Python Project Report

UNO Card Game: Who is Lucky?

01286121 Computer Programming

Software Engineering Program,

Department of Computer Engineering,

School of Engineering, KMITL

By

65011693 Soe Moe Htet

Introduction

Uno Card Game: Who is lucky? is a small GUI-based game where user can interact with cards to find out the card that the program will randomly choose. This can be used when people want to choose which person to do something. This is similar to a lottery game or rock-paper-scissor game that decides which person goes first or goes last.

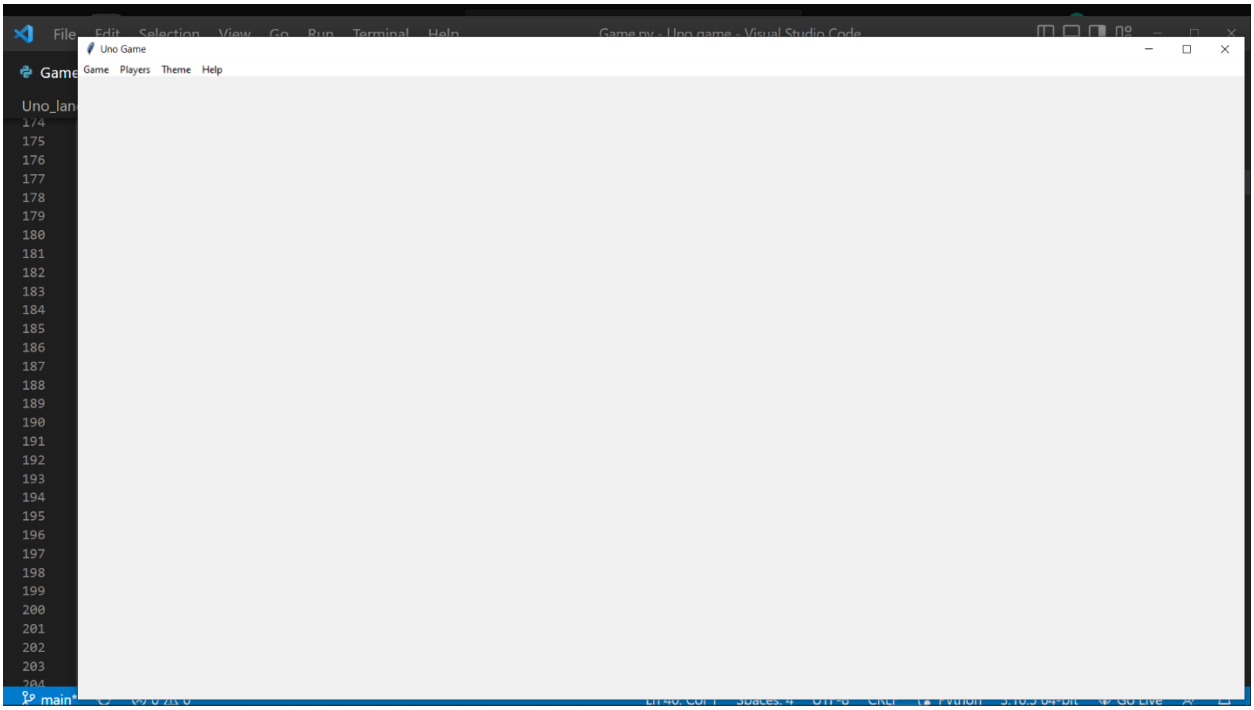
Motivation

My original project idea is to make a translator app like Google translator in our smart phones. However, such project is impossible without the use of a built-in library called google trans API which uses Natural Language Processing related deep learning technology which will allow the program to interact with the servers and translate the text that the program inputs. I transformed my idea into a simpler project, i.e., a small GUI game which will allow me to learn how to create a game with python.

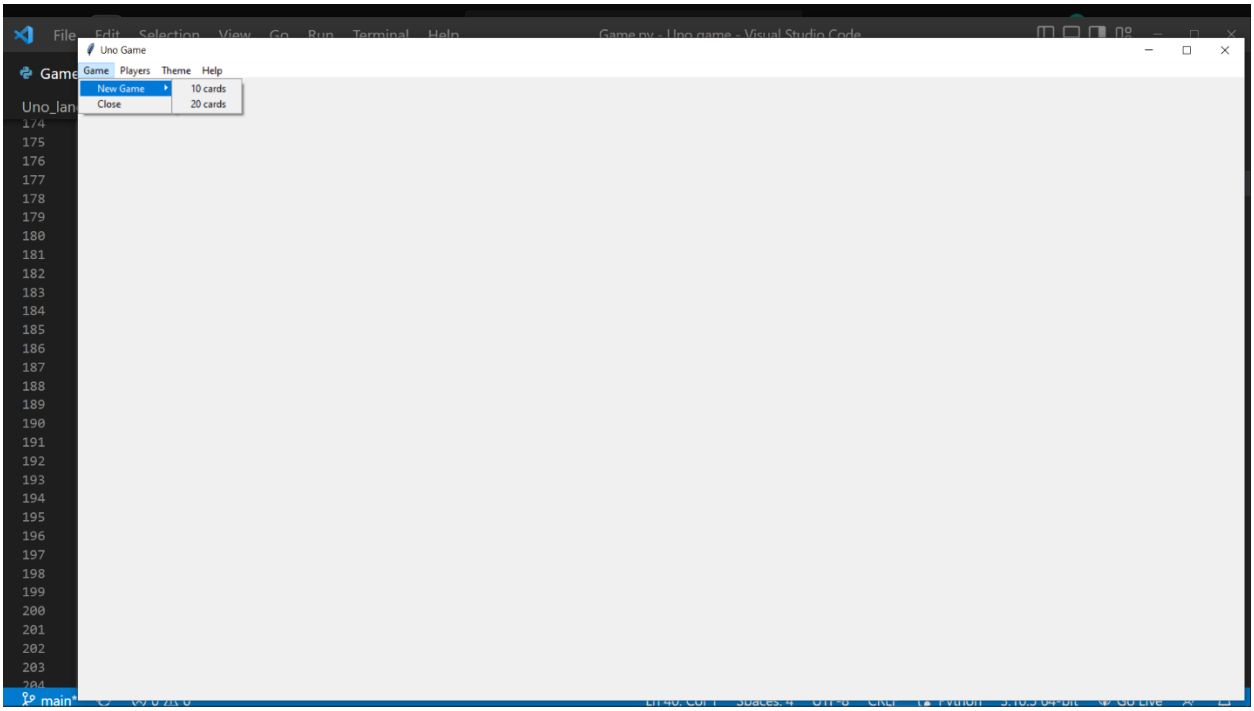
I believe that creating projects that are fun to play or interact more efficiently is better for me in learning phase. This makes me enjoy my learning process and is the main reason why I choose this project idea.

Features

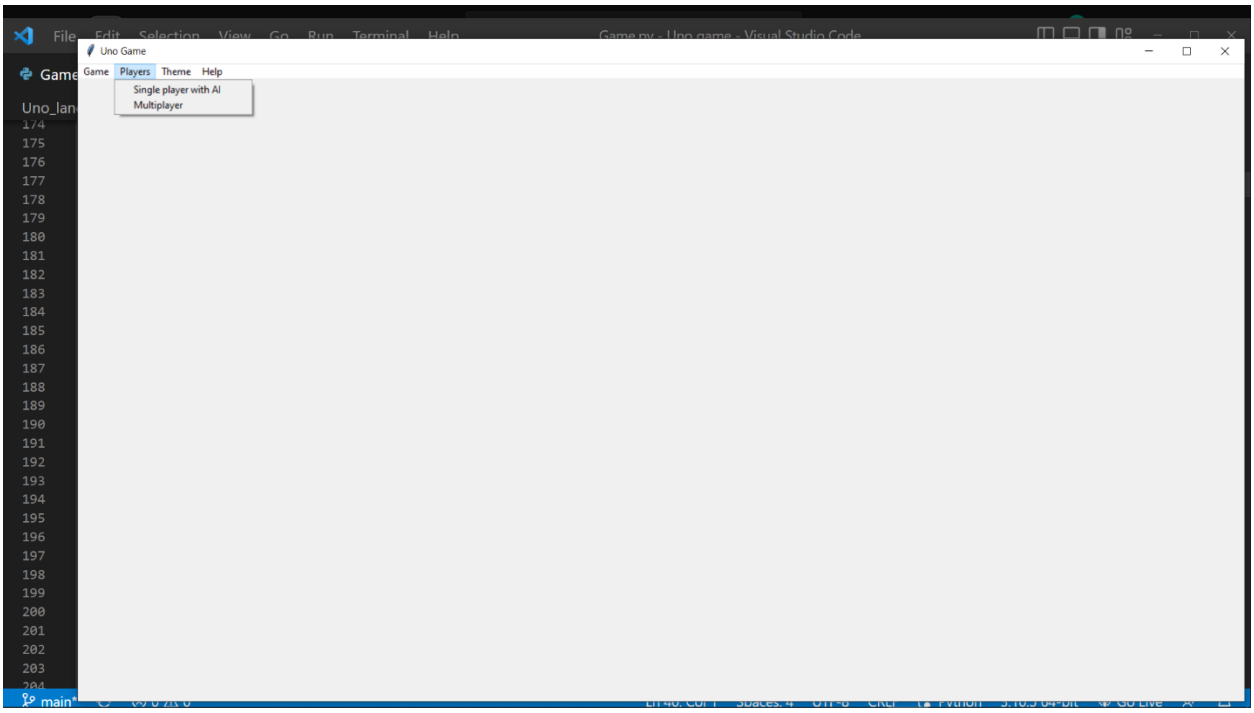
1. First Loading Page



2. What Each Menu does



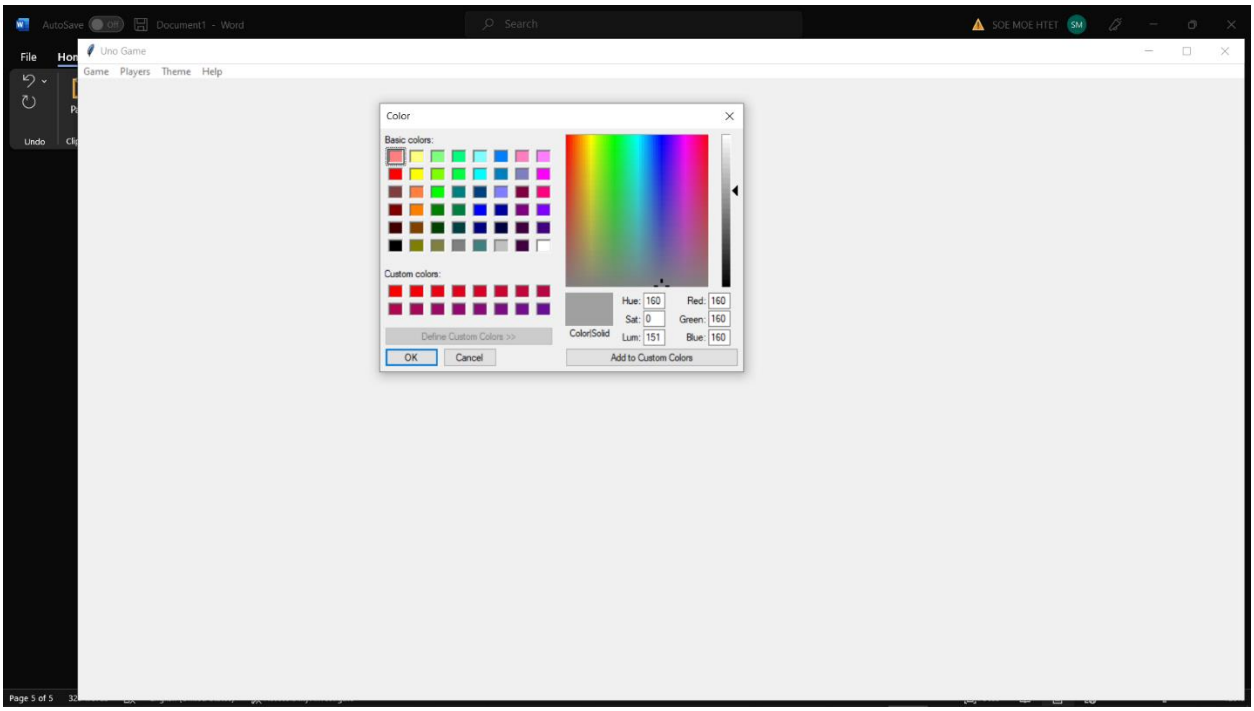
In this menu, there are two options, i.e., “New Game” with a sub-menu which lets you choose 10 cards or 20 cards to play with, and “Close“ that allows to exit the program



In this Players menu, there are two modes, “Single player with AI”, a default mode in which you can play with an AI and “Multiplayer”, in which you can play alone or with others.

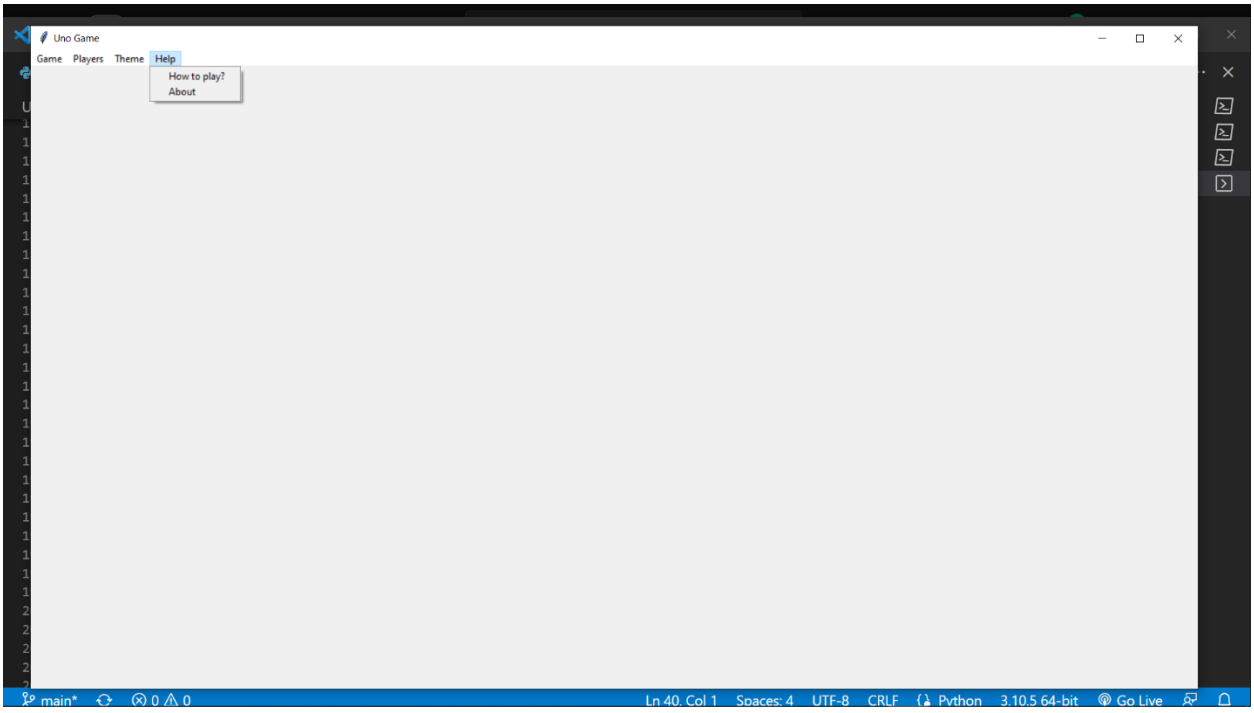
Users can choose any mode anytime, during the game or before starting the game.

This “Background Color” will allow you to change the whole window into any type of color, based on your choice.

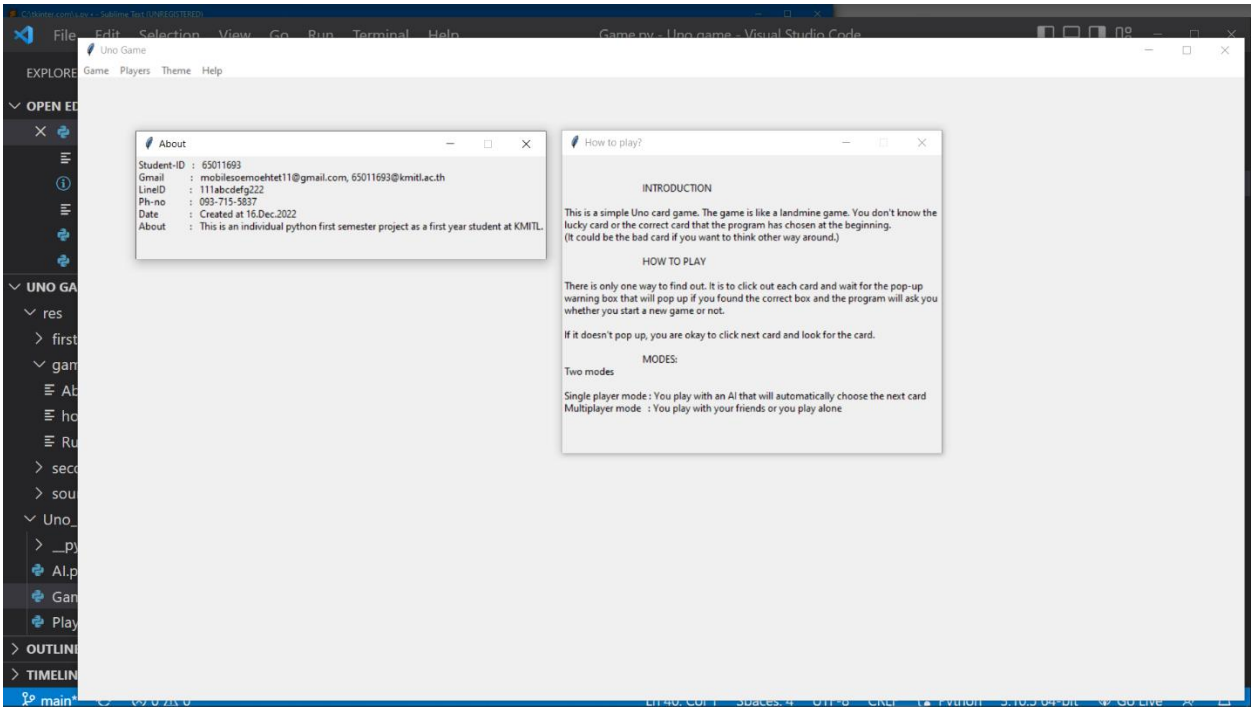

AutoSave
Document1 - Word
Search
SOE MOE HTET
File
Undo
Uno Game
Game
Players
Theme
Help
Color
Basic colors:
Custom colors:
Define Custom Colors >>
OK
Cancel
Hue: 160
Sat: 0
Lum: 151
Red: 160
Green: 160
Blue: 160
Color|Solid
Add to Custom Colors
Page 5 of 5


AutoSave
Document1 - Word
Search
SOE MOE HTET
File
Undo
Uno Game
Game
Players
Theme
Help
Page 5 of 5

The window will change its color based on the color you picked in the color picker pop-up window.

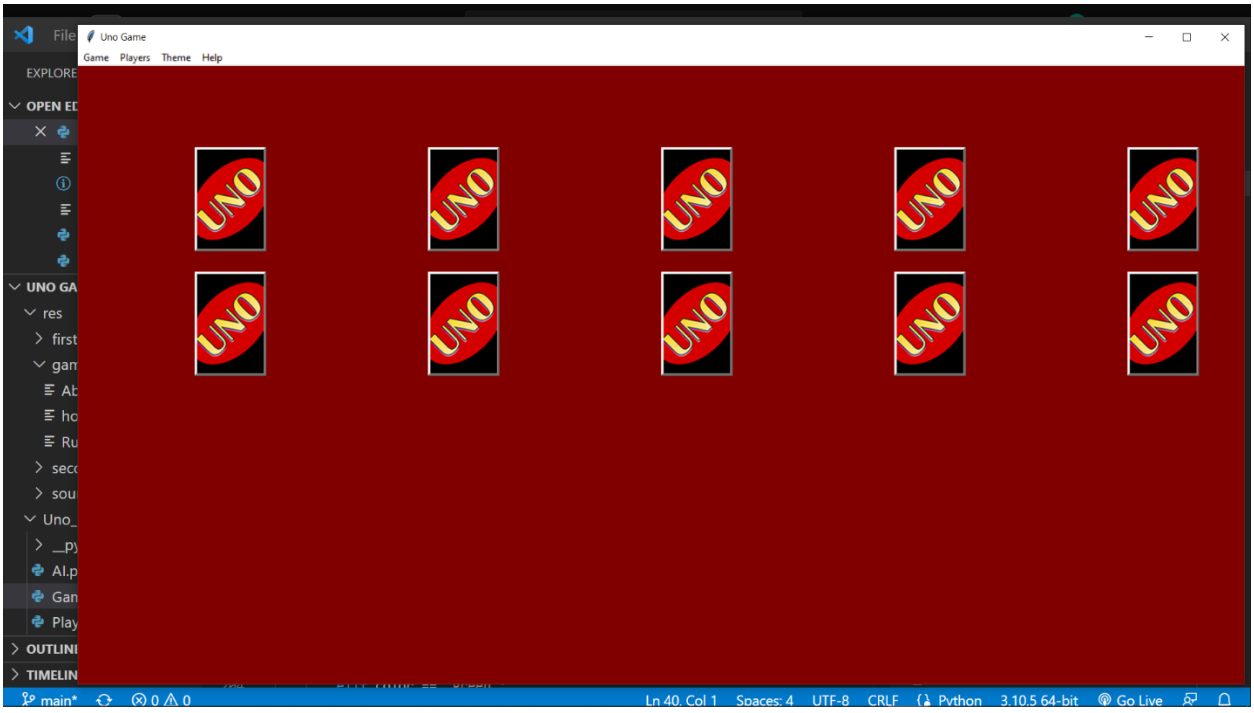


You can see a short introduction about how to play and my personal information in this menu.

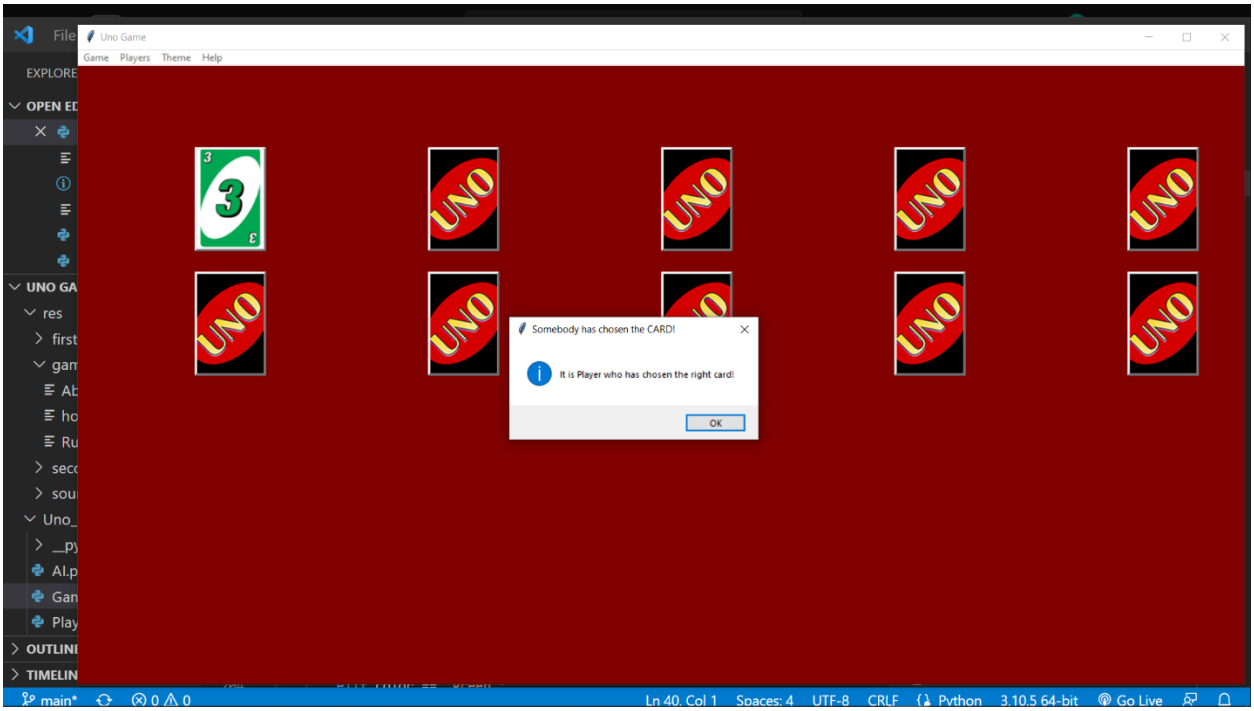


3. Playing the game

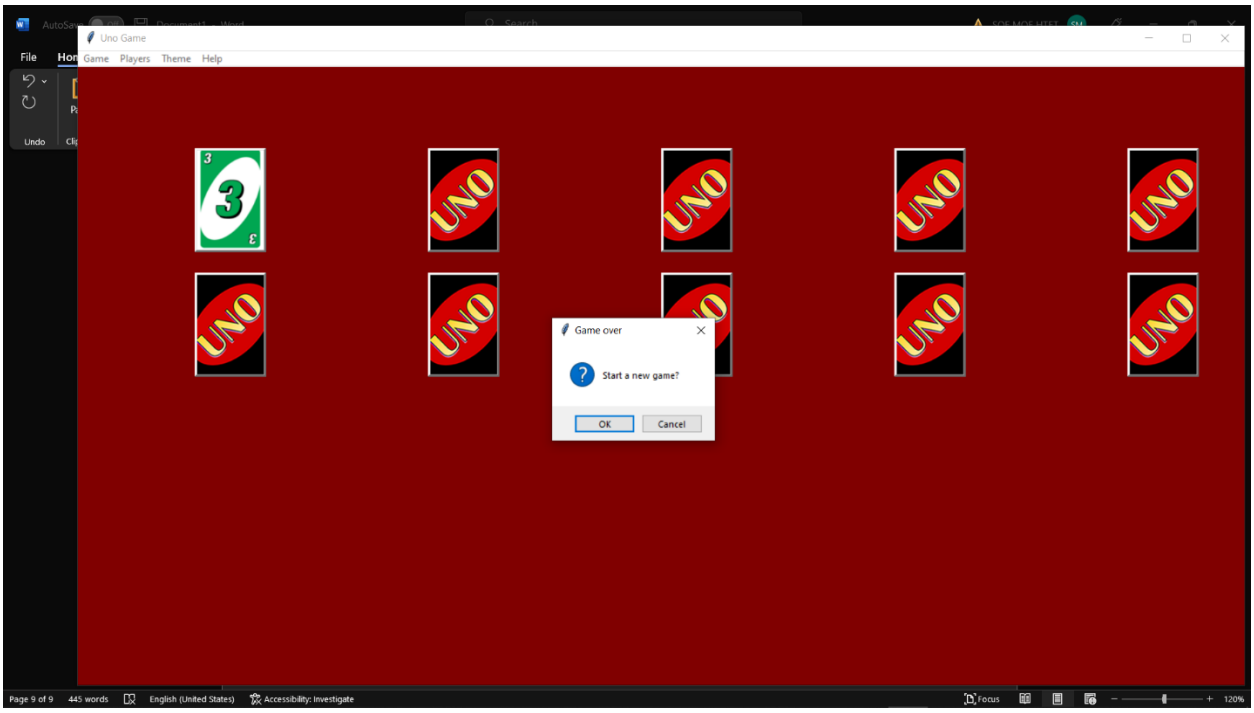
I chose a new game with 10 cards and set the background color to brown.



By default, it is a single player with AI mode, so whenever you clicked a card, an AI will automatically choose next card.

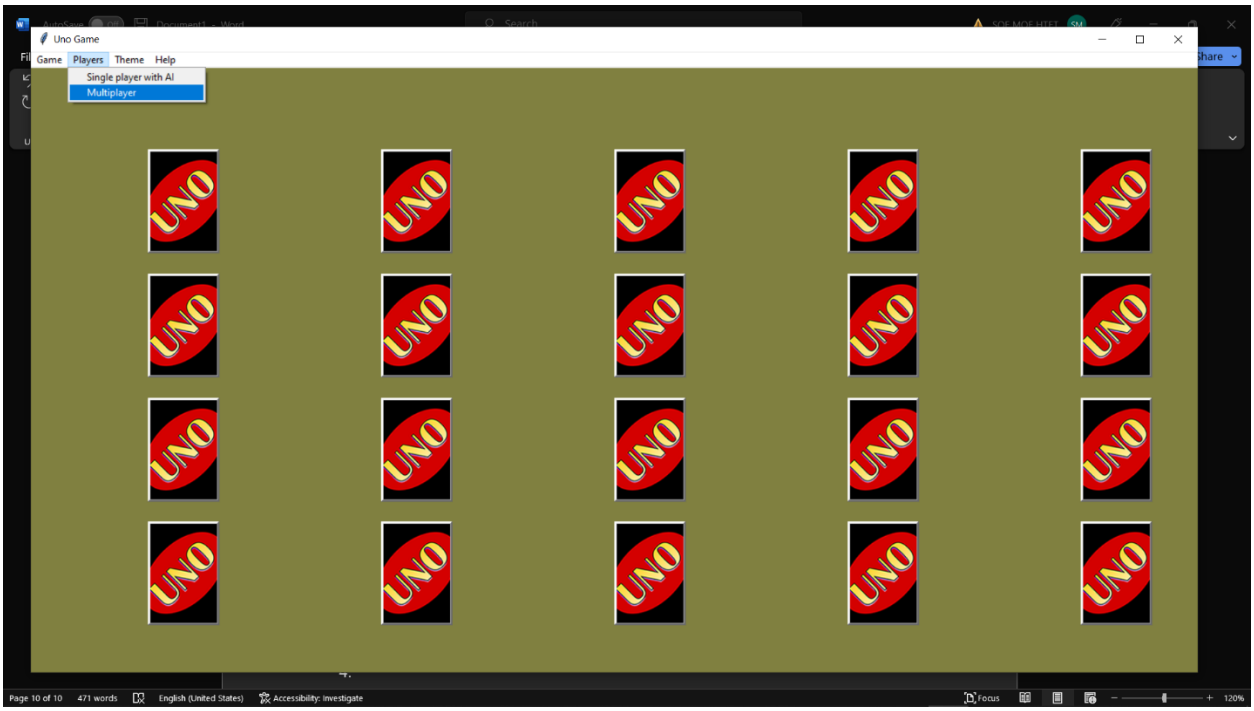


I was lucky this time. The first one is the card the program has chosen, and I clicked it so, it pops out a message box to let me know that I won the game and when I clicked ok,

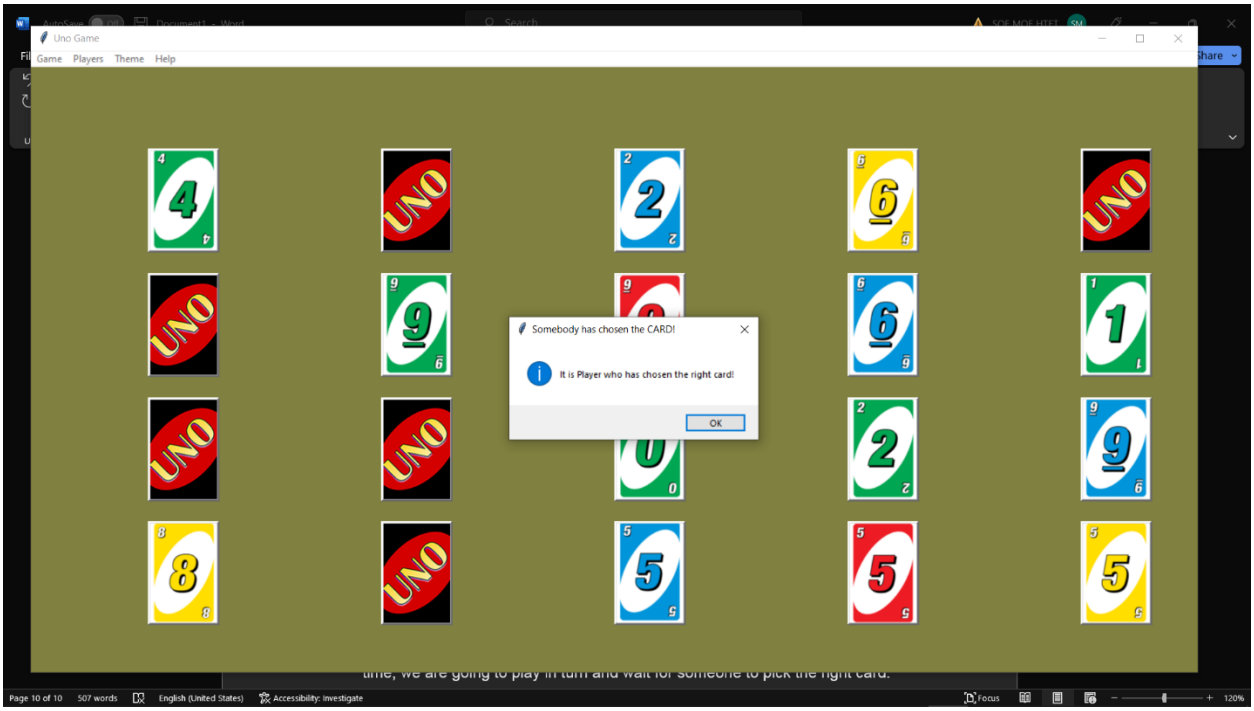


It will ask me to start a new game or close the program. If I clicked OK again,

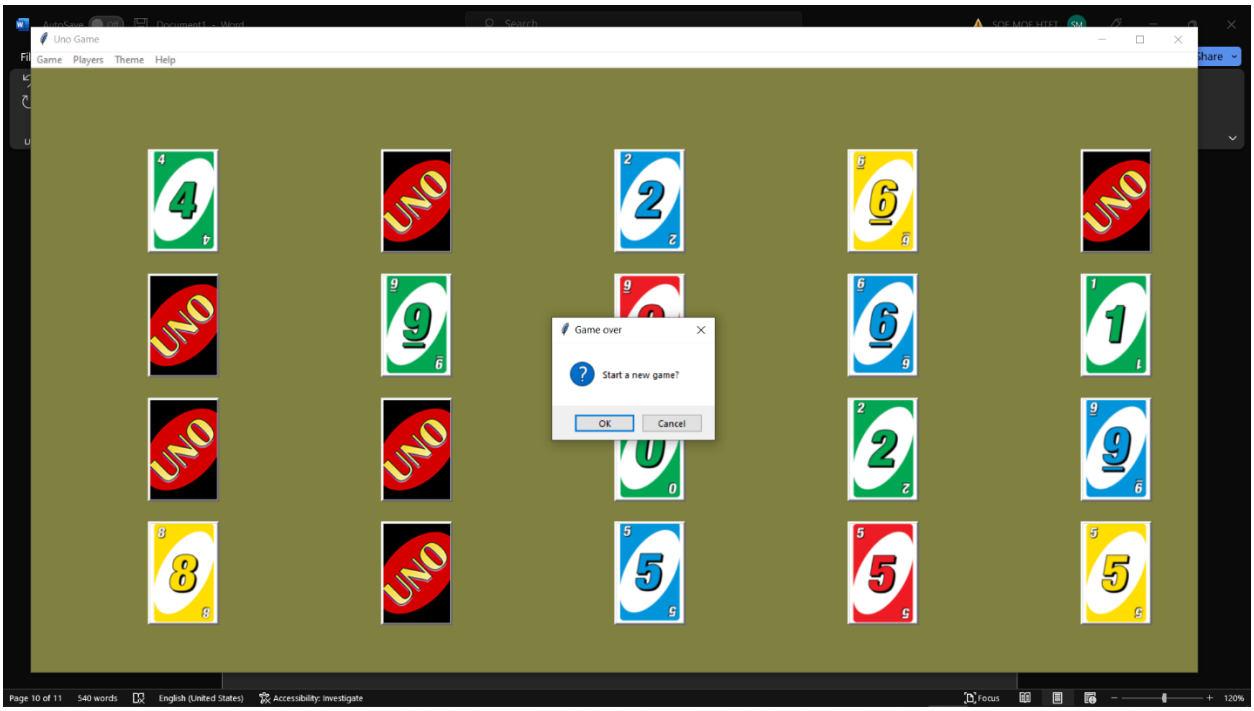
It will get back to its original state.



This time, I chose to play with my friends or alone. If I play with my friends in real time, we are going to play in turn and wait for someone to pick the right card.



After a while playing the game, if someone chooses the right card, it will show as the same as before. And the program will ask you to create a new game or not.



4. Source codes

   If you want to look at the source codes, you can use this link and this link will lead to my GitHub repository and it has every source code file.

   GitHub link- https://github.com/Smh1111/Uno_Game_Python

   If you do not wish to see the source code in the above way, please skip to next part.

Resources

Images that I have used in this project.

https://www.pngfind.com/mpng/iJbmTbR_uno-cards-png-cards-are-in-an-uno/

I gave full credit to the owner of these images and these images are under license meant for personal use only.

Sounds that I have used in this project.

https://mixkit.co/free-sound-effects/click/

These soundtracks are under simple commercial license for any project.

https://elements.envato.com/license-terms

Tools

Photoshop

I cropped the images and obtained the required cards by using adobe photoshop

Visual Studio Code

Raw Codes

Game.py

```
import os
import random
import time
from tkinter import Button, Label, PhotoImage, Toplevel, messagebox
import tkinter as tk

from playsound import playsound
from tkinter import colorchooser

from Player import Player
from AI import AI
import sys


window = tk.Tk()
window.title("Uno Game")
```

```
WIDTH = 1436
HEIGHT = 764
NUMBER_OF_CARDS = 10

SINGLE_PLAYER_With_AI = False

AI_turn = True
Player_turn = True

x = WIDTH / 10
y = 100

COLORS = ["red", "blue", "green", "yellow"]

FIRST_DECK_Cards = []


TOTAL_CARDS = []

BACKPILE = []

ANS_LIST = []

IMAGE_LIST = []

CHOSEN_CARD = ""

REMAINING_COLORS_NUMBERS_LIST = []

def window_setup():



    window.geometry(f"{WIDTH}x{HEIGHT}")

class FirstDeck():
    def __init__(self, number_of_cards):
        global CHOSEN_CARD, ANS_LIST, REMAINING_COLORS_NUMBERS_LIST

        self.number_of_cards = number_of_cards

        self.previous_card = 0

        self.numberList = []
        self.colorList = []

        self.red_imageList = {}
        self.blue_imageList = {}
        self.green_imageList = {}
        self.yellow_imageList = {}

        self.red_labelList = []
        self.blue_labelList = []
```

```
        self.green_labelList = []
        self.yellow_labelList = []

        self.check_dict = self.cards() # This will be a dictionary.

        ANS_LIST = get_swap_dict(self.check_dict)


        random_number = random.randint(0, 9)
        CHOSEN_CARD = ANS_LIST[random_number]

        REMAINING_COLORS_NUMBERS_LIST = list(ANS_LIST.values())


    def get_check_dict(self):
        return self.check_dict

    def card_images(self):
        numberList = [i for i in range(10)]

        try:
            for i in numberList:
                self.red_imageList[i] = PhotoImage(file = f"res/second
version/red/red_{str(numberList[i])}.png")
                #self.red_imageList.append()

                self.blue_imageList[i] = PhotoImage(file = f"res/second
version/blue/blue_{str(numberList[i])}.png")
                #self.blue_imageList.append()

                self.green_imageList[i] = PhotoImage(file = f"res/second
version/green/green_{str(numberList[i])}.png")
                #self.green_imageList.append()

                self.yellow_imageList[i] = PhotoImage(file = f"res/second
version/yellow/yellow_{str(numberList[i])}.png")
                #self.yellow_imageList.append()

        except FileNotFoundError:
            raise FileNotFoundError


    def card_buttons(self):
        self.card_images()
        for i in range(len(self.red_imageList)): # i = key
            self.red_labelList.append(
                Button(window, width= 80, height = 120, border= 3, image =
self.red_imageList[i], command= lambda m = f"{str(i)}": [self.click(f"red {m}",
f"{self.red_labelList[i]}")]))

        for i in range(len(self.blue_imageList)):
            self.blue_labelList.append(Button(window, width= 80, height = 120, border= 3,
image = self.blue_imageList[i], command= lambda m = f"{str(i)}": [self.click(f"blue {m}",
f"{self.blue_labelList[i]}"),]))
```

```
        for i in range(len(self.green_imageList)):
            self.green_labelList.append(Button(window, width= 80, height = 120, border= 3,
image = self.green_imageList[i], command= lambda m = f"{str(i)}": [self.click(f"green {m}",
f"{self.green_labelList[i]}")]))

        for i in range(len(self.yellow_imageList)):
            self.yellow_labelList.append(Button(window, width= 80, height = 120, border= 3,
image = self.yellow_imageList[i], command= lambda m = f"{str(i)}": [self.click(f"yellow {m}",
f"{self.yellow_labelList[i]}"),]))

    def card_setup(self, owned_colorList = 0, owned_numberList = 0, FIRST_DECK = []):

        if owned_colorList != 0 and owned_numberList != 0:
            #print("..............card_setup..............")

            self.colorList = owned_colorList
            self.numberList = owned_numberList

        #print("============================")
        #print("self.colorList:\n", self.colorList)
        #print("self.numberList:\n", self.numberList)

        i = 0
        while i != len(self.colorList):
            color = self.colorList[i]
            number = self.numberList[i]


            if color == "red":
                card = self.red_labelList[int(number)]

            elif color == "blue":
                card = self.blue_labelList[int(number)]

            elif color == "green":
                card = self.green_labelList[int(number)]

            elif color == "yellow":
                card = self.yellow_labelList[int(number)]

            FIRST_DECK.append(card)

            i += 1

    def cards(self):

        i = 0

        check_dict = {}
        temp = False

        while(i != self.number_of_cards): # 10 or 20 cards
            number = random.randint(0, 9)
```

```
            color = random.randint(0,3)

            self.numberList.append(number)
            self.colorList.append(COLORS[color])


            color = self.colorList[i]
            number = self.numberList[i]
            number_str = str(self.numberList[i])
            key = color + f" {number_str}"
            if key in check_dict:
                self.colorList.pop()
                self.numberList.pop()
                #print(Exception("Sorry, No duplicates"))
                continue


            check_dict[key] = i

            i += 1

        # print("numberList : ", self.numberList)
        # print("colorList : ", self.colorList)

        return check_dict

    def find_the_card(self, title):
        color, number = title.split(" ")

        if color == "red":
            card = self.red_labelList[int(number)]
            image = self.red_imageList[int(number)]

        elif color == "blue":
            card = self.blue_labelList[int(number)]
            image = self.blue_imageList[int(number)]

        elif color == "green":
            card = self.green_labelList[int(number)]
            image = self.green_imageList[int(number)]

        elif color == "yellow":
            card = self.yellow_labelList[int(number)]
            image = self.yellow_imageList[int(number)]

        return card, image

    def click(self, title, img_address = ""):
        global REMAINING_INDEXLIST, REMAINING_COLORS_NUMBERS_LIST, ANS_LIST, Player_turn,
AI_turn, SINGLE_PLAYER_With_AI

        if Player_turn == True:
            playsound('res/sounds/mouse_click_close.wav', False)
```

```
        card, image = self.find_the_card(title)

        REMAINING_COLORS_NUMBERS_LIST.remove(title)



        # print("Remaining colors numebr list: ", REMAINING_COLORS_NUMBERS_LIST)
        try:
            card.config(image = image)
        except Exception as e:
            print(e)

        if CHOSEN_CARD == title and AI_turn == True:
            #card.destroy()
            seconddeck = SecondDeck()
            seconddeck.popup("Player")


        if SINGLE_PLAYER_With_AI == True:

            if AI_turn == True:
                self.AI_play()


            try:
                card.config(image = image)
            except Exception as e:
                print(e)

            if CHOSEN_CARD == title and AI_turn == False:
                #card.destroy()
                seconddeck = SecondDeck()
                seconddeck.popup("AI")

            AI_turn = True
        Player_turn = True

    def AI_play(self):
        global AI_turn, Player_turn, REMAINING_COLORS_NUMBERS_LIST

        ai = AI(REMAINING_COLORS_NUMBERS_LIST)

        Colors_numbers, AI_turn = ai.play_random_cards()
        card, image = self.find_the_card(Colors_numbers)



        Player_turn = False
        try:
            card.invoke()
        except Exception as e:
            print(e)
        # print(AI_turn)
```

```
class SecondDeck(FirstDeck):
    def __init__(self):
        super().__init__(NUMBER_OF_CARDS)

    def popup(self, who_wins = ""):

        # pop = Toplevel(window)
        # pop.title("Confirmation")
        # pop.geometry("250x250")
        # pop.config(bg="green3")
        # Create a Label Text
        # label = Label(pop, text="You have chosen the lucky card!", bg="white", fg="black",
font=('Aerial', 12))
        # label.place(x=10, y = 10)
        messagebox.showinfo("Somebody has chosen the CARD!", f"It is {who_wins} who has chosen
the right card!")

        on_closing()

#======================#======================#======================
# Helper Functions
def get_swap_dict(d):
    return {v: k for k, v in d.items()}

def on_closing():
    if messagebox.askokcancel("Game over", "Start a new game?"):
        window.destroy()
        os.startfile("D:/KMITL University/1st year 1st sem/Computer programming (python)/Uno
game/Uno_landmine/Game.py")

    else:
        window.destroy()


#======================#======================#======================

#======================#======================#======================
# Game controls
def game_setup():
    global x,y, FIRST_DECK_Cards, WIDTH, HEIGHT, NUMBER_OF_CARDS

    if NUMBER_OF_CARDS == 10:
        rows = 2
        columns = 5

    elif NUMBER_OF_CARDS == 20:
        rows = 4
        columns = 5

    x = WIDTH / 10
    y = 100

    i = 0
```

```
    k = 0
    while i != rows:
        j = 0
        while j != columns:

            FIRST_DECK_Cards[k].place(x = x, y = y)
            x += WIDTH / 5

            j += 1

            k += 1
        x = WIDTH / 10
        y += HEIGHT / 5
        i += 1

    back_image = PhotoImage(file = f"res/second version/Back/back1.png")

    for i in FIRST_DECK_Cards:
        i.config(image = back_image)


    window.mainloop()

def top_bar():

    top = tk.Menu(window)
    window.config(menu=top)

    jeu = tk.Menu(top, tearoff=False)
    top.add_cascade(label='Game', menu=jeu)
    submenu = tk.Menu(jeu, tearoff=False)
    jeu.add_cascade(label='New Game', menu=submenu)

    submenu.add_command(
        label='10 cards',
        command= lambda m = f"": [set_number(number_of_cards = 10)]
    )
    submenu.add_command(
        label='20 cards',
        command= lambda m = f"": [set_number(number_of_cards = 20)]
    )

    jeu.add_command(label='Close', command=sys.exit)


    players_menu = tk.Menu(top, tearoff=False)
    top.add_cascade(label='Players', menu=players_menu)


    players_menu.add_cascade(
        label='Single player with AI',
        command = set_singleplayer_mode()
    )
```

```
    players_menu.add_command(
        label='Multiplayer',
        command = set_multiplayer_mode
    )
    color_menu = tk.Menu(top, tearoff=False)

    top.add_cascade(label="Theme", menu=color_menu)

    color_menu.add_command(label="Bacground Color", command=color_choser)

    help_menu = tk.Menu(top, tearoff=False)
    top.add_cascade(label='Help', menu=help_menu)
    help_menu.add_command(label='How to play?', command = print_rules)
    help_menu.add_command(label='About', command = about)

def playground_setup() -> None:
    time.sleep(0.25)
    window_setup()

    top_bar()

    global x,y, FIRST_DECK_Cards, CHOSEN_CARD

    first_Deck = FirstDeck(NUMBER_OF_CARDS)
    first_Deck.card_buttons()

    first_Deck.card_setup(FIRST_DECK = FIRST_DECK_Cards)

    first_Deck_Titles = first_Deck.get_check_dict()


    game_setup()

def set_number(number_of_cards):

    global NUMBER_OF_CARDS
    NUMBER_OF_CARDS = number_of_cards
    # print(NUMBER_OF_CARDS)

    playground_setup()

def set_singleplayer_mode() -> None:
    global SINGLE_PLAYER_With_AI

    SINGLE_PLAYER_With_AI = True

def set_multiplayer_mode() -> None:
    global SINGLE_PLAYER_With_AI

    SINGLE_PLAYER_With_AI = False

def print_rules() -> None:
    """
    Display rules and gameplay.
```

```
    Load a text files 'Rules.txt'.
    Open a second window.
    Write the content of the text document.
    Returns
    -------
    None.
    """
    rule_window = tk.Toplevel()
    rule_window.resizable(False, False)
    rule_window.title("How to play?")


    with open(f"res/game_rules/Rules.txt") as f:
        gameRules = f.read()
    lab_Rule = tk.Label(rule_window, text=gameRules,
                        fg="black", anchor="e", justify=tk.LEFT)
    lab_Rule.pack(side=tk.TOP)
    rule_window.mainloop()

def about() -> None:
    """
    Display student id.
    Load the text document 'About.txt'.
    Open a secondary window.
    Write the content of the text document.
    Returns
    -------
    None.
    """
    about_window = tk.Toplevel()
    about_window.resizable(False, False)
    about_window.title("About")
    with open(f"res/game_rules/About.txt") as f:
        about = f.read()
    lbl_about = tk.Label(about_window, text=about,
                         fg="black", anchor="e", justify=tk.LEFT)
    lbl_about.pack(side=tk.TOP)
    about_window.mainloop()

def color_choser() -> None:
    user_color = colorchooser.askcolor()[1]
    window.config(bg = user_color)

def restart():
        import sys
        print("argv was",sys.argv)
        print("sys.executable was", sys.executable)
        print("restart now")

        import os
        os.execv(sys.executable, ['python'] + sys.argv)


#=====================#=====================#=====================
```

```
def main():
    window_setup()

    top_bar()

    window.mainloop()


main()
```

Player.py

```
class Player:
    """A class to represent a player in the game."""
    def __init__(self):

        self.can_play = False
```

AI.py

```
from Player import Player
import random

class AI(Player):
    def __init__(self, remaining_colors_numbers =  []):
        super().__init__()

        self.remaining_colors_numbers = remaining_colors_numbers

        # print(self.remaining_colors_numbers)



    def play_random_cards(self):
        ran_card = random.choice(self.remaining_colors_numbers)

        # print("AI Play: ", ran_card)
        self.can_play = False

        return ran_card, self.can_play
```

You can access every asset and code file in this drive

https://drive.google.com/drive/folders/1RZ-6l3Xb7KeNm8M10Ihej-KER7pwTE-j

References

https://mykindofmeeple.com/games-to-decide-who-goes-first/