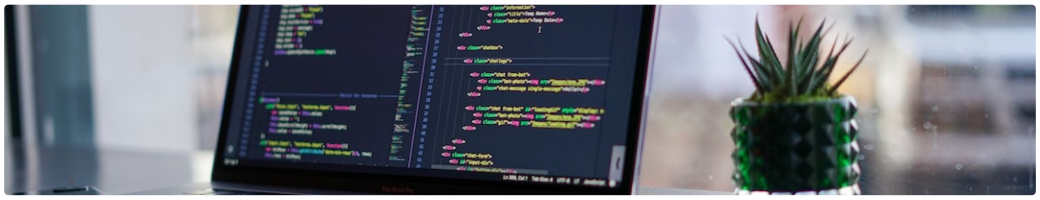


Core Python Event Management System

Introduction:

Welcome to my documentation for my EMS. This console-based system is designed using Python to manage events, and attendees and a seamless experience for event organisers.

Classes

Full Classes.py code

```python
# Making my classes

# 1st class
# creating a parent class for common attributes shared by the Event and Attndees class
class EventManager:
    def __init__(self, event_id, name, date=None, location=None):
        self.event_id = event_id
        self.name = name
        self.date = date
        self.location = location
        self.attendees = []

    # A method (operate within the class)
    def display_details(self):
        print(f'ID: {self.event_id}')
        print(f'Name: {self.name}')

# 2nd class - Event
# it inherites from my EventManager parent class
class Event(EventManager):
    def __init__(self, event_id, name, date, location):
        super().__init__(event_id, name, date, location)
        # super() function is what allows the child class to inherit all the properties of the parent class

    def display_details(self):
        super().display_details()
        print(f'Date: {self.date}')
        print(f'Location: {self.location}')
        print('Attendees:')
        for attendee in self.attendees:
            print(f'{attendee.name} - {attendee.phone}')

    def edit_details(self, new_name, new_date, new_location):
        # method to edit events details
        self.name = new_name
        self.date = new_date
        self.location = new_location

    def add_attendee(self, attendee):
        #Add an Attendee to the Events list
        self.attendees.append(attendee)

    def remove_attendee(self, attendee):
        # remove an attendee from the Events list
        self.attendees.remove(attendee)

# 3rd class
class Attendee(EventManager):
    def __init__(self, attendee_id, name, phone):
        # calling the blueprint of the base class
        super().__init__(attendee_id, name)
        # additional attributes specific to Attendee class
        self.phone = phone

    def display_details(self):
        # displaying detail of an Attendee
        super().display_details()
        print(f'Phone: {self.phone}')
```

1. Event Manager (Parent class)
    a. This is our parent class for common attributes shared by the event and attendees class.
    b. This is the core part of our system, responsible for holding essential details about events, such as names and dates. It's like a digital event planner.
2. Event (Child class)
    a. Events are the heart of the system. This class represents and stores the event information such as event date and location.
3. Attendee (Child class)
    a. The attendee class is for storing the information about the people attending the event, such as names, phones and numbers.

Functions

Functions are needed to add extra

Full Function.py code

```python
# Making my function
import csv
from classes import Attendee

# base/parent class
class EventManager:
    def __init__(self, event_id, name):
        # initilising common attributes for all classes
        self.event_id = event_id
        self.name = name

# creating a method for displaying the details of this base class
    # methods operate within a class
    def display_details(self):
        pass

# a function = operates OUTSIDE a class
# function to create a new event
def create_event(events, Event):

    try:
        event_id = int(input(f'Enter {Event.__name__} ID: '))
        name = input(f'Enter {Event.__name__} Name: ')
        date = input(f'Enter {Event.__name__} Date (YYYY-MM-DD): ')
        location = input(f'Enter {Event.__name__} Lcation: ')

        # checking if the event ID already exits
        if any(event.event_id == event_id for event in events):
            print(f'{Event.__name__} ID already exist. Please choose a different ID.')
            return

        # instance of the Event class and adds it to the list
        event = Event(event_id, name, date, location)
        events.append(event)
        print(f'{Event.__name__} created successfully.')
    except ValueError:
        print('Invalid input. Please enter valid values.')


# this function lists details of all events
def list_all_events(events):
    for event in events:
        event.display_details() #calling

# function to list details of an indvidual event by ID
def list_individual_event(event_id, events):
    # Find and display details of an individual event by ID
    for event in events:
        if event.event_id == event_id:
            event.display_details()
            break
    else:
        print(f'Event with ID {event_id} not found.')

# function to edit details of an existing event
def edit_event(events, Event):
    try:
        # Get input from the user
        event_id = int(input(f'Enter the {Event.__name__} ID to edit: '))
        for event in events:
            if event.event_id == event_id:
                # Get new details from the user and update the event
                new_name = input(f'Enter new {Event.__name__} Name: ')
                new_date = input(f'Enter new {Event.__name__} Date (YYYY-MM-DD): ')
                new_location = input(f'Enter new {Event.__name__} Location: ')

                event.edit_details(new_name, new_date, new_location)
                print(f'{Event.__name__} details have been updated successfully.')
                break
        else:
            print(f'{Event.__name__} with ID {event_id} not found.')
    except ValueError:
        print('Invalid input. Please enter a valid ID.')

# Function to delete an existing event
```

```python
def delete_event(events, Event):
    try:
        # Get input from the user
        event_id = int(input(f'Enter the {Event.__name__} ID to delete: '))
        for event in events:
            if event.event_id == event_id:
                # Remove the event from the list
                events.remove(event)
                print(f'{Event.__name__} deleted successfully.')
                break
        else:
            print(f'{Event.__name__} with ID {event_id} not found.')
    except ValueError:
        print('Invalid input. Please enter a valid ID.')

def list_attendees(event_id, events):
    for event in events:
        if event.event_id == event_id:
            # Display attendees for the selected event
            if event.attendees:
                print('Attendees:')
                for attendee in event.attendees:
                    print(f'{attendee.name} - {attendee.phone}')
            else:
                print('No attendees for this event.')
            break
    else:
        print(f'Event with ID {event_id} not found.')

def add_attendee(events):
    try:
        # Get input from the user
        event_id = int(input('Enter the Event ID to add an attendee: '))
        for event in events:
            if event.event_id == event_id:
                # Ask if the user wants to add multiple attendees
                multiple_attendees = input('Do you want to add multiple attendees? (y/n): ').lower() == 'y'
                while True:
                    attendee_id = int(input('Enter Attendee ID: '))
                    name = input('Enter Attendee Name: ')
                    phone = input('Enter Attendee Phone: ')

                    # Create an instance of the Attendee class and add it to the event
                    attendee = Attendee(attendee_id, name, phone)
                    event.add_attendee(attendee)
                    print('Attendee added successfully.')

                    if not multiple_attendees:
                        break

                    add_more = input('Do you want to add another attendee? (y/n): ').lower()
                    if add_more != 'y':
                        break

                break
        else:
            print(f'Event with ID {event_id} not found.')
    except ValueError:
        print('Invalid input. Please enter valid values.')

def delete_attendee(events):
    try:
        # Get input from the user
        event_id = int(input('Enter the Event ID to delete an attendee: '))
        for event in events:
            if event.event_id == event_id:
                attendee_name = input('Enter Attendee Name: ')
                attendee_phone = input('Enter Attendee Phone: ')

                for attendee in event.attendees:
                    if attendee.name == attendee_name and attendee.phone == attendee_phone:
                        # Remove the attendee from the event
                        event.remove_attendee(attendee)
                        print('Attendee deleted successfully.')
                        break
                else:
                    print('Attendee not found.')
                break
        else:
            print(f'Event with ID {event_id} not found.')
    except ValueError:
        print('Invalid input. Please enter valid values.')

# Function to read events and attendees from a CSV file
def read_events_from_csv(eventfile, Event, Attendee):
    events = []

    try:
        # Try to open the file for reading
        with open(eventfile, mode='r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                event_id = int(row['event_id'])
                event_name = row['event_name']
                date = row['date']
                location = row['location']

                # using a generator expression = Check if the event already exists in the list
                event = next((event for event in events if event.event_id == event_id), None)

                if not event:
                    event = Event(event_id, event_name, date, location)
                    events.append(event)

                attendee_id = int(row['attendee_id'])
                attendee_name = row['attendee_name']
                phone = row['phone']

                attendee = Attendee(attendee_id, attendee_name, phone)
                event.add_attendee(attendee)

    except FileNotFoundError:
        # If the file doesn't exist, create it with headers
        with open('events.csv', 'a', newline='') as file:
            fieldnames = ['event_id', 'event_name', 'date', 'location', 'attendee_id', 'attendee_name', 'pho
            writer = csv.DictWriter(file, fieldnames=fieldnames)
            writer.writeheader()

    return events

# Function to write events and attendees to a CSV file
def write_events_to_csv(eventfile, events):
    with open(eventfile, 'w', newline='') as file:
        fieldnames = ['event_id', 'event_name', 'date', 'location', 'attendee_id', 'attendee_name', 'phone']
        writer = csv.DictWriter(file, fieldnames=fieldnames)
        writer.writeheader()

        for event in events:
            for attendee in event.attendees:
                writer.writerow({
                    'event_id': event.event_id,
                    'event_name': event.name,
                    'date': event.date,
                    'location': event.location,
                    'attendee_id': attendee.event_id,
                    'attendee_name': attendee.name,
                    'phone': attendee.phone
                })
```

Create_event - Use this to create a new event. It's like filling out a form for a new party or gathering.

List_all_events - This function shows a list of all events, like opening your event calendar to see what's coming up.

List_individual_event - Find detailed information about a specific event using this function. It's like checking the details of a single event on your calendar.

Edit_event - Edit the details of an event using this function, similar to updating information about a meeting or party.

Delete_event - Remove an event from the system, just like cancelling a planned event.

List_attendees - See who's attending a specific event, like checking the guest list for a party.

Add_attendee - Use this to add people to an event, just like inviting friends to a gathering.

Delete_attendee - Remove someone from an event, similar to uninviting someone from a gathering.

Read_events_from_csv - This function reads events and attendees from a file, like importing details from a saved list.

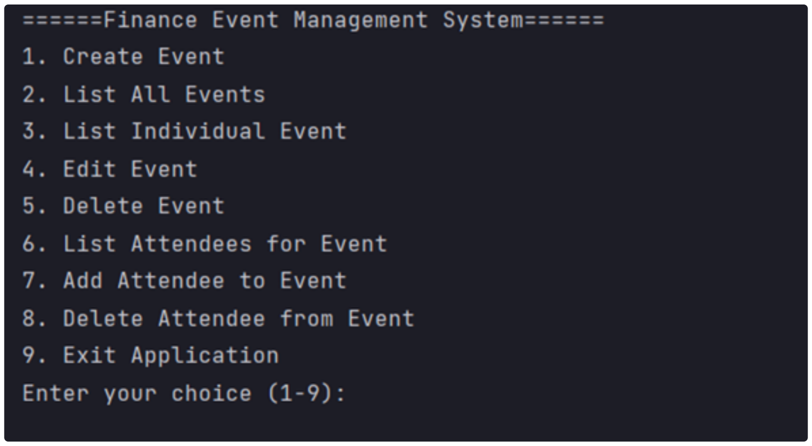
Write_events_to_csv - Write events and attendee details to a file, like saving your event plans.

Main script

Full Main script.py code

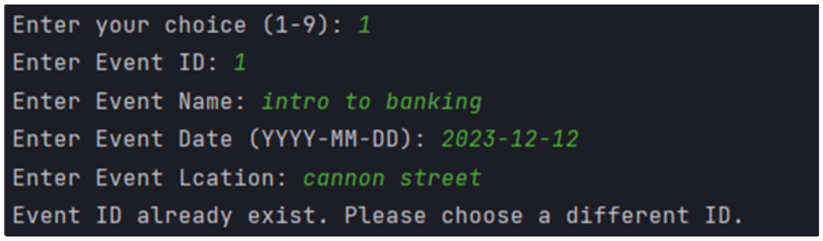
```python
# Importing necessary modules
from classes import Event, Attendee
from functions import (
    create_event, list_all_events, list_individual_event,
    edit_event, delete_event, list_attendees,
    add_attendee, delete_attendee, read_events_from_csv,
    write_events_to_csv
)

def main():
    # Reads events and 2attendees from CSV file
    events = read_events_from_csv('events.csv', Event, Attendee)

    # While loop for the menu
    while True:
        print('======Finance Event Management System======')
        print('1. Create Event')
        print('2. List All Events')
        print('3. List Individual Event')
        print('4. Edit Event')
        print('5. Delete Event')
        print('6. List Attendees for Event')
        print('7. Add Attendee to Event')
        print('8. Delete Attendee from Event')
        print('9. Exit Application')

        # Asking for user input
        user_choice = input('Enter your choice (1-9): ')

        # If statement for the user's choice
        if user_choice == '1':
            create_event(events, Event)
        elif user_choice == '2':
            list_all_events(events)
        elif user_choice == '3':
            event_id = int(input('Enter the Event ID: '))
            list_individual_event(event_id, events)
        elif user_choice == '4':
            edit_event(events, Event)
        elif user_choice == '5':
            delete_event(events, Event)
        elif user_choice == '6':
            event_id = int(input('Enter the Event ID to list attendees: '))
            list_attendees(event_id, events)
        elif user_choice == '7':
            add_attendee(events)
        elif user_choice == '8':
            delete_attendee(events)
        elif user_choice == '9':
            # Write events and attendees to CSV file before exiting
            write_events_to_csv('events.csv', events, [])
            print('Exiting the event application... Goodbye!')
            break
        else:
            print('Invalid choice. Please enter a number between 1 and 9.')

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

This is the main menu of our system, you can choose from options from 1 to 9



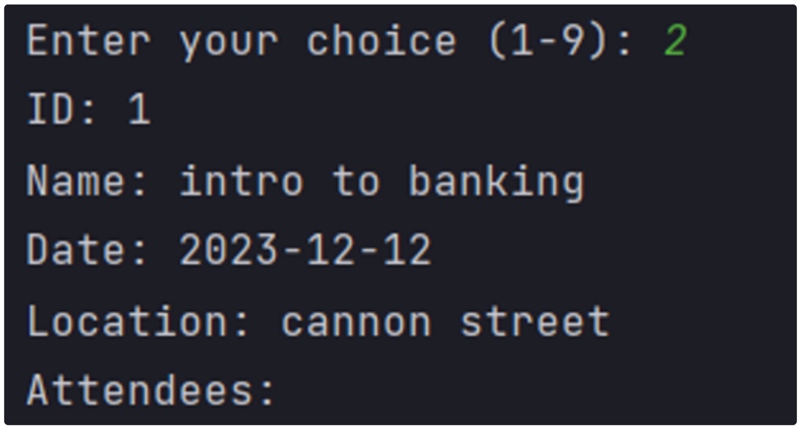
```
======Finance Event Management System======
1. Create Event
2. List All Events
3. List Individual Event
4. Edit Event
5. Delete Event
6. List Attendees for Event
7. Add Attendee to Event
8. Delete Attendee from Event
9. Exit Application
Enter your choice (1-9):
```

Option 1: creating an event



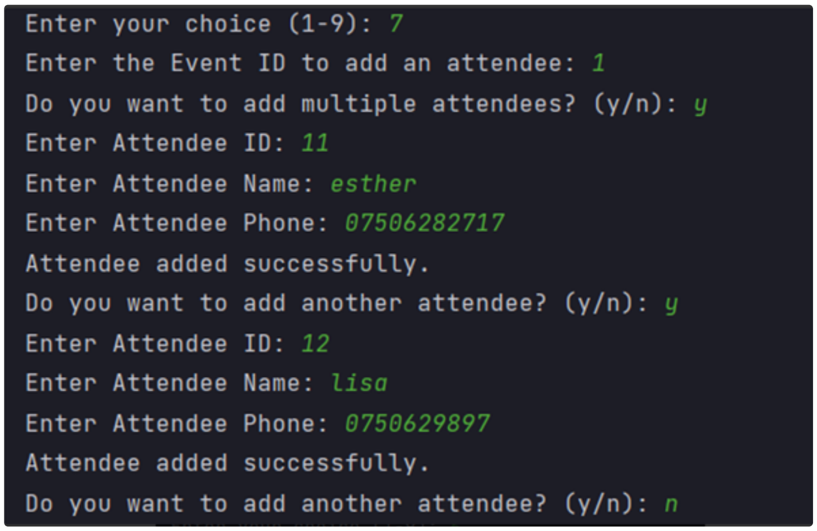
```
Enter your choice (1-9): 1
Enter Event ID: 1
Enter Event Name: intro to banking
Enter Event Date (YYYY-MM-DD): 2023-12-12
Enter Event Lcation: cannon street
Event ID already exist. Please choose a different ID.
```

Option 2: List the events



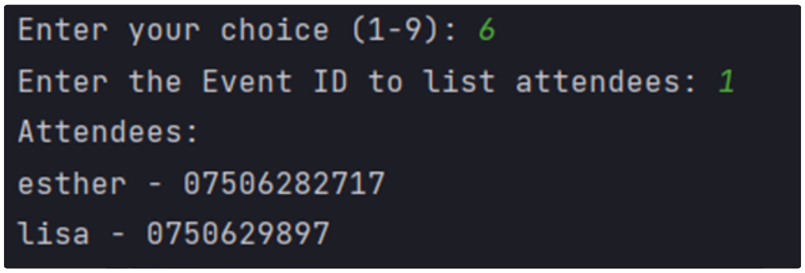
```
Enter your choice (1-9): 2
ID: 1
Name: intro to banking
Date: 2023-12-12
Location: cannon street
Attendees:
```

Option 7: Adding attendees to the event



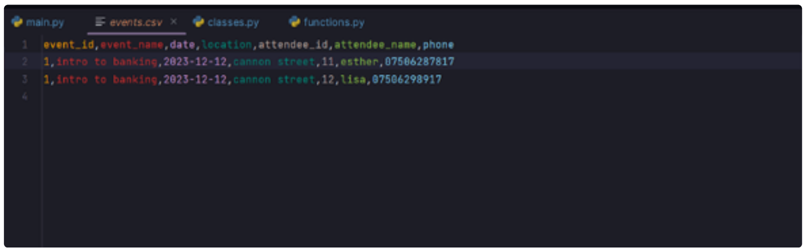
```
Enter your choice (1-9): 7
Enter the Event ID to add an attendee: 1
Do you want to add multiple attendees? (y/n): y
Enter Attendee ID: 11
Enter Attendee Name: esther
Enter Attendee Phone: 07506282717
Attendee added successfully.
Do you want to add another attendee? (y/n): y
Enter Attendee ID: 12
Enter Attendee Name: lisa
Enter Attendee Phone: 0750629897
Attendee added successfully.
Do you want to add another attendee? (y/n): n
```

Option 6: List the attendees for the event ID 1



```
Enter your choice (1-9): 6
Enter the Event ID to list attendees: 1
Attendees:
esther - 07506282717
lisa - 0750629897
```

Output of the system - the information was stored in a CSV file once application is exsited



```
event_id,event_name,date,location,attendee_id,attendee_name,phone
1,intro to banking,2023-12-12,cannon street,11,esther,07506287817
1,intro to banking,2023-12-12,cannon street,12,lisa,07506298917
```