

Core python assessment

Healthcare Industry

Design a Python class ClinicAppointment that manages patient appointments in a clinic. The system should have the following features:

→ Book Appointment:

- Prompt for patient name, age, mobile number, and preferred doctor.
- Show time slots (10am, 11am, 12pm, 2pm, 3pm).
- Check slot availability and confirm booking. → View/Cancel Appointment:
- Allow patient to view or cancel their appointment using mobile number.

→ Doctor Availability:

- Maintain a maximum of 3 appointments per time slot per doctor.

→ Data Persistence:

- Store appointments in memory only (no files/dbs required).

```
class ClinicAppointment:
```

```
    def __init__(self):
```

```
        self.appointments = []
```

```
        self.slots = ["10am", "11am", "12pm", "2pm", "3pm"]
```

```
    def book_appointment(self):
```

```
        name = input("Patient Name: ")
```

```
        age = input("Age: ")
```

```
mobile = input("Mobile Number: ")
doctor = input("Doctor Name: ")

print("Available Time Slots:")
for s in self.slots:
    print(s)

slot = input("Choose Slot: ")

if slot not in self.slots:
    print("Invalid slot!")
    return

count = 0
for appt in self.appointments:
    if appt["doctor"] == doctor and appt["slot"] == slot:
        count += 1

if count >= 3:
    print("Slot full for this doctor!")
    return

self.appointments.append({
    "name": name,
    "age": age,
    "mobile": mobile,
```

```
        "doctor": doctor,  
        "slot": slot  
    })
```

```
print("Appointment Booked Successfully")
```

```
def view_appointment(self):  
    mobile = input("Enter Mobile Number: ")  
    found = False
```

```
    for appt in self.appointments:  
        if appt["mobile"] == mobile:  
            print("\nAppointment Details")  
            print("Name:", appt["name"])  
            print("Doctor:", appt["doctor"])  
            print("Slot:", appt["slot"])  
            found = True
```

```
    if not found:  
        print("No appointment found")
```

```
def cancel_appointment(self):  
    mobile = input("Enter Mobile Number: ")
```

```
    for appt in self.appointments:  
        if appt["mobile"] == mobile:
```

```
        self.appointments.remove(appt)
        print("Appointment Cancelled")
    return
    print("No appointment found")
```

```
def menu(self):
```

```
    while True:
```

```
        print("\n1. Book Appointment")
        print("2. View Appointment")
        print("3. Cancel Appointment")
        print("4. Exit")
```

```
    choice = input("Enter choice: ")
```

```
    if choice == "1":
```

```
        self.book_appointment()
```

```
    elif choice == "2":
```

```
        self.view_appointment()
```

```
    elif choice == "3":
```

```
        self.cancel_appointment()
```

```
    elif choice == "4":
```

```
        print("Thank You!")
```

```
        break
```

```
    else:
```

```
        print("Invalid Choice!")
```

School Management System

Design a Python class SchoolManagement that helps manage student admissions and records. The system should support:

→ New Admission:

- Collect student name, age, class (1–12), and guardian's mobile number.
- Assign a unique student ID automatically.
- Validate age: must be between 5 and 18.
- Validate mobile number: must be 10 digits.

→ View Student Details:

- Allow lookup using student ID.

→ Update Student Info:

- Update mobile number or class.

→ Remove Student Record:

- Remove a student using their student ID.

→ Exit System

```
class SchoolManagement:
```

```
    def __init__(self):
```

```
        self.students = []
```

```
        self.sid = 1
```

```
def new_admission(self):
    name = input("Name: ")
    age = int(input("Age: "))
    clas = int(input("Class (1-12): "))
    mobile = input("Mobile: ")

    if age < 5 or age > 18:
        print("Invalid Age")
        return

    if len(mobile) != 10:
        print("Invalid Mobile Number")
        return

    student = {
        "id": self.sid,
        "name": name,
        "age": age,
        "class": clas,
        "mobile": mobile
    }

    self.students.append(student)
    print("Admission Done. Student ID:", self.sid)
    self.sid += 1
```

```
def view_student(self):  
    sid = int(input("Enter Student ID: "))  
  
    for s in self.students:  
        if s["id"] == sid:  
            print("Name:", s["name"])  
            print("Age:", s["age"])  
            print("Class:", s["class"])  
            print("Mobile:", s["mobile"])  
            return  
  
    print("Student not found")  
  
def update_student(self):  
    sid = int(input("Enter Student ID: "))  
  
    for s in self.students:  
        if s["id"] == sid:  
            print("1. Update Mobile")  
            print("2. Update Class")  
            ch = input("Choice: ")  
  
            if ch == "1":  
                s["mobile"] = input("New Mobile: ")  
                print("Mobile Updated")  
            elif ch == "2":
```

```
s["class"] = int(input("New Class: "))
print("Class Updated")
return

print("Student not found")

def remove_student(self):
    sid = int(input("Enter Student ID: "))

    for s in self.students:
        if s["id"] == sid:
            self.students.remove(s)
            print("Student Removed")
            return

    print("Student not found")

def menu(self):
    while True:
        print("\n1. New Admission")
        print("2. View Student")
        print("3. Update Student")
        print("4. Remove Student")
        print("5. Exit")

        ch = input("Enter choice: ")
```

```
if ch == "1":
    self.new_admission()
elif ch == "2":
    self.view_student()
elif ch == "3":
    self.update_student()
elif ch == "4":
    self.remove_student()
elif ch == "5":
    break
else:
    print("Invalid Choice")
```

Transport Reservation System (Bus Ticketing)

Design a Python class BusReservation that simulates a basic bus ticket booking system. Features should include:

→ Show Available Routes:

- Predefined city routes with fixed prices.
- Example: "Mumbai to Pune - ₹500", "Delhi to Jaipur - ₹600", etc.

→ Book Ticket:

- Enter passenger name, age, mobile, and route.
- Assign seat number (max 40 per bus per route).
- Generate a unique ticket ID.

→ View Ticket:

- Lookup using ticket ID.

→ Cancel Ticket:

- Cancel the ticket if it exists.

→ Exit

```
class BusReservation:
```

```
    def __init__(self):
```

```
        self.routes = ["Mumbai-Pune", "Delhi-Jaipur"]
```

```
        self.tickets = []
```

```
        self.id = 1
```

```
        self.seat = 1
```

```
    def show_routes(self):
```

```
        print("Routes:")
```

```
        for r in self.routes:
```

```
            print(r)
```

```
def book_ticket(self):  
    name = input("Name: ")  
    route = input("Route: ")  
  
    ticket = {  
        "id": self.id,  
        "name": name,  
        "route": route,  
        "seat": self.seat  
    }  
  
    self.tickets.append(ticket)  
    print("Ticket Booked")  
    print("Ticket ID:", self.id)  
  
    self.id += 1  
    self.seat += 1  
  
def view_ticket(self):  
    tid = int(input("Ticket ID: "))  
  
    for t in self.tickets:  
        if t["id"] == tid:  
            print("Name:", t["name"])  
            print("Route:", t["route"])
```

```

        print("Seat:", t["seat"])
        return

    print("Ticket not found")

def cancel_ticket(self):
    tid = int(input("Ticket ID: "))

    for t in self.tickets:
        if t["id"] == tid:
            self.tickets.remove(t)
            print("Ticket Cancelled")
            return

    print("Ticket not found")

def menu(self):
    while True:
        print("\n1. Show Routes")
        print("2. Book Ticket")
        print("3. View Ticket")
        print("4. Cancel Ticket")
        print("5. Exit")
        ch = input("Choice: ")

        if ch == "1":

```

```
        self.show_routes()
elif ch == "2":
    self.book_ticket()
elif ch == "3":
    self.view_ticket()
elif ch == "4":
    self.cancel_ticket()
elif ch == "5":
    break
```