Date: 17/01/2024

19 MAT 205

Assignment – 1

Monty Hall Game

Kaushik Kumbhat

CH.EN.U4CSE22029

CODE:

mont_hall_goats.py > MontyHallGUI > create_doors

```python
import tkinter as tk
from tkinter import messagebox, Canvas, PhotoImage, VERTICAL
from PIL import Image, ImageTk
import random

class MontyHallGUI:
    def __init__(self, master):
        self.master = master
        master.title("Monty Hall Problem - CH.EN.U4CSE22029")
        self.doors_opened_count = 0
        # Label
        self.label = tk.Label(master, text="Select the number of doors:")
        self.label.pack()

        # Entry for number of doors
        self.num_doors_entry = tk.Entry(master)
        self.num_doors_entry.pack()

        # Button to run simulation
        self.run_button = tk.Button(master, text="Run Simulation", command=self.run_simulation)
        self.run_button.pack()

        # Frame for door buttons and canvas with scrollbar
        self.canvas_frame = tk.Frame(master)
        self.canvas_frame.pack()

        self.canvas = Canvas(self.canvas_frame, width=800, height=200, scrollregion=(0, 0, 800, 200))
        self.scrollbar = tk.Scrollbar(self.canvas_frame, orient=VERTICAL, command=self.canvas.yview)
        self.canvas.configure(yscrollcommand=self.scrollbar.set)

```

```
        # List to store door buttons
        self.door_buttons = []

        # Variables for Monty Hall problem
        self.prize_door = None
        self.selected_door = None
        self.doors_opened = False

        # Loa
        self.door_image = Image.open("door.png")
        self.door_image = self.door_image.resize((50, 100), Image.LANCZOS)
        self.door_photo = ImageTk.PhotoImage(self.door_image)

        self.open_door_image = Image.open("open_door.png")
        self.open_door_image = self.open_door_image.resize((50, 100), Image.LANCZOS)
        self.open_door_photo = ImageTk.PhotoImage(self.open_door_image)

    def create_doors(self, num_doors):
        # Clear existing door buttons
        for button in self.door_buttons:
            button.destroy()
        self.door_buttons = []

        # Calculate the number of rows needed
        num_rows = (num_doors + 9) // 10

        # Create new door buttons with space between them
        button_width = 60
        button_height = 120
        space_between = 10
```

(variable) door_image: Image

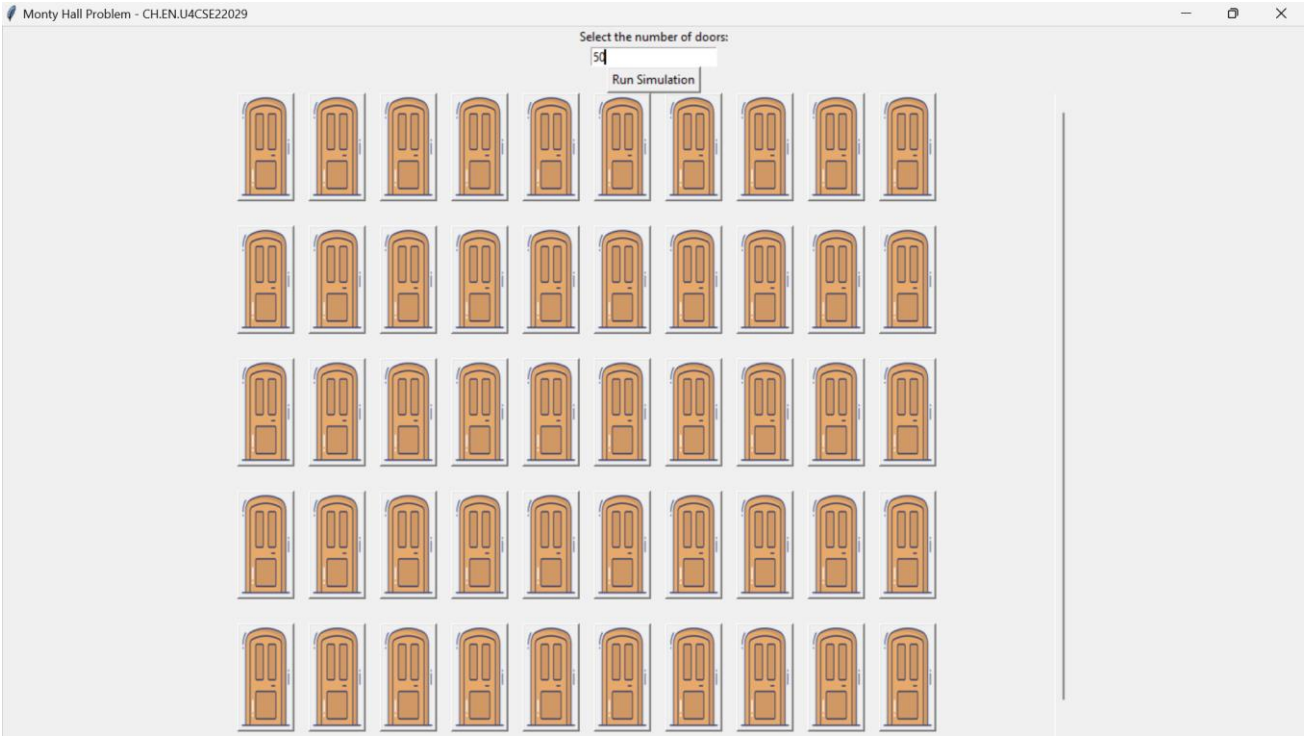
```
        for i in range(num_doors):
            row_index = i // 10
            col_index = i % 10

            x_position = col_index * (button_width + space_between)
            y_position = row_index * (button_height + space_between)

            door_button = tk.Button(self.canvas, image=self.door_photo, text=f"Door {i + 1}",
                                    command=lambda i=i: self.select_door(i))
            door_button.photo = self.door_photo
            self.door_buttons.append(door_button)

            # Add the door button to the canvas using the create_window method
            self.canvas.create_window(x_position, y_position, window=door_button, anchor='nw')

        # Update canvas size and scroll region
        canvas_width = max(800, 10 * (button_width + space_between))
        self.canvas.config(width=canvas_width, height=num_rows * (button_height + space_between),
                           scrollregion=(0, 0, canvas_width, num_rows * (button_height + space_between)))
        self.canvas.pack(side="left", fill="both", expand=True)
        self.scrollbar.pack(side="right", fill="y")

    def run_simulation(self):
        global num_doors
        try:
            num_doors = int(self.num_doors_entry.get())
        except ValueError:
            messagebox.showerror("Error", "Please enter a valid number for doors.")

            return
```

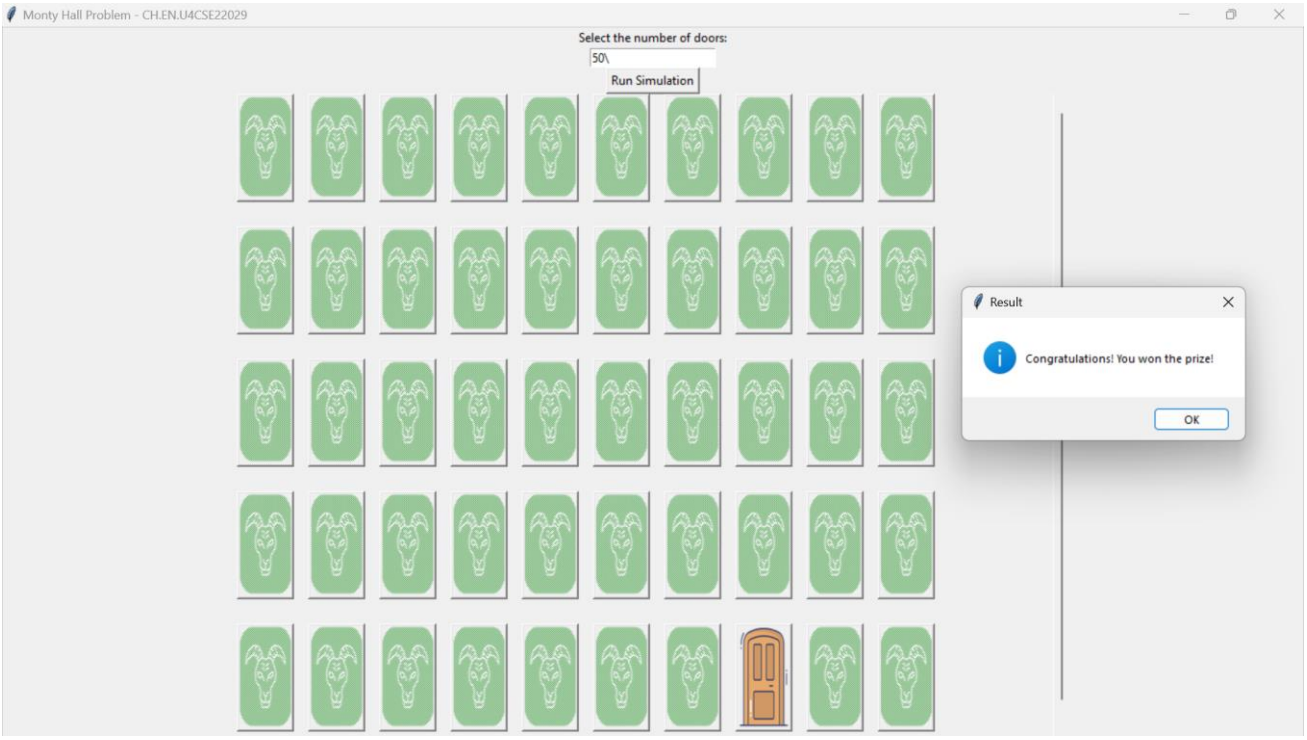
```
        if num_doors < 3:
            messagebox.showerror("Error", "Number of doors should be at least 3.")
            return

        # Create door buttons
        self.create_doors(num_doors)

        # Randomly select a door with the prize
        self.prize_door = random.randint(0, num_doors - 1)
        self.selected_door = None
        self.doors_opened = False

    def select_door(self, door_index):
        #global num_doors
        if self.doors_opened:
            if door_index == self.prize_door:
                messagebox.showinfo("Result", "Congratulations! You won the prize!")
                self.run_simulation()
            else:
                messagebox.showinfo("Result", "Sorry, you selected the wrong door.")
                self.run_simulation()
        else:
            self.selected_door = door_index
            self.open_doors()
            messagebox.showinfo("Monty Hall", f"You selected door {door_index + 1}. Some doors have been opened. Woul

    def open_doors(self):
        # Find the next door to open
        opened_door = self.find_next_door_to_open()
```

```
        # If all doors are open, show the result
        if opened_door is None:
            if self.selected_door == self.prize_door:
                messagebox.showinfo("Result", "Congratulations! You won the prize!")
            else:
                messagebox.showinfo("Result", "Sorry, you selected the wrong door.")
            self.run_simulation()
        else:
            # Open the selected door
            self.door_buttons[opened_door].config(state="disabled", image=self.open_door_photo)
            self.doors_opened_count += 1

            if self.selected_door == self.prize_door:
                messagebox.showinfo("Result", "Congratulations! You won the prize!")
                self.run_simulation()


            # Show a message if all doors are open
            if self.doors_opened_count == num_doors - 2:  # -2 to account for the selected door and the prize door
                messagebox.showinfo("Monty Hall", f"You selected door {self.selected_door + 1}. All other doors have

    def find_next_door_to_open(self):
        # Find the next closed door to open
        for i in range(len(self.door_buttons)):
            if i != self.selected_door and i != self.prize_door and self.door_buttons[i]['state'] == "normal":
                return i
        return None

```

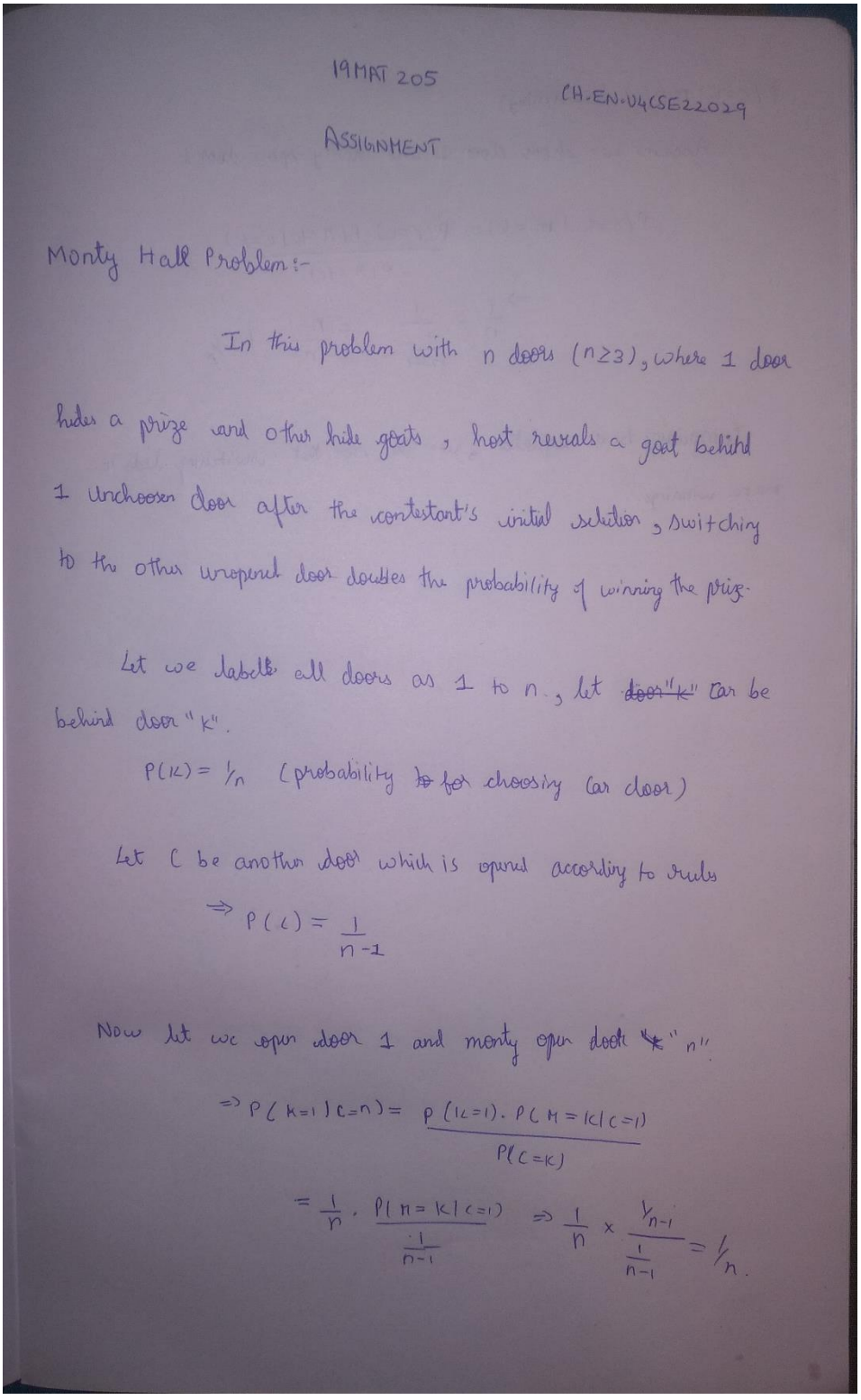
```
if __name__ == "__main__":
    root = tk.Tk()
    root.geometry("200x200")
    app = MontyHallGUI(root)
    root.mainloop()
```

OUTPUT:


Monty Hall Problem - CH.EN.U4CSE22029
Select the number of doors:
50
Run Simulation


Monty Hall Problem - CH.EN.U4CSE22029
Select the number of doors:
50\
Run Simulation
Result
Congratulations! You won the prize!
OK

Bayes' Theorem Proof for Monty Hall problem



19MAT 205 CH.EN.U4CSE22029

ASSIGNMENT

Monty Hall Problem :-

In this problem with n doors ($n \geq 3$), where 1 door hides a prize and other hide goats, host reveals a goat behind 1 unchosen door after the contestant's initial selection, switching to the other unopened door doubles the probability of winning the prize.

Let we labelled all doors as 1 to n., let ~~door "k"~~ car be behind door "k".

$P(K) = 1/n$ (probability ~~to~~ for choosing car door)

Let C be another door which is opened according to rules

$\Rightarrow P(C) = \frac{1}{n-1}$

Now let we open door 1 and monty open door ~~"n"~~ "n".

$$\Rightarrow P(K=1 \mid C=n) = \frac{P(K=1) \cdot P(M=k \mid C=1)}{P(C=k)}$$

$$= \frac{1}{n} \cdot \frac{P(M=k \mid C=1)}{\frac{1}{n-1}} \Rightarrow \frac{1}{n} \times \frac{1/n-1}{\frac{1}{n-1}} = 1/n.$$

$P(C=k \mid m=k)$ (switching)

Assume we choose door 1 and monty opens door k.

$$P(C=k \mid m=k) = \frac{P(C=k) \cdot P(M=k \mid C=k)}{P(M=k)}$$

$$\Rightarrow \frac{1}{n} \cdot \frac{1}{\frac{1}{n-1}} = \frac{n-1}{n} = 1 - \frac{1}{n}.$$

Comparing both equation, we can see that switching leads to more winnings.