MemPal – Memory App

T1A3

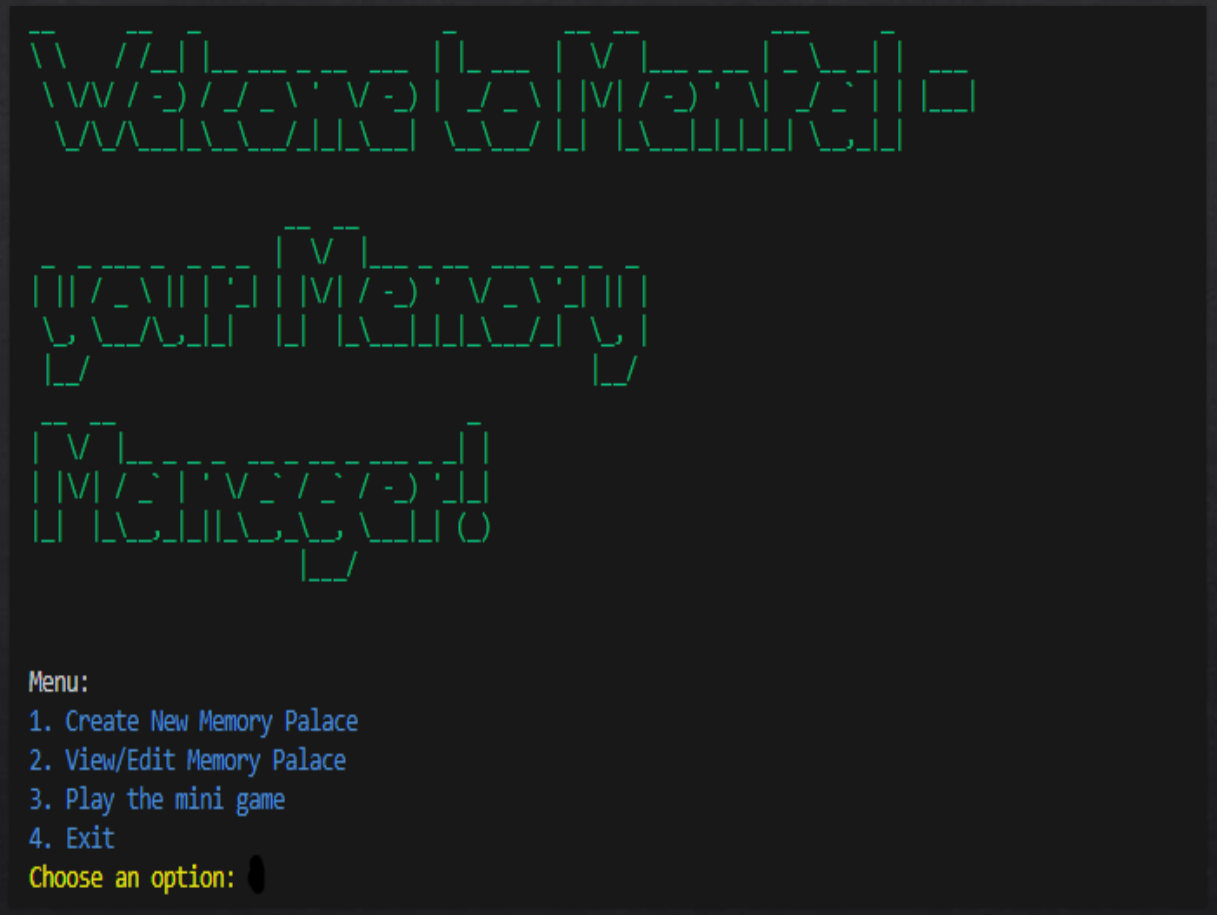
Xuan Lap (Leo) Tran

Topics

- A walk-through of the terminal application, its features and how it is used.
  - Introduction
  - The problem it solves
- A walk-through of the logic of the terminal application and code.
  - Walk-through of the Application - how to navigate the app
  - Key features: creating a new Memory Palace, viewing/editing a Memory Palace, and playing the mini game.
  - Code Walk-through
- A review of the development/build process including challenges, ethical issues, favourite parts, etc.
  - Overview of the development process
  - Challenges, ethical issues, favourite parts

Introduction

- MemPal is a terminal application that allows users to create and manage their own memory palaces.
- A memory palace is a memory technique that involves associating information with specific locations in a familiar environment.
- The problem is that for memory palace user, they don’t have a tool to store their memory palaces, and to test their memory palace with a simple game app. MemPal is the one to do just those 3 problems.



Walk-through of the Application

- how to navigate the app
- Key features: creating a new Memory Palace, viewing/editing a Memory Palace, and playing the mini game.
- Code Walk-through

src > main.py > menu

```python
from functions import add_mempal, create_mempal, view_edit_mempal, minigame
from colorama import init, Fore, Style
from pyfiglet import figlet_format
from colored import fg, bg, attr

init()

file_name = "Mempal.csv"

try:
    mempal_file = open(file_name, "r")
    mempal_file.close()
    print("In try block")
except FileNotFoundError:
    mempal_file = open(file_name, "w")
    mempal_file.write("")
    mempal_file.close()
    print("In except block")

def menu():
    print(Fore.GREEN + figlet_format("Welcome to MemPal - your Memory Manager!", font="small") + Style.RESET_ALL)
    print("\nMenu:")
    print(Fore.BLUE + "1. Create New Memory Palace" + Style.RESET_ALL)
    print(Fore.BLUE + "2. View/Edit Memory Palace" + Style.RESET_ALL)
    print(Fore.BLUE + "3. Play the mini game" + Style.RESET_ALL)
    print(Fore.BLUE + "4. Exit" + Style.RESET_ALL)
    choice = input('%s%s%s' % (fg('yellow'), "Choose an option: ", attr('reset')))
    return choice

users_choice = ""

while users_choice != "4":
    users_choice = menu()
    if (users_choice == "1"):
        create_mempal(file_name)
    elif (users_choice == "2"):
        view_edit_mempal(file_name)
    elif (users_choice == "3"):
        minigame(file_name)
    elif (users_choice == "4"):
        break
    else:
        print("Invalid Input")

print(Fore.YELLOW + figlet_format("THANK YOU FOR USING MEMPAL!!", font="small") + Style.RESET_ALL)

```

Add and ‘Create New’ function

```python
import csv
import random
from colorama import Fore, Style, init
from pyfiglet import figlet_format
from colored import fg, bg, attr

init()

def add_mempal(existing_loci=None):
    loci = existing_loci if existing_loci else []
    while True:
        locus = input(f"Locus {len(loci) + 1}?: ")
        loci.append(locus)

        while True:  # Add this loop to keep asking until a valid choice is made
            choice = input("Choose: 'Add next'(y) or 'Finish'(n): ")

            if choice.lower() == "n":
                return loci
            elif choice.lower() == "y":
                break  # Break the inner loop if a valid choice is made
            else:
                print("Invalid choice. Please enter 'y' or 'n'.")
                continue  # Continue the inner loop if an invalid choice is made
```

```python
def create_mempal(file_name):
    print(Fore.GREEN + figlet_format("Creating New Memory Palace", font="small") + Style.
    RESET_ALL)
    mempal_name = input("Enter the name of this Memory Palace: ")
    loci = add_mempal()

    with open(file_name, "a", newline="") as f:
        writer = csv.writer(f)
        writer.writerow([mempal_name] + loci + [0, 0])  # Initialize score and average score
        to 0
    print("Memory Palace created successfully!")
```

View and Edit Function

```
def view_edit_mempal(file_name):
    print(Fore.GREEN + figlet_format("Viewing/Editing Memory Palace", font="small") + Style.
    RESET_ALL)
    try:
        with open(file_name, "r") as f:
            reader = csv.reader(f)
            memory_palaces = list(reader)

        if len(memory_palaces) == 0:
            print("No Memory Palaces found.")
            return

        # Sort the memory palaces by score
        memory_palaces.sort(key=lambda x: int(x[-1]), reverse=True)

        for i, palace in enumerate(memory_palaces):
            print(f"{Fore.GREEN}{i+1}. {palace[0]} - Score: {palace[-1]}%{Style.RESET_ALL}")

        while True:  # Add this loop to keep asking until a valid choice is made
            try:
                palace_index = input('%s%s%s' % (fg('red'), "Select a Memory Palace (number) or 'b'
                to go back: ", attr('reset')))
                if palace_index.lower() == 'b':
                    return
                palace_index = int(palace_index) - 1
                break  # Break the loop if a valid choice is made
            except ValueError:
                print("Invalid input. Please enter a number or 'b' to go back.")
                continue # Continue the loop if an invalid choice is made

        print(f"Memory Palace: {memory_palaces[palace_index][0]}")
        print("Loci:")
        for i, locus in enumerate(memory_palaces[palace_index][1:-2], start=1):  # Exclude the name
        and scores
            print(f"{i}.{locus}")
        while True:  # Add this loop to keep asking until a valid choice is made
            try:
                choice = input('%s%s%s' % (fg('blue'), "Choose: 'Add next'(y), 'Edit'(e), 'Delete'
                (d), 'Remove Memory Palace'(r), or 'Finish'(n): ", attr('reset')))

                if choice.lower() == "n":
                    break
                elif choice.lower() == "y":
                    new_loci = add_mempal(memory_palaces[palace_index][1:-2])
                    memory_palaces[palace_index] = [memory_palaces[palace_index][0]] + new_loci +
                    memory_palaces[palace_index][-2:]
                elif choice.lower() == "e":
                    locus_index = int(input("Enter the number of the locus to edit: "))
                    if 1 <= locus_index < len(memory_palaces[palace_index]) - 2:  # Check if the
                    index is within the valid range
                        new_locus = input("Enter the new locus: ")
                        memory_palaces[palace_index][locus_index] = new_locus
                    else:
                        print("Invalid locus number. Please enter a valid locus number.")
                elif choice.lower() == "d":
                    locus_index = int(input("Enter the number of the locus to delete: "))
                    if 1 <= locus_index < len(memory_palaces[palace_index]) - 2:  # Check if the
                    index is within the valid range
                        del memory_palaces[palace_index][locus_index]
                    else:
                        print("Invalid locus number. Please enter a valid locus number.")
                elif choice.lower() == "r":
                    del memory_palaces[palace_index]
                    print("Memory Palace removed successfully!")
                    break
                else:
                    print("Invalid choice. Please enter 'y', 'e', 'd', 'r', or 'n'.")
                    continue
            except Exception as e:
                print(f"An error occurred: {e}")
                continue  # Continue the loop if an invalid choice is made

        # Save the changes to the file
        with open(file_name, "w", newline="") as f:
            writer = csv.writer(f)
            writer.writerows(memory_palaces)
    except Exception as e:
        print(f"An error occurred: {e}")
```

Minigame function

```python
def minigame(file_name):
    try:
        with open(file_name, "r") as f:
            reader = csv.reader(f)
            memory_palaces = list(reader)

        if len(memory_palaces) == 0:
            print("No Memory Palaces found.")
            return

        # Sort the memory palaces by score
        memory_palaces.sort(key=lambda x: int(x[-1]), reverse=True)

        print(Fore.GREEN + figlet_format("Welcome! to MemPal minigame", font="speed") + Style.RESET_ALL)

        for i, palace in enumerate(memory_palaces):
            print(f"{Fore.GREEN}{i+1}. {palace[0]} - Score: {palace[-1]}%{Style.RESET_ALL}")

        while True:  # Add this loop to keep asking until a valid choice is made
            try:
                palace_index = input('%s%s%s' % (fg('yellow'), "Select a Memory Palace (number) to play or 'b' to go back: ", attr('reset')))
                if palace_index.lower() == 'b':
                    return
                palace_index = int(palace_index) - 1
                break  # Break the loop if a valid choice is made
            except ValueError:
                print("Invalid input. Please enter a number or 'b' to go back.")
                continue  # Continue the loop if an invalid choice is made

        loci = memory_palaces[palace_index][1:-2]  # Exclude the name and scores
        questions_count = int(len(loci) * 0.75)  # Change to 75% of the total number of loci
        questions = random.sample(loci, questions_count)

        correct_answers = 0
        for question in questions:
            answer = input(f"What is the item at locus {loci.index(question) + 1}?: ")
            if answer == question:
                correct_answers += 1

        score = int((correct_answers / questions_count) * 100)
        print(f"Your score: {score}%")

        # Update the score in the file
        previous_score = int(memory_palaces[palace_index][-2])  # Get the previous score
        average_score = int((previous_score + score) / 2)  # Calculate the average of the recent score and the previous score
        memory_palaces[palace_index] = memory_palaces[palace_index][:1] + loci + [score, average_score]
        with open(file_name, "w", newline="") as f:
            writer = csv.writer(f)
            writer.writerows(memory_palaces)
    except Exception as e:
        print(f"An error occurred: {e}")
```

Overview of the development process

1. Start with the idea
2. Then the ideal features
3. Make use of Trello
4. Start coding
    a. Folder/directory structure: follow the basic requirements of Docs, PPT and SRC.
    b. Create a simple Readme file with simple framework to follow/update. And create a simple Implementation plan to follow/update.
    c. Start coding with menu, file name
    Testing

CODE REQUIREMENTS

- variables and variable scope
- loops and conditional control structures
- write and utilise simple functions
- error handling
- input and output
- importing a Python package
- using functions from a Python package
- Apply DRY (Don't Repeat Yourself) coding principles

Challenges, ethical issues, and favorite parts

- The hardest part is the testing: a lot of research/reading since the app has became complicated. Not just Pytest was used but also manual tests were done to speed up the process.
- The most enjoying part is the idea/implementation plan, where you brainstorm what features could be in the app.
- The favorite part is the coding itself, start small, from print menu to fixing function codes.

- I am aware of and acknowledge the ethical principles that guide research and the use of ChatGPT. I understand that there are potential risks of plagiarism and privacy violation when using AI-generated code.
- However, 90-95% of the codes have been done without the assistance from ChatGPT. ChatGPT was used for lambda sorting code, styling, explaining note for codes #, testing test 3, 1 out of for 4 tests, and most importantly csv sorting/indexing for score calculation as I was stuck with problems and issues.
- All researches done through search engine and ChatGPT have been done carefully and purely for the purpose of obtaining deeper understanding, knowledge, skills and for assistance with understanding the problem and issue when hitting a road block.

That is the end of my presentation!
I hope you enjoy, and
Thank you for your time!