Python Project Report

<Memory Puzzle Learning Resource>

01286121 Computer Programming

Software Engineering Program

By

66011533 Eaint Kay Khaing Kyaw

Python Project

< Memory Puzzle Learning Resource >

Project Introduction

Memory Puzzle Learning resource is an application which includes both game space and learning platform. The application is designed to enhance the memory skills and to use as a fun learning resource for children in studying numbers especially even, odd and prime numbers. The application is designed to be intuitive, allowing users to reset the game, quit, and access learning materials effortlessly.

Features and Functionality

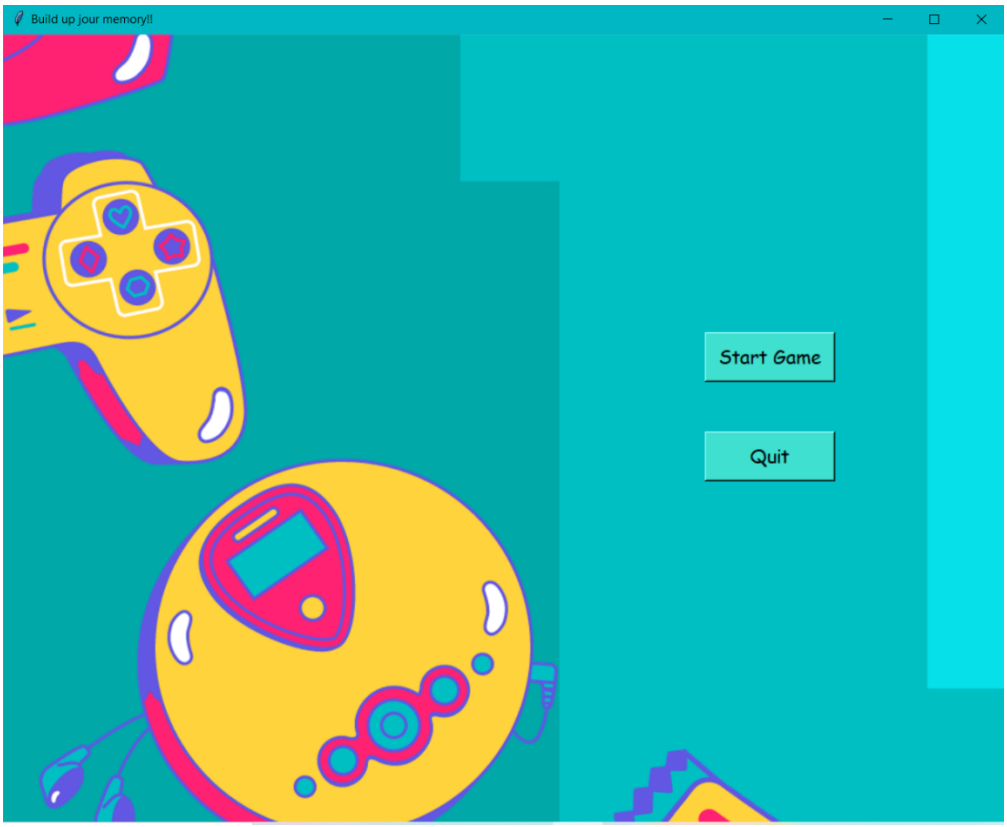
The project employs the Tkinter library to create an interactive and user-friendly graphical interface. It is structured with classes for each game mode, promoting code organization and reusability. It also includes exception handling to handle Index and attribute error. Each game includes a Learn Button that opens a new window with educational content related to the respective number category, enhancing the application's educational value.

The core logic of the game involves matching pairs of numbers by clicking on grid elements. User can access Reset Mode and Learning platform for each of those numbers. It also includes a relax mode where users can match pairs with various shapes and colors.

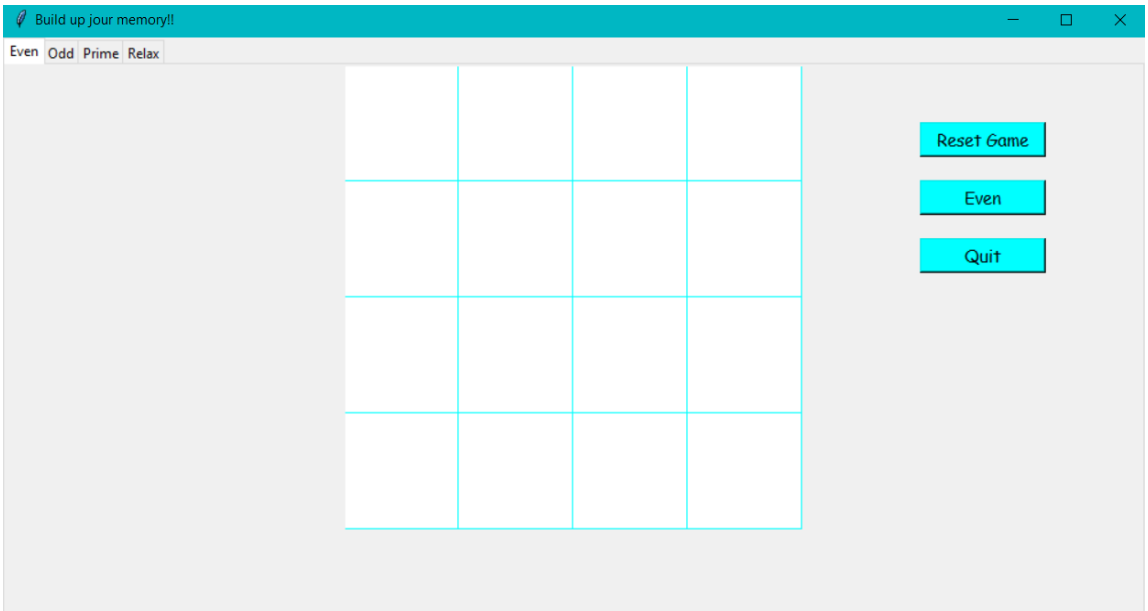
Motivation

I think explaining the concepts of the numbers to children is a challenging task. That's why I wanted to create a learning tool that is not only effective but also engaging for children. By combining learning with play, the aim is to make the process of understanding numbers a more enjoyable and accessible experience for young minds.

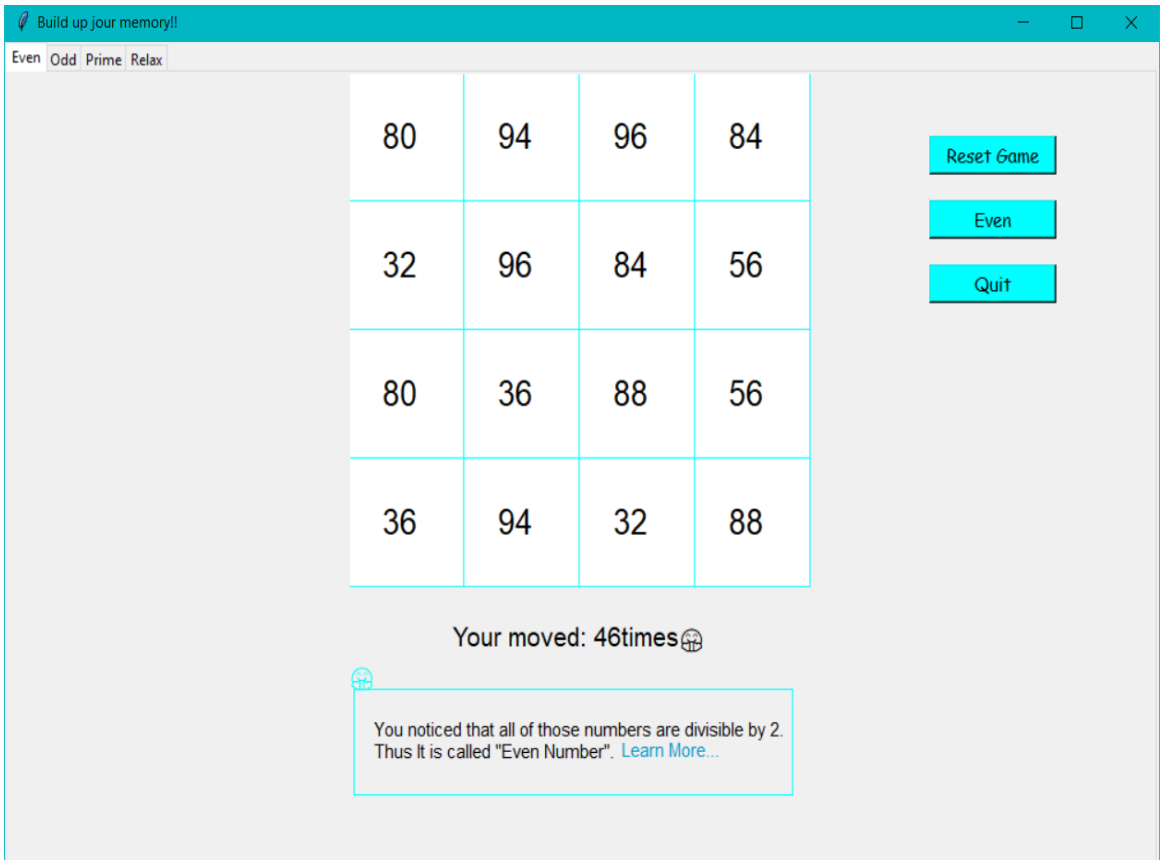
1.HomePage when running the program



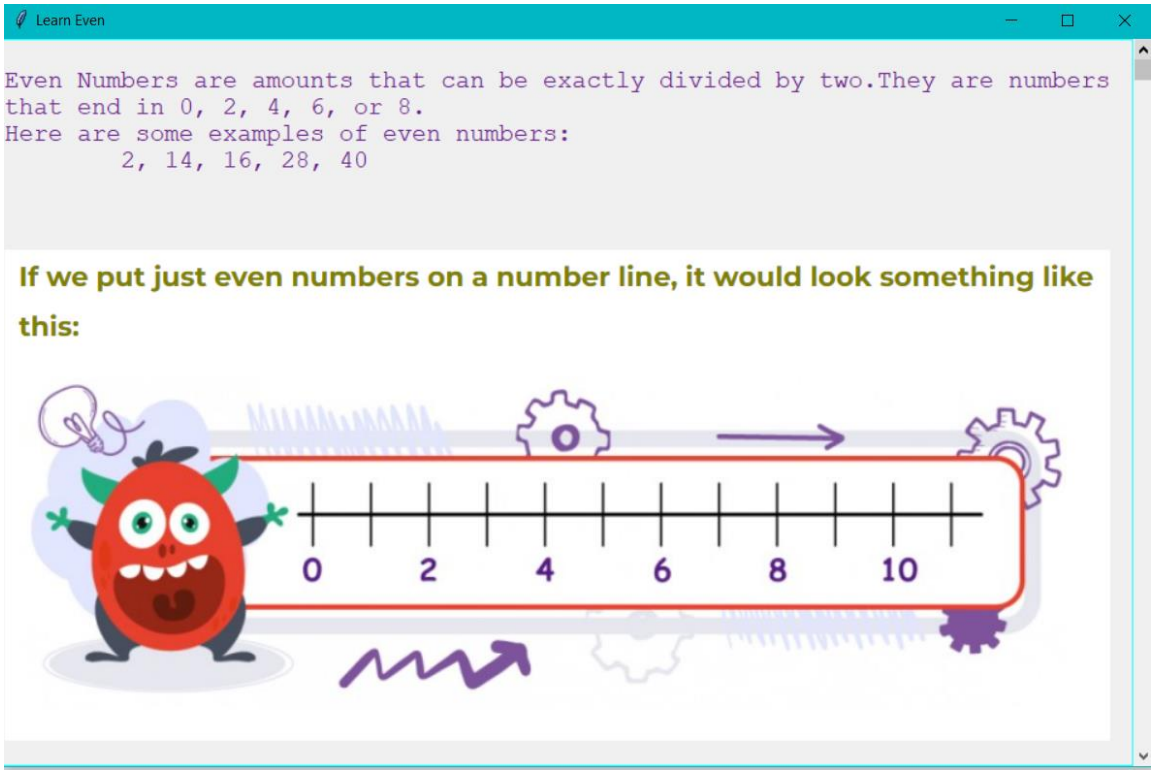
2.When clicking StartGame Button



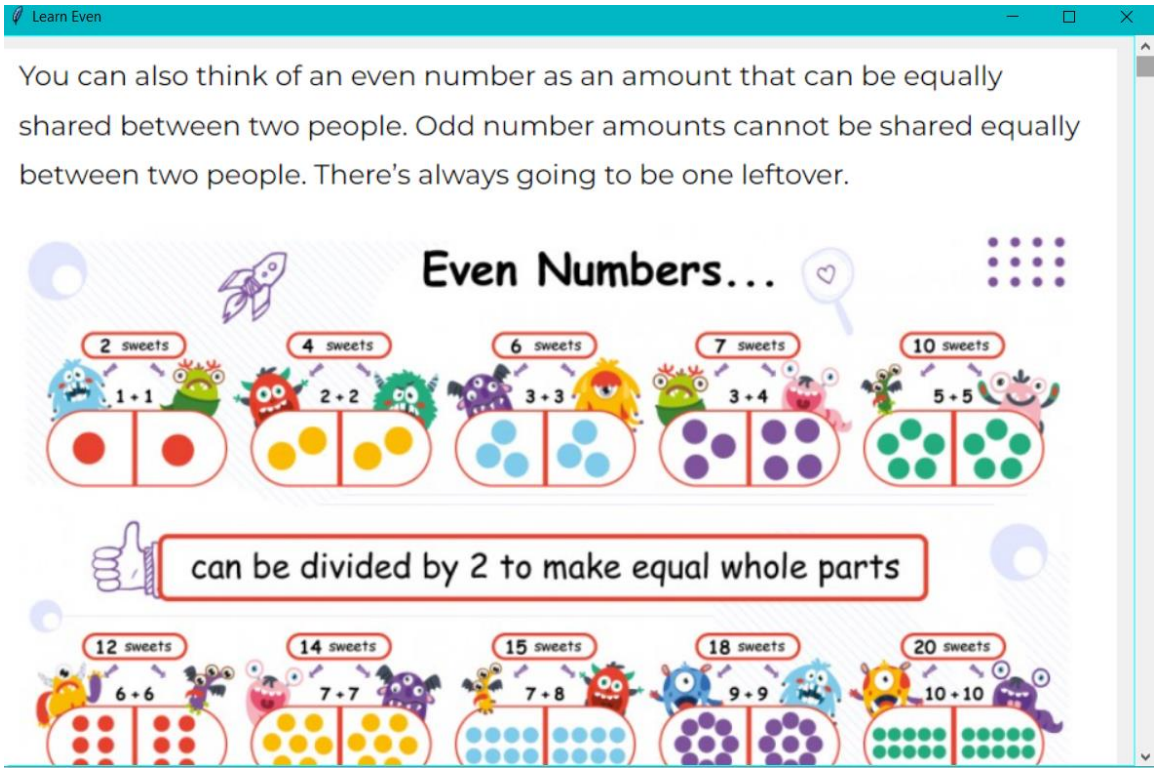
3.Even Game Mode when user flips all the piles. The text box including even number definition and LearnMore Button and the user moves track will appear after flipping all files.



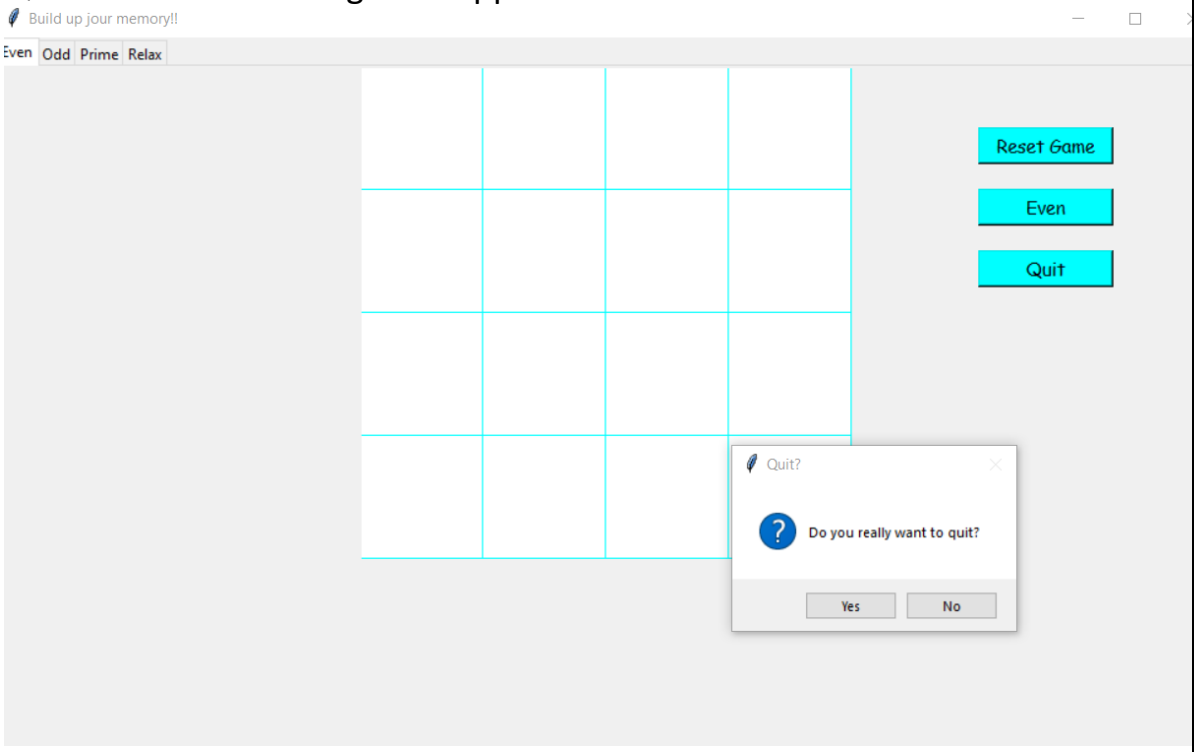
4.When clicking LearnMore and Even Button



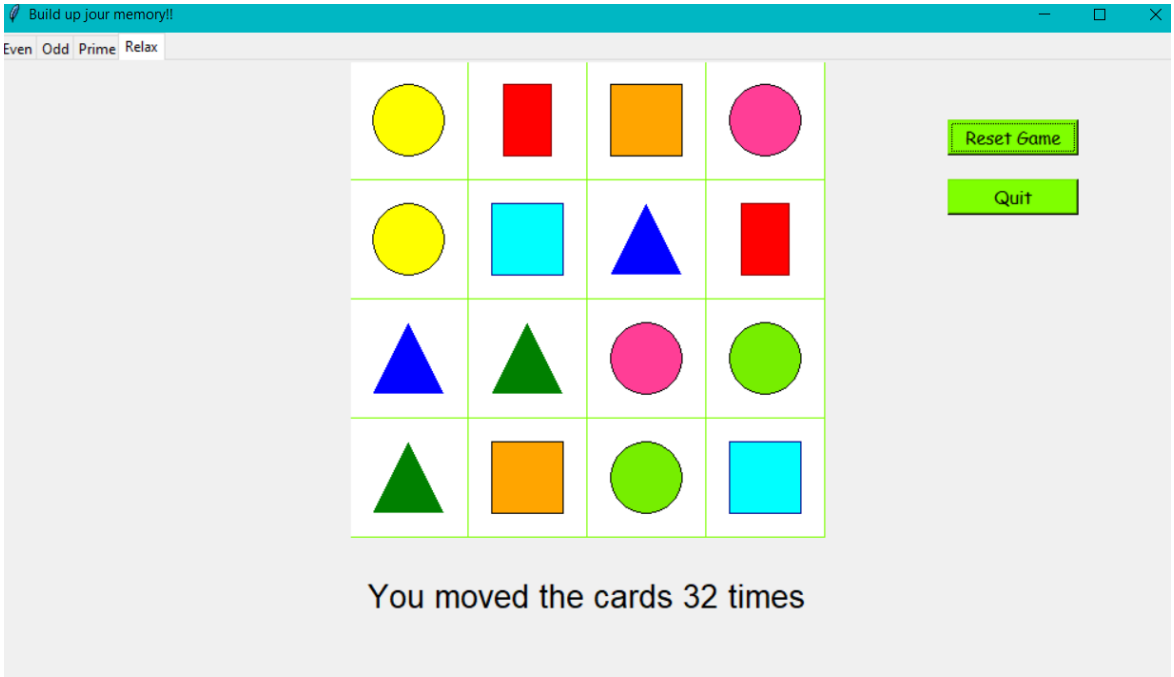
4.Even Window when Scrolling down I created text and inserted images



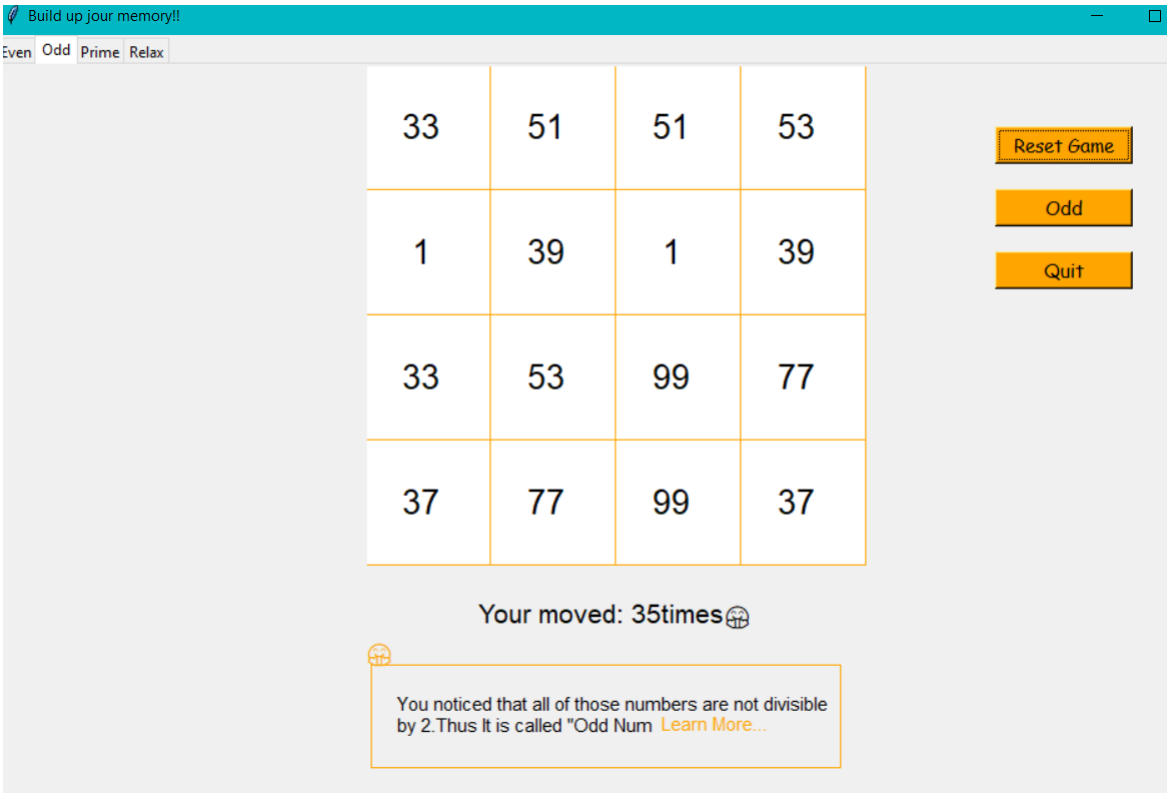
5. When clicking reset, the board will be its initial condition. When clicking Quit button the message box appears as follows.



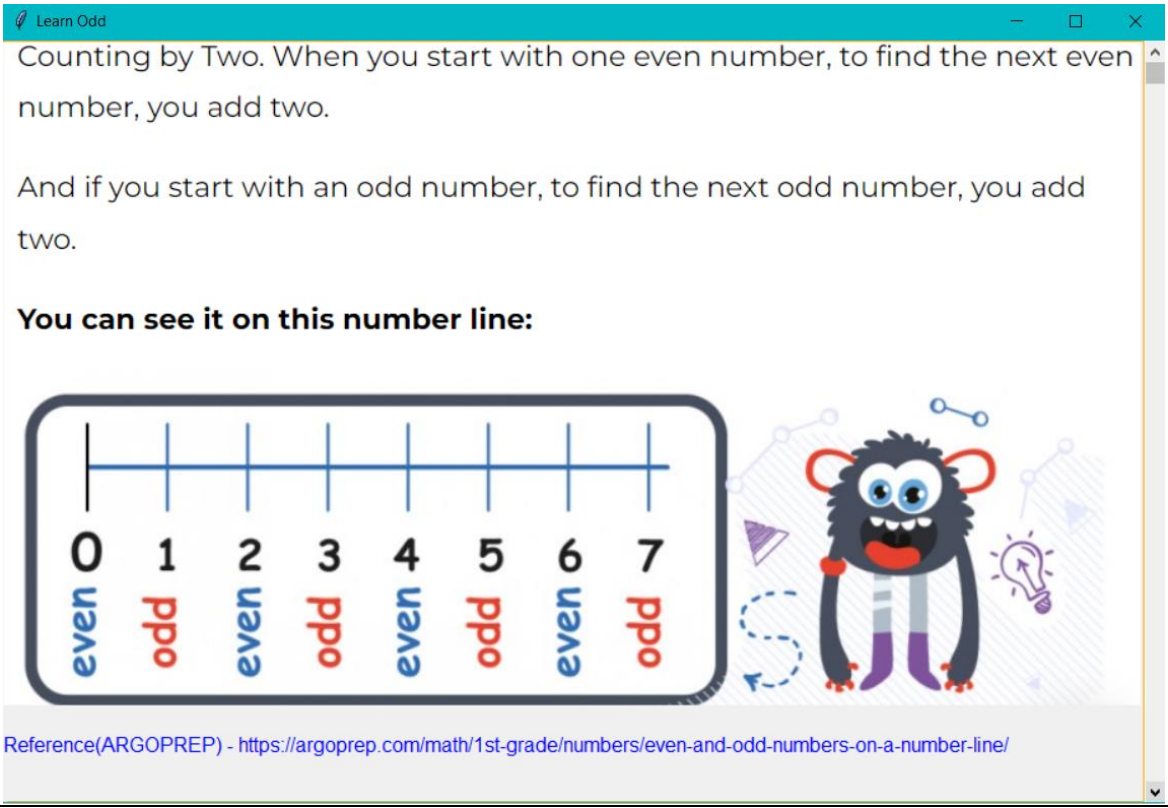
6. Relax Mode with colorful shapes



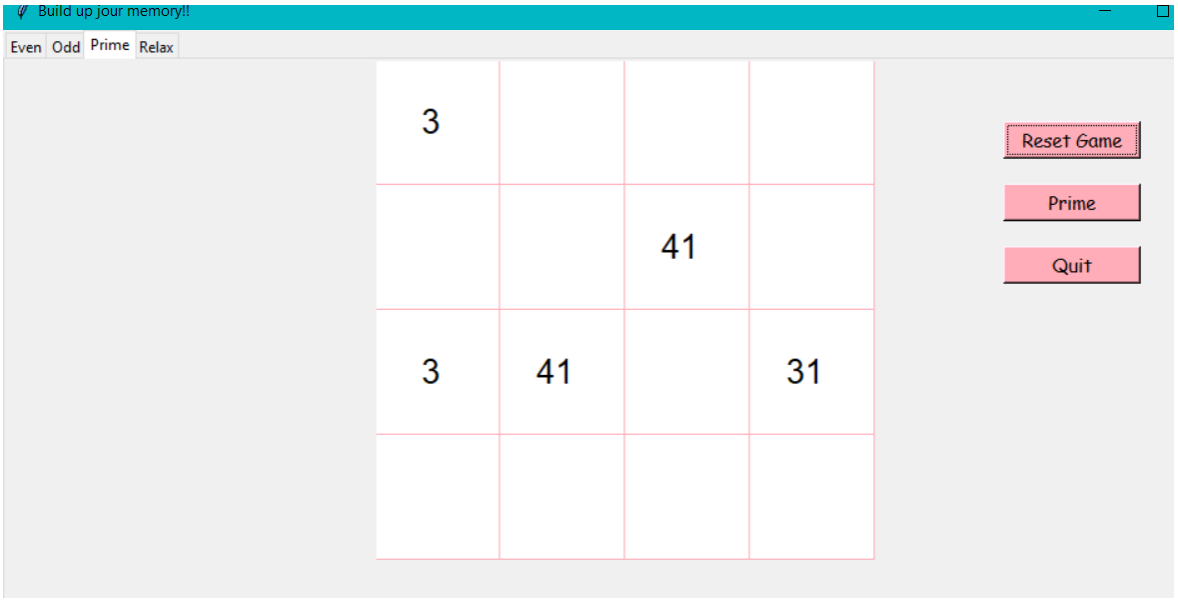
7. Odd Mode is similar to Even. Odd Mode Winning Condition. All numbers are odd.



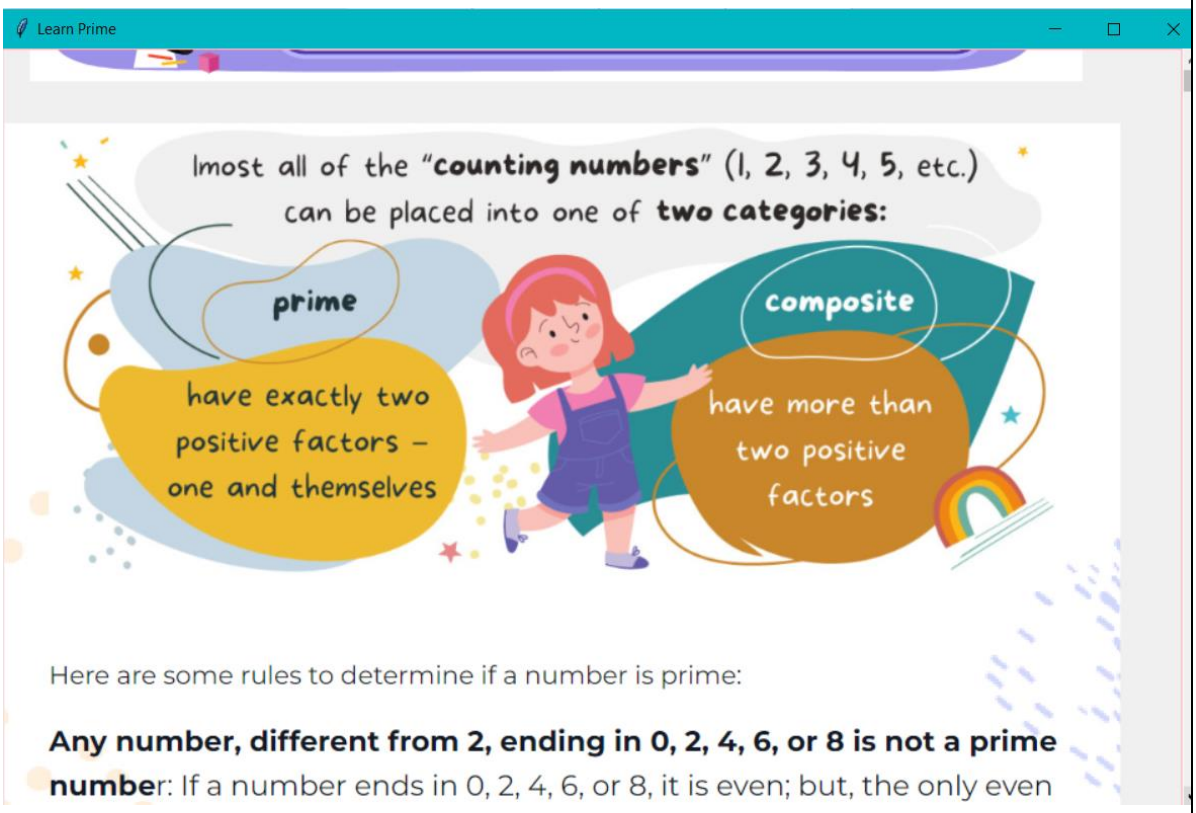
8.OddMode when clicking LearnMore and Odd Buttton



9.PrimeMode while playing. All numbers are Prime.



10. When clicking Prime Button, the Prime def and learn Prime window will appear.



Python Source codes

You can also view it here.

https://github.com/daeunek/python_practice/tree/main/1st%20year%201st%20sem%20Project1

You need to download all files in 1st year 1st sem folder to access background image.

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
import random

class Puzzle(object):
    def __init__(self, root):
        self.Window = root
        self.Window.title("Build up jour memory!!")
        self.Window.geometry('1000x1000')
        self.Window.resizable(0,0)

        # Tabs
        self.tabs = ttk.Notebook(self.Window)
        self.user_moves = 0
        self.clicks = [200,200]
        self.match = 0
        self.homepage = None   #to keep track of the homepage instace

    def show_next(self):
        if self.homepage:
            self.homepage.main.destroy()
        self.tabs.pack(fill = 'both', expand = True)
        self.Window.mainloop()

    def closing(self):
        if messagebox.askyesno(title="Quit?", message="Do you really want to
quit?"):
            self.Window.destroy()
```

```
class EvenNums(Puzzle):
   def __init__(self, notebook):
      super().__init__(notebook.Window)
      self.even = ttk.Frame(notebook.tabs)
      notebook.tabs.add(self.even, text="Even")

      # Canvas
      self.even_canvas = Canvas(self.even, width=400, height=600)
      self.even_canvas.pack()

      self.ans1 = ['A', 'A', 'B', 'B', 'C', 'C', 'D', 'D', 'E', 'E', 'F', 'F', 'G', 'G', 'H', 'H']
      random.shuffle(self.ans1)
      self.ans1 = [self.ans1[:4],
                   self.ans1[4:8],
                   self.ans1[8:12],
                   self.ans1[12:]]

      even_lst = self.get_even(2,100)
      self.even_nums = random.sample(even_lst, 8)

      #initialization of grid with placeholder
      self.grid_1 = [list('.' * 4) for i in range(4)]

      self.reset_but = Button(self.even,font = ('Comic Sans MS', 11),text = 'Reset
Game', bg = 'cyan', fg = 'black', command = self.restart)# relief = RAISED, bd =
2)
      self.reset_but.place(x = 800, y = 50, width = 110, height = 30)
      self.even_but = Button(self.even,font = ('Comic Sans MS', 11),text = 'Even',
bg = 'cyan', fg = 'black', command = self.learn_even)# relief = RAISED, bd = 2)
      self.even_but.place(x = 800, y = 100, width = 110, height = 30)
      self.quit_but = Button(self.even,font = ('Comic Sans MS', 11),text = 'Quit',
bg = 'cyan', fg = 'black', command = self.closing)# relief = RAISED, bd = 2)
      self.quit_but.place(x = 800, y = 150, width = 110, height = 30)


      self.img1_path = '1st year 1st sem Project1/img1.png'
      self.img1 = PhotoImage(file = self.img1_path)

      self.img2_path = '1st year 1st sem Project1/evenimg2.png'
      self.img2 = PhotoImage(file = self.img2_path)
```

```
    def learn_even(self):
        learn_platform = Toplevel(self.Window)
        learn_platform.title("Learn Even")
        learn_platform.geometry("950x600")

        c = Canvas(learn_platform, highlightthickness=1, highlightbackground =
'cyan')
        c.grid(row = 0, column= 0, sticky = 'nsew')

        ver_bar = Scrollbar(learn_platform, command=c.yview)
        ver_bar.grid(row = 0, column= 2, sticky = NS)

        c.create_text(10, 30, anchor='w', text='''\nEven Numbers are amounts
that can be exactly divided by two.They are numbers \nthat end in 0, 2, 4, 6, or
8.\nHere are some examples of even numbers:\n\t2, 14, 16, 28, 40''',
font=('Courier New', 15), fill = '#68228B')
        c.create_image(10,350, anchor = 'w', image = self.img1)
        c.create_image(10,900, anchor = 'w', image = self.img2)
        c.create_text(10,1250,anchor = 'w', text = 'Reference(ARGOPREP) -
https://argoprep.com/math/1st-grade/numbers/even-and-odd-numbers-on-a-
number-line/\n\n', font=('arial', 12), fill = 'blue')

        c.update_idletasks()
        c.configure(scrollregion = c.bbox('all'))

        learn_platform.rowconfigure(0, weight=1)  # Make the first row
expandable
        learn_platform.columnconfigure(0, weight=1)  # Make the first column
expandable

    def get_even(self, a, b):
        even_list = []
        for i in range(a,b):
            if i % 2 == 0:
                even_list.append(i)
            else:
                continue
        return even_list
```

```
    def draw_elements(self, ans, x, y):
        if ans == 'A':
            b = self.even_canvas.create_text(100*x + 45, 100*y + 50, text =
self.even_nums[0], font = ('arial', 20))
        elif ans == 'B':
            b = self.even_canvas.create_text(100*x + 45, 100*y + 50, text =
self.even_nums[1], font = ('arial', 20))
        elif ans == 'C':
            b = self.even_canvas.create_text(100*x + 45, 100*y + 50, text =
self.even_nums[2], font = ('arial', 20))
        elif ans == 'D':
            b = self.even_canvas.create_text(100*x + 45, 100*y + 50, text =
self.even_nums[3], font = ('arial', 20))
        elif ans == 'E':
            b = self.even_canvas.create_text(100*x + 45, 100*y + 50, text =
self.even_nums[4], font = ('arial', 20))
        elif ans == 'F':
            b = self.even_canvas.create_text(100*x + 45, 100*y + 50, text =
self.even_nums[5], font = ('arial', 20))
        elif ans == 'G':
            b = self.even_canvas.create_text(100*x + 45, 100*y + 50, text =
self.even_nums[6], font = ('arial', 20))
        elif ans == 'H':
            b = self.even_canvas.create_text(100*x + 45, 100*y + 50, text =
self.even_nums[7], font = ('arial', 20))

    def draw_board(self):
        for i in range(4):
            for j in range(4):
                r = self.even_canvas.create_rectangle(100 * i, 100 * j, 100 * i + 100,
100 * j + 100, fill="white", outline = 'cyan')
                if self.grid_1[i][j] != '.':
                    print("ans", self.grid_1[i][j])
                    self.draw_elements(self.grid_1[i][j],i,j)

        # win condition
        if self.match == 8:
            self.even_canvas.create_text(200, 440, text=f"Your moved:
{self.user_moves}times😤", font=('Helvetica', 16), anchor ='center')
```

```
            self.even_canvas.create_text(12, 470, text="😤", font=('Helvetica',
16),fill ='cyan')
            self.even_canvas.create_text(200, 520, text="You noticed that all of
those numbers are divisible by 2.\nThus It is called \"Even Number\".", font =
("Helvetica",11),fill = 'black')
            self.even_canvas.create_rectangle(5, 480, 385, 562, outline = "cyan")
            self.con_but = Button(self.even_canvas, font = ('Helvetica', 11), text =
'Learn More...', fg = '#009ACD', command = self.learn_even, relief = FLAT)
            self.con_but.place(x= 230, y = 520, height = 15)

        else:
            pass

    #event is used to track mouse movement
    def set_up(self, event):
        col = event.x // 100  # to track which column was clicked
        row = event.y // 100  # to track which row was clicked

        try:
            if self.grid_1[col][row] == '.':
                self.user_moves += 1

                # First click
                if self.clicks[0] > 4:
                    self.clicks = [col, row]
                    self.grid_1[col][row] = self.ans1[col][row]
                    self.draw_board()
                else:
                    # Subsequent clicks
                    self.grid_1[col][row] = self.ans1[col][row]
                    print(self.grid_1)
                    self.draw_board()

                    # match condition
                    # previous click is noted in self.clicks() if matched, we will reset
previous clicks again
                    if self.grid_1[col][row] == self.grid_1[self.clicks[0]][self.clicks[1]]:
                        print("after match grid", self.grid_1)
                        self.match += 1
                        self.clicks = [100, 100]
```

```
                self.draw_board()

            else:
                # If not matched, we set it to '.' again to draw grid
                self.grid_1[self.clicks[0]][self.clicks[1]] = '.'
                print("after not finding f=grid1", self.grid_1)
                self.draw_board()
                self.clicks = [col, row]
        else:
            print("Index out of range")

    except IndexError as e:
        print(f"IndexError: {e}")


  def restart(self):
    try:
        self.con_but.destroy()
    except AttributeError as e:
        print(f"AttributeError: {e}")

    self.even_canvas.delete("all")
    self.match = 0
    self.user_moves = 0
    self.ans1 = list('AABBCCDDEEFFGGHH')
    random.shuffle(self.ans1)
    self.ans1 = [self.ans1[:4],
                 self.ans1[4:8],
                 self.ans1[8:12],
                 self.ans1[12:]]
    self.grid_1 = [list('.' * 4) for i in range(4)]
    self.even_canvas.bind("<Button-1>", self.set_up)
    self.draw_board()


class OddNums(Puzzle):
  def __init__(self, notebook):
    super().__init__(notebook.Window)
    self.odd = ttk.Frame(notebook.tabs)
    notebook.tabs.add(self.odd, text = "Odd")
```

```
        self.odd_canvas = Canvas(self.odd, width = 400, height = 600)
        self.odd_canvas.pack()

        self.ans2 = list("AABBCCDDEEFFGGHH")
        random.shuffle(self.ans2)
        self.ans2 = [self.ans2[:4],
                     self.ans2[4:8],
                     self.ans2[8:12],
                     self.ans2[12:]]

        odd_lst = self.get_odd(1,100)
        self.odd_nums = random.sample(odd_lst, 8)

        self.grid2 = [list('.' * 4) for i in range(4)]

        self.reset_but = Button(self.odd,font = ('Comic Sans MS', 11),text = 'Reset
Game', bg = 'orange', fg = 'black', command = self.restart)
        self.reset_but.place(x = 800, y = 50, width = 110, height = 30)
        self.odd_but = Button(self.odd,font = ('Comic Sans MS', 11),text = 'Odd', bg
= 'orange', fg = 'black', command = self.learn_odd)
        self.odd_but.place(x = 800, y = 100, width = 110, height = 30)
        self.quit_but = Button(self.odd,font = ('Comic Sans MS', 11),text = 'Quit',
bg = 'orange', fg = 'black', command = self.closing)
        self.quit_but.place(x = 800, y = 150, width = 110, height = 30)

        self.img1_p = "1st year 1st sem Project1/odd1.png"
        self.img1 = PhotoImage(file = self.img1_p)

        self.img2_path = '1st year 1st sem Project1/odd2.png'
        self.img2 = PhotoImage(file = self.img2_path)

    def learn_odd(self):
        learn_platform = Toplevel(self.Window)
        learn_platform.title("Learn Odd")
        learn_platform.geometry("920x600")

        c = Canvas(learn_platform, highlightthickness= 1, highlightbackground =
'orange')
        c.grid(row = 0, column= 0, sticky = 'nsew')
```

```
        ver_bar = Scrollbar(learn_platform, command=c.yview)
        ver_bar.grid(row = 0, column= 2, sticky = NS)

        c.create_text(10, 30, anchor='w', text='''\n\nOdd Numbers are amounts
that cannot be exactly divided by two.They are \nnumbers that end in 1, 3, 5,
7, or 9.\nHere are some examples of odd numbers:\n\t1, 13, 25, 37, 49''',
font=('Courier New', 15), fill = '#68228B')
        c.create_image(10,320, anchor = 'w', image = self.img1)
        c.create_image(10,880, anchor = 'w', image = self.img2)
        c.create_text(10,1250,anchor = 'w', text = '\n\nReference(ARGOPREP) -
https://argoprep.com/math/1st-grade/numbers/even-and-odd-numbers-on-a-
number-line/\n\n', font=('arial', 12), fill = 'blue')

        c.update_idletasks()
        c.configure(scrollregion = c.bbox('all'))

        learn_platform.rowconfigure(0, weight=1)  # Make the first row
expandable
        learn_platform.columnconfigure(0, weight=1)  # Make the first column
expandable


    def get_odd(self, a, b):
        odd_list = []
        for i in range(a,b):
            if i % 2 != 0:
                odd_list.append(i)
            else:
                continue
        return odd_list


    def draw_elements(self, ans, x, y):
        if ans == 'A':
            b = self.odd_canvas.create_text(100*x + 45, 100*y + 50, text =
self.odd_nums[0], font = ('arial', 20))
        elif ans == 'B':
            b = self.odd_canvas.create_text(100*x + 45, 100*y + 50, text =
self.odd_nums[1], font = ('arial', 20))
        elif ans == 'C':
```

```
            b = self.odd_canvas.create_text(100*x + 45, 100*y + 50, text =
self.odd_nums[2], font = ('arial', 20))
        elif ans == 'D':
            b = self.odd_canvas.create_text(100*x + 45, 100*y + 50, text =
self.odd_nums[3], font = ('arial', 20))
        elif ans == 'E':
            b = self.odd_canvas.create_text(100*x + 45, 100*y + 50, text =
self.odd_nums[4], font = ('arial', 20))
        elif ans == 'F':
            b = self.odd_canvas.create_text(100*x + 45, 100*y + 50, text =
self.odd_nums[5], font = ('arial', 20))
        elif ans == 'G':
            b = self.odd_canvas.create_text(100*x + 45, 100*y + 50, text =
self.odd_nums[6], font = ('arial', 20))
        elif ans == 'H':
            b = self.odd_canvas.create_text(100*x + 45, 100*y + 50, text =
self.odd_nums[7], font = ('arial', 20))

    def draw_board(self):
        for i in range(4):
            for j in range(4):
                base = self.odd_canvas.create_rectangle(100 * i, 100 * j, 100 * i + 100,
100*j + 100, fill = "white", outline = 'orange')
                if self.grid2[i][j] != '.':
                    print("ans", self.grid2[i][j])
                    self.draw_elements(self.grid2[i][j], i, j)

        if self.match == 8:
            self.odd_canvas.create_text(200, 440, text=f"Your moved:
{self.user_moves}times😤", font=('Helvetica', 16), anchor ='center')
            self.odd_canvas.create_text(12, 470, text="😤", font=('Helvetica',
16),fill ='orange')
            self.odd_canvas.create_text(200, 520, text="You noticed that all of
those numbers are not divisible \nby 2.Thus It is called \"Odd Number\"", font
= ("Helvetica",11),fill = 'black')
            self.odd_canvas.create_rectangle(5, 480, 380, 562, outline = "orange")
            self.con_but = Button(self.odd_canvas, font = ('Helvetica', 11), text =
'Learn More...', fg = 'orange', command = self.learn_odd, relief = FLAT)
            self.con_but.place(x= 230, y = 520, height = 15)
```

```
    else:
        pass

def set_up(self, event):
    col = event.x // 100
    row = event.y // 100

    try:
        if self.grid2[col][row] == ".":
            self.user_moves += 1

            # First click
            if self.clicks[0] > 4:
                self.grid2[col][row] = self.ans2[col][row]
                self.clicks = [col, row]
                self.draw_board()
            else:
                self.grid2[col][row] = self.ans2[col][row]
                self.draw_board()
                if self.grid2[col][row] == self.grid2[self.clicks[0]][self.clicks[1]]:
                    print("after match grid", self.grid2)
                    self.match += 1
                    self.clicks = [200, 200]
                    self.draw_board()
                else:
                    self.grid2[self.clicks[0]][self.clicks[1]] = '.'
                    print("Not match :", self.grid2)
                    self.draw_board()
                    self.clicks = [col, row]
        else:
            print("Index Error")

    except IndexError:
        print("IndexError: Click only Grid")


def restart(self):
    try:
        self.con_but.destroy()
    except AttributeError as e:
        print(f"AttributeError: {e}")
```

```
        random.shuffle(self.ans2)
        self.an2 = [self.ans2[:4],
                    self.ans2[4:8],
                    self.ans2[8:12],
                    self.ans2[12:]]
        self.grid2 = [list('.' * 4) for i in range(4)]
        self.match = 0
        self.user_moves = 0
        self.odd_canvas.delete("all")
        self.odd_canvas.bind("<Button-1>", self.set_up)
        self.draw_board()


class PrimeNums(Puzzle):
    def __init__(self, notebook):
        super().__init__(notebook.Window)

        self.prime = ttk.Frame(notebook.tabs, style = "TFrame")
        notebook.tabs.add(self.prime, text = "Prime")

        self.prime_canvas = Canvas(self.prime, width = 400, height = 600)
        self.prime_canvas.pack()

        self.ans3 = list('AABBCCDDEEFFGGHH')
        random.shuffle(self.ans3)
        self.ans3 = [self.ans3[:4],
                    self.ans3[4:8],
                    self.ans3[8:12],
                    self.ans3[12:]]

        prime_lst = self.get_prime(0,100)
        print("Prime",prime_lst)
        self.prime_nums = random.sample(prime_lst, 8)

        self.grid_3 = [list('.' * 4) for i in range(4)]
        self.reset_but = Button(self.prime,font = ('Comic Sans MS', 11),text =
'Reset Game', bg = '#FFAEB9', fg = 'black', command = self.restart)
        self.reset_but.place(x = 800, y = 50, width = 110, height = 30)
        self.prime_but = Button(self.prime,font = ('Comic Sans MS', 11),text =
'Prime', bg = '#FFAEB9', fg = 'black', command = self.learn_prime)# relief =
RAISED, bd = 2)
```

```
        self.prime_but.place(x = 800, y = 100, width = 110, height = 30)
        self.quit_but = Button(self.prime,font = ('Comic Sans MS', 11),text = 'Quit',
bg = '#FFAEB9', fg = 'black', command = self.closing)# relief = RAISED, bd = 2)
        self.quit_but.place(x = 800, y = 150, width = 110, height = 30)

        self.img1_path = '1st year 1st sem Project1/prime1.png'
        self.img1 = PhotoImage(file = self.img1_path)

        self.img2_path = '1st year 1st sem Project1/prime2.png'
        self.img2 = PhotoImage(file = self.img2_path)


    def learn_prime(self):
        learn_platform = Toplevel(self.Window)
        learn_platform.title("Learn Prime")
        learn_platform.geometry("950x600")

        c = Canvas(learn_platform, highlightbackground = '#FFAEB9',
highlightthickness=1)
        c.grid(row = 0, column= 0, sticky = 'nsew')

        ver_bar = Scrollbar(learn_platform, command=c.yview)
        ver_bar.grid(row = 0, column= 2, sticky = NS)

        c.create_text(15, 30, anchor='w', text='''\nA prime number is a whole
number greater than 1 with only two factors; itself and 1. A prime number
\ncannot be divided by any other positive integer without leaving a remainder,
decimal or fraction.\nExamples of PrimeNumbers:\n\t2, 3, 5, 7, 11''',
font=('arial', 15), fill = '#474747')
        c.create_image(30,550, anchor = 'w', image = self.img1)
        c.create_image(10,1300, anchor = 'w', image = self.img2)
        c.create_text(10,1650,anchor = 'w', text = 'Reference(ARGOPREP) -
https://argoprep.com/blog/k8/prime-numbers/\n\n', font=('arial', 12), fill =
'blue')

        c.update_idletasks()
        c.configure(scrollregion = c.bbox('all'))

        learn_platform.rowconfigure(0, weight=1)  # Make the first row
expandable
```

```
        learn_platform.columnconfigure(0, weight=1)  # Make the first column
expandable

    def get_prime(self, a, b):
        prime_lst = []
        for num in range(a, b):
            is_prime = True
            for i in range(2, num):
                if num % i == 0:
                    is_prime = False
                    break
            if is_prime and num > 1:
                prime_lst.append(num)
        return prime_lst

    def draw_elements(self, ans, x, y):
        if ans == 'A':
            b = self.prime_canvas.create_text(100*x + 45, 100*y + 50, text =
self.prime_nums[0], font = ('arial', 20))
        elif ans == 'B':
            b = self.prime_canvas.create_text(100*x + 45, 100*y + 50, text =
self.prime_nums[1], font = ('arial', 20))
        elif ans == 'C':
            b = self.prime_canvas.create_text(100*x + 45, 100*y + 50, text =
self.prime_nums[2], font = ('arial', 20))
        elif ans == 'D':
            b = self.prime_canvas.create_text(100*x + 45, 100*y + 50, text =
self.prime_nums[3], font = ('arial', 20))
        elif ans == 'E':
            b = self.prime_canvas.create_text(100*x + 45, 100*y + 50, text =
self.prime_nums[4], font = ('arial', 20))
        elif ans == 'F':
            b = self.prime_canvas.create_text(100*x + 45, 100*y + 50, text =
self.prime_nums[5], font = ('arial', 20))
        elif ans == 'G':
            b = self.prime_canvas.create_text(100*x + 45, 100*y + 50, text =
self.prime_nums[6], font = ('arial', 20))
        elif ans == 'H':
            b = self.prime_canvas.create_text(100*x + 45, 100*y + 50, text =
self.prime_nums[7], font = ('arial', 20))
```

```
    def draw_board(self):
        for i in range(4):
            for j in range(4):
                r = self.prime_canvas.create_rectangle(100 * i, 100 * j, 100 * i + 100,
100 * j + 100, fill="white", outline = "#FFAEB9")
                if self.grid_3[i][j] != '.':
                    self.draw_elements(self.grid_3[i][j],i,j)

        # win condition
        if self.match == 8:
            self.prime_canvas.create_text(200, 440, text=f"Your moved:
{self.user_moves}times😤", font=('Helvetica', 16), anchor ='center')
            self.prime_canvas.create_text(12, 470, text="😤", font=('Helvetica',
16),fill ='#FFAEB9')
            self.prime_canvas.create_text(200, 520, text="You noticed that all of
those numbers are not divisible \nby any numbers.Thus It is called \"Prime
Number\"", font = ("Helvetica",11),fill = 'black')
            self.prime_canvas.create_rectangle(5, 480, 380, 562, outline =
"#FFAEB9")
            self.con_but = Button(self.prime_canvas, font = ('Helvetica', 11), text =
'Learn More...', fg = '#FF6EB4', command = self.learn_prime, relief = FLAT)
            self.con_but.place(x= 18, y = 540, height = 12)

        else:
            pass

    def set_up(self, event):
        col = event.x // 100  # to track which column was clicked
        row = event.y // 100  # to track which row was clicked

        try:
            if self.grid_3[col][row] == '.':
                self.user_moves += 1

                #Fist click
                if self.clicks[0] > 4:
                    self.clicks = [col, row]
                    self.grid_3[col][row] = self.ans3[col][row]
                    self.draw_board()
```

```
            #subsequent clicks
            else:
                self.grid_3[col][row] = self.ans3[col][row]
                print(self.grid_3)
                self.draw_board()
                if self.grid_3[col][row] == self.grid_3[self.clicks[0]][self.clicks[1]]:
                    print("after match grid", self.grid_3)
                    self.match += 1
                    self.clicks = [100, 100]
                    self.draw_board()
                else:
                    self.grid_3[self.clicks[0]][self.clicks[1]] = '.'
                    print("after not finding f=grid1",self.grid_3)
                    self.draw_board()
                    self.clicks = [col, row]
        except IndexError:
            print("Index Error: Index out of range")

    def restart(self):
        try:
            self.con_but.destroy()
        except AttributeError as e:
            print(f"AttributeError: {e}")

        random.shuffle(self.ans3)
        self.an3 = [self.ans3[:4],
                    self.ans3[4:8],
                    self.ans3[8:12],
                    self.ans3[12:]]
        self.grid_3 = [list('.' * 4) for i in range(4)]
        self.match = 0
        self.user_moves = 0
        self.prime_canvas.delete("all")
        self.prime_canvas.bind("<Button-1>", self.set_up)
        self.draw_board()

class Relax(Puzzle):
    def __init__(self, notebook):
        super().__init__(notebook.Window)
```

```
        self.relax = ttk.Frame(notebook.tabs)
        notebook.tabs.add(self.relax, text = "Relax")

        self.relax_canvas = Canvas(self.relax, width = 400, height = 600)
        self.relax_canvas.pack()

        self.ans4 = list('AABBCCDDEEFFGGHH')
        random.shuffle(self.ans4)
        self.ans4 = [self.ans4[:4],
                     self.ans4[4:8],
                     self.ans4[8:12],
                     self.ans4[12:]]

        self.grid_4 = [list('.' * 4) for i in range(4)]
        self.reset_but = Button(self.relax,font = ('Comic Sans MS', 11),text = 'Reset
Game', bg = '#7FFF00', fg = 'black', command = self.restart)
        self.reset_but.place(x = 800, y = 50, width = 110, height = 30)
        self.quit_but = Button(self.relax,font = ('Comic Sans MS', 11),text = 'Quit',
bg = '#7FFF00', fg = 'black', command = self.closing)# relief = RAISED, bd = 2)
        self.quit_but.place(x = 800, y = 100, width = 110, height = 30)


    def draw_elements(self, ans, x, y):
        if ans =='A':
            d=self.relax_canvas.create_rectangle(100*x+20, 100*y + 20, 100*x
+100-20, 100*y +100-20,fill='orange')
        elif ans =='B':
            d=self.relax_canvas.create_rectangle(100*x+30, y*100+20, 100*x+30 +
40, 100*y+20 +60 ,fill='red', outline = 'dark red')
        elif ans =='C':
            d=self.relax_canvas.create_rectangle(100*x+20,y*100+20,100*x+100-
20,100*y+100-20,fill='cyan', outline = 'dark blue')
        elif ans =='D':
            d=self.relax_canvas.create_oval(100*x+20,y*100+20,100*x+100-
20,100*y+100-20,fill='#76EE00')
        elif ans =='E':
            d=self.relax_canvas.create_oval(100*x+20,y*100+20,100*x+100-
20,100*y+100-20,fill='yellow')
        elif ans =='F':
            d=self.relax_canvas.create_oval(100*x+20,y*100+20,100*x+100-
20,100*y+100-20,fill='#FF3E96')
```

```
        elif ans =='G':
            d=self.relax_canvas.create_polygon(100*x+50,y*100+20,100*x+20,100*
y+100-20,100*x+100-20,100*y+100-20,fill='blue')
        elif ans =='H':
            d=self.relax_canvas.create_polygon(100*x+50,y*100+20,100*x+20,100*
y+100-20,100*x+100-20,100*y+100-20,fill='green')
        elif ans =='I':
            d=self.relax_canvas.create_polygon(100*x+50,y*100+20,100*x+20,100*
y+100-20,100*x+100-20,100*y+100-20,fill='#97FFFF')

    def draw_board(self):
        for i in range(4):
            for j in range(4):
                r = self.relax_canvas.create_rectangle(100 * i, 100 * j, 100 * i + 100,
100 * j + 100, fill="white", outline = "#7FFF00")
                if self.grid_4[i][j] != '.':
                    self.draw_elements(self.grid_4[i][j],i,j)

        # win condition
        if self.match == 8:
            self.relax_canvas.create_text(200, 450, text=f"You moved the cards
{self.user_moves} times", font=('arial', 20))

    def set_up(self, event):
        col = event.x // 100  # to track which column was clicked
        row = event.y // 100  # to track which row was clicked

        try:
            if self.grid_4[col][row] == '.':
                self.user_moves += 1

                #Fist click
                if self.clicks[0] > 4:
                    self.clicks = [col, row]
                    self.grid_4[col][row] = self.ans4[col][row]
                    self.draw_board()
                #subsequent clicks
                else:
                    self.grid_4[col][row] = self.ans4[col][row]
                    self.draw_board()
```

```
                if self.grid_4[col][row] == self.grid_4[self.clicks[0]][self.clicks[1]]:
                    self.match += 1
                    self.clicks = [100, 100]
                    self.draw_board()
                else:
                    self.grid_4[self.clicks[0]][self.clicks[1]] = '.'
                    self.draw_board()
                    self.clicks = [col, row]
            else:
                print("Index error")
        except IndexError:
            print("IndexError: Click only grid")

    def restart(self):
        try:
            self.con_but.destroy()
        except AttributeError as e:
            print(f"AttributeError: {e}")
        self.an4 = [self.ans4[:4],
                    self.ans4[4:8],
                    self.ans4[8:12],
                    self.ans4[12:]]
        self.grid_4 = [list('.' * 4) for i in range(4)]
        self.match = 0
        self.user_moves = 0
        self.relax_canvas.delete("all")
        self.relax_canvas.bind("<Button-1>", self.set_up)
        self.draw_board()
        print("drawing")

class HomePage(object):
    def __init__(self, instance):
        self.instance = instance
        img_path = "D:\\KMITL\\pythonProject\\1st year 1st sem
Project1\\final1.png"
        self.bg_image = PhotoImage(file = img_path)
        self.main = Frame()
        self.main.pack()
        self.main.place(x=0, y=0, width=1000, height=1000)
```

```
        # create a label to display the background label
        self.bg_label = Label(self.main, image=self.bg_image)
        self.bg_label.pack()
        # self.bg_label.place(relwidth=1, relheight=1)

        self.b1 = Button(self.main, text='Start Game', font = ("Comic Sans MS",14),
bg = "#40E0D0", command=self.instance.show_next)
        self.b1.place(x=700, y=300, width=130, height=50)

        self.b2 = Button(self.main, text='Quit', font = ("Comic Sans MS",14), bg =
"#40E0D0", command=self.instance.closing)
        self.b2.place(x=700, y=400, width=130, height=50)

def main():
    root = Tk()
    puzzle = Puzzle(root)
    puzzle.homepage = HomePage(puzzle)
    a = EvenNums(puzzle)
    b = OddNums(puzzle)
    c = PrimeNums(puzzle)
    d = Relax(puzzle)

    a.even_canvas.bind("<Button-1>", a.set_up)
    a.draw_board()

    b.odd_canvas.bind("<Button-1>", b.set_up)
    b.draw_board()

    c.prime_canvas.bind("<Button-1>", c.set_up)
    c.draw_board()

    d.relax_canvas.bind("<Button-1>", d.set_up)
    d.draw_board()

    # Closing
    root.protocol("WM_DELETE_WINDOW", puzzle.closing)
    root.mainloop()
main()
```