PRESENTED TO

MR. USMAN SHARIF

PRESENTED BY

SHAYAN FAISAL
46501

ABDUR RAFAY
44866

ALIAN RAFIQ
45969

DOCUMENTATION AND EXPLANATION OF THE CODE

Overview
The provided code is a simple memory-based card-matching game built using Python's *tkinter* library. The game involves flipping over cards to reveal food items, and the objective is to match all pairs of identical cards.

CODE ANALYSIS AND EXPLANATION

- Import Statements
  {import tkinter as tk import random from tkinter import messagebox}
  tkinter: A standard Python library used for creating graphical user interfaces.
  random: Provides the ability to shuffle cards randomly for the game.
  messagebox: A tkinter module used to display dialog boxes (e.g., showing a success message when the game ends).

- Global Variable: Food Items
  {FOODS = ["Pizza", "Burger", "Sushi", "Taco", "Pasta", "Donut", "Salad", "Steak"] }
- This is the list of food items that will appear on the cards. Each food item will have a matching pair, creating a total of **len(FOODS) * 2** cards.

- MemoryGame Class
  {class MemoryGame: def __init__(self, root): ... }
  Purpose: Represents the game logic and UI.
  Parameters:
  root: The main *tkinter* window object passed to the class.

- Initialization (__init__ Method)
  {self.root = root self.root.title("Food Matching Game") }
- Sets the title of the *tkinter* window.

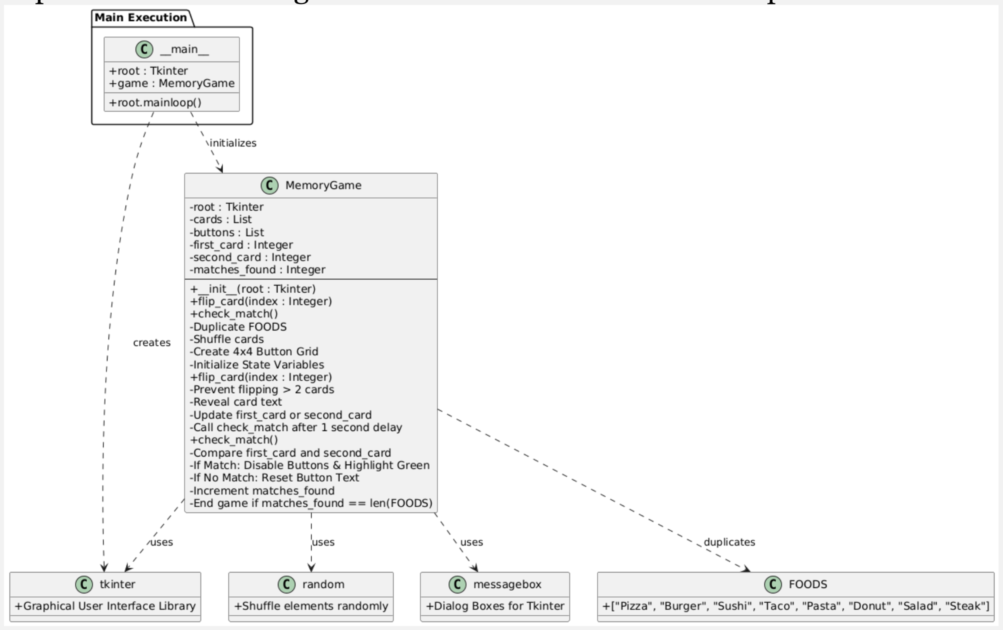
- Duplicating and Shuffling Cards
  {self.cards = FOODS * 2 random.shuffle(self.cards) }
  Logic:
  Duplicates the FOODS list to ensure each item has a matching pair.
  Shuffles the cards randomly to mix their positions.
- UI Components: Buttons
  {self.buttons = [] for i, card in enumerate(self.cards): button = tk.Button(root, text="", width=10, height=5, bg="lightblue", command=lambda i=i: self.flip_card(i)) button.grid(row=i // 4, column=i % 4) self.buttons.append(button)}
  Purpose: Creates a 4x4 grid of buttons representing the cards.
  Key Components:
  text="": Hides the card's content initially.
  command: Links each button to the flip_card method, passing the button's index.
  grid: Organizes the buttons into a grid based on their index.

- Game State Variables
  {self.first_card = None self.second_card = None self.matches_found = 0}
  Tracks the current state of the game:
    - **first_card and second_card:** Hold the indices of the currently selected cards.
    - **matches_found:** Counts the total number of matched pairs.

- Flipping a Card
  {def flip_card(self, index): if self.first_card is not None and self.second_card is not None: return # Wait for mismatched cards to reset button = self.buttons[index] if not button.cget("state") == "disabled": button.config(text=self.cards[index], state="disabled") if self.first_card is None: self.first_card = index elif self.second_card is None: self.second_card = index self.root.after(1000, self.check_match)}
  Purpose: Reveals a card's content and processes card-matching logic.
  Key Components:
  Prevents selecting more than two cards simultaneously.
  Reveals the card by setting its text to the corresponding food item.
  Updates first_card and second_card for tracking the currently flipped cards.
  Delays the check_match call by 1 second using after.

- Checking for a Match
  {def check_match(self): first_idx, second_idx = self.first_card, self.second_card if self.cards[first_idx] == self.cards[second_idx]: self.matches_found += 1 self.buttons[first_idx].config(bg="lightgreen") self.buttons[second_idx].config(bg="lightgreen") else: self.buttons[first_idx].config(text="", state="normal", bg="lightblue") self.buttons[second_idx].config(text="", state="normal", bg="lightblue") self.first_card = None self.second_card = None if self.matches_found == len(FOODS): messagebox.showinfo("Congratulations!", "You've matched all the cards!") self.root.quit()}
  Purpose: Checks whether the two selected cards match and updates the game state.
  Key Components:
  If the cards match:
  Updates their background color to green.
  Increments the match counter (matches_found).
  If they do not match:
  Resets the card text and re-enables the buttons.
  Ends the game when all matches are found, displaying a success message.

- Main Execution Block
  {if __name__ == "__main__": root = tk.Tk() game = MemoryGame(root) root.mainloop()}
- Purpose: Initializes the game and starts the tkinter event loop.


Main Execution
__main__
+root : Tkinter
+game : MemoryGame
+root.mainloop()
initializes
creates
MemoryGame
-root : Tkinter
-cards : List
-buttons : List
-first_card : Integer
-second_card : Integer
-matches_found : Integer
+__init__(root : Tkinter)
+flip_card(index : Integer)
+check_match()
-Duplicate FOODS
-Shuffle cards
-Create 4x4 Button Grid
-Initialize State Variables
+flip_card(index : Integer)
-Prevent flipping > 2 cards
-Reveal card text
-Update first_card or second_card
-Call check_match after 1 second delay
+check_match()
-Compare first_card and second_card
-If Match: Disable Buttons & Highlight Green
-If No Match: Reset Button Text
-Increment matches_found
-End game if matches_found == len(FOODS)
uses
uses
uses
duplicates
tkinter
+Graphical User Interface Library
random
+Shuffle elements randomly
messagebox
+Dialog Boxes for Tkinter
FOODS
+["Pizza", "Burger", "Sushi", "Taco", "Pasta", "Donut", "Salad", "Steak"]

·

COMPLEXITY ANALYSIS

Time Complexity

Card Shuffle: O(n) where n is the total number of cards.

Flip Card:

Revealing a card: O(1).

Checking for a match: O(1) per comparison.

Total flips/comparisons: O(n) since each card is flipped at most once.

Overall: The game involves shuffling and matching, resulting in **O(n)** time complexity.

Space Complexity

Card Storage: O(n) for the self.cards list.

Button List: O(n) for self.buttons.

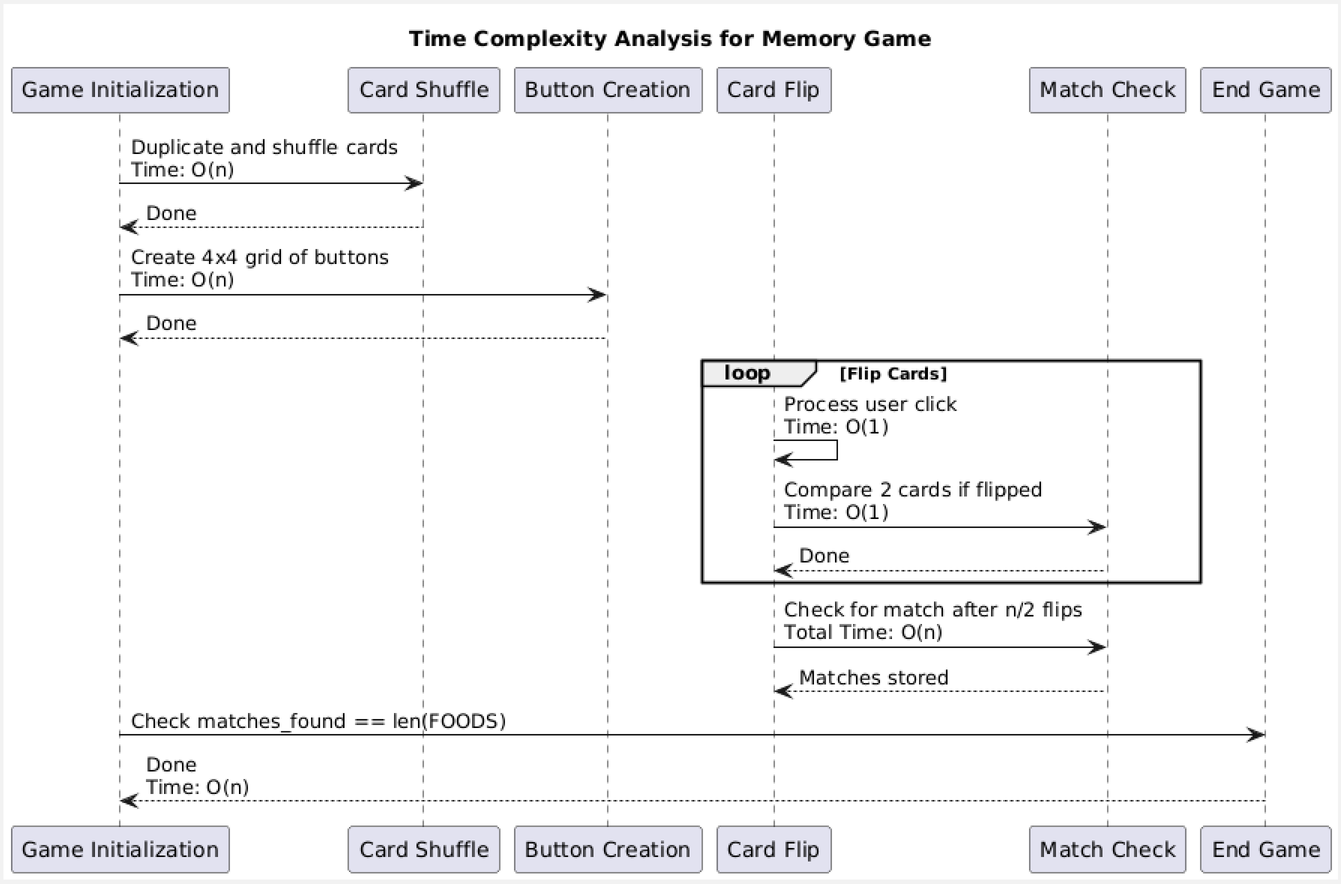
Game State Variables: O(1).

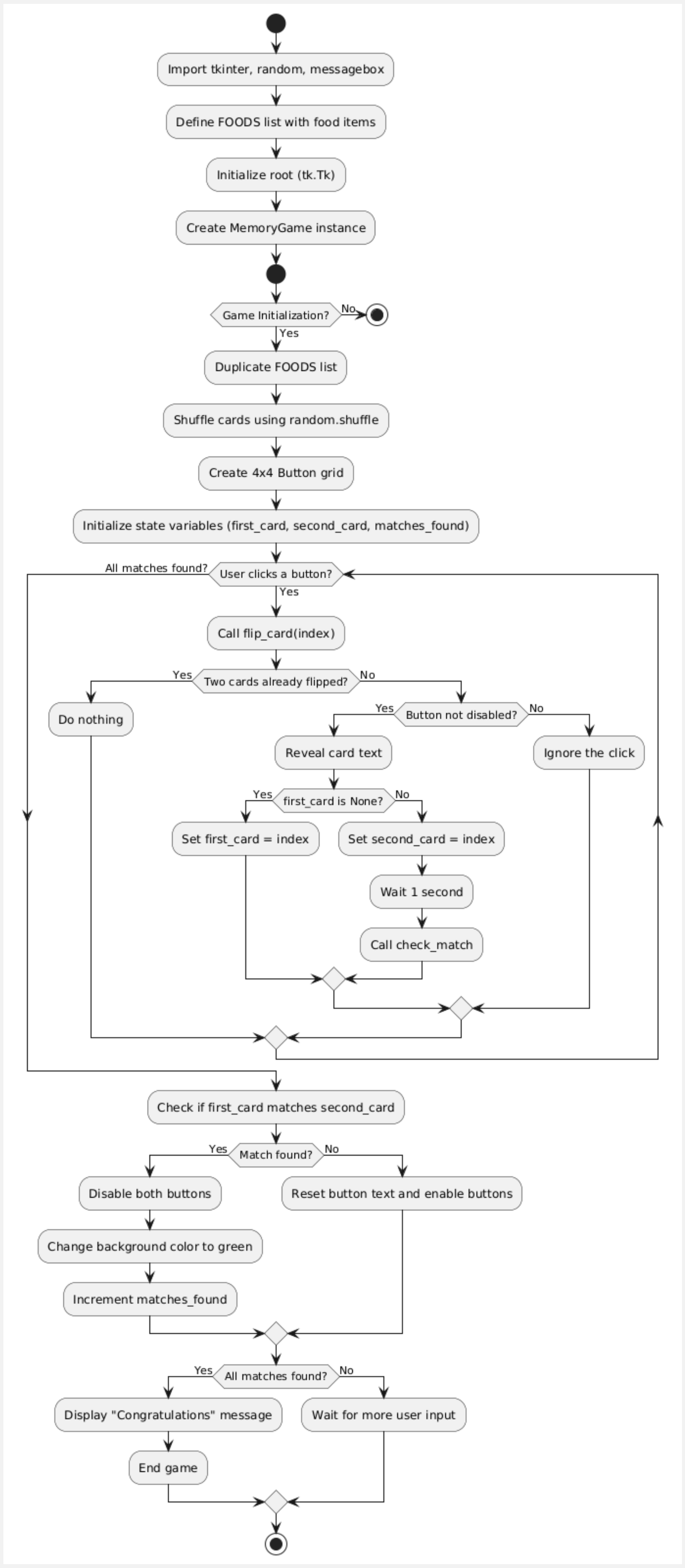
Overall: Space complexity is O(n).

Overall Complexity

- Time Complexity: O(n)
- Space Complexity: O(n)

The algorithm is efficient for a small grid size but can scale linearly with the number of cards.




Import tkinter, random, messagebox
Define FOODS list with food items
Initialize root (tk.Tk)
Create MemoryGame instance
Game Initialization?
No
Yes
Duplicate FOODS list
Shuffle cards using random.shuffle
Create 4x4 Button grid
Initialize state variables (first_card, second_card, matches_found)
All matches found?
User clicks a button?
Yes
Call flip_card(index)
Yes
Two cards already flipped?
No
Do nothing
Yes
Button not disabled?
No
Reveal card text
Ignore the click
Yes
first_card is None?
No
Set first_card = index
Set second_card = index
Wait 1 second
Call check_match
Check if first_card matches second_card
Yes
Match found?
No
Disable both buttons
Reset button text and enable buttons
Change background color to green
Increment matches_found
Yes
All matches found?
No
Display "Congratulations" message
Wait for more user input
End game