Principe de l’algorithme

La plupart des jeux de “plateau” à deux joueurs suivent le même principe :

- Un joueur joue
- S’il a gagné, le jeu s’arrête
- Si ce n’est pas le cas, on reprend au premier point avec l’autre joueur

Ce processus se répète jusqu’à ce qu’un des joueurs remporte la partie, ou que la partie s’achève sur une égalité.

Comme on répète tour à tour le même schéma avec chaque joueur, il est intéressant de programmer ce principe par une boucle. Et comme on ne sait pas à l’avance en combien de tours la partie se termine, on utilise une boucle Tant que.

Dans le cas d’un jeu numérique, il faut également afficher l’état du plateau de jeu à chaque tour. Cet état peut être mémorisé dans un tableau à deux dimensions (un tableau de tableaux).

Dans le cas du jeu du Morpion, on peut mémoriser l’état initial du plateau du jeu dans le tableau suivant.

```
jeu = [['.', '.', '.'],
       ['.', '.', '.'],
       ['.', '.', '.']]
```

Principe de l’algorithme

```
On affiche le plateau
On définit qui joue en premier
Tant que la partie n'est pas gagnée :
    Le joueur joue
    On affiche le plateau après son coup
    On vérifie s'il a gagné
        Si oui, la partie est gagnée et le jeu s'arrête ;
        Sinon, c'est au tour de l'autre joueur.
```

Décomposition en tâches

On peut désormais identifier les tâches principales à accomplir pour programmer notre jeu. Chacune de ces tâches sera programmée par une fonction. Voici les fonctions principales nécessaires :

- `afficher_plateau()` : fonction qui affiche l’état du plateau de jeu à l’écran (dans la console pour nous ici)
- `jouer()` : fonction qui demande à l’utilisateur la case qu’il veut jouer et qui actualise le tableau `jeu`
- `verifier_victoire()` : fonction qui vérifie si la partie est gagnée
- `changer_joueur()` : fonction qui change le joueur.

On utilisera les trois variables suivantes :

- `jeu` est le tableau mémorisant l'état du plateau de jeu (voir plus haut) ;
- `joueur` est une chaîne de caractères prenant les valeurs `'X'` ou `'O'` et qui désigne le joueur qui doit jouer ;
- `gagne` est un booléen qui vaut FAUX si la partie n'est pas remportée et VRAI si un des joueurs a gagné. C'est ce booléen qui permettra de mettre fin au jeu dès lors qu'il vaut VRAI.

Entrées et sorties des différentes fonctions

Les fonctions principales étant identifiées, il faut maintenant réfléchir aux entrées et aux sorties de chacune d'elle : c'est-à-dire aux paramètres de la fonction (de quoi a-t-elle besoin pour travailler ?) et aux valeurs renvoyées par la fonction (que doit-elle renvoyer ?)

La fonction `afficher_plateau`

La fonction `afficher_plateau` a besoin de connaître le contenu du tableau `jeu` pour afficher le plateau de jeu à l'écran. Et c'est tout ! Ce sera donc son seul paramètre et cette fonction ne renvoie rien (elle affiche quelque chose uniquement).

```
def afficher_plateau(jeu):
    """Affiche le contenu du tableau jeu."""
```

Exemple :

```
>>> jeu = [['X', 'O', '.'], ['.', '.', '.'], ['.', '.', 'X']]
>>> afficher_plateau(jeu)
['X', 'O', '.']
['.', '.', '.']
['.', '.', 'X']
```

(amélioration possible de l'affichage) :

```
X | O |
--+---+--
  |   |
--+---+--
  |   | X
```

La fonction `jouer`

La fonction `jouer` a pour rôle de demander à l'utilisateur dans quelle ligne et quelle colonne il veut jouer, puis elle doit actualiser et donc modifier l'état du tableau `jeu` en conséquence. Cette fonction a donc a besoin de ce tableau en paramètre. De plus, cette fonction a besoin de connaître le contenu de la variable `joueur` (de type `str`) pour écrire soit `'X'` soit `'O'` dans le tableau `jeu`. Elle possède donc deux paramètres (`jeu` et `joueur`) et ne renvoie rien.

```
def jouer(jeu, joueur):
    """Demande à l'utilisateur la case à jouer et met à jour le tableau jeu."""
```

Exemple :

```
>>> jeu = [['X', 'O', '.'], ['.', '.', '.'], ['.', '.', 'X']]
>>> jouer(jeu, 'O')  # on suppose que le joueur choisit de jouer dans la case centrale
Quelle colonne veux-tu jouer ? (0, 1 ou 2) : 1
Quelle ligne veux-tu jouer ? (0, 1 ou 2) : 1
>>> jeu
[['X', 'O', '.'], ['.', 'O', '.'], ['.', '.', 'X']]
```

La fonction `verifier_victoire`

La fonction `verifier_victoire` a besoin de connaître l'état du plateau donc le tableau `jeu` pour déterminer si la partie est gagnée. Et c'est tout, ce sera son seul paramètre. Cette fonction doit indiquer si la partie est gagnée ou non, elle renverra donc un booléen (VRAI si la partie est gagnée et FAUX sinon).

```
def verifier_victoire(jeu):
    """Renvoie True si la partie est gagnée et False sinon."""
```

Exemples :

```
>>> jeu = [['X', 'X', 'X'], ['O', '.', '.'], ['O', '.', '.']]
>>> verifier_victoire(jeu)
True
>>> jeu = [['O', '.', '.'], ['.', 'X', '.'], ['X', '.', '.']]
>>> verifier_victoire(jeu)
False
```

La fonction `changer_joueur`

La fonction `changer_joueur` a besoin de connaître qui vient de jouer pour passer à l'autre joueur. Et c'est tout, donc elle n'a besoin que du paramètre `joueur`. Cette fonction doit aussi modifier le contenu de la variable `joueur` donc elle doit renvoyer une valeur : la chaîne de caractères `'X'` ou la chaîne de caractères `'O'`.

```
def changer_joueur(joueur):
    """Renvoie la chaîne 'X' si joueur vaut 'O' et la chaîne 'O' si joueur vaut
'X'."""
```

Écriture de l’algorithme

L’algorithme final du jeu peut alors s’écrire de la façon suivante, dans lequel on utilise les variables `jeu`, `joueur`, `gagne`, ainsi que les fonctions `afficher_plateau()`, `jouer()`, `verifier_victoire()`, `changer_joueur()` définies précédemment.

```
afficher_plateau(jeu)
joueur ← 'X'                              # le joueur 'X' commence (arbitraire)
gagne ← FAUX                              # il n'y a pas de gagnant au départ
Tant que non gagne faire                  # tant qu'il n'y a pas de gagnant
    Afficher "Au tour de", joueur
    jouer(jeu, joueur)                    # Le joueur propose sa case et maj du plateau
    afficher_plateau(jeu)
    si verifier_victoire(jeu)              # Si verifie_victoire(jeu) renvoie VRAI
    alors
        gagne ← VRAI                      # le booleen gagne prend la valeur VRAI (ce qui
                                            stoppera la boucle while)
        Afficher "Le joueur", joueur, "a gagné !"
    sinon
        joueur ← changer_joueur(joueur) # sinon on passe au joueur suivant
```