

# Documentation Complète — Morpion Tkinter (Tic-Tac-Toe)

## Présentation du projet

Ce projet est un **jeu de Morpion (Tic-Tac-Toe)** développé avec le langage **Python** en utilisant la bibliothèque **Tkinter** pour l'interface graphique.

Il permet à deux joueurs (X et O) de jouer à tour de rôle sur une grille 3x3, avec une détection automatique de victoire ou d'égalité.

## Technologies utilisées


Technologie	Utilisation
Python 3	Langage principal
Tkinter	Création d'interface graphique
tkinter.messagebox	Affichage des pop-ups de fin de partie



## Structure du projet

✓ MORPION\_TKINTER

> .venv

 main.py

 README.md



## Captures d'écran recommandées

### 1. Démarrage du jeu



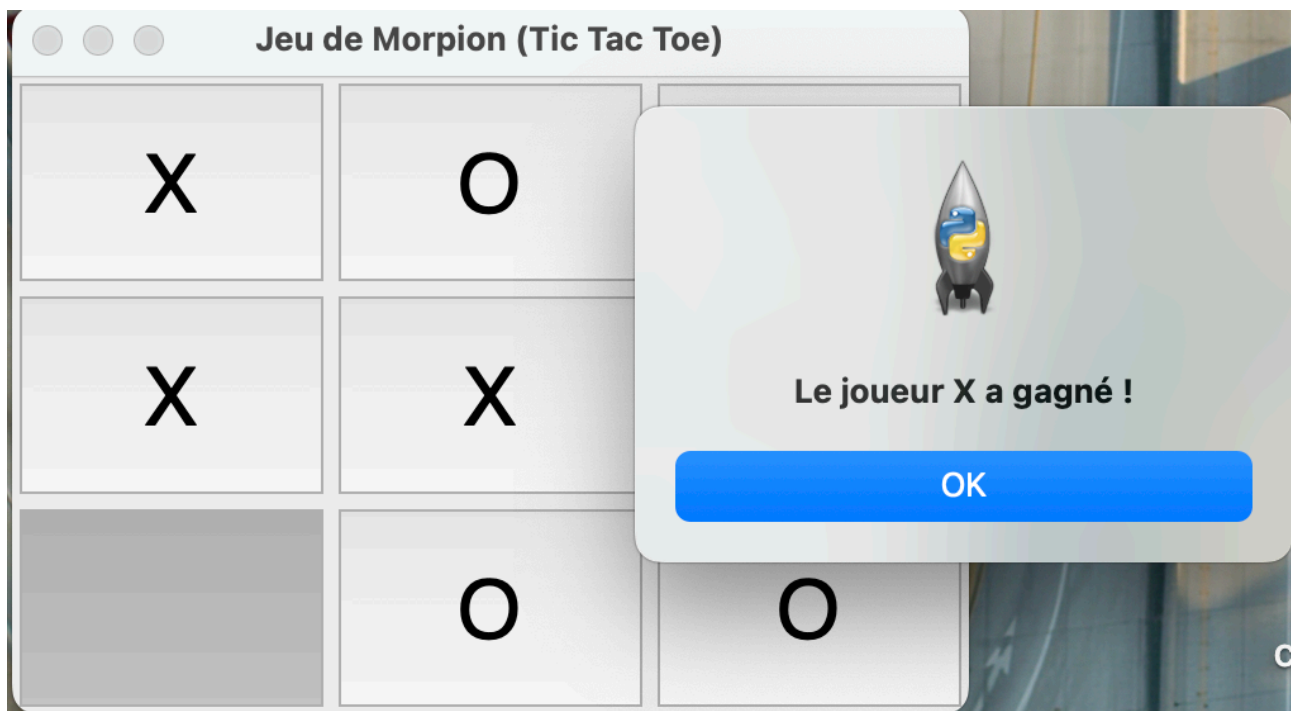
### 2. Partie en cours

- Montrer quelques X et O placés.



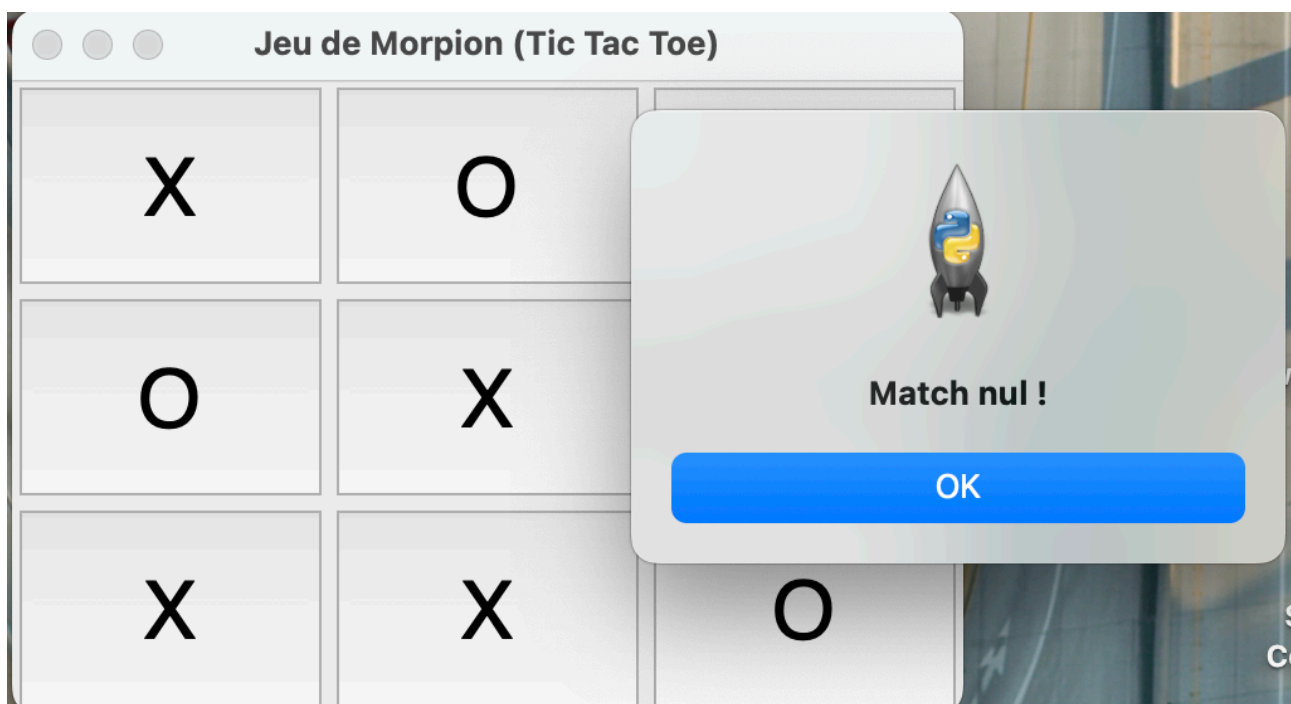
### 3. Message de victoire

- Pop-up annonçant la victoire d'un joueur.



### 4. Égalité

- Grille remplie sans gagnant.





# Étapes d'installation

## **1. Vérifier que Python est installé**

```
python --version
```

Si ce n'est pas installé : <https://www.python.org/downloads/>

```
mkdir morpion_tkinter  
cd morpion_tkinter
```

## **2. Créer le dossier du projet**

## **3. Créer un environnement virtuel (optionnel mais recommandé)**

```
python -m venv .venv
```

Activation :

```
.venv\Scripts\activate
```

### **• Windows :**

```
source .venv/bin/activate
```

### **• macOS / Linux :**

```
python -m tkinter
```

## **4. Vérifier que Tkinter est disponible**

Une fenêtre de test doit s'ouvrir. Si une erreur apparaît :

```
sudo apt install python3-tk
```

- **Ubuntu/Debian :**

```
brew install python-tk
```

- **macOS (avec Homebrew) :**

## **5. Créer et lancer le fichier main.py**

Créer le fichier :

```
touch main.py
```

Y coller le code source donné précédemment.

Lancer le jeu :

```
python main.py
```



## **Explication du code**



### **Classe Morpion**

Elle contient toute la logique du jeu :

- **`__init__`** : Initialise la fenêtre, les joueurs et la grille.
- **`create_widgets`** : Crée les boutons (cases de la grille).
- **`click(row, col)`** : Gère le clic sur une case.
- **`check_winner()`** : Vérifie si un joueur a gagné.
- **`is_draw()`** : Vérifie si la grille est pleine sans gagnant.
- **`reset_game()`** : Réinitialise la grille et le joueur courant.



### **Grille de jeu**

- Représentée par une **liste 2D** self.board contenant les symboles "X" ou "O".
- Les boutons Tkinter sont stockés dans self.buttons.





## Logique de victoire

L'algorithme vérifie :

- Les 3 lignes
- Les 3 colonnes
- Les 2 diagonales

Si une de ces conditions est vraie et que les cases ne sont pas vides ( $\neq \text{" "}$ ), on a un gagnant.

## Améliorations possibles

-  Mode joueur vs ordinateur (IA simple)
-  Interface plus stylisée avec ttk ou customtkinter
-  Compteur de scores (meilleur des 5)
-  Sauvegarde du score dans un fichier

Souhaitez-vous que je vous ajoute un **mode IA**, une **interface sombre**, ou une **fonction d'enregistrement des scores** ?

## Bonnes pratiques utilisées

- Séparation entre **interface** et **logique du jeu**
- Utilisation de **fonctions bien nommées**
- Code **réutilisable** et **modulaire**
- Utilisation de la méthode lambda pour passer les coordonnées aux boutons



## Conclusion

Ce projet est une excellente base pour :

- s'exercer à la programmation Python,
- découvrir l'interface graphique avec Tkinter,
- comprendre les bases de la logique d'un jeu simple.

Il est facilement **évolutif** et personnalisable.

**BOUNGOU MBIMI Gloire Bryan**