

การจัดการคลังสินค้า

Warehouse management

นายวิทวัส แก้ววิเศษ รหัส 6706022511014

นายอนันต์ยศ สายวงษ์ รหัส 6706022510379

นายธีรยุทธ ปาแก้ว รหัส 6706022510395

นางสาวพิยดา ช้างน้อย รหัส 6706022510361

โครงการนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต  
สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ  
คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ  
ปีการศึกษา 2567  
ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

## สารบัญ

สารบัญ	๖
สารบัญภาพ	๗
บทที่ 1 บทนำ	1
1.1 วัตถุประสงค์ของโครงการ	1
1.2. ขอบเขตของโครงการ	1
1.3. ประโยชน์ที่ได้รับ	1
1.4. เครื่องมือที่คาดว่าจะต้องใช้	1
บทที่ 2 ระบบการจัดการคลังสินค้า	2
2.1. ฟลัดในระบบจัดการคลังสินค้า	2
2.2 สรุปฟลัดในระบบจัดการคลังสินค้า	2
บทที่3 การใช้โปรแกรม การจัดการคลังสินค้า	3
3.1 สำหรับผู้ใช้งานโปรแกรม	3
3.2 การใช้โปรแกรม การแสดงข้อมูลทั้งหมด	4
3.3 การใช้โปรแกรม การค้นหาข้อมูล	5
3.4 การใช้โปรแกรม การอัปเดตข้อมูล	6
3.5. การใช้โปรแกรม การลบข้อมูล	7
3.6 การใช้โปรแกรม สร้างรายงาน	8
3.7 การใช้โปรแกรม ออกจากโปรแกรม	9
บทที่4 อธิบายการทำงานของ code	10
การกำหนดรูปแบบข้อมูล	10
4.1 ฟังก์ชันสำหรับเพิ่มข้อมูล	11
4.2 ฟังก์ชันสำหรับแสดงข้อมูลทั้งหมด	11
4.3 ฟังก์ชันสำหรับค้นหาข้อมูลตามรหัสสินค้า	12
4.4 ฟังก์ชันสำหรับอัปเดตข้อมูลสินค้า	13
4.5 ฟังก์ชันสำหรับลบข้อมูลสินค้า	14
4.6 ฟังก์ชันสำหรับเขียนรายงาน	15
4.7 จัดเก็บข้อมูลสินค้า	17
4.8 เมนูสำหรับเลือกฟังก์ชัน	18
4.9 เพิ่มข้อมูลสินค้า	19

## สารบัญภาพ

รูปที่ 3.1.1 การเพิ่มข้อมูล	3
รูปที่ 3.1.2 การเลือกฟังก์ชัน	4
รูปที่ 3.2.1 การแสดงข้อมูลทั้งหมด	4
รูปที่ 3.2.2 แสดงข้อมูลสินค้าทั้งหมด	4
รูปที่ 3.3.1 การค้นหาข้อมูล	5
รูปที่ 3.3.2 การค้นหาข้อมูล	5
รูปที่ 3.4.1 การอัปเดตข้อมูล	6
รูปที่ 3.4.2 การอัปเดตข้อมูล	6
รูปที่ 3.5.1 การลบข้อมูล	7
รูปที่ 3.5.2 การลบข้อมูลรายการสินค้า	7
รูปที่ 3.6.1 การสร้างรายงาน	8
รูปที่ 3.6.2 รายงานแยกประเภทสินค้า	8
รูปที่ 3.7.1 จบโปรแกรม	9
รูปที่ 4 การกำหนดรูปแบบข้อมูล	10
รูปที่ 4.1 ฟังก์ชันสำหรับเพิ่มข้อมูล	11
รูปที่ 4.2 ฟังก์ชันสำหรับแสดงข้อมูลทั้งหมด	12
รูปที่ 4.3 ฟังก์ชันสำหรับค้นหาข้อมูลตามรหัสสินค้า	13
รูปที่ 4.4 ฟังก์ชันสำหรับอัปเดตข้อมูลสินค้า	14
รูปที่ 4.5 ฟังก์ชันสำหรับลบข้อมูลสินค้า	15
รูปที่ 4.6.1 ขั้นตอนการทำงานของฟังก์ชัน	15
รูปที่ 4.6.2 เปิดไฟล์สินค้าในโหมดอ่านไบนารี	16
รูปที่ 4.6.3 แยกข้อมูลแต่ละรายการออกมา	16
รูปที่ 4.7 จัดเก็บข้อมูลสินค้า	17
รูปที่ 4.7.1 สร้างเนื้อหารายงาน	18
รูปที่ 4.7.2 บันทึกรายงานลงไฟล์ report.txt	18
รูปที่ 4.8 เมนูสำหรับเลือกฟังก์ชัน	19
รูปที่ 4.9 เพิ่มข้อมูลสินค้า	20

## คำนำ

การจัดทำโครงการในชื่อเรื่อง “การจัดการคลังสินค้า” นี้เป็นส่วนหนึ่งของวิชา COMPUTER PROGRAMMING ของหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศคณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ เพื่อให้นักศึกษาได้นำความรู้ที่เรียนมาทั้งหมดมาประยุกต์ใช้ในการพัฒนาโปรแกรมที่สามารถทำงานได้จริง โดยเน้นการออกแบบและเขียนโปรแกรมในภาษา Python ซึ่งเป็นภาษาที่เรียนมาในวิชา COMPUTER PROGRAMMING โดยโครงการนี้จะช่วย การคิดวิเคราะห์ และแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมในการประกอบอาชีพด้านวิศวกรรมสารสนเทศและเครือข่ายในอนาคต หากมีข้อผิดพลาดประการใด คณะผู้จัดทำต้องขออภัยไว้ ณ ที่นี้ด้วย

## บทที่ 1

### บทนำ

#### 1.1 วัตถุประสงค์ของโครงการ

- 1.1.1 เพื่อพัฒนาระบบที่สามารถจัดการหนังสือได้อย่างมีประสิทธิภาพ
- 1.1.2 เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วย Python
- 1.1.3 เพื่อเรียนรู้การจัดการข้อมูลและไฟล์
- 1.1.4 เพื่อเรียนรู้การทำงานร่วมกันเป็นทีม

#### 1.2. ขอบเขตของโครงการ

1.2.1ระบบจัดการข้อมูลหนังสือจะมีฟังก์ชันพื้นฐาน 7 ฟังก์ชัน 1.เพิ่มข้อมูล 2. แสดงข้อมูลทั้งหมด 3.ค้นหาข้อมูล 4.อัปเดตข้อมูล 5.ลบข้อมูล 6.สร้างรายงาน 7.ออกจากโปรแกรม

1.2.2ระบบการจัดเรียงสินค้าเก็บข้อมูลสินค้าไว้ text file ชื่อ report.txt ซึ่งมีจำนวนประเภทสินค้าทั้งหมด ประเภทสินค้า รายการสินค้าทั้งหมด เลขIDสินค้า ชื่อสินค้า ราคาสินค้า จำนวนสินค้า ผลรวมราคาสินค้า ผลรวมราคาสินค้า

#### 1.3. ประโยชน์ที่ได้รับ

- 1.3.1 พัฒนาระบบที่สามารถจัดการหนังสือได้อย่างมีประสิทธิภาพ
- 1.3.2 พัฒนาทักษะการเขียนโปรแกรม
- 1.3.3 เรียนรู้การจัดการข้อมูลและไฟล์
- 1.3.4 เรียนรู้การทำงานร่วมกันเป็นทีม

#### 1.4. เครื่องมือที่คาดว่าจะต้องใช้

- 1.4.1 ภาษา Python
- 1.4.2 Microsoft office

## บทที่ 2

### ระบบการจัดการคลังสินค้า

#### 2.1. ฟังก์ชันในระบบการจัดการคลังสินค้า

การจัดการข้อมูลหนังสือในระบบของคุณประกอบด้วย 7 ฟังก์ชันหลัก ซึ่งแต่ละฟังก์ชันมีรายละเอียดและความสำคัญดังนี้

##### 2.1.1 เพิ่มข้อมูล

การเพิ่มข้อมูล คือ การเพิ่มข้อมูลสินค้าเข้าไปใหม่ในคลังสินค้า

##### 2.1.2 แสดงข้อมูลทั้งหมด

การแสดงผลข้อมูลทั้งหมดคือการแสดงผลสินค้าทั้งหมดไม่ว่าจะเป็นราคาหรือจำนวนสินค้า ชนิดของสินค้าต่างๆ รายชื่อสินค้า เลขIDสินค้า

##### 2.1.3 ค้นหาข้อมูล

ค้นหาคือการค้นหาข้อมูลที่มีอยู่ในคลังสินค้า

##### 2.1.4 อัปเดตข้อมูล

การอัปเดตข้อมูลคือการเพิ่มข้อมูลสินค้าที่เราต้องการจะเพิ่มเข้าไปไม่ว่าจะเป็นจำนวนสินค้าหรือราคาสินค้านั้นๆในช่วงเวลานั้นหรือชื่อสินค้า

##### 2.1.5. ลบข้อมูล

การลบข้อมูลคือการลบข้อมูลที่เราไม่ต้องการออกไปเนื่องจากเช่นในสถานการณ์สินค้าหมด

##### 2.1.6. สร้างรายงาน

การสร้างรายงานคือการรายงานสินค้าจะมีราคารวมของสินค้าและบอกราคาแต่ละชนิดอย่างชัดเจน

##### 2.1.7 ออกจากโปรแกรม

การจบการทำงานของโปรแกรม

#### 2.2 สรุปฟังก์ชันในระบบการจัดการคลังสินค้า

ฟังก์ชันทั้ง 7 นี้ทำให้ระบบจัดการข้อมูลหนังสือสามารถทำงานได้อย่างมีประสิทธิภาพ โดยช่วยให้การจัดเก็บ การค้นหา การกรองข้อมูล และการแสดงผลข้อมูลเป็นไปได้อย่างรวดเร็วและมีประสิทธิภาพ นอกจากนี้ยังช่วยให้ผู้ใช้สามารถเข้าใจและเข้าถึงข้อมูลการจัดการคลังสินค้าได้ง่าย

## บทที่ 3

### การใช้โปรแกรม การจัดการคลังสินค้า

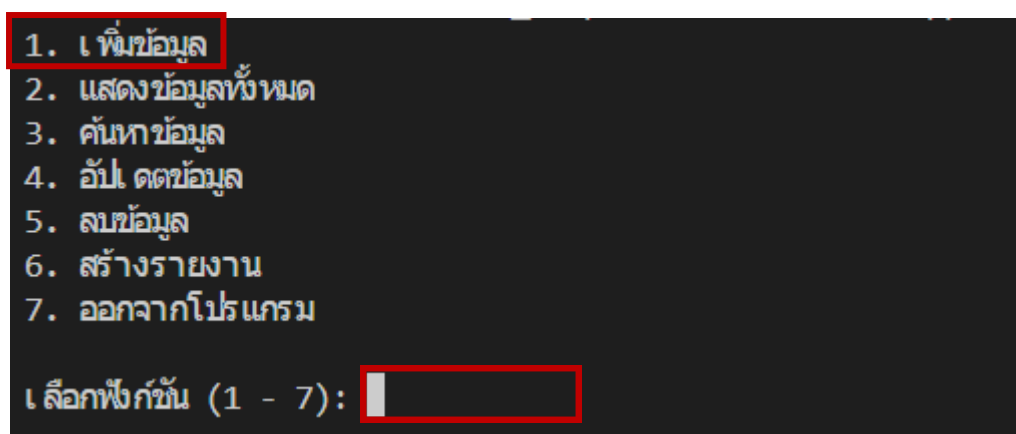
โปรแกรมการจัดการสินค้าคือการช่วยจำแนกประเภทสินค้าให้ดูสะดวกง่ายขึ้นและยังช่วยจัดประเภทของสินค้าให้ดูง่ายขึ้นโดยช่วยทำรายงานไปเก็บไว้ยังไฟล์ text

โปรแกรมจัดการสินค้าประกอบไปด้วย การเพิ่มข้อมูลที่จะเก็บข้อมูล ID, NAME, ราคา, ประเภท, จำนวน แสดงข้อมูลสินค้าทั้งหมดในโปรแกรม ค้นหาข้อมูลโดยใช้เลข ID เพื่อค้นหาสะดวกต่อการใช้งาน อัปเดตข้อมูลสามารถเปลี่ยนแปลงข้อมูลได้แต่ถ้าเราไม่ต้องการเปลี่ยนข้อมูลนั้นให้เราเว้นว่างไว้ ลบข้อมูลโดยใช้เลขเลข ID เพื่อลบข้อมูลทั้งหมดของเลข ID นั้น สร้างรายงานเพื่อทำสรุปของสินค้าแต่ละประเภทและจำนวนในแต่ละประเภทนั้นมีจำนวนเงินและจำนวนทั้งหมดของสินค้า จบการทำงานของโปรแกรม

#### 3.1 สำหรับผู้ใช้งานโปรแกรม

การใช้โปรแกรม การเพิ่มข้อมูล

3.1.1 กรอกรหัสเลข 1.ตามวงสีแดงที่ระบุเพื่อเรียกฟังก์ชันเพิ่มข้อมูลที่ประกอบไปด้วย ID, NAME, ราคาสินค้า, ประเภทสินค้า, จำนวนสินค้า



รูปที่ 3.1.1 การเพิ่มข้อมูล

3.1.2. เมื่อเมนูฟังก์ชันขึ้นมาแล้วจากนั้นก็สมารถระบุตามที่ต้องการได้

```
เลือกฟังก์ชัน (1 - 7):
Enter ID(รหัสสินค้า):
Enter Name(ชื่อสินค้า):
Enter Price(ราคาสินค้า):
Enter Product Type(ประเภทสินค้า):
Enter Quantity(จำนวนสินค้า):
```

รูปที่ 3.1.2 การเลือกฟังก์ชัน

3.2 การใช้โปรแกรม การแสดงข้อมูลทั้งหมด

3.2.1. กรอกหมายเลขที่2เมื่อแสดงข้อมูลทั้งหมดของสินค้า

```
1. เพิ่มข้อมูล
2. แสดงข้อมูลทั้งหมด
3. ค้นหาข้อมูล
4. อัปเดตข้อมูล
5. ลบข้อมูล
6. สร้างรายงาน
7. ออกจากโปรแกรม

เลือกฟังก์ชัน (1 - 7):
```

รูปที่ 3.2.1 การแสดงข้อมูลทั้งหมด

3.2.2. หลังจากกรอกหมายเลข2ข้อที่แล้วก็จะแสดงรายการสินค้ามาตามภาพที่ 2.2

ID	Name	Price	Product Type	Quantity
3	headphones	300.0	electronic	21
4	TV32"	1223.0	electronic	44
5	iphone	4999.0	electronic	55
6	shirt GQ	800.0	clothes	20
7	pantaloons	299.0	clothes	30

รูปที่ 3.2.2 แสดงข้อมูลสินค้าทั้งหมด



### 3.3 การใช้โปรแกรม การค้นหาข้อมูล

3.3.1.กรอกหมายเลขที่3เพื่อเข้าฟังก์ชันในค้นหาข้อมูล ID ดูเลข ID ได้ตามรูปที่2.2

```

1. เพิ่มข้อมูล
2. แสดงข้อมูลทั้งหมด
3. ค้นหาข้อมูล
4. อัปเดตข้อมูล
5. ลบข้อมูล
6. สร้างรายงาน
7. ออกจากโปรแกรม

เลือกฟังก์ชัน (1 - 7): 

```

รูปที่ 3.3.1 การค้นหาข้อมูล

3.3.2.กรอกหมายเลข ID เพื่อทำการค้นหาข้อมูล ในตัวอย่างนี้เราจะทำการกรอกหมายเลข3เพื่อเป็นตัวอย่างในการค้นหาข้อมูล

```

เลือกฟังก์ชัน (1 - 7): 3
Enter ID to search (ค้นหาข้อมูล ID): 3

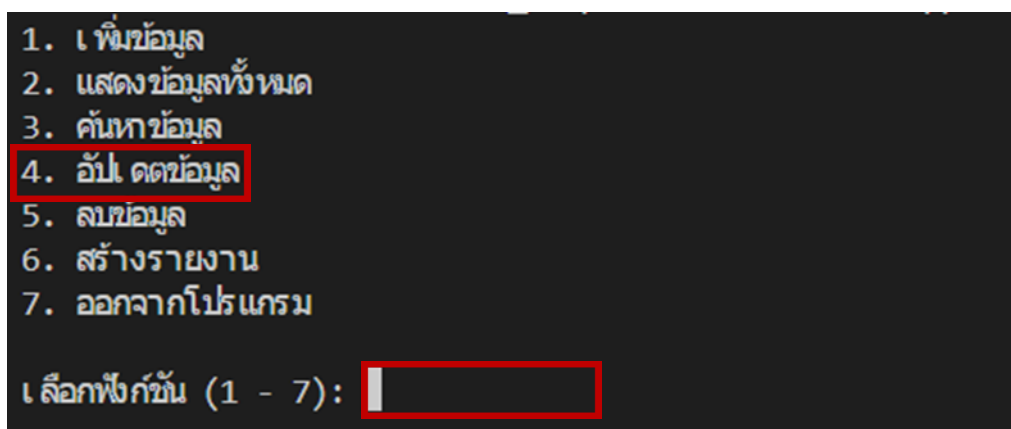
=====
ID      Name      Price      Product Type      Quantity
=====
3       headphones  300.0      electronic         21
=====

```

รูปที่ 3.3.2 การค้นหาข้อมูล

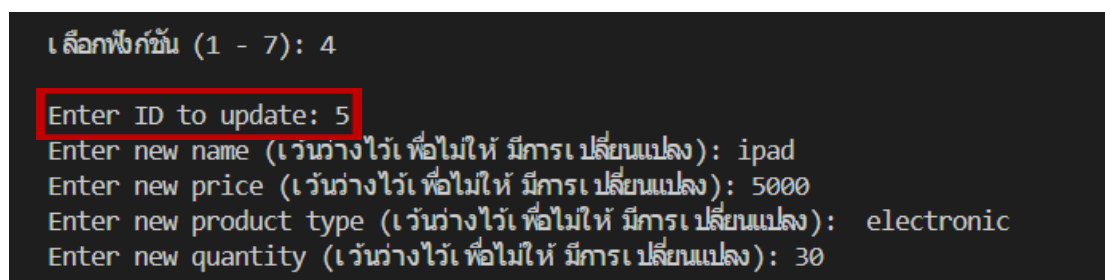
### 3.4 การใช้โปรแกรม การอัปเดตข้อมูล

3.4.1.กรอกหมายเลขที่4เพื่อทำการเลือกฟังก์ชันอัปเดตข้อมูลลงไปในคลังสินค้า



รูปที่3.4.1 การอัปเดตข้อมูล

3.4.2.กรอก ID ที่ต้องการจะเปลี่ยน ในตัวอย่างนี้กรอกหมายเลข5เพื่อเปลี่ยนข้อมูลจาก iphone เป็น ipad เปลี่ยน price และ quantity แต่ product type ยังคงเดิม โดยสามารถดูข้อมูลสินค้าที่ยังไม่อัปเดตรูปที่2.2



รูปที่3.4.2 การอัปเดตข้อมูล

### 3.5. การใช้โปรแกรม การลบข้อมูล

#### 3.5.1.กรอกหมายเลข 5 เพื่อทำการลบข้อมูลในรายการข้อมูล

```

1. เพิ่มข้อมูล
2. แสดงข้อมูลทั้งหมด
3. ค้นหาข้อมูล
4. อัปเดตข้อมูล
5. ลบข้อมูล
6. สร้างรายงาน
7. ออกจากโปรแกรม

เลือกฟังก์ชัน (1 - 7): 

```

รูปที่ 3.5.1 การลบข้อมูล

3.5.2.กรอก ID ที่ต้องการจะลบ โดยรูปที่ 5.2 กรอกหมายเลข 5 เพื่อเป็นตัวอย่างโดยสามารถดูข้อมูลสินค้าที่ยังไม่ได้ลบดังรูปที่ 2.2

```

เลือกฟังก์ชัน (1 - 7): 5
Enter ID to delete (เลือกเลข ID ที่ต้องการลบ): 5
1. เพิ่มข้อมูล
2. แสดงข้อมูลทั้งหมด
3. ค้นหาข้อมูล
4. อัปเดตข้อมูล
5. ลบข้อมูล
6. สร้างรายงาน
7. ออกจากโปรแกรม

เลือกฟังก์ชัน (1 - 7): 2

=====
ID      Name                Price      Product Type    Quantity
=====
3       headphones          300.0      electronic       21
4       TV32"               1223.0     electronic       44
6       shirt GQ             800.0      clothes          20
7       pantaloons           299.0      clothes          30
=====

```

รูปที่ 3.5.2 การลบข้อมูลรายการสินค้า

### 3.6 การใช้โปรแกรม สร้างรายงาน

#### 3.6.1 กรอกรหัส 6 เพื่อสร้างรายงาน

```

1. เพิ่มข้อมูล
2. แสดงข้อมูลทั้งหมด
3. ค้นหาข้อมูล
4. อัปเดตข้อมูล
5. ลบข้อมูล
6. สร้างรายงาน
7. ออกจากโปรแกรม

เลือกฟังก์ชัน (1 - 7): 6

----- รายงานได้ถูกบันทึกลงในไฟล์ 'report.txt' เรียบร้อยแล้ว. -----

```

รูปที่ 3.6.1 การสร้างรายงาน

#### 3.6.2 รายงานแยกประเภทสินค้า

```

----- รายงานแยกประเภทสินค้า -----
จำนวนประเภทสินค้าทั้งหมด: 2
=====
ประเภทสินค้า: electronic
รายการสินค้าทั้งหมด: 2
=====

```

ID	Name	Price	Quantity
3	headphones	300.00	21
4	TV32"	1223.00	44

```

-----
Total items: 65
Total value: 60112.00

=====
ประเภทสินค้า: clothes
รายการสินค้าทั้งหมด: 2
=====

```

ID	Name	Price	Quantity
6	shirt GQ	800.00	20
7	pantaloons	299.00	30

```

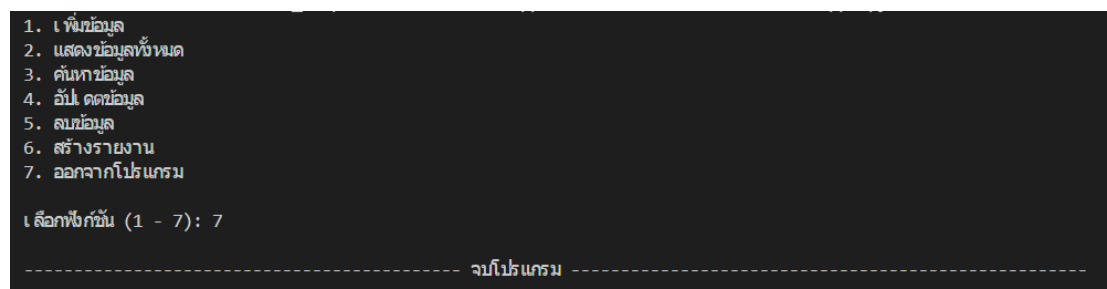
-----
Total items: 50
Total value: 24970.00

```

รูปที่ 3.6.2 รายงานแยกประเภทสินค้า

### 3.7 การใช้โปรแกรม ออกจากโปรแกรม

#### 3.7.1 กรอกหมายเลข 7 เพ้อออกจากโปรแกรม



1. เพิ่มข้อมูล  
2. แสดงข้อมูลทั้งหมด  
3. ค้นหาข้อมูล  
4. อัปเดตข้อมูล  
5. ลบข้อมูล  
6. สร้างรายงาน  
7. ออกจากโปรแกรม

เลือกฟังก์ชัน (1 - 7): 7

----- จบโปรแกรม -----

รูปที่ 3.7.1 จบโปรแกรม

## บทที่ 4

### อธิบายการทำงานของ code

โค้ดนี้เป็นโปรแกรมที่ทำงานกับไฟล์ไบนารีสำหรับการจัดการข้อมูลสินค้า โดยใช้โมดูล สำหรับการแปลงข้อมูลเป็นไบนารี และมีฟังก์ชันต่างๆ สำหรับการเพิ่ม, แสดง, ค้นหา, อัปเดต, ลบข้อมูลสินค้า และสร้างรายงานแยกตามประเภทสินค้า

#### การกำหนดรูปแบบข้อมูล

record\_format: กำหนดรูปแบบของข้อมูลแต่ละรายการสินค้าในไฟล์ ซึ่งประกอบด้วย:

I: ข้อมูลชนิด unsigned int (ID ของสินค้า) ขนาด 4 bytes

20s: ชื่อสินค้า เป็นสตริงความยาว 20 ตัวอักษร (20 bytes)

f: ราคาสินค้า เป็นชนิด float (4 bytes)

10s: ประเภทสินค้า เป็นสตริงความยาว 10 ตัวอักษร (10 bytes)

I: จำนวนสินค้า เป็นชนิด unsigned int (4 bytes)

record\_size: ใช้ฟังก์ชัน struct.calcsize เพื่อคำนวณขนาดของข้อมูลแต่ละรายการตามรูปแบบ record\_format ซึ่งจะเป็น 42 bytes ต่อ 1 รายการ filename: กำหนดชื่อไฟล์ที่จะใช้เก็บข้อมูลเป็น 'data.bin'

```
# Format ของ struct ในไฟล์ไบนารี
record_format = 'I20sf10sI'
record_size = struct.calcsize(record_format)
filename = 'data.bin'
```

รูปที่ 4 การกำหนดรูปแบบข้อมูล

#### 4.1 ฟังก์ชันสำหรับเพิ่มข้อมูล

ฟังก์ชันนี้ใช้สำหรับเพิ่มข้อมูลสินค้าใหม่ลงในไฟล์ใช้ `ljust()` เพื่อตัดหรือเพิ่มช่องว่างให้ชื่อสินค้า (name) และประเภทสินค้า (product\_type) มีความยาวตามที่กำหนด (20 ตัวอักษรและ 10 ตัวอักษรตามลำดับ) ใช้ `struct.pack()` เพื่อแปลงข้อมูลสินค้าให้อยู่ในรูปแบบไบนารีตาม `record_format` ใช้โหมด `ab` (append binary) เพื่อเปิดไฟล์และเขียนข้อมูลใหม่ต่อท้ายไฟล์เดิม

```
# 1 ฟังก์ชันสำหรับเพิ่มข้อมูล
def add_record(record_id, name, price, product_type, quantity):
    with open(filename, 'ab') as file:
        name = name.ljust(20)[:20]
        product_type = product_type.ljust(10)[:10]
        record = struct.pack(record_format, record_id, name.encode('utf-8'), price, product_type.encode('utf-8'), quantity)
        file.write(record)
```

รูปที่ 4.1 ฟังก์ชันสำหรับเพิ่มข้อมูล

#### 4.2 ฟังก์ชันสำหรับแสดงข้อมูลทั้งหมด

ฟังก์ชันนี้ใช้แสดงข้อมูลสินค้าทั้งหมดจากไฟล์ใช้ `struct.unpack()` เพื่อแปลงข้อมูลไบนารีกลับเป็นข้อมูลแบบเดิม แล้วนำมาแสดงผลบนหน้าจอใช้ `rb` (read binary) เพื่อเปิดไฟล์ในโหมดอ่านไบนารี

`print()` : แสดงบรรทัดว่างเพื่อความสวยงาม

`print("=" * 50)` : แสดงเส้นคั่นตารางโดยการพิมพ์เครื่องหมาย = จำนวน 50 ครั้ง (คุณด้วยช่องว่างเพื่อแสดงเป็นเส้นยาว)

`print(f'{"ID":<10}...')` : แสดงหัวตารางด้วยการจัดรูปแบบข้อมูล แต่ละฟิลด์ (ID, Name, Price, Product Type, Quantity) มีความกว้างที่กำหนดไว้

`print("=" * 50)` : แสดงเส้นคั่นตารางหลังหัวตาราง

`with open(filename, 'rb') as file` : เปิดไฟล์ filename ในโหมดอ่านแบบไบนารี (rb)  
`with` ทำหน้าที่ปิดไฟล์อัตโนมัติเมื่อจบการทำงาน

`while chunk := file.read(record_size)` : อ่านไฟล์ทีละบล็อก (ขนาดเท่ากับ `record_size`) จนกว่าจะหมดไฟล์ ตัวแปร `chunk` จะเก็บข้อมูลในรูปแบบไบนารีที่อ่านมาในแต่ละรอบ

`record = struct.unpack(record_format, chunk)` : ใช้ `struct.unpack` เพื่อแปลงข้อมูลไบนารีใน `chunk` ตามรูปแบบที่กำหนดใน `record_format` ซึ่งผลลัพธ์จะเก็บในตัวแปร `record`

`print(f'{record : แสดงข้อมูลในแต่ละบรรทัดตามลำดับฟิลด์ที่อ่านได้จากตัวแปร record โดยการใช้การจัดรูปแบบข้อมูลให้เรียงตามความกว้างที่กำหนด:`

`record[0]:<10: เก็บข้อมูล ID ให้อยู่ใน record[0]`

`record[1].decode('utf-8').strip():<35: เก็บชื่อสินค้า ให้อยู่ใน record[1] ซึ่งถูกแปลงจากไบนารีเป็นข้อความด้วย decode('utf-8') และตัดช่องว่างหน้า-หลังด้วย strip()`

`record[2]:<17: เก็บราคา ให้อยู่ใน record[2]`

`record[3].decode('utf-8').strip():<21: เก็บประเภทสินค้า ให้อยู่ใน record[3] ที่ถูกแปลงเป็นข้อความและจัดรูปแบบให้ชัดเจน`

`record[4]:<10: เก็บจำนวนสินค้า ให้อยู่ใน record[4]`

```
# 2 ฟังก์ชันสำหรับแสดงข้อมูลทั้งหมด
def display_records():
    # หัวตาราง
    print() # บรรทัดว่าง
    print("=" * 50) # เส้นคั่นตาราง
    print(f'{ "ID":<10}{ "Name":<35}{ "Price":<15}{ "Product Type":<20}{ "Quantity":<10}"')
    print("=" * 50) # เส้นคั่นตาราง

    with open(filename, 'rb') as file:
        while chunk := file.read(record_size):
            record = struct.unpack(record_format, chunk)
            # การจัดรูปแบบข้อมูลในแต่ละบรรทัด
            print(f'{record[0]:<10}{record[1].decode('utf-8').strip():<35}{record[2]:<17}{record[3].decode('utf-8').strip():<21}{record[4]:<10}"')
```

รูปที่ 4.2 ฟังก์ชันสำหรับแสดงข้อมูลทั้งหมด

#### 4.3 ฟังก์ชันสำหรับค้นหาข้อมูลตามรหัสสินค้า

ฟังก์ชันนี้ใช้ค้นหาข้อมูลสินค้าโดยใช้ ID ของสินค้าในการอ่านข้อมูลจากไฟล์และตรวจสอบว่ารหัสสินค้า (`record[0]`) ตรงกับ `search_id` ที่ค้นหาหรือไม่ หากพบจะคืนค่าข้อมูลสินค้านั้น

`with open(filename, 'rb') as file :` เปิดไฟล์ที่ชื่อ `filename` ในโหมดไบนารี (`rb`) ซึ่งเป็นโหมดที่เปิดไฟล์เพื่ออ่านข้อมูลในรูปแบบไบนารี การใช้ `with` จะช่วยจัดการปิดไฟล์อัตโนมัติเมื่อสิ้นสุดการทำงานภายในบล็อกนี้

`while chunk := file.read(record_size) :` ววนูปอ่านไฟล์ โดยขนาดของข้อมูลที่อ่านจะมีขนาดเท่ากับ `record_size` ตัวแปร `chunk` จะเก็บข้อมูลที่อ่านมา ถ้าอ่านจนไม่มีข้อมูลเหลือแล้ว (สิ้นสุดไฟล์) ลูปจะหยุดทำงาน



`record = struct.unpack(record_format, chunk)` : เป็นฟังก์ชันที่ใช้แปลงข้อมูลไบนารีในรูปแบบ `chunk` (อ่านข้อมูลออกเป็นช่วง ๆ) ให้เป็นข้อมูลที่สามารถนำมาใช้ได้ตามรูปแบบที่กำหนดโดย `record_format` ซึ่งเป็นโครงสร้างข้อมูล (เช่น `integer`, `string`, `float`) ที่เก็บไว้ใน `record` นั้น ๆ ผลลัพธ์ที่ได้จะถูกเก็บไว้ในตัวแปร `record`

`if record[0] == search_id` : ตรวจสอบว่าค่าของ `record[0]` หรือ ID เท่ากับหมายเลขระบุข้อมูลว่าตรงกับ `search_id` ที่ต้องการค้นหาหรือไม่

`return record` ถ้าพบ `record` ที่ `record[0]` ตรงกับ `search_id` จะคืนค่าข้อมูลของ `record` กลับมา

`return None` ถ้าไม่พบ `record` ที่ตรงกับ `search_id` หลังจากวนลูปจนหมดไฟล์แล้ว ฟังก์ชันจะคืนค่า `None`

```
# 3 ฟังก์ชันสำหรับค้นหาข้อมูลตามรหัสสินค้า
def find_record_by_id(search_id):
    with open(filename, 'rb') as file:
        while chunk := file.read(record_size):
            record = struct.unpack(record_format, chunk)
            if record[0] == search_id:
                return record
    return None
```

รูปที่ 4.3 ฟังก์ชันสำหรับค้นหาข้อมูลตามรหัสสินค้า

#### 4.4 ฟังก์ชันสำหรับอัปเดตข้อมูลสินค้า

ฟังก์ชัน `update_record()` ทำหน้าที่ในการอัปเดตข้อมูลสินค้าที่อยู่ในไฟล์ไบนารี โดยสามารถอัปเดตชื่อสินค้า (`new_name`), ราคาสินค้า (`new_price`), ประเภทสินค้า (`new_product_type`), และจำนวนสินค้า (`new_quantity`) ได้ตามที่ต้องการ โครงสร้างการทำงาน เปิดไฟล์ข้อมูลต้นฉบับ (`filename`) ในโหมดอ่านไบนารี (`'rb+'`) ซึ่งหมายถึงสามารถอ่านและเขียนไฟล์ได้ใช้ลูป `while` เพื่ออ่านข้อมูลที่ละชุดจากไฟล์ โดยแต่ละชุดข้อมูลจะถูกอ่านตามขนาดที่กำหนดในตัวแปร `record_size` สำหรับแต่ละชุดข้อมูล จะบันทึกตำแหน่งปัจจุบันของไฟล์ (ใช้ `file.tell()` เพื่อบันทึกตำแหน่ง) และทำการถอดรหัสข้อมูลไบนารีโดยใช้ `struct.unpack()` ให้ได้ข้อมูลในรูปแบบที่สามารถอ่านได้ตามฟอร์แมตที่กำหนดใน `record_format` ตรวจสอบว่า `record[0]` (ซึ่งคือ `record_id` ของข้อมูลที่อ่านมา) ตรงกับ `record_id` ที่ต้องการอัปเดตหรือไม่ ถ้าตรงกับ `record_id` ที่ต้องการอัปเดต จะเริ่มกระบวนการอัปเดตข้อมูลแต่ละส่วนตามสิ่งที่ส่งเข้ามาใน

ถ้ามีค่า `new_name` จะปรับชื่อสินค้าโดยจัดรูปแบบให้มีความยาว 20 ตัวอักษร

ถ้าไม่มีค่า new\_name จะใช้ชื่อเดิมจากข้อมูลที่อ่านมาเช่นเดียวกันกับ new\_product\_type, new\_price และ new\_quantity ถ้ามีค่าที่ส่งมาใหม่ก็จะอัปเดต แต่ถ้าไม่มีค่าก็จะใช้ข้อมูลเดิม เมื่ออัปเดตข้อมูลเรียบร้อยแล้ว จะทำการแพ็คข้อมูลกลับเป็นไบนารีด้วย struct.pack() แล้วเขียนข้อมูลนั้นกลับไปตำแหน่งเดิมในไฟล์โดยใช้ file.seek() เพื่อเลื่อนไปยังตำแหน่งเดิมในไฟล์ แล้วใช้ file.write() เพื่อเขียนข้อมูลที่อัปเดตใหม่ลงไปหลังจากเขียนข้อมูลที่อัปเดตแล้วจะออกจากลูปโดยใช้ break เพื่อหยุดการทำงาน

```
# 4 ฟังก์ชันสำหรับอัปเดตข้อมูลสินค้า
def update_record(record_id, new_name=None, new_price=None, new_product_type=None, new_quantity=None):
    with open(filename, 'r+b') as file:
        while chunk := file.read(record_size):
            pos = file.tell() - record_size
            record = struct.unpack(record_format, chunk)
            if record[0] == record_id:
                if new_name:
                    new_name = new_name.ljust(20)[:20]
                else:
                    new_name = record[1].decode()

                if new_product_type:
                    new_product_type = new_product_type.ljust(10)[:10]
                else:
                    new_product_type = record[3].decode()

                if new_price is None:
                    new_price = record[2]

                if new_quantity is None:
                    new_quantity = record[4]

                updated_record = struct.pack(record_format, record_id, new_name.encode(), new_price,
                                             new_product_type.encode(), new_quantity)

                file.seek(pos)
                file.write(updated_record)
                break
```

รูปที่ 4.4 ฟังก์ชันสำหรับอัปเดตข้อมูลสินค้า

#### 4.5 ฟังก์ชันสำหรับลบข้อมูลสินค้า

เป็นฟังก์ชันที่ทำหน้าที่ในการลบข้อมูลที่มี id ตรงกับค่าที่มีในไฟล์ข้อมูล โดยวิธีการทำงานสามารถกำหนดไฟล์ชั่วคราวชื่อ temp\_file เพื่อใช้เก็บข้อมูลที่เหลือหลังจากการลบเปิดไฟล์ข้อมูลต้นฉบับ ด้วยโหมดอ่านแบบไบนารี ('rb') และเปิดไฟล์ชั่วคราวด้วยโหมดเขียนแบบไบนารี ('wb') ใช้ลูปวนเพื่ออ่านข้อมูลจากไฟล์ต้นฉบับทีละส่วนและใช้ struct.unpack() เพื่อถอดรหัสข้อมูลไบนารีให้กลับมาเป็นข้อมูลแบบที่อ่านได้ โดย record\_format คือรูปแบบของข้อมูลที่ต้องการถอดรหัส

ตรวจสอบว่า record[0] ซึ่งเป็น id ของข้อมูลนั้น ตรงกับ id ที่เราต้องการลบหรือไม่ถ้าตรง ก็จะเขียนข้อมูลลงไฟล์ชั่วคราวเมื่อเสร็จสิ้นกระบวนการอ่านและเขียนแล้ว จะใช้ os.replace() เพื่อแทนที่ไฟล์ temp\_file เข้าไปบันทึกในไฟล์ต้นฉบับ

```

break
# 5 ฟังก์ชันสำหรับลบข้อมูลสินค้า
def delete_record(record_id):
    temp_file = 'temp.bin'
    with open(filename, 'rb') as infile, open(temp_file, 'wb') as outfile:
        while chunk := infile.read(record_size):
            record = struct.unpack(record_format, chunk)
            if record[0] != record_id:
                outfile.write(chunk)
    import os
    os.replace(temp_file, filename)

```

รูปที่ 4.5 ฟังก์ชันสำหรับลบข้อมูลสินค้า

#### 4.6 ฟังก์ชันสำหรับเขียนรายงาน

หน้าที่ของฟังก์ชัน generate\_report\_by\_product\_type คือการสร้างรายงานที่จัดหมวดหมู่สินค้าแต่ละประเภท โดยจะแสดงรายละเอียดของสินค้าที่อยู่ในประเภทนั้น ๆ รวมถึงสรุปจำนวนสินค้าทั้งหมดและมูลค่ารวมของสินค้าในประเภทนั้น แล้วบันทึกข้อมูลในรูปแบบไฟล์ .txt

##### 4.6.1 ขั้นตอนการทำงานของฟังก์ชัน

ใช้ defaultdict ในการเก็บข้อมูลตามประเภทสินค้า โดย:

product\_details: เก็บรายละเอียดสินค้าตามแต่ละประเภทในรูปแบบของรายการ (list)

product\_totals: เก็บผลรวมของราคาสินค้าตามประเภทนั้น ๆ

product\_quantities: เก็บจำนวนสินค้ารวมของแต่ละประเภท

```

def generate_report_by_product_type():
    product_details = defaultdict(list)
    product_totals = defaultdict(float)
    product_quantities = defaultdict(int)

```

รูปที่ 4.6.1 ขั้นตอนการทำงานของฟังก์ชัน

#### 4.6.2 เปิดไฟล์สินค้าในโหมดอ่านไบนารี

with open(filename, 'rb') as file : เปิดไฟล์ที่ชื่อว่า filename ในโหมดอ่านแบบไบนารี ('rb') with ทำให้เมื่อการทำงานภายในบล็อกเสร็จสิ้น ไฟล์จะถูกปิดอัตโนมัติ

ใช้ฟังก์ชัน struct.unpack() เพื่อแปลงข้อมูลไบนารี (chunk) ให้อยู่ในรูปแบบของข้อมูลตามที่ระบุใน record\_format

```
with open(filename, 'rb') as file:
    while chunk := file.read(record_size):
        record = struct.unpack(record_format, chunk)
```

รูปที่ 4.6.2 เปิดไฟล์สินค้าในโหมดอ่านไบนารี

#### 4.6.3 แยกข้อมูลแต่ละรายการออกมา

กำหนดค่าให้กับตัวแปร record\_id โดยนำค่าแรกในรายการ record[0] มาเก็บไว้ record\_id

นำค่าที่สองในรายการ record[1] มาเก็บในตัวแปร name โดยใช้ decode('utf-8') เพื่อแปลงข้อมูลที่อยู่ในรูปแบบไบนารี (bytes) ให้เป็นสตริง (string) ที่เข้ารหัสด้วย UTF-8 และใช้ strip() เพื่อลบช่องว่าง (whitespace)

กำหนดค่าให้กับตัวแปร price โดยนำข้อมูลมาจาก record[2]

กำหนดตัวแปรชื่อ product type มาเก็บข้อมูลจาก record[3] โดยใช้ decode('utf-8') เพื่อแปลงข้อมูลที่อยู่ในรูปแบบไบนารี (bytes) ให้เป็นสตริง (string) ที่เข้ารหัสด้วย UTF-8 และใช้ strip() เพื่อลบช่องว่างของข้อความ

กำหนดตัวแปรชื่อ quantity โดยกำหนดข้อมูลจาก record[4]

```
record_id = record[0]
name = record[1].decode('utf-8').strip()
price = record[2]
product_type = record[3].decode('utf-8').strip()
quantity = record[4]
```

รูปที่ 4.6.3 แยกข้อมูลแต่ละรายการออกมา

## 4.7 จัดเก็บข้อมูลสินค้า

จากนั้นเราเพิ่มข้อมูลสินค้าลงในรายการ product\_type โดยจัดรูปแบบเป็นตารางแสดงรหัสสินค้า ชื่อสินค้า ราคา และจำนวน จากนั้นคำนวณผลรวมของราคาสินค้าในประเภทนั้นโดยนำราคาสินค้า (price) คูณด้วยจำนวน (quantity) แล้วบวกเพิ่มใน product\_type พร้อมทั้งบวกจำนวนสินค้าที่เพิ่มเข้ามาใน product\_type เพื่อสะสมจำนวนสินค้าในประเภทนั้นๆ

```
# เก็บรายละเอียดสินค้าในรูปแบบตาราง
product_details[product_type].append(
    f"{record_id:<5}| {name:<30}| {price:>10.2f} | {quantity:>10} |"
)

# เพิ่มราคาสินค้าและจำนวนของประเภทนั้นๆ
product_totals[product_type] += price * quantity
product_quantities[product_type] += quantity
```

รูปที่ 4.7 จัดเก็บข้อมูลสินค้า

### 4.7.1 สร้างเนื้อหารายงาน

การสร้างรายงานในรูปแบบตารางเพื่อนำเสนอข้อมูลสินค้าแยกตามประเภท โดยขั้นตอนและการทำงานจะเริ่มต้นจากการสร้างหัวรายงาน report\_content ซึ่งประกอบด้วยจำนวนประเภทสินค้าที่มีทั้งหมดในระบบ ต่อมาจะทำการวนลูปเพื่อดึงข้อมูลในแต่ละประเภทสินค้า และจัดทำรายงานตามประเภทนั้นๆ โดยภายในแต่ละประเภทสินค้า จะเพิ่มบรรทัดที่บอกชื่อประเภทสินค้า และจำนวนรายการสินค้าที่อยู่ในประเภทนั้น

หลังจากนั้นจะสร้างหัวตารางที่ประกอบไปด้วย รหัสสินค้า ID, ชื่อสินค้า Name, ราคา Price และจำนวน Quantity

จากนั้นจะทำลูปที่วนเข้าไปดูข้อมูลแล้วนำมาแสดงรายการสินค้าทั้งหมดในประเภทนั้นๆ โดยแต่ละรายการจะถูกเพิ่มเข้าไปใน report\_content ด้วยข้อมูลของสินค้าจริง ที่มาจากข้อมูลที่เราเพิ่มเข้ามาให้ระบบ

เมื่อรายการสินค้าถูกแสดงทั้งหมดแล้ว และทำการรวมจำนวนของสินค้าทั้งหมดจากนั้นเราจะทำการเพิ่มราคารวมของสินค้านั้นรวมของสินค้าในแต่ละประเภทนั้น

```
# เริ่มสร้างรายงานแยกประเภทในรูปแบบตาราง
report_content = "\n----- รายงานแยกประเภทสินค้า ----- \n"
report_content += f"จำนวนประเภทสินค้าทั้งหมด: {len(product_details)}\n" # ประเภทสินค้าที่มีในสต็อกทั้งหมด

for product_type, items in product_details.items():
    report_content += " " * 32 + "\n" # เส้นคั่นตาราง
    report_content += f"ประเภทสินค้า: {product_type}\n" # ชื่อประเภทสินค้า
    report_content += f"รายการสินค้าทั้งหมด: {len(items)}\n" # ชื่อจำนวนสินค้าทั้งหมด
    report_content += " " * 32 + "\n" # เส้นคั่นตาราง
    report_content += f"{'ID':<5} | {'Name':<30} | {'Price':>10} | {'Quantity':>10} | \n" # หัวข้อของข้อมูล
    report_content += " " * 63 + "\n" # เส้นคั่นตาราง

    for item in items:
        report_content += f"{item}\n"

    report_content += " " * 63 + "\n" # เส้นคั่นตาราง
    # ยอดรวมจำนวนสินค้าและมูลค่าสินค้าของประเภทนั้น
    report_content += f"{'Total items':<38} {product_quantities[product_type]:>10}\n"
    # มูลค่าสินค้าของสินค้าทั้งหมดในประเภทนั้น
    report_content += f"{'Total value':<38} {product_totals[product_type]:>10.2f}\n"
    report_content += " " * 63 + "\n" # เส้นคั่นตาราง
```

รูปที่ 4.7.1 สร้างเนื้อหารายงาน

#### 4.7.2 บันทึกรายงานลงไฟล์ report.txt

เราจะใช้คำสั่งสร้างไฟล์ report.txt โดยการเปิดไฟล์ในโหมดเขียน และ เซอร์หัสเป็นป็น utf-8 เพื่อรองรับการใช้ภาษาไทยหรืออักขระพิเศษอื่นๆ เขียนเนื้อหาของรายงานแล้วเก็บไว้ในตัวแปร report\_content ลงในไฟล์ จากนั้นปิดไฟล์เมื่อการเขียนเสร็จสมบูรณ์ จัดการปิดไฟล์ให้อัตโนมัติเมื่อใช้ with open() เพื่อป้องกันการตั้งชื่อไฟล์ผิด หลังจากบันทึกจะแสดงข้อความว่า รายงานได้ถูกบันทึก ลงไฟล์ report.txt แล้ว

```
# บันทึกรายงานลงไฟล์ report.txt
with open('report.txt', 'w', encoding='utf-8') as report_file:
    report_file.write(report_content)
print()
print("\n----- รายงานได้ถูกบันทึกลงไฟล์ 'report.txt' เรียบร้อยแล้ว. -----")
print()
```

รูปที่ 4.7.2 บันทึกรายงานลงไฟล์ report.txt

#### 4.8 เมนูสำหรับเลือกฟังก์ชัน

ฟังก์ชัน menu ทำหน้าที่เป็นเมนูหลักของโปรแกรมที่ให้ผู้ใช้งานสามารถเลือกฟังก์ชันที่ต้องการดำเนินการ เช่น เพิ่มข้อมูล แสดงข้อมูล ค้นหาข้อมูล อัปเดต ลบ หรือสร้างรายงาน ออกจากโปรแกรม เลือกฟังก์ชัน (1 - 7): รับอินพุตจากผู้ใช้งาน โดยให้ผู้ใช้งานเลือกตัวเลขที่ต้องการ

```
def menu():
    while True:
        print("1. เพิ่มข้อมูล")
        print("2. แสดงข้อมูลทั้งหมด")
        print("3. ค้นหาข้อมูล")
        print("4. อัปเดตข้อมูล")
        print("5. ลบข้อมูล")
        print("6. สร้างรายงาน")
        print("7. ออกจากโปรแกรม")
        print()
        choice = int(input("เลือกฟังก์ชัน (1 - 7): "))
```

รูปที่ 4.8 เมนูสำหรับเลือกฟังก์ชัน

#### 4.9 เพิ่มข้อมูลสินค้า

ถ้าได้รับอินพุต 1:

จะให้ผู้ใช้ใส่ข้อมูล ID, Name, Price, Product Type, Quantity แล้วส่งค่าไปที่ฟังก์ชัน add\_record

ถ้าได้รับอินพุต 2:

จะเรียกใช้ฟังก์ชัน display\_records() เพื่อแสดงรายการสินค้าทั้งหมดที่มีอยู่ในระบบ

ถ้าได้รับอินพุต 3: ค้นหาข้อมูล

จะให้ผู้ใช้ป้อน id เพื่อค้นหาสินค้าเฉพาะตัวตามรหัสสินค้านั้น ถ้าพบสินค้า จะแสดงรายละเอียดของสินค้านั้นโดยแสดง ID, Name, Price, Product Type, และ Quantity

ถ้าได้รับอินพุต 4:

จะให้ผู้ใช้ป้อน id ของสินค้าที่ต้องการแก้ไข แล้วจะรับข้อมูลใหม่จากผู้ใช้ป้อน ชื่อใหม่ , ราคาที่แก้ไข , ประเภทสินค้าใหม่ , และจำนวนใหม่ หากไม่ต้องการเปลี่ยนให้เว้นว่างไว้ หลังจากนั้นจะเรียกฟังก์ชัน update\_record() เพื่ออัปเดตข้อมูลของสินค้านั้นในระบบ

ถ้าได้รับอินพุต 5:

จะให้ผู้ใช้ป้อน id ของสินค้าที่ต้องการลบจากระบบ และเรียกใช้ฟังก์ชัน delete\_record() เพื่อทำการลบสินค้านั้นออกจากระบบ

ถ้าได้รับอินพุต 6:

จะเรียกใช้ฟังก์ชัน generate\_report\_by\_product\_type() เพื่อสร้างรายงานที่แบ่งตามประเภทสินค้า และอาจแสดงหรือบันทึกรายงานนี้ในรูปแบบไฟล์

ถ้าได้รับอินพุต 7:

จะ จบการทำงานของโปรแกรม และทำงานออกจากลูป

ถ้าไม่มีเงื่อนไขใดตรง:

โค้ดจะแสดงข้อความเตือนว่า "ตัวเลือกไม่ถูกต้อง โปรดลองอีกครั้ง"

ซึ่งผู้ใช้ป้อน ตัวเลือกที่กำหนด 1 - 7

```

if choice == 1:
    record_id = int(input("Enter ID(รหัสสินค้า): "))
    name = input("Enter Name(ชื่อสินค้า): ")
    price = float(input("Enter Price(ราคาสินค้า): "))
    product_type = input("Enter Product Type(ประเภทสินค้า): ")
    quantity = int(input("Enter Quantity(จำนวนสินค้า): "))
    add_record(record_id, name, price, product_type, quantity)
elif choice == 2:
    display_records()
elif choice == 3:
    record_id = int(input("Enter ID to search (ค้นหาข้อมูล ID): "))
    record = find_record_by_id(record_id)
    if record:
        # แสดงหัวข้อยของตาราง
        print()
        print("-" * 97)
        print(f"{'ID':<10}{'Name':<35}{'Price':<15}{'Product Type':<20}{'Quantity':<10}")
        print("-" * 97)

        # แสดงข้อมูลของ record ที่ค้นพบ
        print(f"{record[0]:<9}{record[1].decode().strip():<37}{record[2]:<17}{record[3].decode().strip():<20}{record[4]:<9}")
        print("-" * 97)
        print()
    else:
        print("ไม่พบข้อมูลที่บันทึก")
elif choice == 4:
    print()
    record_id = int(input("Enter ID to update: "))
    name = input("Enter new name (เว้นว่างไว้เพื่อไม่ให้ มีการเปลี่ยนแปลง): ")
    price = input("Enter new price (เว้นว่างไว้เพื่อไม่ให้ มีการเปลี่ยนแปลง): ")
    product_type = input("Enter new product type (เว้นว่างไว้เพื่อไม่ให้ มีการเปลี่ยนแปลง): ")
    quantity = input("Enter new quantity (เว้นว่างไว้เพื่อไม่ให้ มีการเปลี่ยนแปลง): ")
    update_record(record_id, name or None, float(price) if price else None, product_type or None, int(quantity) if quantity else None)
    print()
elif choice == 5:
    record_id = int(input("Enter ID to delete (เลือกเลข ID ที่ต้องการลบ): "))
    delete_record(record_id)
elif choice == 6:
    generate_report_by_product_type()
elif choice == 7:
    print()
    print("----- จบโปรแกรม -----")
    print()
    break
else:
    print("\n""ตัวเลือกไม่ถูกต้อง โปรดลองอีกครั้ง ให้เลือก(1 - 7)""\n")

# เริ่มโปรแกรม
menu()

```

รูปที่ 4.9 เพิ่มข้อมูลสินค้า