = input("Proposez un mot :)") # on demande le mot a l'utilisateur
```



# Fonctions de Eduard - Projet Mastermind

```
main.py > jeu
183     """
184     Ici nous verifions que le joueur ne
185     depasse pas le nombre de tentatives.
186     """
187     if (nombre_de_tentatives == tentatives): # si le nombre de tentatives est egal a la limite de tentatives
188         print("Vous avez perdu perdu.") # ici on averti le joueur qu'il a perdu
189         break # on arrete tout
190     else: # sinon
191         nombre_de_tentatives = nombre_de_tentatives + 1 # on ajoute une tentative de plus au nombre de tentatives
192         print("Nombre tentatives:", nombre_de_tentatives)
193
194     if (verification_nombre_de_lettre(mot_utilisateur, mot_myst) == False): # si notre fonction "verification_nombre_de_lettres" retourne une valeur booleen "False"
195         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
196         creation_du_nouveau_mot = False # on definit la variable "creation_du_nouveau_mot" egale a "False"
197
```

# Fonctions de Eduard - Projet Mastermind

```
main.py > jeu
200 Dans cette partie nous allons verifier que la premiere lettre que l'utilisateur
201 a entre, correspond bien a la premiere lettre du mot mystere a trouver, sinon on lui redemande
202 de rentrer le mot jusqu'a ce que le nombre de lettres corresponde et de meme pour la derniere lettre.
203 """
204
205 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"
206     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
207     creation_du_nouveau_mot = False # on definie la variable "creation_du_nouveau_mot" egale a "False"
208
209 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"
210     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)])
211     creation_du_nouveau_mot = False # on definie la variable "creation_du_nouveau_mot" egale a "False"
212
213 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
214     "mot_utilisateur" correspondent a la premiere et la derniere lettre du "mot_myst"
215     partie_ajout_lettres = True # on active la variable "partie_ajout_lettres"
216     creation_du_nouveau_mot = True # on definie la variable "creation_du_nouveau_mot" egale a "True"
```

# Fonctions de Eduard - Projet Mastermind

```
main.py > jeu
218 """
219 Dans cette partie du programme le programme va ajouter les caracteres valides au mot mystere
220 qui est affiche au joueur et elle sert aussi a lui montrer le nombre de lettres qui sont
221 dans le mot mais qui sont mal placees.
222 """
223 if partie_ajout_lettres == True:
224     premiere_demande = False # on desactive la demande de mot a l'utilisateur car on va utiliser celle de cette partie
225
226     """
227     Cette verification permet de verifier a chaque fois le mot que l'utilisateur a introduit
228     pour ne pas avoir d'erreurs lors de l'ajout des nouvelles lettres au mot mystere.
229     """
230     if creation_du_nouveau_mot == True: # si la variable "creation_du_nouveau_mot" est egale a "True"
231         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
232         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
233         "nombre_caracteres_manquent"
234
235     if (le_mot_est_juste(mot_utilisateur, mot_myst) == False): # si la fonction "le_mot_est_juste" nous retourne une valeur booleen "False"
236         print("Mot mystere :", nouveau_mot) # on montrer le mot mystere au joueur avec les nouvelles lettres
237         print("Lettres mal place :", nombre_caracteres) # on lui montre le nombre de lettres qu'il a mal placee
238         mot_utilisateur = input("Proposez un mot :)")
239
240     elif(le_mot_est_juste(mot_utilisateur, mot_myst) == True): # sinon si la fonction "le_mot_est_juste" nous retourne une valeur booleen "True"
241         print('Bravo tu as trouver le mot, nombre de tentatives:', nombre_de_tentatives) # on montre au joueur qu'il a gagner
242         boucle1 = False # on sort de la boucle 1
```

# Fonctions de Eduard - Projet Mastermind

```
main.py > jeu
245
246 """
247 Cette boucle "difficile" est pour le mode difficile ou on n'a plus
248 besoin de faire les verifications et l'ajouts de lettres de la boucle precedente.
249 """
250 if boucle_difficile == True:
251     print("Mot mystere :", resultat_de_la_fonction_m_myst[0])
252     mot_utilisateur = input("Proposez un mot :")
253     mot_mystere_afficher = resultat_de_la_fonction_m_myst[0] # on definie la variable "mot_mystere_afficher" qui contient le mot_mystere
254
255 while boucle_difficile == True:
256     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
257
258     if (le_mot_est_juste(mot_utilisateur, mot_myst) == False): # si la fonction "le_mot_est_juste" nous retourne une valeur booleen "False"
259
260         if (nombre_de_tentatives == tentatives):
261             print("Vous avez perdu.")
262             boucle_difficile = False
263         else:
264             nombre_de_tentatives = nombre_de_tentatives + 1
265
266     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
267         print("Mot mystere :", mot_mystere_afficher) # on affiche le mot mystere qui va etre le resultat de la fonction "mot_mystere"
268     elif (nouveau_mot.count("*") > mot_mystere_afficher.count("*")): # si le joueur ne respecte pas les caracteres deja presents dans le mot la fonction "nouveau_mot" va supprimer les lettres deja
        ajoutes
269         print("Mot mystere :", mot_mystere_afficher) # on affiche le mot mystere qui va etre le resultat de la fonction "mot_mystere"
270     else: #sinon si le joueur a trouver des lettres valides
271         print("Mot mystere :", nouveau_mot) # on affiche la variable "nouveau_mot"
272         mot_mystere_afficher = nouveau_mot # on definie la variable "mot_mystere_afficher" a la variable "nouveau_mot"
273         mot_utilisateur = input("Proposez un mot :")
274     elif (le_mot_est_juste(mot_utilisateur, mot_myst) == True): # sinon si la fonction "le_mot_est_juste" nous retourne une valeur booleen "True"
275         print('Bravo tu as trouver le mot, nombre de tentatives:', nombre_de_tentatives) # on montre au joueur qu'il a gagner
276         boucle_difficile = False #on sort de la boucle 4
277
278 jeu()
```

# FIN

```
73     return n #retourne la valeur booléene
74
75
76 def ajouter_lettres_valides(mot1, mot2):
77     nombre = -1 #au départ le caractere "nombre" est égal a -1 soit 0
78     m = ""
79     for caractere in mot1: #parcours un caractere dans le mot1
80         nombre = nombre + 1
81         if (nombre > nombre_de_lettres(mot2)):
82             return None # ici pour ne pas avoir une erreur on retourne un valeur vide qu'on remplacera plus tard
83             #ajoute 1 a nombre au départ égal a -1
84         if (caractere == mot2[nombre]): #si caractere du mot1 correspond au même caractere du mot2
85             m = m + caractere #on ajoute le caractere a "m"
86         else: #sinon
87             m = m + "*" #puisque caractere n'est pas valide on ajoute un "*"
88     return m #retourne la valeur de m soit un caractere ou bien un "*"
89
90
91 def nombre_caracteres_manquent(mot1, mot2, mot_a_trouver):
92     caracteres_present = [] # ici on definie la variable "caracteres_present" qui est egale a une array
93     mots = "" # ici on definie la variable "mots" a un string vide
94     nombre = -1 # ici on definie la variable "nombre" a -1
95     for caractere1 in mot1: # pour chaque caractere dans la variable "mot1"
96         nombre = nombre + 1 # on ajoute 1 a la variable "nombre"
97         if (not caractere1 == mot2[nombre]): # si le caractere du "mot1" n'est pas egale au meme caractere du "mot2"
98             mots = mots + caractere1 # on ajoute le caractere du "mot1" a notre variable "mots"
99     for caractere2 in mots: #pour chaque caractere dans la variable "mots"
100         caracteres_present.append(caractere2) #on ajoute le caractere de la variable "mots" dans l'array "a"
101     nombre2 = 0 # on definie la variable "nombre2" qui est egale a 0
102     for caractere3 in caracteres_present: # pour chaque caractere present dans l'array "caracteres_present"
103         if caractere3 in mot_a_trouver[1:-1]: # si le caractere est present dans la variable "mot_a_trouver"
104             nombre2 = nombre2+1 # on ajoute 1 a la variable "nombre2"
105     return nombre2 # finalement on retourne la variable "nombre2" qui correspond au nombre de lettres mal placees par l'utilisateur
106
107
```

Preview from Mastermind code