Le jeu du Mastermind

Auteur : Thomas Langlinay

Niveaux scolaires :
Seconde, Première et Terminale ES ou S

Mots clés :
Algorithme et programmation

ENONCE

Ces dernières décennies ont été riches en films de science-fiction nous proposant des histoires incroyables sur l'intelligence artificielle (IA). Depuis les années 50 et les travaux d'Alan Turing avec son fameux "test de Turing", l'IA est bien réelle. Elle est omniprésente dans notre vie de tous les jours, dans des domaines aussi variés que la détection de catastrophes naturelles ou de maladies chez le nourrisson, l'anticipation des risques financiers…
Elle permet aussi de simuler le comportement humain lors de jeux de réflexion. Depuis 1979 et la défaite du champion du monde de Backgammon face au logiciel BKG 9.8, les ordinateurs ont petit à petit battu les champions à différents jeux : échecs, poker et, très récemment, jeu de go.
Le jeu est ainsi l'un des domaines d'application de l'intelligence artificielle. Cette ressource vous propose donc un programme en langage Python permettant à des élèves, avec la calculatrice Graph 90+E, de simuler le jeu du *Mastermind*.
Programmer une IA en Python sur une calculatrice CASIO
Extrait du magazine Tangente Education n°46 septembre 2018

1. Jouer en classe en programmant

Discipline à part entière, l'IA s'alimente de nombreux champs scientifiques comme les mathématiques ou l'informatique.
Dès le collège, les élèves sont confrontés à l'informatique pour appréhender différents problèmes. Python, un des langages de programmation les plus utilisés au monde - en particulier pour programmer une IA - est le langage officiel défini par l'Éducation nationale pour l'enseignement de la programmation au lycée.

En voici donc une application à un jeu que tout le monde (ou presque connaît), le Mastermind. Rappelons-en les règles. Ce jeu de logique fait s'affronter deux joueurs : le codeur et le décodeur.

- Le codeur choisit une combinaison de 4 pions de couleurs parmi 6 couleurs possibles. Plusieurs pions de la même couleur peuvent être utilisés.
- Le décodeur, quant à lui, doit deviner par déduction, les couleurs et les positions des pions de son adversaire.

Après chaque proposition du décodeur, le codeur indique le nombre de pions de la bonne couleur bien placés (en utilisant des pions noirs) et mal placés (pions blancs). Dix essais sont proposés au décodeur pour déterminer la séquence de couleurs.

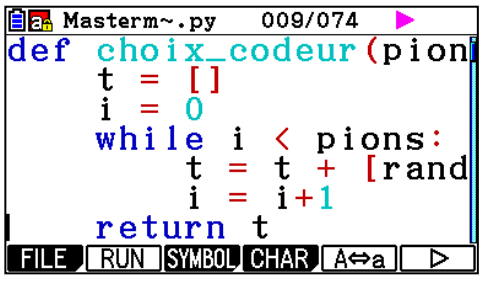
2. Quelques étapes du programme de Mastermind

Les sous-programmes qui suivent constituent certaines des étapes de la programmation du jeu de Mastermind. Chacun d'entre eux peut être réalisé avec un éditeur Python sur ordinateur ou directement sur la calculatrice Graph 90+E. Certaines parties sont présentées ci-dessous. N'hésitez pas à télécharger le fichier Mastermind.py. Vous pourrez ainsi obtenir directement le programme dans son intégralité.

1) Le choix du codeur

La fonction suivante, choix_codeur, est générée en fonction du couple (pions, couleurs). Cette fonction produit une combinaison aléatoire de *k* couleurs pour *n* pions (pions sera définie par la suite égale à 4). Avant de définir cette fonction, il ne faudra pas oublier d'importer le module *random* avec la ligne *from random import**.

```
def choix_codeur(pions,couleurs):
    t = []
    i = 0
    while i < pions:
        t = t + [randrange(1,couleurs+1)]
        i = i+1
    return t
```



2) Le bilan d'une proposition du décodeur

Les deux fonctions (noirs et blancs) créées ici sont fonction de deux variables : la proposition du codeur et de la solution (le choix-codeur précédent). Elles indiquent le nombre de couleurs bien placées et mal placées. On remarque dans le programme qu'afin d'éviter de compter plusieurs fois la même couleur, les chiffres des variables sont remplacés par des lettres (*x* et *y*) après usage.

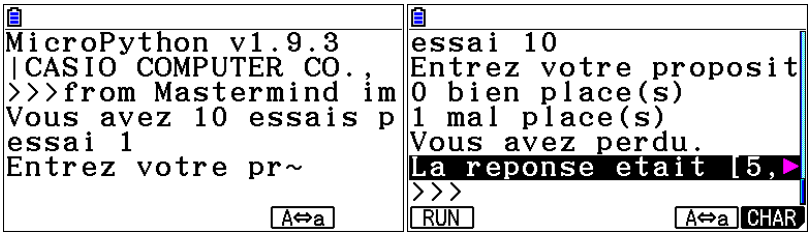
```
def blancs(proposition,solution):
    i = 0
    mal_place = 0
    while i < pions:
        j = 0
        while j < pions:
            if proposition[i] == solution[j]:
                mal_place = mal_place+1
                proposition[i] = 'y'
                solution[j] = 'x'
            j = j+1
        i = i+1
    return mal_place

def noirs(proposition,solution):
    bien_place = 0
    i = 0
    while i < pions:
        if proposition[i] == solution[i]:
            bien_place = bien_place+1
            proposition[i] = 'y'
            solution[i] = 'x'
        i = i+1
    return bien_place
```

3) Le jeu proprement dit

```
print("Vous avez",nb_essai_maximum,"essais pour trouver la bonne combinaison")

while nb_biens_places < pions and essai < nb_essai_maximum:
    essai = essai+1
    print("essai",essai)

    prop = list(input("Entrez votre proposition:"))
    copie_proposition_liste = copie(prop)
    copie_solution_liste = copie(sol)
    nb_biens_places = noirs(copie_proposition_liste, copie_solution_liste)
    print(nb_biens_places,"bien place(s)")

    nb_mals_places = blancs(copie_proposition_liste, copie_solution_liste)
    print(nb_mals_places, "mal place(s)")

if nb_biens_places == pions:
    print("Bravo, vous avez gagne")
else:
    print('"Vous avez perdu.
La reponse etait"',sol)
```

Le programme ci-dessus permet de jouer, une fois les sous-programmes réalisés. Il précise le nombre d'essais, puis, pour chaque essai, applique le sous-programme « noirs et blancs » pour déterminer le nombre de pions bien placés. Si ce nombre est le nombre de pions, il affiche « Bravo, vous avez gagné ». Dans le cas contraire, il permet un autre essai, jusqu'au dernier où il affiche « Vous avez perdu » suivi de la réponse.
On remarque que cette dernière partie du programme contient, outre une boucle "while", une condition "if-else".

Une fois le programme écrit, il peut être exécuté dans la console Shell.



On remarquera que l'intelligence artificielle, qui permet au programme de se mettre à la place du décodeur, n'est pas développée ici. Il faudrait beaucoup plus de place que cet article.