Event Management System

Course Project Report

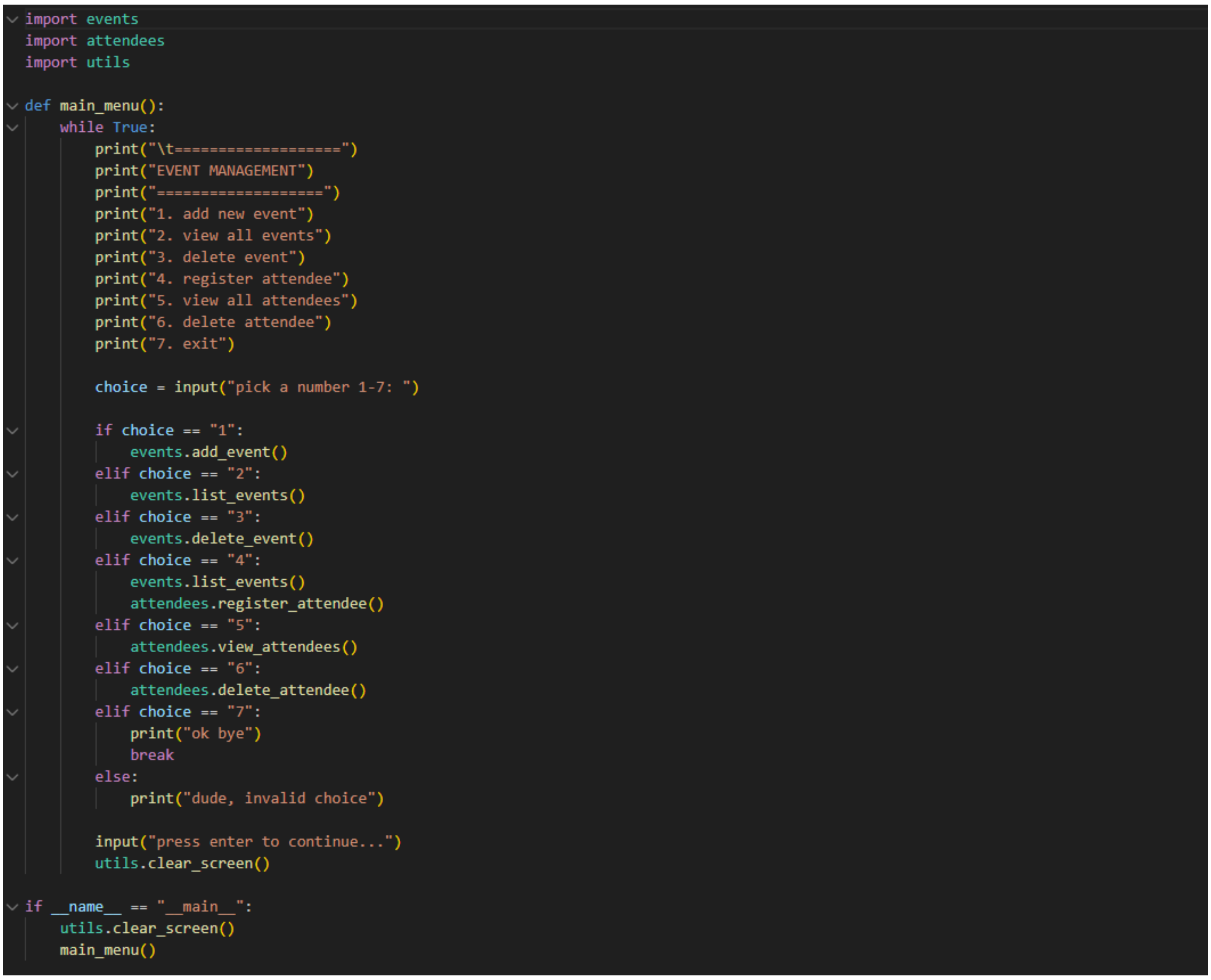
Student Name: Harman Singh
Registration Number: 25BCE10077

Course: CSE1021

Sandip Mal

Date of Submission: 24/11/2025

Contents



```python
import events
import attendees
import utils

def main_menu():
    while True:
        print("\t===================")
        print("EVENT MANAGEMENT")
        print("===================")
        print("1. add new event")
        print("2. view all events")
        print("3. delete event")
        print("4. register attendee")
        print("5. view all attendees")
        print("6. delete attendee")
        print("7. exit")

        choice = input("pick a number 1-7: ")

        if choice == "1":
            events.add_event()
        elif choice == "2":
            events.list_events()
        elif choice == "3":
            events.delete_event()
        elif choice == "4":
            events.list_events()
            attendees.register_attendee()
        elif choice == "5":
            attendees.view_attendees()
        elif choice == "6":
            attendees.delete_attendee()
        elif choice == "7":
            print("ok bye")
            break
        else:
            print("dude, invalid choice")

        input("press enter to continue...")
        utils.clear_screen()

if __name__ == "__main__":
    utils.clear_screen()
    main_menu()
```

Introduction

The **Event Management System** is a solution designed to ease the process of organizing events and managing attendee registrations. In many academic and small business settings, events are still managed using manual methods like paper records or disconnected spreadsheets, leading to inefficiency and data loss. This project provides a centralized, digital platform to handle these tasks effectively using basic Python.

1 Problem Statement

1.1 Current Issue

Event organizers frequently struggle with:

- Loss of participant data due to poor record-keeping.
- Difficulty in retrieving attendee lists for specific events.
- Lack of a unified system to track both event details and registrations simultaneously.

1.2 Proposed Solution

A command-line based application that allows users to create events, securely store them in a file-based system, and manage guest lists without needing complex database software.

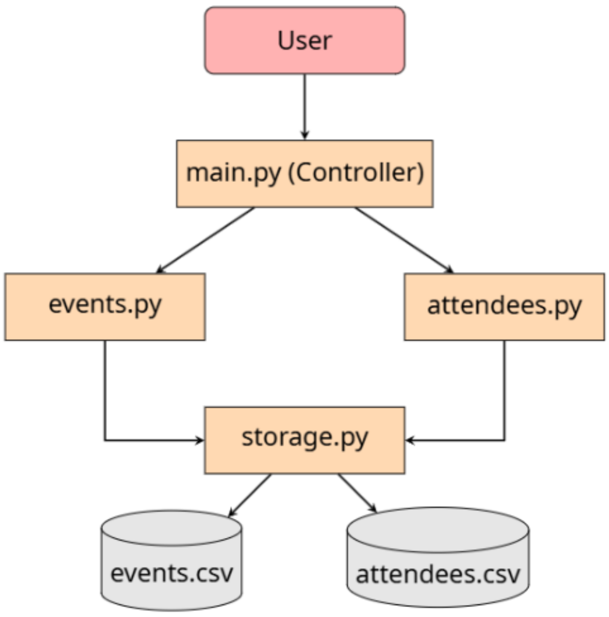
2 Functional Requirements

The system provides the following core functions:

1. **Event Management:**
    - Add new events with Name, Date, and Location.
    - View a list of all upcoming events.
    - Delete cancelled or completed events.
2. **Attendee Management:**
    - Register a participant for a specific event using Event ID.
    - View a master list of all registrations. - Remove an attendee from the list.
3. **Data Persistence:**
    - Automatically save all data to CSV files (events.csv and attendees.csv) so data remains available after restarting the program.

3 System Architecture

The project follows a **Modular Architecture**. The main.py file acts as the controller, accepting user input and directing it to the appropriate logic modules (events.py, attendees.py). These modules then interact with the storage layer (storage.py) to handle data persistence.



4 Design Diagrams

4.1 Workflow Diagram (User Flow)

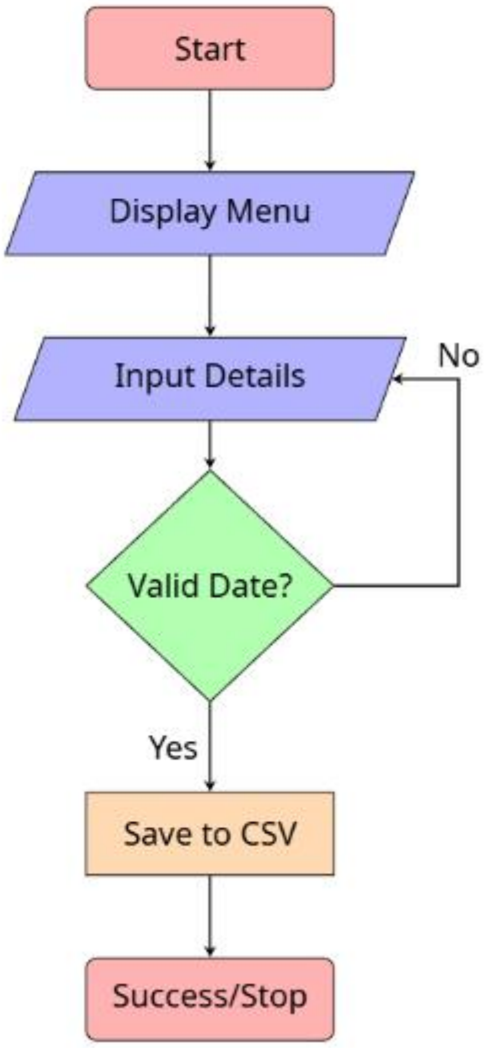
This flow illustrates how a user interacts with the system to add an event.

4.2 ER Diagram (Data Structure)

Since we use CSVs, this represents our logical data structure.

- **Event Entity:** event_id (PK), name, date, location

- **Attendee Entity:** name, email (PK), event_id (FK)



5 Design Decisions & Rationale

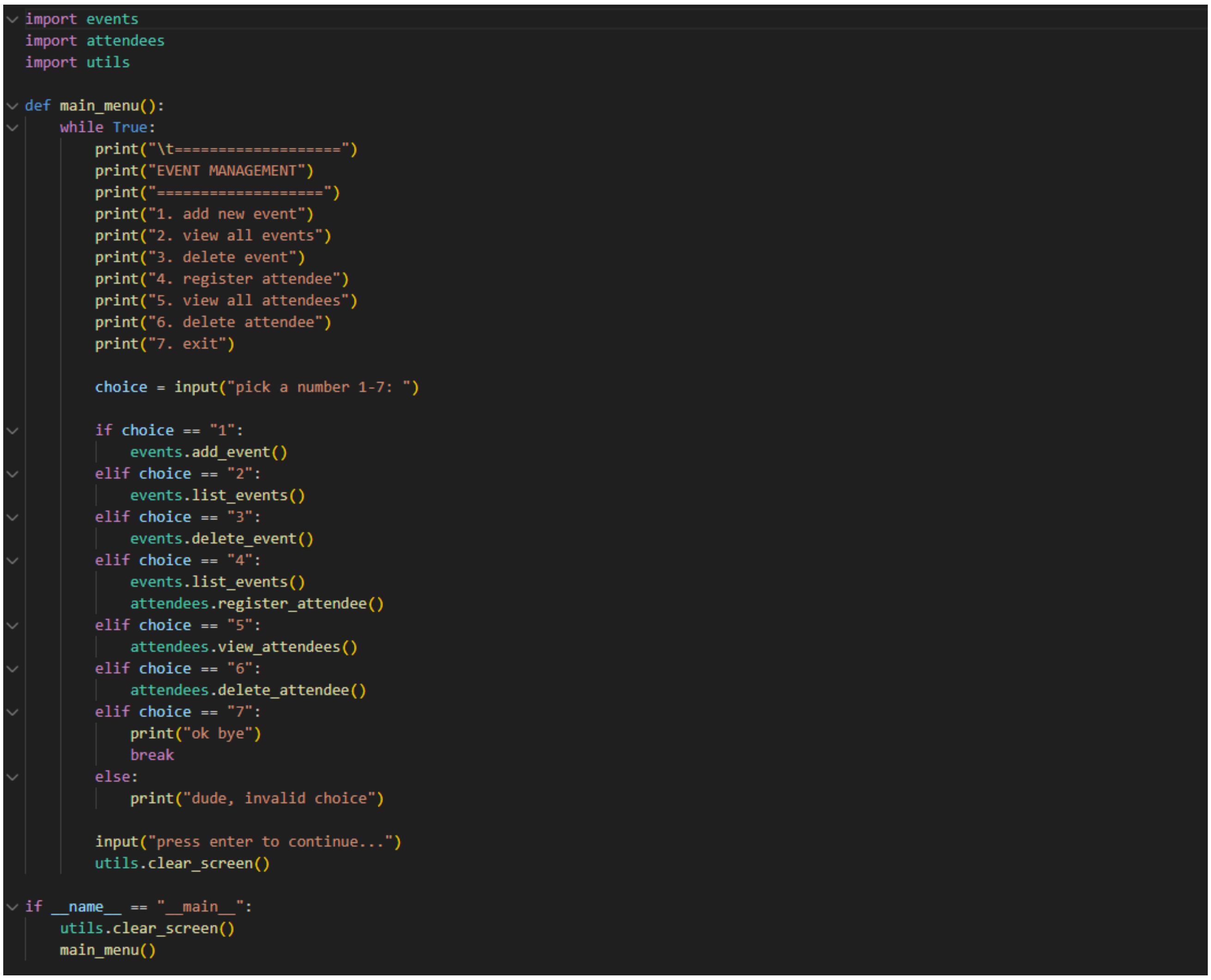
- **Storage(CSVvsDatabase):** CSV files were chosen because it can be opened with excel.
- **ModularStructure:** Instead of putting all code in one file, I split it into 5 modules.

6 Implementation Details

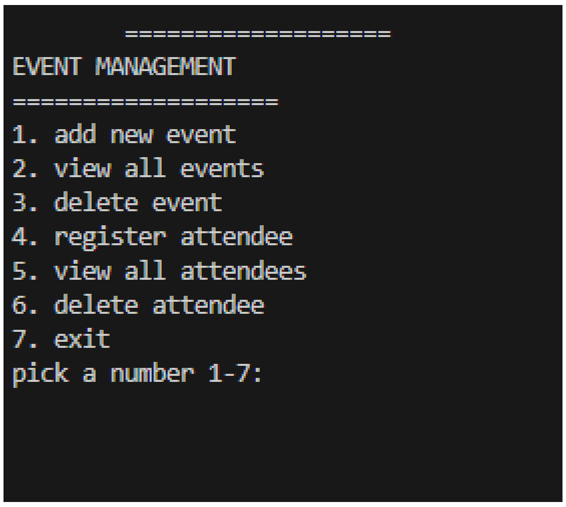
The project consists of 5 core Python files:

- **main.py**: The entry point containing the while loop for the menu system.
- **events.py**: Contains functions add_event, list_events, and delete_event.
- **attendees.py**: Contains logic to link people to event IDs.
- **storage.py**: A reusable module that handles csv.DictReader and csv.DictWriter operations.
- **utils.py**: Contains utility functions like clear_screen() and get_valid_date() to handle errors.

7 Screenshots & Results



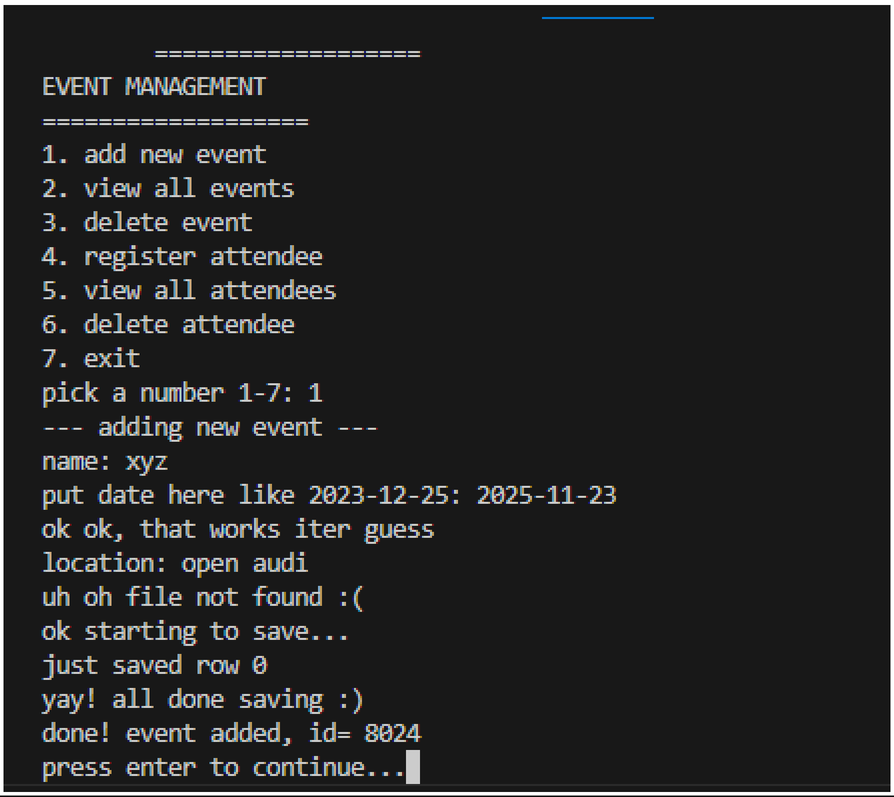
```python
import events
import attendees
import utils

def main_menu():
    while True:
        print("\t===================")
        print("EVENT MANAGEMENT")
        print("===================")
        print("1. add new event")
        print("2. view all events")
        print("3. delete event")
        print("4. register attendee")
        print("5. view all attendees")
        print("6. delete attendee")
        print("7. exit")

        choice = input("pick a number 1-7: ")

        if choice == "1":
            events.add_event()
        elif choice == "2":
            events.list_events()
        elif choice == "3":
            events.delete_event()
        elif choice == "4":
            events.list_events()
            attendees.register_attendee()
        elif choice == "5":
            attendees.view_attendees()
        elif choice == "6":
            attendees.delete_attendee()
        elif choice == "7":
            print("ok bye")
            break
        else:
            print("dude, invalid choice")

        input("press enter to continue...")
        utils.clear_screen()

if __name__ == "__main__":
    utils.clear_screen()
    main_menu()
```



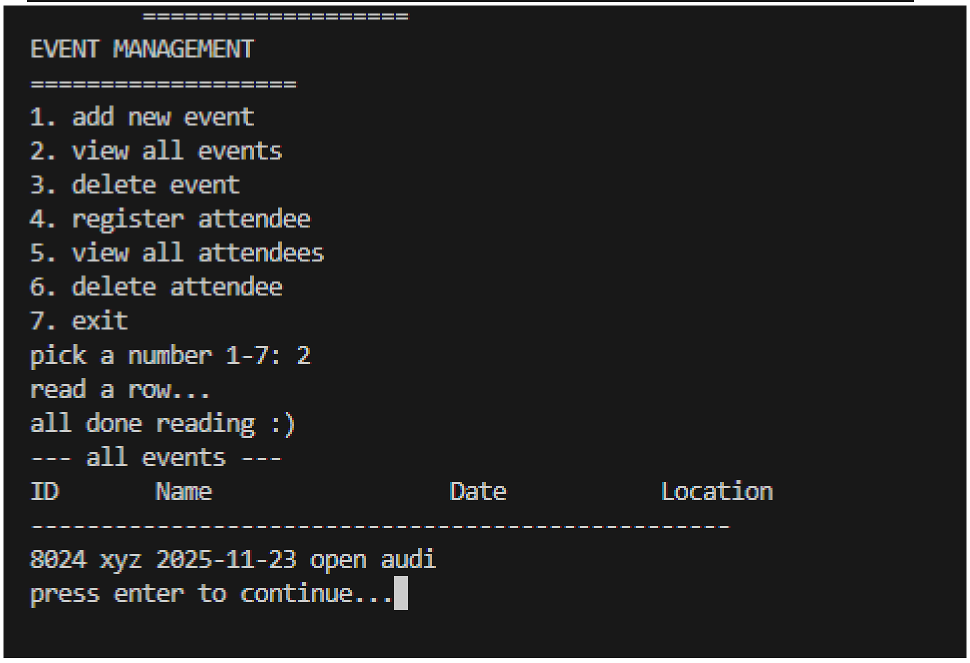
```
        ===================
EVENT MANAGEMENT
===================
1. add new event
2. view all events
3. delete event
4. register attendee
5. view all attendees
6. delete attendee
7. exit
pick a number 1-7:
```

8 Testing Approach

I performed **Black Box Testing** by manually running the program and trying various inputs:



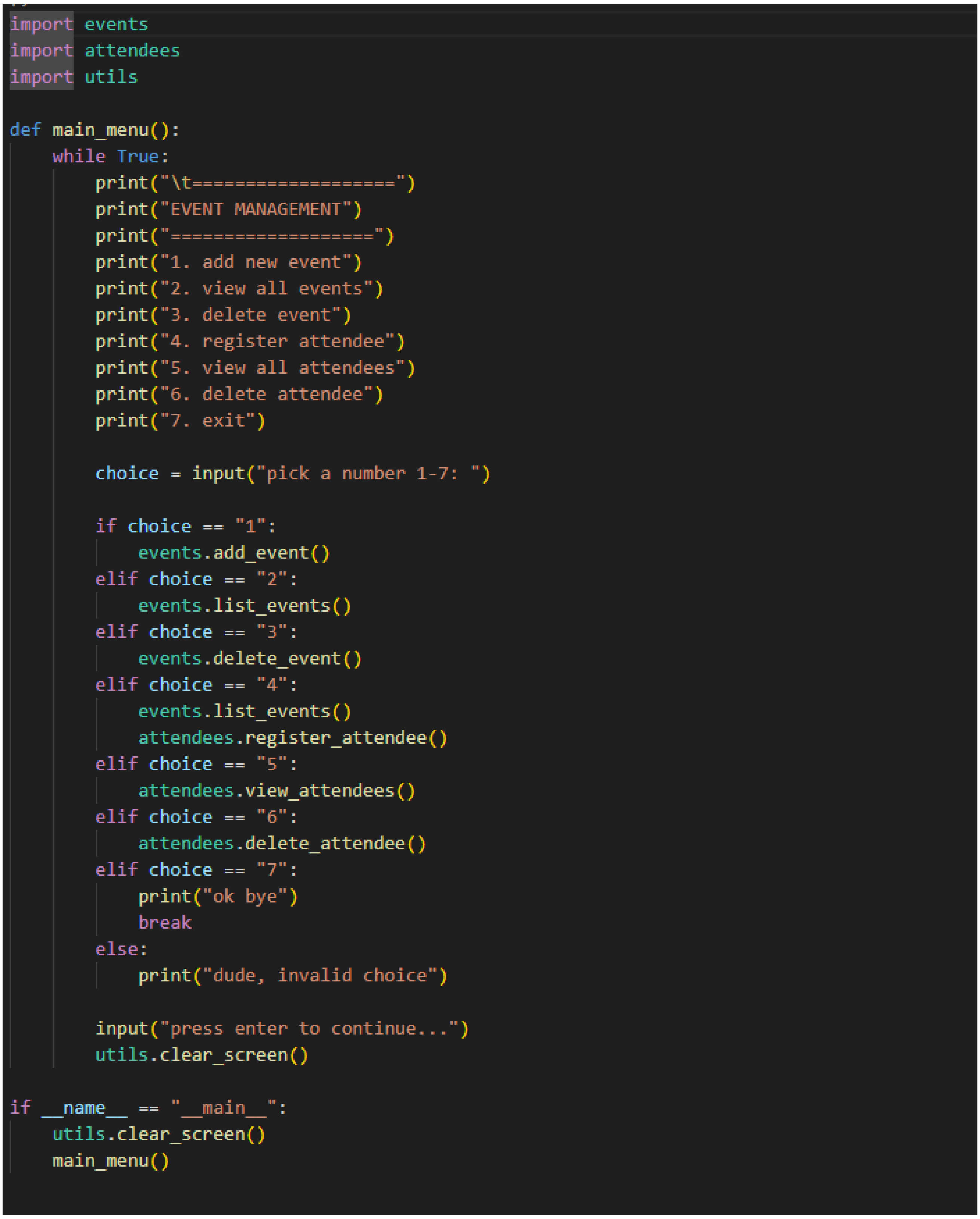
```
        ===================
EVENT MANAGEMENT
===================
1. add new event
2. view all events
3. delete event
4. register attendee
5. view all attendees
6. delete attendee
7. exit
pick a number 1-7: 1
--- adding new event ---
name: xyz
put date here like 2023-12-25: 2025-11-23
ok ok, that works iter guess
location: open audi
uh oh file not found :(
ok starting to save...
just saved row 0
yay! all done saving :)
done! event added, id= 8024
press enter to continue...
```



```
        ===================
EVENT MANAGEMENT
===================
1. add new event
2. view all events
3. delete event
4. register attendee
5. view all attendees
6. delete attendee
7. exit
pick a number 1-7: 2
read a row...
all done reading :)
--- all events ---
ID       Name                  Date           Location
--------------------------------------------------
8024 xyz 2025-11-23 open audi
press enter to continue...
```

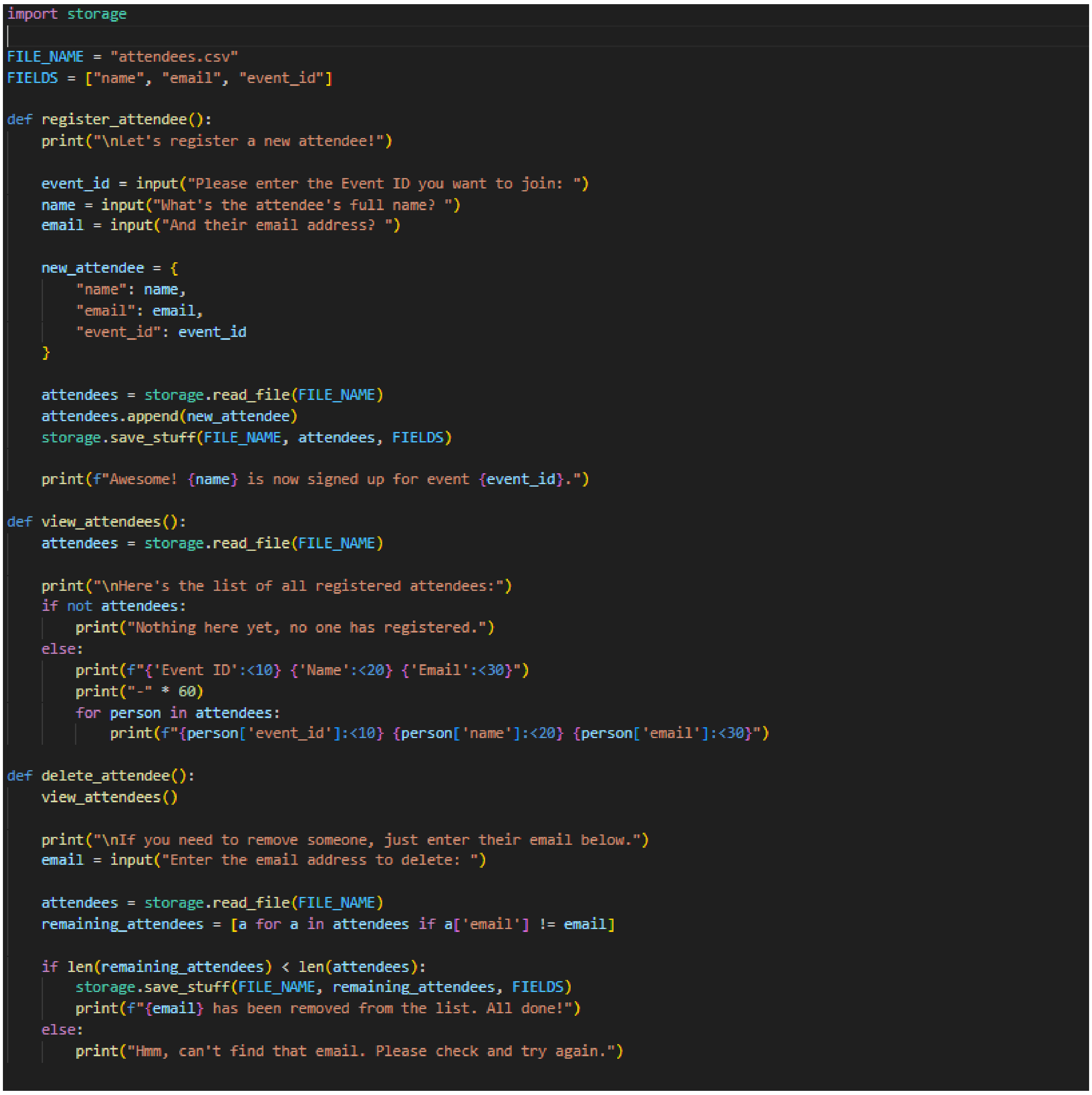
9 Full Source Code

9.1 Main.py



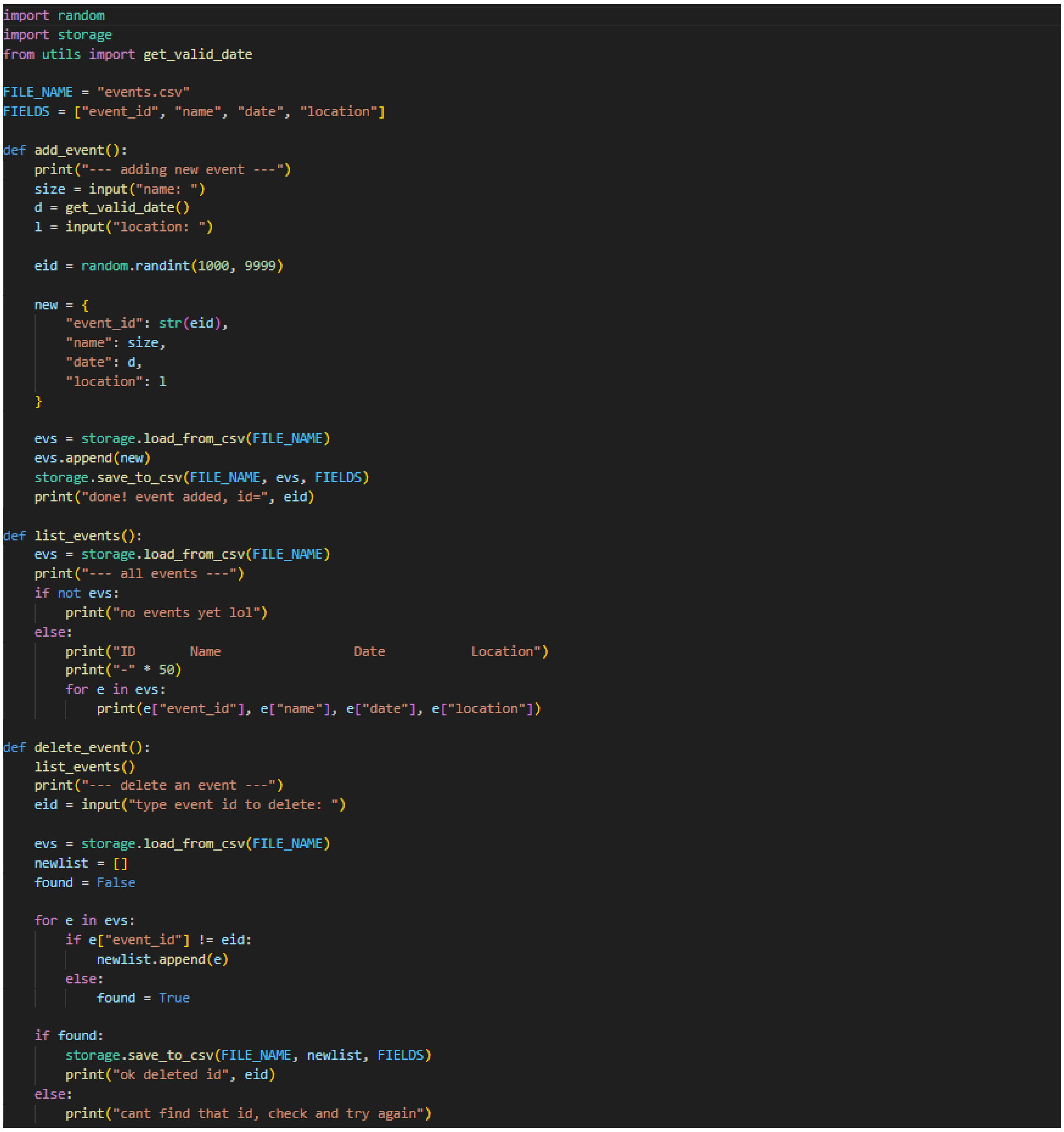
```python
import events
import attendees
import utils

def main_menu():
    while True:
        print("\t==================")
        print("EVENT MANAGEMENT")
        print("==================")
        print("1. add new event")
        print("2. view all events")
        print("3. delete event")
        print("4. register attendee")
        print("5. view all attendees")
        print("6. delete attendee")
        print("7. exit")

        choice = input("pick a number 1-7: ")

        if choice == "1":
            events.add_event()
        elif choice == "2":
            events.list_events()
        elif choice == "3":
            events.delete_event()
        elif choice == "4":
            events.list_events()
            attendees.register_attendee()
        elif choice == "5":
            attendees.view_attendees()
        elif choice == "6":
            attendees.delete_attendee()
        elif choice == "7":
            print("ok bye")
            break
        else:
            print("dude, invalid choice")

        input("press enter to continue...")
        utils.clear_screen()

if __name__ == "__main__":
    utils.clear_screen()
    main_menu()
```

9.2 Attendees.py



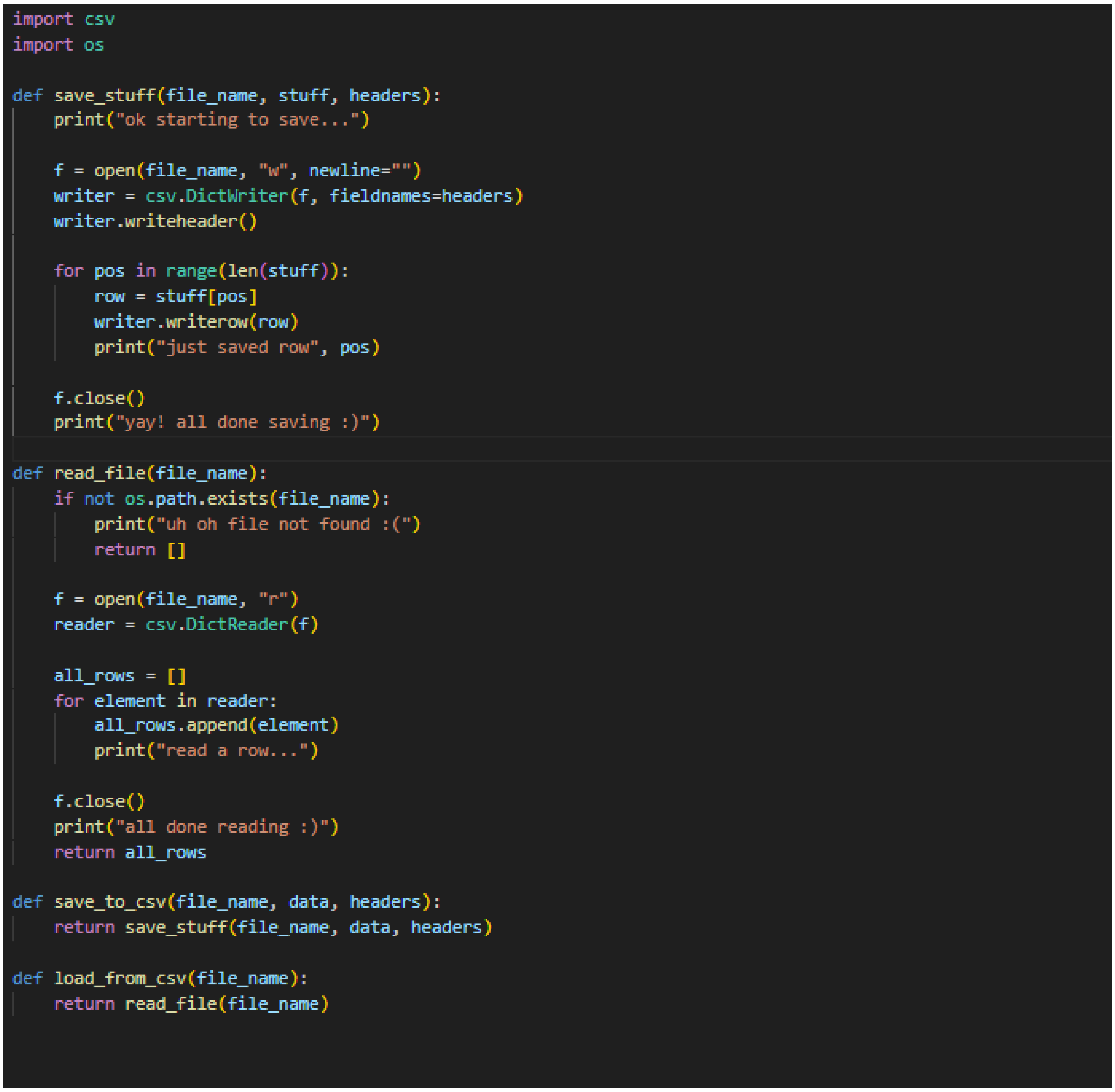
```python
import storage

FILE_NAME = "attendees.csv"
FIELDS = ["name", "email", "event_id"]

def register_attendee():
    print("\nLet's register a new attendee!")

    event_id = input("Please enter the Event ID you want to join: ")
    name = input("What's the attendee's full name? ")
    email = input("And their email address? ")

    new_attendee = {
        "name": name,
        "email": email,
        "event_id": event_id
    }

    attendees = storage.read_file(FILE_NAME)
    attendees.append(new_attendee)
    storage.save_stuff(FILE_NAME, attendees, FIELDS)

    print(f"Awesome! {name} is now signed up for event {event_id}.")

def view_attendees():
    attendees = storage.read_file(FILE_NAME)

    print("\nHere's the list of all registered attendees:")
    if not attendees:
        print("Nothing here yet, no one has registered.")
    else:
        print(f"{'Event ID':<10} {'Name':<20} {'Email':<30}")
        print("-" * 60)
        for person in attendees:
            print(f"{person['event_id']:<10} {person['name']:<20} {person['email']:<30}")

def delete_attendee():
    view_attendees()

    print("\nIf you need to remove someone, just enter their email below.")
    email = input("Enter the email address to delete: ")

    attendees = storage.read_file(FILE_NAME)
    remaining_attendees = [a for a in attendees if a['email'] != email]

    if len(remaining_attendees) < len(attendees):
        storage.save_stuff(FILE_NAME, remaining_attendees, FIELDS)
        print(f"{email} has been removed from the list. All done!")
    else:
        print("Hmm, can't find that email. Please check and try again.")
```

9.3 Events.py



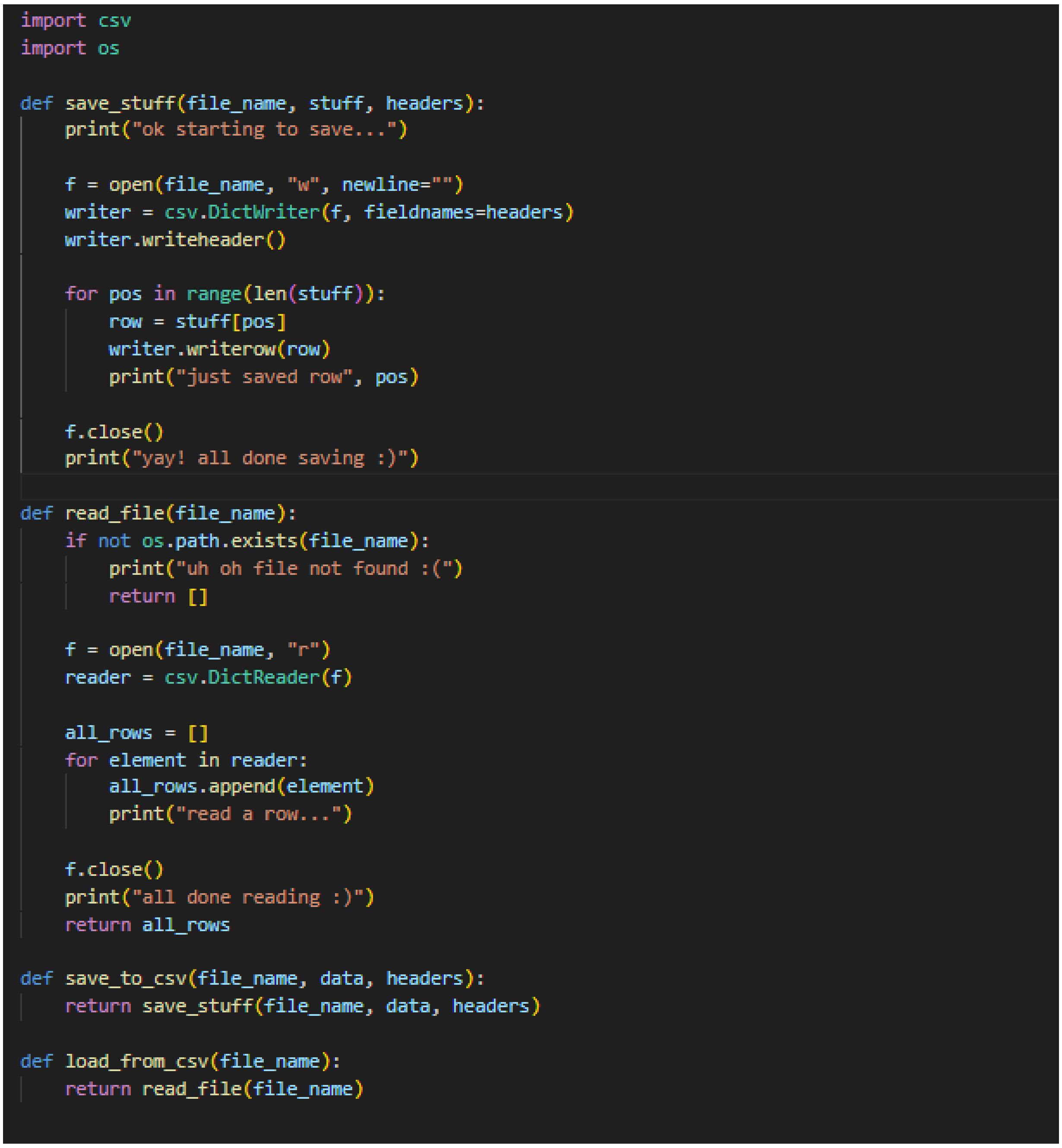
```python
import random
import storage
from utils import get_valid_date

FILE_NAME = "events.csv"
FIELDS = ["event_id", "name", "date", "location"]

def add_event():
    print("--- adding new event ---")
    size = input("name: ")
    d = get_valid_date()
    l = input("location: ")

    eid = random.randint(1000, 9999)

    new = {
        "event_id": str(eid),
        "name": size,
        "date": d,
        "location": l
    }

    evs = storage.load_from_csv(FILE_NAME)
    evs.append(new)
    storage.save_to_csv(FILE_NAME, evs, FIELDS)
    print("done! event added, id=", eid)

def list_events():
    evs = storage.load_from_csv(FILE_NAME)
    print("--- all events ---")
    if not evs:
        print("no events yet lol")
    else:
        print("ID       Name                 Date           Location")
        print("-" * 50)
        for e in evs:
            print(e["event_id"], e["name"], e["date"], e["location"])

def delete_event():
    list_events()
    print("--- delete an event ---")
    eid = input("type event id to delete: ")

    evs = storage.load_from_csv(FILE_NAME)
    newlist = []
    found = False

    for e in evs:
        if e["event_id"] != eid:
            newlist.append(e)
        else:
            found = True

    if found:
        storage.save_to_csv(FILE_NAME, newlist, FIELDS)
        print("ok deleted id", eid)
    else:
        print("cant find that id, check and try again")
```

9.4 Storage.py



```python
import csv
import os

def save_stuff(file_name, stuff, headers):
    print("ok starting to save...")

    f = open(file_name, "w", newline="")
    writer = csv.DictWriter(f, fieldnames=headers)
    writer.writeheader()

    for pos in range(len(stuff)):
        row = stuff[pos]
        writer.writerow(row)
        print("just saved row", pos)

    f.close()
    print("yay! all done saving :)")

def read_file(file_name):
    if not os.path.exists(file_name):
        print("uh oh file not found :(")
        return []

    f = open(file_name, "r")
    reader = csv.DictReader(f)

    all_rows = []
    for element in reader:
        all_rows.append(element)
        print("read a row...")

    f.close()
    print("all done reading :)")
    return all_rows

def save_to_csv(file_name, data, headers):
    return save_stuff(file_name, data, headers)

def load_from_csv(file_name):
    return read_file(file_name)
```

9.5 Utils.py



```python
import csv
import os

def save_stuff(file_name, stuff, headers):
    print("ok starting to save...")

    f = open(file_name, "w", newline="")
    writer = csv.DictWriter(f, fieldnames=headers)
    writer.writeheader()

    for pos in range(len(stuff)):
        row = stuff[pos]
        writer.writerow(row)
        print("just saved row", pos)

    f.close()
    print("yay! all done saving :)")

def read_file(file_name):
    if not os.path.exists(file_name):
        print("uh oh file not found :(")
        return []

    f = open(file_name, "r")
    reader = csv.DictReader(f)

    all_rows = []
    for element in reader:
        all_rows.append(element)
        print("read a row...")

    f.close()
    print("all done reading :)")
    return all_rows

def save_to_csv(file_name, data, headers):
    return save_stuff(file_name, data, headers)

def load_from_csv(file_name):
    return read_file(file_name)
```

10 Conclusion

The Event Management System efficiently automates the process of storing, updating, and analysing Events, attendees' records. It reduces manual errors, ensures no loss of data , makes it easier to access attendee list for a specific event, and is an easy way to track both event and registrations. The system makes Event management faster, easier, and more reliable

11 References

1. Python Official Documentation-https://docs.python.org
2. NumPy Documentation — https://numpy.org
3. Stack Overflow (general debugging help)