INFORMATICS INSTITUTE OF TECHNOLOGY

FOUNDATION CERTIFICATE FOR HIGHER EDUCATION

2018/2019

Name	**: Jawith Afrid**
Student id	**: 2018229**
Module	: Programming (Python)DOC326
Group Name	: janB
Assignment	: Individual Coursework
Date of submission	**:** 1st July 2019

Table of contents:

Contents

Introduction about the problem

Memory game is fun game that you may have played in “Real life” with actual tiles. It is called that because the main skill in the game is your memory how well you can remember the position of tiles.it is also a game that can be recreated on computer and is good example of using computer memory to remember information like with arrays.

While making this game I have many problem .And in my game I have some problems. Time setting while we playing the game the time limit is 60 s but I don’t know how to control it.

The main thing is time is not enough. And in my game “EXIT GAME” button it was not work.

I put that button in game because it is modified the game. The other buttons are working

PSEDEO CODE

```
1. Start
2. GET pygame
3. GET os.path
4. GET sys
5. GET random
6. GET ctypes
7. GET pygame as pg
8. GET random
9. DISPLAY IMAGES_NAME
10. READ IMAGES_EXTENSION = ".png"
11. DISPLAY WINDOW_WIDTH = 1000
12. DISPLAY WINDOW_HEIGHT = 850
13. DISPLAY CARD_WIDTH = 125
14. DISPLAY CARD_HEIGHT = 150
15. DISPLAY MENU_WIDTH =250
16. DISPLAY MENU_HEIGHT = WINDOW_HEIGHT
17. GET last_wrong_time = 0
18. GET  how_many_pairs = 0
19. GET score = 0
20.GET match_time = 0
21. GET leave_button = None
22. GET restart_button = None
23. GET logo = None
24.GET score_sound = None
25.GET flip_sound = None
26.GET victory_sound = None
```

```
27.GET error_sound = None

28.CALL Card

30. CALL Animation

31. IF current_time - self.last_time_called >= self.frame_interval and not self.finished():

          frame = self.images[self.current_index]

          self.last_time_called = current_time

          self.current_index += 1

      ENDIF

          return frame

        ELSE:

          return None

 32.  CALL finished(self):

        return len(self.images) - 1 <= self.current_index

33.CALL draw_load_screen:

34. DISPLAY    logo = pygame.transform.scale

35. DISPLAY pygame.display.set_caption("Python COURSE WORK")

 screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))

36 .DISPLAY score_sound

37. FOR i in range(1, 31):

    frame_name = str(i).zfill(4) + IMAGES_EXTENSION

    cards_back_frames.append(pygame.image.load(os.path.join("animations",
os.path.join(IMAGES_NAME[0], frame_name))))

    draw_load_screen(current_progress / 280)

    current_progress += 1

      ENDFOR

38.FOR i in IMAGES_NAME:

    frames.append(list(cards_back_frames))
```

```
   draw_load_screen(current_progress / 280)
   current_progress += 1
ENDFOR
39.GET index = 0
40.FOR card_name in IMAGES_NAME:
   card_image = pygame.image.load(os.path.join("images", card_name + IMAGES_EXTENSION))
board.append(Card(card_name, card_image))
   board.append(Card(card_name, card_image))
  ENDFOR
41.DISPLAY  current_frames
42. FOR i in range(31, 61):
      frame_name = str(i).zfill(4) + IMAGES_EXTENSION
      current_frames.append(pygame.image.load(os.path.join("animations",
os.path.join(card_name, frame_name))))
      draw_load_screen(current_progress / 280)
      current_progress += 1
     ENDFOR
43.CALL draw_menu():
   # Reset:
   menu_rect = pygame.Rect(0, 0, MENU_WIDTH, MENU_HEIGHT)
   pygame.draw.rect(screen, (255,189,51), menu_rect)
44.GET  start_match
45.FOR card in board:
      card.is_backward = True
 draw_menu()
  create_board()
global score, match_start_time
   ENDFOR
```

```
    46. GET score = 0
    match_start_time = pygame.time.get_ticks()
47.CALL  write_game_data():
    f = pygame.font.Font(os.path.join("fonts", "ROUNDNIB.ttf"), 50)
    text = f.render("SCORE: " + str(score), 1, (0, 0, 0))
    screen.blit(text, (20, 150, 30, 210))
text = f.render("TIME: " + str(match_time) + "S", 1, (0, 0, 0))
    screen.blit(text, (20,600, 30, 210))
    48.CALL create_board():
   49.GET current_index = 0
    50.FOR card in board:
        x = 300 + CARD_WIDTH * (current_index % 4) + 40 * (current_index % 4)
        y = 20 + CARD_HEIGHT * (current_index // 4) + 20 * (current_index // 4)
        rect = pygame.Rect(x, y, CARD_WIDTH, CARD_HEIGHT)
         ENDFOR
51.CALL  click_handler(mouse_x, mouse_y):
    IF leave_button.collidepoint(mouse_x, mouse_y):
        sys.exit()
    ELIF restart_button.collidepoint(mouse_x, mouse_y):
        start_match()
ENDIF
    ELSE:
        52.FOR card in board:
             IF card.card_rectangle.collidepoint(mouse_x, mouse_y) and card.is_backward:
            card.flip_card()
         ENDFOR
            ENDIF
        53. IF count(current_pair) =2:
            IF current_pair[0].card_name != current_pair[1].card_name:
                global last_wrong_time
                last_wrong_time = pygame.time.get_ticks()
                is_wrong = True
                score -= 5
            ELSE:
                pygame.mixer.stop()
                score_sound.play()
                global how_many_pairs
                is_wrong = False
                score += 20
                how_many_pairs += 1
                current_pair.clear()
        ENDIF
        ENDIF
54.GET frames(card_name):
    FOR i in range(0, len(IMAGES_NAME)):
        IF card_name = IMAGES_NAME[i]:
            return frames[i]
    return  None
    ENDFOR
        ENDIF
55.GET wrong_pair():
    error_sound.play()
    card1 = current_pair[0]
```

```
   card2 = current_pair[1]
 current_pair[0].flip_card()
  current_pair[1].flip_card()
   current_pair.clear()
56.CALL check_win():
   IF how_many_pairs =6:
       pygame.mixer.stop()
       victory_sound.play()
       global match_is_running
       match_is_running = False
ENDIF
57.DISPLAY clock
58.CALL run():
   WHILE 1:
       FOR event in pygame.event.get():
           IF event.type = pygame.QUIT:
               sys.exit()
ENDWHILE
      ENDFOR
               ENDIF
           ELIF event.type = pygame.MOUSEBUTTONDOWN and not is_wrong and len(animations)
               clicked_x, clicked_y = pygame.mouse.get_pos()
               click_handler(clicked_x, clicked_y)
          59.DISPLAY time_now
       60.IF is_wrong and time_now - last_wrong_time >= 1000:
           wrong_pair()
            ENDIF
```

```
61. IF match_is_running:
     global match_time
     match_time = (time_now - match_start_time) // 1000
    62.DISPLAY menu()
   63.FOR anim in animations:
     new_frame = anim.update(time_now)
       ENDFOR
    64.IF new_frame is not None:
         pygame.draw.rect(screen, (000, 000, 000), anim.rect)
         new_frame = pygame.transform.scale(new_frame, (CARD_WIDTH, CARD_HEIGHT))
         screen.blit(new_frame, anim.rect)
        ENDIF


     65. IF anim.finished():
         animations.remove(anim)
          ENDIF
    66. IF match_is_running:
     check_win()
           ENDIF
   67.start_match()
   68.run()
   69.END
```

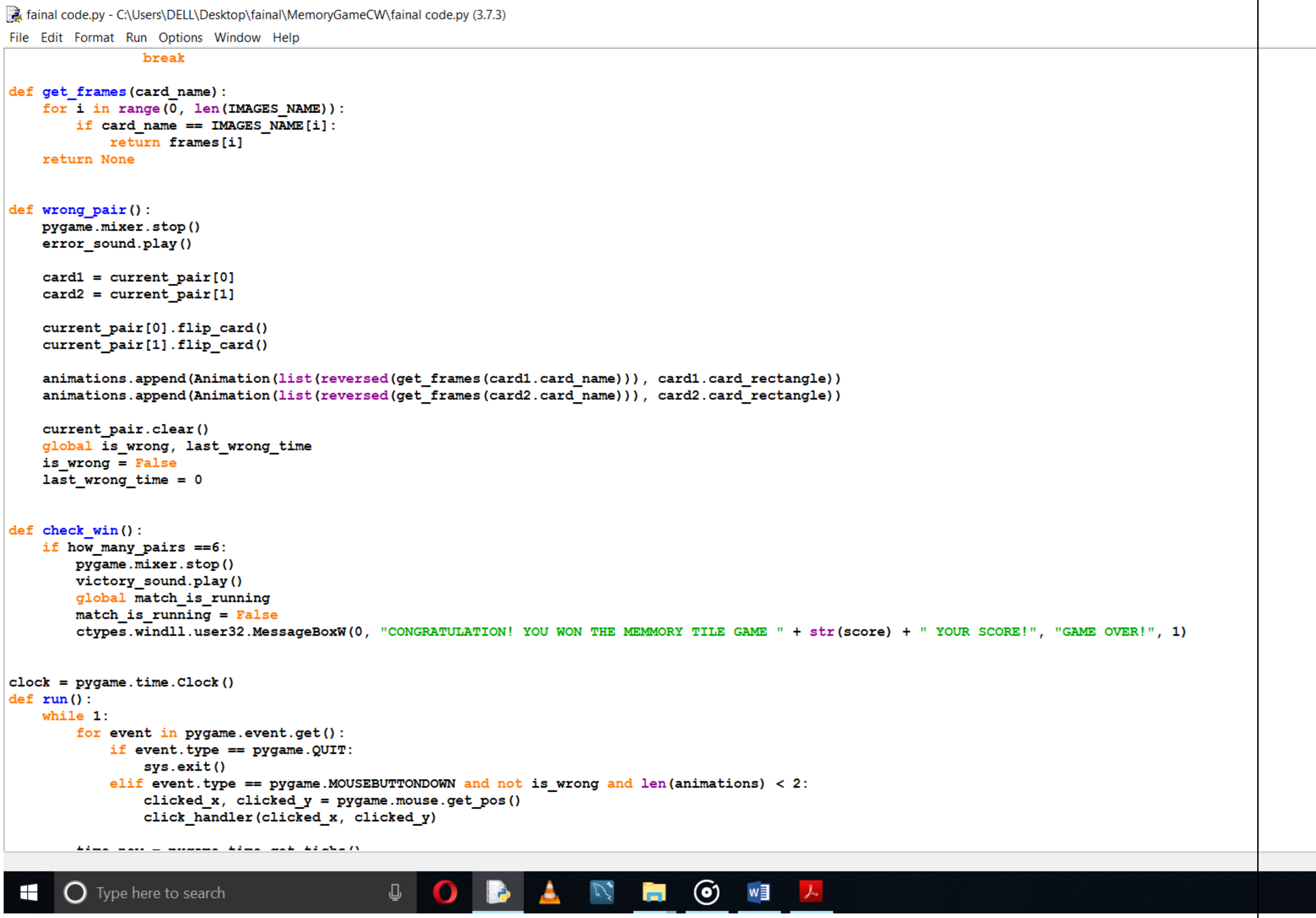
FINAL CODE

```
import pygame

import os.path

import sys

import random

import ctypes

import pygame as pg

import random

IMAGES_NAME = ["card1", "card2", "card3", "card4", "card5", "card6"]


IMAGES_EXTENSION = ".png"


WINDOW_WIDTH = 1000

WINDOW_HEIGHT = 850


# Card data

CARD_WIDTH = 125

CARD_HEIGHT = 150


MENU_WIDTH =250

MENU_HEIGHT = WINDOW_HEIGHT


frames = []


card_backward_image = None

board = []
```

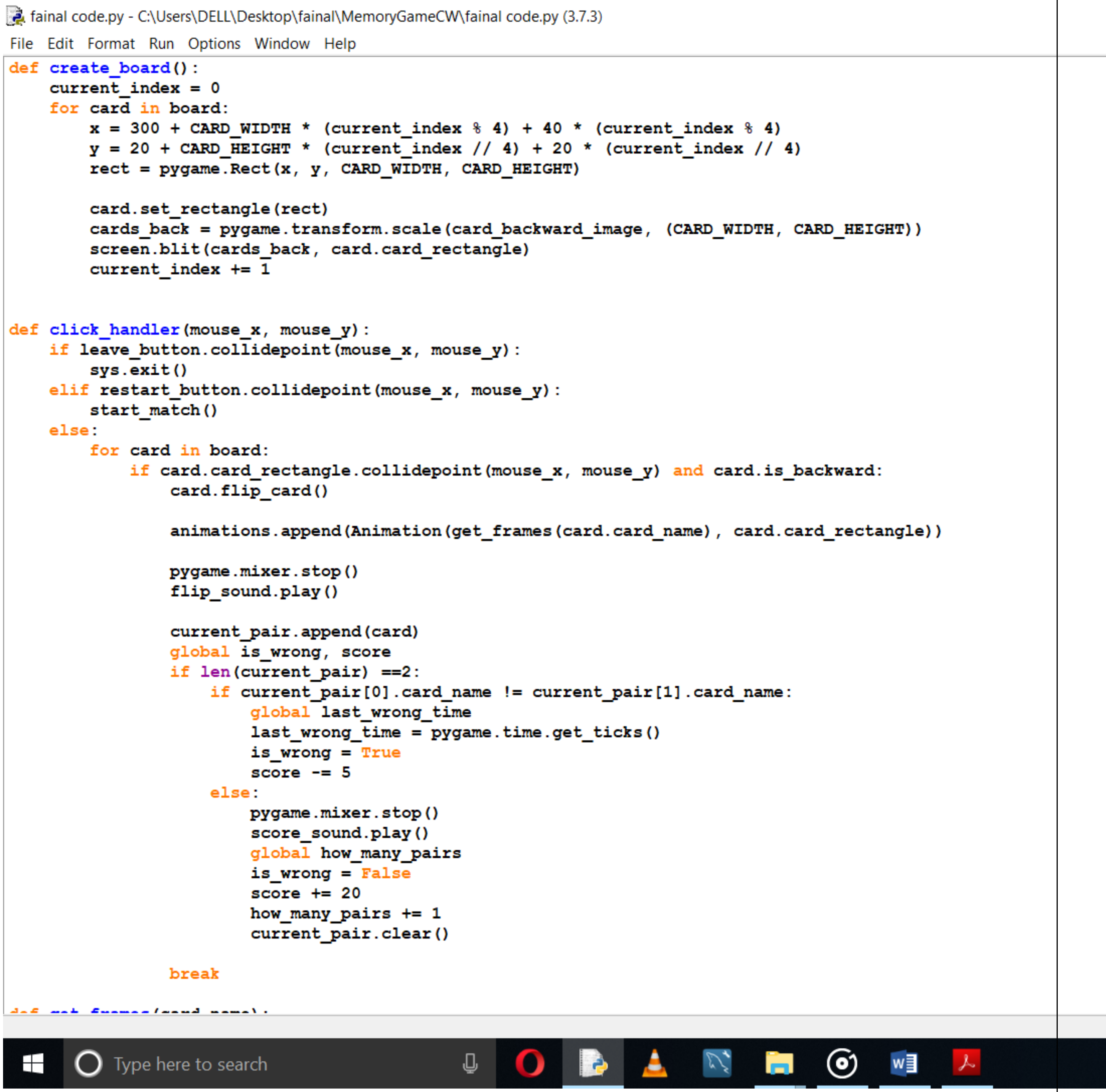
```
animations = []

current_pair = []

is_wrong = False

last_wrong_time = 0

how_many_pairs = 0


score = 0

match_time = 0

match_start_time = None

match_is_running = True

leave_button = None

restart_button = None

logo = None


# Sounds

score_sound = None

flip_sound = None

victory_sound = None

error_sound = None




class Card:

  card_name = ""

  card_image = ""

  card_rectangle = None

  is_backward = True
```

```
    def __init__(self, card_name, card_image, is_backward = True):
        self.card_name = card_name
        self.card_image = card_image
        self.is_backward = is_backward

    def set_rectangle(self, rect):
        self.card_rectangle = rect

    def flip_card(self):
        self.is_backward = not self.is_backward



class Animation:
    images = []
    frame_interval = 0
    current_index = 0
    last_time_called = 0
    rect = None

    def __init__(self, images, rect, frame_interval = 8):
        self.images = images
        self.frame_interval = frame_interval
        self.current_index = 0
        self.rect = rect

    def update(self, current_time):
        if current_time - self.last_time_called >= self.frame_interval and not self.finished():
            frame = self.images[self.current_index]
            self.last_time_called = current_time
            self.current_index += 1
            return frame
        else:
            return None

    def finished(self):
        return len(self.images) - 1 <= self.current_index



def draw_load_screen(progress):
    global logo
    logo = pygame.transform.scale(logo, (int(WINDOW_WIDTH * 0.35), int(WINDOW_WIDTH *
0.20)))
    rect = (int(WINDOW_WIDTH * 0.35), int(WINDOW_HEIGHT * 0.7) - int(WINDOW_WIDTH *
0.12) - 50, int(WINDOW_WIDTH * 0.3), int(WINDOW_WIDTH * 0.12))
    screen.blit(logo, rect)

    progress_bar_border = (int(WINDOW_WIDTH * 0.2), int(WINDOW_HEIGHT * 0.7),
int(WINDOW_WIDTH * 0.6), 500)
    pygame.draw.rect(screen, (0, 0, 0), progress_bar_border)

    progress_bar = (int(WINDOW_WIDTH * 0.2 + 5), int(WINDOW_HEIGHT * 0.7 + 5), int(
(WINDOW_WIDTH * 0.6 - 10) * progress), 820)
    pygame.draw.rect(screen, (0,0,255), progress_bar)
    pygame.display.flip()
pygame.init()

pygame.mixer.init()

pygame.display.set_caption("Python COURSE WORK")

pygame.display.set_icon(pygame.image.load(os.path.join("images", IMAGES_NAME[0] +
IMAGES_EXTENSION)))


screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))

screen.fill((0,0,0))


score_sound = pygame.mixer.Sound(os.path.join("sounds", "point.wav"))

victory_sound = pygame.mixer.Sound(os.path.join("sounds", "victory.wav"))

error_sound = pygame.mixer.Sound(os.path.join("sounds", "error.wav"))

flip_sound = pygame.mixer.Sound(os.path.join("sounds", "flip.wav"))




logo = pygame.image.load(os.path.join("images", "logo2.png"))

logo = pygame.transform.scale(logo, (1000, 500))

draw_load_screen(0 / 280)


card_backward_image = pygame.image.load(os.path.join("images", "find.png"))

draw_load_screen(1 / 280)


current_progress = 2


# Load card's back frames (same for all cards => Reduce loading time)
```

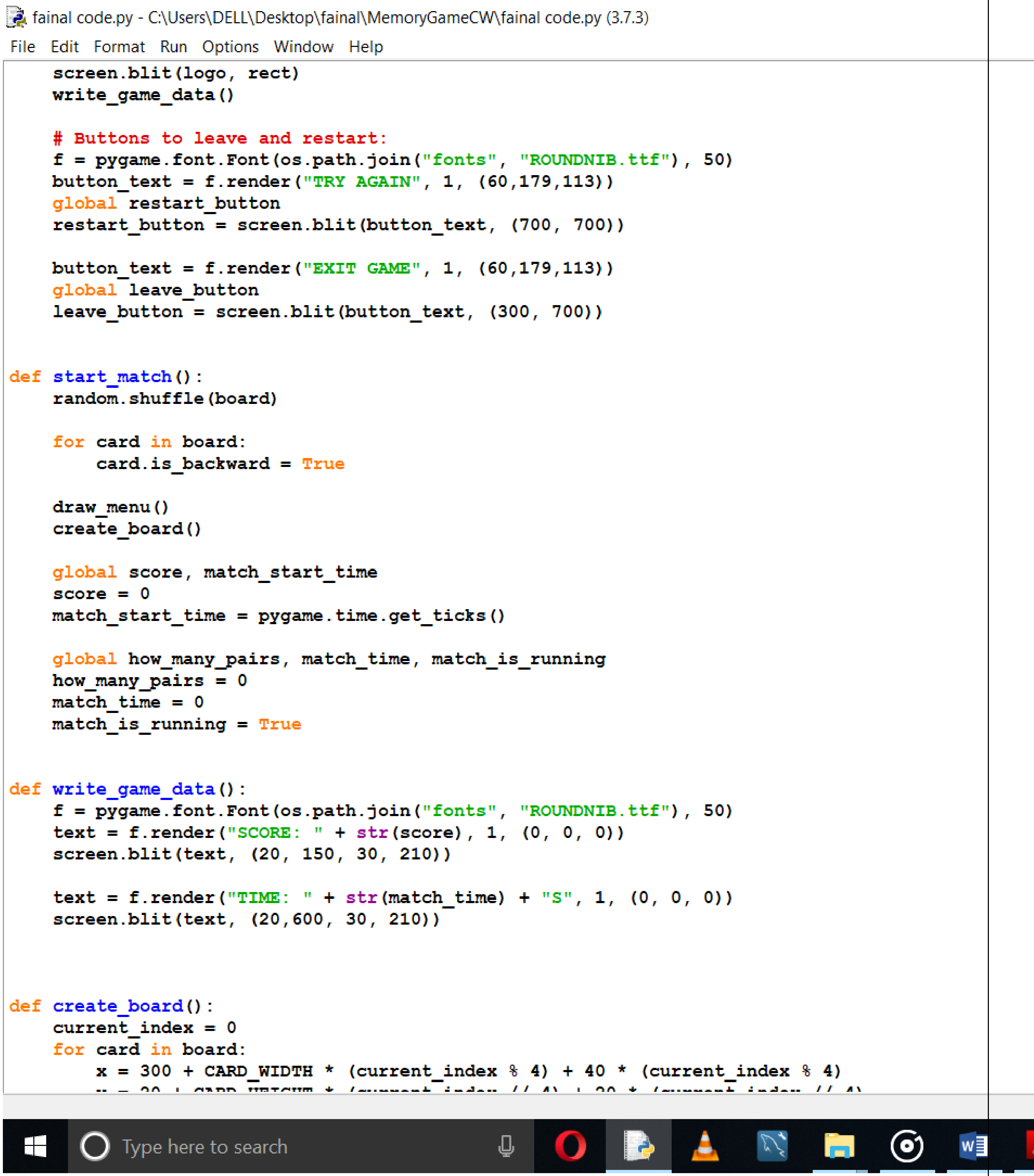
```
cards_back_frames = []
for i in range(1, 31):
    frame_name = str(i).zfill(4) + IMAGES_EXTENSION
    cards_back_frames.append(pygame.image.load(os.path.join("animations",
os.path.join(IMAGES_NAME[0], frame_name))))
    draw_load_screen(current_progress / 280)
    current_progress += 1

for i in IMAGES_NAME:
    frames.append(list(cards_back_frames))
    draw_load_screen(current_progress / 280)
    current_progress += 1

index = 0
for card_name in IMAGES_NAME:
    card_image = pygame.image.load(os.path.join("images", card_name + IMAGES_EXTENSION))

    board.append(Card(card_name, card_image))
    board.append(Card(card_name, card_image))

    current_frames = []
    for i in range(31, 61):
        frame_name = str(i).zfill(4) + IMAGES_EXTENSION
        current_frames.append(pygame.image.load(os.path.join("animations",
os.path.join(card_name, frame_name))))
        draw_load_screen(current_progress / 280)
        current_progress += 1
    frames[index] += current_frames
    index += 1

screen.fill((128,0,0))
pygame.time.delay(100)



def draw_menu():
    # Reset:
    menu_rect = pygame.Rect(0, 0, MENU_WIDTH, MENU_HEIGHT)
    pygame.draw.rect(screen, (255,189,51), menu_rect)

    # Logo:
    global logo
    logo = pygame.transform.scale(logo, (250, 100))
    rect = pygame.Rect(0, 0, MENU_WIDTH, 100)
    screen.blit(logo, rect)
    write_game_data()

    # Buttons to leave and restart:
    f = pygame.font.Font(os.path.join("fonts", "ROUNDNIB.ttf"), 50)
    button_text = f.render("TRY AGAIN", 1, (60,179,113))
    global restart_button
    restart_button = screen.blit(button_text, (700, 700))

    button_text = f.render("EXIT GAME", 1, (60,179,113))
    global leave_button
    leave_button = screen.blit(button_text, (300, 700))



def start_match():
    random.shuffle(board)

    for card in board:
        card.is_backward = True

    draw_menu()
    create_board()

    global score, match_start_time
    score = 0
    match_start_time = pygame.time.get_ticks()

    global how_many_pairs, match_time, match_is_running
    how_many_pairs = 0
    match_time = 0
    match_is_running = True



def write_game_data():
    f = pygame.font.Font(os.path.join("fonts", "ROUNDNIB.ttf"), 50)
    text = f.render("SCORE: " + str(score), 1, (0, 0, 0))
    screen.blit(text, (20, 150, 30, 210))
    text = f.render("TIME: " + str(match_time) + "S", 1, (0, 0, 0))
    screen.blit(text, (20,600, 30, 210))




def create_board():
    current_index = 0
    for card in board:
        x = 300 + CARD_WIDTH * (current_index % 4) + 40 * (current_index % 4)
        y = 20 + CARD_HEIGHT * (current_index // 4) + 20 * (current_index // 4)
        rect = pygame.Rect(x, y, CARD_WIDTH, CARD_HEIGHT)

        card.set_rectangle(rect)
        cards_back = pygame.transform.scale(card_backward_image, (CARD_WIDTH,
CARD_HEIGHT))
        screen.blit(cards_back, card.card_rectangle)
        current_index += 1



def click_handler(mouse_x, mouse_y):
    if leave_button.collidepoint(mouse_x, mouse_y):
        sys.exit()
    elif restart_button.collidepoint(mouse_x, mouse_y):
        start_match()
    else:
        for card in board:
            if card.card_rectangle.collidepoint(mouse_x, mouse_y) and card.is_backward:
                card.flip_card()

                animations.append(Animation(get_frames(card.card_name), card.card_rectangle))

                pygame.mixer.stop()
                flip_sound.play()

                current_pair.append(card)
                global is_wrong, score
                if len(current_pair) ==2:
                    if current_pair[0].card_name != current_pair[1].card_name:
                        global last_wrong_time
                        last_wrong_time = pygame.time.get_ticks()
                        is_wrong = True
                        score -= 5
                    else:
                        pygame.mixer.stop()
                        score_sound.play()
                        global how_many_pairs
                        is_wrong = False
                        score += 20
                        how_many_pairs += 1
                        current_pair.clear()

                break

def get_frames(card_name):
    for i in range(0, len(IMAGES_NAME)):
        if card_name == IMAGES_NAME[i]:
            return frames[i]
    return None



def wrong_pair():
    pygame.mixer.stop()
    error_sound.play()

    card1 = current_pair[0]
    card2 = current_pair[1]

    current_pair[0].flip_card()
    current_pair[1].flip_card()

    animations.append(Animation(list(reversed(get_frames(card1.card_name))),
card1.card_rectangle))
    animations.append(Animation(list(reversed(get_frames(card2.card_name))),
card2.card_rectangle))

    current_pair.clear()
    global is_wrong, last_wrong_time
    is_wrong = False
    last_wrong_time = 0



def check_win():
    if how_many_pairs ==6:
        pygame.mixer.stop()
        victory_sound.play()
        global match_is_running
        match_is_running = False
        ctypes.windll.user32.MessageBoxW(0, "CONGRATULATION! YOU WON THE MEMMORY
TILE GAME " + str(score) + " YOUR SCORE!", "GAME OVER!", 1)



clock = pygame.time.Clock()
def run():
    while 1:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.MOUSEBUTTONDOWN and not is_wrong and len(animations)
< 2:
                clicked_x, clicked_y = pygame.mouse.get_pos()
                click_handler(clicked_x, clicked_y)

        time_now = pygame.time.get_ticks()

        clock.tick()
        global fps
        fps = int(clock.get_fps())
        draw_menu()

        if is_wrong and time_now - last_wrong_time >= 1000:
            wrong_pair()

        if match_is_running:
            global match_time
            match_time = (time_now - match_start_time) // 1000
            draw_menu()

        for anim in animations:
            new_frame = anim.update(time_now)
            if new_frame is not None:
                pygame.draw.rect(screen, (000, 000, 000), anim.rect)
                new_frame = pygame.transform.scale(new_frame, (CARD_WIDTH, CARD_HEIGHT))
                screen.blit(new_frame, anim.rect)

            if anim.finished():
                animations.remove(anim)

        pygame.display.flip()

        if match_is_running:
            check_win()
start_match()
run()
```


fainal code.py - C:\Users\DELL\Desktop\fainal\MemoryGameCW\fainal code.py (3.7.3)
File Edit Format Run Options Window Help
Type here to search

```
                break

def get_frames(card_name):
    for i in range(0, len(IMAGES_NAME)):
        if card_name == IMAGES_NAME[i]:
            return frames[i]
    return None


def wrong_pair():
    pygame.mixer.stop()
    error_sound.play()

    card1 = current_pair[0]
    card2 = current_pair[1]

    current_pair[0].flip_card()
    current_pair[1].flip_card()

    animations.append(Animation(list(reversed(get_frames(card1.card_name))), card1.card_rectangle))
    animations.append(Animation(list(reversed(get_frames(card2.card_name))), card2.card_rectangle))

    current_pair.clear()
    global is_wrong, last_wrong_time
    is_wrong = False
    last_wrong_time = 0


def check_win():
    if how_many_pairs ==6:
        pygame.mixer.stop()
        victory_sound.play()
        global match_is_running
        match_is_running = False
        ctypes.windll.user32.MessageBoxW(0, "CONGRATULATION! YOU WON THE MEMMORY TILE GAME " + str(score) + " YOUR SCORE!", "GAME OVER!", 1)


clock = pygame.time.Clock()
def run():
    while 1:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.MOUSEBUTTONDOWN and not is_wrong and len(animations) < 2:
                clicked_x, clicked_y = pygame.mouse.get_pos()
                click_handler(clicked_x, clicked_y)
```

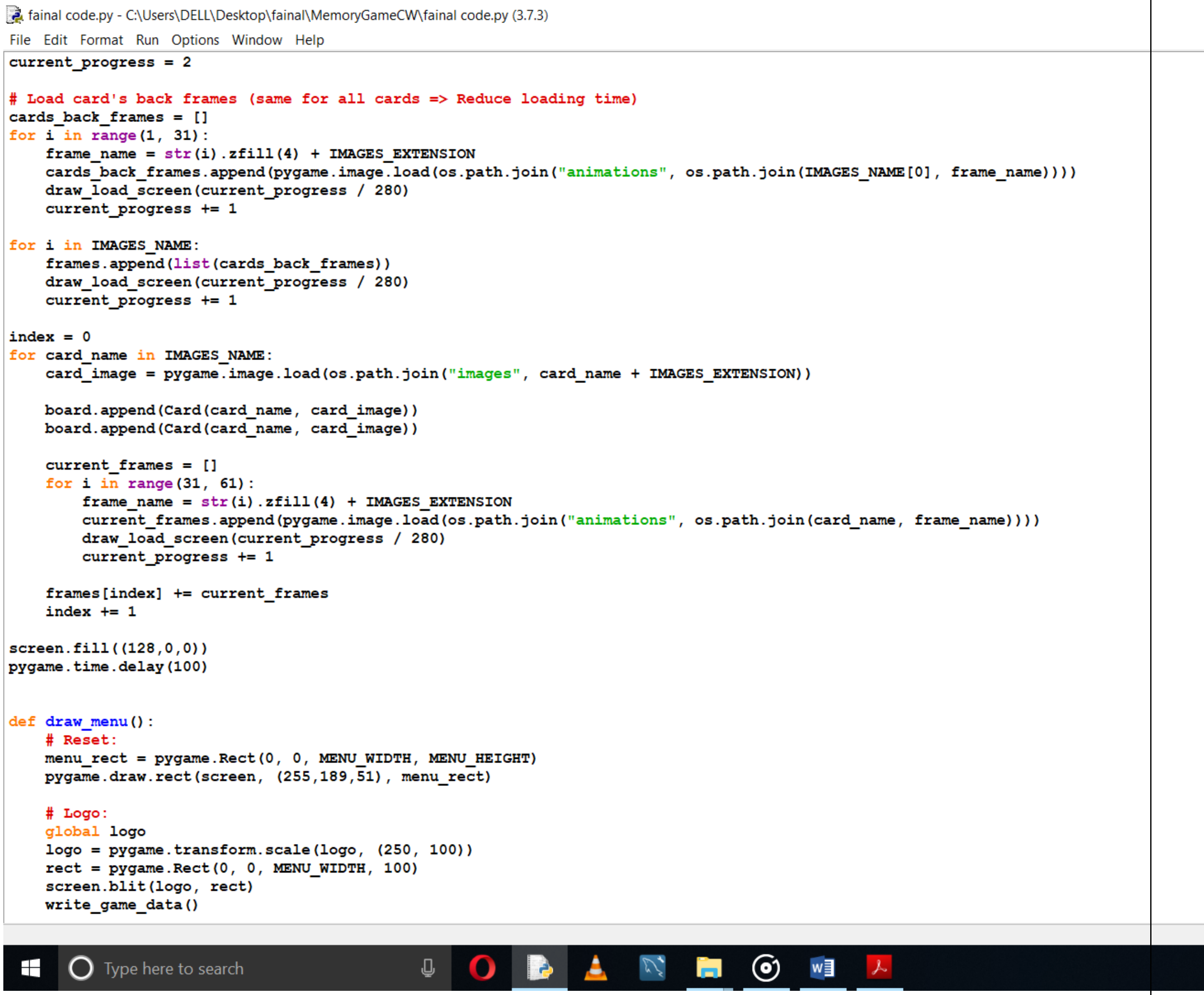

fainal code.py - C:\Users\DELL\Desktop\fainal\MemoryGameCW\fainal code.py (3.7.3)
File Edit Format Run Options Window Help
Type here to search

```python
def create_board():
    current_index = 0
    for card in board:
        x = 300 + CARD_WIDTH * (current_index % 4) + 40 * (current_index % 4)
        y = 20 + CARD_HEIGHT * (current_index // 4) + 20 * (current_index // 4)
        rect = pygame.Rect(x, y, CARD_WIDTH, CARD_HEIGHT)

        card.set_rectangle(rect)
        cards_back = pygame.transform.scale(card_backward_image, (CARD_WIDTH, CARD_HEIGHT))
        screen.blit(cards_back, card.card_rectangle)
        current_index += 1


def click_handler(mouse_x, mouse_y):
    if leave_button.collidepoint(mouse_x, mouse_y):
        sys.exit()
    elif restart_button.collidepoint(mouse_x, mouse_y):
        start_match()
    else:
        for card in board:
            if card.card_rectangle.collidepoint(mouse_x, mouse_y) and card.is_backward:
                card.flip_card()

                animations.append(Animation(get_frames(card.card_name), card.card_rectangle))

                pygame.mixer.stop()
                flip_sound.play()

                current_pair.append(card)
                global is_wrong, score
                if len(current_pair) ==2:
                    if current_pair[0].card_name != current_pair[1].card_name:
                        global last_wrong_time
                        last_wrong_time = pygame.time.get_ticks()
                        is_wrong = True
                        score -= 5
                    else:
                        pygame.mixer.stop()
                        score_sound.play()
                        global how_many_pairs
                        is_wrong = False
                        score += 20
                        how_many_pairs += 1
                    current_pair.clear()

                break
```

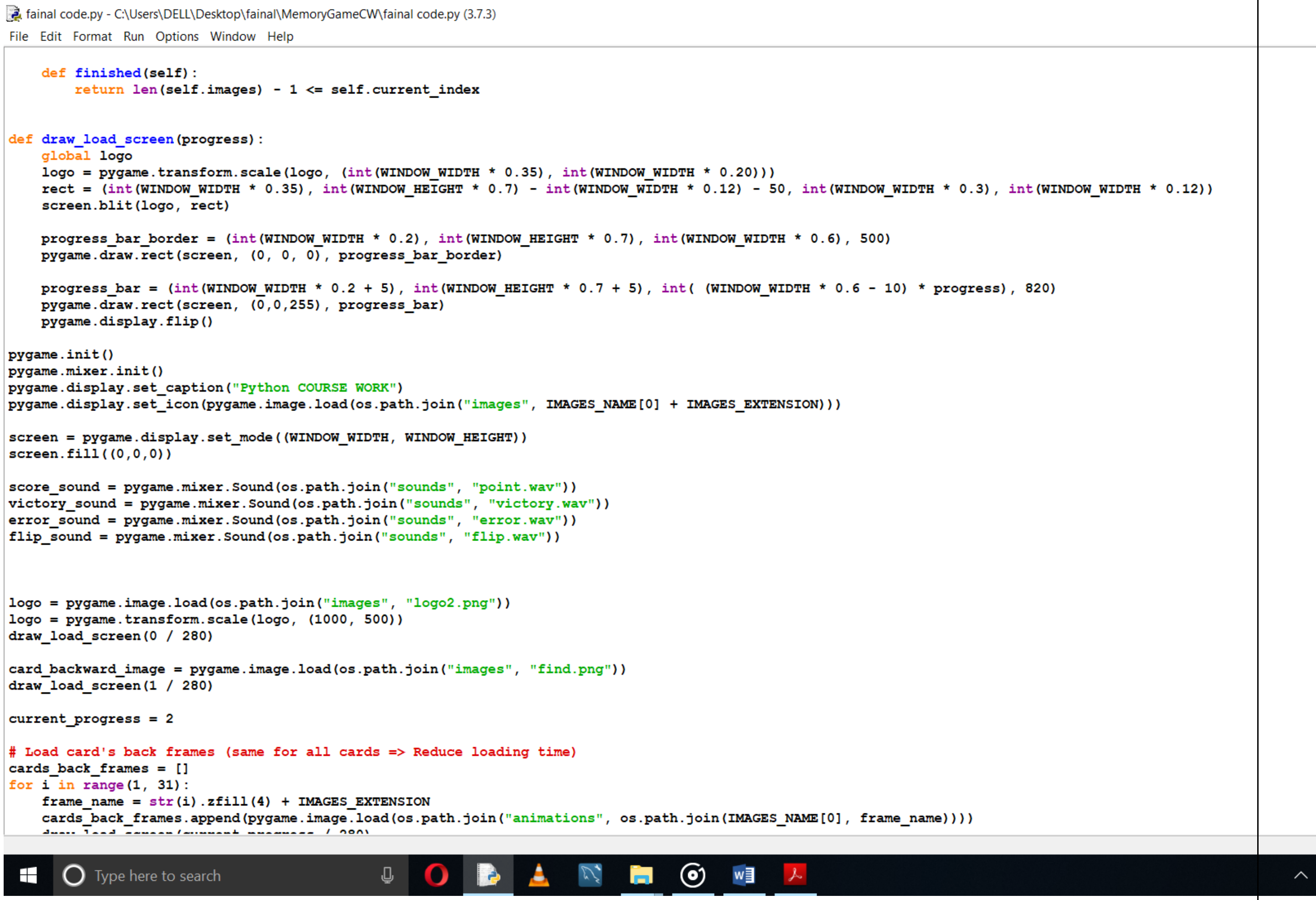


fainal code.py - C:\Users\DELL\Desktop\fainal\MemoryGameCW\fainal code.py (3.7.3)

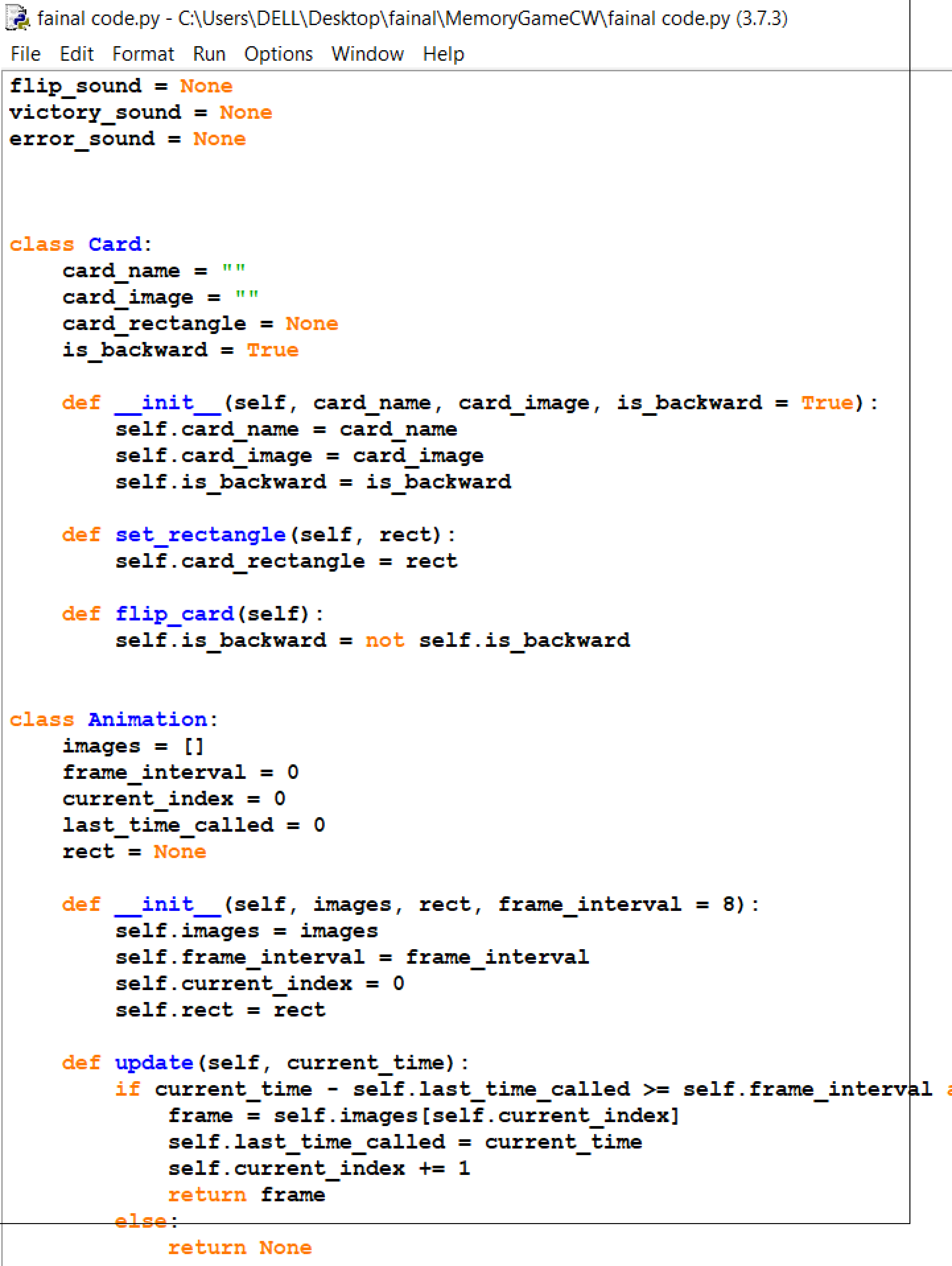
```
    screen.blit(logo, rect)
    write_game_data()

    # Buttons to leave and restart:
    f = pygame.font.Font(os.path.join("fonts", "ROUNDNIB.ttf"), 50)
    button_text = f.render("TRY AGAIN", 1, (60,179,113))
    global restart_button
    restart_button = screen.blit(button_text, (700, 700))

    button_text = f.render("EXIT GAME", 1, (60,179,113))
    global leave_button
    leave_button = screen.blit(button_text, (300, 700))


def start_match():
    random.shuffle(board)

    for card in board:
        card.is_backward = True

    draw_menu()
    create_board()

    global score, match_start_time
    score = 0
    match_start_time = pygame.time.get_ticks()

    global how_many_pairs, match_time, match_is_running
    how_many_pairs = 0
    match_time = 0
    match_is_running = True


def write_game_data():
    f = pygame.font.Font(os.path.join("fonts", "ROUNDNIB.ttf"), 50)
    text = f.render("SCORE: " + str(score), 1, (0, 0, 0))
    screen.blit(text, (20, 150, 30, 210))

    text = f.render("TIME: " + str(match_time) + "S", 1, (0, 0, 0))
    screen.blit(text, (20,600, 30, 210))



def create_board():
    current_index = 0
    for card in board:
        x = 300 + CARD_WIDTH * (current_index % 4) + 40 * (current_index % 4)
        y = 20 + CARD_HEIGHT * (current_index // 4) + 20 * (current_index // 4)
```


fainal code.py - C:\Users\DELL\Desktop\fainal\MemoryGameCW\fainal code.py (3.7.3)
File Edit Format Run Options Window Help
Type here to search

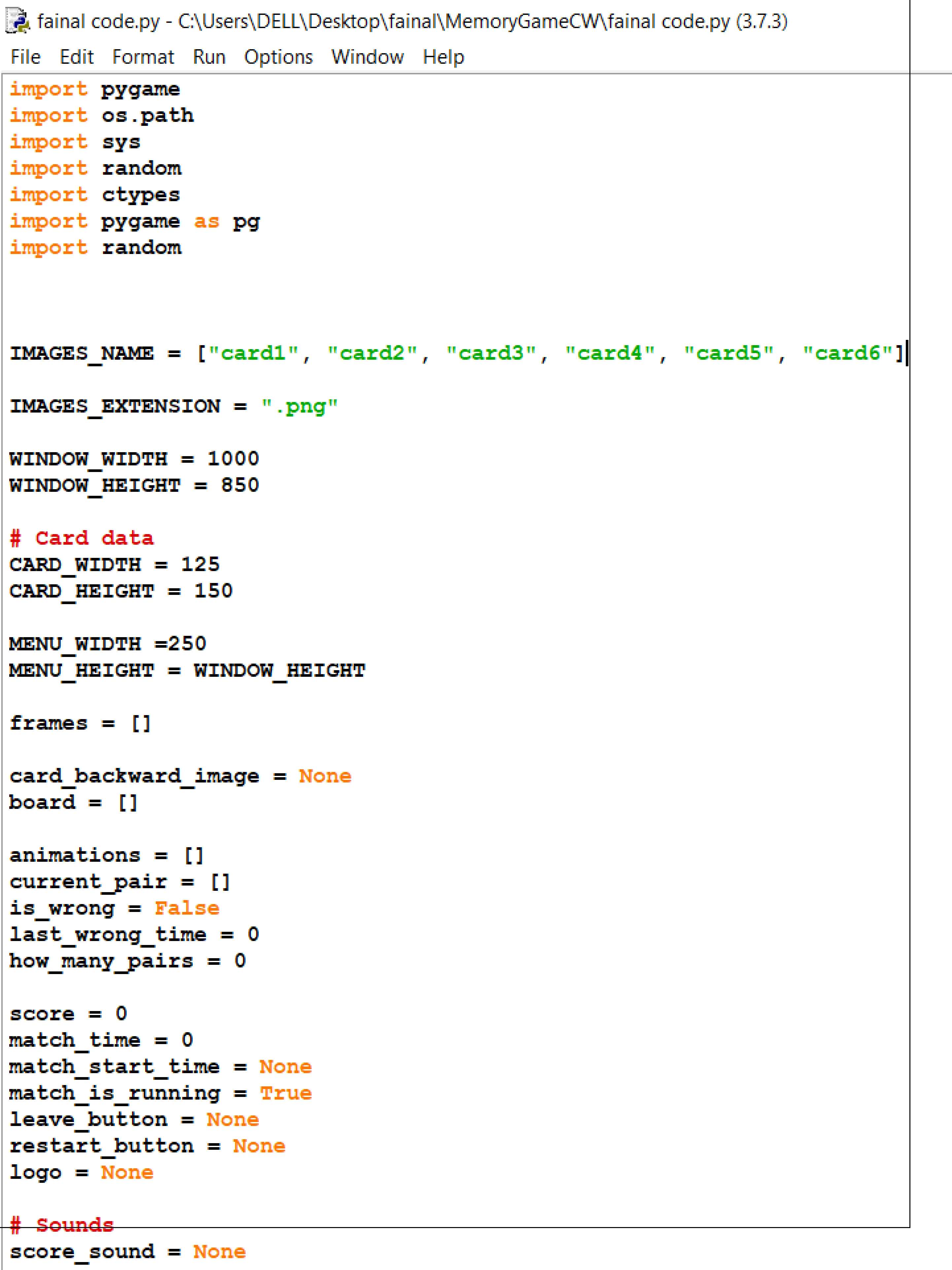
```
current_progress = 2

# Load card's back frames (same for all cards => Reduce loading time)
cards_back_frames = []
for i in range(1, 31):
    frame_name = str(i).zfill(4) + IMAGES_EXTENSION
    cards_back_frames.append(pygame.image.load(os.path.join("animations", os.path.join(IMAGES_NAME[0], frame_name))))
    draw_load_screen(current_progress / 280)
    current_progress += 1

for i in IMAGES_NAME:
    frames.append(list(cards_back_frames))
    draw_load_screen(current_progress / 280)
    current_progress += 1

index = 0
for card_name in IMAGES_NAME:
    card_image = pygame.image.load(os.path.join("images", card_name + IMAGES_EXTENSION))

    board.append(Card(card_name, card_image))
    board.append(Card(card_name, card_image))

    current_frames = []
    for i in range(31, 61):
        frame_name = str(i).zfill(4) + IMAGES_EXTENSION
        current_frames.append(pygame.image.load(os.path.join("animations", os.path.join(card_name, frame_name))))
        draw_load_screen(current_progress / 280)
        current_progress += 1

    frames[index] += current_frames
    index += 1

screen.fill((128,0,0))
pygame.time.delay(100)


def draw_menu():
    # Reset:
    menu_rect = pygame.Rect(0, 0, MENU_WIDTH, MENU_HEIGHT)
    pygame.draw.rect(screen, (255,189,51), menu_rect)

    # Logo:
    global logo
    logo = pygame.transform.scale(logo, (250, 100))
    rect = pygame.Rect(0, 0, MENU_WIDTH, 100)
    screen.blit(logo, rect)
    write_game_data()
```


fainal code.py - C:\Users\DELL\Desktop\fainal\MemoryGameCW\fainal code.py (3.7.3)

```python
    def finished(self):
        return len(self.images) - 1 <= self.current_index


def draw_load_screen(progress):
    global logo
    logo = pygame.transform.scale(logo, (int(WINDOW_WIDTH * 0.35), int(WINDOW_WIDTH * 0.20)))
    rect = (int(WINDOW_WIDTH * 0.35), int(WINDOW_HEIGHT * 0.7) - int(WINDOW_WIDTH * 0.12) - 50, int(WINDOW_WIDTH * 0.3), int(WINDOW_WIDTH * 0.12))
    screen.blit(logo, rect)

    progress_bar_border = (int(WINDOW_WIDTH * 0.2), int(WINDOW_HEIGHT * 0.7), int(WINDOW_WIDTH * 0.6), 500)
    pygame.draw.rect(screen, (0, 0, 0), progress_bar_border)

    progress_bar = (int(WINDOW_WIDTH * 0.2 + 5), int(WINDOW_HEIGHT * 0.7 + 5), int( (WINDOW_WIDTH * 0.6 - 10) * progress), 820)
    pygame.draw.rect(screen, (0,0,255), progress_bar)
    pygame.display.flip()

pygame.init()
pygame.mixer.init()
pygame.display.set_caption("Python COURSE WORK")
pygame.display.set_icon(pygame.image.load(os.path.join("images", IMAGES_NAME[0] + IMAGES_EXTENSION)))

screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
screen.fill((0,0,0))

score_sound = pygame.mixer.Sound(os.path.join("sounds", "point.wav"))
victory_sound = pygame.mixer.Sound(os.path.join("sounds", "victory.wav"))
error_sound = pygame.mixer.Sound(os.path.join("sounds", "error.wav"))
flip_sound = pygame.mixer.Sound(os.path.join("sounds", "flip.wav"))



logo = pygame.image.load(os.path.join("images", "logo2.png"))
logo = pygame.transform.scale(logo, (1000, 500))
draw_load_screen(0 / 280)

card_backward_image = pygame.image.load(os.path.join("images", "find.png"))
draw_load_screen(1 / 280)

current_progress = 2

# Load card's back frames (same for all cards => Reduce loading time)
cards_back_frames = []
for i in range(1, 31):
    frame_name = str(i).zfill(4) + IMAGES_EXTENSION
    cards_back_frames.append(pygame.image.load(os.path.join("animations", os.path.join(IMAGES_NAME[0], frame_name))))
```


fainal code.py - C:\Users\DELL\Desktop\fainal\MemoryGameCW\fainal code.py (3.7.3)
File Edit Format Run Options Window Help

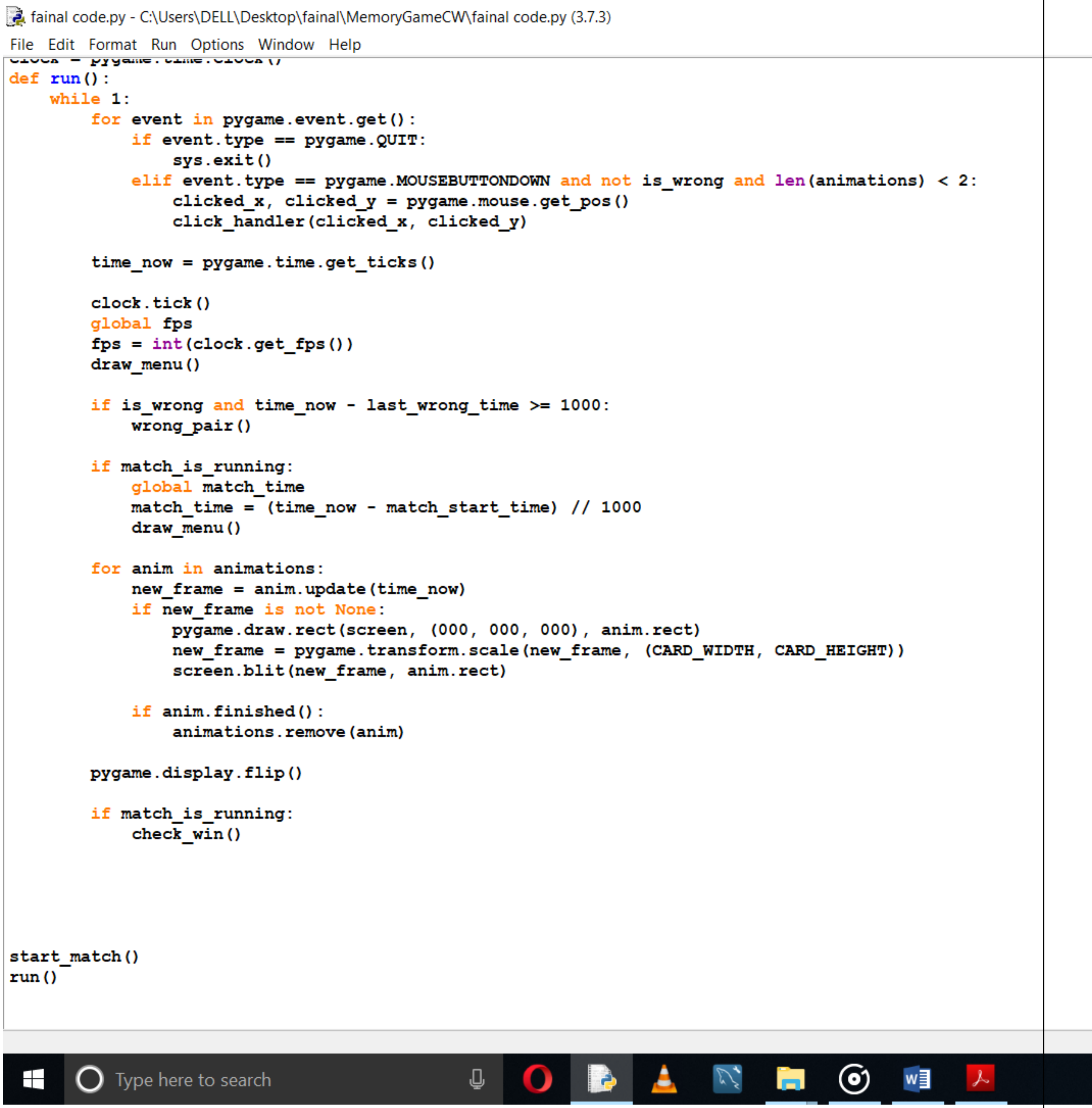
```
flip_sound = None
victory_sound = None
error_sound = None



class Card:
    card_name = ""
    card_image = ""
    card_rectangle = None
    is_backward = True

    def __init__(self, card_name, card_image, is_backward = True):
        self.card_name = card_name
        self.card_image = card_image
        self.is_backward = is_backward

    def set_rectangle(self, rect):
        self.card_rectangle = rect

    def flip_card(self):
        self.is_backward = not self.is_backward


class Animation:
    images = []
    frame_interval = 0
    current_index = 0
    last_time_called = 0
    rect = None

    def __init__(self, images, rect, frame_interval = 8):
        self.images = images
        self.frame_interval = frame_interval
        self.current_index = 0
        self.rect = rect

    def update(self, current_time):
        if current_time - self.last_time_called >= self.frame_interval a
            frame = self.images[self.current_index]
            self.last_time_called = current_time
            self.current_index += 1
            return frame
        else:
            return None
```



fainal code.py - C:\Users\DELL\Desktop\fainal\MemoryGameCW\fainal code.py (3.7.3)

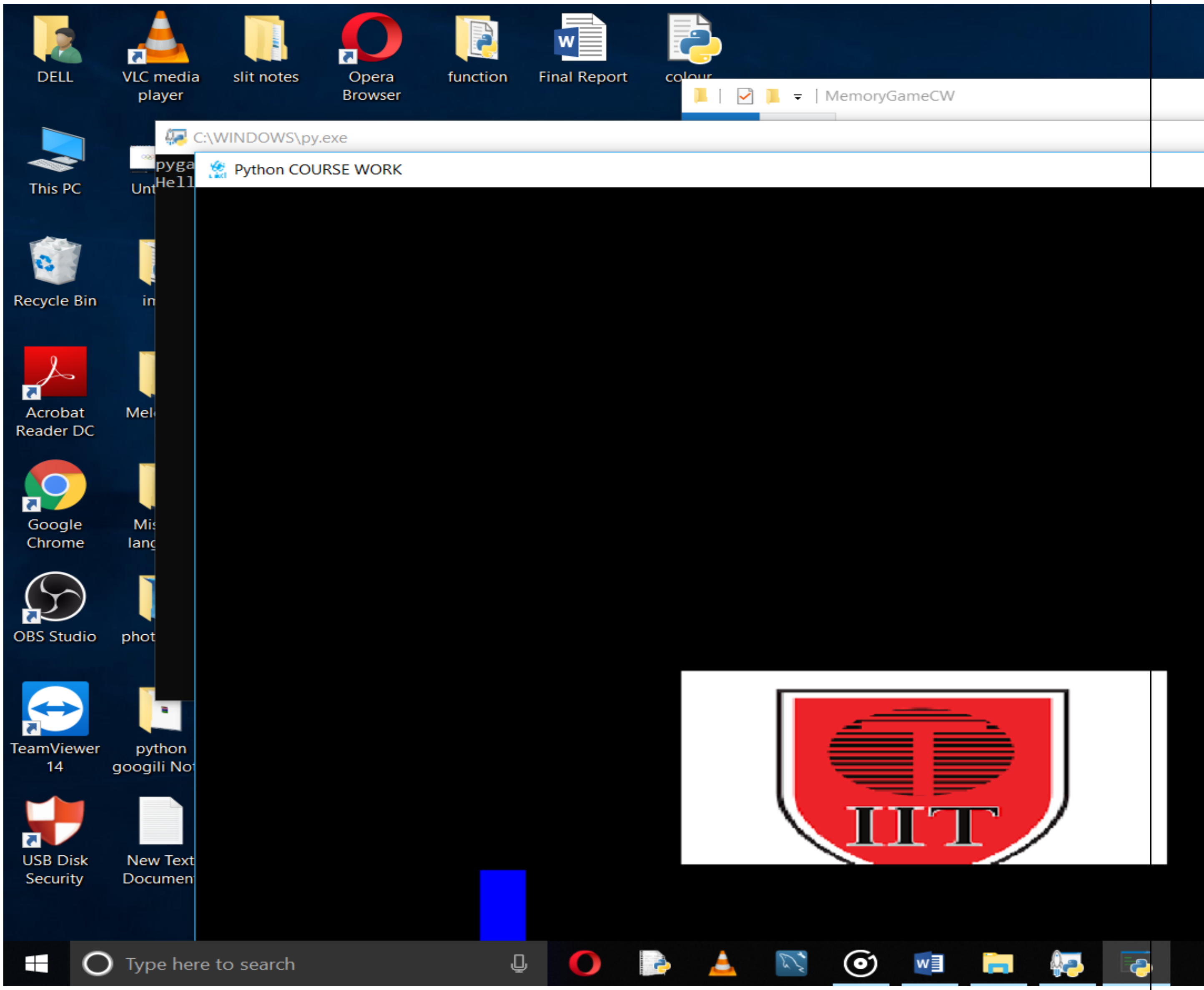
File Edit Format Run Options Window Help

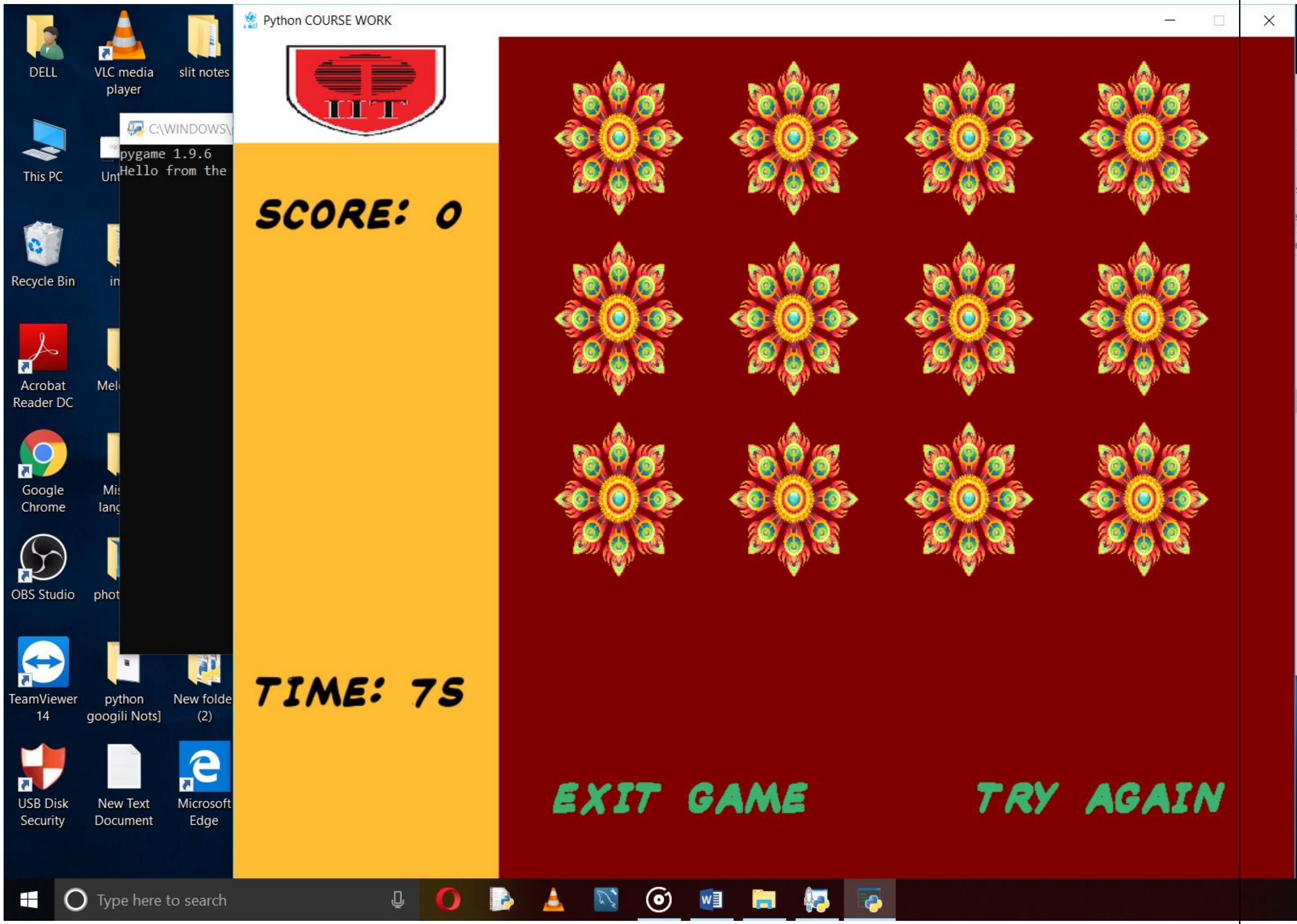
```python
import pygame
import os.path
import sys
import random
import ctypes
import pygame as pg
import random



IMAGES_NAME = ["card1", "card2", "card3", "card4", "card5", "card6"]

IMAGES_EXTENSION = ".png"

WINDOW_WIDTH = 1000
WINDOW_HEIGHT = 850

# Card data
CARD_WIDTH = 125
CARD_HEIGHT = 150

MENU_WIDTH =250
MENU_HEIGHT = WINDOW_HEIGHT

frames = []

card_backward_image = None
board = []

animations = []
current_pair = []
is_wrong = False
last_wrong_time = 0
how_many_pairs = 0

score = 0
match_time = 0
match_start_time = None
match_is_running = True
leave_button = None
restart_button = None
logo = None

# Sounds
score_sound = None
```


fainal code.py - C:\Users\DELL\Desktop\fainal\MemoryGameCW\fainal code.py (3.7.3)

File Edit Format Run Options Window Help

```
def run():
    while 1:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()
            elif event.type == pygame.MOUSEBUTTONDOWN and not is_wrong and len(animations) < 2:
                clicked_x, clicked_y = pygame.mouse.get_pos()
                click_handler(clicked_x, clicked_y)

        time_now = pygame.time.get_ticks()

        clock.tick()
        global fps
        fps = int(clock.get_fps())
        draw_menu()

        if is_wrong and time_now - last_wrong_time >= 1000:
            wrong_pair()

        if match_is_running:
            global match_time
            match_time = (time_now - match_start_time) // 1000
            draw_menu()

        for anim in animations:
            new_frame = anim.update(time_now)
            if new_frame is not None:
                pygame.draw.rect(screen, (000, 000, 000), anim.rect)
                new_frame = pygame.transform.scale(new_frame, (CARD_WIDTH, CARD_HEIGHT))
                screen.blit(new_frame, anim.rect)

            if anim.finished():
                animations.remove(anim)

        pygame.display.flip()

        if match_is_running:
            check_win()



start_match()
run()
```


DELL
VLC media player
slit notes
Opera Browser
function
Final Report
colour
This PC
Recycle Bin
Acrobat Reader DC
Google Chrome
OBS Studio
TeamViewer 14
python googili Not
USB Disk Security
New Text Documen
MemoryGameCW
C:\WINDOWS\py.exe
Python COURSE WORK
IIT
Type here to search



```
pygame 1.9.6
Hello from the
```

<u>CONCLUSION</u>

I have successfully finished my course work…………..so I thank my lectures who support me for this project……and specially thanks to google….

Thank you all