**ระบบจัดการข้อมูลร้านสะดวกซื้อ**

**Convenience Store Management System**

**นายตะวัน คำหนัน รหัส 6706022610128**

**นายธีรดนย์ จันพอน รหัส 6706022610438**

**นางอัมมาล อาแว รหัส 6706022610152**

**นายก้องเกียรติ กองอรรถ รหัส 6706022610225**

**โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิตสาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศคณะเทคโนโลยีและการจัดการอุตสาหกรรม**

**มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ**

**ปีการศึกษา 2567**

**ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ**

# คำนำ

การจัดทําโครงงาน “ระบบจัดการข้อมูลร้านสะดวกซื้อ” นี้เป็นส่วนหนึ่งของวิชาCOMPUTER PROGRAMMIMG ของหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศคณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยี พระจอมเกล้าพระนครเหนือ เพื่อให้นักศึกษาได้นําความรู้ที่เรียนมาทั้งหมดมาประยุกต์ใช้ในการ พัฒนาโปรแกรมที่สามารถทํางานได้จริง โดยเน้นการออกแบบและเขียนโปรแกรมในภาษา Python ซึ่งเป็นภาษาที่เรียนมาในวิชาCOMPUTER PROGRAMMING โดยโครงงานนี้จะช่วย การคิดวิเคราะห์ และแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมในการประกอบอาชีพด้านวิศวกรรสารสนเทศและ เครือข่ายในอนาคต หากมีข้อผิดพลาดประการใด คณะผู้จัดทําต้องขออภัยไว้ ณ ที่นี้ด้วย

# สารบัญ

หน้า

[คำนำ ข](#_Toc179385766)

[สารบัญ ค](#_Toc179385767)

[สารบัญภาพ ช](#_Toc179385768)

[บทที่ 1 บทนำ 1](#_Toc179385769)

[1.1 วัตถุประสงค์ของโครงงาน 1](#_Toc179385770)

[1.2 ขอบเขตของโครงงาน 1](#_Toc179385775)

[1.3 ประโยชน์ที่ได้รับ 1](#_Toc179385780)

[1.4 เครื่องมือที่คาดว่าจะต้องใช้ 1](#_Toc179385785)

[บทที่ 2 ระบบจัดการข้อมูลสินค้า 2](#_Toc179385788)

[2.1 ฟิลด์ในระบบจัดการข้อมูลสินค้า 2](#_Toc179385789)

[2.1.1 Product\_id รหัสสินค้า (Identifier) 2](#_Toc179385790)

[2.1.2 Name ชื่อสินค้า (Product Name) 2](#_Toc179385791)

[2.1.3 price ราคาของหนังสือ (Price) 2](#_Toc179385792)

[2.1.4 category หมวดหมู่ของสินค้า (Category) 3](#_Toc179385793)

[2.1.5 stock\_status สถานะสต็อกของหนังสือ (Stock Status) 3](#_Toc179385794)

[2.1.6 สรุปฟิลด์ในระบบจัดการข้อมูลสินค้า 3](#_Toc179385795)

[2.2 ฟังก์ชันพื้นฐานในระบบจัดการข้อมูลร้านค้า 4](#_Toc179385796)

[2.2.1 โครงสร้างและการทำงานของฟังก์ชัน main() 4](#_Toc179385797)

[2.2.1.1 โครงสร้างและการทำงานของฟังก์ชัน main() 4](#_Toc179385798)

[2.2.1.2 โค้ดของฟังก์ชัน main() 5](#_Toc179385799)

[2.2.1.3 แสดงหัวข้อของโปรแกรม 7](#_Toc179385800)

[2.2.1.4 แสดงเมนูให้ผู้ใช้เลือก 8](#_Toc179385801)

[2.2.1.5 ตรวจสอบการเลือกของผู้ใช้ 9](#_Toc179385802)

[2.2.1.6 การวนลูปกลับสู่เมนูหลัก 10](#_Toc179385803)

[2.2.1.7 การออกจากโปรแกรม 10](#_Toc179385804)

[2.2.1.8 สรุปการทํางานของฟังก์ชัน main() 10](#_Toc179385805)

[2.2.1.9 ผลลัพธ์ของฟังก์ชัน main() 10](#_Toc179385806)

[2.2.2 โครงสร้างและการทำงานของฟังก์ชัน add\_record() 11](#_Toc179385807)

[2.2.2.1 การทำงานของฟังก์ชัน 11](#_Toc179385808)

[2.2.2.2 เปิดไฟล์ในโหมดเพิ่มข้อมูล 12](#_Toc179385809)

[2.2.2.3 บีบอัดข้อมูล 12](#_Toc179385810)

[2.2.2.4 บันทึกข้อมูลลงในไฟล์ 13](#_Toc179385811)

[2.2.2.5 การแสดงผลของฟังก์ชั่น add\_record() 13](#_Toc179385812)

[2.2.2.6 ผลลัพธ์ของฟังก์ชั่น add\_record() 13](#_Toc179385813)

[2.2.3 โครงสร้างและการทำงานของฟังก์ชัน display\_records() 14](#_Toc179385814)

[2.2.3.1 ตรวจสอบการมีอยู่ของไฟล์ 14](#_Toc179385815)

[2.2.3.2 แสดงหัวข้อข้อมูล 14](#_Toc179385816)

[2.2.3.3 เปิดไฟล์ในโหมดอ่านข้อมูล 15](#_Toc179385817)

[2.2.3.4 อ่านข้อมูลจากไฟล์ 15](#_Toc179385818)

[2.2.3.5 แยกข้อมูลและแสดงผล 15](#_Toc179385819)

[2.2.3.6 แสดงเส้นขอบด้านล่าง 16](#_Toc179385820)

[2.2.3.7 การทำงานโดยรวมของฟังก์ชัน display\_records() 16](#_Toc179385821)

[2.2.3.8 ผลลัพธ์ของฟังก์ชัน display\_record() 17](#_Toc179385822)

[2.2.4 โครงสร้างและการทำงานของฟังก์ชัน retrieve\_records() 17](#_Toc179385823)

[2.2.4.1 ตรวจสอบการมีอยู่ของไฟล์ 17](#_Toc179385824)

[2.2.4.2 ตัวแปรสำหรับตรวจสอบการพบข้อมูล 18](#_Toc179385825)

[2.2.4.3 เปิดไฟล์ในโหมดอ่านข้อมูล 18](#_Toc179385826)

[2.2.4.4 อ่านข้อมูลจากไฟล์ 19](#_Toc179385827)

[2.2.4.5 แยกข้อมูลและตรวจสอบการจับคู่ 19](#_Toc179385828)

[2.2.4.6 ค้นหาข้อมูลที่ตรงกัน 20](#_Toc179385829)

[2.2.4.7 แสดงผลเมื่อพบข้อมูลที่ตรงกัน 20](#_Toc179385830)

[2.2.4.8 ตรวจสอบการไม่พบข้อมูล 20](#_Toc179385831)

[2.2.4.9 การทำงานโดยรวมของฟังก์ชัน retrieve\_records() 21](#_Toc179385832)

[2.2.4.10 ผลลัพธ์ของฟังก์ชัน retrieve\_record() 21](#_Toc179385833)

[2.2.5 โครงสร้างและการทำงานของฟังก์ชัน update\_record() 22](#_Toc179385834)

[2.2.5.1 รับพารามิเตอร์ที่ต้องการสำหรับการอัปเดต 22](#_Toc179385835)

[2.2.5.2 ตัวแปรเก็บข้อมูลทั้งหมด 22](#_Toc179385836)

[2.2.5.3 ตัวแปรเพื่อตรวจสอบการอัปเดต 22](#_Toc179385837)

[2.2.5.4 เปิดไฟล์เพื่ออ่านข้อมูล 22](#_Toc179385838)

[2.2.5.5 ลูปอ่านข้อมูล 23](#_Toc179385839)

[2.2.5.6 ตรวจสอบและอัปเดตข้อมูล 23](#_Toc179385840)

[2.2.5.7 เขียนข้อมูลที่อัปเดตกลับไปยังไฟล์: 24](#_Toc179385841)

[2.2.5.8 แสดงผลการอัปเดต 24](#_Toc179385842)

[2.2.5.9 การทำงานโดยรวมของฟังก์ชัน update\_record() 25](#_Toc179385843)

[2.2.5.10 ผลลัพธ์ของฟังก์ชัน display\_record() 25](#_Toc179385844)

[2.2.6 โครงสร้างและการทำงานของฟังก์ชัน delete\_record() 26](#_Toc179385845)

[2.2.6.1 ประกาศตัวแปรสำหรับเก็บข้อมูลและสถานะการลบ 26](#_Toc179385846)

[2.2.6.2 เปิดไฟล์ในโหมดอ่าน 26](#_Toc179385847)

[2.2.6.3 ลูปอ่านข้อมูลจากไฟล์ 26](#_Toc179385848)

[2.2.6.4 ตรวจสอบและลบข้อมูล 27](#_Toc179385849)

[2.2.6.5 เขียนข้อมูลที่เหลือกลับลงในไฟล์ 27](#_Toc179385850)

[2.2.6.6 แสดงผลการลบข้อมูล 27](#_Toc179385851)

[2.2.6.7 การทำงานโดยรวมของฟังก์ชัน delete\_record() 27](#_Toc179385852)

[2.2.6.8 ผลลัพธ์การทำงานของฟังก์ชัน delete\_record() 28](#_Toc179385853)

[2.2.7 โครงสร้างและการทำงานของฟังก์ชัน create\_report() 29](#_Toc179385854)

[2.2.7.1 ตรวจสอบการมีอยู่ของไฟล์ 29](#_Toc179385855)

[2.2.7.2 สร้างตัวแปรสำหรับรายงาน 29](#_Toc179385856)

[2.2.7.3 เปิดไฟล์ในโหมดอ่าน 29](#_Toc179385857)

[2.2.7.4 อ่านและจัดรูปแบบข้อมูล 29](#_Toc179385858)

[2.2.7.5 เขียนรายงานลงในไฟล์ 30](#_Toc179385859)

[2.2.7.6 แสดงผลลัพธ์การสร้างรายงาน 30](#_Toc179385860)

[2.2.7.7 การทำงานโดยรวมของฟังก์ชัน create\_report() 30](#_Toc179385861)

[2.2.7.8 ผลลัพธ์ของฟังก์ชัน create\_report() 31](#_Toc179385862)

# สารบัญภาพ

ภาพที่ หน้า

[ภาพที่ 2‑1 โค้ดของฟังก์ชัน main() 5](#_Toc179386195)

[ภาพที่ 2‑2 โค้ดของฟังก์ชัน main() (2) 6](#_Toc179386196)

[ภาพที่ 2‑3 แสดงหัวข้อของฟังก์ชัน main() 7](#_Toc179386197)

[ภาพที่ 2‑4 แสดงเมนูให้ผู้ใช้เลือกของฟังก์ชัน main() 8](#_Toc179386198)

[ภาพที่ 2‑5 การเลือกของผู้ใช้ฟังก์ชันหมายเลขที่ไม่อยู่ในช่วง 1-7 main() 9](#_Toc179386199)

[ภาพที่ 2‑6 การออกจากโปรแกรมฟังก์ชัน main() 10](#_Toc179386200)

[ภาพที่ 2‑7 ผลลัพธ์ของฟังก์ชัน main() 10](#_Toc179386201)

[ภาพที่ 2‑8 การใส่ข้อมูลสินค้าใหม่ main() 11](#_Toc179386202)

[ภาพที่ 2‑9 การเปิดไฟล์ในโหมด 'ab' add\_record() 12](#_Toc179386203)

[ภาพที่ 2‑10 การบีบอัดข้อมูลในรูปแบบ binary 12](#_Toc179386204)

[ภาพที่ 2‑11 ข้อมูลที่ใส่บีบอัดลงไฟล์ add\_record() 13](#_Toc179386205)

[ภาพที่ 2‑12 ผลลัพธ์ของฟังก์ชั่น add\_record() 13](#_Toc179386206)

[ภาพที่ 2‑13 การตรวจสอบไฟล์ระบบ display\_records() 14](#_Toc179386207)

[ภาพที่ 2‑14 เตรียมแสดงข้อมูลหากพบไฟล์ display\_record() 14](#_Toc179386208)

[ภาพที่ 2‑15 เปิดไฟล์ data.bin ในโหมด 'rb' display\_record() 15](#_Toc179386209)

[ภาพที่ 2‑16 ใช้ลูป while เพื่ออ่านข้อมูลจากไฟล์เป็นชุด display\_record() 15](#_Toc179386210)

[ภาพที่ 2‑17 การแยกข้อมูลออกเป็นฟิลด์ต่าง ๆ display\_record() 15](#_Toc179386211)

[ภาพที่ 2‑18 แสดงข้อมูลแต่ละรายการ display\_record() 16](#_Toc179386212)

[ภาพที่ 2‑19 พิมพ์เส้นขอบด้านล่าง display\_record() 16](#_Toc179386213)

[ภาพที่ 2‑20 ทำงานโดยรวมของฟังก์ชัน display\_records() 16](#_Toc179386214)

[ภาพที่ 2‑21 ผลลัพธ์ของฟังก์ชัน display\_record() 17](#_Toc179386215)

[ภาพที่ 2‑22 การตรวจสอบไฟล์ในระบบ retrieve\_records() 18](#_Toc179386216)

[ภาพที่ 2‑23 การตรวจสอบข้อมูลที่ตรง retrieve\_records() 18](#_Toc179386217)

[ภาพที่ 2‑24 ฟังก์ชันจะเปิดไฟล์ในโหมด 'rb' retrieve\_records() 18](#_Toc179386218)

[ภาพที่ 2‑25 ใช้ลูป while เพื่ออ่านข้อมูลจากไฟล์เป็นชุด ๆ retrieve\_records() 19](#_Toc179386219)

[ภาพที่ 2‑26 การแยกข้อมูลออกเป็นฟิลด์ต่าง ๆ retrieve\_records() 19](#_Toc179386220)

[ภาพที่ 2‑27 แปลงจากไบนารีเป็นข้อความ retrieve\_records() 19](#_Toc179386221)

[ภาพที่ 2‑28 การตรวจสอบว่าค่าที่ป้อนเข้ามาตรงกับ record\_id หรือ name 20](#_Toc179386222)

[ภาพที่ 2‑29 แสดงผลบันทึกที่ตรงกันในรูปแบบที่เป็นมิตรต่อผู้ใช้ 20](#_Toc179386223)

[ภาพที่ 2‑30 แสดงข้อความว่ามีข้อมูลที่ค้นหาไม่พบ recrieve\_record() 20](#_Toc179386224)

[ภาพที่ 2‑31 การทำงานโดยรวมของฟังก์ชัน retrieve\_records() 21](#_Toc179386225)

[ภาพที่ 2‑32 ผลลัพธ์ของฟังก์ชัน retrieve\_record() 21](#_Toc179386226)

[ภาพที่ 2‑33 การเก็บข้อมูลทั้งหมดที่ถูกอ่านมาจากไฟล์ 22](#_Toc179386227)

[ภาพที่ 2‑34 การตรวจสอบว่ามีการอัปเดต 22](#_Toc179386228)

[ภาพที่ 2‑35 เปิดไฟล์ในโหมด 'rb' update\_record() 22](#_Toc179386229)

[ภาพที่ 2‑36 ลูปอ่านข้อมูลเพื่อแปลงข้อมูลออกเป็นฟิลด์ updated\_record() 23](#_Toc179386230)

[ภาพที่ 2‑37 ถ้า rec\_id ของบันทึกตรงกับ record\_id ที่ต้องการอัปเดต 23](#_Toc179386231)

[ภาพที่ 2‑38 แต่ถ้า rec\_id ไม่ตรงกัน 23](#_Toc179386232)

[ภาพที่ 2‑39 เปิดไฟล์อีกครั้งในโหมด 'wb' update\_record() 24](#_Toc179386233)

[ภาพที่ 2‑40 หากมีการอัปเดตข้อมูลฟังก์ชันจะแสดงข้อความ update\_record() 24](#_Toc179386234)

[ภาพที่ 2‑41 การทำงานโดยรวมของฟังก์ชัน update\_record() 25](#_Toc179386235)

[ภาพที่ 2‑42 ผลลัพธ์ของฟังก์ชัน display\_record() 25](#_Toc179386236)

[ภาพที่ 2‑43 ประกาศตัวแปรสำหรับเก็บข้อมูลและสถานการณ์ลบ 26](#_Toc179386237)

[ภาพที่ 2‑44 เปิดไฟล์ไบนารีในโหมดอ่าน delete\_record() 26](#_Toc179386238)

[ภาพที่ 2‑45 ใช้ลูปในการอ่านข้อมูลทีละบันทึก delelte\_record() 26](#_Toc179386239)

[ภาพที่ 2‑46 ตรวจสอบและลบข้อมูล delete\_record() 27](#_Toc179386240)

[ภาพที่ 2‑47 เปิดไฟล์ในโหมดเขียน ('wb') delete\_record() 27](#_Toc179386241)

[ภาพที่ 2‑48 แสดงผลการลบข้อมูล delete\_record() 27](#_Toc179386242)

[ภาพที่ 2‑49 การทำงานโดยรวมของฟังก์ชัน delete\_record() 28](#_Toc179386243)

[ภาพที่ 2‑50 ผลลัพธ์การทำงานของฟังก์ชัน delete\_record() 28](#_Toc179386244)

[ภาพที่ 2‑51 ตรวจสอบการมีอยู่ของไฟล์ create\_report() 29](#_Toc179386245)

[ภาพที่ 2‑52 สร้างตัวแปรสำหรับรายงาน create\_report() 29](#_Toc179386246)

[ภาพที่ 2‑53 เปิดไฟล์ในโหมดอ่าน create\_report() 29](#_Toc179386247)

[ภาพที่ 2‑54 อ่านและจัดรูปแบบข้อมูล create\_report() 30](#_Toc179386248)

[ภาพที่ 2‑55 เขียนรายงานลงในไฟล์ create\_report() 30](#_Toc179386249)

[ภาพที่ 2‑56 แสดงผลลัพธ์การสร้างรายงาน create\_report() 30](#_Toc179386250)

[ภาพที่ 2‑57 การทำงานโดยรวมของฟังก์ชัน create\_report() 31](#_Toc179386251)

[ภาพที่ 2‑58 ผลลัพธ์ของฟังก์ชัน create\_report() 31](#_Toc179386252)

# บทนำ

## วัตถุประสงค์ของโครงงาน

### เพื่อพัฒนาระบบที่สามารถจัดการหนังสือได้อย่างมีประสิทธิภาพ

### เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วย Python

### เพื่อเรียนรู้การจัดการข้อมูลและไฟล์

### เพื่อเรียนรู้การทํางานร่วมกันเป็นทีม

## ขอบเขตของโครงงาน

### ระบบจัดการข้อมูลสินค้าจะมีฟังก์ชันพื้นฐาน 8 ฟังก์ชัน เช่น 1. เมนูหลัก 2. การแสดง รายการหนังสือ 3. การเพิ่มหนังสือใหม่ 4. การอัปเดตสินค้า 5. การค้นหาสินค้า 6. การลบสินค้า 7. สร้างรายงาน 8. ออกจากโปรแกรม

### ระบบจัดการข้อมูลหนังสือ ประกอบด้วย 5 ฟิลด์ ได้แก่ 1. record\_id 2. name 3. price 4. category 5. stock\_status

### ระบบจัดการข้อมูลสินค้ามีการจัดเก็บข้อมูลหนังสือไว้ในไฟล์ text file

### ระบบจัดการข้อมูลสินค้าจะมีเมนูเพื่อให้ผู้ใช้สามารถเลือกดําเนินการได้

## ประโยชน์ที่ได้รับ

### พัฒนาระบบที่สามารถจัดการหนังสือได้อย่างมีประสิทธิภาพ

### พัฒนาทักษะการเขียนโปรแกรม

### เรียนรู้การจัดการข้อมูลและไฟล์

### เรียนรู้การทํางานร่วมกันเป็นทีม

## เครื่องมือที่คาดว่าจะต้องใช้

### ภาษา Python

### Microsoft office

# ระบบจัดการข้อมูลสินค้า

## ฟิลด์ในระบบจัดการข้อมูลสินค้า

การจัดการข้อมูลสินค้าในระบบของคุณประกอบด้วย 5 ฟิลด์หลัก ซึ่งแต่ละฟิลด์

มีรายละเอียดและความสําคัญดังนี้

### **Product\_id รหัสสินค้า (Identifier)**

record\_id เป็นรหัสประจำสินค้าที่ใช้ในการระบุสินค้าแต่ละชิ้นอย่างเฉพาะเจาะจง ซึ่งมักจะเป็นตัวเลขที่ไม่ซ้ำกันในระบบ เช่น 1000, 1001, 1002 เป็นต้น รูปแบบของฟิลด์นี้เป็นประเภทข้อมูล **integer** (จำนวนเต็ม) มีความสำคัญในการค้นหาหรืออ้างอิงสินค้าในฐานข้อมูลได้ง่าย และช่วยป้องกันความสับสนระหว่างสินค้าที่อาจมีชื่อหรือรายละเอียดคล้ายกัน

### **Name ชื่อสินค้า (Product Name)**

name คือชื่อของสินค้า ซึ่งสามารถระบุถึงชื่อเฉพาะของสินค้าชนิดนั้นๆ  
รูปแบบของฟิลด์นี้เป็นประเภทข้อมูล string (ข้อความ) เช่น "Water", "Apple", "Carrot" เป็นต้น ความสำคัญของฟิลด์คือชื่อสินค้านี้เป็นข้อมูลที่สำคัญสำหรับการค้นหาและการแสดงผลข้อมูลสินค้า โดยเฉพาะเมื่อผู้ใช้ต้องการค้นหาสินค้าโดยชื่อ ซึ่งช่วยในการจัดการสินค้าภายในระบบได้อย่างมีประสิทธิภาพและรวดเร็ว

### **price ราคาของหนังสือ (Price)**

price คือ ราคาของหนังสือซึ่งแสดงถึงมูลค่าของหนังสือที่กำหนด ซึ่งรูปแบบของฟิลด์เป็นประเภทข้อมูล float (ทศนิยม) เช่น 150.50, 250.00 เป็นต้น ความสำคัญของฟิลด์นี้คือ ข้อมูลราคามีความสำคัญสำหรับผู้ใช้ในการตัดสินใจซื้อหนังสือ และช่วยให้ผู้จัดการระบบสามารถคำนวณยอดขายหรือกำไรได้ ข้อมูลราคายังช่วยให้ผู้ใช้เปรียบเทียบราคาของหนังสือเล่มต่าง ๆ ได้อย่างสะดวก โดยเฉพาะในกรณีที่มีหนังสือหลายเล่มที่มีความคล้ายคลึงกัน

### **category หมวดหมู่ของสินค้า (Category)**

category คือ หมวดหมู่ของสินค้าที่ระบุประเภทหรือกลุ่มของหนังสือ เช่น Drink (เครื่องดื่ม), Vegetable (ผัก), Fruit (ผลไม้) เป็นต้น รูปแบบของฟิลด์เป็นประเภทข้อมูล string (ข้อความ) ความสำคัญของฟิลด์นี้คือ การจัดหมวดหมู่หนังสือช่วยให้ผู้ใช้สามารถค้นหาหนังสือได้ง่ายขึ้น และช่วยในการจัดการข้อมูลหนังสือในระบบให้มีระเบียบเรียบร้อย นอกจากนี้ยังช่วยในการสร้างรายงานหรือสถิติที่เกี่ยวข้องกับประเภทหนังสือ

### **stock\_status สถานะสต็อกของหนังสือ (Stock Status)**

stock\_status คือ สถานะของสต็อกหนังสือซึ่งบ่งบอกว่าสินค้าเล่มนั้นมีอยู่ในสต็อกหรือไม่ โดยมีค่าที่เป็นไปได้เช่น "In Stock" หรือ "Out Stock" รูปแบบของฟิลด์เป็นประเภทข้อมูล string (ข้อความ) ความสำคัญของฟิลด์นี้คือ สถานะสต็อกช่วยให้ผู้ใช้สามารถทราบได้ว่าหนังสือที่ต้องการมีพร้อมให้สั่งซื้อหรือไม่ ซึ่งจะช่วยให้ผู้ใช้สามารถตัดสินใจซื้อได้

### **สรุปฟิลด์ในระบบจัดการข้อมูลสินค้า**

ฟิลด์ทั้ง 5 นี้ทำให้ระบบจัดการข้อมูลหนังสือสามารถทำงานได้อย่างมีประสิทธิภาพ โดยช่วยให้การจัดเก็บ การค้นหา การกรองข้อมูล และการแสดงผลข้อมูลเป็นไปได้อย่างรวดเร็วและมีประสิทธิภาพ นอกจากนี้ยังช่วยให้ผู้ใช้สามารถเข้าใจและเข้าถึงข้อมูลหนังสือได้ง่ายขึ้น ซึ่งจะส่งผลดีต่อประสบการณ์ของผู้ใช้ในการค้นหาและเลือกซื้อหนังสือ

## ฟังก์ชันพื้นฐานในระบบจัดการข้อมูลร้านค้า

### โครงสร้างและการทำงานของฟังก์ชัน main()

ฟังก์ชัน main() เป็นฟังก์ชันหลักของโปรแกรมระบบจัดการข้อมูลร้านค้า ทำหน้าที่หลักในการแสดงเมนูการจัดการข้อมูลและรับคำสั่งจากผู้ใช้เพื่อนำไปเรียกใช้ฟังก์ชันย่อยต่างๆ ตามที่เลือกจากเมนู ฟังก์ชันนี้ทำงานวนซ้ำในลูปจนกว่าผู้ใช้จะเลือกออกจากโปรแกรม โดยรายละเอียดของฟังก์ชันมีดังนี้:

#### โครงสร้างและการทำงานของฟังก์ชัน main()

แสดงเมนูหลัก: เมื่อโปรแกรมเริ่มต้น ฟังก์ชัน main() จะทำการแสดงเมนูหลักที่มีตัวเลือกต่างๆ ให้กับผู้ใช้ เช่น:

กด 1 เพื่อแสดงรายการสินค้า

กด 2 เพื่อเพิ่มสินค้าใหม่

กด 3 เพื่ออัปเดตข้อมูลสินค้า

กด 4 เพื่อค้นหาสินค้า

กด 5 เพื่อลบสินค้า

กด 6 เพื่อสร้างรายงาน

กด 7 เพื่อออกจากโปรแกรม

**การรับค่าการเลือกจากผู้ใช้**

ฟังก์ชัน input() จะถูกใช้ในการรับค่าจากผู้ใช้ โดยผู้ใช้ต้องใส่หมายเลขที่สอดคล้องกับฟังก์ชันที่ต้องการทำงาน

**การเรียกใช้ฟังก์ชันต่างๆ** **ตามการเลือกของผู้ใช้**

เมื่อผู้ใช้ใส่หมายเลขเลือกแล้ว โปรแกรมจะตรวจสอบค่าและเรียกใช้ฟังก์ชันที่สอดคล้องตามการเลือกนั้น หากผู้ใช้เลือกหมายเลขที่ไม่ถูกต้อง จะมีการแจ้งเตือนให้ผู้ใช้ทราบและขอให้ใส่หมายเลขใหม่

**การทำงานต่อเนื่อง**

หลังจากการทำงานของฟังก์ชันที่ถูกเลือกเสร็จสิ้น ฟังก์ชัน main() จะถูกเรียกซ้ำเพื่อแสดงเมนูอีกครั้ง ทำให้โปรแกรมทำงานต่อเนื่องจนกว่าผู้ใช้จะเลือกออกจากโปรแกรม

#### **โค้ดของฟังก์ชัน main()**

A screen shot of a computer program

Description automatically generated  
ภาพที่ 2‑1 โค้ดของฟังก์ชัน main()A screen shot of a computer program

Description automatically generated  
A computer screen with text

Description automatically generated

ภาพที่ 2‑2 โค้ดของฟังก์ชัน main() (2)

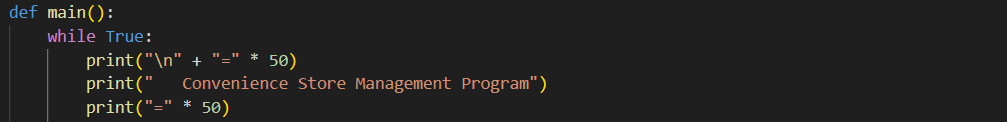
#### **แสดงหัวข้อของโปรแกรม**

เมื่อโปรแกรมเริ่มทำงาน ฟังก์ชัน main() จะพิมพ์หัวข้อและเมนูหลักของโปรแกรมด้วยสัญลักษณ์เครื่องหมายเท่ากับ (=) จำนวน 50 ตัว เพื่อสร้างความสวยงามในการแสดงผล จากนั้นจะแสดงชื่อโปรแกรมและเมนูการจัดการข้อมูล โดยการพิมพ์ข้อความถูกพิมพ์ด้วยคำสั่ง print()

**ชื่อโปรแกรม**: "Convenience Store Management"

**เส้นคั่น**: ใช้สัญลักษณ์ "=" เพื่อสร้างเส้นคั่น

**เมนูการจัดการ**: จะแสดงตัวเลือกต่าง ๆ ให้กับผู้ใช้



ภาพที่ 2‑3 แสดงหัวข้อของฟังก์ชัน main()

#### **แสดงเมนูให้ผู้ใช้เลือก**

โปรแกรมจะเข้าสู่ลูป while True เพื่อแสดงเมนูซ้ำ ๆ และรับอินพุตจากผู้ใช้เรื่อย ๆจนกว่าจะเลือกเมนู "Exit" เพื่อออกจากโปรแกรม โดยลูป while True เป็นลูปที่ไม่เงื่อนไขสิ้นสุด ทําให้โปรแกรมทํางานต่อเนื่องไม่หยุด และรับอินพุตจากผู้ใช้ใช้คําสั่ง input() เพื่อให้ผู้ใช้เลือกหมายเลขเมนู จากนั้นแปลงข้อมูลที่ได้รับเป็นประเภทตัวเลขจํานวนเต็ม (int)

เมนูที่ผู้ใช้สามารถเลือกได้

Add new data: เพิ่มข้อมูลสินค้าใหม่

Display all data: แสดงข้อมูลสินค้าทั้งหมด

Retrieve specific data: ค้นหาสินค้าจากชื่อหรือรหัสสินค้า

Update data: แก้ไขข้อมูลสินค้าที่มีอยู่

Delete data: ลบข้อมูลสินค้า

Create data: สร้างรายงานของสินค้า

Exit program: ยืนยันการออกจากโปรแกรม

A screen shot of a computer

Description automatically generated

ภาพที่ 2‑4 แสดงเมนูให้ผู้ใช้เลือกของฟังก์ชัน main()

#### **ตรวจสอบการเลือกของผู้ใช้**

เมื่อผู้ใช้เลือกหมายเลขจากเมนู ระบบจะตรวจสอบหมายเลขที่ผู้ใช้เลือก และเรียกใช้ฟังก์ชันที่เกี่ยวข้องโดยใช้โครงสร้างควบคุม if-elif-else ดังนี้:

ฟังก์ชันที่ถูกเรียกใช้:

add\_record(): เรียกใช้เพื่อเพิ่มข้อมูลใหม่ลงในไฟล์.

display\_records(): เรียกใช้เพื่อแสดงข้อมูลทั้งหมดในไฟล์.

retrieve\_records(): เรียกใช้เพื่อค้นหาข้อมูลเฉพาะจากชื่อหรือรหัส.

update\_record(): เรียกใช้เพื่ออัปเดตข้อมูลที่มีอยู่.

delete\_record(): เรียกใช้เพื่อลบข้อมูลที่เลือก.

create\_report(): เรียกใช้เพื่อสร้างรายงานจากข้อมูลทั้งหมด.

exit\_program(): เรียกใช้เพื่อยืนยันการออกจากโปรแกรม.

หากผู้ใช้ป้อนหมายเลขที่ไม่อยู่ในช่วง 1 ถึง 7 โปรแกรมจะแสดงข้อความเตือนว่า "Invalid options" เพื่อแจ้งให้ผู้ใช้ทราบว่าเลขที่เลือกไม่ถูกต้อง

A black rectangle with white text

Description automatically generated

ภาพที่ 2‑5 การเลือกของผู้ใช้ฟังก์ชันหมายเลขที่ไม่อยู่ในช่วง 1-7 main()

#### **การวนลูปกลับสู่เมนูหลัก**

เนื่องจากลูป while True: จะทํางานต่อเนื่อง โปรแกรมจึงจะแสดงเมนูให้ผู้ใช้เลือกใหม่เรื่อย ๆ หลังจากทํางานเสร็จในแต่ละฟังก์ชัน เช่น เมื่อผู้ใช้แสดงรายการหนังสือหรือเพิ่มหนังสือเสร็จแล้ว โปรแกรมจะกลับมาที่เมนูหลักโดยอัตโนมัติ จนกว่าผู้ใช้จะเลือกเมนู "Exit program" (7) เพื่อออกจากโปรแกรม

#### **การออกจากโปรแกรม**

เมื่อผู้ใช้เลือกเมนู 7 โปรแกรมจะทำการ break main() และโปรแกรมหยุดทํางาน



ภาพที่ 2‑6 การออกจากโปรแกรมฟังก์ชัน main()

#### **สรุปการทํางานของฟังก์ชัน main()**

ฟังก์ชัน main() เป็นศูนย์กลางการทํางานของโปรแกรมทําหน้าที่แสดงเมนูและรับคําสั่งจากผู้ใช้มีการตรวจสอบอินพุตจากผู้ใช้และเรียกใช้ฟังก์ชันย่อยต่าง ๆ ตามเมนูที่เลือกและโปรแกรมจะวนลูปไปเรื่อย ๆ จนกว่าผู้ใช้จะเลือกออกจากโปรแกรม

#### **ผลลัพธ์ของฟังก์ชัน main()**

A screen shot of a computer

Description automatically generated

ภาพที่ 2‑7 ผลลัพธ์ของฟังก์ชัน main()

### **โครงสร้างและการทำงานของฟังก์ชัน add\_record()**

ฟังก์ชัน add\_record() ทำหน้าที่ในการเพิ่มข้อมูลหนังสือใหม่ลงในไฟล์ data.bin โดยข้อมูลที่ถูกบันทึกจะมีรหัสหนังสือ ชื่อหนังสือ ราคา หมวดหมู่ และสถานะสต็อก

#### **การทำงานของฟังก์ชัน**

ฟังก์ชันนี้อนุญาตให้ผู้ใช้เพิ่มข้อมูลผลิตภัณฑ์ใหม่ลงในระบบ โดยจะต้องป้อนค่าตามฟิลด์ที่กำหนดไว้ ได้แก่ record\_id, name, price, category, และ stock\_status ซึ่งมีรายละเอียดการทำงานดังนี้

**รับข้อมูลจากผู้ใช้**

ฟังก์ชัน add\_record(record\_id, name, price, category, stock\_status) ถูกเรียกใช้เมื่อผู้ใช้ต้องการเพิ่มข้อมูลผลิตภัณฑ์ใหม่ โดยผู้ใช้จะต้องป้อนข้อมูลที่จำเป็นตามฟิลด์ที่กำหนด ซึ่งประกอบด้วย:

record\_id: รหัสผลิตภัณฑ์ (ID)

name: ชื่อผลิตภัณฑ์

price: ราคาของผลิตภัณฑ์

category: หมวดหมู่ของผลิตภัณฑ์

stock\_status: สถานะของสต็อก (In Stock / Out Stock)

A screen shot of a computer code

Description automatically generated

ภาพที่ 2‑8 การใส่ข้อมูลสินค้าใหม่ main()

#### **เปิดไฟล์ในโหมดเพิ่มข้อมูล**

ฟังก์ชันจะเปิดไฟล์ data.bin ในโหมด 'ab' (append binary) เพื่อให้สามารถเขียนข้อมูลใหม่ลงในไฟล์โดยไม่ทำให้ข้อมูลเดิมที่มีอยู่ในไฟล์หายไป



ภาพที่ 2‑9 การเปิดไฟล์ในโหมด 'ab' add\_record()

#### **บีบอัดข้อมูล**

ข้อมูลที่ได้รับจากผู้ใช้จะถูกบีบอัดในรูปแบบไบนารี โดยใช้ฟังก์ชัน struct.pack() ซึ่งจะจัดเก็บข้อมูลตามรูปแบบที่กำหนดในตัวแปร RECORD\_FORMAT ซึ่งประกอบด้วย:

I: รหัสผลิตภัณฑ์เป็นจำนวนเต็ม

20s: ชื่อผลิตภัณฑ์ (สูงสุด 20 ตัวอักษร)

f: ราคาของผลิตภัณฑ์เป็นจำนวนทศนิยม

20s: หมวดหมู่ของผลิตภัณฑ์ (สูงสุด 20 ตัวอักษร)

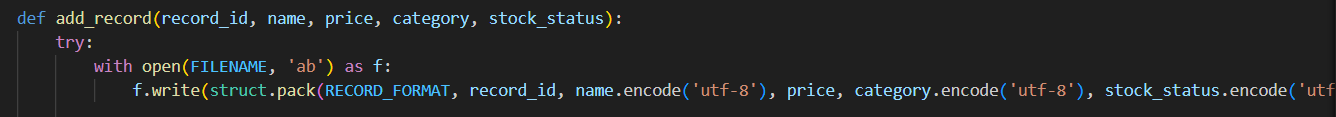
10s: สถานะของสต็อก (สูงสุด 10 ตัวอักษร)



ภาพที่ 2‑10 การบีบอัดข้อมูลในรูปแบบ binary

#### **บันทึกข้อมูลลงในไฟล์**

ข้อมูลที่ถูกบีบอัดจะถูกเขียนลงในไฟล์ data.bin ซึ่งจะทำให้ข้อมูลผลิตภัณฑ์ใหม่ถูกเพิ่มเข้าไปในไฟล์ทันที



ภาพที่ 2‑11 ข้อมูลที่ใส่บีบอัดลงไฟล์ add\_record()

#### **การแสดงผลของฟังก์ชั่น add\_record()**

เมื่อข้อมูลถูกบันทึกลงในไฟล์เรียบร้อยแล้ว จะมีการแสดงข้อความยืนยันความสำเร็จและความผิดพลาด พร้อมกับรายละเอียดของสินค้าที่ถูกเพิ่มลงในระบบ

A black background with colorful text

Description automatically generated

#### **ผลลัพธ์ของฟังก์ชั่น add\_record()**

A screenshot of a computer program

Description automatically generated

ภาพที่ 2‑12 ผลลัพธ์ของฟังก์ชั่น add\_record()

### **โครงสร้างและการทำงานของฟังก์ชัน display\_records()**

ฟังก์ชันนี้ทำหน้าที่ในการแสดงข้อมูลผลิตภัณฑ์ทั้งหมดที่มีอยู่ในไฟล์ data.bin โดยการอ่านข้อมูลจากไฟล์ในรูปแบบไบนารีและแสดงผลในรูปแบบที่เข้าใจง่าย ฟังก์ชันนี้มีรายละเอียดการทำงานดังนี้

#### **ตรวจสอบการมีอยู่ของไฟล์**

ฟังก์ชันเริ่มต้นด้วยการตรวจสอบว่าไฟล์ data.bin มีอยู่ในระบบหรือไม่ หากไฟล์ไม่พบ ฟังก์ชันจะแสดงข้อความว่า "Data file not found" และหยุดการทำงาน



ภาพที่ 2‑13 การตรวจสอบไฟล์ระบบ display\_records()

#### **แสดงหัวข้อข้อมูล**

หากไฟล์พบ ฟังก์ชันจะทำการพิมพ์เส้นขอบด้านบนและหัวข้อเพื่อเตรียมแสดงข้อมูล



ภาพที่ 2‑14 เตรียมแสดงข้อมูลหากพบไฟล์ display\_record()

#### **เปิดไฟล์ในโหมดอ่านข้อมูล**

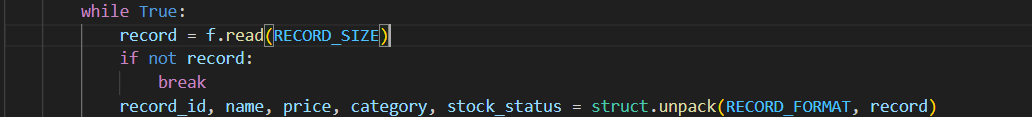
ฟังก์ชันจะเปิดไฟล์ data.bin ในโหมด 'rb' (read binary) เพื่อเริ่มการอ่านข้อมูลจากไฟล์



ภาพที่ 2‑15 เปิดไฟล์ data.bin ในโหมด 'rb' display\_record()

#### **อ่านข้อมูลจากไฟล์**

ฟังก์ชันจะใช้ลูป while เพื่ออ่านข้อมูลจากไฟล์เป็นชุด ๆ (records) ขนาดของแต่ละชุดจะถูกกำหนดโดย RECORD\_SIZE ซึ่งคำนวณจากโครงสร้างข้อมูล



ภาพที่ 2‑16 ใช้ลูป while เพื่ออ่านข้อมูลจากไฟล์เป็นชุด display\_record()

#### **แยกข้อมูลและแสดงผล**

สำหรับข้อมูลที่อ่านเข้ามาแต่ละชุด ฟังก์ชันจะทำการแยกข้อมูลออกเป็นฟิลด์ต่าง ๆ เช่น record\_id, name, price, category, และ stock\_status โดยใช้ struct.unpack(RECORD\_FORMAT, record)



ภาพที่ 2‑17 การแยกข้อมูลออกเป็นฟิลด์ต่าง ๆ display\_record()

จากนั้น ฟังก์ชันจะแสดงข้อมูลแต่ละรายการในรูปแบบที่เป็นมิตรต่อผู้ใช้ โดยทำการแปลง name, category, และ stock\_status จากไบนารีเป็นข้อความ (string) และตัดช่องว่างที่ไม่จำเป็นออก



ภาพที่ 2‑18 แสดงข้อมูลแต่ละรายการ display\_record()

#### **แสดงเส้นขอบด้านล่าง**

หลังจากที่แสดงข้อมูลทั้งหมดแล้ว ฟังก์ชันจะพิมพ์เส้นขอบด้านล่างเพื่อให้รู้ว่าการแสดงข้อมูลสิ้นสุดลง



ภาพที่ 2‑19 พิมพ์เส้นขอบด้านล่าง display\_record()

#### **การทำงานโดยรวมของฟังก์ชัน display\_records()**

A computer code on a black background

Description automatically generated

ภาพที่ 2‑20 ทำงานโดยรวมของฟังก์ชัน display\_records()

#### **ผลลัพธ์ของฟังก์ชัน display\_record()**

A screenshot of a computer program

Description automatically generated

ภาพที่ 2‑21 ผลลัพธ์ของฟังก์ชัน display\_record()

### **โครงสร้างและการทำงานของฟังก์ชัน retrieve\_records()**

ฟังก์ชันนี้มีวัตถุประสงค์เพื่อค้นหาและแสดงบันทึกข้อมูลจากไฟล์ที่ตรงตามค่าที่ผู้ใช้ระบุ โดยรายละเอียดการทำงานมีดังนี้:

#### **ตรวจสอบการมีอยู่ของไฟล์**

ฟังก์ชันเริ่มต้นด้วยการตรวจสอบว่าไฟล์ data.bin มีอยู่ในระบบหรือไม่ โดยใช้คำสั่ง os.path.exists(FILENAME) ถ้าไฟล์ไม่พบ ฟังก์ชันจะแสดงข้อความ "Data file not found" และใช้ return เพื่อหยุดการทำงาน



ภาพที่ 2‑22 การตรวจสอบไฟล์ในระบบ retrieve\_records()

#### **ตัวแปรสำหรับตรวจสอบการพบข้อมูล**

ฟังก์ชันจะสร้างตัวแปร found และตั้งค่าเป็น False เพื่อใช้ในการตรวจสอบว่ามีการพบข้อมูลที่ตรงตามที่ค้นหาหรือไม่



ภาพที่ 2‑23 การตรวจสอบข้อมูลที่ตรง retrieve\_records()

#### **เปิดไฟล์ในโหมดอ่านข้อมูล**

ฟังก์ชันจะเปิดไฟล์ data.bin ในโหมด 'rb' (read binary) โดยใช้คำสั่ง with open(FILENAME, 'rb') as f:



ภาพที่ 2‑24 ฟังก์ชันจะเปิดไฟล์ในโหมด 'rb' retrieve\_records()

#### **อ่านข้อมูลจากไฟล์**

ฟังก์ชันจะใช้ลูป while เพื่ออ่านข้อมูลจากไฟล์เป็นชุด ๆ ขนาดของแต่ละชุดจะถูกกำหนดโดย RECORD\_SIZE



ภาพที่ 2‑25 ใช้ลูป while เพื่ออ่านข้อมูลจากไฟล์เป็นชุด ๆ retrieve\_records()

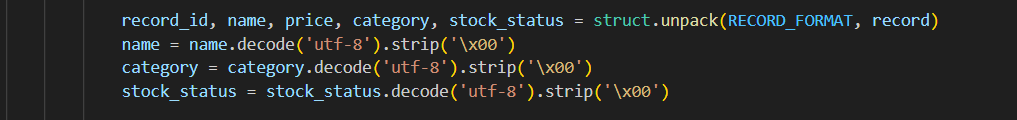
#### **แยกข้อมูลและตรวจสอบการจับคู่**

สำหรับข้อมูลที่อ่านเข้ามาแต่ละชุด ฟังก์ชันจะทำการแยกข้อมูลออกเป็นฟิลด์ต่าง ๆ เช่น record\_id, name, price, category, และ stock\_status โดยใช้ struct.unpack(RECORD\_FORMAT, record)



ภาพที่ 2‑26 การแยกข้อมูลออกเป็นฟิลด์ต่าง ๆ retrieve\_records()

ข้อมูล name, category, และ stock\_status จะถูกแปลงจากไบนารีเป็นข้อความ (string) โดยใช้ decode('utf-8') และลบช่องว่างที่ไม่จำเป็นออกด้วย strip('\x00'):



ภาพที่ 2‑27 แปลงจากไบนารีเป็นข้อความ retrieve\_records()

#### **ค้นหาข้อมูลที่ตรงกัน**

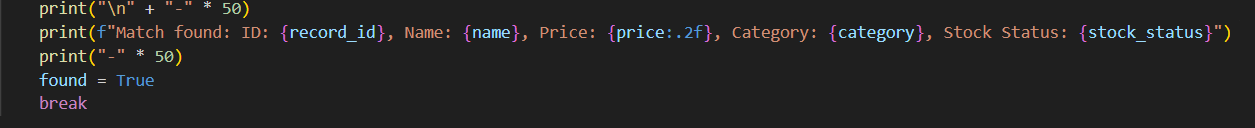
ฟังก์ชันจะทำการตรวจสอบว่าค่าที่ป้อนเข้ามา (search\_value) ตรงกับ record\_id หรือ name หรือไม่ โดยใช้คำสั่ง if str(record\_id) == search\_value or name.lower() == search\_value.lower()::



ภาพที่ 2‑28 การตรวจสอบว่าค่าที่ป้อนเข้ามาตรงกับ record\_id หรือ name

#### **แสดงผลเมื่อพบข้อมูลที่ตรงกัน**

หากพบข้อมูลที่ตรงกัน ฟังก์ชันจะแสดงผลบันทึกที่ตรงกันในรูปแบบที่เป็นมิตรต่อผู้ใช้ และตั้งค่า found เป็น True เพื่อบ่งบอกว่าพบข้อมูล:



ภาพที่ 2‑29 แสดงผลบันทึกที่ตรงกันในรูปแบบที่เป็นมิตรต่อผู้ใช้

#### **ตรวจสอบการไม่พบข้อมูล**

หลังจากลูปอ่านข้อมูลเสร็จสิ้น ฟังก์ชันจะตรวจสอบว่า found ยังคงเป็น False อยู่หรือไม่ หากใช่จะแสดงข้อความ "No matching records found." เพื่อบ่งบอกว่ามีข้อมูลที่ค้นหาไม่พบ



ภาพที่ 2‑30 แสดงข้อความว่ามีข้อมูลที่ค้นหาไม่พบ recrieve\_record()

#### **การทำงานโดยรวมของฟังก์ชัน retrieve\_records()**

A computer screen shot of text

Description automatically generated

ภาพที่ 2‑31 การทำงานโดยรวมของฟังก์ชัน retrieve\_records()

#### **ผลลัพธ์ของฟังก์ชัน retrieve\_record()**

A screenshot of a computer program

Description automatically generated

ภาพที่ 2‑32 ผลลัพธ์ของฟังก์ชัน retrieve\_record()

### **โครงสร้างและการทำงานของฟังก์ชัน update\_record()**

#### **รับพารามิเตอร์ที่ต้องการสำหรับการอัปเดต**

ฟังก์ชันนี้รับค่าพารามิเตอร์ 5 ค่า record\_id, new\_name, new\_price, new\_category, และ new\_stock\_status ซึ่งเป็นค่าข้อมูลที่ต้องการอัปเดตสำหรับบันทึกข้อมูลที่มี record\_id ตรงกัน

#### **ตัวแปรเก็บข้อมูลทั้งหมด**

ตัวแปร records ถูกใช้ในการเก็บข้อมูลทั้งหมดที่ถูกอ่านมาจากไฟล์เพื่อนำกลับไปเขียนใหม่ โดยจะเก็บทั้งบันทึกข้อมูลที่อัปเดตและบันทึกข้อมูลที่ไม่ต้องเปลี่ยนแปลง

**

ภาพที่ ‑ การเก็บข้อมูลทั้งหมดที่ถูกอ่านมาจากไฟล์

#### **ตัวแปรเพื่อตรวจสอบการอัปเดต**

ตัวแปร updated ถูกกำหนดค่าเริ่มต้นเป็น False เพื่อใช้ในการตรวจสอบว่ามีการอัปเดตบันทึกข้อมูลในไฟล์หรือไม่



ภาพที่ 2‑34 การตรวจสอบว่ามีการอัปเดต

#### **เปิดไฟล์เพื่ออ่านข้อมูล**

ฟังก์ชันจะเปิดไฟล์ในโหมด 'rb' (read binary) เพื่อทำการอ่านข้อมูลทั้งหมดจากไฟล์และนำไปเก็บในตัวแปร records



ภาพที่ 2‑35 เปิดไฟล์ในโหมด 'rb' update\_record()

#### **ลูปอ่านข้อมูล**

ข้อมูลจะถูกอ่านเป็นบันทึก ๆ โดยแต่ละบันทึกมีขนาด RECORD\_SIZE ฟังก์ชันจะใช้ struct.unpack(RECORD\_FORMAT, record) เพื่อแปลงข้อมูลออกเป็นฟิลด์ต่าง ๆ ได้แก่ rec\_id, name, price, category, และ stock\_status:

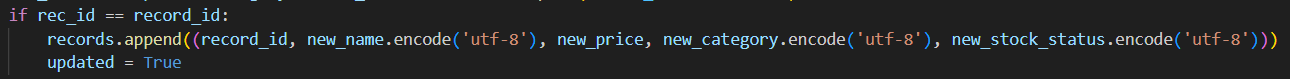
A black screen with white text

Description automatically generated

ภาพที่ 2‑36 ลูปอ่านข้อมูลเพื่อแปลงข้อมูลออกเป็นฟิลด์ updated\_record()

#### **ตรวจสอบและอัปเดตข้อมูล**

ถ้า rec\_id ของบันทึกตรงกับ record\_id ที่ต้องการอัปเดต ฟังก์ชันจะเพิ่มข้อมูลที่อัปเดตลงในรายการ records โดยใช้ค่าพารามิเตอร์ใหม่ที่ผู้ใช้ป้อนเข้ามา และเปลี่ยนค่า updated เป็น True



ภาพที่ 2‑37 ถ้า rec\_id ของบันทึกตรงกับ record\_id ที่ต้องการอัปเดต

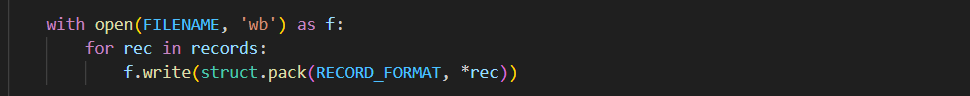
แต่ถ้า rec\_id ไม่ตรงกัน บันทึกข้อมูลเดิมจะถูกเพิ่มเข้าไปใน records โดยไม่ทำการเปลี่ยนแปลง



ภาพที่ 2‑38 แต่ถ้า rec\_id ไม่ตรงกัน

#### **เขียนข้อมูลที่อัปเดตกลับไปยังไฟล์:**

หลังจากอ่านข้อมูลเสร็จแล้ว ฟังก์ชันจะเปิดไฟล์อีกครั้งในโหมด 'wb' (write binary) เพื่อทำการเขียนข้อมูลทั้งหมดใน records กลับไปยังไฟล์



ภาพที่ 2‑39 เปิดไฟล์อีกครั้งในโหมด 'wb' update\_record()

#### **แสดงผลการอัปเดต**

หากมีการอัปเดตข้อมูล (updated เป็น True) ฟังก์ชันจะแสดงข้อความว่า "Data has been updated!" มิฉะนั้นจะแสดงข้อความว่า "No matching record found." เพื่อบอกว่าข้อมูลที่ต้องการอัปเดตไม่พบ

A black rectangular object with a black line

Description automatically generated

ภาพที่ 2‑40 หากมีการอัปเดตข้อมูลฟังก์ชันจะแสดงข้อความ update\_record()

#### **การทำงานโดยรวมของฟังก์ชัน update\_record()**

1. ฟังก์ชันจะเปิดไฟล์และอ่านข้อมูลทั้งหมดเก็บไว้ในรายการ records

2. ฟังก์ชันตรวจสอบว่า record\_id ที่ต้องการอัปเดตตรงกับข้อมูลใดในไฟล์หรือไม่ หากตรงจะอัปเดตข้อมูลในบันทึกนั้นด้วยค่าพารามิเตอร์ใหม่ที่ส่งมา

3. ข้อมูลทั้งหมดถูกเขียนกลับลงในไฟล์อีกครั้ง

4. แสดงผลลัพธ์ว่ามีการอัปเดตสำเร็จหรือไม่

A screen shot of a computer code

Description automatically generated

ภาพที่ 2‑41 การทำงานโดยรวมของฟังก์ชัน update\_record()

#### **ผลลัพธ์ของฟังก์ชัน display\_record()**

A screenshot of a computer program

Description automatically generated

ภาพที่ 2‑42 ผลลัพธ์ของฟังก์ชัน display\_record()

### **โครงสร้างและการทำงานของฟังก์ชัน delete\_record()**

#### **ประกาศตัวแปรสำหรับเก็บข้อมูลและสถานะการลบ**

ฟังก์ชันเริ่มต้นด้วยการประกาศตัวแปร records เพื่อเก็บข้อมูลทั้งหมดที่ถูกอ่านมาจากไฟล์ และตัวแปร deleted ที่มีค่าเริ่มต้นเป็น False เพื่อใช้ตรวจสอบว่ามีการลบข้อมูลหรือไม่

A black background with a blue and white object

Description automatically generated with medium confidence

ภาพที่ 2‑43 ประกาศตัวแปรสำหรับเก็บข้อมูลและสถานการณ์ลบ

#### **เปิดไฟล์ในโหมดอ่าน**

ฟังก์ชันจะเปิดไฟล์ไบนารีในโหมดอ่าน ('rb') เพื่ออ่านข้อมูลทั้งหมดจากไฟล์



ภาพที่ 2‑44 เปิดไฟล์ไบนารีในโหมดอ่าน delete\_record()

#### **ลูปอ่านข้อมูลจากไฟล์**

ฟังก์ชันจะใช้ลูปในการอ่านข้อมูลทีละบันทึก ขนาดของแต่ละบันทึกถูกกำหนดโดย RECORD\_SIZE และข้อมูลในบันทึกจะถูกถอดรหัสโดยใช้ struct.unpack(RECORD\_FORMAT, record) เพื่อแยกข้อมูลเป็นฟิลด์ต่าง ๆ ได้แก่ rec\_id, name, price, category, และ stock\_status

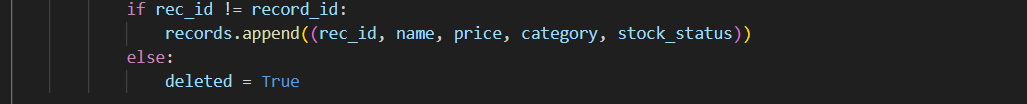
A black screen with white text

Description automatically generated

ภาพที่ 2‑45 ใช้ลูปในการอ่านข้อมูลทีละบันทึก delelte\_record()

#### **ตรวจสอบและลบข้อมูล**

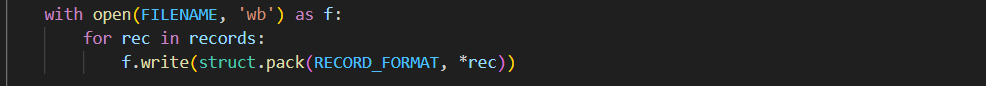
หาก rec\_id ของบันทึกตรงกับ record\_id ที่ผู้ใช้ต้องการลบ ฟังก์ชันจะไม่เพิ่มบันทึกนั้นลงในรายการ records และตั้งค่า deleted เป็น True เพื่อระบุว่าพบและลบข้อมูลแล้ว



ภาพที่ 2‑46 ตรวจสอบและลบข้อมูล delete\_record()

#### **เขียนข้อมูลที่เหลือกลับลงในไฟล์**

หลังจากอ่านข้อมูลทั้งหมดแล้ว ฟังก์ชันจะเปิดไฟล์ในโหมดเขียน ('wb') เพื่อเขียนข้อมูลทั้งหมดที่ไม่ถูกลบลงในไฟล์ใหม่



ภาพที่ 2‑47 เปิดไฟล์ในโหมดเขียน ('wb') delete\_record()

#### **แสดงผลการลบข้อมูล**

หากมีการลบข้อมูล (ค่า deleted เป็น True) ฟังก์ชันจะแสดงข้อความว่า "Data has been deleted!" แต่หากไม่มีบันทึกที่ตรงกับ record\_id จะมีการแสดงข้อความว่า "No matching record found." เพื่อระบุว่าไม่พบข้อมูลที่ต้องการลบ

A black rectangle with white text

Description automatically generated

ภาพที่ 2‑48 แสดงผลการลบข้อมูล delete\_record()

#### **การทำงานโดยรวมของฟังก์ชัน delete\_record()**

1. ฟังก์ชันจะเปิดไฟล์และอ่านข้อมูลทั้งหมด

2. ตรวจสอบว่าบันทึกข้อมูลที่อ่านมาตรงกับ record\_id หรือไม่ หากตรง ข้อมูลจะไม่ถูกเพิ่มไปในรายการ records ทำให้ข้อมูลนั้นถูกลบออกจากไฟล์

3. ข้อมูลที่เหลือจะถูกเขียนกลับลงในไฟล์

4. ฟังก์ชันแสดงข้อความเพื่อบอกผลลัพธ์ว่าข้อมูลถูกลบหรือไม่

A screen shot of a computer program

Description automatically generated

ภาพที่ 2‑49 การทำงานโดยรวมของฟังก์ชัน delete\_record()

#### **ผลลัพธ์การทำงานของฟังก์ชัน delete\_record()**

A screenshot of a computer

Description automatically generated

ภาพที่ 2‑50 ผลลัพธ์การทำงานของฟังก์ชัน delete\_record()

### **โครงสร้างและการทำงานของฟังก์ชัน create\_report()**

#### **ตรวจสอบการมีอยู่ของไฟล์**

ฟังก์ชันเริ่มต้นด้วยการตรวจสอบว่าไฟล์ข้อมูลมีอยู่หรือไม่ ถ้าไม่มีไฟล์อยู่ ฟังก์ชันจะพิมพ์ข้อความแจ้งเตือนและหยุดการทำงาน



ภาพที่ 2‑51 ตรวจสอบการมีอยู่ของไฟล์ create\_report()

#### **สร้างตัวแปรสำหรับรายงาน**

ประกาศตัวแปร report\_lines เพื่อใช้เก็บข้อมูลจากไฟล์เป็นบรรทัดต่าง ๆ ซึ่งจะถูกใช้ในการเขียนลงไฟล์ report.txt ภายหลัง



ภาพที่ 2‑52 สร้างตัวแปรสำหรับรายงาน create\_report()

#### **เปิดไฟล์ในโหมดอ่าน**

ฟังก์ชันจะเปิดไฟล์ไบนารีในโหมดอ่าน ('rb') เพื่ออ่านข้อมูลทีละบันทึก โดยใช้ลูปเพื่ออ่านบันทึกทั้งหมดในไฟล์



ภาพที่ 2‑53 เปิดไฟล์ในโหมดอ่าน create\_report()

#### **อ่านและจัดรูปแบบข้อมูล**

ข้อมูลที่อ่านมาจากไฟล์จะถูกจัดรูปแบบเป็นบรรทัดที่มีฟอร์แมตอ่านง่าย โดยใช้ struct.unpack() เพื่อถอดข้อมูลเป็นฟิลด์ต่าง ๆ เช่น record\_id, name, price, category, และ stock\_status ข้อมูลเหล่านี้จะถูกแปลงเป็นข้อความและถูกเพิ่มลงใน report\_lines

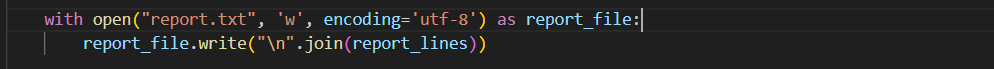
A screen shot of a computer code

Description automatically generated

ภาพที่ 2‑54 อ่านและจัดรูปแบบข้อมูล create\_report()

#### **เขียนรายงานลงในไฟล์**

ฟังก์ชันจะเปิดไฟล์ report.txt ในโหมดเขียน ('w') และเขียนบรรทัดทั้งหมดที่ถูกเก็บไว้ใน report\_lines ลงในไฟล์ โดยใช้การเชื่อมบรรทัดด้วย \n



ภาพที่ 2‑55 เขียนรายงานลงในไฟล์ create\_report()

#### **แสดงผลลัพธ์การสร้างรายงาน**

หลังจากสร้างรายงานเสร็จ ฟังก์ชันจะพิมพ์ข้อความแสดงความสำเร็จว่ารายงานได้ถูกบันทึกลงในไฟล์ report.txt แล้ว



ภาพที่ 2‑56 แสดงผลลัพธ์การสร้างรายงาน create\_report()

#### **การทำงานโดยรวมของฟังก์ชัน create\_report()**

1. ตรวจสอบว่ามีไฟล์ข้อมูลอยู่หรือไม่ หากไม่พบไฟล์จะแสดงข้อความและหยุดการทำงาน

2. เปิดไฟล์ข้อมูลไบนารีเพื่ออ่านข้อมูลแต่ละบันทึก

3. จัดรูปแบบข้อมูลให้อยู่ในรูปแบบบรรทัดที่อ่านง่ายและเพิ่มลงในรายการ report\_lines

4. บันทึกรายงานที่ประกอบด้วยข้อมูลทั้งหมดลงในไฟล์ report.txt

5. แสดงผลลัพธ์ให้ผู้ใช้ทราบว่ารายงานถูกบันทึกสำเร็จ

A screen shot of a computer code

Description automatically generated

ภาพที่ 2‑57 การทำงานโดยรวมของฟังก์ชัน create\_report()

#### **ผลลัพธ์ของฟังก์ชัน create\_report()**

ฟังก์ชันจะสร้างรายงานในรูปแบบข้อความที่อ่านง่าย โดยแต่ละรายการจะถูกแสดงในบรรทัดเดียว และข้อมูลแต่ละฟิลด์จะถูกจัดเรียงอย่างชัดเจน

A screen shot of a computer screen

Description automatically generated

ภาพที่ 2‑58 ผลลัพธ์ของฟังก์ชัน create\_report()