Core Python Assignment

December 25, 2024

# E-Commerce Cart System

**Problem Statement:** Write a program to calculate the total price of items in a user’s cart. If the cart contains more than 5 items, apply 10% discount.

## Solution:

1

def calculate\_total\_price(cart\_items): if not cart\_items:

return "The cart is empty." total\_price = sum(cart\_items.values())

if len(cart\_items) > 5:

total\_price \*= 0.9 # Apply 10% discount return total\_price

# Example usage

cart\_items = {’Laptop’: 50000, ’Headphones’: 2000, ’Mouse’: 500, ’Keyboard’: 1500} total\_price = calculate\_total\_price(cart\_items)

print(f"Total Price: {total\_price}")

2

3

4

5

6

7

8

9

10

11

12

13

14

15

# Restaurant Menu Management

**Problem Statement:** Write a program to update the menu by adding or removing items and allow users to check if a particular item is available.

## Solution:

1

def add\_item(menu, item): menu.append(item)

def remove\_item(menu, item): if item in menu,

menu.remove(item) else:

print(f"{item} is not in the menu.")

def check\_item(menu, item): return item in menu

# Example usage

initial\_menu = ["Pizza", "Burger", "Pasta", "Salad"] add\_item(initial\_menu, "Tacos") remove\_item(initial\_menu, "Salad")

availability = check\_item(initial\_menu, "Pizza") print(f"Updated menu: {initial\_menu}")

print(f"Availability: {’Pizza is available’ if availability else ’Pizza is not available’}")

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

# Classroom Performance Tracker

**Problem Statement:** Write a program to calculate the average marks of students and identify the top performer.

## Solution:

1

def calculate\_average\_marks(students):

averages = {student: sum(marks)/len(marks) for student, marks in students.items()} return averages

def identify\_top\_performer(averages): return max(averages, key=averages.get)

# Example usage

students = {"John": [85, 78, 92], "Alice": [88, 79, 95], "Bob": [70, 75, 80]}

averages = calculate\_average\_marks(students) top\_performer = identify\_top\_performer(averages) print(f"Average Marks: {averages}")

print(f"Top Performer: {top\_performer}")

2

3

4

5

6

7

8

9

10

11

12

13

# Movie Ticket Booking System

**Problem Statement:** Write a program to keep track of available and booked seats. Allow users to book or cancel a seat.

## Solution:

1

def book\_seat(total\_seats, booked\_seats, seat\_number): if seat\_number in booked\_seats:

return f"Seat {seat\_number} is already booked." else:

booked\_seats.append(seat\_number)

return f"Seat {seat\_number} successfully booked."

def cancel\_seat(booked\_seats, seat\_number): if seat\_number in booked\_seats:

booked\_seats.remove(seat\_number)

return f"Seat {seat\_number} successfully cancelled." else:

return f"Seat {seat\_number} is not booked."

def available\_seats(total\_seats, booked\_seats):

return [seat for seat in range(1, total\_seats + 1) if seat not in booked\_seats]

# Example usage total\_seats = 10

booked\_seats = [2, 5, 7]

print(book\_seat(total\_seats, booked\_seats, 3))

print(cancel\_seat(booked\_seats, 5))

print(f"Available seats: {available\_seats(total\_seats, booked\_seats)}")

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

# Hospital Patient Management

**Problem Statement:** Write a program to store patient data, including name, age, and disease, and allow the admin to search for patients by disease.

## Solution:

1

def search\_patients\_by\_disease(patients, disease):

return [patient["Name"] for patient in patients if patient["Disease"].lower() == disease.lower()]

2

3

# Example usage patients = [

{"Name": "Alice", "Age": 30, "Disease": "Flu"},

{"Name": "Bob", "Age": 45, "Disease": "Diabetes"},

{"Name": "Charlie", "Age": 35, "Disease": "Flu"}

]

search\_disease = "Flu"

print(f"Patients with {search\_disease}: {search\_patients\_by\_disease(patients, search\_disease)}")

4

5

6

7

8

9

10

11

# Customer Feedback Analysis

**Problem Statement:** Write a program to calculate the percentage of positive feedback (4 or 5).

## Solution:

1

def calculate\_positive\_feedback\_percentage(ratings): if not ratings:

return "No ratings available."

positive\_ratings = [rating for rating in ratings if rating >= 4] percentage = (len(positive\_ratings) / len(ratings)) \* 100

return percentage

# Example usage

ratings = [5, 4, 3, 5, 2, 4, 1, 5]

positive\_feedback = calculate\_positive\_feedback\_percentage(ratings) print(f"Positive Feedback: {positive\_feedback:.1f}%")

2

3

4

5

6

7

8

9

10

11

12

# Taxi Fare Calculation

**Problem Statement:** Write a program to calculate the total fare for multiple trips.

## Solution:

1

def calculate\_fare(trip\_distance): base\_fare = 50

distance\_fare = 10 \* trip\_distance total\_fare = base\_fare + distance\_fare return total\_fare

# Example usage

trips = [5, 10, 3] # Distances in km

total\_fare = 0

for i, trip in enumerate(trips): fare = calculate\_fare(trip) total\_fare += fare print(f"Trip {i+1}: ${fare}")

print(f"Total Fare: ${total\_fare}")

2

3

4

5

6

7

8

9

10

11

12

13

14