

SUPINFO Academic Dept.

**Algorithmique et programmation en Python**

Mini - Projet

Mastermind

Version 1.0

Last update: 18/12/2014

Use: Students/Staff

Author: Laurent GODEFROY

SOMMAIRE

1 Préambule 3

2 Généralités sur ce projet 3

3 Codage du jeu en mode “console“ 4

3.1 Initialisations 4

3.2 L’évaluation d’une proposition 5

3.3 La procédure de jeu 5

4 on rajoute une ia simple 6

4.1 Les structures de données à utiliser 7

4.2 La fonction de Choix 7

4.3 La procédure de jeu 7

5 codage du jeu en mode “interface graphique“ 8

5.1 Création du plateau de Jeu 9

5.2 Des procédures de mise à jour du plateau 10

5.3 La procédure de jeu 10

6 des bonuS 12

6.1 Sauvegarder une partie 13

6.2 Une animation pour la fin de jeu 13

6.3 Soyez créatifs 13

7 barème indicatif 13

# Préambule

Cet examen est à réaliser par groupes de deux étudiants. Dans l’unique cas où le nombre d’étudiants de la promotion est impair, un et un seul groupe de trois est autorisé.

Toute forme de plagiat ou utilisation de codes disponibles sur internet ou tout autre support, même de manière partielle, est strictement interdite et se verra sanctionnée d’un 0, d’une mention « cheater », et le cas échéant d’un conseil de discipline.

Vous devrez envoyer les codes sources de votre projet par mail à **votre formateur** avant le dimanche 18 janvier 2015 à 23h59, heure locale. Au delà de cette date et heure votre note sera de 0. Vous comprimerez ces codes sources dans une archive au format « .zip » que vous nommerez en respectant la convention **ID-1ADS-votreCampus-MP** (exemple : 66280-1ADS-Tours-MP). L’objet du mail sera ce même nom. Vous mettrez également **en copie** la boîte [1ADS@supinfo.com](mailto:1ADS@supinfo.com). Si vous ne mettez pas cette dernière adresse en copie aucune réclamation ultérieure ne sera recevable.

Ce mini-projet donnera lieu à des soutenances qui se dérouleront la semaine du 19 janvier 2015. Vos horaires de passages vous seront communiqués par votre campus.

Les soutenances sont également par groupes de deux. Elles dureront **20 minutes** pendant lesquelles vous montrerez à votre examinateur le bon fonctionnement de votre programme en en faisant la démonstration. Si vous n’avez pas implémenté le projet dans sa totalité, vous exposerez les parties fonctionnelles.

Pour appuyer votre présentation, vous devrez préparer un fichier de type Powerpoint, dans lesquels vous expliquerez les points du code que vous jugez les plus importants et significatifs. Il n’est pas nécessaire d’envoyer ce fichier à votre examinateur, ce dernier le découvrira le jour de la soutenance. Une communication précisant tout cela vous sera envoyé début janvier.

Un barème **indicatif** vous est donné dans la dernière partie de ce sujet.

Dernière remarque, **n'hésitez pas à utiliser le forum pour échanger sur ce projet, j'ai ouvert un fil de discussion à cet usage.**

# Généralités sur ce projet

**Remarque importante**: aucun code n’est demandé dans cette partie qui n’est qu’explicative.

Le but de ce mini-projet est d’écrire en langage Python un programme permettant de jouer au célèbre jeu de déduction « Mastermind ».

Si vous n’êtes pas familier avec ses règles, je vous invite à les consulter, par exemple sur Wikipédia : <http://fr.wikipedia.org/wiki/Mastermind>

Vous aurez peut-être besoin de plusieurs lectures du sujet pour avoir une bonne vue d’ensemble du projet, prenez donc le temps nécessaire à une bonne compréhension avant de commencer les codes demandés dans les parties suivantes.

Dans la partie 3 on implémentera un mode de jeu en console où un joueur essaiera de deviner une suite de couleurs choisies aléatoirement par l’ordinateur. Ici le nombre de couleurs existantes, la longueur de la suite et le nombre d’essai pourront être quelconques.

Dans la partie 4 on implémentera une petite intelligence artificielle qui permettra à l’ordinateur de gagner quelque soit la suite de couleurs à trouver. On se restreindra à des suites de 4 couleurs choisies parmi six, et à dix essais.

Dans la partie 5 on implémentera un mode de jeu avec une interface graphique, où un joueur essaiera de deviner une suite de couleurs choisies aléatoirement par l’ordinateur. On se restreindra là aussi à des suites de 4 couleurs choisies parmi six, et à dix essais.

Et enfin, dans la partie 6, on proposera quelques bonus et extensions possibles.

**Remarque importante**: on supposera **toujours** (i.e. dans toutes les parties de ce projet) que la suite à deviner pourra comporter plusieurs fois la même couleur.

# Codage du jeu en mode “console“

**Il vous est fortement recommandé de lire l’intégralité de cette partie avant de commencer à coder. Les travaux demandés sont mis en évidence avec une couleur bleue.**

**Remarque importante**: dans cette partie où l’on ne travaillera qu’en mode « console », les couleurs seront représentées par des nombres entiers supérieurs ou égaux à 1. Par exemple si l’on dispose de 6 couleurs, elles seront représentées par les nombres 1, 2, 3, 4, 5 et 6 . La suite à deviner sera alors une liste constituée d’entiers supérieurs ou égaux à 1. Voici un exemple de suite constituée de 4 couleurs : [1,2,3,2].

## Initialisations

**Dans un fichier que l’on nommera « initialisations.py », implémenter les sous-programmes suivants :**

* Une fonction retournant les trois paramètres nécessaires au déroulement d’une partie : le nombre de couleurs existantes, la longueur de la suite à deviner et le nombre d’essai. Ces paramètres seront saisis par l’utilisateur et l’on fera en sorte que les valeurs qu’il rentre soient valides.
* Une fonction d’initialisation prenant en paramètre le nombre de couleurs existantes et la longueur d’une suite, et qui retourne une suite de couleurs (sous forme de liste comme précisé précédemment) choisies aléatoirement. Cette liste devra être de la bonne longueur (i.e. égale à la longueur passée en paramètre) et les couleurs devront être correctes (i.e. entre 1 et le nombre de couleurs existantes).

## L’évaluation d’une proposition

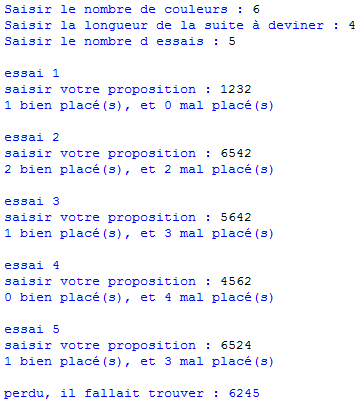
**Dans un fichier que l’on nommera « tourDeJeu.py », implémenter les sous-programmes suivants :**

* Une fonction de choix prenant en paramètre le nombre de couleurs existantes et la longueur d’une suite. Cette fonction fait saisir au joueur une proposition de couleurs sous la forme d’un nombre entier et la retournera sous forme de liste. Par exemple si le joueur saisit 1232 on retournera [1,2,3,2]. On fera en sorte de s’assurer que la liste ait le bon nombre d’élément (i.e. égal à la longueur passée en paramètre) et que les couleurs soient correctes (i.e. entre 1 et le nombre de couleurs existantes). Tant que ces deux conditions ne sont pas vérifiées on recommencera la saisie.
* Une fonction d’évaluation prenant en paramètre deux listes d’entiers supposées de même longueur. La première liste représentera une liste choisie par le joueur et la seconde la liste à deviner. Elle retournera le nombre d’éléments bien placés et le nombre d’éléments mal placés de la liste choisie par rapport à la liste à deviner.
* Une procédure d’affichage prenant en paramètres un nombre de couleurs bien placées et un nombre de couleurs mal placées et qui réalise l’affichage du résultat d’un essai.
* Une procédure d’affichage prenant en paramètre une liste d’entiers et qui réalise l’affichage de cette liste sous forme de nombres.

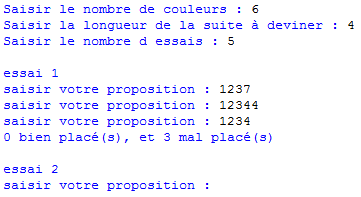
## La procédure de jeu

**Dans un fichier que l’on nommera « MastermindConsole.py », implémenter une procédure permettant de jouer au Mastermind.**

Voici un exemple de partie (perdue) :



Voici un exemple de début de partie où l’on voit bien les vérifications sur les saisies du joueur :



# on rajoute une ia simple

**Il vous est fortement recommandé de lire l’intégralité de cette partie avant de commencer à coder. Les travaux demandés sont mis en évidence avec une couleur bleue.**

**Rappel** : on se restreindra dans cette partie à des suites de 4 couleurs choisies parmi six, et à dix essais.

## Les structures de données à utiliser

Deux ensembles constitués de toutes les t-uples de couleurs possibles : (1,1,1,1), (1,1,1,2), (1,1,1,3), etc. L’un sera amené à être mis à jour et l’autre non. On les appellera par exemple “S“ et “possibles“.

Un ensemble constitué de tous les t-uples possibles de résultats de la forme (bien placés, mal placés) : (0,0), (0,1), (0,2), etc. On le nommera par exemple “résultats“.

## La fonction de Choix

**Dans un fichier que l’on nommera « MastermindIA.py », implémenter la fonction « choisirIA » suivante :**

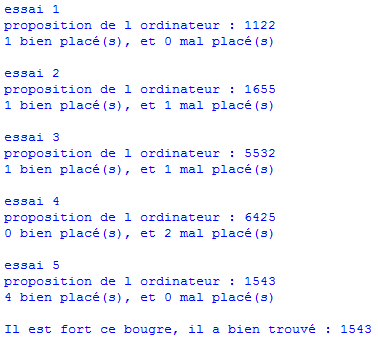
* Cette fonction prend en paramètres “S“, “possibles“, “résultats“ et le numéro de l’essai en cours.
* S’il s’agit du premier essai, cette fonction retournera [1,1,2,2].
* Sinon, elle calcule le maximum sur l’ensemble des éléments 'x’ de “possibles“, du minimum sur l’ensemble des éléments ‘res’ de “résultats“ du nombre d’éléments de S dont l’évaluation (au sens du jeu Mastermind) avec ‘s‘ est différente de ‘res’.

## La procédure de jeu

**Compléter le fichier précédent avec une procédure permettant à l’ordinateur de jouer au Mastermind selon la tactique suivante :**

1. Initialiser les structures de données présentées dans la sous-partie 4.1.
2. Choisir un coup ‘c’ grâce à la fonction de choix de la partie précédente.
3. Evaluer ce coup. On obtient alors un résultat de la forme (bien, mal).
4. Si ce choix est le bon (i.e. si bien = 4), le jeu s’arrête évidemment.
5. Sinon , retirer de ‘S’ les éléments dont l’évaluation avec ‘c’ ne donnent pas le résultat (bien, mal) obtenu à l’étape 2, repartir à l’étape 1.

Voici un exemple de partie prouvant la force de cet algorithme :



# codage du jeu en mode “interface graphique“

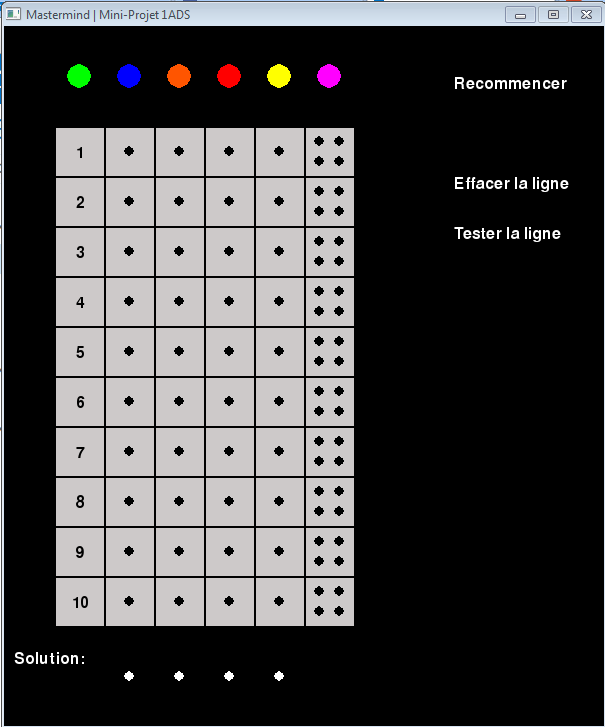
**Il vous est fortement recommandé de lire l’intégralité de cette partie avant de commencer à coder. Les travaux demandés sont mis en évidence avec une couleur bleue. On devra nécessairement utiliser la librairie graphique Pygame. L’usage d’une autre librairie ne sera pas pris en compte.**

**Remarque importante**: dans cette partie on continuera à mémoriser les couleurs par des nombres entiers supérieurs ou égaux à 1. La suite à deviner sera donc toujours une liste constituée d’entiers supérieurs ou égaux à 1. L’intérêt étant évidemment de se servir à nouveau de certains sous-programmes écrits lors de la partie 3. La suite à deviner sera toujours choisie aléatoirement.

**Rappel** : on se restreindra dans cette partie à des suites de 4 couleurs choisies parmi six, et à dix essais.

## Création du plateau de Jeu

**Dans un fichier que l’on nommera « mastermindGUI.py », implémenter une procédure permettant de dessiner un plateau de jeu qui devra ressembler à cela :**



## Des procédures de mise à jour du plateau

**Implémenter dans le fichier précédent deux procédures permettant de mettre à jour le plateau :**

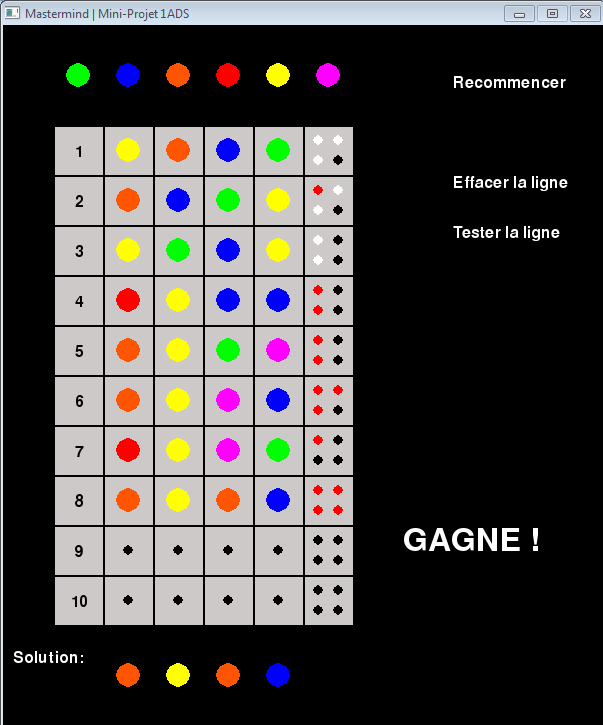
* Une procédure prenant en paramètre un entier représentant le numéro de l’essai en cours et qui efface les couleurs sur la ligne du plateau correspondant à ce numéro.
* Une procédure d’affichage prenant en paramètres un nombre de couleurs bien placées et un nombre de couleurs mal placées et qui réalise l’affichage du résultat d’un essai avec autant de petits pions rouges que de couleurs bien placées et autant de petits pions blancs que de couleurs mal placées. Cet affichage se fera dans la dernière colonne du plateau.

## La procédure de jeu

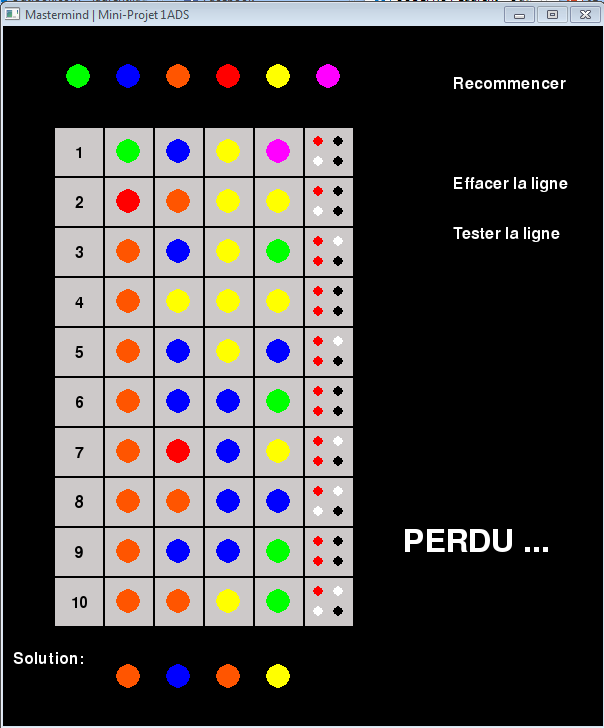
**Implémenter dans le fichier précédent une procédure permettant de jouer au Mastermind avec une interface graphique :**

* Pour sélectionner les couleurs de la ligne courante le joueur devra cliquer sur les disques au dessus du plateau.
* Quand il a rempli une ligne il cliquera sur “Tester la ligne“ pour voir le résultat apparaître dans la dernière colonne sous forme de petits pions rouges et blancs.
* Avant de valider une ligne, le joueur pourra l’effacer en cliquant sur “Effacer la ligne“, même si elle n’est pas complètement remplie.
* A tout moment un mauvais joueur pourra cliquer sur “Recommencer“.
* A la fin de la partie, qu’elle soit gagnée ou perdue, le joueur verra apparaître la solution en dessous du plateau.

Voici un exemple de partie gagnée :



Et voici un exemple de partie perdue :



# des bonuS

## Sauvegarder une partie

Rajouter une option (en mode console et/ou en mode graphique) permettant de sauvegarder l’état d’une partie dans un fichier texte.

Implémenter également une possibilité de reprendre le cours d’une partie mémorisée dans un fichier texte.

## Une animation pour la fin de jeu

Dans le jeu en mode graphique, rajouter une animation visuelle et sonore (se documenter sur ce dernier point) afin d’indiquer la victoire ou la défaite.

## Soyez créatifs

Implémenter, après l’avoir présenté, toute(s) nouvelle(s) fonctionnalité(s) que vous trouverez pertinente(s).

# barème indicatif

Ce barème peut-être amener à évoluer, il n’est donc qu’i**ndicatif**.

* Soutenance : 20 points (détails à venir)
* Partie 3: 15
* Partie 4 : 5
* Partie 5 : 20
* Bonus : 6

Ce qui fait un total de 60 points, ramené sur 20 par proportionnalité.