Sacha Coulombel – Michel Wang

SAE R1_01 :

Voici 4 Codes avec leurs jeux d'essais :

Programme Menu :

Programme python qui va faire office de Menu. Début avec le programme qui va demander à l'utilisateur de saisir un entier, 1 pour jouer au Devinette, 2 pour Allumette, 3 pour Morpion et 4 pour Puissance4.

- *Ensuite, la vérification par le programme pour voir si la saisie correspond à 1 ou 2 ou 3 ou 4.*
- *Ensuite, l'utilisation des fonctions en fonction de la sais*ie. Avec le fonctionnement des *scores*
- *Attribution des scores* à une variable *« score_final ». Pour ensuite attribuer les différents scores des différents jeux, à une liste (seulement celui du meilleur joueur)*
- *Tous ceci dans une boucle, dans laquelle le programme va demander à l'utilisateur s'il veut rejouer ou non à un jeu. Ensuite la vérification de la saisie : soit 1 pour continuer, soit 2 pour arrêter*
- *Ensuite on sort de la boucle si l'utilisateur saisi 2*
- *On tri la liste avec une fonction tri à bulle*
- *Et puis on affiche le classement des différents jeux du meilleure joueur*

```python
from fonction_mor import morpions
from fonction_allu import allumettes
from fonction_dev import devinettes
from fonction_puiss import puissance
from fonction_tri import tri_bulle

if __name__ == "__main__" :
    """
    #-------------------------------------------------------------------------------
    Fonction : fonction morpions, fonction allumettes, fonction devinettes et fonction
    puissance 4
    Programme qui va faire office de menu afin d'accéder aux différents jeux comme le
    morpion, la devinette, les allumettes ou bien le puissance quatre.
    #-------------------------------------------------------------------------------
    """

    score1 : int            #score après la fin d'un seul jeu du joueur 1
    score2 : int            #score après la fin d'un seul jeu du joueur 1
    score1_dev : int        #score du joueur 1 pour la Devinette
    score2_dev : int        #score du joueur 2 pour la Devinette
    score1_all : int        #score du joueur 1 pour l'Allumette
    score2_all : int        #score du joueur 2 pour l'Allumette
    score1_mor : int        #score du joueur 1 pour le Morpion
    score2_mor : int        #score du joueur 2 pour le Morpion
    score1_pui : int        #score du joueur 1 pour le Puissance 4
    score2_pui : int        #score du joueur 2 pour le Puissance 4
    score1_final : int      #score final du joueur 1
    score2_final : int      #score final du joueur 2
    saisi : int
    jouer : int
    liste : list[int]       #liste commençant par le score du jeu Devinette, Allumette, Morpion puis Puissance 4
    liste_nom : list[str]   #liste

    jouer = 1
    liste = [0, 0, 0, 0]
    liste_nom = ["Devinette", "Allumette", "Morpion", "Puissance 4"]
    score1_dev = 0
    score2_dev = 0
    score1_all = 0
    score2_all = 0
    score1_mor = 0
    score2_mor = 0
    score1_pui = 0
    score2_pui = 0
    score1_final = 0
    score2_final = 0
```

```python
        #choix des jeux avec
        if saisi == 1 :
            score1, score2 = devinettes(score1_dev, score2_dev)
            score1_dev = score1_dev + score1
            score2_dev = score2_dev + score2
        else :
```

```python
        #score final
        score1_final = score1_all + score1_dev + score1_mor + score1_pui
        score2_final = score2_all + score2_dev + score2_mor + score2_pui

        if score1_final > score2_final :
            #liste des scores du joueur 1 des différents jeux
            liste[0] = score1_dev
            liste[1] = score1_all
            liste[2] = score1_mor
            liste[3] = score1_pui
        else :
            #liste des scores du joueur 2 des différents jeux
            liste[0] = score2_dev
            liste[1] = score2_all
            liste[2] = score2_mor
            liste[3] = score2_pui
```

```python
#condition l'utilisateur doit taper 1 ou 2, 1 s'il veut continue
r à jouer et 2 s'il veut arrêter
        while (jouer != 1) and (jouer != 2) :
            print("\n" * 50, "Votre saisi est fausse")
            print("Taper 1 pour si vous voulez jouer")
            print("Taper 2 pour si vous ne voulez plus jouer")
            jouer = int(input("Votre saisi :"))
```

```python
    #tri des scores des différents jeux pour le joueur gagnant final
    liste, liste_nom = tri_bulle(liste, liste_nom)

    if (score1_final > score2_final) :
        print("\nLe classement du joueur 1")
        print("\n" * 3, "classement", 4,":", liste_nom[0], ":", liste[0])
        print("classement", 3,":", liste_nom[1], ":", liste[1])
        print("classement", 2,":", liste_nom[2], ":", liste[2])
        print("classement", 1,":", liste_nom[3], ":", liste[3])
    else :
        if (score2_final > score1_final) :
            print("\nLe classement du joueur 2")
            print("\n" * 3, "classement", 4,":", liste_nom[0], ":", liste[0])
            print("classement", 3,":", liste_nom[1], ":", liste[1])
            print("classement", 2,":", liste_nom[2], ":", liste[2])
            print("classement", 1,":", liste_nom[3], ":", liste[3])
        else :
            print("Match nul")
```

Jeux d'essai :

```
Menu

- Taper 1 pour jouer au Devinette
- Taper 2 pour jouer à l'Allumettes
- Taper 3 pour jouer au Morpion
- Taper 4 pour jouer au Puissance Quatre
```

```
Votre saisi :1
Le joueur 1 choisi un entier et une intervalle
Le joueur 2 a 5 essais et s'il réussit à deviner l'entier, il gagne
Sinon le joueur 1 gagne
Joueur1, choisissez la limite maximale :
```

```
Votre saisi :3
Jeu de Morpion simple
X pour le joueur 1
O pour le joueur 2

Veuilez saisir le nombre de partie que vous voulez jouer :
```

```
Votre saisi :2
Il y a 20 allumettes.
Chacun son tour attrape 1, 2 ou 3 allumettes.
Le premier qui attrape la dernière allumette a gagné !
Veuilez saisir le nombre de partie qui vous voulez jouer :
```

```
Votre saisi :4
Jeu de Puissance 4 simple
X pour le joueur 1
O pour le joueur 2

Veuilez saisir le nombre de partie que vous voulez jouer :
```

```
Votre saisi :0
```

```
 Votre saisi est fausse
- Taper 1 pour jouer au Devinette
- Taper 2 pour jouer à l'Allumettes
- Taper 3 pour jouer au Morpion
- Taper 4 pour jouer au Puissance Quatre
Votre saisi :
```

```
Votre saisi :5
```

```
 Votre saisi est fausse
- Taper 1 pour jouer au Devinette
- Taper 2 pour jouer à l'Allumettes
- Taper 3 pour jouer au Morpion
- Taper 4 pour jouer au Puissance Quatre
Votre saisi :
```

Programme Devinettes :

Fonction qui va prendre en paramètre d'entrée le score du joueur 1 et du joueur 2. Et va renvoyer les mêmes scores mais avec des différentes valeurs en fonction de qui a gagné :

- *Le joueur 1 qui saisi l'entier à deviner. Et va ensuite saisir un intervalle entre 1 et un nombre maximal qu'il va également saisir.*
- *Le joueur 2 qui aura 5 tentatives de deviner ce nombre. Pour cela, une boucle tentative qui baisse de 1 à chaque itération. A 0, le joueur 2 perd et le joueur 1 gagne*
- *Contrôle de la saisie du joueur 2, que sa saisie est bien dans l'intervalle*
- *Après chaque réponse du joueur 2, le joueur 1 donne son avis.*
- *Attribution des scores en fonction du gagnant et la renvoie au programme menu*

```python
def devinettes(score1 : int, score2 : int) -> int :
    """
    #-------------------------------------------------------------------------------
    Fonction qui va prendre en paramètre d'entrée, les scores des deux joueurs. Et
    va renvoyer en paramètre de sortie, les scores des deux joueurs après avoir terminés
    les parties. JEU DE DEVINETTE
    #-------------------------------------------------------------------------------
    """

    nb_J1 : int
    nb_J2 : int
    Max : int
    tentative : int
    rejouer : str
    reponse : int

    nb_J1 = 0
    nb_J2 = 0
    Max = 0
    rejouer = ""
    tentative = 5
    reponse = 0

    print("\n" *100)
    while True :
```

```python
        while (nb_J1 < 1 or nb_J1 > Max and Max < 1):
            Max = int(input("Joueur1, choisissez la limite maximale : "))
            nb_J1 = int(input("Choisissez votre nombre entier entre 1 et " + str(Max) + " : "))
```

```python
        while (tentative > 0) :

            nb_J2 = 0
            while(nb_J2 > Max or nb_J2 < 1) :
                print("\n" * 3)
                print("Joueur2, vous devez choisir un nombre entre 1 et "+ str(Max))
                nb_J2 = int(input("Joueur2 choisissez un nombre : "))

                if (nb_J2 > Max or nb_J2 < 1):
                    print("Vous devez choisir un nombre dans la limite indiqué")
                else :
                    break


            tentative = tentative - 1
```

```python
            #condition de la réponse, soit 1 ou 2 ou 3
            while reponse != 1 or reponse != 2 or reponse != 3 :
                print("Joueur1, le nombre choisi par le Joueur2 est :", nb_J2)
                reponse = int(input(print(nb_J2," est :"
                    "\n1. Plus petit que votre nombre"
                    "\n2. Plus grand que votre nombre"
                    "\n3. Egale à votre nombre")))

                print("\n" * 20)
                if reponse == 1 :
                    if nb_J2 < nb_J1 :
                        print("Le nombre du Joueur2 est bien trop petit")
                        break
                    else :
                        print("Menteur! Le nombre du Joueur2 n'est pas plus petit")
```

```python
            if nb_J2 == nb_J1:
                break
            print("il vous reste ",tentative, " tentatives")



        if nb_J1 == nb_J2:
            print("Le joueur2 a remporté la partie")
            score2 = score2 + 1
        else :
            print("Le joueur 1 a remporté la partie ")
            score1 = score1 + 1
        rejouer = str(input("Voulez vous rejouer ? (oui/non) : "))

        if rejouer != "oui" :
            break
```

Jeux d'essais :

```
Le joueur 1 choisi un entier et une intervalle
Le joueur 2 a 5 essais et s'il réussit à deviner l'entier, il gagne
Sinon le joueur 1 gagne




Joueur1, choisissez la limite maximale : 50
Choisissez votre nombre entier entre 1 et 50 : 25
```

```
Joueur1, choisissez la limite maximale : 50
Choisissez votre nombre entier entre 1 et 50 : 55
Joueur1, choisissez la limite maximale : 50
Choisissez votre nombre entier entre 1 et 50 : 15
```

```
Joueur2, vous devez choisir un nombre entre 1 et 50
Joueur2 choisissez un nombre : 36
```

```
Joueur2, vous devez choisir un nombre entre 1 et 50
Joueur2 choisissez un nombre : 55
Vous devez choisir un nombre dans la limite indiqué
```

```
Joueur2, vous devez choisir un nombre entre 1 et 50
Joueur2 choisissez un nombre : -1
Vous devez choisir un nombre dans la limite indiqué
```

```
Menteur! Le nombre du Joueur2 n'est pas plus grand
Joueur1, le nombre choisi par le Joueur2 est : 20
20  est :
1. Plus petit que votre nombre
2. Plus grand que votre nombre
3. Egale à votre nombre
```

```
Le nombre du Joueur2 est bien trop petit
il vous reste  3  tentatives
```

```
Menteur! Le Joueur2 n'a pas donné la bonne réponse
Joueur1, le nombre choisi par le Joueur2 est : 20
20  est :
1. Plus petit que votre nombre
2. Plus grand que votre nombre
3. Egale à votre nombre
```

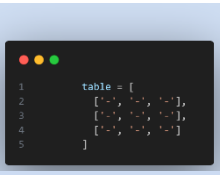
```
Menteur! Le Joueur2 n'a pas donné la bonne réponse
Joueur1, le nombre choisi par le Joueur2 est : 20
20  est :
1. Plus petit que votre nombre
2. Plus grand que votre nombre
3. Egale à votre nombre
```

```
Menteur! Le Joueur2 n'a pas donné la bonne réponse
Joueur1, le nombre choisi par le Joueur2 est : 20
20  est :
1. Plus petit que votre nombre
2. Plus grand que votre nombre
3. Egale à votre nombre
```

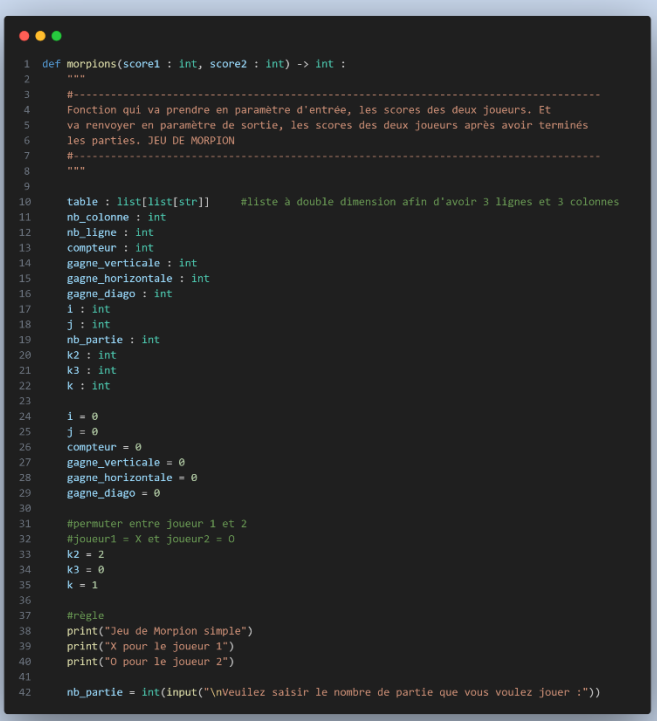
Programme Morpion :

Fonction qui va prendre en paramètre d'entrée le score du joueur 1 et du joueur 2. Et va renvoyer les mêmes scores mais avec des différentes valeurs en fonction de qui a gagné :

- *Initialisation d'une liste à double dimension. Soit 3 sous-listes de 3 cases. Afin d'avoir une matrice de 3 lignes et 3 colonnes pour le jeu de morpion*
- *Boucle « for » pour afficher une matrice qui ressemble au morpion*
- *Entrée dans une boucle pour saisir le nombre de partie souhaité*
- *Entrée dans une autre boucle qui vérifie si toute les cases sont remplies ou si l'un des joueurs a gagné ou non*
- *Dans cette boucle, demande à l'utilisateur de saisir la colonne et ligne pour jouer. Vérification de sa réponse (si la case est déjà prise ou non, et s'il a bien saisi un nombre entre 1 et 3)*
- *Si pas de problème, entrée de 0 ou X en fonction du joueur à la case choisie*
- *Sort de cette boucle si l'une des conditions de la boucle est remplie*
- *Attribution du score en fonction du gagnant*
- *Renvoie au programme Menu à la toute fin*

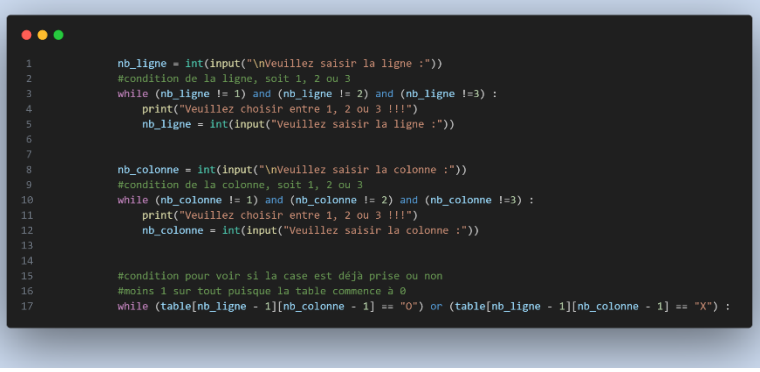


```
table = [
  ['-', '-', '-'],
  ['-', '-', '-'],
  ['-', '-', '-']
]
```

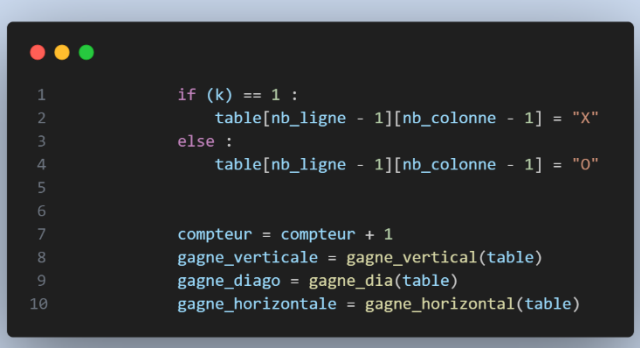


```
def morpions(score1 : int, score2 : int) -> int :
    """
    #-------------------------------------------------------------------------------
    Fonction qui va prendre en paramètre d'entrée, les scores des deux joueurs. Et
    va renvoyer en paramètre de sortie, les scores des deux joueurs après avoir terminés
    les parties. JEU DE MORPION
    #-------------------------------------------------------------------------------
    """

    table : list[list[str]]     #liste à double dimension afin d'avoir 3 lignes et 3 colonnes
    nb_colonne : int
    nb_ligne : int
    compteur : int
    gagne_verticale : int
    gagne_horizontale : int
    gagne_diago : int
    i : int
    j : int
    nb_partie : int
    k2 : int
    k3 : int
    k : int

    i = 0
    j = 0
    compteur = 0
    gagne_verticale = 0
    gagne_horizontale = 0
    gagne_diago = 0

    #permuter entre joueur 1 et 2
    #joueur1 = X et joueur2 = O
    k2 = 2
    k3 = 0
    k = 1

    #règle
    print("Jeu de Morpion simple")
    print("X pour le joueur 1")
    print("O pour le joueur 2")

    nb_partie = int(input("\nVeuilez saisir le nombre de partie que vous voulez jouer :"))
```

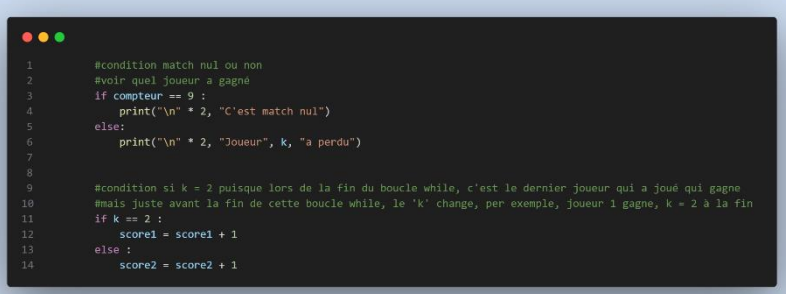
```
#affichage de la matrice, pour qu'il ressemble à un morpion
for i in range(0, 3) :
    print("\n")
    for j in range(0, 3) :
        print(table[i][j], end="  ")
```



```
nb_ligne = int(input("\nVeuillez saisir la ligne :"))
#condition de la ligne, soit 1, 2 ou 3
while (nb_ligne != 1) and (nb_ligne != 2) and (nb_ligne !=3) :
    print("Veuillez choisir entre 1, 2 ou 3 !!!")
    nb_ligne = int(input("Veuillez saisir la ligne :"))


nb_colonne = int(input("\nVeuillez saisir la colonne :"))
#condition de la colonne, soit 1, 2 ou 3
while (nb_colonne != 1) and (nb_colonne != 2) and (nb_colonne !=3) :
    print("Veuillez choisir entre 1, 2 ou 3 !!!")
    nb_colonne = int(input("Veuillez saisir la colonne :"))


#condition pour voir si la case est déjà prise ou non
#moins 1 sur tout puisque la table commence à 0
while (table[nb_ligne - 1][nb_colonne - 1] == "O") or (table[nb_ligne - 1][nb_colonne - 1] == "X") :
```



```
if (k) == 1 :
    table[nb_ligne - 1][nb_colonne - 1] = "X"
else :
    table[nb_ligne - 1][nb_colonne - 1] = "O"


compteur = compteur + 1
gagne_verticale = gagne_vertical(table)
gagne_diago = gagne_dia(table)
gagne_horizontale = gagne_horizontal(table)
```

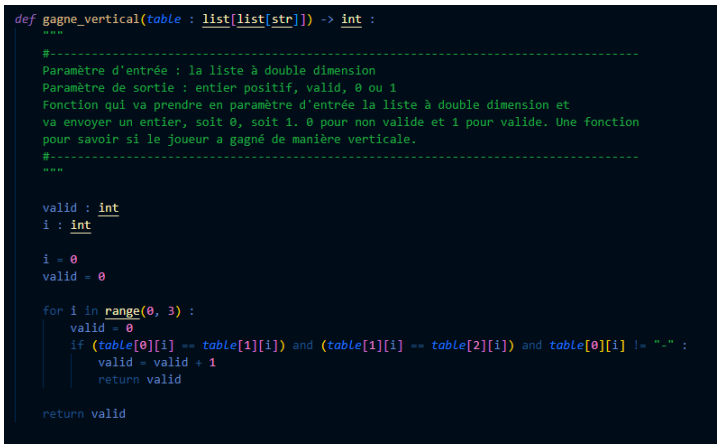


```
#condition match nul ou non
#voir quel joueur a gagné
if compteur == 9 :
    print("\n" * 2, "C'est match nul")
else:
    print("\n" * 2, "Joueur", k, "a perdu")


#condition si k = 2 puisque lors de la fin du boucle while, c'est le dernier joueur qui a joué qui gagne
#mais juste avant la fin de cette boucle while, le 'k' change, per exemple, joueur 1 gagne, k = 2 à la fin
if k == 2 :
    score1 = score1 + 1
else :
    score2 = score2 + 1
```

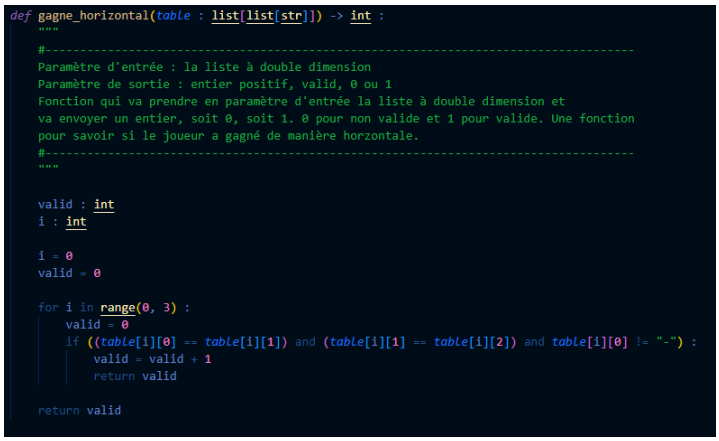
3 fonctions complémentaire pour la fonction principale de morpion. Une fonction pour gagner verticalement, une autre pour gagner horizontalement et une dernière pour gagner de manière diagonale : (dans tous ces cas, il faut aligner 3 cases).

Les 3 fonctions prennent en paramètre d'entrée la liste qui représente le morpion. Et vont renvoyer un entier. L'entier sera égale à 1 si la condition pour gagner est valide. Et puis va renvoyer 0 si la condition n'est pas vérifiée :

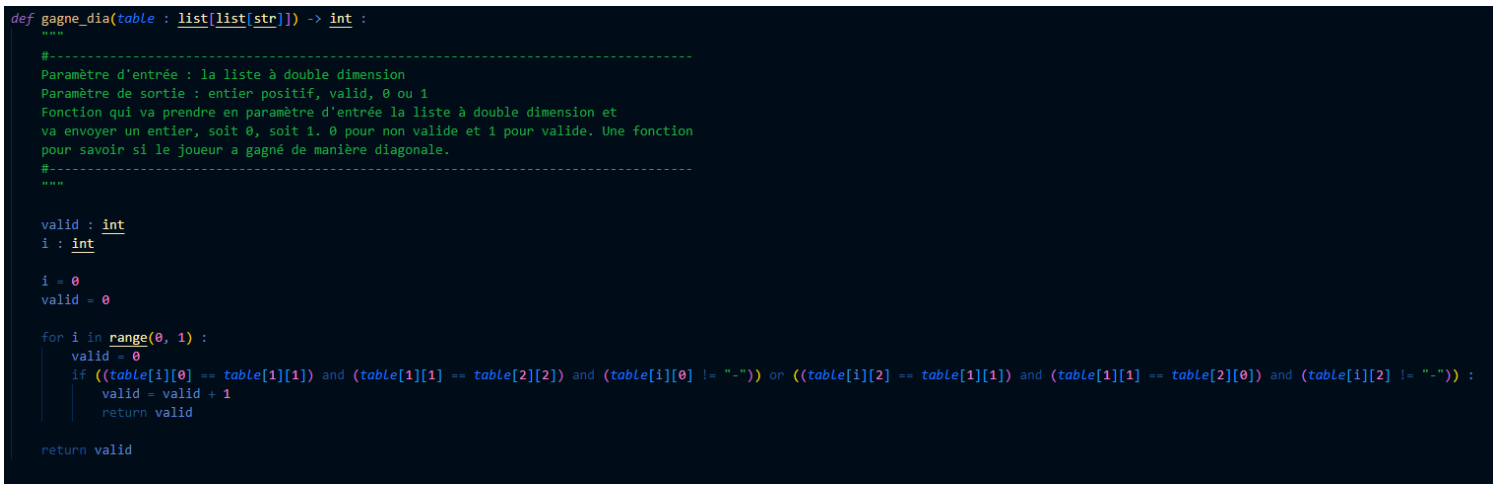
- *Pour la fonction gagne_vertical. On utilise une boucle « for » pour parcourir toute la liste et vérifie si les valeurs de la liste d'une même colonne, sont similaires*
- *Pour la fonction gagne_horizontal. On fait la même chose mais on vérifie cette fois s'ils ont de la même ligne*
- *Pour la fonction gagne_dia. On utilise une boucle « for » pour parcourir toute la liste et vérifie si les valeurs de la liste de même ligne et même colonne sont similaires*



```
def gagne_vertical(table : list[list[str]]) -> int :
    """
    #-------------------------------------------------------------------------------
    Paramètre d'entrée : la liste à double dimension
    Paramètre de sortie : entier positif, valid, 0 ou 1
    Fonction qui va prendre en paramètre d'entrée la liste à double dimension et
    va envoyer un entier, soit 0, soit 1. 0 pour non valide et 1 pour valide. Une fonction
    pour savoir si le joueur a gagné de manière verticale.
    #-------------------------------------------------------------------------------
    """

    valid : int
    i : int

    i = 0
    valid = 0

    for i in range(0, 3) :
        valid = 0
        if (table[0][i] == table[1][i]) and (table[1][i] == table[2][i]) and table[0][i] != "-" :
            valid = valid + 1
            return valid

    return valid
```



```
def gagne_horizontal(table : list[list[str]]) -> int :
    """
    #-------------------------------------------------------------------------------
    Paramètre d'entrée : la liste à double dimension
    Paramètre de sortie : entier positif, valid, 0 ou 1
    Fonction qui va prendre en paramètre d'entrée la liste à double dimension et
    va envoyer un entier, soit 0, soit 1. 0 pour non valide et 1 pour valide. Une fonction
    pour savoir si le joueur a gagné de manière horzontale.
    #-------------------------------------------------------------------------------
    """

    valid : int
    i : int

    i = 0
    valid = 0

    for i in range(0, 3) :
        valid = 0
        if ((table[i][0] == table[i][1]) and (table[i][1] == table[i][2]) and table[i][0] != "-") :
            valid = valid + 1
            return valid

    return valid
```



```
def gagne_dia(table : list[list[str]]) -> int :
    """
    #-------------------------------------------------------------------------------
    Paramètre d'entrée : la liste à double dimension
    Paramètre de sortie : entier positif, valid, 0 ou 1
    Fonction qui va prendre en paramètre d'entrée la liste à double dimension et
    va envoyer un entier, soit 0, soit 1. 0 pour non valide et 1 pour valide. Une fonction
    pour savoir si le joueur a gagné de manière diagonale.
    #-------------------------------------------------------------------------------
    """

    valid : int
    i : int

    i = 0
    valid = 0

    for i in range(0, 1) :
        valid = 0
        if ((table[i][0] == table[1][1]) and (table[1][1] == table[2][2]) and (table[i][0] != "-")) or ((table[i][2] == table[1][1]) and (table[1][1] == table[2][0]) and (table[i][2] != "-")) :
            valid = valid + 1
            return valid

    return valid
```

Jeu d'essai :

```
Veuilez saisir le nombre de partie que vous voulez jouer :1
```

```
 Joueur 2 Veuillez choisir le numéro de la ligne puis le numéro de la colonne !

 Veuillez saisir la ligne :1

 Veuillez saisir la colonne :1

 * 50


X  -  -

-  -  -

-  -  -
Votre case choisie est déjà prise !!!
Veuillez saisir la ligne :
```

```
-  -  -

-  -  -

-  -  -
 Joueur 1 Veuillez choisir le numéro de la ligne puis le numéro de la colonne !

 Veuillez saisir la ligne :1

 Veuillez saisir la colonne :1

 * 50


X  -  -

-  -  -

-  -  -
```

```
 Joueur 1 Veuillez choisir le numéro de la ligne puis le numéro de la colonne !

 Veuillez saisir la ligne :3

 Veuillez saisir la colonne :1

 * 50


X  O  O

X  -  -

X  -  -
 Joueur 2 a perdu
le score du joueur 1 : 1
le score du joueur 2 : 0
```

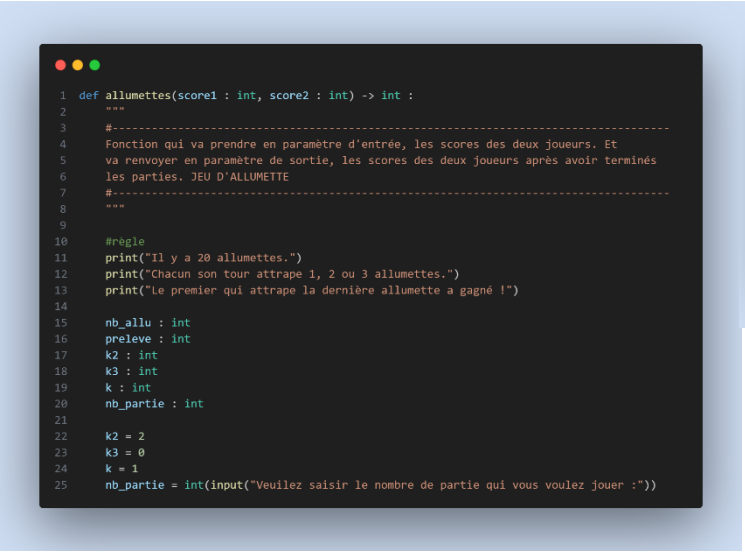
```
 Devinette score : 0
Allumette score : 0
Morpion score : 1
Puissance 4 score : 0
```

```
 Taper 1 pour si vous voulez continuer à jouer
Taper 2 pour si vous ne voulez plus jouer
Votre saisi :
```

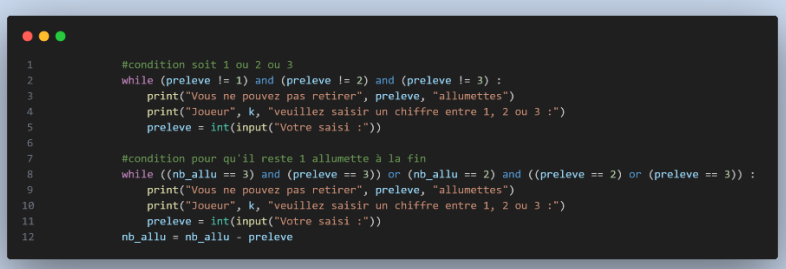
Programme Allumettes :

Fonction qui va prendre en paramètre d'entrée le score du joueur 1 et du joueur 2. Et va renvoyer les mêmes scores mais avec des différentes valeurs en fonction de qui a gagné :

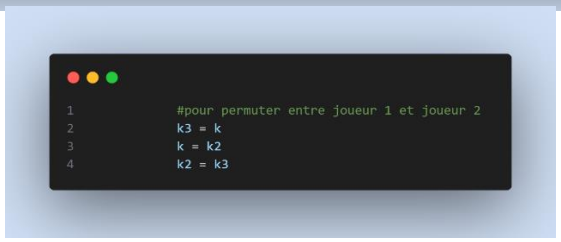
- *Commence avec une boucle qui va permettre à l'utilisateur de choisir le nombre de partie qu'il voudra jouer*
- *Ensuite demande à l'utilisateur de saisir un entier entre 1 et 3. Vérification de la saisie si c'est bien 1 ou 2 ou 3. De plus la vérification si le joueur peut bien enlever n nombre d'allumette en sachant qu'il faut qu'il en reste un seul*
- *Le nombre total d'allumette qui baisse en fonction de la saisie de l'utilisateur*
- *La permutation de la variable k pour permuter entre joueur 1 et joueur 2*
- *A la fin lorsque le nombre total d'allumette atteint 1, on attribue plus un au score en fonction du joueur gagnant*



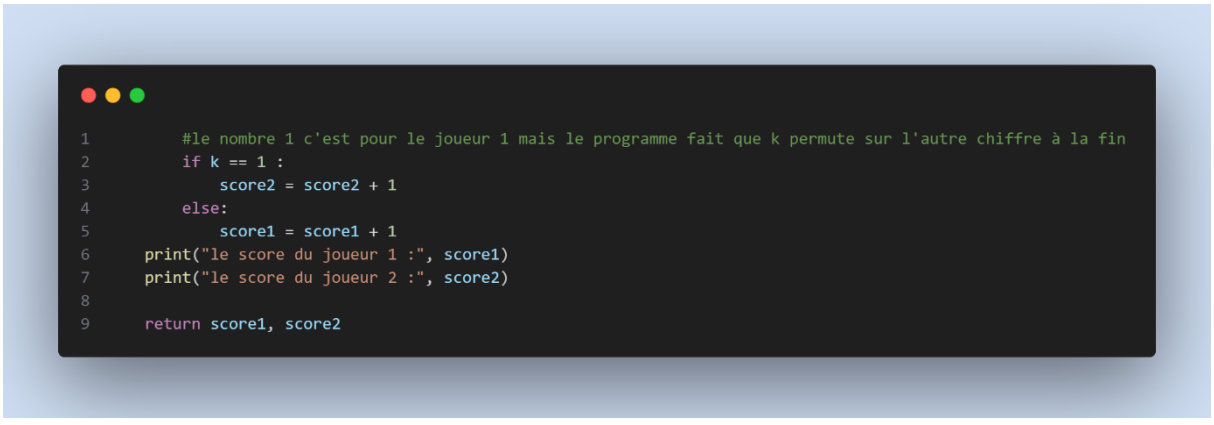
```
def allumettes(score1 : int, score2 : int) -> int :
    """
    #-------------------------------------------------------------------------------
    Fonction qui va prendre en paramètre d'entrée, les scores des deux joueurs. Et
    va renvoyer en paramètre de sortie, les scores des deux joueurs après avoir terminés
    les parties. JEU D'ALLUMETTE
    #-------------------------------------------------------------------------------
    """

    #règle
    print("Il y a 20 allumettes.")
    print("Chacun son tour attrape 1, 2 ou 3 allumettes.")
    print("Le premier qui attrape la dernière allumette a gagné !")

    nb_allu : int
    preleve : int
    k2 : int
    k3 : int
    k : int
    nb_partie : int

    k2 = 2
    k3 = 0
    k = 1
    nb_partie = int(input("Veuilez saisir le nombre de partie qui vous voulez jouer :"))
```



```
        #condition soit 1 ou 2 ou 3
        while (preleve != 1) and (preleve != 2) and (preleve != 3) :
            print("Vous ne pouvez pas retirer", preleve, "allumettes")
            print("Joueur", k, "veuillez saisir un chiffre entre 1, 2 ou 3 :")
            preleve = int(input("Votre saisi :"))

        #condition pour qu'il reste 1 allumette à la fin
        while ((nb_allu == 3) and (preleve == 3)) or (nb_allu == 2) and ((preleve == 2) or (preleve == 3)) :
            print("Vous ne pouvez pas retirer", preleve, "allumettes")
            print("Joueur", k, "veuillez saisir un chiffre entre 1, 2 ou 3 :")
            preleve = int(input("Votre saisi :"))
        nb_allu = nb_allu - preleve
```



```
        #pour permuter entre joueur 1 et joueur 2
        k3 = k
        k = k2
        k2 = k3
```



```
        #le nombre 1 c'est pour le joueur 1 mais le programme fait que k permute sur l'autre chiffre à la fin
        if k == 1 :
            score2 = score2 + 1
        else:
            score1 = score1 + 1
    print("le score du joueur 1 :", score1)
    print("le score du joueur 2 :", score2)

    return score1, score2
```

Jeux d'essai :

```
Votre saisi :2
Il y a 20 allumettes.
Chacun son tour attrape 1, 2 ou 3 allumettes.
Le premier qui attrape la dernière allumette a gagné !
Veuilez saisir le nombre de partie qui vous voulez jouer :1
il reste 20 allumettes
Joueur 1 veuillez saisir un chiffre entre 1, 2 ou 3 :
Votre saisi :1
il reste 19 allumettes
```

```
Joueur 2 veuillez saisir un chiffre entre 1, 2 ou 3 :
Votre saisi :3
il reste 16 allumettes
```

```
Joueur 1 veuillez saisir un chiffre entre 1, 2 ou 3 :
Votre saisi :4
Vous ne pouvez pas retirer 4 allumettes
Joueur 1 veuillez saisir un chiffre entre 1, 2 ou 3 :
Votre saisi :3
il reste 10 allumettes
Joueur 2 veuillez saisir un chiffre entre 1, 2 ou 3 :
```

```
il reste 2 allumettes
Joueur 1 veuillez saisir un chiffre entre 1, 2 ou 3 :
Votre saisi :3
Vous ne pouvez pas retirer 3 allumettes
```

```
Joueur 1 veuillez saisir un chiffre entre 1, 2 ou 3 :
Votre saisi :2
Vous ne pouvez pas retirer 2 allumettes
```

```
Joueur 1 veuillez saisir un chiffre entre 1, 2 ou 3 :
Votre saisi :1
il reste 1 allumettes
Joueur 2 a perdu
le score du joueur 1 : 1
le score du joueur 2 : 0
```

```
 Devinette score : 0
Allumette score : 1
Morpion score : 0
Puissance 4 score : 0



 Taper 1 pour si vous voulez continuer à jouer
Taper 2 pour si vous ne voulez plus jouer
Votre saisi :
```

```
Votre saisi :1

- Taper 1 pour jouer au Devinette
- Taper 2 pour jouer à l'Allumettes
- Taper 3 pour jouer au Morpion
- Taper 4 pour jouer au Puissance Quatre
Votre saisi :
```

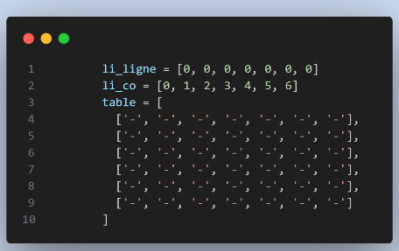
```
Votre saisi :2

Le classement du joueur 1



 classement 4 : Devinette : 0
classement 3 : Morpion : 0
classement 2 : Puissance 4 : 0
classement 1 : Allumette : 1
```

Programme Puissance 4 :

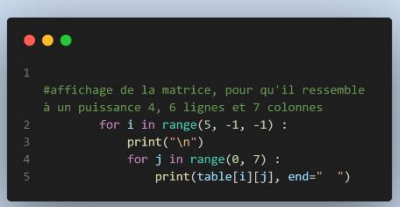
Fonction qui va prendre en paramètre d'entrée le score du joueur 1 et du joueur 2. Et va renvoyer les mêmes scores mais avec des différentes valeurs en fonction de qui a gagné :

- *Initialisation d'une liste à double dimension. Soit 6 sous-listes de 7 cases. Afin d'avoir une matrice de 6 lignes et 7 colonnes*
- *Boucle « for » pour afficher une matrice qui ressemble au jeu de puissance 4*
- *Entrée dans une boucle pour saisir le nombre de partie souhaité*
- *Entrée dans une autre boucle qui vérifie si toute les cases sont remplies ou si l'un des joueurs a gagné ou non*
- *Dans cette boucle, demande à l'utilisateur de saisir seulement la colonne. Vérification de sa réponse (si la case est déjà prise ou non, et s'il a bien saisi un nombre entre 1 et 7)*
- *Si pas de problème, entrée de 0 ou X en fonction du joueur à la case choisie*
- *Sort de cette boucle si l'une des conditions de la boucle est remplie*
- *Attribution du score en fonction du gagnant*
- *Renvoie au programme Menu à la toute fin*

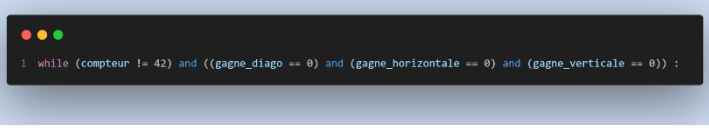
```
def puissance(score1 : int, score2 : int) -> int :
    """
    #-------------------------------------------------------------------------------
    Fonction qui va prendre en paramètre d'entrée, les scores des deux joueurs. Et
    va renvoyer en paramètre de sortie, les scores des deux joueurs après avoir terminés
    les parties. JEU DE PUISSANCE QUATRE
    #-------------------------------------------------------------------------------
    """

    table : list[list[str]]     #liste à double dimension afin d'avoir 3 lignes et 3 colonnes
    li_co : list[int]
    li_ligne : list[int]
    nb_colonne : int
    compteur : int
    gagne_verticale : int
    gagne_horizontale : int
    gagne_diago : int
    i : int
    j : int
    nb_partie : int
    k2 : int
    k3 : int
    k : int

    i = 0
    j = 0
    compteur = 0
    gagne_verticale = 0
    gagne_horizontale = 0
    gagne_diago = 0

    #permuter entre joueur 1 et 2
    #joueur1 = X et joueur2 = O
    k2 = 2
    k3 = 0
    k = 1

    #règle
    print("Jeu de Puissance 4 simple")
    print("X pour le joueur 1")
    print("O pour le joueur 2")

    nb_partie = int(input("\nVeuilez saisir le nombre de partie que vous voulez jouer :"))
```



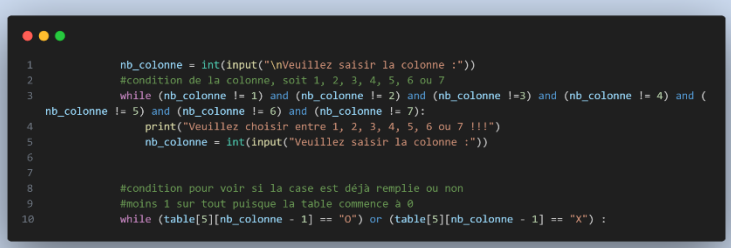
```
        li_ligne = [0, 0, 0, 0, 0, 0, 0]
        li_co = [0, 1, 2, 3, 4, 5, 6]
        table = [
          ['-', '-', '-', '-', '-', '-', '-'],
          ['-', '-', '-', '-', '-', '-', '-'],
          ['-', '-', '-', '-', '-', '-', '-'],
          ['-', '-', '-', '-', '-', '-', '-'],
          ['-', '-', '-', '-', '-', '-', '-'],
          ['-', '-', '-', '-', '-', '-', '-']
        ]
```



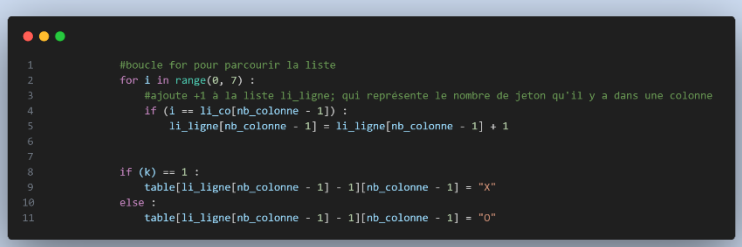
```
#affichage de la matrice, pour qu'il ressemble
à un puissance 4, 6 lignes et 7 colonnes
        for i in range(5, -1, -1) :
            print("\n")
            for j in range(0, 7) :
                print(table[i][j], end="  ")
```



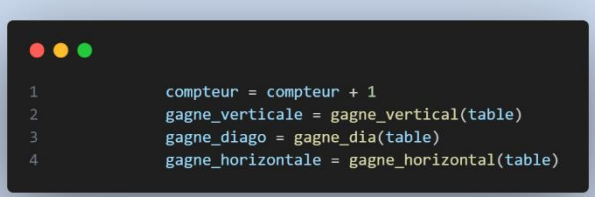
```
while (compteur != 42) and ((gagne_diago == 0) and (gagne_horizontale == 0) and (gagne_verticale == 0)) :
```



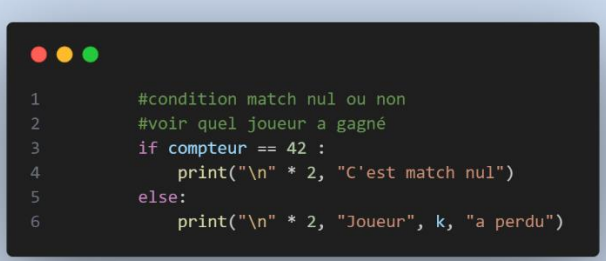
```
            nb_colonne = int(input("\nVeuillez saisir la colonne :"))
            #condition de la colonne, soit 1, 2, 3, 4, 5, 6 ou 7
            while (nb_colonne != 1) and (nb_colonne != 2) and (nb_colonne !=3) and (nb_colonne != 4) and (
nb_colonne != 5) and (nb_colonne != 6) and (nb_colonne != 7):
                print("Veuillez choisir entre 1, 2, 3, 4, 5, 6 ou 7 !!!")
                nb_colonne = int(input("Veuillez saisir la colonne :"))


            #condition pour voir si la case est déjà remplie ou non
            #moins 1 sur tout puisque la table commence à 0
            while (table[5][nb_colonne - 1] == "O") or (table[5][nb_colonne - 1] == "X") :
```



```
            #boucle for pour parcourir la liste
            for i in range(0, 7) :
                #ajoute +1 à la liste li_ligne; qui représente le nombre de jeton qu'il y a dans une colonne
                if (i == li_co[nb_colonne - 1]) :
                    li_ligne[nb_colonne - 1] = li_ligne[nb_colonne - 1] + 1


            if (k) == 1 :
                table[li_ligne[nb_colonne - 1] - 1][nb_colonne - 1] = "X"
            else :
                table[li_ligne[nb_colonne - 1] - 1][nb_colonne - 1] = "O"
```



```
            compteur = compteur + 1
            gagne_verticale = gagne_vertical(table)
            gagne_diago = gagne_dia(table)
            gagne_horizontale = gagne_horizontal(table)
```



```
        #condition match nul ou non
        #voir quel joueur a gagné
        if compteur == 42 :
            print("\n" * 2, "C'est match nul")
        else:
            print("\n" * 2, "Joueur", k, "a perdu")
```

3 fonctions complémentaires, pour la fonction principale du jeu de Puissance 4. Une fonction pour gagner verticalement, une autre pour gagner horizontalement et une dernière pour gagner de manière diagonale : (Dans tous les cas, il faut aligner 4 cases)

Les explications sont quasi les même que le morpion. La différence c'est qu'on teste sur une plus grande liste. Et donc les conditions sont plus grandes

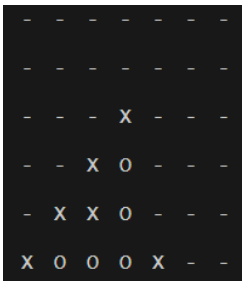
Pour cela, on utilise cette fois, une boucle « while » dans une autre boucle « for » (pour la fonction_vertical et fonction_horizontal)

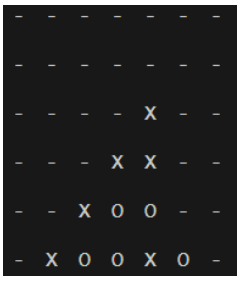
Pour la fonction_dia, on utilisera deux boucle « while » afin d'imiter la boucle « for ». A cause des problèmes d'intervalle de la boucle « for »

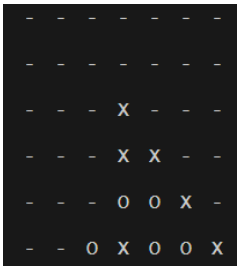
```
    for i in range(0, 7) :
        valid = 0
        trou_1 = 0
        trou_2 = 1
        trou_3 = 2
        trou_4 = 3
        compteur = 0

        #utilisation de "for j" non possible, donc utilisation de while pour l'imiter (for i in range(0, 4))
        while (compteur != 3) :
            compteur = compteur + 1
            if ((table[trou_1][i] == table[trou_2][i]) and (table[trou_2][i] == table[trou_3][i]) and (table[
trou_3][i] == table[trou_4][i]) and (table[trou_1][i] != "-")) :
                valid = valid + 1
                return valid
            trou_1 = trou_1 + 1
            trou_2 = trou_2 + 1
            trou_3 = trou_3 + 1
            trou_4 = trou_4 + 1
```

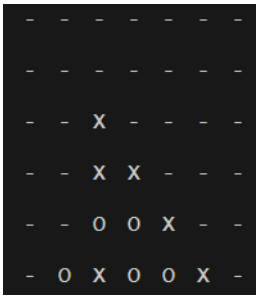
```
    for i in range(0, 6) :
        valid = 0
        trou_1 = 0
        trou_2 = 1
        trou_3 = 2
        trou_4 = 3
        compteur = 0

        #utilisation de "for j" non possible, donc utilisation de while pour l'imiter (for i in range(0, 4))
        while (compteur != 4) :
            compteur = compteur + 1
            if ((table[i][trou_1] == table[i][trou_2]) and (table[i][trou_2] == table[i][trou_3]) and (table[
i][trou_3] == table[i][trou_4]) and (table[i][trou_1] != "-")) :
                valid = valid + 1
                return valid
            trou_1 = trou_1 + 1
            trou_2 = trou_2 + 1
            trou_3 = trou_3 + 1
            trou_4 = trou_4 + 1
```

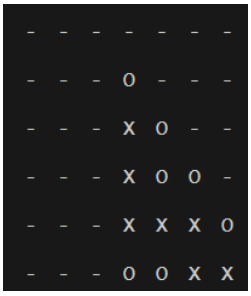
```
    #utilisation de "for j" non possible, donc utilisation de while pour l'imiter (for i in range(0, 3))
    while (compteur_i != 3) :
        compteur_j = 0
        compteur_i = compteur_i + 1
        valid = 0
        trou_1 = 0
        trou_2 = 1
        trou_3 = 2
        trou_4 = 3
        trou_1_i = trou_1_i + 1
        trou_2_i = trou_2_i + 1
        trou_3_i = trou_3_i + 1
        trou_4_i = trou_4_i + 1

        while (compteur_j != 3) :
            compteur_j = compteur_j + 1
            #condition même ligne et même colonne par la gauche (ex : table[0][0] == table[1][1])
            if ((table[trou_1_i][trou_1] == table[trou_2_i][trou_2]) and (table[trou_2_i][trou_2] == table[trou_3_i][trou_3])
and (table[trou_3_i][trou_3] == table[trou_4_i][trou_4]) and (table[trou_1_i][trou_1] != "-")) :
                valid = valid + 1
                return valid
            trou_1 = trou_1 + 1
            trou_2 = trou_2 + 1
            trou_3 = trou_3 + 1
            trou_4 = trou_4 + 1
        trou_1 = 0
        trou_2 = 1
        trou_3 = 2
        trou_4 = 3
        compteur_j = 0

        while (compteur_j != 3) :
            compteur_j = compteur_j + 1
            #condition même ligne et même colonne + 1 par la gauche (ex : table[0][1] == table[1][2])
            if ((table[trou_1_i][trou_1 + 1] == table[trou_2_i][trou_2 + 1]) and (table[trou_2_i][trou_2 + 1] == table[trou_3_i
][trou_3 + 1]) and (table[trou_3_i][trou_3 + 1] == table[trou_4_i][trou_4 + 1]) and (table[trou_1_i][trou_1 + 1] != "-")) :
                valid = valid + 1
                return valid
            trou_1 = trou_1 + 1
            trou_2 = trou_2 + 1
            trou_3 = trou_3 + 1
            trou_4 = trou_4 + 1

        trou_1 = 6
        trou_2 = 5
        trou_3 = 4
        trou_4 = 3
        compteur_j = 0

        while (compteur_j != 3) :
            compteur_j = compteur_j + 1
            #condition même ligne et même colonne par la droite
            if ((table[trou_1_i][trou_1] == table[trou_2_i][trou_2]) and (table[trou_2_i][trou_2] == table[trou_3_i][trou_3])
and (table[trou_3_i][trou_3] == table[trou_4_i][trou_4]) and (table[trou_1_i][trou_1] != "-")) :
                valid = valid + 1
                return valid
            trou_1 = trou_1 - 1
            trou_2 = trou_2 - 1
            trou_3 = trou_3 - 1
            trou_4 = trou_4 - 1
        trou_1 = 6
        trou_2 = 5
        trou_3 = 4
        trou_4 = 3
        compteur_j = 0

        while (compteur_j != 3) :
            compteur_j = compteur_j + 1
            #condition même ligne et même colonne + 1 par la gauche (ex : table[0][5] == table[1][4])
            if ((table[trou_1_i][trou_1 - 1] == table[trou_2_i][trou_2 - 1]) and (table[trou_2_i][trou_2 - 1] == table[trou_3_i
][trou_3 - 1]) and (table[trou_3_i][trou_3 - 1] == table[trou_4_i][trou_4 - 1]) and (table[trou_1_i][trou_1 - 1] != "-")) :
                valid = valid + 1
                return valid
            trou_1 = trou_1 - 1
            trou_2 = trou_2 - 1
            trou_3 = trou_3 - 1
            trou_4 = trou_4 - 1


    compteur_i = 0
    trou_1_i = -1
    trou_2_i = 0
    trou_3_i = 1
    trou_4_i = 2

    #utilisation de "for j" non possible, donc utilisation de while pour l'imiter (for i in range(0, 3))
    while (compteur_i != 2) :
        compteur_j = 0
        compteur_i = compteur_i + 1
        valid = 0
        trou_1 = 0
        trou_2 = 1
        trou_3 = 2
        trou_4 = 3
        trou_1_i = trou_1_i + 1
        trou_2_i = trou_2_i + 1
        trou_3_i = trou_3_i + 1
        trou_4_i = trou_4_i + 1

        while (compteur_j != 4) :
            compteur_j = compteur_j + 1
            #condition même ligne et même colonne + 1 par la gauche (ex : table[1][0] == table[2][1])
            if ((table[trou_1_i - 1][trou_1] == table[trou_2_i - 1][trou_2]) and (table[trou_2_i - 1][trou_2] == table[trou_3_i
- 1][trou_3]) and (table[trou_3_i - 1][trou_3] == table[trou_4_i - 1][trou_4]) and (table[trou_1_i - 1][trou_1] != "-")) :
                valid = valid + 1
                return valid
            trou_1 = trou_1 - 1
            trou_2 = trou_2 - 1
            trou_3 = trou_3 - 1
            trou_4 = trou_4 - 1

        while (compteur_j != 4) :
            compteur_j = compteur_j + 1
            #condition même ligne et même colonne + 1 par la gauche (ex : table[1][0] == table[2][1])
            if ((table[trou_1_i + 1][trou_1] == table[trou_2_i + 1][trou_2]) and (table[trou_2_i + 1][trou_2] == table[trou_3_i
+ 1][trou_3]) and (table[trou_3_i + 1][trou_3] == table[trou_4_i + 1][trou_4]) and (table[trou_1_i + 1][trou_1] != "-")) :
                valid = valid + 1
                return valid
            trou_1 = trou_1 + 1
            trou_2 = trou_2 + 1
            trou_3 = trou_3 + 1
            trou_4 = trou_4 + 1
```

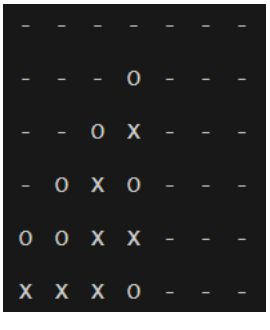




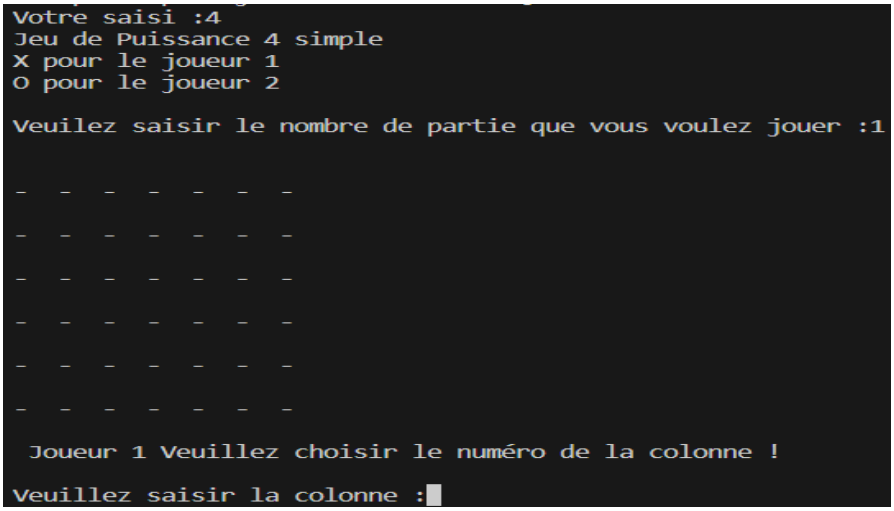


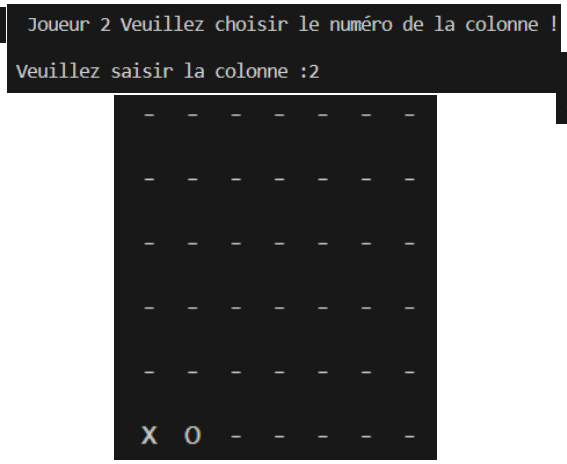


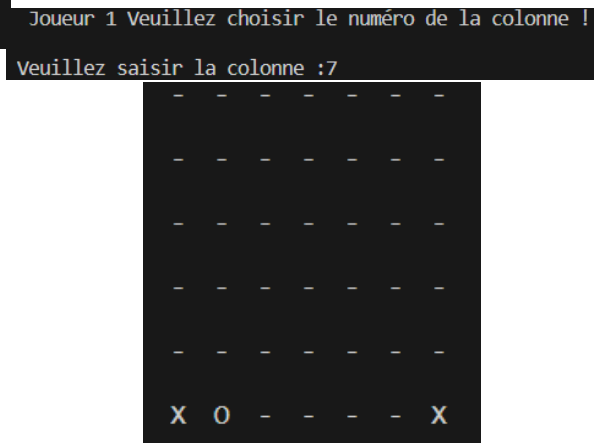




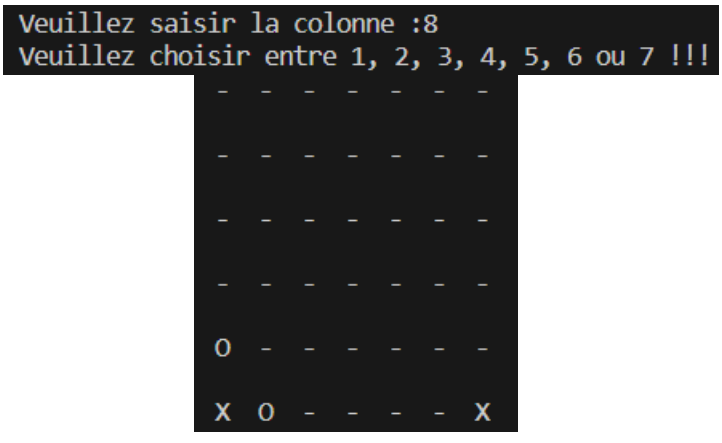
Jeux d'essai :



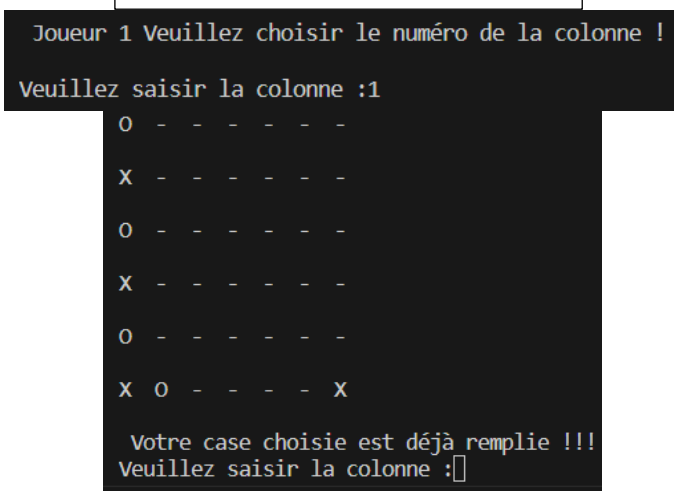




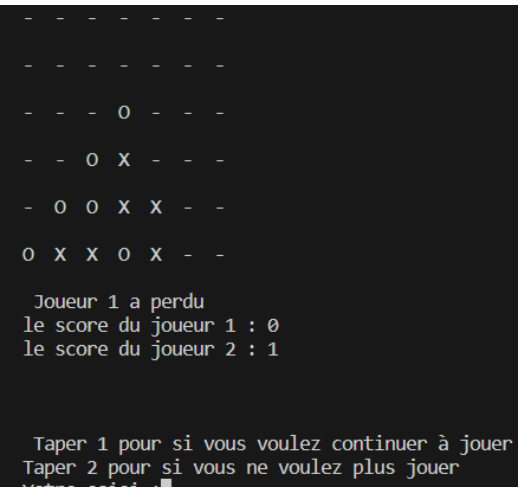
Si on essaye avec 8

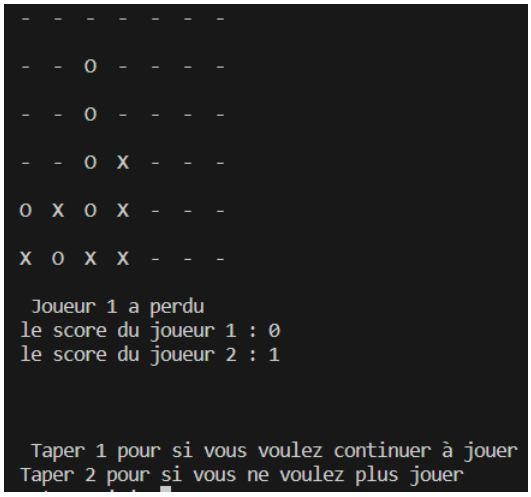


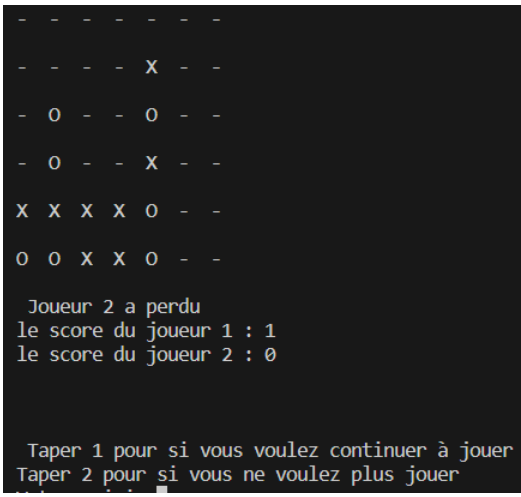
Si on essaye avec une colonne déjà



3 manières de gagner :







Imaginons que le joueur 1 a gagné deux fois et le joueur 2 a gagné une fois. Et que ce soit la fin du programme :