**ระบบจัดการการเช่ารถ**

**Car Rental Management System**

**นางสาวพัตตรีมา โพธิสัตย์ รหัส 6806022510360 Sec3**

**นายธนบดี เอี่ยมสะอาด รหัส 6806022510386 Sec3**

**นายจิรวัฒน์ เชิดสูงเนิน รหัส 6806022510505 Sec3**

**นายมีสุข ดีเสมอ รหัส 6806022510513 Sec3**

**โครงการสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาตรี**

**สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ**

**คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ**

**ปีการศึกษา 2568**

**ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ**

คำนำ

การจัดทำโครงงาน “ระบบจัดการการเช่ารถ” ฉบับนี้ เป็นส่วนหนึ่งของรายวิชา Computer Programming ในหลักสูตรปริญญาตรี สาขาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ โดยมีวัตถุประสงค์เพื่อให้นักศึกษาได้ฝึกฝน และประยุกต์ใช้ความรู้ด้านการเขียนโปรแกรมในการพัฒนาระบบที่สามารถใช้งานได้จริง

โครงงานนี้มุ่งเน้นการออกแบบและพัฒนาโปรแกรมด้วยภาษา Python ซึ่งเป็นภาษาที่ใช้ในการเรียนการสอนรายวิชา Computer Programming โดยตรง ทั้งนี้จะช่วยเสริมสร้างทักษะการคิดวิเคราะห์ การออกแบบเชิงระบบ และการแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมให้แก่นักศึกษาในการประกอบวิชาชีพด้านวิศวกรรมสารสนเทศ และเครือข่ายต่อไปในอนาคต

คณะผู้จัดทำหวังเป็นอย่างยิ่งว่า รายงานฉบับนี้จะเป็นประโยชน์ต่อผู้ที่สนใจศึกษา นักเรียน นักศึกษา ตลอดจนผู้ที่ต้องการค้นคว้าหาความรู้เกี่ยวกับการพัฒนาโปรแกรมระบบงาน หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด คณะผู้จัดทำขอน้อมรับไว้ด้วยความยินดี และขออภัยมา ณ ที่นี้

สารบัญ

หน้า

[คำนำ ก](#_Toc210530364)

[สารบัญ ข](#_Toc210530365)

[สารบัญ(ต่อ) ค](#_Toc210530366)

[สารบัญภาพ ง](#_Toc210530367)

[สารบัญภาพ(ต่อ) จ](#_Toc210530368)

[สารบัญภาพ(ต่อ) ฉ](#_Toc210530369)

[สารบัญตาราง ช](#_Toc210530370)

[บทที่ 1 บทนำ 1](#_Toc210530371)

[1.1 วัตถุประสงค์ของโครงการ 1](#_Toc210530372)

[1.2 ขอบเขตของโครงการ 1](#_Toc210530373)

[1.3 ประโยชน์ที่ได้รับ 2](#_Toc210530374)

[1.4 เครื่องมือที่คาดว่าจะต้องใช้ 2](#_Toc210530375)

[บทที่ 2 ระบบจัดการการเช่ารถ 3](#_Toc210530376)

[2.1 แฟ้มเก็บข้อมูลรถ cars.dat 3](#_Toc210530377)

[2.2 แฟ้มเก็บข้อมูลลูกค้า customers.dat 5](#_Toc210530378)

[2.3 แฟ้มเก็บข้อมูลการเช่า rentals.dat 7](#_Toc210530379)

[2.4 รายงานสรุป Report.txt 9](#_Toc210530380)

[บทที่ 3 ขั้นตอนการดำเนินงาน 11](#_Toc210530381)

[3.1 การจัดการข้อมูลรถ (Cars Management) 11](#_Toc210530382)

[3.2 การจัดการข้อมูลลูกค้า (Customer Management) 16](#_Toc210530383)

[3.3 จัดการข้อมูลการเช่า (Rentals Management) 18](#_Toc210530384)

[บทที่ 4 ผลการดำเนินงาน 20](#_Toc210530385)

[4.1 ฟังก์ชันในงานพื้นฐานในระบบเช่ารถ 20](#_Toc210530386)

[4.2 ฟังก์ชัน Pack และ Unpack 25](#_Toc210530387)

[4.3 ฟังก์ชันจัดการข้อมูล (CRUD) 28](#_Toc210530388)

[4.4 ฟังก์ชันการสร้างรายงาน 33](#_Toc210530389)

[4.5 ฟังก์ชันสร้างข้อมูลตัวอย่าง (Sample Data Generators) 37](#_Toc210530390)

[4.6 ฟังก์ชันเมนูหลัก (Main Menu และ Sub-menu) 39](#_Toc210530391)

สารบัญ(ต่อ)

หน้า

[4.7 สรุปการทำงานของระบบทั้งหมด 42](#_Toc210530392)

[บทที่ 5 สรุปผลการดำเนินงานและข้อเสนอแนะ 46](#_Toc210530393)

[5.1 สรุปผลการดำเนินงาน 46](#_Toc210530394)

[5.2 ปัญหาและอุปสรรคในการทำงาน 46](#_Toc210530395)

[5.3 ข้อเสนอแนะ 46](#_Toc210530396)

[5.4 สิ่งที่ผู้จัดทำได้รับในการพัฒนาโครงการ 47](#_Toc210530397)

สารบัญภาพ

หน้า

[รูปภาพที่ 2‑1ไฟล์ Report 9](#_Toc210529985)

[รูปภาพที่ 3‑1 ตัวอย่างหน้าโปรแกรมระบบจัดการการเช่ารถ 11](#_Toc210529986)

[รูปภาพที่ 3‑2 ตัวอย่างรูปแบบการรับข้อมูลฟิลด์ Add car 12](#_Toc210529987)

[รูปภาพที่ 3‑3 หน้าแสดงข้อความยืนยันการเพิ่มข้อมูลรถ 12](#_Toc210529988)

[รูปภาพที่ 3‑4 หน้าแสดงการแก้ไขข้อมูลรถ 13](#_Toc210529989)

[รูปภาพที่ 3‑5 หน้าแสดงการลบข้อมูลรถ 13](#_Toc210529990)

[รูปภาพที่ 3‑6 หน้าแสดงการดูข้อมูลรถ 14](#_Toc210529991)

[รูปภาพที่ 3‑7 หน้าแสดงการดูข้อมูลรถ View one (a) 14](#_Toc210529992)

[รูปภาพที่ 3‑8 หน้าแสดงการดูข้อมูลรถ View one (b) 15](#_Toc210529993)

[รูปภาพที่ 3‑9 หน้าแสดงการดูข้อมูลรถ View one (c) 15](#_Toc210529994)

[รูปภาพที่ 3‑10 หน้าแสดงการดูข้อมูลรถ View one (d) 16](#_Toc210529995)

[รูปภาพที่ 3‑11 หน้าการเพิ่มข้อมูลลูกค้า 16](#_Toc210529996)

[รูปภาพที่ 3‑12 หน้าการดูข้อมูลลูกค้า 17](#_Toc210529997)

[รูปภาพที่ 3‑13 หน้าการเพิ่มข้อมูลการเช่ารถ 18](#_Toc210529998)

[รูปภาพที่ 3‑14 หน้าการดูข้อมูลการเช่ารถ 19](#_Toc210529999)

[รูปภาพที่ 4‑1 struct 20](#_Toc210530000)

[รูปภาพที่ 4‑2 pack และ unpack car 21](#_Toc210530001)

[รูปภาพที่ 4‑3 pack และ unpack car 21](#_Toc210530002)

[รูปภาพที่ 4‑4 debug 22](#_Toc210530003)

[รูปภาพที่ 4‑5 ensure 22](#_Toc210530004)

[รูปภาพที่ 4‑6 write และ read 23](#_Toc210530005)

[รูปภาพที่ 4‑7 get\_record 23](#_Toc210530006)

[รูปภาพที่ 4‑8 append 24](#_Toc210530007)

[รูปภาพที่ 4‑9 write\_record 24](#_Toc210530008)

[รูปภาพที่ 4‑10 find 24](#_Toc210530009)

[รูปภาพที่ 4‑11 Time 25](#_Toc210530010)

สารบัญภาพ(ต่อ)

หน้า

[รูปภาพที่ 4‑12 datetime 25](#_Toc210530011)

[รูปภาพที่ 4‑13 Pack and unpack (car) 26](#_Toc210530012)

[รูปภาพที่ 4‑14 Pack and unpack (customer) 27](#_Toc210530013)

[รูปภาพที่ 4‑15 Pack and unpack (rentel) 27](#_Toc210530014)

[รูปภาพที่ 4‑16add\_car 28](#_Toc210530015)

[รูปภาพที่ 4‑17 update\_car\_interactive 29](#_Toc210530016)

[รูปภาพที่ 4‑18 delete\_car 30](#_Toc210530017)

[รูปภาพที่ 4‑19 view\_all\_cars 30](#_Toc210530018)

[รูปภาพที่ 4‑20 add\_customer 31](#_Toc210530019)

[รูปภาพที่ 4‑21 view\_all\_customers 31](#_Toc210530020)

[รูปภาพที่ 4‑22 add\_rental 32](#_Toc210530021)

[รูปภาพที่ 4‑23 view\_all\_rentals 32](#_Toc210530022)

[รูปภาพที่ 4‑24 generate\_report 33](#_Toc210530023)

[รูปภาพที่ 4‑25 โหลดข้อมูล 33](#_Toc210530024)

[รูปภาพที่ 4‑26 Report A 34](#_Toc210530025)

[รูปภาพที่ 4‑27 Output Report A 34](#_Toc210530026)

[รูปภาพที่ 4‑28 Report B 35](#_Toc210530027)

[รูปภาพที่ 4‑29 Output Report B 35](#_Toc210530028)

[รูปภาพที่ 4‑30 Report C 35](#_Toc210530029)

[รูปภาพที่ 4‑31 Output Report C and Car Summary 36](#_Toc210530030)

[รูปภาพที่ 4‑32 การเขียนไฟล์ Report 36](#_Toc210530031)

[รูปภาพที่ 4‑33 sample\_cars 37](#_Toc210530032)

[รูปภาพที่ 4‑34 sample\_customers 38](#_Toc210530033)

[รูปภาพที่ 4‑35 Main Menu 39](#_Toc210530034)

[รูปภาพที่ 4‑36 main\_loop 40](#_Toc210530035)

[รูปภาพที่ 4‑37 Sample Data Menu 41](#_Toc210530036)

สารบัญภาพ(ต่อ)

หน้า

[รูปภาพที่ 4‑38 Customers Menu 41](#_Toc210530037)

[รูปภาพที่ 4‑39 Rentals Menu 42](#_Toc210530038)

สารบัญตาราง

หน้า

[**ตารางที่ 2‑1** แฟ้มข้อมูลรถ 3](#_Toc210528757)

[**ตารางที่ 2‑2** แฟ้มข้อมูลลูกค้า 5](#_Toc210528758)

[**ตารางที่ 2‑3** แฟ้มข้อมูลการเช่า 7](#_Toc210528759)

# บทนำ

## วัตถุประสงค์ของโครงการ

1.1.1 เพื่อพัฒนาระบบจัดการการเช่ารถให้สามารถทำงานได้อย่างมีประสิทธิภาพ

1.1.2 เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วยภาษา Python

1.1.3 เพื่อเรียนรู้วิธีการจัดการข้อมูลและไฟล์แบบ Binary File I/O โดยใช้โครงสร้างข้อมูล (struct)

1.1.4 เพื่อเรียนรู้การทำงานร่วมกันเป็นทีมขอบเขตของโครงงาน

## ขอบเขตของโครงการ

1.2.1 ระบบจัดการการเช่ารถ มีฟังก์ชันหลัก ๆ ดังนี้:

1.2.1.1 เพิ่มรถ (Add Car)

1.2.1.2 แก้ไขข้อมูลรถ (Update Car)

1.2.1.3 ลบรถ (Logical Delete Car)

1.2.1.4 ดูข้อมูลรถ (View Cars: All, Active, Deleted, One)

1.2.1.5 เพิ่มลูกค้า (Add Customer)

1.2.1.6 ดูรายชื่อลูกค้า (View All Customers)

1.2.1.7 เพิ่มรายการเช่า (Add Rental)

1.2.1.8 ดูรายการเช่าทั้งหมด (View All Rentals)

1.2.1.9 สร้างข้อมูลตัวอย่าง (Create Sample Data)

1.2.1.10 สร้างรายงาน (Generate Report)

1.2.1.11 เมนูหลักและเมนูย่อยสำหรับดำเนินการต่าง ๆ

1.2.1.12 ออกจากโปรแกรม

1.2.2 ระบบจัดการการเช่ารถ ประกอบด้วย 3 ไฟล์ข้อมูลหลักที่เป็นไฟล์ไบนารี ได้แก่:

1.2.2.1 ไฟล์ข้อมูลรถ cars.dat

1.2.2.2 ไฟล์ข้อมูลลูกค้า customers.dat

1.2.2.3 ไฟล์ข้อมูลการเช่า rentals.dat

1.2.2.4 ไฟล์รายงานที่เป็น Text File ชื่อ report.txt

1.2.3 ระบบจัดการการเช่ารถ มีการจัดเก็บข้อมูลในไฟล์ไบนารีรูปแบบ Fixed-length Record และมีส่วน Header สำหรับจัดเก็บข้อมูลเมตา (Metadata)

1.2.4 ระบบจัดการการเช่ารถ เป็นโปรแกรมที่ทำงานผ่าน Command Line Interface (CLI) ที่มีเมนูให้ผู้ใช้สามารถเลือกดำเนินการได้

## ประโยชน์ที่ได้รับ

1.3.1 พัฒนาระบบที่สามารถใช้ในการจัดการการเช่ารถได้อย่างมีประสิทธิภาพ

1.3.2 พัฒนาทักษะการเขียนโปรแกรมด้วยภาษา Python และการประยุกต์ใช้ Python Standard Library

1.3.3 เรียนรู้การจัดการข้อมูลและไฟล์ในรูปแบบไบนารี (Binary File I/O) และการใช้โมดูล struct ในการจัดเรียงข้อมูล

1.3.4 เรียนรู้การทำงานร่วมกันเป็นทีมในการพัฒนาซอฟต์แวร์

## เครื่องมือที่คาดว่าจะต้องใช้

1.4.1 โปรแกรม Visual Studio Code หรือ Editor อื่น ๆ ที่รองรับภาษา Python

1.4.2 โปรแกรม Microsoft Office สำหรับจัดทำเอกสาร

# **ระบบจัดการการเช่ารถ**

## แฟ้มเก็บข้อมูลรถ cars.dat

แฟ้มข้อมูลรถประกอบด้วย 11 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญ ดังนี้

|  |  |  |  |
| --- | --- | --- | --- |
| ฟิลด์ | ชนิด | ขนาด (bytes) | ตัวอย่าง |
| car\_id | i | 4 | รหัสรถ |
| status | i | 4 | สถานะ Active=1, Deleted=0 |
| is\_rented | i | 4 | การเช่า Rented=1, Available=0 |
| year | i | 4 | ปีที่ผลิต |
| daily\_rate\_thb | f | 4 | ค่าเช่าต่อวัน (บาท) |
| odometer\_km | i | 4 | เลขไมล์ |
| license\_plate | 12s | 12 | ทะเบียนรถ |
| brand | 12s | 12 | ยี่ห้อ |
| model | 16s | 16 | รุ่นรถ |
| created\_at | i | 4 | timestamp สร้าง record |
| updated\_at | i | 4 | timestamp แก้ไข record |

**ตารางที่ 2‑1** แฟ้มข้อมูลรถ

2.1.1 car\_id รหัสรถ

car\_id เป็นรหัสรถที่ใช้ในการระบุรถแต่ละคันไม่ให้ซ้ำกัน ฟิลด์นี้ถูกสร้างขึ้นในรูปแบบตัวเลขจำนวนเต็ม (integer) เช่น 1001, 1002, 1003 เป็นต้น การมีรหัสรถที่ไม่ซ้ำกันช่วยหลีกเลี่ยงความสับสนระหว่างรถหลายคัน และช่วยให้สามารถค้นหา เรียกดู และอ้างอิงข้อมูลรถได้อย่างแม่นยำและรวดเร็ว โดยเฉพาะอย่างยิ่งเมื่อมีข้อมูลรถจำนวนมากในระบบ

2.1.2 status สถานะของรถ

status ใช้เก็บสถานะของระเบียนรถ โดยเป็นตัวเลขจำนวนเต็ม (integer) กำหนดค่า 1 หมายถึงข้อมูลยัง Active และ 0 หมายถึงข้อมูลถูกลบ (Deleted) การใช้สถานะแบบนี้ช่วยให้ระบบสามารถจัดการลบข้อมูลแบบ Logical Delete ได้โดยไม่ต้องลบระเบียนจริงในแฟ้ม ซึ่งมีประโยชน์ต่อ  
การบันทึกประวัติและสามารถกู้คืนข้อมูลได้หากจำเป็น  
 2.1.3 is\_rented สถานะการเช่า

is\_rented เป็นค่าจำนวนเต็ม (integer) ที่ใช้กำหนดสถานะการเช่าของรถ หากค่าเป็น 1 หมายถึงรถกำลังถูกเช่าอยู่ และถ้าเป็น 0 หมายถึงรถว่างพร้อมให้เช่า ฟิลด์นี้มีความสำคัญในการตรวจสอบการใช้งานรถในปัจจุบัน และป้องกันไม่ให้รถคันเดียวกันถูกจองหรือปล่อยเช่าซ้ำซ้อน

2.1.4 year ปีที่ผลิต

year เก็บข้อมูลปีที่รถถูกผลิตออกมา ใช้ชนิดจำนวนเต็ม (integer) เช่น 2020, 2021 เป็นต้น การมีข้อมูลปีที่ผลิตช่วยให้ผู้ใช้หรือเจ้าหน้าที่ทราบอายุการใช้งานของรถ ซึ่งมีผลต่อราคาค่าเช่า และการบำรุงรักษา

2.1.5 daily\_rate\_thb ค่าเช่าต่อวัน

daily\_rate\_thb เป็นตัวเลขชนิดทศนิยม (float) ที่ใช้เก็บราคาค่าเช่ารถต่อวันในหน่วยบาท เช่น 1200.0, 1500.0 การกำหนดค่าเช่าต่อวันเป็นฟิลด์เฉพาะช่วยให้ระบบสามารถคำนวณค่าบริการรวมได้สะดวกเมื่อลูกค้าเช่ารถหลายวัน

2.1.6 odometer\_km เลขไมล์

odometer\_km ใช้เก็บตัวเลขจำนวนเต็ม (integer) ที่บอกระยะทางการใช้งานของรถ (กิโลเมตร) เช่น 35000, 78000 เป็นต้น ฟิลด์นี้มีความสำคัญต่อการตรวจสอบสภาพรถ การบำรุงรักษาตามระยะ และการประเมินความคุ้มค่าในการปล่อยเช่า

2.1.7 license\_plate ทะเบียนรถ

license\_plate ใช้เก็บข้อมูลตัวอักษร (string) แบบความยาวคงที่ 12 ไบต์ สำหรับบันทึกหมายเลขทะเบียนรถ เช่น "ABC-1234" การเก็บทะเบียนรถทำให้สามารถระบุตัวตนรถคันจริงได้ตรงกับเอกสารทางราชการ ป้องกันความผิดพลาดในการเช่าและการบันทึกข้อมูล

2.1.8 brand ยี่ห้อรถ

brand เป็นฟิลด์ข้อความ (string) ความยาวคงที่ 12 ไบต์ ใช้เก็บยี่ห้อรถ เช่น "Toyota", "Honda" การมีข้อมูลยี่ห้อช่วยให้ผู้เช่าเลือกได้ตามความต้องการและเป็นข้อมูลสำคัญต่อการวิเคราะห์ความนิยมของรถแต่ละยี่ห้อในระบบ

2.1.9 model รุ่นรถ

model เป็นฟิลด์ข้อความ (string) ความยาวคงที่ 16 ไบต์ ใช้เก็บชื่อรุ่นของรถ เช่น "Vios", "Civic" การมีข้อมูลรุ่นช่วยระบุรายละเอียดรถได้ชัดเจนมากขึ้น เพราะรถแต่ละยี่ห้ออาจมีหลายรุ่นย่อยที่แตกต่างกันทั้งรูปลักษณ์และสมรรถนะ

2.1.10 created\_at เวลาที่สร้างระเบียน

created\_at ใช้เก็บค่าเวลาในรูปแบบ timestamp (ชนิดจำนวนเต็ม integer) เพื่อบันทึกเวลาที่มีการสร้างข้อมูลรถระเบียนนั้น ๆ ฟิลด์นี้ช่วยให้สามารถตรวจสอบได้ว่าข้อมูลรถถูกเพิ่มเข้าสู่ระบบตั้งแต่เมื่อไร และสามารถใช้ในการจัดทำรายงานหรือสถิติได้

2.1.11 updated\_at เวลาที่แก้ไขล่าสุด

updated\_at ใช้เก็บค่าเวลาในรูปแบบ timestamp (ชนิดจำนวนเต็ม integer) เช่นเดียวกับ created\_at แต่บันทึกเฉพาะเวลาที่มีการแก้ไขหรืออัปเดตข้อมูลล่าสุดของรถ ฟิลด์นี้มีความสำคัญในการติดตามการเปลี่ยนแปลงของข้อมูล ช่วยเพิ่มความถูกต้อง และความน่าเชื่อถือในการจัดการข้อมูล

## แฟ้มเก็บข้อมูลลูกค้า customers.dat

แฟ้มข้อมูลลูกค้าประกอบด้วยฟิลด์หลัก 10 ฟิลด์ ใช้สำหรับบันทึกข้อมูลของผู้เช่ารถ เพื่อ  
ให้ระบบสามารถจัดการการเช่าได้อย่างถูกต้องและสะดวก

|  |  |  |  |
| --- | --- | --- | --- |
| ฟิลด์ | ชนิด | ขนาด (bytes) | ตัวอย่าง |
| cust\_id | i | 4 | รหัสลูกค้า |
| status | i | 4 | สถานะ Active=1, Deleted=0 |
| citizen\_id | 16s | 16 | เลขบัตรประชาชน |
| name | 32s | 32 | ชื่อลูกค้า |
| surname | 32s | 32 | นามสกุล |
| phone | 16s | 16 | เบอร์โทรศัพท์ |
| email | 32s | 32 | อีเมล |
| address | 64s | 64 | ที่อยู่ |
| created\_at | i | 4 | timestamp สร้าง record |
| updated\_at | i | 4 | timestamp แก้ไข record |

**ตารางที่ 2‑2** แฟ้มข้อมูลลูกค้า

2.2.1 cust\_id รหัสลูกค้า

cust\_id เป็นตัวเลขจำนวนเต็ม (integer) ใช้ในการระบุเอกลักษณ์ของลูกค้าแต่ละคนไม่ให้ซ้ำกัน เช่น 2001, 2002 เป็นต้น รหัสลูกค้าช่วยให้ค้นหาและเชื่อมโยงข้อมูลการเช่ากับลูกค้าได้อย่างรวดเร็ว โดยไม่สับสนแม้ว่าจะมีชื่อลูกค้าที่เหมือนกัน

2.2.2 status สถานะของลูกค้า

status เป็นค่าจำนวนเต็ม (integer) ใช้กำหนดสถานะของระเบียนลูกค้า ค่า 1 หมายถึงข้อมูลยัง Active และ 0 หมายถึงข้อมูลถูกลบ (Deleted) การใช้ Logical Delete ทำให้ยังคงบันทึกประวัติเดิมได้แม้ข้อมูลไม่ถูกใช้งานแล้ว

2.2.3 citizen\_id เลขบัตรประชาชน

citizen\_id ใช้เก็บหมายเลขบัตรประชาชน ความยาวคงที่ 16 ไบต์ (string) เช่น  
 "1234567890123" ข้อมูลนี้ช่วยยืนยันตัวตนลูกค้า และป้องกันการสวมรอยหรือการใช้ข้อมูลซ้ำซ้อน

2.2.4 name ชื่อลูกค้า

name เป็นข้อความ (string) ความยาวคงที่ 32 ไบต์ ใช้เก็บชื่อลูกค้า เช่น "สมชาย" การเก็บชื่อจริงช่วยให้ติดต่อและบริการลูกค้าได้ตรงตัวตน และยังเป็นข้อมูลสำคัญในสัญญาเช่ารถ

2.2.5 surname นามสกุล

surname เป็นข้อความ (string) ความยาวคงที่ 32 ไบต์ ใช้เก็บนามสกุลลูกค้า เช่น "ใจดี" เมื่อนำมาใช้คู่กับชื่อจริงจะทำให้ระบุบุคคลได้อย่างถูกต้องและชัดเจน

2.2.6 phone เบอร์โทรศัพท์

phone ใช้เก็บข้อมูลหมายเลขโทรศัพท์ ความยาวคงที่ 16 ไบต์ (string) เช่น "0812345678" ฟิลด์นี้มีความสำคัญสำหรับการติดต่อกับลูกค้า ทั้งในกรณีการจองรถ การยืนยันการเช่า หรือการติดต่อกรณีฉุกเฉิน

2.2.7 email อีเมล

email เป็นข้อความ (string) ที่มีความยาวคงที่ 32 ไบต์ โดยจะใช้ในการเก็บอีเมลลูกค้า เช่น "customer@gmail.com" อีเมลช่วยให้สามารถส่งข้อมูลการจอง ใบเสร็จ หรือการแจ้งเตือนต่าง ๆ ไปยังลูกค้าได้สะดวก

2.2.8 address ที่อยู่

address เป็นข้อความ (string) ความยาวคงที่ 64 ไบต์ ใช้เก็บที่อยู่ของลูกค้า เช่น "123 หมู่ 4 แขวงลาดพร้าว เขตลาดพร้าว กรุงเทพฯ" การบันทึกที่อยู่มีความสำคัญต่อการทำสัญญา การตรวจสอบข้อมูล และการติดต่ออย่างเป็นทางการ

2.2.9 created\_at เวลาที่สร้างระเบียน

created\_at ใช้เก็บค่าเวลาในรูปแบบ timestamp (integer) เพื่อบันทึกเวลาที่สร้างข้อมูลลูกค้าในระบบ การมีฟิลด์นี้ช่วยให้ทราบว่าลูกค้าคนนี้เริ่มต้นใช้บริการตั้งแต่เมื่อใด และใช้ประกอบการทำรายงาน

2.2.10 updated\_at เวลาที่แก้ไขล่าสุด

updated\_at เป็น timestamp (integer) เช่นเดียวกับ created\_at แต่ใช้เก็บเวลาที่  
มีการแก้ไขหรืออัปเดตข้อมูลล่าสุดของลูกค้า เพื่อช่วยติดตามความเปลี่ยนแปลงของข้อมูล และทำให้

ระบบมีความถูกต้องทันสมัย

## แฟ้มเก็บข้อมูลการเช่า rentals.dat

แฟ้มข้อมูลการเช่าใช้สำหรับเก็บรายละเอียดของการเช่ารถแต่ละครั้ง โดยมีการเชื่อมโยงระหว่าง รถ (cars.dat) และ ลูกค้า (customers.dat) เพื่อให้ระบบสามารถติดตามการใช้งานรถและข้อมูลลูกค้าได้ครบถ้วน

|  |  |  |  |
| --- | --- | --- | --- |
| ฟิลด์ | ชนิด | ขนาด (bytes | ตัวอย่าง |
| rental\_id | i | 4 | รหัสการเช่า |
| status | i | 4 | สถานะ Active=1, Deleted=0 |
| car\_id | I | 4 | รหัสรถที่เช่า |
| cust\_id | I | 4 | รหัสลูกค้า |
| Rent\_date | I | 4 | วันที่เริ่มเช่า (timestamp) |
| return\_date | I | 4 | วันที่คืนจริง  (timestamp) |
| due\_date | I | 4 | กำหนดวันคืน  (timestamp) |
| total\_days | i | 4 | จำนวนวันที่เช่า |
| total\_amount | f | 4 | ค่าเช่ารวม (บาท) |
| created\_at | i | 4 | timestamp สร้าง record |
| updated\_at | i | 4 | timestamp แก้ไข record |

**ตารางที่ 2‑3** แฟ้มข้อมูลการเช่า

2.3.1 rental\_id รหัสการเช่า

rental\_id เป็นรหัสไม่ซ้ำกันที่ใช้ระบุธุรกรรมการเช่าแต่ละครั้ง มีชนิดเป็นตัวเลขจำนวนเต็ม (integer) เช่น 3001, 3002 เป็นต้น การใช้รหัสเช่าแยกเฉพาะทำให้สามารถอ้างอิงธุรกรรมได้อย่างรวดเร็ว และช่วยแยกความแตกต่างแม้ว่าจะเป็นลูกค้าคนเดียวกันที่เช่าหลายครั้ง

2.3.2 status สถานะของการเช่า

status เป็นจำนวนเต็ม (integer) ใช้กำหนดสถานะของระเบียนการเช่า ค่า 1 หมายถึงยังใช้งานอยู่ (Active) และ 0 หมายถึงถูกลบ (Deleted) เพื่อช่วยจัดการข้อมูลด้วยวิธี Logical Delete และรักษาประวัติการเช่าเดิม

2.3.3 car\_id รหัสรถที่เช่า

car\_id เป็นจำนวนเต็ม (integer) ที่อ้างอิงไปยังฟิลด์ car\_id ของแฟ้ม cars.dat เพื่อระบุว่าธุรกรรมการเช่านี้ใช้รถคันใด การเชื่อมโยงข้อมูลแบบนี้ช่วยให้ระบบสามารถติดตามประวัติการใช้งานของรถแต่ละคันได้อย่างถูกต้อง

2.3.4 cust\_id รหัสลูกค้า

cust\_id เป็นจำนวนเต็ม (integer) ที่อ้างอิงไปยังฟิลด์ cust\_id ของแฟ้ม customers.dat เพื่อบอกว่าการเช่านี้เป็นของลูกค้าคนใด ช่วยให้สามารถตรวจสอบประวัติการเช่าและข้อมูลลูกค้าได้สะดวก

2.3.5 rent\_date วันที่เริ่มเช่า

rent\_date ใช้เก็บค่า timestamp (integer) ที่ระบุวันและเวลาที่เริ่มเช่ารถจริง ๆ ฟิลด์นี้มีความสำคัญต่อการคำนวณจำนวนวันเช่า และเป็นหลักฐานยืนยันการเริ่มสัญญาเช่า

2.3.6 return\_date วันที่คืนจริง

return\_date ใช้เก็บค่า timestamp (integer) เพื่อระบุวันและเวลาที่ลูกค้าคืนรถจริง ฟิลด์นี้ช่วยตรวจสอบความตรงต่อเวลาของการคืนรถ และใช้เปรียบเทียบกับ due\_date เพื่อพิจารณาค่าปรับหรือค่าใช้จ่ายเพิ่มเติม

2.3.7 due\_date กำหนดวันคืน

due\_date เป็น timestamp (integer) ที่กำหนดวันครบกำหนดคืนรถตามสัญญา ช่วยให้สามารถเปรียบเทียบกับ return\_date ได้ว่าลูกค้าคืนรถตรงเวลาหรือไม่ และใช้เป็นเงื่อนไขใน  
การคิดค่าปรับกรณีคืนช้า

2.3.8 total\_days จำนวนวันที่เช่า

total\_days ใช้เก็บจำนวนเต็ม (integer) ของจำนวนวันทั้งหมดที่ลูกค้าเช่ารถ เช่น   
3, 7, 10 วัน การเก็บข้อมูลนี้ทำให้สามารถตรวจสอบและยืนยันการคิดค่าเช่ารวมได้อย่างถูกต้อง

2.3.9 total\_amount ค่าเช่ารวม

total\_amount เป็นตัวเลขทศนิยม (float) ใช้เก็บผลรวมค่าเช่าทั้งหมด (หน่วยบาท) ซึ่งคำนวณจากจำนวนวัน (total\_days) × ค่าเช่าต่อวัน (daily\_rate\_thb) ของรถ ฟิลด์นี้ทำให้สามารถแสดงค่าใช้จ่ายของลูกค้าได้ทันทีโดยไม่ต้องคำนวณซ้ำ

2.3.10 created\_at เวลาที่สร้างระเบียน

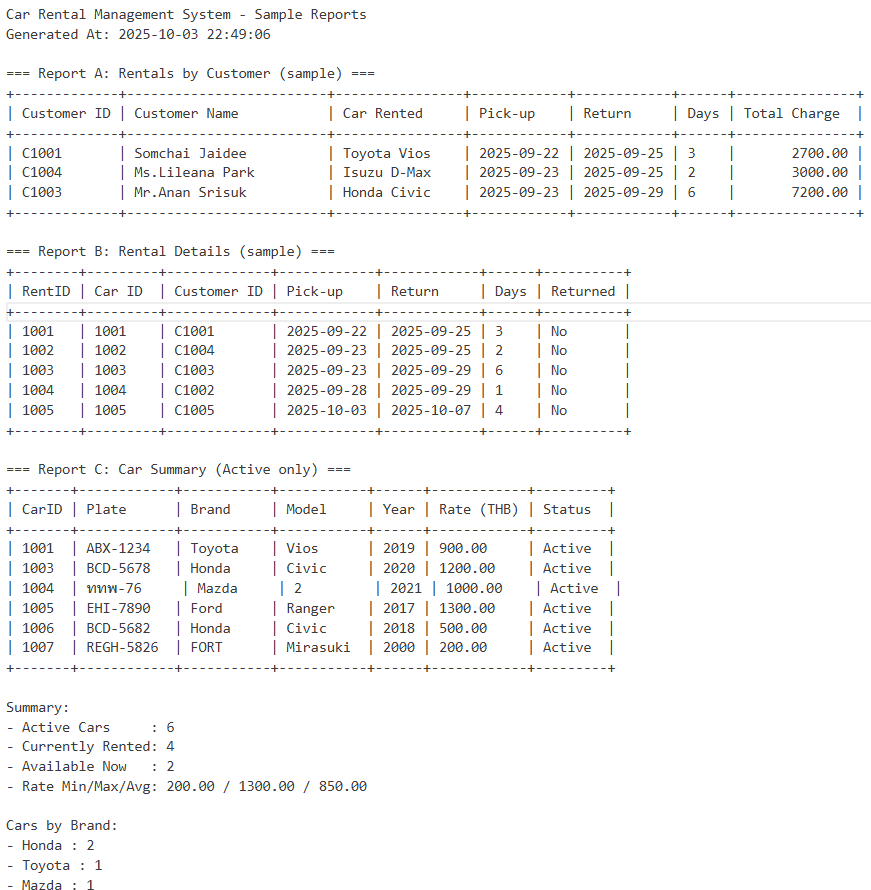
created\_at เป็น timestamp (integer) เก็บเวลาที่สร้างระเบียนการเช่า เพื่อใช้ตรวจสอบว่าสัญญาการเช่าเริ่มถูกบันทึกตั้งแต่เมื่อใด และเป็นประโยชน์ในการทำรายงานสถิติ

2.3.11 updated\_at เวลาที่แก้ไขล่าสุด

updated\_at เป็น timestamp (integer) ใช้เก็บเวลาที่มีการอัปเดตข้อมูลการเช่า  
  
ล่าสุด เช่น มีการเปลี่ยนแปลงจำนวนวันหรือการแก้ไขค่าเช่า ช่วยให้การติดตามความถูกต้องของข้อมูลเป็นไปอย่างโปร่งใสและน่าเชื่อถือ

## รายงานสรุป Report.txt

แฟ้ม report.txt เป็นแฟ้มข้อความที่ใช้สำหรับบันทึกข้อมูลการรายงานและสรุปผลการ  
ดำเนินงาน เช่น รายงานรถทั้งหมด รายงานลูกค้าทั้งหมด รายงานการเช่ารถ และรายงาน  
สถิติการใช้งาน โดยสามารถเปิดอ่านได้ด้วยโปรแกรม Text Editor ทั่วไป



**รูปภาพที่ 2‑1** ไฟล์ Report

2.4.1 header\_text ส่วนหัวรายงาน

เป็นฟิลด์ข้อความ (string 100 bytes) ใช้เก็บข้อความส่วนหัวรายงาน เช่น "Car Rental Management System – Sample Reports" เพื่อแสดงชื่อหรือวัตถุประสงค์ของรายงาน

2.4.2 generated\_at วันที่และเวลาที่สร้างรายงาน (YYYY-MM-DD HH:MM:SS)

เป็นฟิลด์ข้อความ (string 25 bytes) ใช้เก็บวันและเวลาที่รายงานถูกสร้างขึ้น เช่น "2025-10-03 22:49:06"

2.4.3 report\_sections ส่วนรายงานย่อย

ไฟล์ report.txt ถูกแบ่งเป็น 3 ส่วนหลัก ได้แก่ Report A: Rentals by Customer, Report B: Rental Details และ Report C: Car Summary

2.4.4 section\_table\_header หัวตาราง

เป็นฟิลด์ข้อความ (string 80 bytes) ใช้เก็บหัวตารางสำหรับแต่ละส่วน เช่น Customer ID, Customer Name, Car Rented, Pick-up, Return, Days, Total Charge

2.4.5 section\_records ข้อมูลตาราง

เป็นฟิลด์ข้อความ (string 120 \* N bytes, โดย N = จำนวนเรคอร์ด) ใช้เก็บข้อมูลจริงของแต่ละบรรทัด เช่น C1001, Somchai Jaidee, Toyota Vios, 2025-09-22, 2025-09-25, 3, 2700.00

2.4.6 summary\_section สรุปข้อมูล

เป็นฟิลด์ข้อความ (string 150 bytes) ใช้เก็บข้อมูลสรุประบบ เช่น Active Cars, Currently Rented, Available Now และ Rate Min/Max/Avg

2.4.7 statistics\_section สถิติการใช้งาน

เป็นฟิลด์ข้อความ (string 150 bytes) ใช้เก็บข้อมูลสถิติ เช่น Cars by Brand: Honda 2, Toyota 1, Mazda 1 เพื่อช่วยวิเคราะห์เชิงสถิติ

# ขั้นตอนการดำเนินงาน

โปรแกรมการใช้งานระบบจัดการการเช่ารถ เป็นระบบจัดการการเช่ารถ จะช่วยในการจัดเก็บและบริหารข้อมูลรถให้ง่ายขึ้น ผู้ใช้สามารถเพิ่ม แก้ไข ลบ (แบบ Logical Delete) และแสดงข้อมูลรถได้สะดวก โดยข้อมูลของรถที่จัดเก็บในระบบจะประกอบไปด้วย Car ID, License Plate, Brand, Model, Year, Daily Rate, Odometer, Status และข้อมูลอื่น ๆ ที่เกี่ยวข้อง เพื่อความสะดวกในการเรียกใช้งาน โปรแกรมยังรองรับการค้นหาหรือดูข้อมูลเฉพาะที่ต้องการ เช่น ดูเฉพาะรถที่ Active หรือรถที่ถูกลบแล้ว

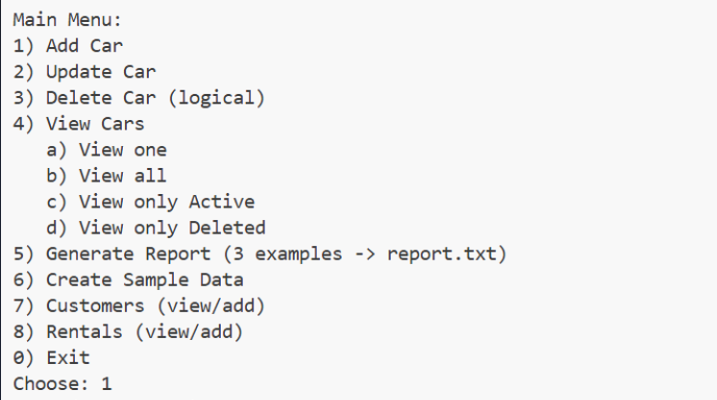
การจัดการข้อมูลรถทั้งหมดทำงานผ่านเมนูหลักของโปรแกรม ผู้ใช้สามารถเลือกเมนู Manage Cars เพื่อเข้าสู่การทำงานเกี่ยวกับข้อมูลรถ ซึ่งจะมีฟังก์ชันให้เลือกใช้งาน ได้แก่ เพิ่มรถใหม่ แก้ไขข้อมูลรถ ลบข้อมูลรถ และแสดงข้อมูลรถ

**สำหรับผู้ใช้งานโปรแกรม**

## การจัดการข้อมูลรถ (Cars Management)

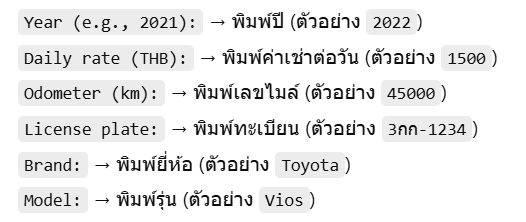
3.1.1 วิธีการเพิ่มข้อมูลรถ Add Car

3.1.1.1 กรอกหมายเลข 1 หลังคำว่า choose: ตามด้วยการคลิกที่ Enter เพื่อเรียกฟังก์ชัน Manage Cars เมื่อเลือก 1 เมนูย่อยของ Manage Cars จะขึ้นมา ประกอบด้วยตัวเลือกหลักดังนี้



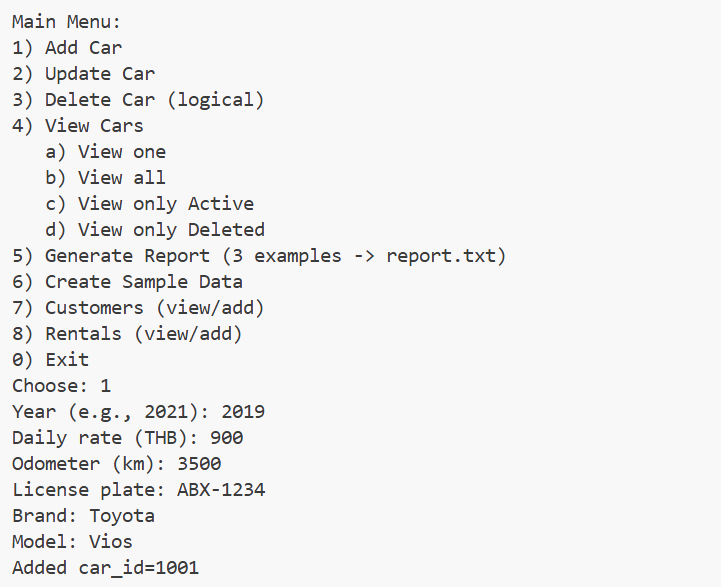
**รูปภาพที่ 3‑1** ตัวอย่างหน้าโปรแกรมระบบจัดการการเช่ารถ

3.1.1.2 โปรแกรมจะถามข้อมูลทีละฟิลด์ โดยแต่ละฟิลด์จะมีรูปแบบการเก็บข้อมูลดังนี้



**รูปภาพที่ 3‑2** ตัวอย่างรูปแบบการรับข้อมูลฟิลด์ Add car

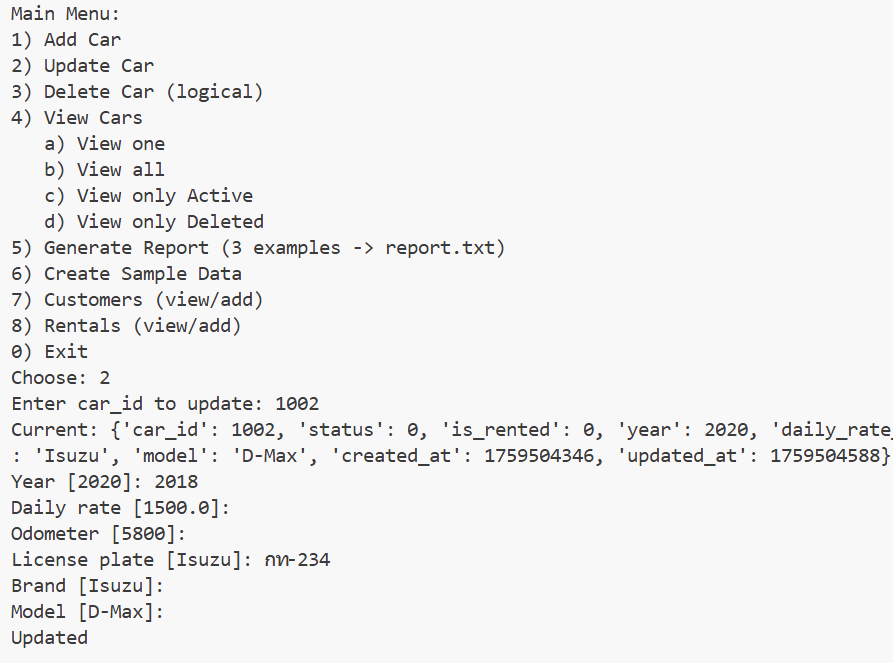
3.1.1.3 เมื่อกรอกข้อมูลเสร็จสิ้น ระบบจะเพิ่มเรคอร์ดใหม่ และแสดงข้อความยืนยัน เช่น Added car\_id=1001



**รูปภาพที่ 3‑3** หน้าแสดงข้อความยืนยันการเพิ่มข้อมูลรถ

3.1.2 วิธีการแก้ไขข้อมูลรถ Add Car

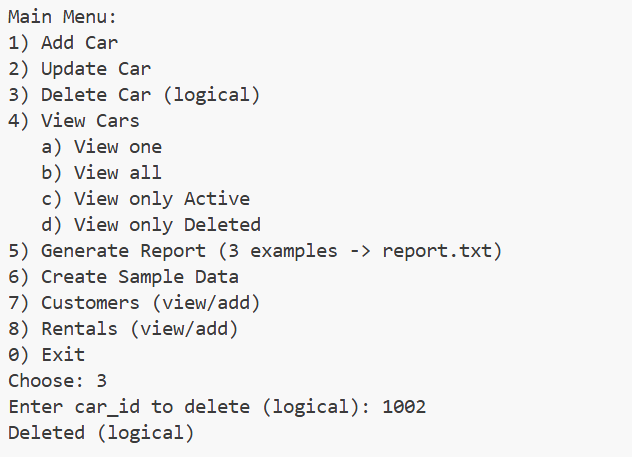
กรอกหมายเลข 2 หลังคำว่า choose: ตามด้วย Enter เพื่อไปที่ Update car เพื่อแก้ไขข้อมูลรถ โปรแกรมจะถามทีละฟิลด์อีกครั้ง หากข้อมูลที่ไม่ต้องการแก้ไขให้กด Enter



**รูปภาพที่ 3‑4** หน้าแสดงการแก้ไขข้อมูลรถ

3.1.3 การลบรายการรถ Delete car

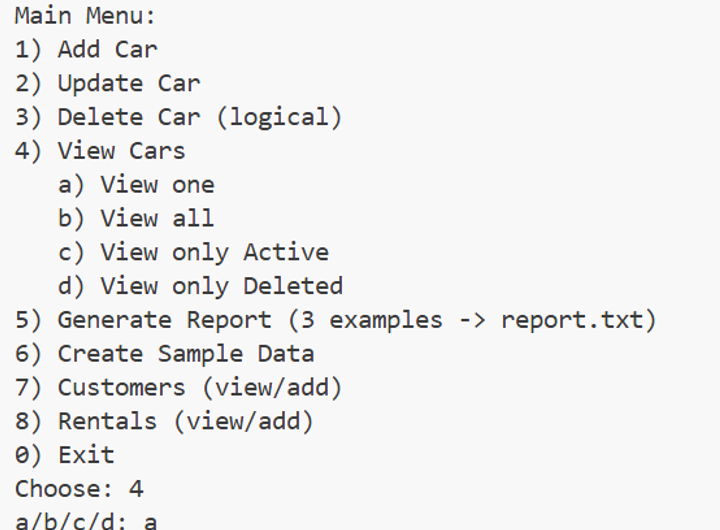
กรอกหมายเลข 3 หลังคำว่า choose: ตามด้วย Enter เพื่อไปที่ Delete car เพื่อแก้ไขข้อมูลรถ โปรแกรมจะถามว่า Enter car\_id to delete (logical): ให้ผู้ใช้กรอกหมายเลข Car ID ของรถที่ต้องการลบ



**รูปภาพที่ 3‑5** หน้าแสดงการลบข้อมูลรถ

3.1.4 วิธีการดูหน้ารายการข้อมูลรถ View All Cars

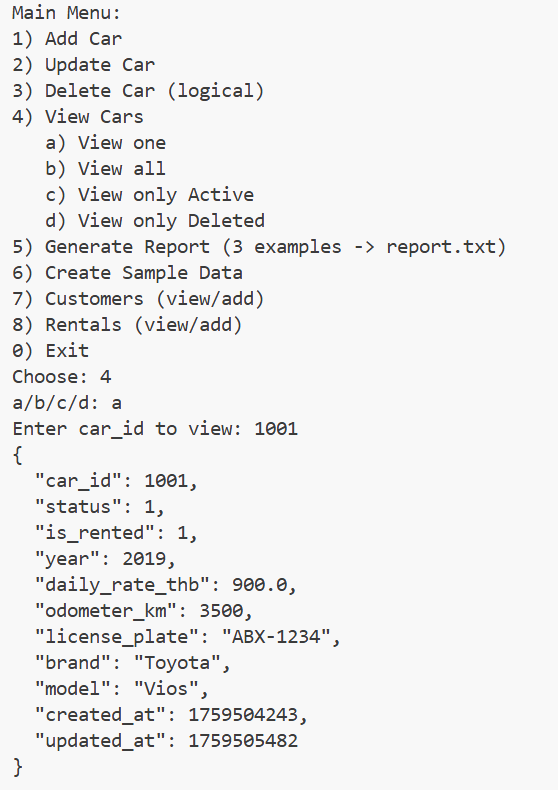
กรอกหมายเลข 4 หลังคำว่า choose: ตามด้วย Enter เพื่อไปที่ View cars เพื่อดูข้อมูลรถ โปรแกรมจะถาม sub-menu ย่อยให้เลือก a/b/c/d ดังนี้



**รูปภาพที่ 3‑6** หน้าแสดงการดูข้อมูลรถ

3.1.4.1 View one (a)

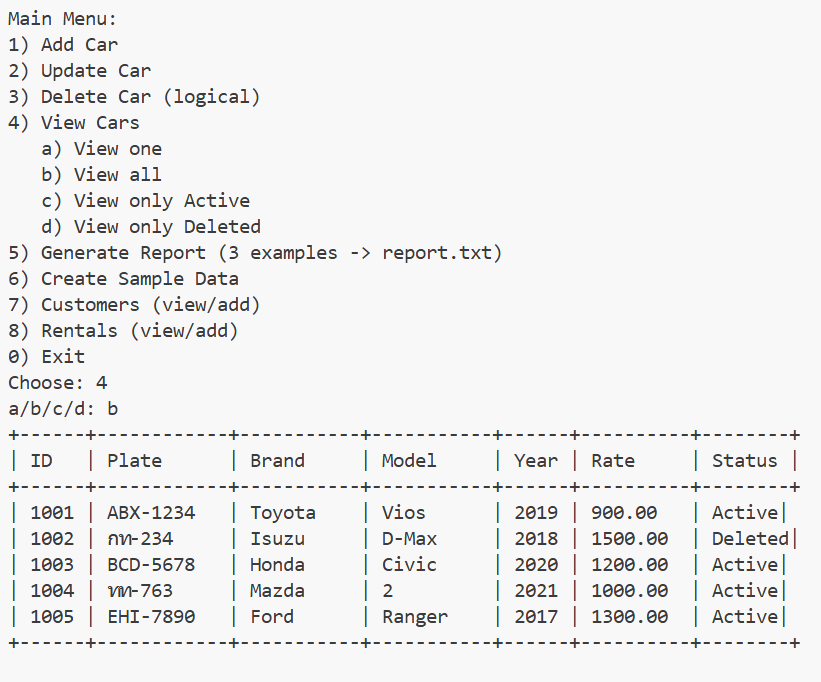
ใช้ดูรายละเอียดรถทีละคัน โดยวิธีการใช้งานคือให้พิมพ์ 4 ตามด้วย Enter แล้วพิมพ์ a ลงใน a/b/c/d: หลังจากนั้นโปรแกรมจะถามผู้ใช้ต้องการดูข้อมูลรถเฉพาะรหัสรถอะไร ให้กรอกหมายเลข Car ID ในช่อง Enter car\_id to view: ตามตัวอย่างรูป ดังนี้



**รูปภาพที่ 3‑7** หน้าแสดงการดูข้อมูลรถ View one (a)

3.1.4.2 View one (b)

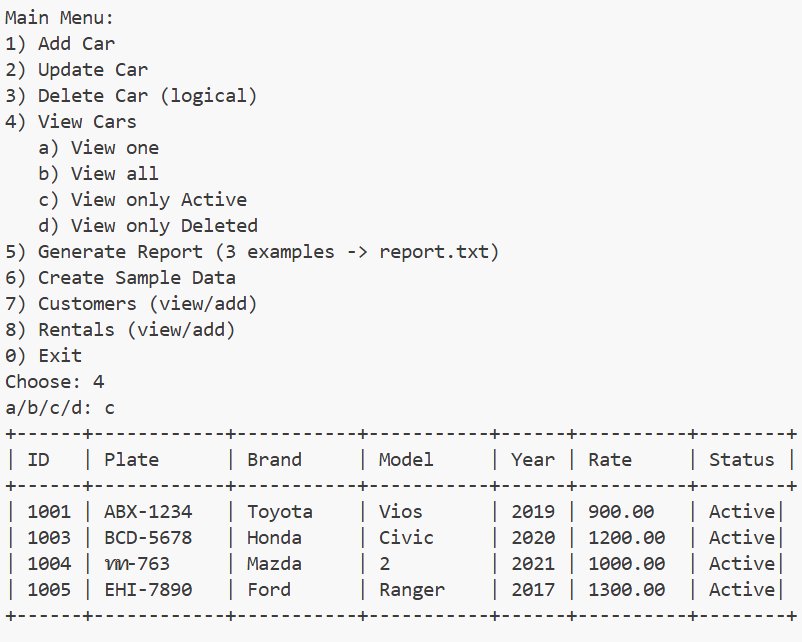
ใช้ดูรายละเอียดรถทีละคัน โดยวิธีการใช้งานคือให้พิมพ์ 4 ตามด้วย Enter แล้วพิมพ์ b ลงใน a/b/c/d: โปรแกรมจะแสดงข้อมูลรถทุกคันตามตัวอย่างรูป ดังนี้



**รูปภาพที่ 3‑8** หน้าแสดงการดูข้อมูลรถ View one (b)

3.1.4.3 View one (c)

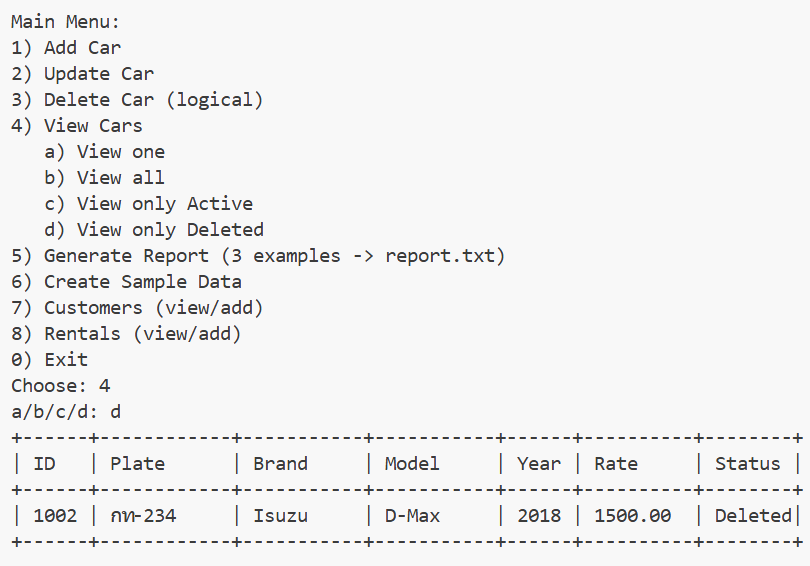
ใช้ดูรายละเอียดรถทีละคัน โดยวิธีการใช้งานคือให้พิมพ์ 4 ตามด้วย Enter แล้วพิมพ์ c ลงใน a/b/c/d: โปรแกรมจะแสดงข้อมูลรถทุกคันแต่ยกเว้นรถที่ถูกลบออกไปแล้วตามตัวอย่างรูป ดังนี้



**รูปภาพที่ 3‑9** หน้าแสดงการดูข้อมูลรถ View one (c)

3.1.4.4 View one (d)

ใช้ดูรายละเอียดรถทีละคัน โดยวิธีการใช้งานคือให้พิมพ์ 4 ตามด้วย Enter แล้วพิมพ์ c ลงใน a/b/c/d: โปรแกรมจะแสดงข้อมูลเฉพาะรถที่ถูกลบตามตัวอย่างรูป ดังนี้

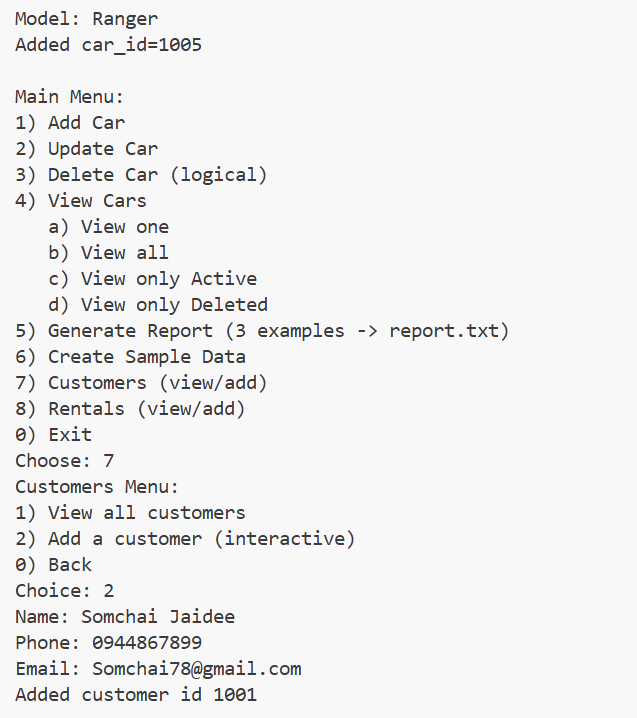


**รูปภาพที่ 3‑10** หน้าแสดงการดูข้อมูลรถ View one (d)

## การจัดการข้อมูลลูกค้า (Customer Management)

3.2.1 วิธีการเพิ่มข้อมูลลูกค้า Add Customer

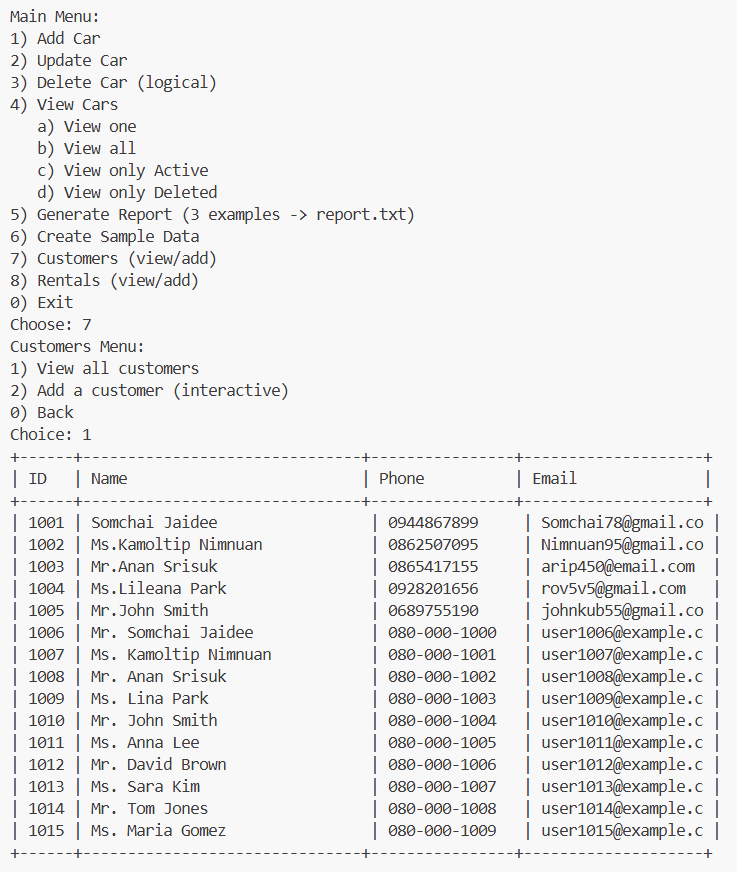
กรอกหมายเลข 7 หลังคำว่า choose: ตามด้วย Enter หลังจากนั้นโปรแกรมจะให้เลือกระหว่าง 1 การดูข้อมูลลูกค้า, 2 การเพิ่มข้อมูลลูกค้า, 3 การย้อนกลับไปที่เมนูหลัก ให้เลือก 2 เพื่อเพิ่มข้อมูลลูกค้า แล้วกรอกข้อมูลลูกค้าทีละบรรทัดตามตัวอย่าง ดังนี้



**รูปภาพที่ 3‑11** หน้าการเพิ่มข้อมูลลูกค้า

3.2.2 วิธีการดูหน้ารายการข้อมูลของลูกค้า

กรอกหมายเลข 7 หลังคำว่า choose: ตามด้วย Enter หลังจากนั้นโปรแกรมจะให้เลือกระหว่าง 1 การดูข้อมูลลูกค้า, 2 การเพิ่มข้อมูลลูกค้า, 3 การย้อนกลับไปที่เมนูหลัก ให้เลือก 1 เพื่อดูข้อมูลลูกค้าทั้งหมดตามตัวอย่าง ดังนี้

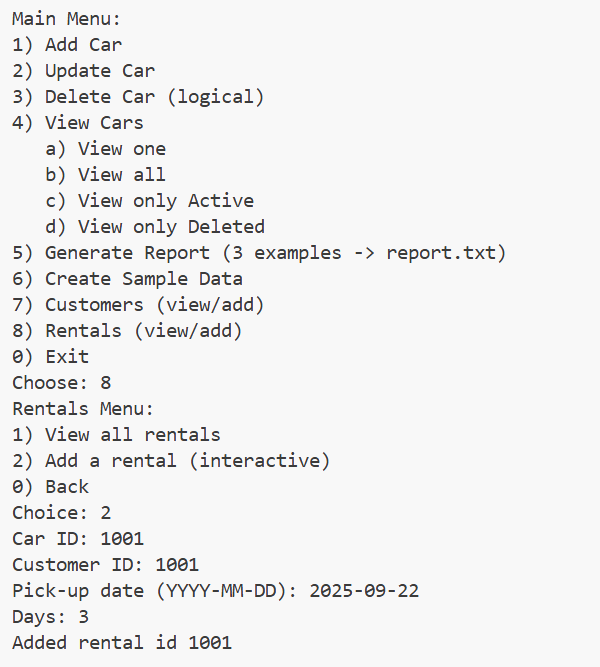


**รูปภาพที่ 3‑12** หน้าการดูข้อมูลลูกค้า

## จัดการข้อมูลการเช่า (Rentals Management)

3.3.1 เพิ่มรายการเช่า (Add Rental)

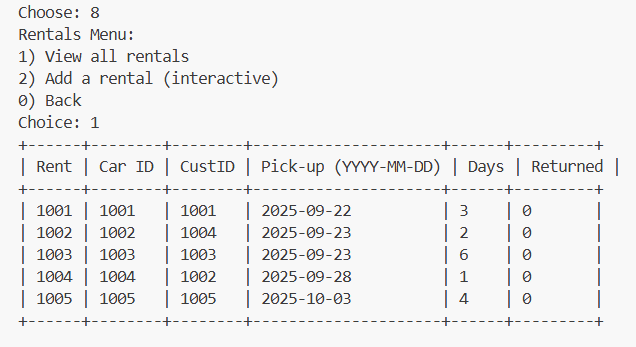
กรอกหมายเลข 8 หลังคำว่า choose: ตามด้วย Enter หลังจากนั้นโปรแกรมจะให้เลือกระหว่าง 1 การดูข้อมูลการเช่ารถ, 2 การเพิ่มข้อมูลการเช่ารถ, 3 การย้อนกลับไปที่เมนูหลัก ให้เลือก 2 เพื่อเพิ่มข้อมูลการเช่ารถ แล้วกรอกข้อมูลลูกค้าทีละบรรทัดตามตัวอย่าง ดังนี้



**รูปภาพที่ 3‑13** หน้าการเพิ่มข้อมูลการเช่ารถ

3.3.2 แสดงรายการเช่าทั้งหมด (View All Rentals)

กรอกหมายเลข 8 หลังคำว่า choose: ตามด้วย Enter หลังจากนั้นโปรแกรมจะให้เลือกระหว่าง 1 การดูข้อมูลการเช่ารถ, 2 การเพิ่มข้อมูลการเช่ารถ, 3 การย้อนกลับไปที่เมนูหลัก ให้เลือก 1 เพื่อดูข้อมูลการเช่ารถตามตัวอย่าง ดังนี้



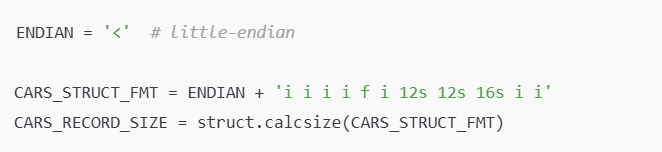
**รูปภาพที่ 3‑14** หน้าการดูข้อมูลการเช่ารถ

# ผลการดำเนินงาน

## ฟังก์ชันในงานพื้นฐานในระบบเช่ารถ

4.1.1 struct กำหนดรูปแบบ record และ pack/unpack

ไฟล์ cars.dat, customers.dat, rentals.dat เก็บข้อมูลแบบ fixed-length records โดยใช้โมดูล struct เพื่อแปลงข้อมูล Python → bytes (pack) และ bytes → Python (unpack)



**รูปภาพที่ 4‑1** struct

4.1.1.1 อธิบาย format 1. < : little-endian (กำหนด byte order) 2. i : 4-byte signed integer 3. f : 4-byte float 4. 12s, 16s : fixed-length byte string (12 bytes / 16 bytes)

4.1.1.2 การแมปฟิลด์ (cars)

1. car\_id (i)

2. status (i) — 1=active, 0=deleted

3.is\_rented (i) — 1=rented, 0=free

4. year (i)

5. daily\_rate\_thb (f)

6. odometer\_km (i)

7. license\_plate (12s)

8. brand (12s)

9. model (16s)

10. created\_at (i) — unix timestamp (int)

11. updated\_at (i)

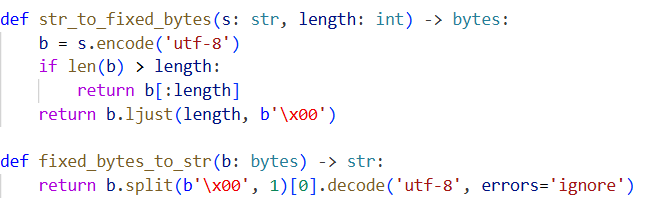
4.1.1.3 ตัวอย่างฟังก์ชัน pack / unpack



**รูปภาพที่ 4‑2** pack และ unpack car

4.1.2 Helper: fixed-length string (padding / truncate)

ในโค้ดระบบจัดการการเช่ารถนี้มี 2 ฟังก์ชันสำคัญ ดังนี้



**รูปภาพที่ 4‑3** pack และ unpack car

4.1.2.1 อธิบายฟังก์ชัน

4.1.2.1.1 str\_to\_fixed\_bytes 1. แปลง s → UTF-8 bytes 2. ถ้ายาวกว่า length ให้ตัดทิ้ง (truncate) ระวัง ตัดใน byte-level อาจตัดเพียงส่วนของ multi-byte char 3. ถ้าสั้นกว่าก็ ljust เติม \x00 (null bytes) จนได้ความยาวที่กำหนด

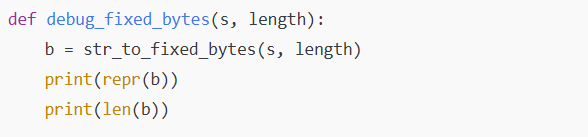
4.1.2.2 fixed\_bytes\_to\_str 1. ตัดที่ null byte ตัวแรก (padding) แล้ว decode เป็น string 2. ใช้ errors='ignore' เพื่อไม่ให้เกิด exception หาก byte sequence ไม่สมบูรณ์

4.1.2.2 ตัวอย่างผลลัพธ์

"TOYOTA" → b'TOYOTA\x00\x00\x00...' (เต็มความยาว 12/16)

b'TOYOTA\x00...' → "TOYOTA"

4.1.2.3 ฟังก์ชันเสริมแนะนำ (เพื่อดูตัวอย่าง byte/padding เมื่อ debug):



**รูปภาพที่ 4‑4** debug

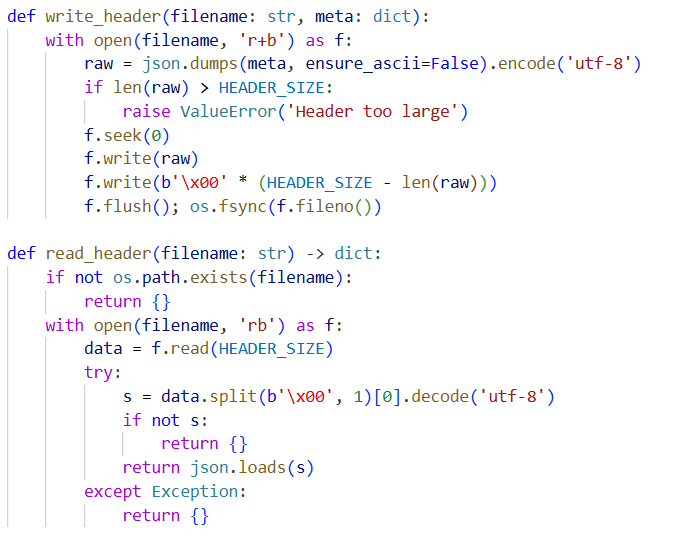
4.1.3 Header management — เก็บ metadata ในส่วนหัวของไฟล์

4.1.3.1 ระบบสำรองที่ HEADER\_SIZE = 256 ไบต์สำหรับเก็บ JSON header (เช่น version, count, next\_id)



**รูปภาพที่ 4‑5** ensure

4.1.3.2 และการเขียน/อ่าน header ปัจจุบัน



**รูปภาพที่ 4‑6** write และ read

4.1.3.1 อธิบายฟังก์ชัน

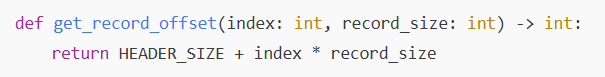
Header เป็น JSON แบบง่ายๆ เก็บ count (จำนวน record ที่มี), next\_id (id ถัดไป), last\_updated (อาจถูกเพิ่มโดย write\_header) 2. เมื่อสร้างไฟล์ใหม่ ensure\_file จะเขียน header เริ่มต้น + padding ให้พอดี 256 ไบต์ 3. write\_header จะอัปเดตข้อมูล meta ในตำแหน่งเริ่มต้นของไฟล์เสมอ (r+b) 4. การจำกัด HEADER\_SIZE ป้องกัน header ใหญ่เกิน จึงเป็น good practice

4.1.3.2 ข้อเสนอแนะเพิ่มเติม

ถ้าต้องการ header ขนาดใหญ่ขึ้น ให้เพิ่ม HEADER\_SIZE และระวังให้ read\_header / write\_header สอดคล้องกัน 2. เก็บ free\_list เพื่อ reuse บันทึกที่ลบแบบ logical

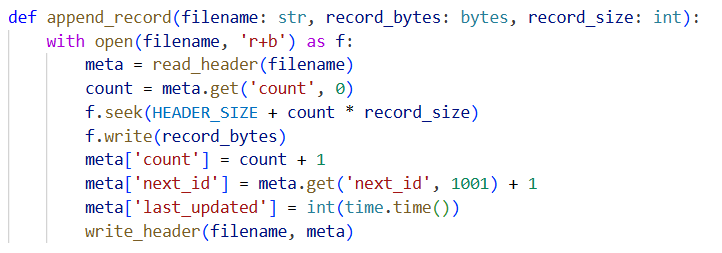
4.1.4 Low-level record access (offsets, append, write-at, read-at)

1. การคำนวณตำแหน่ง record ในไฟล์:



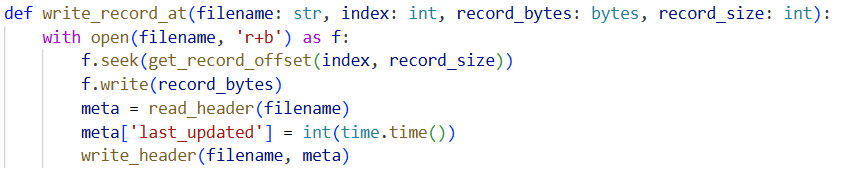
**รูปภาพที่ 4‑7** get\_record

2. append\_record (เพิ่ม record ท้ายไฟล์ + อัปเดต header)



**รูปภาพที่ 4‑8** append

3. write\_record\_at (เขียนซ้ำที่ index ที่กำหนด)

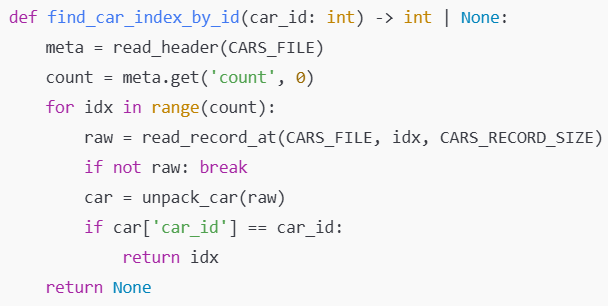


**รูปภาพที่ 4‑9** write\_record

4.1.4.1 อธิบายฟังก์ชัน

append\_record อ่าน header เพื่อดู count แล้วเขียนบันทึกที่ offset = HEADER + count\*record\_size 2. อัปเดต meta['count'], meta['next\_id'] และ last\_updated → แล้วเรียก write\_header 3. write\_record\_at ใช้เขียนซ้ำ record เดิม (ใช้เมื่อ update หรือลบแบบ logical) 4. read\_record\_at เช็กขนาดไฟล์ก่อนอ่าน เพื่อไม่ให้อ่านนอกขอบเขต

4.1.5 Find helpers — ค้นหา index ของ record ตาม id



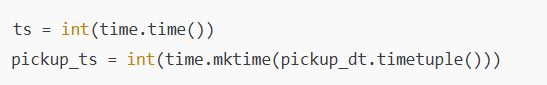
รูปภาพที่ ‑ find

4.1.5.1 อธิบายฟังก์ชัน

1. อ่าน header เพื่อทราบจำนวน count 2. วน loop ตั้งแต่ 0..count-1: อ่าน raw record แต่ละตำแหน่ง → unpack → เช็ก id 3. คืนค่า index เมื่อพบ (ใช้ต่อกับ read\_record\_at / write\_record\_at) 4. ถ้าไม่พบคืน None

4.1.6 เวลา (timestamp) — เก็บ, แปลง และใช้งาน

ระบบใช้ int unix timestamp (วินาที) เก็บเวลา created\_at/updated\_at/pickup\_ts



**รูปภาพที่ 4‑11** Time

การแปลงกลับเป็นวันที่สำหรับแสดง



**รูปภาพที่ 4‑12** datetime

4.1.6.1 อธิบายฟังก์ชัน

time.time() → timestamp ปัจจุบัน (float) → แปลงเป็น int เพื่อบันทึก 2. time.mktime(date.timetuple()) → แปลง datetime.date เป็น timestamp (local time) 3. fromtimestamp → แปลง timestamp → datetime เพื่อแสดง human-readable 4. ข้อควรระวังเรื่อง timezone: โค้ดนี้ทำงานบน timezone ของเครื่อง (local); ถ้าต้องการ cross 5. timezone ควรเก็บ UTC หรือใช้ datetime ที่มี timezone-aware

## ฟังก์ชัน Pack และ Unpack

ระบบจัดการเช่ารถนี้ ใช้การจัดเก็บข้อมูลลงไฟล์ .dat ในรูปแบบ fixed-length binary record

4.2.1 Pack: เอา dict → struct.pack → bytes

4.2.2 Unpack: เอา raw bytes → struct.unpack → dict

การทำแบบนี้ทำให้ระบบประหยัดพื้นที่และเข้าถึง record ได้รวดเร็ว (random access) เพราะทุก record มีขนาดเท่ากัน

4.2.1 Cars (รถ)

โครงสร้าง record (จาก CARS\_STRUCT\_FMT) มี car\_id | status | is\_rented | year | daily\_rate | odometer | license\_plate | brand | model | created\_at | updated\_at

โค้ด pack/unpack:



**รูปภาพที่ 4‑13** Pack and unpack (car)

4.2.1.1 การทำงานทีละขั้นตอน (pack\_car):

1. รับ dictionary car เช่น {'car\_id':1001, 'brand':'Toyota', ...} 2. แปลงค่า string เป็น fixed-length bytes เช่น brand → b'Toyota\x00\x00... (12 bytes) 3. ใช้ struct  
.pack รวมทั้งหมดเป็น binary record 4. คืนค่าเป็น bytes → พร้อมเขียนลงไฟล์

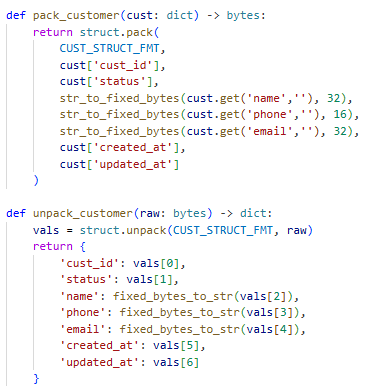
4.2.1.2 การทำงานทีละขั้นตอน (unpack\_car):

1. อ่าน raw bytes จากไฟล์ (ยาว 72 bytes) 2. ใช้ struct.unpack แตกเป็น tuple (ค่าตามลำดับ) 3. นำมา map กลับเป็น dict พร้อมแปลง string (fixed\_bytes → str)

4.2.2 Customers (ลูกค้า)

โครงสร้าง record (CUST\_STRUCT\_FMT) มี cust\_id | status | name(32) | phone  
(16) | email(32) | created\_at | updated\_at

โค้ด pack/unpack



**รูปภาพที่ 4‑14** Pack and unpack (customer)

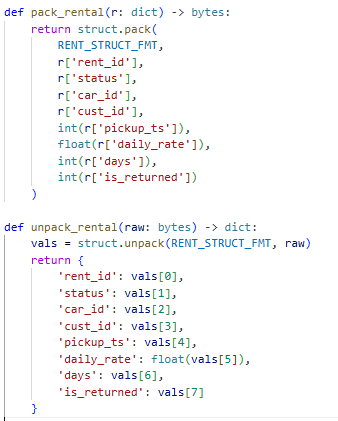
4.2.2.1 อธิบายฟังก์ชัน

1. name ความยาวสูงสุด 32 ตัวอักษร, phone 16, email 32 2. pack: แปลงเป็น bytes + padding 3. unpack: ตัด null byte ออกแล้ว decode

4.2.3 Rentals (การเช่า)

โครงสร้าง record (RENT\_STRUCT\_FMT) มี rent\_id | status | car\_id | cust\_id | pickup\_ts | daily\_rate | days | is\_returned

โค้ด pack/unpack:



**รูปภาพที่ 4‑15** Pack and unpack (rentel)

4.2.3.1 อธิบายฟังก์ชัน

1. pickup\_ts เก็บเป็น timestamp (int) เช่น 1696005000 2. is\_returned เป็น flag 1=คืนแล้ว, 0=ยังไม่คืน 3. ใช้ daily\_rate กับ days เพื่อคำนวณค่าเช่ารวม

4.2.4 ภาพรวมการใช้งาน Pack/Unpack

4.2.4.1 ตอน เพิ่มข้อมูลใหม่ → สร้าง dict → pack → append\_record() → เขียนลงไฟล์

4.2.4.2 ตอน อ่านข้อมูล → read\_record\_at() → unpack → dict → ใช้งานในโปรแกรม

4.2.4.3 ตอน แก้ไข/ลบ → read\_record\_at → unpack → แก้ค่า → pack → write\_record\_at

## ฟังก์ชันจัดการข้อมูล (CRUD)

ในระบบเช่ารถนี้ มีข้อมูลหลัก 3 ประเภท:

1. Cars (ข้อมูลรถ)

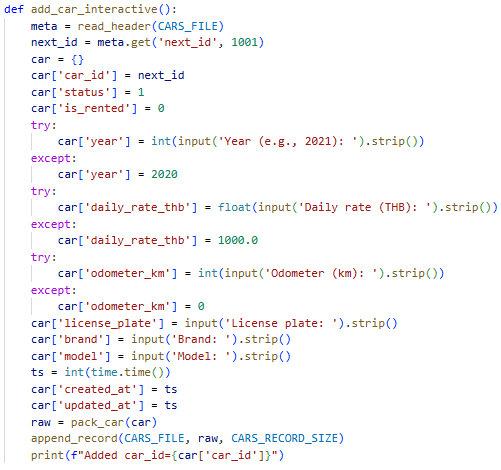
2. Customers (ข้อมูลลูกค้า)

3. Rentals (ข้อมูลการเช่า)

แต่ละกลุ่มจะมี CRUD (Create, Read, Update, Delete) หรือบางกรณีเป็น CRU (เพราะลบแบบ logical เท่านั้น)

4.3.1 การจัดการ Cars (รถ)

4.3.1.1 เพิ่มข้อมูลรถ add\_car\_interactive()



**รูปภาพที่ 4‑16** add\_car

4.3.1.2 ขั้นตอนการทำงาน

4.3.1.2.1 อ่าน next\_id จาก header → ใช้เป็น car\_id

4.3.1.2.2 รับค่าจากผู้ใช้ (ปี, ราคาเช่าต่อวัน, เลขไมล์, ป้ายทะเบียน, ยี่ห้อ, รุ่น)

4.3.1.2.3 ถ้าผู้ใช้กรอกผิด → ใส่ค่า default (year=2020, daily\_rate  
=1000, odometer=0)

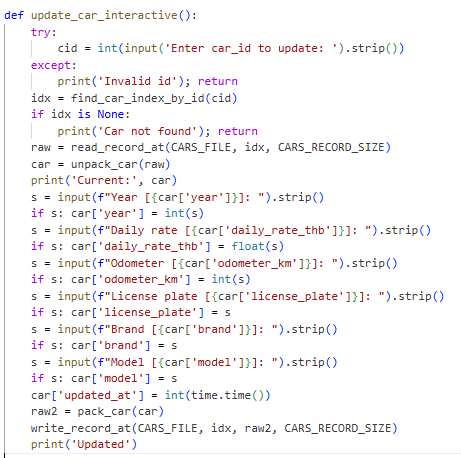
4.3.1.2.1 เพิ่ม timestamp created\_at, updated\_at

4.3.1.2.2 pack\_car → แปลงเป็น binary record

4.3.1.2.3 เขียนไฟล์ด้วย append\_record

4.3.1.2.4 แสดงผล “Added car\_id=…”

4.3.1.3 แก้ไขข้อมูลรถ — update\_car\_interactive()



**รูปภาพที่ 4‑17** update\_car\_interactive

4.3.1.3.1 อธิบายฟังก์ชัน

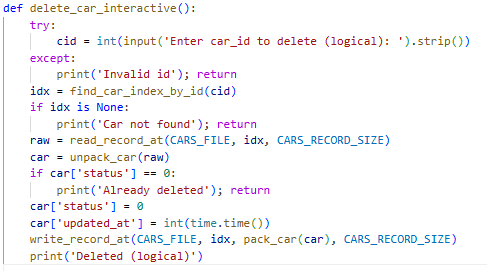
4.3.1.3.1 ค้นหา car\_id → ถ้าไม่เจอ return

4.3.1.3.1 อ่าน record เดิม → แสดงค่า

4.3.1.3.1 รับค่าจากผู้ใช้ → ถ้าไม่ใส่จะคงค่าเดิม

4.3.1.3.1 update timestamp + pack\_car → write\_record\_at

4.3.1.4 ลบข้อมูลรถ (logical delete) — delete\_car\_interactive()

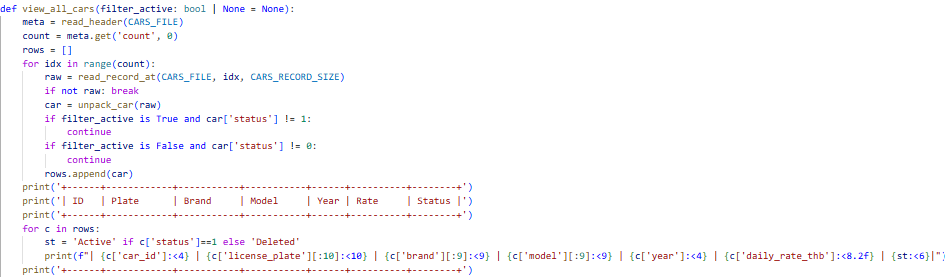


**รูปภาพที่ 4‑18** delete\_car

4.3.1.5 ดูข้อมูลรถ

4.3.1.5.1 view\_all\_cars(filter\_active=None) ดูทั้งหมด/เฉพาะ Active/เฉพาะ Deleted

4.3.1.5.2 view\_one\_car() — ดูรถคันเดียว (json format)



**รูปภาพที่ 4‑19** view\_all\_cars

4.3.2 การจัดการ Customers (ลูกค้า)

4.3.2.1 เพิ่มลูกค้า add\_customer()



**รูปภาพที่ 4‑20** add\_customer

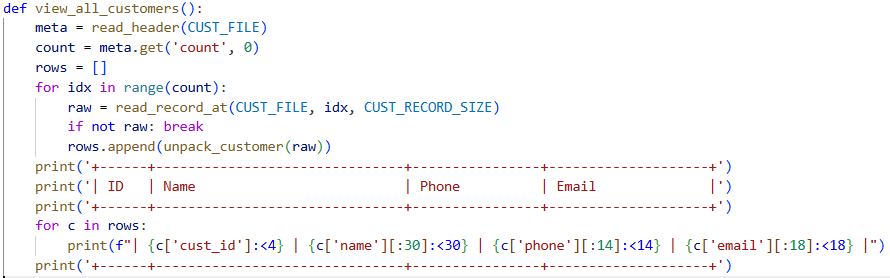
4.3.2.1.1 อธิบายฟอร์ม

4.3.2.1.1 ใช้ header next\_id → กำหนด customer id

4.3.2.1.2 บันทึกข้อมูลลูกค้า (status=1)

4.3.2.1.3 ใช้ pack\_customer → append ลงไฟล์

4.3.2.2 ดูลูกค้าทั้งหมด view\_all\_customers()



**รูปภาพที่ 4‑21** view\_all\_customers

4.3.2.2.1 อธิบาย

อ่านทุก record → แปลงเป็น dict → แสดงเป็นตาราง

4.3.3 การจัดการ Rentals (การเช่า)

4.3.3.1 เพิ่มการเช่า add\_rental()



**รูปภาพที่ 4‑22** add\_rental

4.3.3.1.1 ขั้นตอน

4.3.3.1.1.1 กำหนด rent\_id ใหม่จาก header

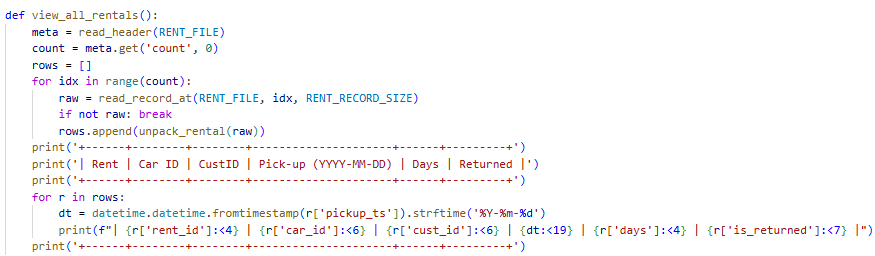
4.3.3.1.1.2 เก็บข้อมูลการเช่า: car\_id, cust\_id, pickup\_ts, daily  
\_rate, days

4.3.3.1.1.3 เขียนลง rentals.dat

4.3.3.1.1.4 อัปเดตสถานะรถ → is\_rented=1

4.3.3.1.1.5 คืนค่า rent\_id

4.3.3.2 ดูการเช่าทั้งหมด — view\_all\_rentals()



**รูปภาพที่ 4‑23** view\_all\_rentals

## ฟังก์ชันการสร้างรายงาน

ฟังก์ชันหลักคือ



**รูปภาพที่ 4‑24** generate\_report

ใช้สร้างไฟล์ report.txt โดยมี 3 ส่วนสำคัญ:

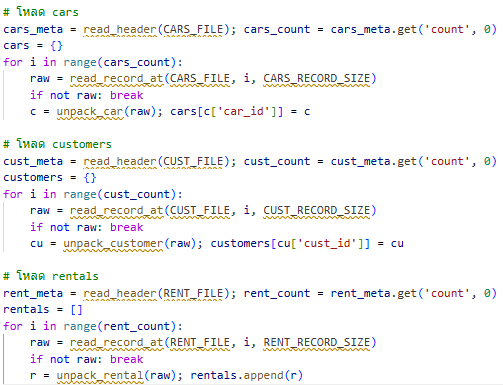
1. Report A: Rentals by Customer → สรุปการเช่ารถของลูกค้า

2. Report B: Rental Details → รายการเช่ารถทั้งหมด

3. Report C: Car Summary → รายงานรถที่มีอยู่ + สรุปสถิติ

4.4.1 การโหลดข้อมูลก่อนทำรายงาน

ก่อนทำรายงาน ระบบต้องโหลดข้อมูลจากทั้ง 3 ไฟล์



**รูปภาพที่ 4‑25** โหลดข้อมูล

4.4.1.1 อธิบาย

4.4.1.1.1 ใช้ read\_header → ดูจำนวน record

4.4.1.1.2 วน loop อ่าน record ทั้งหมด

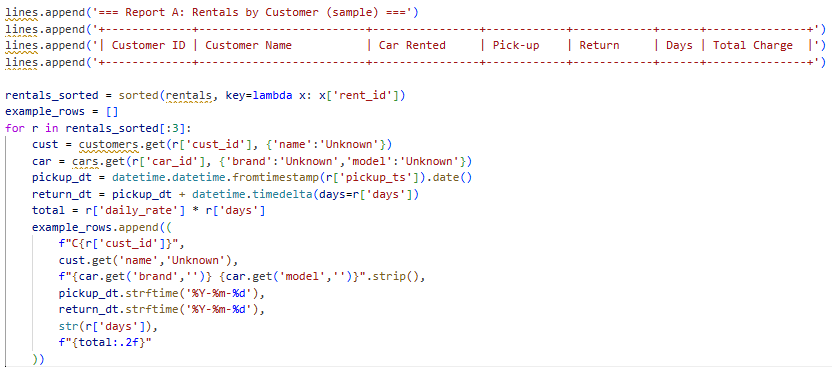
4.4.1.1.3 ใช้ unpack\_\* แปลงเป็น dict

4.4.1.1.4 cars, customers เก็บใน dict (key=id) เพื่อค้นหาง่าย

4.4.1.1.5 rentals เก็บเป็น list

4.4.2 Report A: Rentals by Customer

เป็นรายงานสรุปว่า ลูกค้าคนไหน เช่ารถอะไร วันที่เท่าไร และค่าใช้จ่ายรวม



**รูปภาพที่ 4‑26** Report A

4.4.2.1 อธิบายทีละขั้นตอน

4.4.2.1.1 สร้าง header ตาราง

4.4.2.1.2 ดึง rentals (ล่าสุด 3 รายการ)

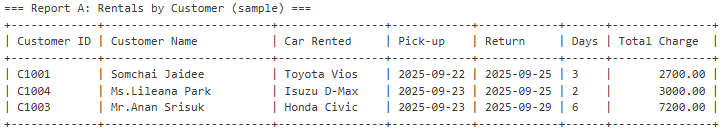
4.4.2.1.3 หาลูกค้า + รถ จาก dict

4.4.2.1.4 คำนวณ วันที่คืนรถ = pickup + days

4.4.2.1.5 คำนวณ ค่าใช้จ่ายรวม = daily\_rate × days

4.4.2.1.5เพิ่มเป็น row ของรายงาน

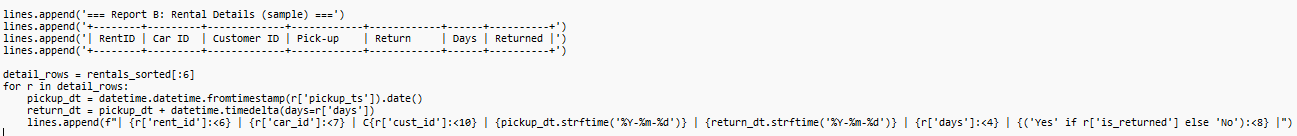
4.4.2.2 ตัวอย่าง Output



**รูปภาพที่ 4‑27** Output Report A

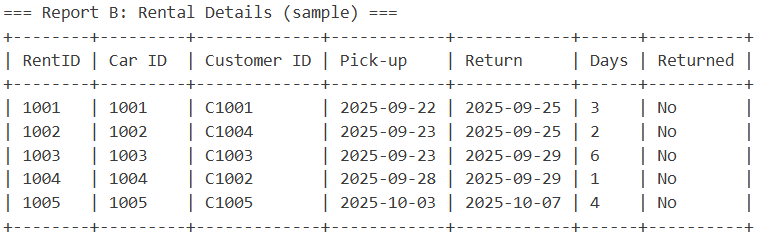
4.4.3 Report B: Rental Details

รายงานนี้เป็น รายการเช่ารถทั้งหมด (หรืออย่างน้อย 6 รายการล่าสุด)



**รูปภาพที่ 4‑28** Report B

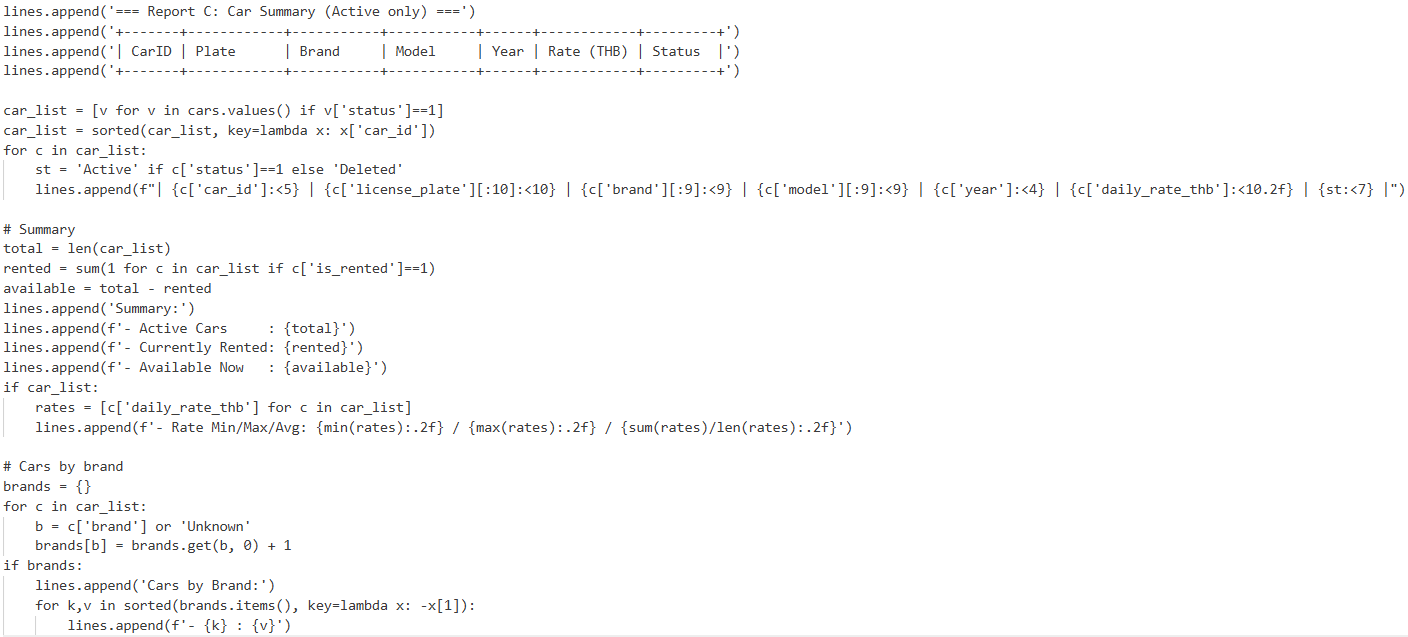
4.4.3.1 ตัวอย่าง Output



**รูปภาพที่ 4‑29** Output Report B

4.4.4 Report C: Car Summary

รายงานนี้สรุป รถทั้งหมดที่ Active + จำนวนรถที่เช่าแล้ว/ว่างอยู่ + ค่าเช่าต่ำสุด/สูงสุด/เฉลี่ย + จำนวนรถตามแบรนด์



**รูปภาพที่ 4‑30** Report C

4.4.4.1 อธิบาย

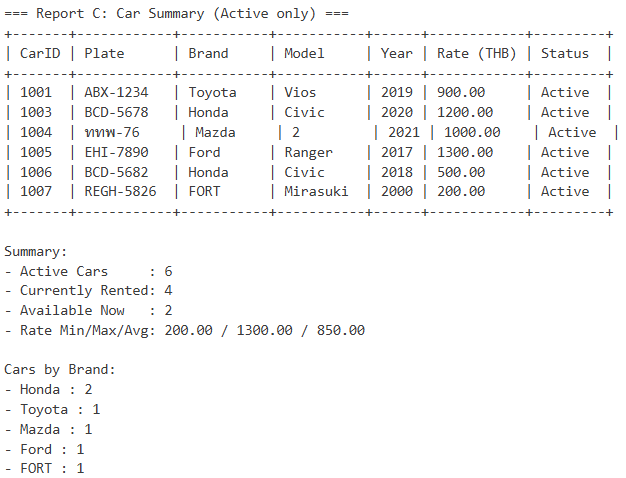
4.4.4.1.1 แสดงตารางรถที่ active เท่านั้น

4.4.4.1.2 นับจำนวนรถ → Active, Rented, Available

4.4.4.1.3 คำนวณ min/max/avg ของ daily\_rate

4.4.4.1.4 นับจำนวนรถตามยี่ห้อ (brand)

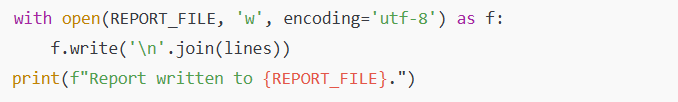
4.4.4.2 ตัวอย่าง Output



**รูปภาพที่ 4‑31** Output Report C and Car Summary

4.4.5 การเขียนไฟล์ Report

สุดท้าย รายงานทั้งหมดถูกบันทึกลง report.txt



**รูปภาพที่ 4‑32** การเขียนไฟล์ Report

## ฟังก์ชันสร้างข้อมูลตัวอย่าง (Sample Data Generators)

ระบบเช่ารถนี้มีฟังก์ชันสำหรับ สร้างข้อมูลตัวอย่างอัตโนมัติ เพื่อช่วยในการทดสอบการทำงาน โดยข้อมูลตัวอย่างจะถูกบันทึกลงไฟล์ .dat เหมือนกับข้อมูลจริง ฟังก์ชันที่มีได้แก่:

1. create\_sample\_cars() สร้างข้อมูลรถ

2. create\_sample\_customers() สร้างข้อมูลลูกค้า

3. create\_sample\_rentals() สร้างข้อมูลการเช่า

4.5.1 สร้างข้อมูลรถ create\_sample\_cars()



**รูปภาพที่ 4‑33** sample\_cars

4.5.1.1 การทำงาน

4.5.1.1.1 กำหนดจำนวน record ที่ต้องการ (n=50)

4.5.1.1.2 สุ่มค่า brand, model, year, daily\_rate, odometer, license\_  
plate

4.5.1.1.3 กำหนด status=1 (active), is\_rented = 30% โอกาสที่รถจะถูกเช่าอยู่

4.5.1.1.4 ใช้ append\_record บันทึกลงไฟล์

4.5.1.2 ผลลัพธ์

4.5.1.2.1 ได้รถตัวอย่าง 50 คันขึ้นไป

4.5.1.2.3 มีความหลากหลายของยี่ห้อ, รุ่น, ราคา

4.5.2 สร้างข้อมูลลูกค้า create\_sample\_customers()



**รูปภาพที่ 4‑34** sample\_customers

4.5.2.1 การทำงาน

4.5.2.1.1 ใช้ชื่อสมมติจาก list เช่น Somchai, Kamoltip, Anan, Lina

4.5.2.1.2 สร้างหมายเลขโทรศัพท์ในรูปแบบ 080-000-xxxx

4.5.2.1.3 สร้างอีเมลสมมติ userxxxx@example.com

4.5.2.1.4 บันทึกลงไฟล์ด้วย append\_record

4.5.2.2 ผลลัพธ์

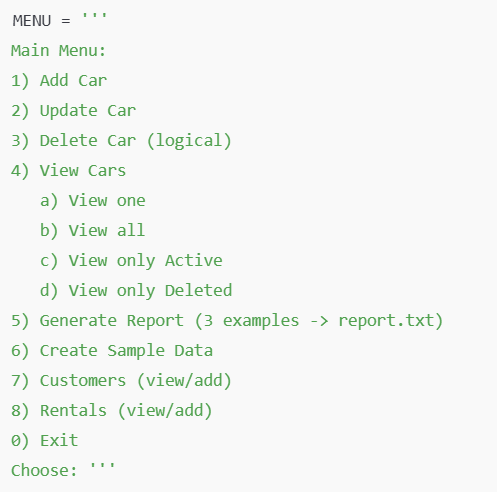
4.5.2.2.1 ได้ลูกค้าตัวอย่าง 10 คน

4.5.2.2.2 แต่ละคนมีข้อมูลพื้นฐานครบ: ชื่อ, เบอร์, อีเมล

## ฟังก์ชันเมนูหลัก (Main Menu และ Sub-menu)

โค้ดในไฟล์ Car Rental.py ได้จัดทำเมนูแบบ Text-based (Console menu) ที่ใช้การวนลูป while True เพื่อรอให้ผู้ใช้เลือกคำสั่ง

4.6.1 Main Menu



**รูปภาพที่ 4‑35** Main Menu

4.6.1.1 อธิบายเมนู

1–3 → จัดการรถ (เพิ่ม, แก้ไข, ลบ)

4 → ดูรถ (แบบย่อย a–d)

5 → สร้างรายงาน

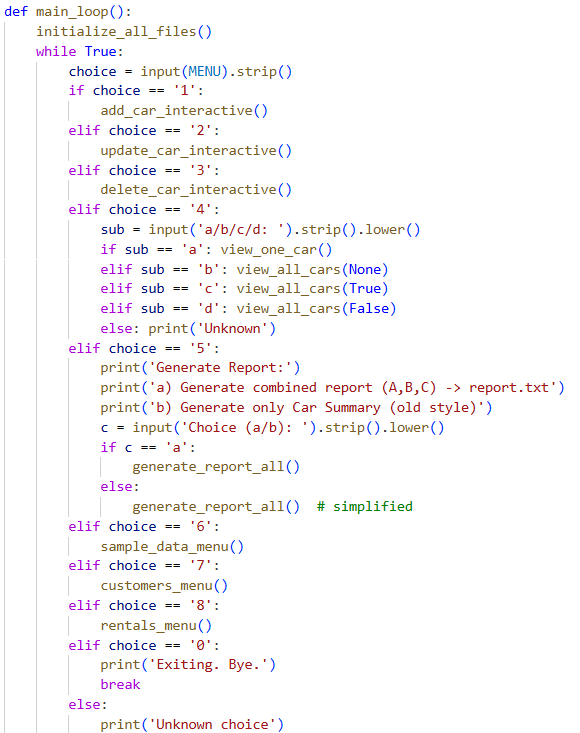
6 → สร้างข้อมูลตัวอย่าง

7 → จัดการลูกค้า

8 → จัดการการเช่า

0 → ออกจากโปรแกรม

4.6.2 ฟังก์ชัน main\_loop()



**รูปภาพที่ 4‑36** main\_loop

4.6.2.1 อธิบายการทำงานทีละขั้นตอน

1. .เริ่มจาก initialize\_all\_files() ตรวจสอบว่ามีไฟล์ .dat หรือยัง ถ้าไม่มีจะสร้างใหม่

2. แสดงเมนู รับค่า choice จากผู้ใช้

3. ใช้ if-elif แยกทำงานตามที่เลือก

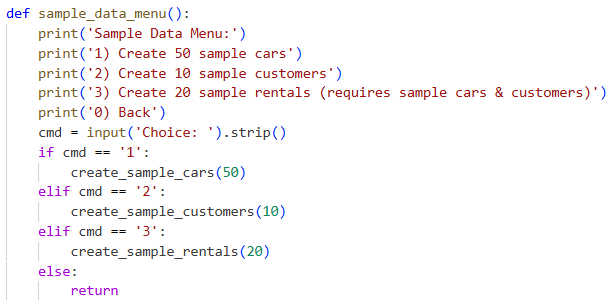
4. เมื่อเลือก 4 (View Cars) มี sub-menu (a–d) ให้เลือกดูข้อมูล

5. เมื่อเลือก 5 (Generate Report) เลือกว่าจะทำรายงานเต็ม (A,B,C) หรือเฉพาะ Car Summary

6. เมื่อเลือก 6–8 เข้าสู่ sub-menu ของ Sample Data, Customers, Rentals

7. ถ้าเลือก 0 → ออกจากลูป

4.6.3 Sub-menu สำหรับ Sample Data



**รูปภาพที่ 4‑37** Sample Data Menu

4.6.3.1 อธิบายฟังก์ชัน

4.6.3.1.1 เลือก 1–3 เพื่อสร้างข้อมูลตัวอย่าง (cars/customers/rentals)

4.6.3.1.2 เลือก 0 → กลับไปเมนูหลัก

4.6.4 Sub-menu สำหรับ Customers



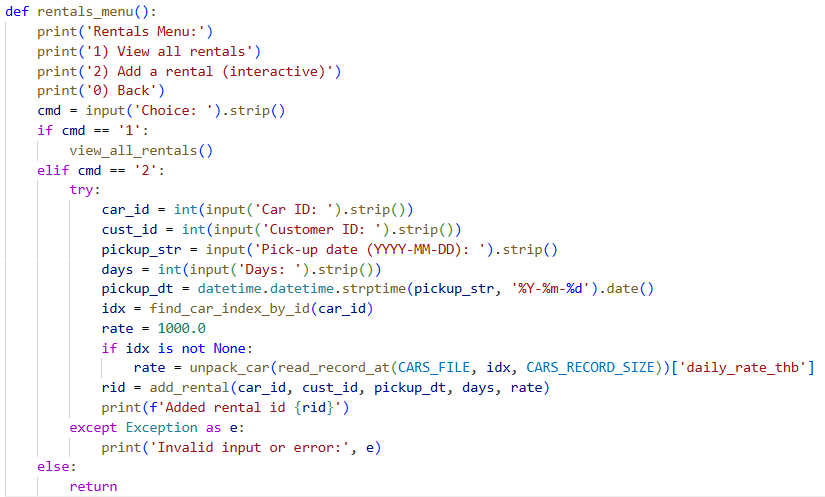
**รูปภาพที่ 4‑38** Customers Menu

4.6.4.1 อธิบาย

4.6.4.1.1 ดูข้อมูลลูกค้าทั้งหมด

4.6.4.1.2 เพิ่มลูกค้าใหม่ (กรอกชื่อ, เบอร์, อีเมล)

4.6.5 Sub-menu สำหรับ Rentals



**รูปภาพที่ 4‑39** Rentals Menu

4.6.5.1 อธิบาย

4.6.5.1.1 ดูรายการเช่าทั้งหมด

4.6.5.1.2 เพิ่มการเช่าใหม่:

4.6.5.1.2.1 ระบุ car\_id และ cust\_id

4.6.5.1.2.2 กำหนดวันเริ่มต้น (pickup date) และจำนวนวัน

4.6.5.1.2.3 ใช้ daily\_rate ของรถจริง (ถ้ามี)

4.6.5.1.2.4 เรียก add\_rental() เพื่อบันทึก

## สรุปการทำงานของระบบทั้งหมด

ระบบ Car Rental Management System ที่พัฒนาด้วยภาษา Python และใช้ Binary File ในการเก็บข้อมูล ได้ถูกออกแบบให้ทำงานตามลำดับขั้นตอนและฟังก์ชันหลักดังนี้

4.7.1 โครงสร้างข้อมูล (Data Structures)

ระบบเก็บข้อมูล 3 ประเภทหลักในไฟล์แยกกัน:

4.7.1.1 Cars เก็บข้อมูลรถแต่ละคัน เช่น id, brand, model, year, daily\_rate, status

4.7.1.2 Customers เก็บข้อมูลลูกค้า เช่น id, name, phone, email

4.7.1.3 Rentals เก็บข้อมูลการเช่า เช่น id, car\_id, cust\_id, pickup\_date, days, is\_returned

4.7.1.4 รูปแบบไฟล์: Binary File (fixed-length records)

4.7.1.4.1. ใช้ pack\_\* และ unpack\_\* ในการแปลงระหว่าง dict กับ binary

4.7.1.4.2. แต่ละไฟล์มี header เพื่อเก็บ metadata (เช่น count, next\_id)

4.7.2 การทำงานของฟังก์ชันหลัก

4.7.2.1 ฟังก์ชันพื้นฐาน (4.1)

4.7.2.1.1 pack\_car(), pack\_customer(), pack\_rental() → แปลง dict → binary

4.7.2.1.2 unpack\_car() ฯลฯ → binary → dict

4.7.2.1.3. append\_record(), read\_record\_at(), write\_record\_at() → จัดการบันทึก/แก้ไขไฟล์

4.7.2.2 ฟังก์ชันอ่าน–เขียนไฟล์ (4.2)

4.7.2.2.1 จัดการ header ของไฟล์ (count, next\_id)

4.7.2.2.2 จัดเก็บ record แบบ fixed size

4.7.2.2.3 รองรับการแก้ไขและลบแบบ logical delete

4.7.2.3 ฟังก์ชัน CRUD (4.3)

4.7.2.3.1 Cars: เพิ่ม, แก้ไข, ลบ (logical), ดูข้อมูล

4.7.2.3.2 Customers: เพิ่ม, ดูข้อมูล

4.7.2.3.3 Rentals: เพิ่ม (พร้อมอัปเดตสถานะรถ), ดูข้อมูล

4.7.2.4 ฟังก์ชันรายงาน (4.4)

4.7.2.4.1 Report A: Rentals by Customer → สรุปว่าลูกค้าเช่าอะไร, วันไหน, รวมค่าใช้จ่าย

4.7.2.4.2 Report B: Rental Details → รายละเอียดการเช่าทั้งหมด

4.7.2.4.3 Report C: Car Summary → รายการรถทั้งหมด Active/  
Deleted, ค่าธรรมเนียม min/max/avg, จำนวนรถแต่ละแบรนด์

4.7.2.5 ฟังก์ชันสร้างข้อมูลตัวอย่าง (4.5)

4.7.2.5.1. create\_sample\_cars() → สร้างรถตัวอย่าง 50 คัน

4.7.2.5.2. create\_sample\_customers() → ลูกค้าตัวอย่าง 10 คน

4.7.2.5.3. create\_sample\_rentals() → ประวัติการเช่า 10–20 รายการ

4.7.2.6 ฟังก์ชันเมนูหลัก (4.6)

4.7.2.6.1 Main Menu ผู้ใช้เลือกการทำงาน: Cars, Customers, Rentals, Report, Sample Data

4.7.2.6.2 Sub-menu แบ่งย่อยแต่ละส่วน (view, add, update, delete)

4.7.2.6.3 ใช้ while True ให้ผู้ใช้สั่งงานต่อเนื่องจนกว่าจะ exit

4.7.3 การทำงานเชื่อมโยง (Workflow)

ภาพรวมการทำงานของระบบคือ

4.7.3.1 ผู้ใช้เปิดโปรแกรม

4.7.3.1.1 ระบบตรวจสอบไฟล์ .dat ทั้งหมด

4.7.3.1.2 ถ้าไม่มี สร้างไฟล์ใหม่พร้อม header

4.7.3.2 ผู้ใช้เลือกเมนู

4.7.3.2.1 จัดการรถ → เพิ่ม/แก้ไข/ลบ/ดู

4.7.3.2.2 จัดการลูกค้า → เพิ่ม/ดู

4.7.3.2.3 จัดการการเช่า → เพิ่ม/ดู

4.7.3.2.4 สร้างรายงาน → สรุปผลรวมใน report.txt

4.7.3.2.5 สร้างข้อมูลตัวอย่าง → เติมข้อมูลจำลองเพื่อทดสอบ

4.7.3.3 ระบบทำงานกับไฟล์

4.7.3.3.1 ทุกการเปลี่ยนแปลง (เพิ่ม, แก้ไข, ลบ) เขียนลง binary file โดยตรง

4.7.3.3.2 ทุกการดูข้อมูล → อ่านจากไฟล์ + แปลงเป็นตาราง

4.7.3.4 ผู้ใช้สิ้นสุดการทำงาน

4.7.3.4.1 กด 0 (Exit) → โปรแกรมจบการทำงาน

4.7.4 จุดเด่นของระบบ

4.7.4.1 ใช้ Binary File + Fixed-Length Record → ข้อมูลเข้าถึงได้เร็วและไม่ต้องพึ่งฐานข้อมูล

4.7.4.2 รองรับ CRUD ครบ → Cars, Customers, Rentals

4.7.4.3 สามารถสร้าง Report สรุปแบบอัตโนมัติ → ช่วยผู้ดูแลตรวจสอบสถานะรถและลูกค้าได้ง่าย

4.7.4.4 มี ฟังก์ชันสร้างข้อมูลจำลอง → สะดวกสำหรับการทดสอบ

4.7.4.5 ใช้ เมนูแบบ Text-based → เข้าใจง่ายและใช้งานได้ทันที

4.7.5 ข้อสังเกตและข้อจำกัด

4.7.5.1 ระบบใช้ไฟล์แบบ binary → ยากต่อการแก้ไขข้อมูลโดยตรง (ต้องผ่านโปรแกรม)

4.7.5.2 การลบเป็น Logical Delete (status=0) → ข้อมูลยังอยู่ในไฟล์ ไม่ได้ถูกลบจริง

4.7.5.3 ไม่มีระบบ ตรวจสอบสิทธิ์ (Authentication) → ใครก็เข้ามาแก้ไขได้

4.7.5.4 ยังไม่รองรับการทำงานแบบ หลายผู้ใช้พร้อมกัน (Concurrency)

# สรุปผลการดำเนินงานและข้อเสนอแนะ

## สรุปผลการดำเนินงาน

ระบบจัดการการเช่ารถที่พัฒนาขึ้นนี้ สามารถดำเนินการจัดเก็บและจัดการข้อมูล รถ ลูกค้า และรายการเช่าได้อย่างมีประสิทธิภาพ โดยมีการจัดเก็บข้อมูลในรูปแบบไฟล์ไบนารี (binary files) ซึ่งช่วยให้การเข้าถึงข้อมูลรวดเร็ว และทำให้สามารถสร้างรายการสรุปผลการดำเนินงาน เช่น จำนวนรถที่มีอยู่ ข้อมูลลูกค้า และสถิติการเช่าโดยรวมได้อย่างสะดวก ข้อมูลเหล่านี้มีประโยชน์อย่างยิ่งต่อการบริหารจัดการระบบการเช่ารถ

## ปัญหาและอุปสรรคในการทำงาน

ในการพัฒนาระบบนี้ ปัญหาหลัก คือความซับซ้อนของการจัดการไฟล์ไบนารี (binary files) เนื่องจากต้องกำหนดโครงสร้างข้อมูล (struct format) สำหรับไฟล์ข้อมูลรถ (cars.dat) ลูกค้า (customers.dat) และการเช่า (rentals.dat) อย่างแม่นยำ ซึ่งอาจเกิดข้อผิดพลาดในการเขียนหรือถอดรหัสข้อมูลที่ไม่ถูกต้องได้ นอกจากนี้ การจัดการข้อมูลแบบไฟล์ไบนารียังไม่รองรับการทำงานพร้อมกันของผู้ใช้งานหลายคน (multi-user concurrency) ทำให้เกิดข้อจำกัดในการจัดการข้อมูลจำนวนมากหรือการเข้าถึงพร้อมกัน

## ข้อเสนอแนะ

เพื่อปรับปรุงและขยายขีดความสามารถของระบบให้พร้อมใช้งานในอนาคต มีข้อเสนอแนะ ดังนี้:

5.3.1 ปรับเปลี่ยนมาใช้ระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database) เช่น MySQL หรือ SQLite เพื่อรองรับการจัดเก็บข้อมูลจำนวนมากและให้การเข้าถึงข้อมูลหลายผู้ใช้พร้อมกัน

5.3.2 เพิ่มเติมฟังก์ชันการค้นหาและกรองข้อมูล เช่น ค้นหารถตามยี่ห้อ รุ่น หรือราคา, ค้นหาลูกค้าตามชื่อ หรือ ID

5.3.3 ปรับปรุงระบบยืนยันตัวตน และกำหนดสิทธิ์การเข้าถึงของผู้ใช้แต่ละประเภท\*\* (เช่น พนักงาน, ผู้จัดการ) เพื่อเพิ่มความปลอดภัย

5.3.4 พัฒนาโปรแกรมให้มีส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) หรือเว็บแอปพลิเคชัน เพื่อเพิ่มความสะดวกในการใช้งานให้แก่ผู้ใช้

## สิ่งที่ผู้จัดทำได้รับในการพัฒนาโครงการ

จากการพัฒนาโครงงานระบบจัดการการเช่ารถในครั้งนี้ ผู้จัดทำได้รับความรู้และประสบการณ์ ด้านการออกแบบระบบ การเขียนโปรแกรมด้วยภาษา Python การใช้โครงสร้างข้อมูลแบบไบนารี (binary data structures) รวมถึงการคิดวิเคราะห์และแก้ไขปัญหาเชิงตรรกะ นอกจากนี้ยังได้ฝึกทักษะการทำงานเป็นทีม การแบ่งหน้าที่รับผิดชอบ และการจัดเวลาให้สอดคล้องกับแผนงาน ทำให้ผู้จัดทำมีความเข้าใจในกระบวนการพัฒนาระบบซอฟต์แวร์มากยิ่งขึ้น และสามารถนำไปประยุกต์ใช้ในโครงการหรือการทำงานจริงในอนาคตได้เป็นอย่างดี