**Lab#12**

**Questions**

Q # 1: Write a program that Allows a user add records in a text file. The program asks the user for a record in the following format: Name, Phone number and then saves this record with a unique ID. (Starting from 1) in a comma separated Format in the file.

Input:

Def add\_record(file\_name):

Try:

With open(file\_name, ‘a’) as file:

Name = input(“Enter name: “)

Phone\_number = input(“Enter phone numbers”)

With open(file\_name, ‘r’) as f:

Num\_records = sum(1 for line in f)

File.write(f”{num\_records + 1}, {name}, {phone\_number}\n”)

Print(“Record added successfully!”)

Except IOError:

Print(“Error: Unable to write to file.”)

File\_name = “records.txt”

Add\_record(file\_name)

Output:

Enter name: Jawaria

Enter phone number: 0333333390

Record added successfully!

[Process completed – press Enter]

Q # 2: write a program that implements a small-scale security surveillance system. The Program allows the user to enter the name, phone number and then save this information into a visitor log File along with the arrival time information.

Input:

Import datetime

Def log\_visitor(file\_name):

Try:

With open(file\_name, ‘a’) as file:

Name = input(“Enter visitor’s name: “)

Phone\_number = input(“Enter visitor’s phone number: “)

Arrival\_time = datetime.datetime.now().strftime(“%Y-%m-%d %H:%M:%S”)

File.write(f”{name}, {phone\_number}, {arrival\_time}\n”)

Print(“Visitor logged successfully!”)

Except IOError:

Print(“Error: Unable to write to file.”)

File\_name = “visitor\_log.txt”

Log\_visitor(file\_name)

Output:

Enter visitor’s name: Jawaria

Enter visitor’s phone number: 033333390

Visitor logged successfully!

[Process completed – press Enter]

Q # 3: Write a program that extends the car rental system from Lab no. 7. Use files to store the information about the cars. Moreover, the car rental system should also allow logging of the car Rentals in a separate file i.e., saving information (appending) to a file each time a car is rented and Updating the information when it is returned.

Input:

Class Car:

Def \_\_init\_\_(self, car\_id, make, model, year, is\_rented=False):

Self.car\_id = car\_id

Self.make = make

Self.model = model

Self.year = year

Self.is\_rented = is\_rented

Def \_\_str\_\_(self):

Return f”{self.make} {self.model} ({self.year})”

Class CarRentalSystem:

Def \_\_init\_\_(self):

Self.cars = []

Self.load\_cars\_from\_file(“cars.txt”)

Def load\_cars\_from\_file(self, file\_name):

Try:

With open(file\_name, ‘r’) as file:

For line in file:

Car\_id, make, model, year, is\_rented = line.strip().split(‘,’)

Car = Car(int(car\_id), make, model, int(year), is\_rented == “True”)

Self.cars.append(car)

Except FileNotFoundError:

Print(“File not found. Creating a new car file.”)

Self.save\_cars\_to\_file(file\_name)

Def save\_cars\_to\_file(self, file\_name):

Try:

With open(file\_name, ‘w’) as file:

For car in self.cars: File.write(f”{car.car\_id},{car.make},{car.model},{car.year},{car.is\_rented}\n”)

Except IOError:

Print(“Error: Unable to write to file.”)

Def rent\_car(self, car\_id, customer\_name):

For car in self.cars:

If car.car\_id == car\_id and not car.is\_rented:

Car.is\_rented = True

Self.save\_cars\_to\_file(“cars.txt”)

Self.log\_rental(“rentals.txt”, car, customer\_name)

Return True

Return False

Def return\_car(self, car\_id):

For car in self.cars:

If car.car\_id == car\_id and car.is\_rented:

Car.is\_rented = False

Self.save\_cars\_to\_file(“cars.txt”)

Self.update\_rental\_log(“rentals.txt”, car\_id)

Return True

Return False

Def log\_rental(self, file\_name, car, customer\_name):

Try:

With open(file\_name, ‘a’) as file: File.write(f”{car.car\_id},{customer\_name},rented\n”)

Except IOError:

Print(“Error: Unable to write to rental log file.”)

Def update\_rental\_log(self, file\_name, car\_id):

Try:

With open(file\_name, ‘r’) as file:

Lines = file.readlines()

With open(file\_name, ‘w’) as file:

For line in lines:

Parts = line.strip().split(‘,’)

If int(parts[0]) == car\_id:

Parts[-1] = “returned”

Line = ‘,’.join(parts) + ‘\n’

File.write(line)

Except IOError:

Print(“Error: Unable to update rental log file.”)

Def display\_available\_cars(self):

Available\_cars = [car for car in self.cars if not car.is\_rented]

If available\_cars:

Print(“Available Cars:”)

For car in available\_cars:

Print(f”{car.car\_id}: {car}”)

Else:

Print(“No cars available for rent.”)

Def display\_rented\_cars(self):

Rented\_cars = [car for car in self.cars if car.is\_rented]

If rented\_cars:

Print(“Rented Cars:”)

For car in rented\_cars:

Print(f”{car.car\_id}: {car}”)

Else:

Print(“No cars are currently rented.”)

Car\_rental\_system = CarRentalSystem()

Car\_rental\_system.display\_available\_cars()

Car\_id\_to\_rent = int(input(“Enter the car ID to rent: “))

Customer\_name = input(“Enter your name: “)

If car\_rental\_system.rent\_car(car\_id\_to\_rent, customer\_name):

Print(“Car rented successfully!”)

Else:

Print(“Car not available for rent or invalid car ID.”)

Car\_id\_to\_return = int(input(“Enter the car ID to return: “))

If car\_rental\_system.return\_car(car\_id\_to\_return):

Print(“Car returned successfully!”)

Else:

Print(“Car not currently rented or invalid car ID.”)

Output:

No cars available for rent.

Enter the car ID to rent: 123456

Enter your name: Jawaria

Car not available for rent or invalid car ID.

Enter the car ID to return:

**Questions**

Q # 1: What is the difference between opening a file in binary or text mode?

Ans: The main difference between opening a file in binary mode and text mode is that text mode performs newline translation, converting platform-specific newline characters, while binary mode does not perform any translation and reads/writes data exactly as it is stored in the file.

Q # 2: A file “test.txt” has some data and is stored in the same folder as your script. Write some code to open The file and find a line containing “Total cost:”. Then, from this line, read the numbers after the term Total cost” and print the information on the screen.

Ans:

File\_path = “test.txt”

Try:

With open(file\_path, ‘r’) as file:

For line in file:

If “Total cost:” in line:

# Extract the numbers after “Total cost:”

Total\_cost\_index = line.index(“Total cost:”)

Total\_cost\_value = line[total\_cost\_index + len(“Total cost:”):].strip()

Print(“Total cost:”, total\_cost\_value)

Break

Else:

Print(“Total cost not found in the file.”)

Except FileNotFoundError:

Print(f”Error: File ‘{file\_path}’ not found.”)

Except Exception as e:

Print(“An error occurred:”, e)

Q # 3: Write some to compare two files “file1.dat” and “file2.dat”. The program should print only those lines which are different in both files.

Ans: file1\_path = “file1.dat”

File2\_path = “file2.dat”

Try:

With open(file1\_path, ‘r’) as file1, open(file2\_path, ‘r’) as file2:

Lines1 = file1.readlines()

Lines2 = file2.readlines()

# Compare lines from both files

For line\_num, (line1, line2) in enumerate(zip(lines1, lines2), start=1):

If line1 != line2:

Print(f”Line {line\_num}:\nFile 1: {line1.strip()}\nFile 2: {line2.strip()}”)

Except FileNotFoundError as e:

Print(f”Error: {e.strerror}”)

Except Exception as e:

Print(“An error occurred:”, e)

Name: Jawaria

Roll : 23BS(AI)39

Date: 3-march-2024