

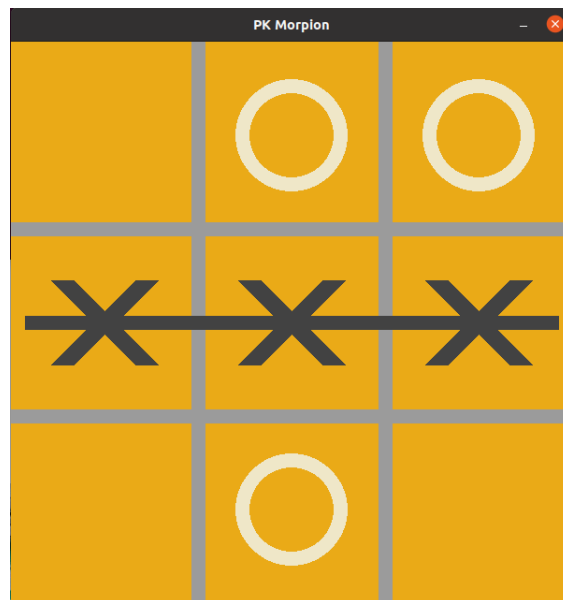


---

# PYTHON Project

## Implementation of mini games

---



Corentin BATARD

12 juin 2023

For the attention of M. Radosław Kycia

## Contents

<b>1</b>	<b>General Introduction</b>	<b>2</b>
<b>2</b>	<b>Main Menu</b>	<b>2</b>
2.1	Goal . . . . .	2
2.2	Important functions . . . . .	2
<b>3</b>	<b>Morpion game</b>	<b>3</b>
3.1	Goal . . . . .	3
3.2	Important functions . . . . .	3
<b>4</b>	<b>Snake game</b>	<b>4</b>
4.1	Goal . . . . .	4
4.2	Important functions . . . . .	4

June 13, 2023

## 1 General Introduction

The aim of this project is to put into practice all the knowledge we have acquired since the beginning of the semester in the Python course. To do this, I chose to create a programme that offers the user several mini-games.

The program can of course offer a multitude of different games, but for lack of time to implement it, I've decided to limit the program to two mini-games: morpion (tic-tac-toe) and snake.

In this report, I'm going to detail the three main code files, that is to say the main and the two games that I implemented myself using the Python 2023 course taught by Mr Radosław Kycia, and resources available on internet.

Please note that this report is only intended to help you understand the project properly. All the codes will of course be provided in the annex.

## 2 Main Menu

### 2.1 Goal

The aim of this program is to create an interface enabling the user to choose the mini-game they want to play from among all the games available.

The program must be able to launch the correct game chosen.

### 2.2 Important functions

All the function in this program are very simple

```
def recover_choice():
    choice = input("Choose a game (1 or 2): ")
    while choice != "1" and choice != "2":
        choice = input("Invalid choice. Please choose 1 or 2 : ")
    return choice
```

The function recover\_choice() checks that the number entered is correct.

```
def main():
    display_menu()
    choice = recover_choice()
    if choice == "1":
        launch_morpion()
    elif choice == "2":
        launch_snake()
```

Here is the main function to launch the choosen game

## 3 Morpion game

### 3.1 Goal

The aim of this programme is to play the well-known game of tic-tac-toe. Each player takes his turn.

The aim of the game is to be the first player to succeed in aligning three of its symbols vertically, horizontally or diagonally.

For a better visual aspect I used pygame

### 3.2 Important functions

To keep things simple, I've represented the game grid as a table with 3 rows and 3 columns. Each square in the table corresponds to a square in the game.

I have initialised an array of zero and I consider that the array can only have three values:

- 0 , which means that the location is available
- 1, which means that the location has been chosen by player 1
- 2 , which means that the location has been chosen by player 2

Here is the implementation of the function that design the pattern , circle for player 1 and cross for player 2

```
1 def genere_design_pattern():
2     for row in range(TABLE_ROW):
3         for col in range(TABLE_COL):
4             if Table[row][col] == 1:
5                 pygame.draw.circle(screen, CIRCLE_COLOUR, (int(col * 200 +
6                     ↪ 100), int(row * 200 + 100)), CIRCLE_RAYON,
7                     ↪ CIRCLE_WIDTH)
8             elif Table[row][col] == 2:
9                 pygame.draw.line(screen, CROSS_COLOUR, (col * 200 +
10                    ↪ SPACE, row * 200 + 200 - SPACE), (col * 200 + 200 -
11                    ↪ SPACE, row * 200 + SPACE), CROSS_WIDTH)
12                 pygame.draw.line(screen, CROSS_COLOUR, (col * 200 +
13                    ↪ SPACE, row * 200 + SPACE), (col * 200 + 200 - SPACE,
14                    ↪ row * 200 + 200 - SPACE), CROSS_WIDTH)
```

To check if a player has won, I just check if 3 squares are equal for each possible direction

```
def play():
    create_line()
    player = 1
    game_over = False
    # Boucle principale du jeu
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
```

```

    sys.exit()
if event.type == pygame.MOUSEBUTTONDOWN and not game_over:
    mouseX = event.pos[0] # Abscissa of mouse position
    mouseY = event.pos[1] # Ordinate of mouse position
    clicked_row = int(mouseY // 200)
    clicked_col = int(mouseX // 200)
    if placement_available(clicked_row, clicked_col):
        placement(clicked_row, clicked_col, player)
        if victory(player) or full_table():
            print(p button to restart)
            game_over = True
        player = 2 if player == 1 else 1
        genere_design_pattern()
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_p:
        restart()
        player = 1
        game_over = False
pygame.display.update()

```

What's important here is to be able to retrieve the box clicked by the user, which I do here using pygame's event module.

## 4 Snake game

### 4.1 Goal

The aim of this programme is to play the well-known game of snake .

The aim of the game is to get the snake to eat as many apples as possible without going off track

As a reminder, the player cannot turn around.

For a better visual aspect I used pygame

### 4.2 Important functions

The snake has several attributes, a body, a position and a direction.

In this game, all the snake's movements are managed by pygame's event module, which retrieves the keys pressed by the user and the resulting if loops on the direction.

I've also introduced the next\_direction variable to block changes in opposite directions.

```

} # Main logic
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        # Whenever a key is pressed down
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP :
                change_to = 'UP'
            if event.key == pygame.K_DOWN :
                change_to = 'DOWN'
            if event.key == pygame.K_LEFT :
                change_to = 'LEFT'
            if event.key == pygame.K_RIGHT :
                change_to = 'RIGHT'
            # Esc -> Create event to quit the game
            if event.key == pygame.K_ESCAPE:
                pygame.event.post(pygame.event.Event(pygame.QUIT))

    # Making sure the snake cannot move in the opposite direction instantaneously
    if change_to == 'UP' and direction != 'DOWN':
        direction = 'UP'
    if change_to == 'DOWN' and direction != 'UP':
        direction = 'DOWN'
    if change_to == 'LEFT' and direction != 'RIGHT':
        direction = 'LEFT'
    if change_to == 'RIGHT' and direction != 'LEFT':
        direction = 'RIGHT'

    # Moving the snake
    if direction == 'UP':
        snake_pos[1] -= 10
    if direction == 'DOWN':
        snake_pos[1] += 10
    if direction == 'LEFT':
        snake_pos[0] -= 10
    if direction == 'RIGHT':
        snake_pos[0] += 10

```

Figure 1: change in snake position

I then added several features:

- change the difficulty
  - save your score
  - display the best scores
- The difficulty correspond to the snake's speed of movement

```

# Difficulty settings
# Simple -> 15
# Normal -> 25
# Difficult -> 50
# Hard -> 100
difficulty = 25
fps_controller.tick(difficulty)

```

The function save is performed thanks to the write of file module

```
# Save score
def save_score(score, login):
    with open('scores.txt', 'a') as file:
        file.write(login + ',' + str(score) + '\n')
```

And the function show\_top\_scores

```
# Show top scores
def show_top_scores():
    scores = []
    # Here we we read the file and retrieve the login and score for
    ↪ each item of data, which are separated in the file by a comma
    with open('scores.txt', 'r') as file:
        for line in file:
            login, score = line.strip().split(',')
            scores.append((login, int(score)))
    # Here we sort in descending order of score
    scores.sort(key=lambda x: x[1], reverse=True)
    score_font = pygame.font.SysFont('times new roman', 30)
    score_y = SCREEN_HEIGHT/4
    game_window.fill(BLACK)
    title_surface = score_font.render('Top Scores:', True, RED)
    title_rect = title_surface.get_rect (center=(SCREEN_WIDTH /
    ↪ 2, score_y))
    game_window.blit(title_surface, title_rect)
    score_y += 30
    # Here is the loop to display the five best scores
    for i in range(min(5, len(scores))):
        score_text = f'{i+1}. {scores[i][0]} - {scores[i][1]}'
        score_surface = score_font.render(score_text, True,
        ↪ WHITE)
        score_rect = score_surface.get_rect(center=(SCREEN_WIDTH
        ↪ / 2, score_y))
        game_window.blit(score_surface, score_rect)
        score_y += 30
    pygame.display.flip()
```