Programmer un jeu de plateau à 2 joueurs : Le morpion

1 Principe du jeu

Il y aura deux joueurs dans une partie. Deux signes représentent chaque joueur. Les signes utilisés dans le jeu sont **X** et **O**. Enfin, il y aura un tableau avec **9** cases.

Voici le visuel du tableau de morpion :

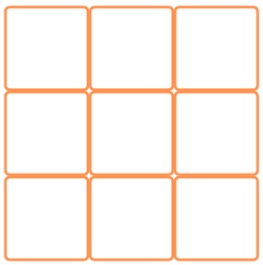
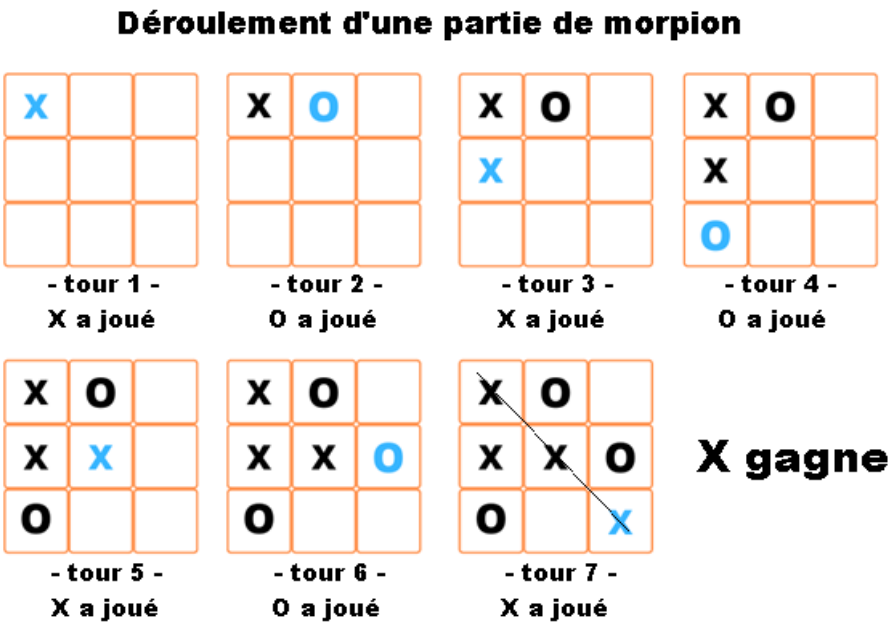


Tableau du morpion

Le déroulement du jeu sera le suivant.

- Tout d'abord, un joueur placera son signe dans l'une des cases vides disponibles.
- Ensuite, le deuxième joueur placera son signe dans l'une des cases vides disponibles.
- Le but des joueurs est de placer leurs signes respectifs complètement en ligne ou en colonne, ou en diagonale.
- Le jeu continue jusqu'à ce qu'un joueur gagne le jeu ou qu'il se termine par une partie nulle en remplissant toutes les cases sans gagnant.

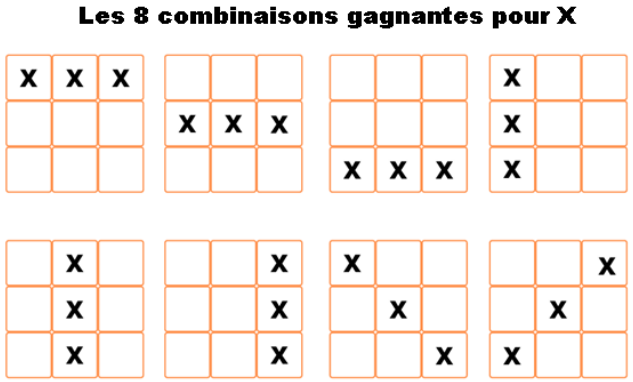
Voyons quelques exemples de jeu visuellement.



Déroulement d'une partie gagnante

Le joueur **X** gagne la partie dans le déroulement ci-dessus. Toutes les cases en diagonale sur la table ont été remplies avec des symboles **X** . Ainsi, le joueur **X** gagne la partie.

Il y a un total de **8** combinaisons permettant de gagner la partie.



2 Algorithme

Cahier des charges : Votre programme contenant plusieurs fonctions écrites en Python aboutira à une réalisation ressemblant à celle visible dans cette vidéo astrovirtuel.fr/nsi/p125-2.mp4

1. Écrivez une fonction **`init_table()`** qui renvoie une nouvelle `table` à 2 dimensions contenant trois lignes de trois éléments. La fonction initialise chaque élément comme vide.
    - Vous pouvez représenter le vide en utilisant n'importe quel symbole que vous aimez. Ici, nous allons utiliser un tiret. '-'.
2. Écrivez une fonction **`check_full(table)`** qui renvoie un booléen pour vérifier si le tableau est rempli.
    - Itérer sur le tableau et renvoyez False si le tableau contient un signe vide ou bien retournez True.
3. Écrivez une fonction **`check_victory(player, table)`** qui renvoie un booléen et qui effectue les tâches :
    - Vérifier toutes les possibilités dont nous avons discuté dans la section précédente.
    - Renvoyer True si le tableau contient une combinaison gagnante ou bien retourner False.
4. Écrivez une fonction **`show_grid(table)`** pour afficher le tableau car nous montrerons le tableau plusieurs fois aux joueurs pendant qu'ils jouent.
5. Écrivez une fonction **`check_choice(player, table)`** qui affiche "X joue" ou bien "O joue" puis qui demande au joueur son choix et enfin qui met à jour la table d'après le choix du joueur.
6. Écrivez une fonction **`change_player(player)`** qui renvoie "X" si `player` vaut "O" et vice versa.
7. Écrivez une fonction **`programme()`** pour démarrer le jeu et qui effectue les tâches suivantes :
    - Affecter à la variable `player` la valeur 'O' ou 'X' aléatoirement
    - Initialiser la table
    - Initialiser la variable `condition_d_arret` à False.
    - Exécuter une boucle non bornée *tant que* `condition_d_arret` différente de True.
        - Montrer le tableau aux joueurs.
        - Demander au joueur actuel dans quel case il joue.
        - Mettre à jour la table.
        - Vérifier si la table a une combinaison gagnante.
        - Vérifier si la table est pleine.
        - Mettre à jour `condition_d_arret.`
        - Changer le joueur.
    - Après la sortie de la boucle non bornée :
    - Si la sortie de boucle s'est faite sur victoire, afficher le nom du gagnant. Sinon afficher "partie nulle".
    - Montrer une dernière fois le tableau aux joueurs.