Project 5: Adventure Story Game (Expanded)

Project Explanation: The expansion of my earlier text-based adventure game introduces several new features to enhance the player experience. Firstly, the narrative has been enhanced and offers a more detailed backstory about the mansion and the mysterious disappearance of its family. This enhancement aims to immerse players more deeply into the game world, fostering a sense of intrigue and suspense. Additionally, the gameplay mechanics have been expanded to include the exploration of various rooms within the mansion, each presenting unique challenges and rewards. This adds depth to the gameplay and provides players with a greater sense of exploration and discovery. Moreover, a new feature has been added for statistical analysis, allowing the game to track and display key metrics such as the average time spent exploring and the maximum time spent exploring. This addition allows players to reflect on their progress and performance throughout their adventure.

Expected Inputs/Outputs: In the game's opening, players are invited to introduce themselves by entering their name. Next, they're presented with a pivotal decision: to reveal their motive for venturing into the mansion, whether it be for sheer "fun" or in pursuit of the elusive "hidden treasure." As they progress, players engage with their surroundings by selecting their path forward, deciding which room to explore next or how to navigate encounters with the unknown. At any point, players have the option to either delve deeper into the mystery or conclude their adventure. Throughout the game, vivid scenes are painted, guiding players on their immersive journey through the mansion's eerie corridors. At pivotal moments, players are prompted to make choices, shaping the course of their adventure. As for the expected outputs, players can anticipate receiving text-based descriptions of the mansion and its surroundings, providing immersive imagery and context for their exploration. Messages indicating the outcomes of player choices, whether they result in success or failure, dynamically shape the storyline and impact subsequent gameplay. Furthermore, players can expect an inventory of items found during exploration, allowing them to keep track of their discoveries and utilize them strategically. Finally, statistical data on gameplay, such as the average time spent exploring, offers insights into the player's progress and performance, enhancing their overall gaming experience.

Computing Concepts Utilized: Several computing concepts are employed in this adventure game:

- User Input: Players provide input by entering their name and stating their purpose for entering the mansion.
- File I/O: The program reads data from and writes data to CSV files to store player information and game statistics.
- Decision Structures: (if-elif-else statements) to evaluate the player's choices and determine actions/ outcomes.
- Loops: While loops are used for continuous interaction.
- Randomization: The game incorporates random number generation to introduce variability.
- Functions: Functions are defined to encapsulate specific actions or behaviors, such as exploring different areas of the mansion or generating random items.
- String Manipulation: String methods like strip() are utilized to process user input to ensure consistency.
- Error Handling: Basic error handling is implemented to manage invalid user inputs.
- Modularity: The code is organized into functions, each responsible for a specific aspect of the game. This modular approach enhances the readability, maintainability, and reusability of code segments.
- Conditional Statements: used to evaluate conditions and determine the flow of the game

Limitations: Interaction with the game world is primarily limited to selecting predefined choices or options presented by the game. Players may desire more freedom to explore or interact with the environment in

different ways. The game is designed for a single-player, limiting opportunities for multiplayer interaction or cooperative gameplay.

Reflection: Expanding on my earlier adventure game was fun but tougher than expected. I tried to incorporate more try-except functions to catch errors because my code got lengthy. This was my most in-depth project. I had to do a lot of deleting and readjusting code to get items to flow to my CSV correctly. Additionally, I aimed to maintain proper code readability so I added plenty of comments which also added to the length.

Three (3) examples of program use – including input and output.

```
You've entered the Haunted Mansion. State your name: Kilandra
Welcome, Kilandra to the Haunted Mansion, an estate that has been abandoned for 100 y
ears! You must be brave to dare to enter.

You found some items:
- map
- key
- riddle scroll
You enter a mysterious room...
What has brought you here, Kilandra? Fun
Maybe you will solve the mystery!
As you step into a dimly lit hallway, you see a large metal door, while you also spy
a creaking staircase to your right, leading up to the attic. To the left of the door
is an entrance that leads to the dark, eerie basement.

What do you want to inspect?[basement/staircase/secret door] 
```

```
What do you want to inspect?[basement/staircase/secret door] basement

You walk downstairs to the basement and the lights start flickering.
An unknown voice says:
"Hey, you're trespassing!"
A booby trap goes off and you're suddenly trapped in the basement.
You lose Kilandra, you're considered a thief and have been caught!
Do you want to play again? [yes/no] 
```

```
What do you want to inspect?[basement/staircase/secret door] secret door

You open the secret door and find yourself in an old, dusty library.
You grab a shiny book and it triggers a hidden passage to emerge leading to a small c
hamber.
You walk in the chamber and see a treasure chest...
You manage to solve the puzzle on the treasure chest easily.
Congratulations, Kilandra, you've found the hidden treasure!
You win!
You also find a map and a compass in the chamber!
Writing data to CSV.
Data history has been successfully written to the CSV file.
Items found so far: ['hidden treasure', 'map', 'compass']
Do you want to play again? [yes/no] 
```

Project 5: Adventure Story Game (Expanded)

Python Code:

```python
import random
import csv
import time
import sys
import statistics
import datetime

delay = 1.0  # For pauses for dramatic effect.

def read_data_history():
  data = []
  try:
    with open("P5_BassKilandra.csv", "r") as csvfile:
      reader = csv.DictReader(csvfile)
      for row in reader:
        data.append(row)
  except FileNotFoundError:
    print("P5_BassKilandra file not found. Creating a new one.")
  return data

def write_data_history(data_history, time_spent):
  print("Writing data to CSV.")
  fieldnames = ["Name", "Purpose", "Found Treasure", "Escape Status", "Seconds Spent Exploring"]

  try:
    #Read existing data from CSV
    existing_data = read_data_history()

    #Extract existing names
    existing_names = {entry["Name"] for entry in existing_data}

    #Filter out entries already present in the CSV
    new_data = [entry for entry in data_history if entry["Name"] not in existing_names]

    #Write only new data to CSV
    with open("P5_BassKilandra.csv", "a", newline="") as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        if len(existing_data) == 0:  # Write header if file is empty
            writer.writeheader()

        for entry in new_data:
          #Format seconds spent exploring as minutes and seconds
          entry["Seconds Spent Exploring"] = time_spent
          writer.writerow(entry)

    print("Data history has been successfully written to the CSV file.")

  except Exception as e:
```

Project 5: Adventure Story Game (Expanded)

```
        print("An error occurred while writing data to the CSV file:", e)

def calculate_time_spent_exploring():
    start_time = time.time() #Record the start time
    #Wait for player to finish exploring
    input("Press Enter when you finish exploring: ")
    end_time = time.time() #Record the end time
    time_spent = end_time - start_time #Calculate the time spent exploring
    return time_spent

def print_statistics(data_history):
    print("Calculating statistics with data:")
    #Extract time spent exploring data from data_history
    time_spent_data = []
    for row in data_history:
        if "Seconds Spent Exploring" in row:
            try:
                time_spent_str = row["Seconds Spent Exploring"]
                time_spent_data.append(float(time_spent_str))

            except ValueError:
                #Handle the case where the value cannot be converted to datetime
                pass

    if time_spent_data: #Check if time_spent_data is not empty
        #Calculate mean average for time spent exploring
        mean_seconds = round(statistics.mean(time_spent_data), 4)
        mean_minutes, mean_seconds = divmod(mean_seconds, 60)
        #divmod returns quotient and remainder

        #Calculate maximum value for time spent exploring
        max_seconds = max(time_spent_data)
        max_minutes, max_seconds = divmod(max_seconds, 60)
        #divmod returns quotient and remainder

        print("Statistics for Seconds Spent Exploring:")
        print("Mean Average:", f"{int(mean_minutes):02}:{int(mean_seconds):02}")
        print("Maximum Value:", f"{int(max_minutes):02}:{int(max_seconds):02}")
    else:
        print("No valid data points found for 'Seconds Spent Exploring'.")

def introduction(): #intro backstory of the mansion
  print("Over 100 years ago, a wealthy family lived in a mansion in the middle of the woods.", end="")
  time.sleep(delay)
  print(" Legend has it that, one day the family went missing, and now their ghosts occupy the space. No one has determined what happened to the family, but there's a theory that they went into hiding.")

def generateRandomItems(num_items):
  items = ['key', 'potion', 'map', 'compass', 'torch', 'riddle scroll'] #generate random items for user to find
  random_items = [random.choice(items) for _ in range(num_items)]
  return random_items

#Explore Mansion and give the user choices
def exploreMansion(username, found_items, purpose, data_history, data_to_write):
  print("As you step into a dimly lit hallway, you see a large metal door, while you also spy a creaking staircase to your right, leading up to the attic. To the left of the door is an entrance that leads to the dark, eerie basement.")
  time.sleep(delay)
  print()
  start_time = time.time()  # Record the start time
  while True:
    choice1 = input("What do you want to inspect?[basement/staircase/secret door] ").strip()  #Choice 1
    print()
    # Decision structure . Evaluates choice for the three options presented.
    if choice1 == "basement":  # First option
      exploreBasement(username, found_items, purpose, data_history, data_to_write)
    elif choice1 == "staircase":
      exploreStaircase(username, found_items, purpose, data_history, data_to_write)
      break
    elif choice1 == "secret door":
      exploreSecretDoor(username, found_items, purpose,data_history, data_to_write)
      break
    else:
      print("Invalid input. Please try again.")
```

```
    end_time = time.time() #Record the end time
    time_spent = end_time - start_time  #Calculate the time spent exploring

    data_to_write.append({"Name": username,
      "Purpose": purpose,
      "Found Treasure": "Yes" if "hidden treasure" in found_items else "No",
      "Escape Status": "Won" if username != "" else "Lost",
      "Seconds Spent Exploring": time.strftime("%Y-%m-%d %H:%M:%S")})
    # Call write_data_history at the end to write accumulated data
    write_data_history(data_to_write, time_spent)

    print("Items found so far:", found_items)


def exploreBasement(username, found_items, purpose, data_history,data_to_write): #Explore the basement
    thief = True
    print("You walk downstairs to the basement and the lights start flickering.")
    time.sleep(delay)
    if thief:  #Use boolean
      print("An unknown voice says: \n\"Hey, you're trespassing!\"")
      print("A booby trap goes off and you're suddenly trapped in the basement.")
      print("You lose {}, you're considered a thief and have been caught!".format(username))
      continue_exploring(username, found_items, purpose, data_history, data_to_write)  #Continuation loop
      sys.exit()
    else:
```

```
      #Find a random item
      found_item = random.choice(generateRandomItems(1))
      found_items.append(found_item)
      print("You find a {} in the basement.".format(found_item))
      print("You continue exploring the basement.")

      continue_exploring(username, found_items, purpose, data_history, data_to_write)

def exploreStaircase(username, found_items, purpose, data_history, data_to_write): #Explore the staircase
    print("You walk up the staircase with your heart pounding from fear.")
    print()
    time.sleep(delay)  # Another use of the dramatic pause feature
    print("Wind from an open window blows your lantern out and you end up lost in the mansion with no way out.")
    print()
    print("You lose, {}, you're considered a thief and have been caught!".format(username))
    continue_exploring(username, found_items, purpose, data_history, data_to_write)  #Continuation loop
    sys.exit()

def exploreSecretDoor(username, found_items, purpose, data_history, data_to_write): #Explore secrect door
    thief = False
    print("You open the secret door and find yourself in an old, dusty library.")
    time.sleep(delay)
    print("You grab a shiny book and it triggers a hidden passage to emerge leading to a small chamber.")
    time.sleep(delay)
    print("You walk in the chamber and see a treasure chest...")
```

Project 5: Adventure Story Game (Expanded)

```
    time.sleep(delay)
    puzzle_difficulty = random.randint(1, 10) #Simulate puzzle difficulty with a random number
    if puzzle_difficulty <= 5: #Higher probability of success for lower difficulty
        print("You manage to solve the puzzle on the treasure chest easily.")
        print("Congratulations, {}, you've found the hidden treasure!".format(username))
        print("You win!")
        found_items.append("hidden treasure")  #Add the found item to the list
    else:
        print("The puzzle on the treasure chest seems too complex to solve.")
        print("You couldn't unlock the treasure chest.")
        print("Better luck next time, {}!".format(username))

    #After solving the puzzle, the player may find additional items
    print("You also find a map and a compass in the chamber!")
    found_items.extend(["map", "compass"])  # Add the found items to the list

def exploreItems(found_items): #Print items found during the game
    print("You found the following items during your exploration:")
    for item in found_items:
        print("-", item)

#Allow users to continue exploring the mansion or exit the game
def continue_exploring(username, found_items, purpose, data_history, data_to_write):
    while True:
        choice = input("Do you want to play again? [yes/no] ").lower()
```

```
        if choice =="yes":
            found_items.clear()
            exploreMansion(username, found_items, purpose, data_history, data_to_write)
        elif choice == "no":
            print("Thanks for playing, {}. Goodbye!".format(username))
            data_history.append({"Name": username,
                                 "Purpose": purpose,
                                 "Found Treasure": "Yes" if "hidden treasure" in found_items else "No",
                                 "Escape Status": "Won" if username != "" else "Lost",
                                 "Seconds Spent Exploring": time.strftime("%Y-%m-%d %H:%M:%S")})
            print_statistics(data_history)  #Calculate statistics
            return False  #Signal to end the game
        else:
            print("Invalid input. Please enter 'yes' or 'no'.")

# Novel function: Calculate the average length of usernames
def calculate_average_username_length(data_history):
    total_length = sum(len(entry["Name"]) for entry in data_history)
    average_length = total_length / len(data_history) if len(data_history) > 0 else 0.0
    return average_length

def createUpdatedDataHistory(newData):
    # Read existing data from CSV
    existing_data = read_data_history()
```

```

    # Combine existing data with new data
    updated_data = existing_data + newData

    # Write updated data to a new CSV file
    fieldnames = ["Name", "Purpose", "Found Treasure", "Escape Status"]
    with open("P5_outputFile.csv", "w", newline="") as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerows(updated_data)
    print("Updated data history has been written to P5_outputFile.csv.")

def main():
    data_history = read_data_history() #Read data history first
    introduction()
    print()

    print_statistics(data_history)

    #User's name
    username = input("You've entered the Haunted Mansion. State your name: ")
    print("Welcome,", username, "to the Haunted Mansion, an estate that has been abandoned for 100 years! You must be brave to dare to enter.")
    print()
    #  Find starting items
    found_items = generateRandomItems(3)  #Adjust number if needed
```

```
    print("You found some items: ")
    for item in found_items:
      print("- " + item)
    #  Player encounters a room where they may find random items
    print("You enter a mysterious room...")
    time.sleep(delay)

    purpose = input("What has brought you here, " + username + "? ") #User's purpose

  #Decision structure 1. Evaluates purpose variable for two specific options. Also has a default response
    if purpose.lower() == "fun":
      print("Maybe you will solve the mystery!")
    elif purpose == "hidden treasure":
      print("I hear there is a treasure hidden somewhere in the mansion.")
    else:  #For any response other than "fun" or "hidden treasure"
      print("Well, I hope that you find what you're looking for.")
      print("It is time for you to enter the mansion!\n\n")

    found_items = []
    data_to_write = []   #Create an empty list to store data for writing
    exploreMansion(username, found_items, purpose, data_history,   data_to_write)
    exploreItems(found_items) #Prints items found

    print("Data read from CSV.")
```

```


      #  Explore mansion and append data to write after each exploration
      exploreMansion(username, found_items, purpose, data_history, data_to_write)

      # Call the novel function to calculate average username length
      average_username_length = calculate_average_username_length(data_history)
      print("Average Username Length:", average_username_length)

      if not continue_exploring(username, found_items, purpose, data_history, data_to_write):
          print_statistics(data_history)  #Add this line to print statistics
          return



  if __name__ == "__main__":
    main()
```