PYTHON PROGRAMMING INTERNSHIP PROJECT BY: SAHIL PRASAD

TASK 3 : MEMORY PUZZLE GAME

CODE:

```
import tkinter as tk
from tkinter import messagebox
import random


class MemoryGame:
    def __init__(self, root):
        self.root = root
        self.root.title("Memory Puzzle Game")


        self.time_limit = 60  # Total time in seconds
        self.time_left = self.time_limit
        self.first_card = None
        self.second_card = None
```

```
        self.lock = False
        self.buttons = []
        self.matches_found = 0

        # Symbols to be displayed on the cards
        self.symbols = ['🍏', '🍌', '🍇', '🍒', '🍉', '🍍', '🥝', '🍊']
        self.cards = self.symbols * 2
        random.shuffle(self.cards)

        # Timer display
        self.timer_label = tk.Label(root, text=f"Time Left: {self.time_left}s", font=("Arial", 14))
        self.timer_label.grid(row=0, column=0, columnspan=4, pady=10)

        self.create_board()
        self.start_timer()

        # Restart button
```

```
        restart_button = tk.Button(root,
text="Restart", command=self.restart_game,
font=("Arial", 12))
        restart_button.grid(row=5, column=1,
columnspan=2, pady=10)


    def create_board(self):
        """Creates a 4x4 grid of buttons (cards)."""
        for i in range(4):
            for j in range(4):
                button = tk.Button(
                    self.root,
                    text='',
                    font=("Arial", 24),
                    width=4,
                    height=2,
                    command=lambda idx=(i * 4 + j):
self.flip_card(idx)
                )
                button.grid(row=i + 1, column=j, padx=5,
pady=5)
```

```
                self.buttons.append(button)

    def flip_card(self, idx):
        """Handles card flip action."""
        if self.lock or self.buttons[idx]['state'] ==
'disabled':
            return

        self.buttons[idx].config(text=self.cards[idx])

        if not self.first_card:
            self.first_card = idx
        else:
            self.second_card = idx
            self.check_match()

    def check_match(self):
        """Checks if two flipped cards match."""
        self.lock = True
        if self.cards[self.first_card] ==
self.cards[self.second_card]:
```

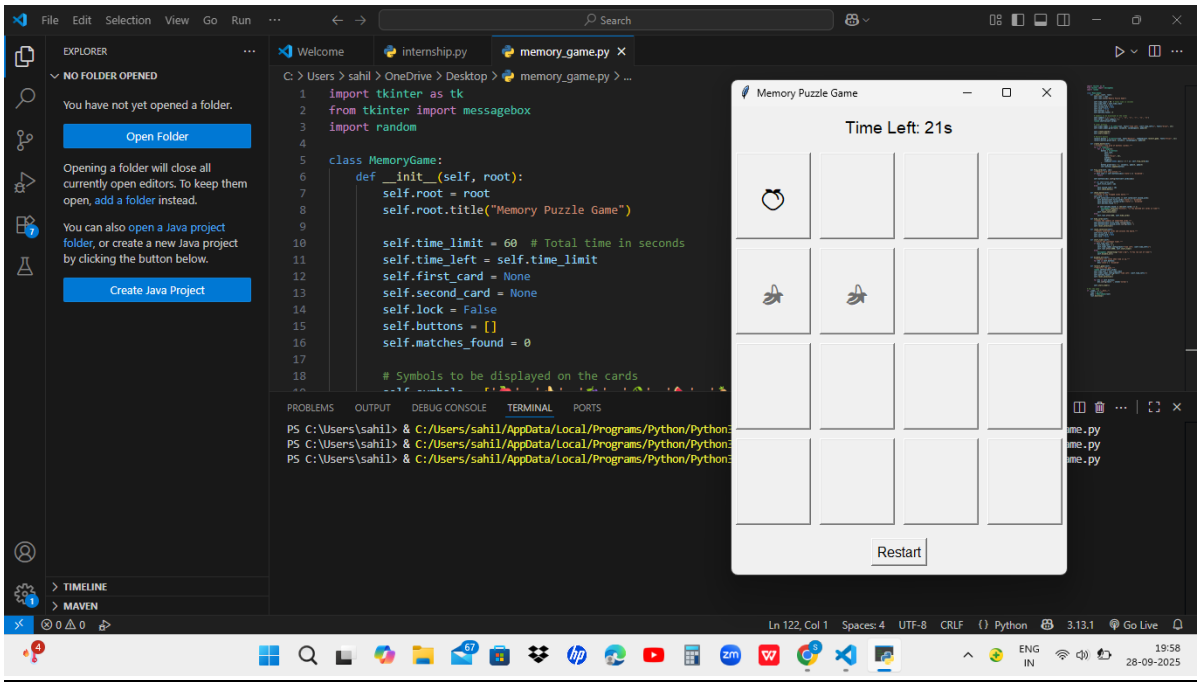
```
            self.buttons[self.first_card]['state'] =
'disabled'
            self.buttons[self.second_card]['state'] =
'disabled'
            self.matches_found += 1


            if self.matches_found == len(self.cards) //
2:
                messagebox.showinfo("Victory", " 🎉
You matched all cards in time!")
                self.restart_game()
            self.reset_selection()
        else:
            self.root.after(800, self.hide_cards)


    def hide_cards(self):
        """Hides the symbols of unmatched cards."""
        self.buttons[self.first_card].config(text='')
        self.buttons[self.second_card].config(text='')
        self.reset_selection()
```

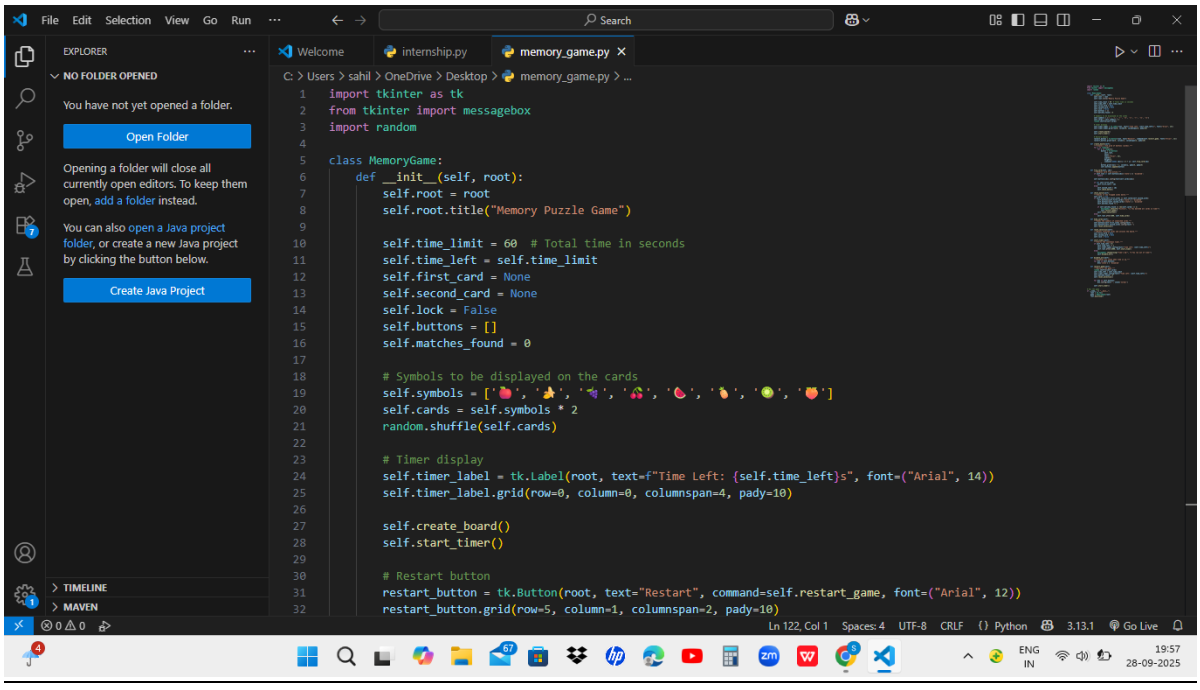
```
    def reset_selection(self):
        """Resets selected cards and unlocks the
board."""
        self.first_card = None
        self.second_card = None
        self.lock = False

    def start_timer(self):
        """Starts the countdown timer."""
        if self.time_left > 0:
            self.time_left -= 1
            self.timer_label.config(text=f"Time Left:
{self.time_left}s")
            self.root.after(1000, self.start_timer)
        else:
            messagebox.showwarning("Time's Up!",
" ⏳ You ran out of time!")
            self.disable_all()

    def disable_all(self):
        """Disables all cards when time is up."""
```

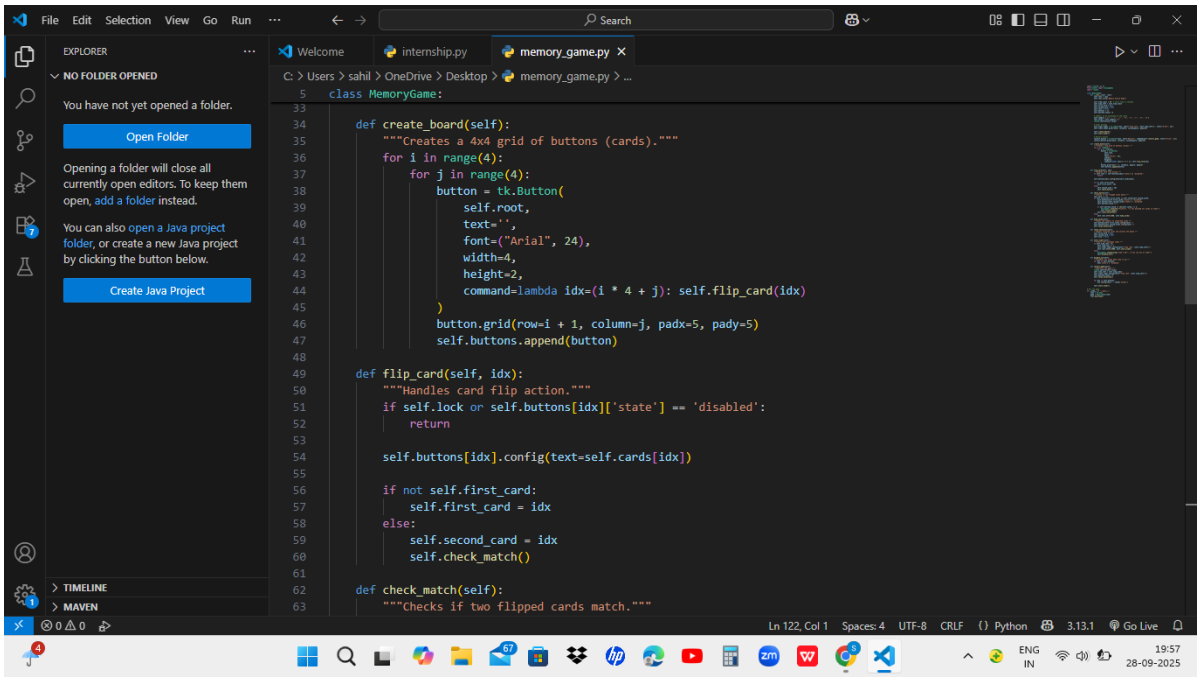
```
        for btn in self.buttons:
            btn['state'] = 'disabled'

    def restart_game(self):
        """Restarts the game."""
        random.shuffle(self.cards)
        self.time_left = self.time_limit
        self.timer_label.config(text=f"Time Left:
{self.time_left}s")
        self.matches_found = 0
        self.reset_selection()

        for btn in self.buttons:
            btn.config(text='', state='normal')

        self.start_timer()

# Run the game
if __name__ == "__main__":
    root = tk.Tk()
    game = MemoryGame(root)
```

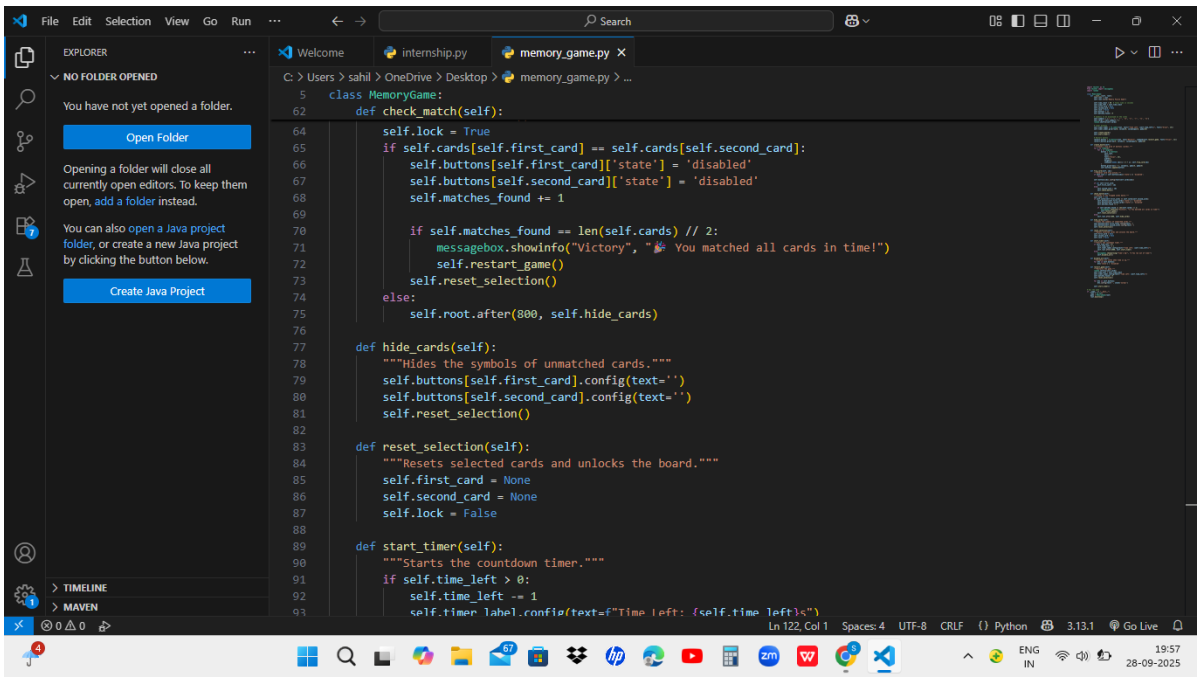
root.mainloop()

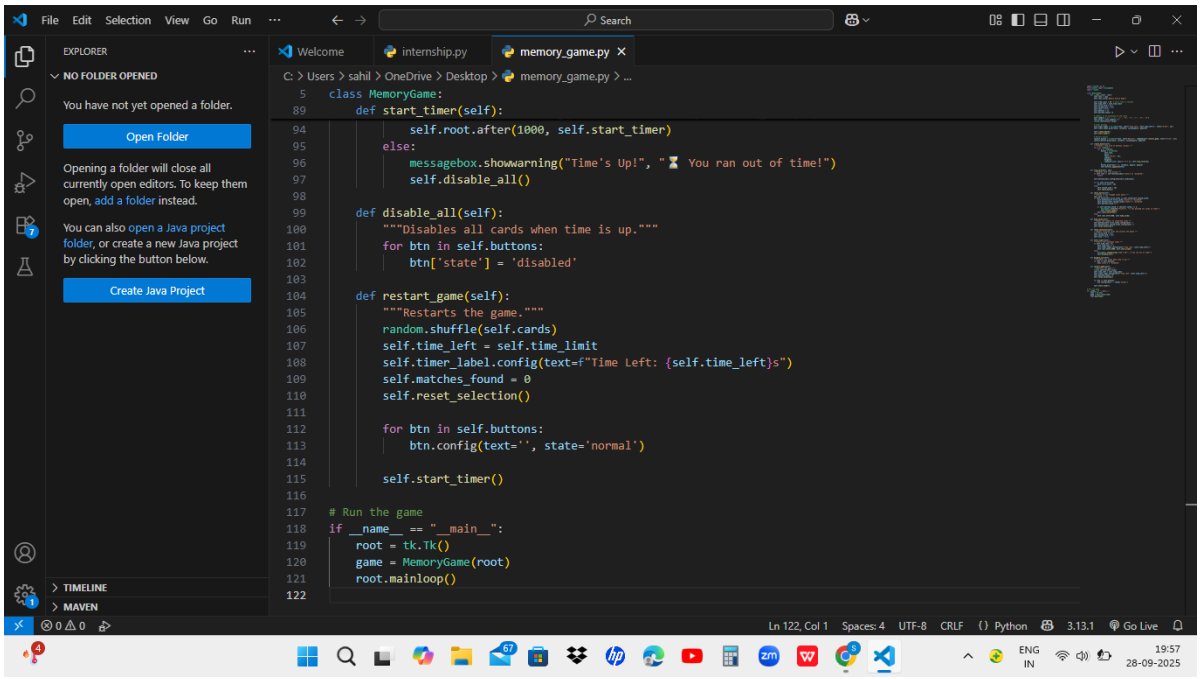
Output






File Edit Selection View Go Run
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can also open a Java project folder, or create a new Java project by clicking the button below.
Create Java Project
TIMELINE
MAVEN
Welcome
internship.py
memory_game.py
C: > Users > sahil > OneDrive > Desktop > memory_game.py > ...
class MemoryGame:
def create_board(self):
"""Creates a 4x4 grid of buttons (cards)."""
for i in range(4):
for j in range(4):
button = tk.Button(
self.root,
text='',
font=("Arial", 24),
width=4,
height=2,
command=lambda idx=(i * 4 + j): self.flip_card(idx)
)
button.grid(row=i + 1, column=j, padx=5, pady=5)
self.buttons.append(button)
def flip_card(self, idx):
"""Handles card flip action."""
if self.lock or self.buttons[idx]['state'] == 'disabled':
return
self.buttons[idx].config(text=self.cards[idx])
if not self.first_card:
self.first_card = idx
else:
self.second_card = idx
self.check_match()
def check_match(self):
"""Checks if two flipped cards match."""
Ln 122, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.1 Go Live
ENG IN
19:57
28-09-2025


File Edit Selection View Go Run
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can also open a Java project folder, or create a new Java project by clicking the button below.
Create Java Project
TIMELINE
MAVEN
Welcome
internship.py
memory_game.py
C: > Users > sahil > OneDrive > Desktop > memory_game.py > ...
class MemoryGame:
def check_match(self):
self.lock = True
if self.cards[self.first_card] == self.cards[self.second_card]:
self.buttons[self.first_card]['state'] = 'disabled'
self.buttons[self.second_card]['state'] = 'disabled'
self.matches_found += 1
if self.matches_found == len(self.cards) // 2:
messagebox.showinfo("Victory", "🎉 You matched all cards in time!")
self.restart_game()
self.reset_selection()
else:
self.root.after(800, self.hide_cards)
def hide_cards(self):
"""Hides the symbols of unmatched cards."""
self.buttons[self.first_card].config(text='')
self.buttons[self.second_card].config(text='')
self.reset_selection()
def reset_selection(self):
"""Resets selected cards and unlocks the board."""
self.first_card = None
self.second_card = None
self.lock = False
def start_timer(self):
"""Starts the countdown timer."""
if self.time_left > 0:
self.time_left -= 1
self.timer_label.config(text=f"Time Left: {self.time_left}s")
Ln 122, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.1 Go Live
ENG IN
19:57
28-09-2025


File Edit Selection View Go Run
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can also open a Java project folder, or create a new Java project by clicking the button below.
Create Java Project
TIMELINE
MAVEN
Welcome
internship.py
memory_game.py
C: > Users > sahil > OneDrive > Desktop > memory_game.py > ...
class MemoryGame:
def start_timer(self):
self.root.after(1000, self.start_timer)
else:
messagebox.showwarning("Time's Up!", "⌛ You ran out of time!")
self.disable_all()
def disable_all(self):
"""Disables all cards when time is up."""
for btn in self.buttons:
btn['state'] = 'disabled'
def restart_game(self):
"""Restarts the game."""
random.shuffle(self.cards)
self.time_left = self.time_limit
self.timer_label.config(text=f"Time Left: {self.time_left}s")
self.matches_found = 0
self.reset_selection()
for btn in self.buttons:
btn.config(text='', state='normal')
self.start_timer()
# Run the game
if __name__ == "__main__":
root = tk.Tk()
game = MemoryGame(root)
root.mainloop()
Ln 122, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.1 Go Live
ENG IN
19:57
28-09-2025