**1. Implement an event management system using OOP concepts to organize and manage various aspects of college festivals or events. Design classes for events, organizers, participants, and activities. Include methods for event registration, scheduling, participant management, and activity coordination.**

class Participant:

def \_\_init\_\_(self, name, email):

self.name = name

self.email = email

class Organizer:

def \_\_init\_\_(self, name, contact):

self.name = name

self.contact = contact

class Activity:

def \_\_init\_\_(self, name, time, organizer):

self.name = name

self.time = time

self.organizer = organizer

class Event:

def \_\_init\_\_(self, name, date):

self.name = name

self.date = date

self.participants = []

self.activities = []

def register\_participant(self, participant):

self.participants.append(participant)

print(f"Participant {participant.name} registered.")

def add\_activity(self, activity):

self.activities.append(activity)

print(f"Activity {activity.name} added.")

def schedule\_activities(self):

print(f"Schedule for event '{self.name}':")

for activity in self.activities:

print(f"{activity.time}: {activity.name} by {activity.organizer.name}")

def list\_participants(self):

print(f"Participants for event '{self.name}':")

for participant in self.participants:

print(f"{participant.name} ({participant.email})")

# Example usage

# Create organizers

organizer1 = Organizer("Alice", "alice@example.com")

organizer2 = Organizer("Bob", "bob@example.com")

# Create activities

activity1 = Activity("Opening Ceremony", "10:00 AM", organizer1)

activity2 = Activity("Tech Talk", "11:00 AM", organizer2)

# Create event

event = Event("College Fest", "2025-02-20")

# Register participants

participant1 = Participant("John Doe", "john.doe@example.com")

participant2 = Participant("Jane Smith", "jane.smith@example.com")

event.register\_participant(participant1)

event.register\_participant(participant2)

# Add activities to event

event.add\_activity(activity1)

event.add\_activity(activity2)

# Schedule activities

event.schedule\_activities()

# List participants

event.list\_participants()

**Explanation:**

* **Classes**:
  + Participant: Represents a participant with attributes name and email.
  + Organizer: Represents an organizer with attributes name and contact.
  + Activity: Represents an activity with attributes name, time, and organizer.
  + Event: Represents an event with attributes name, date, a list of participants, and a list of activities. Methods include register\_participant, add\_activity, schedule\_activities, and list\_participants.
* **Methods**:
  + register\_participant: Adds a participant to the event.
  + add\_activity: Adds an activity to the event.
  + schedule\_activities: Prints the schedule of activities for the event.
  + list\_participants: Prints the list of registered participants for the event.

**2. Develop classes for products, customers, and shopping carts. Include methods for adding items to the cart, calculating total costs, processing orders, and managing inventory.**

**class Product:**

**def \_\_init\_\_(self, id, name, price, stock):**

**self.id = id**

**self.name = name**

**self.price = price**

**self.stock = stock**

**def update\_stock(self, quantity):**

**self.stock += quantity**

**class Customer:**

**def \_\_init\_\_(self, id, name, email):**

**self.id = id**

**self.name = name**

**self.email = email**

**self.cart = ShoppingCart()**

**def add\_to\_cart(self, product, quantity):**

**self.cart.add\_item(product, quantity)**

**def checkout(self):**

**total = self.cart.calculate\_total()**

**self.cart.process\_order()**

**return total**

**class ShoppingCart:**

**def \_\_init\_\_(self):**

**self.items = {}**

**def add\_item(self, product, quantity):**

**if product.id in self.items:**

**self.items[product.id]['quantity'] += quantity**

**else:**

**self.items[product.id] = {'product': product, 'quantity': quantity}**

**def calculate\_total(self):**

**total = 0**

**for item in self.items.values():**

**total += item['product'].price \* item['quantity']**

**return total**

**def process\_order(self):**

**for item in self.items.values():**

**item['product'].update\_stock(-item['quantity'])**

**self.items = {}**

**# Example usage**

**apple = Product(1, 'Apple', 0.5, 100)**

**banana = Product(2, 'Banana', 0.3, 150)**

**john = Customer(1, 'John Doe', 'john@example.com')**

**john.add\_to\_cart(apple, 10)**

**john.add\_to\_cart(banana, 5)**

**print("Total Cost:", john.checkout())**

In this code:

* The Product class manages product details and stock levels.
* The Customer class handles customer information and their shopping cart.
* The ShoppingCart class maintains items in the cart, calculates the total cost, and processes orders by updating stock levels.

**3.Design a system using classes for vehicles, rental agencies, and**

**rental transactions. Implement methods to handle vehicle availability, rental periods, pricing, and customer bookings.**

**class Participant:**

**def \_\_init\_\_(self, name, email):**

**self.name = name**

**self.email = email**

**class Organizer:**

**def \_\_init\_\_(self, name, email):**

**self.name = name**

**self.email = email**

**self.events = []**

**def create\_event(self, event):**

**self.events.append(event)**

**class Activity:**

**def \_\_init\_\_(self, name, time, location):**

**self.name = name**

**self.time = time**

**self.location = location**

**class Event:**

**def \_\_init\_\_(self, name, date, location):**

**self.name = name**

**self.date = date**

**self.location = location**

**self.participants = []**

**self.activities = []**

**def register\_participant(self, participant):**

**self.participants.append(participant)**

**def add\_activity(self, activity):**

**self.activities.append(activity)**

**def schedule\_activity(self):**

**self.activities.sort(key=lambda x: x.time)**

**# Example usage**

**def main():**

**# Creating organizers**

**organizer1 = Organizer("Alice", "alice@example.com")**

**organizer2 = Organizer("Bob", "bob@example.com")**

**# Creating an event**

**event1 = Event("Tech Fest", "2025-03-25", "Auditorium")**

**# Organizers creating events**

**organizer1.create\_event(event1)**

**# Creating participants**

**participant1 = Participant("Charlie", "charlie@example.com")**

**participant2 = Participant("David", "david@example.com")**

**# Registering participants**

**event1.register\_participant(participant1)**

**event1.register\_participant(participant2)**

**# Creating activities**

**activity1 = Activity("Opening Ceremony", "10:00 AM", "Main Hall")**

**activity2 = Activity("Coding Competition", "01:00 PM", "Room 101")**

**# Adding activities to the event**

**event1.add\_activity(activity1)**

**event1.add\_activity(activity2)**

**# Scheduling activities**

**event1.schedule\_activity()**

**print(f"Event: {event1.name}")**

**print("Participants:")**

**for participant in event1.participants:**

**print(f"{participant.name} - {participant.email}")**

**print("Activities:")**

**for activity in event1.activities:**

**print(f"{activity.name} at {activity.time} in {activity.location}")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

This script creates:

* **Participant**: Class to represent participants with their name and email.
* **Organizer**: Class to represent organizers who can create events.
* **Activity**: Class to represent activities within an event, including name, time, and location.
* **Event**: Class to manage an event, including participant registration, adding activities, and scheduling them.