What is memory game

In this game, 10 characters are chosen. Each character is repeated twice. The characters are put in a random order and covered; we will call each character position as a “cell”. Each player chooses two cells, the characters in these two cells are then shown to the players while the rest are still covered. If the two cells’ characters are matching, then the two cell remain shown for the rest of the game and the current player will gain a point. The player with greater score wins.

How does it work

At the start of the game, 10 x 2 cells are rendered on screen in the middle, while a text at the top indicating whose turn it is. At the bottom, two texts are rendered indicating the players’ scores.

The current player, clicks on a cell and it is shown, then clicks on the second and it is also revealed, then wait for 500 ms. After that the game checks if the cells are matching. If they were, a point is added to score and the two cells are shown for the rest game. If the two cells weren’t matching, they are hidden again. Then the player change. The process is repeat until all cells are revealed.

Understanding the code.

There are two main files, the memory.py and the cell.py. Memory.py is where the main game is, while cell.py has a Cell class, that defines what a cell is.

In the Cell class:

Defining the attributes of Cell.

- character: the character at the cell
- position: a tuple containing the coordinates of the top-left corner of the cell on the screen.
- is_revealed: a Boolean representing whether the cell is shown for the rest of the game or not.
- is_shown: a Boolean representing the state of the cell whether shown or not.
- size: an integer representing the size of the cell on the screen
- surface: is the parent object of the cell, the display object in memory.py, it is used to draw the cell.
- font: is a pygame font object used to create text objects
- text: is a object representing the character of the cell, it is used to draw the character on the surface object i.e. display.
- cover: is a Surface object drawn on self.surface when the cell is hidden.
- self.r: is a rect object in pygame, it is used to detect mouse clicks on the cell

functions In Cell

draw(): if either is_revealed or is_shown is true, it draws the text on the top of the cover, this gives a red background for the text.

Otherwise, it draws only the cover with out the text.

on_cell(): it uses the self.r to check if the mouse on the cell or not. It returns a Boolean.

In memory.py:

Defining the global variables:

- WHITE, BLACK are colors
- SCREEN_SIZE, CENTER are coordinates
- DELAY, is the amount time waited after two cells have been chosen
- cells: is a list of the cells in the game
- clicked_cells: is a list of maximum size two, used to store the clicked cells to check afterwards if the cells are matching or not
- counter: counts the number of clicked cells
- player1, player2: are the scores of each player.
- gameover: a Boolean representing whether all cells are revealed or not.

- game_running: a Boolean representing whether the game is running or not.
- font: a font object used to create text objects
- display: the main surface where all texts and cells are drawn.

defining the functions:

all_revealed(): returns true if all cells are revealed

initialize_cells(): creates the cells objects with a random character and specify the position of each cell.

draw_score(): renders the score of each player on the screen

update_screen(): a function that contains all the rendering for the game, including draw_score() and rendering of each cell and the current turn.

main(): where the main loop is, on each iteration the game update_screen. Then listen for a mouse click event. On a mouse click event, if the mouse is on a cell, the cell is shown and counter is increased, if the counter is 2, then game is delayed by half a second. After that, we check if the cells are matching and add a point to the player if that was the case. The process is repeated