```
"""import numpy as np
import pandas as pd

room_types = np.array(["Standard", "Deluxe", "Suite"])
room_prices = np.array([1500, 2500, 4000])

columns = ["Customer Name", "Room Type", "Days", "Total Bill"]
hotel_data = pd.DataFrame(columns=columns)

def show_rooms():
    print("\nAvailable Rooms:")
    for i in range(len(room_types)):
        print(f"{i+1}. {room_types[i]} - ₹{room_prices[i]} per day")

def book_room():
    global hotel_data
    name = input("\nEnter customer name: ")
    show_rooms()

    choice = int(input("Choose room type (1-3): ")) - 1
    days = int(input("Enter number of days: "))

    total_bill = room_prices[choice] * days

    booking = {
        "Customer Name": name,
        "Room Type": room_types[choice],
        "Days": days,
        "Total Bill": total_bill
    }

    hotel_data = pd.concat([hotel_data, pd.DataFrame([booking])],
ignore_index=True)
    print(f"\nRoom booked successfully! Total Bill: ₹{total_bill}")

def view_bookings():
    if hotel_data.empty:
        print("\nNo bookings available.")
    else:
        print("\nHotel Booking Records:")
        print(hotel_data)

def main():
    while True:
        print("\n--- HOTEL MANAGEMENT SYSTEM ---")
        print("1. Show Rooms")
        print("2. Book Room")
        print("3. View Bookings")
        print("4. Exit")
```

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

        if choice == "1":
            show_rooms()
        elif choice == "2":
            book_room()
        elif choice == "3":
            view_bookings()
        elif choice == "4":
            print("Thank you for using the system!")
            break
        else:
            print("Invalid choice! Try again.")

main()"""

"""import matplotlib.pyplot as plt

def generate_room_occupancy_graph(room_counts):

    Generates a bar graph of room types and their counts.
    :param room_counts: A dictionary of counts for each room type.


    if not isinstance(room_counts, dict):
        print("Error: room_counts must be a dictionary.")
        return

    room_types = list(room_counts.keys())
    counts = list(room_counts.values())

    plt.figure(figsize=(8, 5))
    plt.bar(room_types, counts, color=['blue', 'green', 'red', 'purple'])
    plt.xlabel("Types of Rooms")
    plt.ylabel("Number of Rooms Booked")
    plt.title("Hotel Room Occupancy Analysis")
    plt.show()

sample_data = {
    'Single': 15,
    'Double': 22,
    'Suite': 8,
    'Deluxe': 12
}

generate_room_occupancy_graph(sample_data)"""

"""import random

roomno = []
```

```
custid = []

rn = random.randrange(40) + 300
cid = random.randrange(40) + 10

while rn in roomno or cid in custid:
    rn = random.randrange(60) + 300
    cid = random.randrange(60) + 10

roomno.append(rn)
custid.append(cid)

print(rn)
print(cid)"""

"""rooms = {room_num: None for room_num in range(101, 111)}

def display_rooms():
    print("\n--- Hotel Room Layout ---")
    for room, guest in rooms.items():
        status = "Available ☑" if guest is None else f"Occupied by {guest} ❌"
        print(f"Room {room}: {status}")
    print("-" * 25)

while True:
    print("\n=== 🏨 Welcome to Hotel ===")
    print("1. Book Room")
    print("2. Display Rooms")
    print("3. Exit")
    choice = input("Choose an option (1-3): ")

    if choice == '1':

        name = input("Enter guest name: ")
        room_num = int(input("Enter room number (101-110): "))
        if room_num in rooms and rooms[room_num] is None:
            rooms[room_num] = name
            print(f"☑ Room {room_num} booked for {name}.")
        else:
            print("⚠ Room unavailable or invalid number.")
    elif choice == '2':
        display_rooms()
    elif choice == '3':
        print("👌 Thank you for choosing our Hotel.")
        break
    else:
        print("⚠ Invalid choice. Try again.\n")"""


"""import datetime
```

```
rooms_data = {
    101: {"type": "Standard", "price": 100, "status": "Available"},
    102: {"type": "Deluxe", "price": 150, "status": "Occupied"},
    103: {"type": "Standard", "price": 100, "status": "Available"},
    104: {"type": "Suite", "price": 250, "status": "Available"},
}

def filter_available_rooms(room_type=None):

    Filters available rooms based on type.


    available_rooms = [
        room_id for room_id, details in rooms_data.items()
        if details["status"] == "Available" and (room_type is None or
details["type"] == room_type)
    ]
    return available_rooms

print("Available Standard rooms:", filter_available_rooms(room_type="Standard"))
print("All available rooms:", filter_available_rooms())"""

"""def book_room(name, phone_number, check_in, check_out, room_type, room_no):
    return f"Booking confirmed for {name} in room {room_no} ({room_type})."

result = book_room(
    name="sirisha",
    phone_number="555-1234",
    check_in="2025-12-25",
    check_out="2025-12-28",
    room_type="Suite AC",
    room_no=405
)

print(result)"""

"""def calculate_restaurant_bill(customer_name, **order_details):
    print(f"Order for {customer_name}:")
    total_bill = 0
    for item, price in order_details.items():
        print(f"- {item}: Rs. {price}")
        total_bill += price
    print(f"Total restaurant bill: Rs. {total_bill}")

calculate_restaurant_bill("sirisha", Masala_Dosa=130, Butter_Naan=20,
Paneer_Dosa=130)"""

"""class Room:
    def __init__(self, room_num, room_type, price):
```

```
        self.room_num = room_num
        self.room_type = room_type
        self.price = price
        self.is_available = True

class Guest:
    def __init__(self, name, contact):
        self.name = name
        self.contact = contact

class Booking:
    def __init__(self, guest, room):
        self.guest = guest
        self.room = room

class HotelManagementSystem:
    def __init__(self):
        self.rooms = []
        self.bookings = []

    def add_room(self, room_num, room_type, price):
        new_room = Room(room_num, room_type, price)
        self.rooms.append(new_room)

    def book_room(self, guest_name, contact, room_num):

        found_room = None
        for room in self.rooms:
            if room.room_num == room_num:
                found_room = room
                break

        if found_room is None:
            print(f"Error: Room number {room_num} not found.")
            return False


        if not found_room.is_available:
            print(f"Error: Room {room_num} is already booked.")
            return False


        new_guest = Guest(guest_name, contact)
        new_booking = Booking(new_guest, found_room)
        self.bookings.append(new_booking)
        found_room.is_available = False

        print(f"Success: Room {room_num} booked for {guest_name}.")
        return True
```

```
    def display_available_rooms(self):
        print("\nAvailable Rooms:")
        for room in self.rooms:
            if room.is_available:
                print(f"Room {room.room_num}: Type - {room.room_type}, Price -
${room.price}")
        if not any(room.is_available for room in self.rooms):
            print("No rooms are currently available.")

    def checkout_room(self, room_num):

        found_room = None
        for room in self.rooms:
            if room.room_num == room_num:
                found_room = room
                break

        if found_room is None:
            print(f"Error: Room number {room_num} not found.")
            return False


        if found_room.is_available:
            print(f"Error: Room {room_num} is already available (not booked).")
            return False


        booking_to_remove = None
        for booking in self.bookings:
            if booking.room.room_num == room_num:
                booking_to_remove = booking
                break

        if booking_to_remove:
            self.bookings.remove(booking_to_remove)
            found_room.is_available = True
            print(f"Success: Room {room_num} checked out.")
            return True
        else:

            print(f"Error: Internal data mismatch for room {room_num}.")
            return False



hotel_sys = HotelManagementSystem()
hotel_sys.add_room(101, "Single", 100)
hotel_sys.add_room(102, "Double", 150)
hotel_sys.add_room(201, "Suite", 300)
```

```
hotel_sys.display_available_rooms()

print("\nAttempting to book room 102...")
hotel_sys.book_room("mahalakshmi", "555-1234", 102)

print("\nAttempting to book room 102 again...")
hotel_sys.book_room("priyanka", "555-5678", 102)

print("\nAttempting to book room 201...")
hotel_sys.book_room("sirisha", "555-8765", 201)

hotel_sys.display_available_rooms()

print("\nAttempting to check out room 102...")
hotel_sys.checkout_room(102)

hotel_sys.display_available_rooms()"""

"""def generate_bill(room_rate, days_stayed, food_bill, laundry_bill):

    Calculates the total bill for a hotel stay using arithmetic operations.


    room_charges = room_rate * days_stayed


    service_charges = food_bill + laundry_bill


    tax_rate = 0.10
    tax_amount = (room_charges + service_charges) * tax_rate


    total_amount = room_charges + service_charges + tax_amount

    return total_amount


rate_per_night = 100
nights = 4
food = 75
laundry = 30

final_cost = generate_bill(rate_per_night, nights, food, laundry)
print(f"The total bill is: ${final_cost}")"""


"""DATA_FILE = 'hotel_data.txt'

def add_booking_file(room_no, name, checkin, checkout, total_cost):
```

```
    record = f'{room_no},{name},{checkin},{checkout},{total_cost}\\n'


    with open(DATA_FILE, 'a', encoding='utf-8') as f:
        f.write(record)
    print('Booking saved to file.')

def view_bookings_file():
    records = []
    try:

        with open(DATA_FILE, 'r', encoding='utf-8') as f:
            for line in f:
                records.append(line.strip().split(','))
    except FileNotFoundError:
        print("No booking records found.")
        return

    for r in records:
        print(f"Room {r[0]} | Name: {r[1]} | Dates: {r[2]} to {r[3]} | Total:
${r[4]}")

add_booking_file('101', 'priya', '2025-12-28', '2025-12-30', '400')

view_bookings_file()"""
```