



Project Term Assignment

โปรแกรม ลงทะเบียนเรียน Register for classes

รายวิชา Computer Programming รหัสรายวิชา 060233115

กลุ่ม All Star To The Goal

Sec เรียนที่ 1

เสนอ

รศ.ดร อนิราช มีงขวัญ

สมาชิก

นาย ธนบุร์ย อิมเจริญ รหัสนักศึกษา: 68060225510017

นาย วัชโรทัย เลิศวิถัย รหัสนักศึกษา: 68060225510025

นาย กัณติกรณ์ สรสุริยวงศ์ รหัสนักศึกษา: 6806022510068

นาย พีรพล บัวขยาย รหัสนักศึกษา: 6806022510049

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ วิทยาเขตปราจีนบุรี

คณะเทคโนโลยีและการจัดการอุตสาหกรรม

ภาควิชาเทคโนโลยีสารสนเทศ สาขาวิชา วิศวกรรมสารสนเทศและเครื่องข่าย

คำนำ

การจัดทำโครงการ “ระบบจัดการข้อมูลหนังสือ” นี้เป็นส่วนหนึ่งของวิชา COMPUTER PROGRAMMING ของหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศคณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ เพื่อให้นักศึกษาได้знакомรู้ที่เรียนมาทั้งหมดมาประยุกต์ใช้ในการพัฒนาโปรแกรมที่สามารถทำงานได้จริง โดยเน้นการออกแบบและเขียนโปรแกรมในภาษา Python ซึ่งเป็นภาษาที่เรียนมาในวิชา COMPUTER PROGRAMMING โดยโครงการนี้จะช่วย การคิดวิเคราะห์ และแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมในการประกอบอาชีพด้านวิศวกรรมสารสนเทศและเครือข่ายในอนาคต หากมีข้อผิดพลาดประการใด คณะผู้จัดทำต้องขออภัยไว้ ณ ที่นี่ด้วย

สารบัญ

	หน้า
บทที่ 1	1
บทนำ	1
1.1 วัตถุประสงค์ของโครงการ	1
1.2 ขอบเขตของโครงการ	1
1.3 ประโยชน์ที่ได้รับ	1
1.4 เครื่องมือที่คาดว่าจะต้องใช้	1
บทที่ 2	2
ระบบ ลงทะเบียนเรียน	2
1. StudentData.bin (ข้อมูลนักเรียน)	2
2. CourseSubject.bin (ข้อมูลวิชา)	3
3. RegisterCourse.bin (ข้อมูลการลงทะเบียน)	4
4. โครงสร้างไฟล์ Report.txt (ฉบับสมบูรณ์)	5
โครงสร้างไฟล์ ของโปรแกรม	6
1. module * __pycache__ (โฟลเดอร์สำหรับเก็บไฟล์คอมไพล์ของ Python)	6
2. ภายใต้ Folder module Module	6
3. ไฟล์ Main หลัก ในการทำงานของโปรแกรม ปฏิบัติการและเอกสาร (Main Execution & Documentation)	6
4. ไฟล์ฐานข้อมูลbinary (Binary Database Files)	6
ขอบเขตการทำงาน ของโปรแกรม	7
1. ขอบเขตการทำงานของไฟล์ main.py	7
1.1 จุดเริ่มต้นการทำงานของโปรแกรม	7
1.2 การจัดการส่วนประกอบและโครงสร้าง (Module Management)	7
1.3 การแสดงผลและการจัดการเมนูหลัก (Main Menu Handling)	7
1.4 การควบคุมการไหลของโปรแกรม (Flow Control)	8

สารบัญ(ต่อ)

	หน้า
1.5 การสิ้นสุดการทำงาน (Program Termination)	8
2. ขอบเขตการทำงานของ Module: การจัดการข้อมูลนักเรียน (student.py)	8
2.1 การจัดการโครงสร้างข้อมูลและการอ่าน/เขียนไฟล์ (Data & File Handling)	8
2.2 ฟังก์ชันการจัดการข้อมูลหลัก (CRUD Operations)	9
2.3 การค้นหาและการเรียกดูข้อมูล (View & Search)	10
2.4 เมนูย่อย (Control Flow)	10
3. ขอบเขตการทำงานของ Module: การจัดการข้อมูลการลงทะเบียน	11
3.1 การจัดการโครงสร้างข้อมูลและการจัดการไฟล์ (Data & File Handling)	11
3.2 การจัดการข้อมูลหลัก (Core Operations - CRUD)	11
3.2.1 เพิ่มการลงทะเบียน (add_registration):	11
3.2.2 แก้ไขการลงทะเบียน (update_registration):	12
3.2.3 ลบการลงทะเบียน (delete_registration):	12
3.3 การค้นหาและการเรียกดูข้อมูล (View & Search)	12
3.4 การพึ่งพาข้อมูล (Data Dependency)	13
3.5 เมนูย่อย (Control Flow)	13
4. ขอบเขตการทำงานของ Module: การจัดการข้อมูลรายวิชา Module	13
4.1 การจัดการโครงสร้างข้อมูลและไฟล์ (Data Structure & File Management)	13
4.2 การดำเนินการหลัก (CRUD) Module นี้รองรับการจัดการข้อมูลรายวิชาอย่างครบถ้วน:	14
4.2.1 เพิ่มรายวิชา (add_course):	14
4.2.2 แก้ไขรายวิชา (update_course):	14
4.2.3 ลบรายวิชา (delete_course):	14
4.3 การแสดงรายงานและการกรองข้อมูล (Reporting, Viewing, and Filtering)	14
4.4 ส่วนต่อประสานกับผู้ใช้ (User Interface)	15
5. ขอบเขตการทำงานของ Module: การสร้างรายงานและการวิเคราะห์ Module	15
5.1 การเชื่อมโยงข้อมูลและแหล่งที่มา (Data Integration)	15

สารบัญ(ต่อ)

	หน้า
5.2 รายงานประเภทที่ 1: รายงานนักศึกษา (print_student_report)	16
5.3 รายงานประเภทที่ 2: รายงานการลงทะเบียนและการวิเคราะห์สถิติ (Core Analysis)	16
5.4 ผลลัพธ์ ขอบเขต รายงาน ของโปรแกรม	17
6. ขอบเขตการทำงานและความสอดคล้องของ Binary File ทั้ง 3	18
6.1 student.bin (ฐานข้อมูลนักศึกษา)	18
6.2 CourseSubject.bin (ฐานข้อมูลรายวิชา)	19
6.3 registration.bin (ฐานข้อมูลการลงทะเบียน)	19
6.4 ทั้งสามไฟล์ทำงานร่วมกันผ่านรหัสที่ถูกแชร์ระหว่างไฟล์:	20
อธิบายโค้ดการทำงานของโปรแกรม	21
1.ไฟล์ main.py	21
2.ไฟล์ student.py	22
3.ไฟล์ register.py	30
4.ไฟล์ course.py	45
5.ไฟล์ report.py	52
ขั้นตอนการทำงานของโปรแกรม	65

สารบัญภาพ

	หน้า
ภาพที่ 1-1 โค้ดส่วนการทำงานหลักของโปรแกรม (main.py)	21
ภาพที่ 2-1 "โค้ดส่วนการแปลงข้อมูลนักศึกษาเป็นรูปแบบใบหนารี (Packing)	22
ภาพที่ 2-2 โค้ดไฟล์ student.py ฟังก์ชัน read_student_record	23
ภาพที่ 2-3 ไฟล์ student.py ฟังก์ชัน read_all_records_from_file และ print_student_report	24
ภาพที่ 2-4 ไฟล์โค้ดฟังก์ชันเพิ่มข้อมูล แสดงข้อมูลทั้งหมด และแสดงข้อมูลรายบุคคล	25
ภาพที่ 2-5 โค้ด การทำงาน ของ ฟังก์ชัน view_filtered_students (กรองข้อมูล)	26
ภาพที่ 2-6 ไฟล์ ฟังก์ชันนี้ทำหน้าที่ แก้ไขข้อมูลนักศึกษา	27
ภาพที่ 2-7 โค้ด ฟังก์ชัน delete_student (ลบข้อมูล) และ ฟังก์ชัน student_menu	29
ภาพที่ 3-1 โค้ด ฟังก์ชัน create_registration_record(บันทึกข้อมูลการลงทะเบียน)	30
ภาพที่ 3-2 โค้ดฟังก์ชัน read_registration_record(อ่านบันทึกข้อมูลการลงทะเบียน)	31
ภาพที่ 3-3 โค้ดwrite_record_to_file (เขียนบันทึกข้อมูลลงในไฟล์)	32
ภาพที่ 3-4 โค้ดฟังก์ชัน read_all_records_from_file(อ่านข้อมูลบันทึกไฟล์ทั้งหมด)	33
ภาพที่ 3-5โค้ด ฟังก์ชัน get_next_register_id(ทำการลงทะเบียนถัดไป)	34
ภาพที่ 3-6 โค้ดฟังก์ชัน read_student_by_id(อ่านข้อมูลนักเรียน)	35
ภาพที่ 3-7 โค้ด ฟังก์ชัน get_student_info_for_registration(ดึงข้อมูลนักเรียน)	36
ภาพที่ 3-8 โค้ดฟังก์ชัน print_registration_report(แสดงรายงานการลงทะเบียน)	37
ภาพที่ 3-9 โค้ดฟังก์ชัน add_registration(เพิ่มข้อมูลการลงทะเบียน)	38
ภาพที่ 3-10 ฟังก์ชัน view_registration(แสดงข้อมูลการลงทะเบียนทั้งหมด)	39
ภาพที่ 3-11 โค้ดฟังก์ชัน view_single_registration(แสดงข้อมูลการลงทะเบียน)	40
ภาพที่ 3-12 โค้ดฟังก์ชัน view_filtered_registration(แสดงการลงทะเบียน)	41
ภาพที่ 3-13 โค้ดฟังก์ชัน update_registration (แก้ไขข้อมูลนักเรียน)	42
ภาพที่ 3-14 โค้ดฟังก์ชัน dalete_registration(แสดงการลงทะเบียนแบบลบ)	43
ภาพที่ 3-15 โค้ดฟังก์ชัน registration_menu(เมนูย่อยสำหรับการจัดการข้อมูลลงทะเบียน)	44
ภาพที่ 4-1 ภาพโค้ดฟังก์ชันกำหนดชื่อไฟล์เป็น CourseSubject.bin และ create_course_record()	45
ภาพที่ 4-2 ไฟล์โค้ด read_course_record (อ่านข้อมูล) และ write_record_to_file (เขียนข้อมูล)	46
ภาพที่ 4-3 ไฟล์โค้ด ฟังก์ชัน read_all_records_from_file (อ่านทั้งหมด) และ แสดงรายงาน	47
ภาพที่ 4-4 ไฟล์โค้ด ฟังก์ชัน add_course (เพิ่มข้อมูล) แสดงข้อมูลทั้งหมด และแสดงรายบุคคล	48
ภาพที่ 4-5 ไฟล์โค้ด แสดงรายงานรายวิชาเฉพาะกลุ่ม ตามเงื่อนไขที่ผู้ใช้เลือก	49
ภาพที่ 4-6 โค้ดไฟล์ฟังก์ชันนี้ทำหน้าที่ แก้ไขข้อมูลรายวิชา	50
ภาพที่ 4-7 โค้ดไฟล์ ฟังก์ชัน delete_course (ลบข้อมูล) และ ฟังก์ชัน course_menu	51

สารบัญภาพ(ต่อ)

	หน้า
ภาพที่ 5-1 ไฟล์โค้ด การกำหนดพารช (Path Definition) การกำหนดรูปแบบใบหน้าและ ตั้งค่าอื่นๆ	52
ภาพที่ 5-2 ไฟล์โค้ด พังก์ชัน read_student_record และ พังก์ชัน read_all_students	53
ภาพที่ 5-4 ไฟล์โค้ด พังก์ชัน read_register_record และ พังก์ชัน read_all_registrations	55
ภาพที่ 5-5 ไฟล์โค้ด พังก์ชัน print_student_report (รายงานนักศึกษา)	56
ภาพที่ 5-6 ไฟล์โค้ด พังก์ชัน analyze_registration_statistic	57
ภาพที่ 5-7 ไฟล์โค้ด พังก์ชัน print_register_report (รายงานการลงทะเบียน)	58
ภาพที่ 5-8 ไฟล์โค้ด พังก์ชัน print_register_report (รายงานการลงทะเบียน)	59
ภาพที่ 5-9 ไฟล์โค้ด พังก์ชัน print_register_report (รายงานการลงทะเบียน)	60
ภาพที่ 5-10 ไฟล์โค้ด พังก์ชัน print_register_report (รายงานการลงทะเบียน)	61
ภาพที่ 6-1 โค้ดพังก์ชัน create_registration_record (สร้างบันทึกข้อมูลการลงทะเบียน)	62
ภาพที่ 6-2 โค้ดพังก์ชัน write_record_to_file (เขียนบันทึกข้อมูลลงในไฟล์)	63
ภาพที่ 6-3 โค้ดพังก์ชัน read_course_ids (อ่านรหัสวิชาทั้งหมดจากไฟล์)	64

สารบัญภาพ(ต่อ)

	หน้า
ขั้นตอนการทำงานของโปรแกรม	
ภาพที่ 1 หน้าหลักของโปรแกรม	65
ภาพที่ 2 เมนูหลักของหน้าจัดการข้อมูลนักเรียน	65
ภาพที่ 3 เมนูเพิ่มข้อมูลนักเรียน	65
ภาพที่ 4 เพิ่มข้อมูลนักเรียน	66
ภาพที่ 5 เมนูดูข้อมูลนักเรียนทั้งหมด	66
ภาพที่ 6 ตารางแสดงรายชื่อนักเรียนทั้งหมด	67
ภาพที่ 7 เมนูดูข้อมูลนักเรียนแบบรายบุคคล	67
ภาพที่ 8 แสดงข้อมูลนักเรียนรายบุคคล	68
ภาพที่ 9 เมนูดูข้อมูลนักเรียนแบบกรอง	68
ภาพที่ 10 เมนูกรองนักเรียนตามสาขา	69
ภาพที่ 11 เมนูกรองนักเรียนตามชั้นปี	69
ภาพที่ 12 เมนูกรองนักเรียนตามสถานะ	70
ภาพที่ 13 เมนูแก้ไขข้อมูลนักเรียน	70
ภาพที่ 14 เมนูลบข้อมูลนักเรียน	71
ภาพที่ 15 เมนูหลักของ จัดการข้อมูลรายวิชา	71
ภาพที่ 16 เมนูเพิ่มข้อมูลรายวิชา	72
ภาพที่ 17 เมนูดูข้อมูลรายวิชาทั้งหมด	72
ภาพที่ 18 เมนูดูข้อมูลรายวิชาตามรายวิชา	73
ภาพที่ 19 เมนูดูข้อมูลรายวิชาแบบกรอง	73
ภาพที่ 20 เมนูกรองรายวิชาตามปีการศึกษา	74
ภาพที่ 21 เมนูกรองรายวิชาตามภาคเรียน	74
ภาพที่ 22 เมนูกรองรายวิชาตามสถานะ	75
ภาพที่ 23 เมนูกรองรายวิชาตามปีการศึกษาและภาคเรียน	75
ภาพที่ 23 เมนูแก้ไขข้อมูลรายวิชา	76
ภาพที่ 24 แก้ไขข้อมูลรายวิชาตามรหัสวิชา	76
ภาพที่ 25 เมนูลบข้อมูลรายวิชา	77
ภาพที่ 26 ลบข้อมูลรายวิชาอย่างถาวร	77
ภาพที่ 27 เมนูจัดการข้อมูลการลงทะเบียน	78
ภาพที่ 28 เมนูเพิ่มข้อมูลการลงทะเบียน	78
ภาพที่ 29 เพิ่มข้อมูลการลงทะเบียน	78

สารบัญภาพ(ต่อ)

	หน้า
ภาพที่ 30 ดูข้อมูลการลงทะเบียนทั้งหมด	79
ภาพที่ 31 ดูข้อมูลการลงทะเบียนรายการเดียว	79
ภาพที่ 32 เมนูดูข้อมูลการลงทะเบียนแบบกรอง	79
ภาพที่ 33 เมนูดูข้อการลงทะเบียนกรองตามรหัสนักเรียน	80
ภาพที่ 34 เมนูดูข้อการลงทะเบียนกรองตามรหัสวิชา	80
ภาพที่ 35 เมนูดูข้อการลงทะเบียนกรองตามสถานะ	80
ภาพที่ 36 เมนูแก้ไขข้อมูลการลงทะเบียน	81
ภาพที่ 37 แก้ไขข้อมูลการลงทะเบียน	81
ภาพที่ 38 เมนูลบข้อมูลการลงทะเบียน	81
ภาพที่ 39 เมนูสร้างไฟล์รายงาน	82
ภาพที่ 40 ดูรายงานนักเรียน	82
ภาพที่ 41 ดูรายงานการลงทะเบียน	83

บทที่ 1

บทนำ

1.1 วัตถุประสงค์ของโครงการ

วัตถุประสงค์หลักของโครงการนี้คือ เพื่อพัฒนาระบบที่สามารถจัดการข้อมูลนักศึกษา รายวิชา และการลงทะเบียนเรียน ได้อย่างมีประสิทธิภาพ นอกจากนี้ยังมุ่งเน้น เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วยภาษา Python โดยเฉพาะการใช้งานจัดการกับไฟล์เบนารี (.bin) และเรียนรู้เทคนิคการจัดการและเข้มข้นข้อมูล ข้ามไฟล์ในโครงสร้างฐานข้อมูลเบื้องต้น สุดท้ายคือเพื่อพัฒนาทักษะการวิเคราะห์ข้อมูลเชิงสถิติและการทำงานร่วมกันเป็นทีม

1.2 ขอบเขตของโครงการ

ขอบเขตของระบบนี้จะประกอบด้วย 3 Modules หลัก ที่ทำงานสัมพันธ์กัน ได้แก่ Module Student, Module Course, และ Module Registration ซึ่งแต่ละ Module จะมีฟังก์ชันพื้นฐานครบถ้วน เช่น การเพิ่ม, การดู, การแก้ไข, และการลบข้อมูล (CRUD) รวมถึงมี Module การสร้างรายงาน (Reporting) โดยเฉพาะระบบจะมีการจัดเก็บข้อมูลในรูปแบบไฟล์เบนารี (.bin) จำนวน 3 ไฟล์ ได้แก่ student.bin, CourseSubject.bin, และ registration.bin

ข้อมูลที่จัดเก็บในระบบแบ่งเป็น 3 ส่วนหลัก คือ นักศึกษา (Student ID, ชื่อ-นามสกุล, สาขา, ชั้นปี, สถานะ), รายวิชา (Course ID, ชื่อวิชา, หน่วยกิต, ปีการศึกษา, ภาคเรียน, สถานะ), และ การลงทะเบียน (Register ID, Student ID, Course ID, วันที่ลงทะเบียน, สถานะลง/ถอน) โดยระบบจะมี เมนูหลัก และ เมนูย่อย ที่ชัดเจนเพื่อให้ผู้ใช้สามารถเลือกดำเนินการได้อย่างสะดวก

1.3 ประโยชน์ที่ได้รับ

โครงการนี้จะช่วยให้ได้ระบบที่สามารถจัดการเข้มข้น และวิเคราะห์สถิติ ข้อมูลการลงทะเบียนเรียนได้อย่างมีประสิทธิภาพสูง และเป็นการพัฒนาทักษะการเขียนโปรแกรม Python เชิงระบบ โดยเฉพาะการใช้ไลบรารี struct ในการจัดการข้อมูลขนาดใหญ่ รวมถึงการเรียนรู้เทคนิคการจัดการและเข้มข้นฐานข้อมูลเบื้องต้น ระหว่างไฟล์เบนารีหลายชุด ซึ่งรวมถึงการพัฒนาทักษะการสร้างรายงานเชิงวิเคราะห์ เช่น การคำนวณอัตราการถอน

1.4 เครื่องมือที่คาดว่าจะต้องใช้

เครื่องมือหลักที่ใช้คือ ภาษา Python พร้อมการใช้งานไลบรารีที่จำเป็น ได้แก่ struct, os, และ datetime และอาจใช้ Microsoft Office สำหรับการจัดทำเอกสารและการตรวจสอบข้อมูล

บทที่ 2

ระบบ ลงทะเบียนเรียน

โครงสร้างของไฟล์ สำหรับเก็บข้อมูล รายละเอียด ไฟล์เอกสาร ในกรากรอบแบบ เพิ่มข้อมูลของการเก็บข้อมูลระบบ
การลงทะเบียนเรียน ของนักศึกษา

สารบัญ เพิ่มข้อมูล โดยหลัก

- StudentData.bin (ข้อมูลนักเรียน)
- CourseSubject.bin (ข้อมูลวิชา)
- RegisterCourse.bin (ข้อมูลการลงทะเบียน)
- Report.txt (สรุปข้อมูล)

1. StudentData.bin (ข้อมูลนักเรียน)

ตารางนี้ใช้เก็บข้อมูลส่วนตัวของนักเรียน รวมถึงข้อมูลด้านการศึกษา

Field ฟิลด์	Type ประเภท	Size ขนาด	Example ตัวอย่าง	คำอธิบายและความสัมพันธ์
student_id	String	16 bytes	6801010100001	รหัสประจำตัวนักเรียน, Primary Key ของตารางนี้ ใช้ในการอ้างอิงกับตาราง RegisterCourse
first_name	String	50 bytes	กัณติกรณ์	ชื่อจริงของนักเรียน
last_name	String	50 bytes	สรสุริยวงศ์	นามสกุลของนักเรียน
major	String	10 bytes	CS	สาขาวิชาที่นักเรียนสังกัด เช่น "CS" สำหรับ Computer Science

year_level	Integer	1 byte	2	ชั้นปีของนักเรียน เช่น 1 = ปีหนึ่ง, 2 = ปีสอง
status	Integer	1 byte	1	สถานะของนักเรียน: 1 = Active (กำลังศึกษา), 0 = Inactive (ไม่ใช้งาน)

2. CourseSubject.bin (ข้อมูลวิชา)

ตารางนี้เก็บรายละเอียดของวิชาที่เปิดสอน โดยเพิ่มฟิลด์สำหรับปีการศึกษาและภาคเรียน

เพื่อให้สามารถแยกวิชาตามช่วงเวลาที่เปิดสอนได้ชัดเจน

Field ฟิลด์ข้อมูล	Type ประเภท	Size ขนาด	Example ตัวอย่าง	คำอธิบายและความสัมพันธ์
course_id	String	16 bytes	CS101-2568-1-1	รหัสวิชา, Primary Key ของตารางนี้ และถูกอ้างอิงในตาราง RegisterCourse เพื่อรับบุวิชาที่ลงทะเบียน
course_name	String	100 bytes	Computer Programming	ชื่อเต็มของวิชา
credit	Integer	1 byte	3	จำนวนหน่วยกิตของวิชา
academic_year	Integer	4 bytes	2568	ปีการศึกษาที่เปิดสอน
semester	Integer	1 byte	1	ภาคเรียนที่เปิดสอน (1, 2, 3)
is_active	Integer	1 byte	1	สถานะของวิชา: 1 = Active (เปิดสอน), 0 = Cancelled (ยกเลิก)

3. RegisterCourse.bin (ข้อมูลการลงทะเบียน)

ตารางนี้เป็นตารางเชื่อมโยงระหว่างข้อมูลนักเรียนและวิชา

Field	Type	Size	Example	คำอธิบายและความสัมพันธ์
register_id	Integer	4 bytes	5001	รหัสการลงทะเบียน, Primary Key ของตารางนี้
student_id	String	16 bytes	6801010100001	Foreign Key ที่อ้างอิงไปยัง student_id ในตาราง StudentData.bin เพื่อระบุตัวนักเรียน
course_id	String	16 bytes	CS101-2568-1-1	Foreign Key ที่อ้างอิงไปยัง course_id ในตาราง CourseSubject.bin เพื่อระบุวิชาและ Section
registration_date	Date/Time	8 bytes	2025-09-12	วันที่และเวลาที่นักเรียนทำการลงทะเบียนเรียน
status	Integer	1 byte	1	สถานะการลงทะเบียน: 1 = Registered (ลงทะเบียนแล้ว), 0 = Dropped (ยกเลิก)

4. โครงสร้างไฟล์ Report.txt (ฉบับสมบูรณ์)

รายงานฉบับนี้จะทำการสรุปข้อมูลการลงทะเบียนและสถานะของนักศึกษาในแต่ละ Section อย่างละเอียด

Student Registration Report Generated At: 2025-09-12 17:01:00 +07:00

COURSE: CS101 - Intro to Programming (Academic Year 2568, Semester 1)

SECTION: 1 -

STUDENT ID	FIRST NAME	LAST NAME	MAJOR	YEAR	REGISTRATION DATE	STATUS
6801010100001	สมชาย	ใจดี	CS	2	2025-09-08	Registered
6801010100005	นัฐพงษ์	รักเรียน	CS	2	2025-09-09	Registered
6801010100007	พัชราภรณ์	สุจิ	CE	1	2025-09-09	Registered
6801010100008	พิพัฒน์	รุ่งโรจน์	CE	2	2025-09-10	Registered
6801010100009	ศรีสุดา	อิมเมอม	CS	1	2025-09-11	Registered

Total students in this section:

5 - Students by Major:

Computer Science (CS): 3

Computer Engineering (CE): 2

Summary of Status:

- Registered: 5
- Dropped: 0

Summary

- Year with the most registrations: Year 2 (3 students)
- Major with the most registrations: CS (3 students)
- Major with the least registrations: CE (2 students)
- Date with the most registrations: 2025-09-09 (2 students)

โครงสร้างไฟล์ ของโปรแกรม

โฟลเดอร์ (Directories)

1. module * _ _pycache_ (โฟลเดอร์สำหรับเก็บไฟล์คอมไพล์ของ Python)

- ไฟล์ Python Modules (Modules หลักของระบบ)

2. ภายในได้ Folder module Module

- course.py (Module จัดการข้อมูลรายวิชา)
- register.py (Module จัดการข้อมูลการลงทะเบียน)
- report.py (Module สำหรับสร้างรายงานและวิเคราะห์สถิติ)
- sample_generate.py (Module สำหรับสร้างข้อมูลตัวอย่าง/ทดสอบ)
- student.py (Module จัดการข้อมูลนักศึกษา)

3. ไฟล์ Main หลัก ในการทำงานของโปรแกรม ปฏิบัติการและเอกสาร (Main Execution & Documentation)

- main.py (ไฟล์โปรแกรมหลักสำหรับเรียกใช้งานเมนูทั้งหมด)
- test.py (ไฟล์สำหรับทดสอบฟังก์ชันต่างๆ)
- README.md (ไฟล์เอกสาร/คำอธิบายโครงการ)

4.ไฟล์ฐานข้อมูลในรูปแบบ Binary (Binary Database Files)

- CourseSubject.bin (ไฟล์เก็บข้อมูลรายวิชา)
- registration.bin (ไฟล์เก็บข้อมูลการลงทะเบียน)
- student.bin (ไฟล์เก็บข้อมูลนักศึกษา)

5. ไฟล์รายงาน (Generated Report Files)

- register_report.txt (ไฟล์รายงานผลการลงทะเบียน)
- student_report.txt (ไฟล์รายงานผลข้อมูลนักศึกษา)

ขอบเขตการทำงาน ของโปรแกรม

1. ขอบเขตการทำงานของไฟล์ main.py

ไฟล์ main.py ทำหน้าที่เป็น ศูนย์กลางการควบคุม (Control Center) และ จุดเริ่มต้น (Entry Point) ของโปรแกรมทั้งหมด โดยมีข้อบอกร่างดังนี้:

1.1 จุดเริ่มต้นการทำงานของโปรแกรม

- เป็นจุดเริ่มต้น: กำหนดให้ฟังก์ชัน main_menu()
เป็นฟังก์ชันแรกที่ถูกเรียกใช้งานเมื่อเปิดโปรแกรม (if __name__ == "__main__":)

1.2 การจัดการส่วนประกอบและโครงสร้าง (Module Management)

- การนำเข้า Module: ทำหน้าที่นำเข้า (Import) ส่วนประกอบทั้งหมดของระบบเข้ามาใช้งาน ไม่ว่าจะเป็น Module มาตรฐาน (struct, os) และ Module หลักของระบบ ได้แก่:
 - student
 - register
 - course
 - report

1.3 การแสดงผลและจัดการเมนูหลัก (Main Menu Handling)

- แสดงเมนูหลัก: แสดงรายการตัวเลือกหลักของระบบ ("ระบบจัดการข้อมูลลงทะเบียนเรียน") ให้ผู้ใช้งานเห็น
- รับและตรวจสอบคำสั่ง: รับค่าตัวเลือก (Choice) จากผู้ใช้งาน และตรวจสอบว่าตัวเลือกนั้นถูกต้องหรือไม่

1.4 การควบคุมการไหลของโปรแกรม (Flow Control)

- นำทางไปยังระบบอย่าง: ทำหน้าที่เป็นตัวเชื่อมโยง (Router) นำผู้ใช้งานเข้าสู่ระบบอย่างต่าง ๆ

ตามตัวเลือกที่เลือก:

- เลือก 1 เพื่อเข้าสู่ระบบจัดการข้อมูลนักเรียน (student_menu())
- เลือก 2 เพื่อเข้าสู่ระบบจัดการข้อมูลรายวิชา (course_menu())
- เลือก 3 เพื่อเข้าสู่ระบบจัดการข้อมูลการลงทะเบียน (registration_menu())
- เลือก 4 เพื่อเข้าสู่ระบบสร้างไฟล์รายงาน (generate_report())

1.5 การสิ้นสุดการทำงาน (Program Termination)

- การออกจากโปรแกรม: จัดการคำสั่ง '0'

เพื่อจบการทำงานของโปรแกรมอย่างถูกต้องและออกจากลูป while True

2. ขอบเขตการทำงานของ Module: การจัดการข้อมูลนักเรียน (student.py)

Module นี้มีขอบเขตการทำงานหลักคือ การจัดการข้อมูลนักเรียนแบบครบวงจร (CRUD) และการจัดการไฟล์ข้อมูลแบบไบนาเรีย โดยแบ่งเป็น 3 ส่วนหลัก ดังนี้:

2.1 การจัดการโครงสร้างข้อมูลและการอ่าน/เขียนไฟล์ (Data & File Handling)

- โครงสร้างข้อมูล: กำหนดรูปแบบและขนาดของบันทึกข้อมูลนักเรียน (Record) ด้วย Module struct โดยมีรายละเอียดดังนี้:
 - รหัสนักเรียน (16 อักขระ)
 - ชื่อจริง (50 อักขระ)
 - นามสกุล (50 อักขระ)
 - สาขาวิชา (20 อักขระ)
 - ชั้นปี (เลขจำนวนเต็มขนาด 1 ไปต์)
 - สถานะ (เลขจำนวนเต็มขนาด 1 ไปต์: 1=Active, 0=Inactive)

การจัดการไฟล์: จัดการพารของไฟล์ข้อมูล (ตั้งชื่อไฟล์ว่า `student.bin`) โดยใช้ Module `os`

และมีฟังก์ชันหลักสามทั่วไป:

- สร้าง บันทึกข้อมูลนักเรียนในรูปแบบไบนาเรี่ย (create_student_record)
- แปลง ข้อมูลไบนาเริกลับมาเป็นข้อมูลที่มนุษย์อ่านได้ (read_student_record)
- เขียน ข้อมูลบันทึกใหม่ลงท้ายไฟล์ (write_record_to_file)
- อ่าน ข้อมูลบันทึกทั้งหมดจากไฟล์ (read_all_records_from_file)

2.2 ฟังก์ชันการจัดการข้อมูลหลัก (CRUD Operations)

Module นี้รองรับการดำเนินการพื้นฐานกับข้อมูลนักเรียน

การดำเนินการ	ฟังก์ชันที่เกี่ยวข้อง	ขอบเขตหน้าที่
เพิ่ม (Create)	<code>add_student()</code>	รับข้อมูลนักเรียนใหม่จากผู้ใช้ ตรวจสอบความถูกต้องเบื้องต้นแล้วบันทึกเป็น Record ใหม่ลงในไฟล์ <code>student.bin</code>
แก้ไข (Update)	<code>update_student()</code>	ค้นหาบันทึกนักเรียนด้วยรหัสที่ระบุ อนุญาตให้ผู้ใช้แก้ไขข้อมูลในไฟล์ต่อๆ จากนั้นเขียนไฟล์ข้อมูลทั้งหมดใหม่ด้วยข้อมูลที่แก้ไขแล้ว
ลบ (Delete)	<code>delete_student()</code>	ค้นหาบันทึกนักเรียนด้วยรหัสที่ระบุ ลบบันทึกนั้นออกจากรายการแล้วเขียนไฟล์ข้อมูลทั้งหมดใหม่โดยไม่รวมบันทึกที่ถูกลบ

2.3 การค้นหาและการเรียกคุณูป (View & Search)

Module นี้มีฟังก์ชันเพื่อเรียกคุณูปในรูปแบบต่าง ๆ:

- **ดูข้อมูลทั้งหมด:** แสดงรายงานข้อมูลนักเรียนทั้งหมดในระบบ (view_students())
- **ค้นหารายบุคคล:** ค้นหาและแสดงข้อมูลนักเรียนที่ลูกค้าคนด้วยรหัสประจำตัว (view_single_student())
- **ค้นหาแบบมีเงื่อนไข (Filter):**
อนุญาตให้ผู้ใช้กรองข้อมูลและแสดงรายงานเฉพาะนักเรียนที่ตรงตามเงื่อนไข ดังนี้:
 - กรองตาม สาขาวิชา
 - กรองตาม ชั้นปี
 - กรองตาม สถานะ (Active/Inactive)
- **แสดงรายงาน:** จัดรูปแบบข้อมูลที่ค้นหาได้ให้อยู่ในรูปแบบ ตาราง ที่อ่านง่าย (print_student_report)

2.4 เมนูย่อย (Control Flow)

- **student_menu():** หน้าที่เป็นเมนูย่อยให้ผู้ใช้เลือกดำเนินการในขอบเขตงานทั้งหมดข้างต้น และอนุญาตให้ กลับสู่เมนูหลัก ของโปรแกรมได้

3. ขอบเขตการทำงานของ Module: การจัดการข้อมูลการลงทะเบียน

Module นี้รับผิดชอบการจัดการข้อมูลการลงทะเบียนเรียน (Registration) โดยเฉพาะ
ซึ่งรวมถึงการเข้ามายังกับข้อมูลนักเรียน (Student) และการจัดการไฟล์ใบหนารีสำหรับบันทึกการลงทะเบียน

3.1 การจัดการโครงสร้างข้อมูลและการจัดการไฟล์ (Data & File Handling)

- โครงสร้างข้อมูล: กำหนดรูปแบบและขนาดบันทึกด้วย struct สำหรับไฟล์ registration.bin ซึ่งประกอบด้วย:
 - ID การลงทะเบียน (Register ID)
 - รหัสนักเรียน (Student ID)
 - รหัสวิชา (Course ID)
 - วันที่/เวลาลงทะเบียน (Registration Date)
 - สถานะการลงทะเบียน (Status: Registered / Dropped)
- การแปลงข้อมูล: มีฟังก์ชันสำหรับ สร้าง บันทึกใบหนารี (create_registration_record) และ อ่าน ข้อมูลกลับมาเป็นรูปแบบที่ใช้งานได้ (read_registration_record)
- การจัดการไฟล์: จัดการไฟล์ registration.bin และมีฟังก์ชันหลักในการ เขียนต่อท้าย (write_record_to_file) และ อ่านข้อมูลทั้งหมด (read_all_records_from_file)

3.2 การจัดการข้อมูลหลัก (Core Operations - CRUD)

Module นี้รองรับการดำเนินการพื้นฐานกับรายการลงทะเบียน:

3.2.1 เพิ่มการลงทะเบียน (add_registration):

- มีการ ตรวจสอบข้อมูลนักเรียน ในไฟล์ student.bin ก่อน (เรียกว่า read_student_by_id)
- ตรวจสอบว่านักเรียนมีสถานะ Active จึงจะอนุญาตให้ลงทะเบียน
- สร้าง ID การลงทะเบียนถัดไปแบบอัตโนมัติ (get_next_register_id)
และบันทึกเวลาปัจจุบัน

3.2.2 แก้ไขการลงทะเบียน (update_registration):

- ค้นหารายการลงทะเบียนด้วย ID
- อนุญาตให้ผู้ใช้ แก้ไขเฉพาะสถานะ (เปลี่ยนเป็น Registered หรือ Dropped)
- ทำการ เขียนไฟล์ข้อมูลทั้งหมดใหม่ หลังจากแก้ไข

3.2.3 ลบการลงทะเบียน (delete_registration):

- ค้นหารายการด้วย ID
- ทำการ ลบรายการ นับอุบัติการณ์จากการลงทะเบียนทั้งหมด
- ทำการ เขียนไฟล์ข้อมูลทั้งหมดใหม่ โดยไม่รวมรายการที่ถูกลบ

3.3 การค้นหาและการเรียกดูข้อมูล (View & Search)

Module นี้สามารถเรียกดูข้อมูลในหลายรูปแบบ โดยใช้ฟังก์ชัน print_registration_report

ในการแสดงผลแบบตาราง:

- ดูทั้งหมด (view_registrations): แสดงรายการการลงทะเบียนทั้งหมดในระบบ
- ดูรายการเดียว (view_single_registration): ค้นหาและแสดงผลรายการลงทะเบียนด้วย รหัส ID
- ค้นหาแบบกรอง (view_filtered_registrations):

อนุญาตให้ผู้ใช้กรองและแสดงผลรายงานตามเงื่อนไข ดังนี้:

- กรองตาม รหัสนักเรียน
- กรองตาม รหัสวิชา
- กรองตาม สถานะ (Registered)

3.4 การพึ่งพาข้อมูล (Data Dependency)

- มีการ พึ่งพา ไฟล์ข้อมูล student.bin

เพื่อใช้ตรวจสอบความถูกต้องและสถานะของนักเรียนก่อนที่จะอนุญาตให้ทำการลงทะเบียนเรียนได้ (`get_student_info_for_registration`)

3.5 เมนูย่อย (Control Flow)

- `registration_menu()`:

ทำหน้าที่เป็นเมนูย่อยเพื่อเข้าถึงฟังก์ชันการจัดการข้อมูลการลงทะเบียนทั้งหมด และอนุญาตให้กลับสู่เมนูหลัก ของโปรแกรมได้

4. ขอบเขตการทำงานของ Module: การจัดการข้อมูลรายวิชา Module

นี้มีหน้าที่หลักในการดำเนินการพื้นฐาน (CRUD) และการจัดการไฟล์ในารีสำหรับข้อมูลรายวิชาทั้งหมด

4.1 การจัดการโครงสร้างข้อมูลและไฟล์ (Data Structure & File Management)

- โครงสร้างข้อมูลหลัก: กำหนดรูปแบบบันทึกข้อมูลรายวิชาด้วย struct ในไฟล์ `CourseSubject.bin` ซึ่งประกอบด้วย:

- รหัสวิชา (COURSE ID): ไม่เกิน 10 ตัวอักษร
- ชื่อวิชา (COURSE NAME): ไม่เกิน 50 ตัวอักษร
- หน่วยกิต (CREDIT)
- ปีการศึกษา (ACADEMIC YEAR)
- ภาคเรียน (SEMESTER)
- สถานะ (STATUS): กำหนดเป็น 1 (Active) หรือ 0 (Inactive)
- การแปลงข้อมูล: มีฟังก์ชันสำหรับแปลงข้อมูลจาก Python Object

ให้เป็นรูปแบบใบารีที่จัดเก็บได้ (`create_course_record`)

และแปลงกลับจากใบารีเป็น Dictionary ที่อ่านได้ (`read_course_record`)

- การจัดการไฟล์: มีฟังก์ชันสำหรับ อ่านข้อมูลทั้งหมด จากไฟล์ (`read_all_records_from_file`) และ เขียนต่อท้าย ข้อมูลใบารีลงในไฟล์ (`write_record_to_file`)

4.2 การดำเนินการหลัก (CRUD) Module นี้รองรับการจัดการข้อมูลรายวิชาอย่างครบถ้วน:

4.2.1 เพิ่มรายวิชา (add_course):

- รับข้อมูลรายละเอียดรายวิชาจากผู้ใช้ (รหัส, ชื่อ, หน่วยกิต, ปี, ภาคเรียน, สถานะ).
- บรรจุข้อมูลในรูปแบบไบนาเรียและเขียนลงในไฟล์ CourseSubject.bin

4.2.2 แก้ไขรายวิชา (update_course):

- ให้ผู้ใช้ป้อน รหัสวิชา ที่ต้องการแก้ไข.
- แสดงข้อมูลเดิมและอนุญาตให้แก้ไขฟิลด์ต่างๆ (ชื่อ, หน่วยกิต, ปีการศึกษา, ภาคเรียน, สถานะ).
- เมื่อแก้ไขเสร็จ จะทำการ เขียนไฟล์ข้อมูลทั้งหมดใหม่ (Overwrite)
ด้วยข้อมูลที่อัปเดตแล้ว

4.2.3 ลบรายวิชา (delete_course):

- ให้ผู้ใช้ป้อน รหัสวิชา ที่ต้องการลบ.
- ค้นหาและข้ามการบันทึกข้อมูลของรายวิชาดังกล่าว.
- ทำการ เขียนไฟล์ข้อมูลทั้งหมดใหม่ (Overwrite) โดยตัดรายการที่ถูกลบออกจาก

4.3 การแสดงรายงานและการกรองข้อมูล (Reporting, Viewing, and Filtering)

- แสดงทั้งหมด (view_all_courses):

อ่านและแสดงผลรายการรายวิชาทั้งหมดในรูปแบบตารางที่จัดโครงสร้างอย่างสวยงาม (print_course_report).

- แสดงรายการเดียว (view_single_course):

ค้นหารายวิชาด้วยรหัสวิชาและแสดงผลในรูปแบบรายการรายละเอียด.

- การกรองข้อมูล (view_filtered_courses): มีเมนูย่อยให้เลือกกรองข้อมูลตามเงื่อนไขเฉพาะ ได้แก่:

- กรองตาม ปีการศึกษา.
- กรองตาม ภาคเรียน.
- กรองตาม สถานะ Active เท่านั้น.
- กรองตาม ปีการศึกษาและภาคเรียน พื้นฐาน.

4.4 ส่วนต่อประสานกับผู้ใช้ (User Interface)

- `course_menu()`: ทำหน้าที่เป็นเมนูหลักของระบบจัดการรายวิชา โดยรับตัวเลือกจากผู้ใช้ (1-6 สำหรับ CRUD และ 0 เพื่อออกจากเมนู) และเรียกใช้ฟังก์ชันที่เกี่ยวข้องตามตัวเลือกที่ได้รับ

Module

นี่จึงเป็นระบบย่อยที่สมบูรณ์แบบสำหรับการจัดการข้อมูลหลักของรายวิชาในระบบการศึกษาโดยใช้การจัดเก็บข้อมูลแบบเป็นนารีเป็นหลัก.

5. ขอบเขตการทำงานของ Module: การสร้างรายงานและการวิเคราะห์ Module

นี่ถือเป็นส่วนสำคัญในการวิเคราะห์ข้อมูลของระบบโดยเฉพาะข้อมูลการลงทะเบียนซึ่งมีความสามารถในการเชื่อมโยงข้อมูลระหว่าง นักศึกษา (Student), รายวิชา (Course), และ การลงทะเบียน (Registration) เข้าด้วยกันอย่างครบถ้วน

5.1 การเชื่อมโยงข้อมูลและแหล่งที่มา (Data Integration)

Module นี้มีฟังก์ชันสำหรับอ่านและประมวลผลข้อมูลจากไฟล์ใบนำรีหลักทั้งหมด:

- นักศึกษา (`read_all_students`): อ่านข้อมูลนักศึกษาทั้งหมดเพื่อใช้ในการแสดงรายละเอียดและสถิติ (เช่น สาขาและชั้นปี).
- รายวิชา (`load_course_dict`): อ่านข้อมูลรายวิชาทั้งหมดและจัดเก็บในรูปแบบ Dictionary เพื่อใช้ค้นหา ชื่อวิชา และ หน่วยกิต จาก รหัสวิชา ในรายงานการลงทะเบียน.
- การลงทะเบียน (`read_all_registrations`): อ่านบันทึกการลงทะเบียนทั้งหมด ซึ่งรวมถึง รหัสลงทะเบียน, รหัสนักศึกษา, รหัสวิชา, วันที่ และสถานะการลงทะเบียน/ถอน.

5.2 รายงานประเภทที่ 1: รายงานนักศึกษา (print_student_report)

รายงานนี้จะแสดงผลข้อมูลพื้นฐานของนักศึกษาพร้อมการสรุปสถิติเชิงปริมาณ:

- **รายละเอียด:** แสดงรายชื่อนักศึกษาทั้งหมดในรูปแบบตาราง (ID, ชื่อ, สาขา, ชั้นปี, สถานะ).
- **สถิติเชิงปริมาณ:**
 - **จำนวนรวม:** นับจำนวนนักศึกษาทั้งหมด.
 - **แยกตามสาขา:** สรุปจำนวนนักศึกษาตาม สาขาวิชาหลัก (MAJOR).
 - **แยกตามชั้นปี:** สรุปจำนวนนักศึกษาตาม ชั้นปี (YEAR)
และระบุชั้นปีที่มีจำนวนนักศึกษามากที่สุด.
- **ผลลัพธ์:** พิมพ์ออกทางหน้าจอและบันทึกลงไฟล์ `student_report.txt`

5.3 รายงานประเภทที่ 2: รายงานการลงทะเบียนและการวิเคราะห์สถิติ (Core Analysis)

นี่คือส่วนที่ซับซ้อนและมีคุณค่าที่สุดของModuleซึ่งรวมเอาข้อมูลทั้งหมดเข้าด้วยกันเพื่อสร้างการวิเคราะห์เชิงลึก: รายงานจะจัดกลุ่มการลงทะเบียนตามรายวิชาและแสดงรายละเอียดนักศึกษาที่ลงทะเบียนในแต่ละวิชาจากนั้นทำการวิเคราะห์สถิติทันทีภายใต้ฟังก์ชัน `analyze_registration_statistics`

หมวดการวิเคราะห์	รายละเอียด
วิชายอดนิยม (Popularity)	จัดอันดับ 10 อันดับแรก ของวิชาตามจำนวนผู้ลงทะเบียน (registered) มากที่สุด
อัตราการถอน (Drop Rate)	จัดอันดับ 5 อันดับแรก ของวิชาที่มีอัตราการถอนสูงที่สุด สูตรคำนวณ: $(\text{Drop Rate}) = \frac{\text{จำนวนถอน}}{\text{จำนวนลงทะเบียน}} + \frac{\text{จำนวนถอน}}{\text{จำนวนถอน}}$
ตามสาขาและชั้นปี	แสดงสถิติการลงทะเบียนและการถอน แยกตาม สาขา และ ชั้นปี พร้อมคำนวณอัตราการถอนของแต่ละกลุ่ม
การวิเคราะห์ตามเวลา	ระบุ วันที่มีการลงทะเบียนมากที่สุด 5 อันดับแรก
สรุปภาพรวม	แสดง: จำนวนการลงทะเบียนรวม, จำนวนการถอนรวม, อัตราการถอนโดยรวม, จำนวนนักศึกษาที่ไม่เข้ากันที่ลงทะเบียน

5.4 ผลลัพธ์ ขอบเขต รายงาน ของโปรแกรม

- รายงานฉบับเต็มจะถูกพิมพ์ออกทางหน้าจอ.
- รายงานฉบับเต็มจะถูกบันทึกลงในไฟล์ register_report.txt เพื่อใช้ในการเก็บข้อมูลหรือนำเสนอ.

Module นี้ได้ทำหน้าที่ได้อย่างครบถ้วนในฐานะศูนย์กลางการรายงานของระบบการจัดการข้อมูล โดยเฉพาะการดึงข้อมูลจากหลายแหล่งมาผนวกกันเพื่อวิเคราะห์สถิติการลงทะเบียนอย่างละเอียด

6. ขอบเขตการทำงานและความสอดคล้องของ Binary File ทั้ง 3

ไฟล์ฐานข้อมูลใบหนารีเหล่านี้ทำหน้าที่เป็นพื้นที่เก็บข้อมูลถาวร(PersistentStorage)โดยถูกออกแบบมาให้มีโครงสร้างที่แน่นอน เพื่อให้โค้ด Python สามารถอ่านและเขียนข้อมูลได้อย่างรวดเร็วผ่านไลบรารี struct

6.1 student.bin (ฐานข้อมูลนักศึกษา)

ขอบเขตการทำงานของไฟล์	ความสอดคล้องกับ Module (ตามโค้ดที่เคยให้)
โครงสร้างข้อมูล	เก็บข้อมูลของนักศึกษาแต่ละคน
การจัดการ	ถูกจัดการโดย Module Student Management (ในโค้ดที่ไม่ได้ให้มาทั้งหมด แต่ถูกอ้างถึงในฟังก์ชัน read_student_by_id)
การนำไปใช้	1. ใช้ในการ ตรวจสอบสิทธิ์ ก่อนการลงทะเบียน (ต้องเป็นนักเรียน Active) 2. ใช้ในการ สร้างรายงาน เพื่อดึงข้อมูล ชื่อ, สาขา, และชั้นปี มาประกอบในรายงานการลงทะเบียน

6.2 CourseSubject.bin (ฐานข้อมูลรายวิชา)

ขอบเขตการทำงานของไฟล์	ความสอดคล้องกับ Module (ตามโค้ดที่เคยให้)
โครงสร้างข้อมูล	เก็บรายละเอียดรายวิชา
การจัดการ	ถูกจัดการโดย Module Course Subject Management (อ้างอิงจากโค้ดในคำานวณที่สอง)
การนำไปใช้	1. ใช้เป็น ตารางอ้างอิง เพื่อดึงข้อมูล ชื่อวิชา, ปีการศึกษา, ภาคเรียน มาแสดงในรายงานการลงทะเบียน 2. ใช้ในการดำเนินการ CRUD ของข้อมูลวิชาโดยตรง

6.3 registration.bin (ฐานข้อมูลการลงทะเบียน)

ขอบเขตการทำงานของไฟล์	ความสอดคล้องกับ Module (ตามโค้ดที่เคยให้)
โครงสร้างข้อมูล	เก็บบันทึกการเชื่อมโยงระหว่างนักศึกษาและรายวิชา
การจัดการ	ถูกจัดการโดย Module Registration Management (อ้างอิงจากโค้ดในคำานวณแรก)
การนำไปใช้	1. เป็น หัวใจหลัก สำหรับการบันทึกประวัติการเรียน 2. เป็น แหล่งข้อมูลหลัก สำหรับ Module Report Generation ในการวิเคราะห์ผลติ เช่น อัตราการถอน และวิชาอยอดนิยม

6.4 ทั้งสามไฟล์ทำงานร่วมกันผ่านรหัสที่ถูกแชร์ระหว่างไฟล์:

1. การลงทะเบียน (CRUD): Module Registration Management อ่าน student.bin เพื่อ
ยืนยันตัวตน และเขียนบันทึกใหม่ลงใน registration.bin โดยใช้ รหัสนักเรียน และ รหัสวิชา
เป็นกุญแจสำคัญ
2. การรายงาน (Reporting): Module Report Generation อ่านไฟล์ registration.bin เป็นหลัก
จากนั้นใช้ รหัสนักเรียน ที่อยู่ในบันทึกการลงทะเบียนไปค้นหาชื่อนักเรียนใน student.bin และใช้
รหัสวิชา ไปค้นหาชื่อวิชาและข้อมูลภาคเรียนใน CourseSubject.bin
เพื่อรวมเป็นรายงานที่สมบูรณ์และมีการวิเคราะห์

อธิบายโค้ดการทำงานของโปรแกรม

1.ไฟล์ main.py

ทำหน้าที่เป็น ศูนย์กลางการควบคุม (Controller) และ แผนผังนำทางหลัก (Main Menu) ของระบบ โดยไม่ได้ทำหน้าที่ประมวลผลข้อมูลใด ๆ เองเลย แต่จะอาศัยการทำงานจากไฟล์อยู่ในโฟลเดอร์ module/เท่านั้น

- การนำเข้า (Imports):** นำเข้าฟังก์ชันเมนูหลักทั้งหมด (student_menu, course_menu, registration_menu, generate_report) จาก Module อย่างทั่วไป มาเตรียมพร้อมใช้งาน
- ฟังก์ชัน main_menu:** ทำหน้าที่เป็น Router (ตัวนำทาง) แสดงตัวเลือกให้ผู้ใช้เห็น และเมื่อผู้ใช้เลือกตัวเลข (1-4) ระบบจะ เรียกใช้ฟังก์ชันเมนูย่อย ที่สอดคล้องกับทันที เพื่อส่งต่อการควบคุมไปยัง Module นั้น ๆ
- การเริ่มโปรแกรม:** โค้ดจะ เรียกใช้ main_menu() เป็นอันดับแรก เมื่อรันไฟล์ เพื่อให้ระบบเริ่มทำงานและแสดงเมนูหลักแก่ผู้ใช้ทันทีก่อนโดยสรุป: main.py คือ ด้านหน้า ที่รับคำสั่งจากผู้ใช้และ ส่งต่อคำสั่ง ไปยัง Module ที่รับผิดชอบในการจัดการข้อมูลแต่ละส่วน

```

● ● ●
1 import struct
2 import os
3 from module.student import *
4 from module.register import *
5 from module.course import *
6 from module.report import *
7
8 def main_menu():
9     """เมนูหลักของโปรแกรม"""
10    while True:
11        print("\n===== ระบบจัดการข้อมูลลงทะเบียน =====")
12        print("1. จัดการข้อมูลนักเรียน")
13        print("2. จัดการข้อมูลรายวิชา")
14        print("3. จัดการข้อมูลการลงทะเบียน")
15        print("4. สร้างไฟล์รายงาน")
16        print("0. ออกจากโปรแกรม")
17
18        choice = input("กรุณาเลือกเมนูหลัก: ")
19
20        if choice == '1':
21            student_menu()
22        elif choice == '2':
23            course_menu()
24        elif choice == '3':
25            registration_menu()
26        elif choice == '4':
27            generate_report()
28        elif choice == '0':
29            print("ออกจากโปรแกรม...")
30            break
31        else:
32            print("ตัวเลือกไม่ถูกต้อง กรุณาลองใหม่อีกครั้ง")
33
34 if __name__ == "__main__":
35     main_menu()

```

ภาพที่ 1-1 โค้ดส่วนการทำงานหลักของโปรแกรม (main.py)

2.ไฟล์ student.py

การกำหนดโครงสร้าง (Configuration)

- กำหนดให้ไฟล์ฐานข้อมูลชื่อ student.bin และระบุตำแหน่งไฟล์ให้บันทึกในไฟล์เดอร์หลัก
- กำหนดรูปแบบการจัดเก็บข้อมูลด้วย STUDENT_RECORD_FORMAT (<16s50s50s20sB B) ซึ่งเป็นพิมพ์เขียว (Blueprint) ที่ระบุว่าข้อมูลแต่ละส่วน (เช่น รหัส, ชื่อ, สาขา) ต้องมีขนาดเท่าใดและเป็นชนิดใด

2.1 พังก์ชัน create_student_record (หัวใจสำคัญ) พังก์ชันนี้ทำหน้าที่หลักในการ แปลงข้อมูล โดยมีขั้นตอนดังนี้:

การเข้ารหัส (Encoding): แปลงข้อมูลที่เป็นข้อความ (String) เช่น ชื่อและรหัสนักเรียน ให้เป็นข้อมูลไบต์ (Bytes).

การปรับขนาด (Padding): ตรวจสอบให้แน่ใจว่าข้อความที่เข้ารหัสมีขนาดตรงตามที่กำหนดไว้ในรูปแบบ (เช่น รหัสต้องมี 16 ไบต์, ชื่อต้องมี 50 ไบต์) โดยจะ ตัดส่วนที่เกินออก การแพ็ค (Packing): ใช้ struct.pack() นำข้อมูลไบต์และตัวเลขที่ปรับขนาดแล้วทั้งหมดมา รวมเป็นก้อนเดียว (Record) ตามลำดับที่กำหนดไว้ใน STUDENT_RECORD_FORMAT

ผลลัพธ์: ส่งกลับข้อมูลนักเรียนหนึ่งคนในรูปแบบ ไบนาเรียริสุทธิ์ ซึ่งพร้อมสำหรับเขียนลงในไฟล์ student.bin

```

● ● ●
1 # ชื่อไฟล์สำหรับจัดเก็บข้อมูลนักเรียน
2 STUDENT_FILE_NAME = 'student.bin'
3
4 # กำหนดพารามิเตอร์สำหรับไฟล์บันทึกในไฟล์เดอร์หลัก (main)
5 current_dir = os.path.dirname(os.path.abspath(__file__))
6 main_dir = os.path.dirname(current_dir)
7 STUDENT_FILE_PATH = os.path.join(main_dir, STUDENT_FILE_NAME)
8
9 # รูปแบบของข้อมูลที่จะจัดเก็บในrecord
10 STUDENT_RECORD_FORMAT = '<16s50s50s20sB B'
11 STUDENT_RECORD_SIZE = struct.calcsize(STUDENT_RECORD_FORMAT)
12
13 def create_student_record(student_id, first_name, last_name, major, year_level, status):
14     """สร้างบันทึกข้อมูลนักเรียนในรูปแบบในนี้"""
15     packed_student_id = student_id.encode('utf-8')[::16].ljust(16, b'\x00')
16     packed_first_name = first_name.encode('utf-8')[::50].ljust(50, b'\x00')
17     packed_last_name = last_name.encode('utf-8')[::50].ljust(50, b'\x00')
18     packed_major = major.encode('utf-8')[::20].ljust(20, b'\x00')
19     try:
20         record = struct.pack(
21             STUDENT_RECORD_FORMAT,
22             packed_student_id,
23             packed_first_name,
24             packed_last_name,
25             packed_major,
26             year_level,
27             status
28         )
29         return record
30     except struct.error as e:
31         print(f"เกิดข้อผิดพลาดในการแพ็คข้อมูล: {e}")
32     return None

```

ภาพที่ 2-1 "โค้ดส่วนการแปลงข้อมูลนักศึกษาเป็นรูปแบบไบนาเรีย (Packing)

2.2 ไฟล์ [student.py](#) พังก์ชัน `read_student_record` (อ่านข้อมูล) พังก์ชันนี้ทำหน้าที่ แปลงบันทึกไบนาเรีย (Binary Record) ที่ได้รับเข้ามา ให้กลับเป็น ข้อมูลที่มุซย์อ่านได้ (Dictionary Python)

- หลักการทำงาน: ใช้ `struct.unpack()` เพื่อแยกข้อมูลตามโครงสร้างที่กำหนด (Format)
- การแปลง: ลบค่าว่างที่เติมไว้ (`.strip(b'\x00')`) และ ถอดรหัส (`.decode('utf-8')`)
ข้อมูลที่เป็นข้อความให้กลับมาเป็น String.
- ผลลัพธ์: คืนค่าเป็น Dictionary ที่มีคีย์เป็นชื่อฟิลด์และค่าที่อ่านได้ รวมถึงแปลงรหัสสถานะ (1/0)
เป็นข้อความ (Active/Inactive).

2.3 พังก์ชัน `write_record_to_file` (เขียนข้อมูล)

พังก์ชันนี้ทำหน้าที่ บันทึกบันทึกไบนาเรีย ที่ถูกเตรียมไว้แล้วลงในไฟล์ฐานข้อมูล

- หลักการทำงาน: เปิดไฟล์ `student.bin` ในโหมด 'ab' (Append Binary)
- การบันทึก: ใช้ `f.write(record)` เพื่อ เพิ่ม (Append) บันทึกไบนาเรียใหม่ต่อท้ายไฟล์ที่มีอยู่เดิม

```

● ● ●
1 def read_student_record(record_data):
2     """อ่านบันทึกข้อมูลนักเรียนจากรูปแบบไบนาเรีย"""
3     try:
4         unpacked_data = struct.unpack(STUDENT_RECORD_FORMAT, record_data)
5         student_id = unpacked_data[0].strip(b'\x00').decode('utf-8')
6         first_name = unpacked_data[1].strip(b'\x00').decode('utf-8')
7         last_name = unpacked_data[2].strip(b'\x00').decode('utf-8')
8         major = unpacked_data[3].strip(b'\x00').decode('utf-8')
9         return {
10             'STUDENT ID': student_id,
11             'FIRST NAME': first_name,
12             'LAST NAME': last_name,
13             'MAJOR': major,
14             'YEAR': unpacked_data[4],
15             'STATUS': 'Active' if unpacked_data[5] == 1 else 'Inactive'
16         }
17     except struct.error as e:
18         print(f"เกิดข้อผิดพลาดในการอ่านแพ็คข้อมูล: {e}")
19

```

ภาพที่ 2-2 โค้ดไฟล์ `student.py` พังก์ชัน `read_student_record`

2.4 ไฟล์ [student.py](#) พังก์ชัน read_all_records_from_file (อ่านบันทึกทั้งหมด)

พังก์ชันนี้ ดึงข้อมูลนักศึกษาทั้งหมด จากไฟล์ student.bin โดยการวนอ่านข้อมูลทีลับล็อก (ตามขนาด Record) จากนั้นส่งแต่ละบล็อกไป แกะรหัสเบนารี เป็น Dictionary ก่อนจะรวมไว้ในลิสต์ทั้งหมด.

2.5 พังก์ชัน print_student_report (แสดงรายงาน)

พังก์ชันนี้ นำลิสต์ข้อมูลนักศึกษา มาจัดรูปแบบเป็น ตารางรายงานที่สวยงาม พร้อมหัวข้อและเส้นแบ่ง โดยมีการจัดความกว้างของคอลัมน์ให้ตรงกัน และพิมพ์รายงานออกทางหน้าจอ

```

● ● ●
1 def read_all_records_from_file(file_path=STUDENT_FILE_PATH):
2     """อ่านบันทึกข้อมูลทั้งหมดจากไฟล์ในนารี"""
3     records = []
4     try:
5         if not os.path.exists(file_path):
6             return records
7         with open(file_path, 'rb') as f:
8             while True:
9                 record_data = f.read(STUDENT_RECORD_SIZE)
10                if not record_data:
11                    break
12                record = read_student_record(record_data)
13                if record:
14                    records.append(record)
15    except IOError as e:
16        print(f"เกิดข้อผิดพลาดในการอ่านไฟล์: {e}")
17    return records
18
19 def print_student_report(records, title="รายงานนักศึกษา"):
20     """แสดงรายงานนักศึกษาในรูปแบบตาราง"""
21     report = ""
22     report += "=====\n"
23     report += f"          {title}\n"
24     report += "=====\n"
25
26     headers = ["STUDENT ID", "FIRST NAME", "LAST NAME", "MAJOR", "YEAR", "STATUS"]
27     col_widths = [20, 20, 20, 15, 8, 15]
28
29     header_line = " | ".join(f"{{h:<{col_widths[i]}}}" for i, h in enumerate(headers))
30     report += header_line + "\n"
31     report += "-" * len(header_line) + "\n"
32
33     for rec in records:
34         row_data = [str(rec[h]) for h in headers]
35         # ตัดข้อความหากยาวเกิน
36         for i in range(len(row_data)):
37             if len(row_data[i]) > col_widths[i]:
38                 row_data[i] = row_data[i][:col_widths[i]-3] + "..."
39         row_line = " | ".join(f"{{row_data[i]:<{col_widths[i]}}}" for i in range(len(headers)))
40         report += row_line + "\n"
41
42     report += "-----\n"
43     print(report)
44

```

ภาพที่ 2-3 ไฟล์ [student.py](#) พังก์ชัน read_all_records_from_file และ พังก์ชัน print_student_report
(แสดงรายงาน)

2.6 พังก์ชัน add_student (เพิ่มข้อมูล)

- รับข้อมูลนักเรียนจากผู้ใช้ แล้วใช้ฟังก์ชัน create_student_record เพื่อ แปลงข้อมูลเป็นไบนาเรี่ยน จากนั้นใช้ write_record_to_file เพื่อ บันทึกข้อมูลไบนาเรี่ยนต่อท้ายไฟล์ student.bin.

2.7 พังก์ชัน view_students (แสดงทั้งหมด)

- ใช้ read_all_records_from_file เพื่อ ดึงข้อมูลนักศึกษาทั้งหมด จากรูปแบบข้อมูลไบนาเรี่ยนมาเป็น List ของ Dictionary จากนั้นใช้ฟังก์ชัน print_student_report เพื่อแสดงผลในรูปแบบตาราง.

2.8 พังก์ชัน view_single_student (แสดงรายบุคคล)

- รับรหัสนักเรียน ที่ต้องการค้นหา จากนั้น อ่านข้อมูลทั้งหมด เข้ามาในหน่วยความจำ แล้ว วนลูปค้นหา บันทึกที่มีรหัสตรงกัน หากพบจะแสดงรายละเอียดของนักศึกษาคนนั้นทันที

```

● ● ●
1 def add_student():
2     """เพิ่มข้อมูลนักเรียนใหม่"""
3     student_id = input("ป้อนรหัสนักเรียน (ไม่เกิน 16 ตัวอักษร): ")
4     first_name = input("ป้อนชื่อจริง (ไม่เกิน 50 ตัวอักษร): ")
5     last_name = input("ป้อนนามสกุล (ไม่เกิน 50 ตัวอักษร): ")
6     major = input("ป้อนสาขาวิชา (ไม่เกิน 20 ตัวอักษร): ")
7     try:
8         year_level = int(input("ป้อนชั้นปี (1, 2, 3, 4, ...): "))
9         status = int(input("ป้อนสถานะ (1 = Active, 0 = Inactive): "))
10    except ValueError:
11        print("ข้อมูลที่ป้อนไม่ถูกต้อง กรุณาป้อนเป็นตัวเลข")
12    return
13
14    record = create_student_record(student_id, first_name, last_name, major, year_level, status)
15    if record:
16        write_record_to_file(record)
17        print("เพิ่มข้อมูลนักเรียนสำเร็จ!")
18
19 def view_students():
20     """แสดงข้อมูลนักเรียนทั้งหมด"""
21     students = read_all_records_from_file()
22     if not students:
23         print("ไม่พบข้อมูลนักเรียนในระบบ")
24         return
25     print_student_report(students, title="รายงานนักศึกษา")
26
27 def view_single_student():
28     """แสดงข้อมูลนักเรียนรายบุคคลโดยค้นหาด้วยรหัสนักเรียน"""
29     student_id_to_view = input("ป้อนรหัสนักเรียนที่ต้องการดู: ")
30     students = read_all_records_from_file()
31     found = False
32
33     for student in students:
34         if student and student['STUDENT ID'] == student_id_to_view:
35             found = True
36             print("\n--- ข้อมูลนักเรียน ---")
37             print(f"รหัสนักเรียน: {student['STUDENT ID']}")
38             print(f"ชื่อจริ: {student['FIRST NAME']}")
39             print(f"นามสกุล: {student['LAST NAME']}")
40             print(f"สาขาวิชา: {student['MAJOR']}")
41             print(f"ชั้นปี: {student['YEAR']}")
42             print(f"สถานะ: {student['STATUS']}")
43             print("-----")
44             break
45
46     if not found:
47         print("ไม่พบรหัสนักเรียนที่ต้องการ")

```

ภาพที่ 2-4 ไฟล์โค้ดพังก์ชันเพิ่มข้อมูล แสดงข้อมูลทั้งหมด และแสดงข้อมูลรายบุคคล

2.9 พังก์ชัน view_filtered_students (กรองข้อมูล) พังก์ชันนี้ทำหน้าที่ แสดงรายงานนักศึกษาเฉพาะกลุ่ม โดยให้ผู้ใช้เลือกเงื่อนไขการกรอง

- หลักการทำงาน:

1. เลือกเงื่อนไข: แสดงเมนูให้ผู้ใช้เลือกเงื่อนไขการกรอง (สาขาวิชา, ชั้นปี, หรือสถานะ).
2. ดึงข้อมูลหลัก: เรียก read_all_records_from_file เพื่อดึงข้อมูลนักศึกษา ทั้งหมด เข้ามาในหน่วยความจำก่อน.
3. กรองข้อมูล: นำข้อมูลทั้งหมดมา วนลูปตรวจสอบ ตามเงื่อนไขที่ผู้ใช้ป้อน (เช่น ถ้าเลือก 1 จะกรองตามสาขาวิชา, เลือก 2 กรองตามชั้นปี).
4. แสดงผล: ถ้าพบข้อมูลที่ตรงตามเงื่อนไข จะส่งลิสต์ filtered_students ที่ผ่านการกรองแล้วไปให้พังก์ชัน print_student_report เพื่อแสดงเป็นตารางรายงานเฉพาะกลุ่มนั้นๆ

```

1 def view_filtered_students():
2     """แสดงข้อมูลนักเรียนโดยกรองตามเงื่อนไข"""
3     print("\n--- กรองข้อมูลนักเรียน ---")
4     print("เลือกเงื่อนไขการกรอง:")
5     print("1. กรองตามสาขาวิชา")
6     print("2. กรองตามชั้นปี")
7     print("3. กรองตามสถานะ")
8     filter_choice = input("กรุณาเลือก (1-3): ")
9
10    students = read_all_records_from_file()
11    if not students:
12        print("ไม่พบข้อมูลนักเรียนในระบบ")
13        return
14
15    filtered_students = []
16
17    if filter_choice == '1':
18        major_filter = input("ป้อนสาขาวิชาที่ต้องการกรอง: ")
19        filtered_students = [s for s in students if s and s['MAJOR'].lower() == major_filter.lower()]
20    elif filter_choice == '2':
21        try:
22            year_filter = int(input("ป้อนชั้นปีที่ต้องการกรอง (1, 2, 3, 4, ...): "))
23            filtered_students = [s for s in students if s and s['YEAR'] == year_filter]
24        except ValueError:
25            print("กรุณาป้อนชั้นปีเป็นตัวเลข")
26            return
27    elif filter_choice == '3':
28        status_filter = input("ป้อนสถานะที่ต้องการกรอง (Active/Inactive): ")
29        filtered_students = [s for s in students if s and s['STATUS'].lower() == status_filter.lower()]
30    else:
31        print("ตัวเลือกไม่ถูกต้อง")
32        return
33
34    if not filtered_students:
35        print("ไม่พบข้อมูลนักเรียนที่ตรงกับเงื่อนไข")
```

ภาพที่ 2-5 โค้ด การทำงาน ของ พังก์ชัน view_filtered_students (กรองข้อมูล)

2.10 พงก์ชันนีทำหน้าที่ แก้ไขข้อมูลนักศึกษา โดยใช้กลยุทธ์ "อ่านทั้งหมด-แก้ไข-เขียนใหม่ทั้งหมด"

1. ค้นหาและดึงข้อมูล: รับ รหัสนักเรียน ที่ต้องการแก้ไข จากนั้น อ่านข้อมูลนักศึกษาทั้งหมด จากไฟล์ student.bin เข้ามาในหน่วยความจำ
2. แก้ไข: วนลูปห้าบันทึกที่ตรงกัน เมื่อพบจะ แสดงข้อมูลเดิม และ รับค่าใหม่ (ชื่อ, สาขา, ชั้นปี ฯลฯ) จากผู้ใช้ โดยผู้ใช้สามารถกด Enter เพื่อใช้ค่าเดิมได้
3. สร้างไฟล์ใหม่ (Overwrite): ลบไฟล์เดิม: หากพบข้อมูลที่ต้องการแก้ไข ระบบจะ ลบไฟล์ student.bin เดิมทิ้ง ก่อน เขียนใหม่: วนลูป เขียนบันทึกใบnaireทั้งหมด

```

● ● ●
1 def update_student():
2     """แก้ไขข้อมูลนักเรียน"""
3     student_id_to_update = input("ป้อนรหัสนักเรียนที่ต้องการแก้ไข: ")
4     students = read_all_records_from_file()
5     found = False
6     new_records = []
7
8     for student in students:
9         if student and student['STUDENT ID'] == student_id_to_update:
10            found = True
11            print("=====")
12            print("    พบข้อมูลนักเรียนที่ต้องการแก้ไข")
13            print("=====")
14            print(f"รหัสนักเรียน: {student['STUDENT ID']}")
15            print(f"ชื่อจริง: {student['FIRST NAME']}")
16            print(f"นามสกุล: {student['LAST NAME']}")
17            print(f"สาขาวิชา: {student['MAJOR']}")
18            print(f"ชั้นปี: {student['YEAR']}")
19            print(f"สถานะ: {student['STATUS']}")
20            print("=====")
21
22            new_first_name = input(f"ป้อนชื่อจริงใหม่ (Enter เพื่อใช้ค่าเดิม): ")
23            if new_first_name:
24                student['FIRST NAME'] = new_first_name
25
26            new_last_name = input(f"ป้อนนามสกุลใหม่ (Enter เพื่อใช้ค่าเดิม): ")
27            if new_last_name:
28                student['LAST NAME'] = new_last_name
29
30            new_major = input(f"ป้อนสาขาวิชาใหม่ (Enter เพื่อใช้ค่าเดิม): ")
31            if new_major:
32                student['MAJOR'] = new_major
33
34            new_year_level = input(f"ป้อนชั้นปีใหม่ (Enter เพื่อใช้ค่าเดิม): ")
35            if new_year_level:
36                try:
37                    student['YEAR'] = int(new_year_level)
38                except ValueError:
39                    print("ชั้นปีไม่ถูกต้อง ใช้ค่าเดิม")
40
41            new_status = input(f"ป้อนสถานะใหม่ (1=Active, 0=Inactive) (Enter เพื่อใช้ค่าเดิม): ")
42            if new_status:
43                try:
44                    student['STATUS'] = 'Active' if int(new_status) == 1 else 'Inactive'
45                except ValueError:
46                    print("สถานะไม่ถูกต้อง ใช้ค่าเดิม")
47
48            if student:
49                updated_record = create_student_record(
50                    student['STUDENT ID'],
51                    student['FIRST NAME'],
52                    student['LAST NAME'],
53                    student['MAJOR'],
54                    student['YEAR'],
55                    1 if student['STATUS'] == 'Active' else 0
56                )
57                if updated_record:
58                    new_records.append(updated_record)
59
60        if found:
61            try:
62                os.remove(STUDENT_FILE_PATH)
63                for record in new_records:
64                    write_record_to_file(record)
65                    print("แก้ไขข้อมูลสำเร็จ!")
66            except IOError as e:
67                print(f"เกิดข้อผิดพลาดในการแก้ไขไฟล์: {e}")
68        else:
69            print("ไม่พบรหัสนักเรียนที่ต้องการแก้ไข")

```

ภาพที่ 2-6 ไฟล์ พงก์ชันนีทำหน้าที่ แก้ไขข้อมูลนักศึกษา

2.11 พังก์ชัน delete_student (ลบข้อมูล)

พังก์ชันนี้ทำหน้าที่ ลบบันทึกนักศึกษา ออกจากฐานข้อมูลอย่างถาวร โดยใช้กลยุทธ์ "อ่านทั้งหมด-ยกเว้นรายการที่ต้องการลบ-เขียนใหม่ทั้งหมด"

- หลักการทำงาน:

1. รับ รหัสนักเรียน ที่ต้องการลบ และ อ่านข้อมูลทั้งหมด เข้ามาในหน่วยความจำ
2. วนลูปสร้างลิสต์ remaining_records โดย ข้าม บันทึกที่มีรหัสร่วงกับที่ผู้ใช้ป้อน
3. หากพรหัส: ระบบจะ ลบไฟล์ student.bin เดิมทั้ง ก่อน จากนั้นจะวนลูป
สร้างบันทึกใบหนารี จาก remaining_records แล้ว เขียนบันทึกที่เหลือทั้งหมด
กลับลงในไฟล์ student.bin ใหม่

2.12 พังก์ชัน student_menu (เมนูย่อยนักศึกษา)

พังก์ชันนี้ทำหน้าที่เป็น หน้าต่างควบคุมหลัก สำหรับงานที่เกี่ยวข้องกับข้อมูลนักศึกษาโดยเฉพาะ

- หลักการทำงาน: แสดงเมนูย่อย 6 ตัวเลือกสำหรับดำเนินการจัดการข้อมูลนักศึกษา (CRUD: เพิ่ม, ดู,
แก้ไข, ลบ)
- การนำทาง: เมื่อผู้ใช้เลือกตัวเลือกใด ๆ (1-6) ระบบจะ เรียกใช้พังก์ชันย่อยที่เกี่ยวข้อง (เช่น
add_student(), update_student(), delete_student()) ที่ได้อธิบายไปก่อนหน้านี้
เพื่อดำเนินการตามคำสั่ง

กล่าวโดยสรุป: delete_student เป็นกลไกที่ ถอนบันทึกเดียว ออกจากไฟล์ใบหนารีอย่างถาวร และ
student_menu เป็น ตัวจัดการการเข้าถึง พังก์ชัน CRUD ทั้งหมดของ Module นักศึกษา

```

● ● ●
1 def delete_student():
2     """ลบข้อมูลนักเรียนแบบดาวร"""
3     student_id_to_delete = input("ป้อนรหัสนักเรียนที่ต้องการลบ: ")
4     students = read_all_records_from_file()
5     found = False
6
7     remaining_records = []
8
9     for student in students:
10         if student and student['STUDENT ID'] == student_id_to_delete:
11             found = True
12             print("ลบข้อมูลนักเรียนสำเร็จ!")
13         elif student:
14             remaining_records.append(student)
15
16     if found:
17         try:
18             os.remove(STUDENT_FILE_PATH)
19             for student in remaining_records:
20                 record = create_student_record(
21                     student['STUDENT ID'],
22                     student['FIRST NAME'],
23                     student['LAST NAME'],
24                     student['MAJOR'],
25                     student['YEAR'],
26                     1 if student['STATUS'] == 'Active' else 0
27                 )
28                 if record:
29                     write_record_to_file(record)
30             except IOError as e:
31                 print(f"เกิดข้อผิดพลาดในการลบไฟล์: {e}")
32         else:
33             print("ไม่พบรหัสนักเรียนที่ต้องการลบ")
34
35 def student_menu():
36     """เมนูย่อยสำหรับจัดการข้อมูลนักเรียน (CRUD)"""
37     while True:
38         print("\n===== ระบบจัดการข้อมูลนักเรียน =====")
39         print("1. เพิ่มข้อมูลนักเรียน")
40         print("2. ดูข้อมูลนักเรียนทั้งหมด")
41         print("3. ดูข้อมูลนักเรียนรายบุคคล")
42         print("4. ดูข้อมูลนักเรียนแบบกรอง")
43         print("5. แก้ไขข้อมูลนักเรียน")
44         print("6. ลบข้อมูลนักเรียน")
45         print("0. กลับสู่เมนูหลัก")
46
47         choice = input("กรุณาเลือกเมนู: ")
48
49         if choice == '1':
50             add_student()
51         elif choice == '2':
52             view_students()
53         elif choice == '3':
54             view_single_student()
55         elif choice == '4':
56             view_filtered_students()
57         elif choice == '5':
58             update_student()
59         elif choice == '6':
60             delete_student()
61         elif choice == '0':
62             print("ย้อนกลับสู่เมนูหลัก...")
63             break
64         else:
65             print("ตัวเลือกไม่ถูกต้อง กรุณาลองใหม่อีกครั้ง")
66
67 if __name__ == "__main__":
68     student_menu()

```

ภาพที่ 2-7 โค้ด พิ้งก์ชัน delete_student (ลบข้อมูล) และ พิ้งก์ชัน student_menu (เมนูย่อยนักศึกษา)

3.ไฟล์ register.py

3.1 พังก์ชัน create_registration_record(บันทึกข้อมูลการลงทะเบียน)

1.การเตรียมสตริง (String Preparation):

- การเตรียมสตริง (String Preparation):
- ตัดสตริงไปต่อกับความยาวสูงสุดตามที่กำหนด ([:10] สำหรับ ID, [:50] สำหรับชื่อ)

2.การแพ็คข้อมูล (Packing):

- ใช้พังก์ชัน struct.pack() เพื่อร่วมข้อมูลที่เตรียมไว้ทั้งหมด (ID, ชื่อ, credit, academic_year, semester, is_active)

3.การจัดการข้อผิดพลาด (Error Handling):

- หากเกิดข้อผิดพลาดในการแพ็ค (เช่น ชนิดข้อมูลไม่ตรง) จะถูกจับโดย try...except struct.error และพังก์ชันจะส่งคืนค่า None

```

● ● ●
1 import struct
2 import os
3 from datetime import datetime
4
5 # กำหนดพารามของไฟล์ร้านข้อมูล
6 current_dir = os.path.dirname(os.path.abspath(__file__))
7 main_dir = os.path.dirname(current_dir)
8 REGISTRATION_FILE_PATH = os.path.join(main_dir, 'registration.bin')
9 STUDENT_FILE_PATH = os.path.join(main_dir, 'student.bin')
10
11 # รูปแบบของข้อมูลนักเรียน
12 STUDENT_RECORD_FORMAT = '<16s50s50s20sBB'
13 STUDENT_RECORD_SIZE = struct.calcsize(STUDENT_RECORD_FORMAT)
14
15 # รูปแบบของข้อมูลการลงทะเบียนเรียน
16 REGISTRATION_RECORD_FORMAT = '<I16s16sdB'
17 REGISTRATION_RECORD_SIZE = struct.calcsize(REGISTRATION_RECORD_FORMAT)
18
19 def create_registration_record(register_id, student_id, course_id, registration_date, status):
20     """สร้างบันทึกข้อมูลการลงทะเบียนในรูปแบบบинаรี"""
21     packed_student_id = student_id.encode('utf-8')[:16].ljust(16, b'\x00')
22     packed_course_id = course_id.encode('utf-8')[:16].ljust(16, b'\x00')
23     try:
24         record = struct.pack(
25             REGISTRATION_RECORD_FORMAT,
26             register_id,
27             packed_student_id,
28             packed_course_id,
29             registration_date,
30             status
31         )
32         return record
33     except struct.error as e:
34         print(f"เกิดข้อผิดพลาดในการแพ็คข้อมูล: {e}")
35         return None

```

ภาพที่ 3-1 โค้ด พังก์ชัน create_registration_record(บันทึกข้อมูลการลงทะเบียน)

3.2 ฟังก์ชัน read_registration_record(อ่านบันทึกข้อมูลการลงทะเบียน)

1. การแยกข้อมูลในนารี (struct.unpack)

- unpacked_data = struct.unpack(REGISTRATION_RECORD_FORMAT, record_data):
 - struct.unpack(): ฟังก์ชันหลักที่ทำการแยก (unpack) ข้อมูลในนารีที่รับเข้ามา (record_data)
 - REGISTRATION_RECORD_FORMAT: (ต้องมีการกำหนดไว้ก่อน) คือสตริงที่ระบุ รูปแบบ (format string) ที่ใช้ในการแพ็คข้อมูลนี้ตั้งแต่แรก เพื่อให้ unpack แยกข้อมูลแต่ละส่วน ออกมากได้อย่างถูกต้อง

2. การแปลงข้อมูลและจัดรูปแบบ

- student_id = unpacked_data[1].strip(b'\x00').decode('utf-8') และ course_id = unpacked_data[2].strip(b'\x00').decode('utf-8');

```

● ● ●
1 def read_registration_record(record_data):
2     """อ่านบันทึกข้อมูลการลงทะเบียนจากรูปแบบใบนารี"""
3     try:
4         unpacked_data = struct.unpack(REGISTRATION_RECORD_FORMAT, record_data)
5         register_id = unpacked_data[0]
6         student_id = unpacked_data[1].strip(b'\x00').decode('utf-8')
7         course_id = unpacked_data[2].strip(b'\x00').decode('utf-8')
8         registration_date = datetime.fromtimestamp(unpacked_data[3])
9         status = 'Registered' if unpacked_data[4] == 1 else 'Dropped'
10        return {
11            'ID': register_id,
12            'STUDENT ID': student_id,
13            'COURSE ID': course_id,
14            'REGISTRATION DATE': registration_date,
15            'STATUS': status
16        }
17    except (struct.error, ValueError) as e:
18        print(f"เกิดข้อผิดพลาดในการอ่านแพ็คข้อมูล: {e}")
19    return None

```

ภาพที่ 3-2 โค้ดฟังก์ชัน read_registration_record(อ่านบันทึกข้อมูลการลงทะเบียน)

3.3 พังก์ชัน write_record_to_file (เขียนบันทึกข้อมูลลงในไฟล์)

1. การเปิดไฟล์ (Opening the File):

- ใช้ `with open(file_path, 'ab') as f:` เพื่อเปิดไฟล์ที่ระบุด้วย `file_path`
- '`a`' (`append`): โหมดนี้ทำให้ไฟล์ถูกเปิดเพื่อ เขียนต่อท้าย (ข้อมูลใหม่จะไม่ทับข้อมูลเก่า)
- '`b`' (`binary`): โหมดนี้ระบุว่าเป็นการทำงานกับ ข้อมูลในนารี (ไม่ใช่ข้อความปกติ)

2. การเขียนข้อมูล (Writing Data):

- `f.write(record):` ทำการ เขียน ข้อมูลที่รับเข้ามาในพารามิเตอร์ `record` (ซึ่งต้องเป็นข้อมูลในนารีที่ถูกแพ็คแล้ว) ลงในไฟล์ทันที โดยจะต่อท้ายข้อมูลที่มีอยู่เดิม

3. การจัดการข้อผิดพลาด (Error Handling):

- `try...except IOError as e::` หากเกิดข้อผิดพลาดเกี่ยวกับการทำงานของไฟล์ (เช่น ไม่มีสิทธิ์ในการเขียน, ไฟล์ไม่มีอยู่) จะถูกจับโดย `IOError` และพิมพ์ข้อความแจ้งเตือน

```
● ● ●
1 def write_record_to_file(record, file_path=REGISTRATION_FILE_PATH):
2     """เขียนบันทึกข้อมูลลงในไฟล์ในนารี"""
3     try:
4         with open(file_path, 'ab') as f:
5             f.write(record)
6     except IOError as e:
7         print(f"เกิดข้อผิดพลาดในการเขียนไฟล์: {e}")
```

ภาพที่ 3-3 โค้ด `write_record_to_file` (เขียนบันทึกข้อมูลลงในไฟล์)

3.4 พังก์ชัน read_all_records_from_file(อ่านข้อมูลบันทึกไฟล์ทั้งหมด)

1. การตรวจสอบและเตรียมการ (Check and Setup)

- `records = []:` สร้าง รายการว่างเปล่า สำหรับเก็บข้อมูลบันทึกทั้งหมดที่จะอ่านได้
- `if not os.path.exists(file_path): return records:` ใช้โมดูล `os.path.exists` เพื่อ ตรวจสอบว่าไฟล์ มีอยู่จริงหรือไม่ ถ้าไม่มีไฟล์ จะส่งกลับรายการว่างเปล่าทันที

2. การเปิดและวนอ่านไฟล์ (Open and Loop Reading)

- `with open(file_path, 'rb') as f::`
 - เปิดไฟล์ในโหมด '`rb`' (`r` = read, `b` = binary) เพื่อ อ่านข้อมูลในนารี
 - `while True::` เริ่มลูปการอ่านไปเรื่อย ๆ จนกว่าจะสิ้นสุดไฟล์
- `record_data = f.read(REGISTRATION_RECORD_SIZE):`

```

● ● ●
1 def read_all_records_from_file(file_path=REGISTRATION_FILE_PATH):
2     """อ่านบันทึกข้อมูลทั้งหมดจากไฟล์ใบนารี"""
3     records = []
4     try:
5         if not os.path.exists(file_path):
6             return records
7         with open(file_path, 'rb') as f:
8             while True:
9                 record_data = f.read(REGISTRATION_RECORD_SIZE)
10                if not record_data:
11                    break
12                record = read_registration_record(record_data)
13                if record:
14                    records.append(record)
15    except IOError as e:
16        print(f"เกิดข้อผิดพลาดในการอ่านไฟล์: {e}")
17    return records

```

ภาพที่ 3-4 โค้ดฟังก์ชัน read_all_records_from_file(อ่านข้อมูลบันทึกไฟล์ทั้งหมด)

3.5 ฟังก์ชัน get_next_register_id(ทำการลงทะเบียนถัดไป)

1. การอ่านข้อมูลทั้งหมด:

- records = read_all_records_from_file(): เรียกใช้ฟังก์ชันก่อนหน้าเพื่อ อ่านข้อมูลบันทึกทั้งหมด จากไฟล์ใบนาเริ่มต้นในรูปแบบรายการ (List) ของ Dictionary

2. การจัดการกรณีไม่มีข้อมูล:

- if not records: return 1: ถ้าอ่านไฟล์มาแล้ว ไม่พบรายการใด ๆ เลย (รายการว่างเปล่า) แสดงว่า เป็นการลงทะเบียนแรก ให้ส่งกลับ ID ลำดับที่ 1

3. การกรองข้อมูลที่ถูกต้อง:

- valid_records = [r for r in records if r is not None]: สร้างรายการใหม่ที่ประกอบด้วยเฉพาะ บันทึกที่ แปลงข้อมูลสำเร็จ (r is not None) เพื่อหลีกเลี่ยงการคำนวณ ID จากบันทึกที่เสียหาย

4. การคำนวณ ID ถัดไป:

- return max(r['ID'] for r in valid_records) + 1:
 - max(r['ID'] for r in valid_records): หา ค่า ID สูงสุด จากบันทึกที่ถูกต้องทั้งหมด



```

1 def get_next_register_id():
2     """หา ID การลงทะเบียนถัดไป"""
3     records = read_all_records_from_file()
4     if not records:
5         return 1
6     valid_records = [r for r in records if r is not None]
7     if not valid_records:
8         return 1
9     return max(r['ID'] for r in valid_records) + 1

```

ภาพที่ 3-5 โค้ด พิมพ์ชั้น get_next_register_id(ทำการลงทะเบียนถัดไป)

3.6 พิมพ์ชั้น read_student_by_id(อ่านข้อมูลนักเรียน)

1. การตรวจสอบไฟล์:

- if not os.path.exists(STUDENT_FILE_PATH): ตรวจสอบว่าไฟล์ในนารีมีอยู่จริงหรือไม่ ถ้าไม่พบไฟล์ จะพิมพ์ข้อความและส่งกลับ None

2. การอ่านไฟล์ทีละบล็อก (Sequential Scan):

- with open(STUDENT_FILE_PATH, 'rb') as f: เปิดไฟล์ในโหมด 'rb' (อ่านข้อมูลในนารี)
- while True: วนลูปเพื่ออ่านข้อมูลไปเรื่อยๆ ตั้งแต่ต้นไฟล์จนจบ
- record_data = f.read(STUDENT_RECORD_SIZE) อ่านข้อมูลในนารีออกมากทีละบล็อกตามขนาดที่ของบันทึกนักเรียนหนึ่งรายการ

3. การแยกและตรวจสอบข้อมูล (Unpack and Check):

- unpacked_data = struct.unpack(STUDENT_RECORD_FORMAT, record_data): ใช้ struct.unpack() เพื่อแยกข้อมูลในนารีเป็นล็อกกันน้อยออกมาตามรูปแบบที่กำหนด

```

1 def read_student_by_id(student_id):
2     """อ่านข้อมูลนักเรียนจาก student.bin โดยใช้รหัสนักเรียน"""
3     try:
4         if not os.path.exists(STUDENT_FILE_PATH):
5             print("ไม่พบไฟล์ student.bin")
6             return None
7
8         with open(STUDENT_FILE_PATH, 'rb') as f:
9             while True:
10                 record_data = f.read(STUDENT_RECORD_SIZE)
11                 if not record_data:
12                     break
13
14             try:
15                 unpacked_data = struct.unpack(STUDENT_RECORD_FORMAT, record_data)
16                 current_student_id = unpacked_data[0].strip(b'\x00').decode('utf-8')
17
18                 if current_student_id == student_id:
19                     status_map = {1: 'Active', 0: 'Inactive'}
20                     status_text = status_map.get(unpacked_data[5], 'Unknown')
21
22                     return {
23                         'student_id': current_student_id,
24                         'first_name': unpacked_data[1].strip(b'\x00').decode('utf-8'),
25                         'last_name': unpacked_data[2].strip(b'\x00').decode('utf-8'),
26                         'major': unpacked_data[3].strip(b'\x00').decode('utf-8'),
27                         'year_level': unpacked_data[4],
28                         'status': status_text,
29                         'status_code': unpacked_data[5]
30                     }
31
32             except (struct.error, UnicodeDecodeError) as e:
33                 print(f"ข้อผิดพลาดในการอ่าน record: {e}")
34                 continue
35
36     return None
37
38 except IOError as e:
39     print(f"เกิดข้อผิดพลาดในการอ่านไฟล์นักเรียน: {e}")
40     return None

```

ภาพที่ 3-6 โค้ดฟังก์ชัน read_student_by_id(อ่านข้อมูลนักเรียน)

3.7 พังก์ชัน get_student_info_for_registration(ดึงข้อมูลนักเรียน)

1. การรับรหัสและการค้นหา:

- student_id = input(...).strip(): รับรหัสนักเรียน จากผู้ใช้ผ่านทางคีย์บอร์ดและตัดซองว่าหัวท้ายออก
- student = read_student_by_id(student_id): เรียกใช้ฟังก์ชัน read_student_by_id เพื่อค้นหาข้อมูลนักเรียนจากไฟล์เบนารี

2. การตรวจสอบข้อมูลเบื้องต้น:

- if not student: ... return None: ถ้าฟังก์ชันค้นหา ไม่พบข้อมูลนักเรียน จะพิมพ์ข้อความแจ้งเตือนและส่งกลับ None

3. การแสดงและยืนยันข้อมูล:

- print(...): แสดงข้อมูลนักเรียน ที่ค้นพบ (ชื่อ, สาขา, ชั้นปี, สถานะ) ให้ผู้ใช้ตรวจสอบ
- confirm = input("ใช่คนนี้ใช่หรือไม่? (y/n): ").lower(): สອบถามการยืนยัน จากผู้ใช้

```

● ● ●
1 def get_student_info_for_registration():
2     """ดึงข้อมูลนักเรียนและยืนยันก่อนลงทะเบียน"""
3     student_id = input("ป้อนรหัสนักเรียน: ").strip()
4
5     student = read_student_by_id(student_id)
6
7     if not student:
8         print(f"ไม่พบนักเรียนที่มีรหัส {student_id} ในระบบ")
9         return None
10
11    if student['status_code'] == 0:
12        print(f"นักเรียนรหัส {student_id} มีสถานะ Inactive ไม่สามารถลงทะเบียนได้")
13        return None
14
15    print("\n==== พบข้อมูลนักเรียน ====")
16    print(f"รหัสนักเรียน: {student['student_id']} ")
17    print(f"ชื่อ: {student['first_name']} {student['last_name']} ")
18    print(f"สาขา: {student['major']} ")
19    print(f"ชั้นปี: {student['year_level']} ")
20    print(f"สถานะ: {student['status']} ")
21    print("=====")
22
23    confirm = input("ใช่คนนี้ใช่หรือไม่? (y/n): ").lower()
24    if confirm == 'y' or confirm == 'yes'

```

ภาพ 3-7 โค้ด พังก์ชัน get_student_info_for_registration(ดึงข้อมูลนักเรียน)

3.8 พัฟ์ชั่น print_registration_report(แสดงรายงานการลงทะเบียน)

1. การเตรียมส่วนหัวรายงาน (Header Generation):

- พัฟ์ชั่นเริ่มต้นด้วยการสร้างสตริง report ว่างเปล่า
- เพิ่มเส้นแบ่งและ ชื่อรายงาน (title) ที่อยู่กึ่งกลางเข้าไปใน report

2. การวนลูปสร้างแถวข้อมูล (Data Row Iteration):

- for reg in records: วนลูปอ่านข้อมูลบันทึกการลงทะเบียนที่ละรายการ (reg)
- การจัดรูปแบบวันที่:
 - date_str = reg['REGISTRATION DATE'].strftime('%Y-%m-%d %H:%M:%S'): แปลงวัตถุ datetime ในช่อง 'REGISTRATION DATE' ให้เป็น สตริงวันที่และเวลา ที่อ่านง่าย

```

1 def print_registration_report(records, title="รายงานการลงทะเบียน"):
2     """แสดงรายงานการลงทะเบียนในรูปแบบตาราง"""
3     report = ""
4     report += "=====\n"
5     report += f"          {title}\n"
6     report += "=====\n"
7
8     headers = ["ID", "STUDENT ID", "COURSE ID", "REGISTRATION DATE", "STATUS"]
9     col_widths = [8, 20, 20, 25, 15]
10
11    header_line = " | ".join(f"{{h:<{col_widths[i]}}}" for i, h in enumerate(headers))
12    report += header_line + "\n"
13    report += "-" * len(header_line) + "\n"
14
15    for reg in records:
16        try:
17            date_str = reg['REGISTRATION DATE'].strftime('%Y-%m-%d %H:%M:%S')
18        except Exception:
19            date_str = "Invalid Date"
20
21        row_data = [
22            str(reg['ID']),
23            reg['STUDENT ID'],
24            reg['COURSE ID'],
25            date_str,
26            reg['STATUS']
27        ]
28        # ตัดชื่อความหากยาวเกิน
29        for i in range(len(row_data)):
30            if len(row_data[i]) > col_widths[i]:
31                row_data[i] = row_data[i][:col_widths[i]-3] + "..."
32        row_line = " | ".join(f"{{row_data[i]:<{col_widths[i]}}}" for i in range(len(headers)))
33        report += row_line + "\n"
34
35    report += "-----\n"
36    print(report)

```

ภาพที่ 3-8 โค้ดพัฟ์ชั่น print_registration_report(แสดงรายงานการลงทะเบียน)

3.9 พังก์ชัน add_registration(เพิ่มข้อมูลการลงทะเบียน)

1. การดึงและตรวจสอบข้อมูลนักเรียน:

- student = get_student_info_for_registration(): เรียกใช้พังก์ชันเพื่อรับรหัสนักเรียน ค้นหาและยืนยันตัวตน

2. การรับรหัสวิชา:

- course_id = input(...): รับรหัสวิชา จากผู้ใช้
- if not course_id: return: ตรวจสอบว่ารหัสวิชาไม่เป็นค่าว่าง

3. การรับและตรวจสอบสถานะ:

- try...except ValueError: จัดการการป้อนข้อมูลที่ผิดพลาด

```

● ● ●
1 def add_registration():
2     """เพิ่มข้อมูลการลงทะเบียนใหม่"""
3     student = get_student_info_for_registration()
4     if not student:
5         return
6
7     course_id = input("ป้อนรหัสวิชา: ").strip()
8     if not course_id:
9         print("รหัสวิชาว่าง กรุณาลองใหม่")
10        return
11
12    try:
13        status = int(input("ป้อนสถานะ (1 = Registered, 0 = Dropped): "))
14        if status not in (0, 1):
15            raise ValueError
16    except ValueError:
17        print("☒ สถานะไม่ถูกต้อง กรุณาป้อนเป็นตัวเลข 0 หรือ 1")
18        return
19
20    register_id = get_next_register_id()
21    registration_date = datetime.now().timestamp()
22
23    record = create_registration_record(
24        register_id,
25        student['student_id'],
26        course_id,
27        registration_date,
28        status
29    )
30    if record:
31        write_record_to_file(record)
32        print("✓ เพิ่มข้อมูลการลงทะเบียนสำเร็จ!")

```

ภาพที่ 3.9 โค้ดพังก์ชัน add_registration(เพิ่มข้อมูลการลงทะเบียน)

3.10 พังก์ชัน view_registration(แสดงข้อมูลการลงทะเบียนทั้งหมด)

1. การดึงข้อมูล:

- registrations = read_all_records_from_file(): เรียกใช้ฟังก์ชัน read_all_records_from_file เพื่ออ่านไฟล์ใบหนารีทั้งหมด และแปลงบันทึกแต่ละรายการให้เป็นรายการ (List) ของ Dictionary

2. การตรวจสอบความว่างเปล่า:

- if not registrations: ... return: ตรวจสอบ ว่ารายการที่อ่านได้มีข้อมูลอยู่หรือไม่ ถ้าเป็นรายการว่างเปล่า (not registrations เป็นจริง) จะพิมพ์ข้อความแจ้งว่า "ไม่พบข้อมูลการลงทะเบียนในระบบ" และหยุดทำงานทันที

3. การแสดงผลรายงาน:

- print_registration_report(registrations, title="รายงานการลงทะเบียน"): หากมีข้อมูล จะเรียกใช้ฟังก์ชัน print_registration_report เพื่อนำรายการของ Dictionary ที่ได้มา จัดรูปแบบและพิมพ์ออกเป็นตารางรายงาน ที่สวยงามและอ่านง่าย

```

● ● ●

1 def view_registrations():
2     """แสดงข้อมูลการลงทะเบียนทั้งหมด"""
3     registrations = read_all_records_from_file()
4     if not registrations:
5         print("ไม่พบข้อมูลการลงทะเบียนในระบบ")
6         return
7     print_registration_report(registrations, title="รายงานการลงทะเบียน")

```

ภาพที่ 3.10 พังก์ชัน view_registration(แสดงข้อมูลการลงทะเบียนทั้งหมด)

3.11 พังก์ชัน view_single_registration(แสดงข้อมูลการลงทะเบียน)

1 การรับและตรวจสอบรหัส ID:

- try...except ValueError: พยายามรับรหัส ID จากผู้ใช้ด้วย input() และแปลงเป็น จำนวนเต็ม (int())
- หากผู้ใช้ป้อนค่าที่ไม่ใช่ตัวเลข จะเกิด ValueError พังก์ชันจะแจ้งข้อผิดพลาดและหยุดทำงาน

2. การดึงข้อมูลทั้งหมด:

- registrations = read_all_records_from_file(): เรียกใช้ฟังก์ชันก่อนหน้าเพื่อ อ่านและแปลงข้อมูลบันทึกทั้งหมด จากไฟล์ใบหนารีให้เป็นรายการของ Dictionary

3. การกรองข้อมูลเพื่อค้นหารายการเดียว:

- filtered_registrations = [r for r in registrations if r and r['ID'] == reg_id]:

```

1 def view_single_registration():
2     """แสดงข้อมูลการลงทะเบียนรายการเดียวตามรหัส ID"""
3     try:
4         reg_id = int(input("ป้อนรหัส ID การลงทะเบียนที่ต้องการดู: "))
5     except ValueError:
6         print("รหัส ID ไม่ถูกต้อง กรุณาป้อนเป็นตัวเลข")
7         return
8
9     registrations = read_all_records_from_file()
10    filtered_registrations = [r for r in registrations if r and r['ID'] == reg_id]
11
12    if not filtered_registrations:
13        print("ไม่พบรหัส ID การลงทะเบียนที่ต้องการดู")
14        return
15
16    print_registration_report(filtered_registrations, title="รายงานการลงทะเบียนรายการเดียว")

```

ภาพที่ 3-11 โค้ดฟังก์ชัน view_single_registration(แสดงข้อมูลการลงทะเบียน)

3.12 ฟังก์ชัน view_filtered_registration(แสดงการลงทะเบียน)

1. ฟังก์ชัน view_filtered_registrations(): แสดงรายงานแบบกรอง

- การแสดงเมนูและการเลือก: แสดงตัวเลือกการกรอง 3 แบบ (รหัสนักเรียน, รหัสวิชา, สถานะ 'Registered') และรับค่า filter_choice จากผู้ใช้

2. ฟังก์ชัน update_registration(): เริ่มต้นการแก้ไขข้อมูล

- การรับ ID และตรวจสอบ:
 - ใช้ try...except ValueError เพื่อรับรหัส ID จากผู้ใช้และพยายามแปลงเป็น จำนวนเต็ม (int)

```

● ● ● ● ●
1 def view_filtered_registrations():
2     """แสดงข้อมูลการลงทะเบียนที่กรองตามเงื่อนไข"""
3     print("\n--- ด้วยการกรอง ---")
4     print("1. กรองตามรหัสนักเรียน")
5     print("2. กรองตามรหัสวิชา")
6     print("3. กรองตามสถานะ (Registered)")
7     print("4. กลับไปเมนูหลัก")
8     filter_choice = input("กรุณาเลือกการกรอง (1-4): ")
9
10    registrations = read_all_records_from_file()
11    if not registrations:
12        print("ไม่พบข้อมูลการลงทะเบียนในระบบ")
13        return
14
15    filtered_registrations = []
16
17    if filter_choice == '1':
18        student_id = input("ป้อนรหัสนักเรียนที่ต้องการกรอง: ").strip()
19        filtered_registrations = [r for r in registrations if r and r['STUDENT ID'] == student_id]
20
21    elif filter_choice == '2':
22        course_id = input("ป้อนรหัสวิชาที่ต้องการกรอง: ").strip()
23        filtered_registrations = [r for r in registrations if r and r['COURSE ID'] == course_id]
24
25    elif filter_choice == '3':
26        filtered_registrations = [r for r in registrations if r and r['STATUS'] == 'Registered']
27
28    elif filter_choice == '4':
29        return
30
31    else:
32        print("ด้วยไม่ตุกต้อง")
33        return
34
35    if not filtered_registrations:
36        print("ไม่พบข้อมูลที่ตรงตามเงื่อนไขการกรอง")
37        return
38
39    print_registration_report(filtered_registrations, title=f"ราย"

```

ภาพที่ 3-12 โค้ดฟังก์ชัน view_filtered_registration(แสดงการลงทะเบียน)

3.13 พังก์ชัน update_registration (แก้ไขข้อมูลนักเรียน)

1. การรับและตรวจสอบ ID

- รับ ID: รับรหัส ID การลงทะเบียนที่ต้องการแก้ไขจากผู้ใช้และตรวจสอบว่า เป็นตัวเลข โดยใช้ try...except ValueError

2. การอ่านและค้นหาข้อมูล

- registrations = read_all_records_from_file(): อ่านบันทึกทั้งหมด จากไฟล์ในนารี และแปลงเป็นรายการของ Dictionary
- วนลูปค้นหา: วนลูปผ่านบันทึกทั้งหมดเพื่อหารายการที่ reg['ID'] == reg_id_to_update

3. การแสดงผลและการแก้ไขข้อมูล

- แสดงผล: เมื่อพบบันทึกที่ตรงกัน (found = True) จะ แสดงรายละเอียดปัจจุบัน ให้ผู้ใช้ดู
- รับสถานะใหม่: รับค่าสถานะใหม่ (1 หรือ 0) จากผู้ใช้

```

1 def update_registration():
2     """แก้ไขข้อมูลการลงทะเบียน"""
3     try:
4         reg_id_to_update = int(input("ป้อนรหัส ID การลงทะเบียนที่ต้องการแก้ไข: "))
5     except ValueError:
6         print("รหัส ID ไม่ถูกต้อง กรุณาป้อนเป็นตัวเลข")
7         return
8
9     registrations = read_all_records_from_file()
10    found = False
11    new_records = []
12
13    for reg in registrations:
14        if reg and reg['ID'] == reg_id_to_update:
15            found = True
16            print("====")
17            print("  บันทึกการลงทะเบียนที่ต้องการแก้ไข")
18            print("====")
19            print(f"ID การลงทะเบียน: {reg['ID']}")
20            print(f"รหัสนักเรียน: {reg['STUDENT ID']}")
21            print(f"รหัสวิชา: {reg['COURSE ID']}")
22            print(f"วันลงทะเบียน: {reg['REGISTRATION DATE'].strftime('%Y-%m-%d %H:%M:%S')}")
23            print(f"สถานะ: {reg['STATUS']}")
24            print("====")
25
26            new_status = input("ป้อนสถานะใหม่ (1=Registered, 0=Dropped) (Enter เพื่อใส่ค่าเดิม): ")
27            if new_status:
28                try:
29                    new_status = int(new_status)
30                    if new_status not in [0, 1]:
31                        raise ValueError
32                    reg['STATUS'] = 'Registered' if new_status == 1 else 'Dropped'
33                except ValueError:
34                    print("สถานะไม่ถูกต้อง ใช้ค่าเดิม")
35
36                if reg:
37                    updated_record = create_registration_record(
38                        reg['ID'],
39                        reg['STUDENT ID'],
40                        reg['COURSE ID'],
41                        reg['REGISTRATION DATE'].timestamp(),
42                        1 if reg['STATUS'] == 'Registered' else 0
43                    )
44                    if updated_record:
45                        new_records.append(updated_record)
46
47                if found:
48                    try:
49                        if os.path.exists(REGISTRATION_FILE_PATH):
50                            os.remove(REGISTRATION_FILE_PATH)

```

ภาพที่ 3-13 โค้ดพังก์ชัน update_registration (แก้ไขข้อมูลนักเรียน)

3.14 พังก์ชัน delete_registration(แสดงการลงทะเบียนแบบภาคร)

1. การรับและตรวจสอบ ID ที่จะลบ

- try...except ValueError: รับรหัส ID จากผู้ใช้และ แปลงเป็นจำนวนเต็ม หากป้อนค่าไม่ถูกต้อง จะแจ้งเตือนและหยุดทำงาน

2. การอ่านและกรองข้อมูล

- registrations = read_all_records_from_file(): อ่านบันทึกทั้งหมด จากไฟล์ใบnaire และแปลงเป็นรายการของ Dictionary

3. การจัดการกรณีไม่พบข้อมูล

else: print("ไม่พบรหัส ID..."): หากการวนลูปไม่พบ ID ที่ต้องการลบ (found เป็น False) จะแจ้งให้ผู้ใช้ทราบ

```

● ● ●
1 def delete_registration():
2     """ลบข้อมูลการลงทะเบียนแบบภาคร"""
3     try:
4         reg_id_to_delete = int(input("ป้อนรหัส ID การลงทะเบียนที่ต้องการลบ: "))
5     except ValueError:
6         print("รหัส ID ไม่ถูกต้อง กรุณาป้อนเป็นตัวเลข")
7         return
8
9     registrations = read_all_records_from_file()
10    found = False
11    remaining_records = []
12
13    for reg in registrations:
14        if reg and reg['ID'] == reg_id_to_delete:
15            found = True
16            print("ลบข้อมูลการลงทะเบียนสำเร็จ!")
17        elif reg:
18            remaining_records.append(reg)
19
20    if found:
21        try:
22            if os.path.exists(REGISTRATION_FILE_PATH):
23                os.remove(REGISTRATION_FILE_PATH)
24                for reg in remaining_records:
25                    record = create_registration_record(
26                        reg['ID'],
27                        reg['STUDENT ID'],
28                        reg['COURSE ID'],
29                        reg['REGISTRATION DATE'].timestamp(),
30                        1 if reg['STATUS'] == 'Registered' else 0
31                    )
32                    if record:
33                        write_record_to_file(record)
34        except IOError as e:
35            print(f"เกิดข้อผิดพลาดในการลบไฟล์: {e}")
36    else:
37        print("ไม่พบรหัส ID การลงทะเบียนที่ต้องการลบ")

```

ภาพที่ 3-14 โค้ดพังก์ชัน delete_registration(แสดงการลงทะเบียนแบบภาคร)

3.15 พังก์ชัน registration_menu(เมนูย่อยสำหรับการจัดการข้อมูลลงทะเบียน)

- การวนลูปและการแสดงเมนู
- while True:: ทำให้โปรแกรม แสดงเมนูและรอรับคำสั่งช้าๆ จนกว่าผู้ใช้จะเลือกออกจากเมนู
- print(...): แสดงตัวเลือกเมนูต่างๆ ให้ผู้ใช้เลือก ได้แก่:
- 1: เพิ่มข้อมูล (Create - add_registration())
- 2-4: ดูข้อมูล (Read - view_registrations(), view_single_registration(), view_filtered_registrations())
- 5: แก้ไขข้อมูล (Update - update_registration())
- 6: ลบข้อมูล (Delete - delete_registration())
- 0: ออกจากเมนู การรับและดำเนินการตามตัวเลือก
- choice = input("กรุณาเลือกเมนู: "): รับตัวเลือกจากผู้ใช้เป็นสตริง

```

● ● ●
1 def registration_menu():
2     """เมนูย่อยสำหรับจัดการข้อมูลการลงทะเบียน (CRUD)"""
3     while True:
4         print("\n===== ระบบจัดการข้อมูลการลงทะเบียน =====")
5         print("1. เพิ่มข้อมูลการลงทะเบียนทั้งหมด")
6         print("2. ดูข้อมูลการลงทะเบียนรายเดียว")
7         print("3. ดูข้อมูลการลงทะเบียนรายเดียว")
8         print("4. ดูข้อมูลการลงทะเบียนแบบกรอง")
9         print("5. แก้ไขข้อมูลการลงทะเบียน")
10        print("6. ลบข้อมูลการลงทะเบียน")
11        print("0. กลับสู่เมนูหลัก")
12
13        choice = input("กรุณาเลือกเมนู: ")
14
15        if choice == '1':
16            add_registration()
17        elif choice == '2':
18            view_registrations()
19        elif choice == '3':
20            view_single_registration()
21        elif choice == '4':
22            view_filtered_registrations()
23        elif choice == '5':
24            update_registration()
25        elif choice == '6':
26            delete_registration()
27        elif choice == '0':
28            print("ย้อนกลับสู่เมนูหลัก... ")
29            break
30        else:
31            print("ตัวเลือกไม่ถูกต้อง กรุณาลองใหม่อีกครั้ง")
32
33 if __name__ == "__main__":
34     registration_menu()

```

ภาพที่ 3-15 โค้ดพังก์ชัน registration_menu(เมนูย่อยสำหรับการจัดการข้อมูลลงทะเบียน)

4.ไฟล์ course.py

4.1 การกำหนดโครงสร้าง: กำหนดชื่อไฟล์เป็น CourseSubject.bin และตั้งรูปแบบการจัดเก็บข้อมูลด้วย COURSE_RECORD_FORMAT (<10s50sB H B B)

ซึ่งเป็นแม่แบบที่กำหนดขนาดและชนิดข้อมูลของแต่ละพิล็อต (รหัสวิชา, ชื่อวิชา, หน่วยกิต, ปีการศึกษา, ภาคเรียน, สถานะ).

4.2 พังก์ชัน create_course_record: ทำหน้าที่ "บรรจุหีบห่อ" ข้อมูล: แปลงข้อความ:

แปลงข้อมูลที่เป็นข้อความ (รหัสวิชา, ชื่อวิชา) ให้เป็นไบต์และ ปรับขนาด ให้ตรงตามแม่แบบ (Padding).
แพ็ค: ใช้ struct.pack() เพื่อรวมข้อมูลทั้งหมด (ทั้งข้อความและตัวเลข) ให้เป็น ก้อนไบนาเรียขนาดคงที่ ที่พร้อมบันทึกลงในไฟล์ฐานข้อมูล

```

● ● ●
1 import struct
2 import os
3
4 COURSE_FILE_NAME = 'CourseSubject.bin'
5 COURSE_RECORD_FORMAT = '<10s50sB H B B'
6 COURSE_RECORD_SIZE = struct.calcsize(COURSE_RECORD_FORMAT)
7
8 current_dir = os.path.dirname(os.path.abspath(__file__))
9 main_dir = os.path.dirname(current_dir)
10 COURSE_FILE_PATH = os.path.join(main_dir, COURSE_FILE_NAME)
11
12 def create_course_record(course_id, course_name, credit, academic_year, semester, is_active):
13     """สร้างบันทึกข้อมูลรายวิชาในรูปแบบไบนาเรีย"""
14     packed_course_id = course_id.encode('utf-8')[:10].ljust(10, b'\x00')
15     packed_course_name = course_name.encode('utf-8')[:50].ljust(50, b'\x00')
16     try:
17         record = struct.pack(
18             COURSE_RECORD_FORMAT,
19             packed_course_id,
20             packed_course_name,
21             credit,
22             academic_year,
23             semester,
24             is_active
25         )
26         return record
27     except struct.error as e:
28         print(f"เกิดข้อผิดพลาดในการแพ็คข้อมูล: {e}")
29     return None

```

ภาพที่ 4-1 ภาพโค้ดพังก์ชัน การกำหนดโครงสร้าง: กำหนดชื่อไฟล์เป็น CourseSubject.bin และ พังก์ชัน create_course_record

4.3 พังก์ชัน `read_course_record` (อ่านข้อมูล) พังก์ชันนี้ทำหน้าที่ แปลงบันทึกใบnaire ที่อ่านได้จากไฟล์ให้กลับเป็น ข้อมูลรายวิชาในรูปแบบ `Dictionary` ที่โปรแกรมนำไปใช้ต่อได้

หลักการทำงาน: ใช้ `struct.unpack()` เพื่อแกะข้อมูลตามโครงสร้างที่กำหนด (Format)

การแปลง: ลบค่าว่างที่เติมไว้ (`.strip(b'\x00')`) และ ถอดรหัส ข้อมูลที่เป็นไบต์กลับเป็นข้อความ (String)

ผลลัพธ์: คืนค่าเป็น `Dictionary` ที่มีข้อมูลรายวิชาที่อ่านง่าย รวมถึงแปลงรหัสสถานะ (1/0) เป็นข้อความ (Active/Inactive).

4.4 พังก์ชัน `write_record_to_file` (เขียนข้อมูล) พังก์ชันนี้ทำหน้าที่ บันทึกบันทึกใบnaire ที่ถูกสร้างไว้แล้วลงในไฟล์ฐานข้อมูลรายวิชา

หลักการทำงาน: เปิดไฟล์ `CourseSubject.bin` ในโหมด '`ab`' (Append Binary)

การบันทึก: ใช้ `f.write(record)` เพื่อ เพิ่ม (Append) บันทึกใบnaireใหม่ต่อท้ายไฟล์ที่มีอยู่เดิม

```

● ● ●

1 def read_course_record(record_data):
2     """อ่านบันทึกข้อมูลรายวิชาจากรูปแบบใบnaire"""
3     try:
4         unpacked_data = struct.unpack(COURSE_RECORD_FORMAT, record_data)
5         course_id = unpacked_data[0].strip(b'\x00').decode('utf-8')
6         course_name = unpacked_data[1].strip(b'\x00').decode('utf-8')
7         return {
8             'COURSE ID': course_id,
9             'COURSE NAME': course_name,
10            'CREDIT': unpacked_data[2],
11            'ACADEMIC YEAR': unpacked_data[3],
12            'SEMESTER': unpacked_data[4],
13            'STATUS': 'Active' if unpacked_data[5] == 1 else 'Inactive'
14        }
15    except struct.error as e:
16        print(f"เกิดข้อผิดพลาดในการอ่านแพ็คข้อมูล: {e}")
17        return None
18
19 def write_record_to_file(record, file_path=COURSE_FILE_PATH):
20     """เขียนบันทึกข้อมูลลงในไฟล์ใบnaire"""
21     try:
22         with open(file_path, 'ab') as f:
23             f.write(record)
24     except IOError as e:
25         print(f"เกิดข้อผิดพลาดในการเขียนไฟล์: {e}")
26
27 def read_all_records_from_file(file_path=COURSE_FILE_PATH):
28     """อ่านบันทึกข้อมูลทั้งหมดจากไฟล์ใบnaire"""

```

ภาพที่ 4-2 ไฟล์โค้ด `read_course_record` (อ่านข้อมูล) และ พังก์ชัน `write_record_to_file` (เขียนข้อมูล)

4.5 พังก์ชัน `read_all_records_from_file` (อ่านทั้งหมด) พังก์ชันนี้ ดึงข้อมูลรายวิชาทั้งหมด จากไฟล์ `CourseSubject.bin` โดยการวนอ่านข้อมูลที่ระบบล็อกตามขนาดที่กำหนด แล้วส่งแต่ละบล็อกไปแกะรหัสใบnaire ให้เป็น Dictionary ก่อนจะรวมทั้งหมดไว้ใน List.

4.6 พังก์ชัน `print_course_report` (แสดงรายงาน)

พังก์ชันนี้ จัดรูปแบบข้อมูลรายวิชา ที่อ่านได้ให้เป็น รายงานตาราง ที่สวยงามและเป็นระเบียบ โดยกำหนดความกว้างของคอลัมน์ให้คงที่ และพิมพ์รายงานออกทางหน้าจอ

```

● ● ●
1 def read_all_records_from_file(file_path=COURSE_FILE_PATH):
2     """อ่านบันทึกข้อมูลทั้งหมดจากไฟล์ใบnaire"""
3     records = []
4     try:
5         if not os.path.exists(file_path):
6             return records
7         with open(file_path, 'rb') as f:
8             while True:
9                 record_data = f.read(COURSE_RECORD_SIZE)
10                if not record_data:
11                    break
12                record = read_course_record(record_data)
13                if record:
14                    records.append(record)
15    except IOError as e:
16        print(f"เกิดข้อผิดพลาดในการอ่านไฟล์: {e}")
17    return records
18
19 def print_course_report(records, title="รายงานรายวิชา"):
20     """แสดงรายงานรายวิชาในรูปแบบตาราง"""
21     report = ""
22     report += "=====\n"
23     report += f"          {title}\n"
24     report += "=====\n"
25
26     headers = ["COURSE ID", "COURSE NAME", "CREDIT", "ACADEMIC YEAR", "SEMESTER", "STATUS"]
27     col_widths = [20, 20, 20, 15, 8, 15]
28
29     header_line = " | ".join(f"{{h:{col_widths[i]}}}" for i, h in enumerate(headers))
30     report += header_line + "\n"
31     report += "-" * len(header_line) + "\n"
32
33     for rec in records:
34         row_data = [str(rec[h]) for h in headers]
35         # ตัดข้อความหากยาวเกิน
36         for i in range(len(row_data)):
37             if len(row_data[i]) > col_widths[i]:
38                 row_data[i] = row_data[i][:col_widths[i]-3] + "..."
39         row_line = " | ".join(f"{{row_data[i]:{col_widths[i]}}}" for i in range(len(headers)))
40         report += row_line + "\n"
41
42     report += "-----\n"
43     print(report)
44

```

ภาพที่ 4-3 ไฟล์โค้ด พังก์ชัน `read_all_records_from_file` (อ่านทั้งหมด) และ แสดงรายงาน

4.7 พังก์ชัน add_course (เพิ่มข้อมูล)

รับข้อมูลวิชาจากผู้ใช้ และแปลงเป็นบинаรี ด้วย create_course_record จากนั้น บันทึกข้อมูลในไฟล์ CourseSubject.bin ด้วย write_record_to_file.

4.8 พังก์ชัน view_all_courses (แสดงทั้งหมด)

ดึงข้อมูลวิชาทั้งหมด จากไฟล์ในบинаรีด้วย read_all_records_from_file และจัดรูปแบบข้อมูลที่ได้ให้เป็นรายงานตารางบนหน้าจอด้วย print_course_report.

4.9 พังก์ชัน view_single_course (แสดงรายบุคคล)

รับรหัสวิชา ที่ต้องการค้นหา จากนั้น อ่านข้อมูลทั้งหมด และวนลูปค้นหา รหัสที่ตรงกัน หากพบจะแสดงรายละเอียดของรายวิชานั้นทันที

```

● ● ●

1 def view_all_courses():
2     """แสดงข้อมูลรายวิชาทั้งหมด"""
3     courses = read_all_records_from_file()
4     if not courses:
5         print("ไม่พบข้อมูลรายวิชาในระบบ")
6         return
7     print_course_report(courses, title="รายงานรายวิชา")
8
9 def view_single_course():
10    """แสดงข้อมูลรายวิชาเดียวตามรหัสวิชา"""
11    course_id = input("ป้อนรหัสวิชาที่ต้องการดู: ")
12    courses = read_all_records_from_file()
13    found = False
14
15    for course in courses:
16        if course and course['COURSE ID'] == course_id:
17            found = True
18            print("\n=====")
19            print("      ข้อมูลรายวิชาที่ค้นหา")
20            print("=====")
21            print(f"รหัสวิชา: {course['COURSE ID']} ")
22            print(f"ชื่อวิชา: {course['COURSE NAME']} ")
23            print(f"หน่วยกิต: {course['CREDIT']} ")
24            print(f"ปีการศึกษา: {course['ACADEMIC YEAR']} ")
25            print(f"ภาคเรียน: {course['SEMESTER']} ")
26            print(f"สถานะ: {course['STATUS']} ")
27            print("=====")
28            break
29
30    if not found:
31        print("ไม่พบรหัสวิชาที่ต้องการดู")

```

ภาพที่ 4-4 ไฟล์โค้ด พังก์ชัน add_course (เพิ่มข้อมูล) และ view_all_courses (แสดงทั้งหมด) และ view_single_course (แสดงรายบุคคล)

4.10. พังก์ชันนี้ทำหน้าที่ แสดงรายงานรายวิชาเฉพาะกลุ่ม ตามเงื่อนไขที่ผู้ใช้เลือก ดึงและเลือกเงื่อนไข: ดึง ข้อมูลรายวิชาทั้งหมด จากไฟล์ใบnairexam และแสดงเมนูให้ผู้ใช้เลือกเงื่อนไขการกรอง (ปีการศึกษา, ภาคเรียน, สถานะ Active, หรือรวมกัน). กรองข้อมูล: นำข้อมูลทั้งหมดมา วนลูปคัดเลือก บันทึกที่ตรงตามเงื่อนไขที่ผู้ใช้ป้อน และแสดงผล: หากพบรายการที่ตรงกัน จะส่งลิสต์ filtered_courses ไปให้พังก์ชัน print_course_report เพื่อแสดงผลเป็นรายงานตารางเฉพาะกลุ่มนั้นๆ

```

● ● ●
1 def view_filtered_courses():
2     """แสดงข้อมูลรายวิชาที่กรองตามเงื่อนไข"""
3     print("\n--- ดำเนินการกรอง ---")
4     print("1. กรองตามปีการศึกษา")
5     print("2. กรองตามภาคเรียน")
6     print("3. กรองตามสถานะ (Active)")
7     print("4. กรองตามปีการศึกษาและภาคเรียน")
8     print("5. กลับไปเมนูหลัก")
9     filter_choice = input("กรุณาเลือกการกรอง (1-5): ")
10
11    courses = read_all_records_from_file()
12    if not courses:
13        print("ไม่พบข้อมูลรายวิชาในระบบ")
14        return
15
16    filtered_courses = []
17
18    if filter_choice == '1':
19        try:
20            year = int(input("ป้อนปีการศึกษาที่ต้องการกรอง: "))
21            filtered_courses = [c for c in courses if c and c['ACADEMIC YEAR'] == year]
22        except ValueError:
23            print("ปีการศึกษาไม่ถูกต้อง")
24            return
25
26    elif filter_choice == '2':
27        try:
28            sem = int(input("ป้อนภาคเรียนที่ต้องการกรอง (1, 2, 3): "))
29            filtered_courses = [c for c in courses if c and c['SEMESTER'] == sem]
30        except ValueError:
31            print("ภาคเรียนไม่ถูกต้อง")
32            return
33
34    elif filter_choice == '3':
35        filtered_courses = [c for c in courses if c and c['STATUS'] == 'Active']
36
37    elif filter_choice == '4':
38        try:
39            year = int(input("ป้อนปีการศึกษาที่ต้องการกรอง: "))
40            sem = int(input("ป้อนภาคเรียนที่ต้องการกรอง (1, 2, 3): "))
41            filtered_courses = [c for c in courses if c and c['ACADEMIC YEAR'] == year and c['SEMESTER'] == sem]
42        except ValueError:
43            print("ข้อมูลกรองไม่ถูกต้อง")

```

ภาพที่ 4-5 ไฟล์โค้ด แสดงรายงานรายวิชาเฉพาะกลุ่ม ตามเงื่อนไขที่ผู้ใช้เลือก

4.11 พังก์ชันนี้ทำหน้าที่ แก้ไขข้อมูลรายวิชา โดยใช้กลยุทธ์ "ค้นหา-แก้ไข-เขียนใหม่ทั้งหมด" ค้นหาและแก้ไข: รับ รหัสวิชา ที่ต้องการแก้ไข จากนั้น อ่านข้อมูลวิชาทั้งหมด เข้ามาในหน่วยความจำ แล้ว วนลูป หานักศึกษาที่ตรงกัน เมื่อพบจะ แสดงข้อมูลเดิม และ รับค่าใหม่ (ชื่อวิชา, หน่วยกิต, ปี/ภาคเรียน, สถานะ) โดยอนุญาตให้ผู้ใช้กด Enter เพื่อใช้ค่าเดิมได้ สร้างและบันทึกไฟล์ใหม่ (Overwrite): เขียนไฟล์ใหม่: เมื่อแก้ไขข้อมูลในหน่วยความจำเสร็จแล้ว ระบบจะ เปิดไฟล์ CourseSubject.bin ในโหมด 'wb' (Write Binary) ซึ่งเป็นการลบข้อมูลเดิมทั้งหมดในไฟล์ บันทึกข้อมูล: จากนั้นจะวนลูป สร้างบันทึกใบนำร่อง จากข้อมูลที่อยู่ในหน่วยความจำ (รวมถึงรายการที่ถูกแก้ไข) แล้ว เขียนบันทึกทั้งหมด กลับลงในไฟล์

```

● ● ●
1 def update_course():
2     """แก้ไขข้อมูลรายวิชา"""
3     course_id_to_update = input("ป้อนรหัสวิชาที่ต้องการแก้ไข: ")
4     courses = read_all_records_from_file()
5     found = False
6     new_records = []
7
8     for course in courses:
9         if course['COURSE_ID'] == course_id_to_update:
10            found = True
11            print("====")
12            print("----บันทึกรายวิชาที่ต้องการแก้ไข----")
13            print("====")
14            print(f"รหัสวิชา: {course['COURSE_ID']}")
15            print(f"ชื่อวิชา: {course['COURSE_NAME']}")
16            print(f"หน่วยกิต: {course['CREDIT']}")
17            print(f"ภาคศึกษา: {course['ACADEMIC_YEAR']}")
18            print(f"ภาคเรียน: {course['SEMESTER']}")
19            print(f"สถานะ: {course['STATUS']}")
20            print("====")
21
22            new_name = input("ป้อนชื่อวิชาใหม่ (Enter เพื่อใช้ค่าเดิม): ")
23            if new_name:
24                course['COURSE_NAME'] = new_name
25
26            new_credit = input("ป้อนหน่วยกิตใหม่ (Enter เพื่อใช้ค่าเดิม): ")
27            if new_credit:
28                try:
29                    course['CREDIT'] = int(new_credit)
30                except ValueError:
31                    print("หน่วยกิตไม่ถูกต้อง ใช้ค่าเดิม")
32
33            new_academic_year = input("ป้อนภาคศึกษาใหม่ (Enter เพื่อใช้ค่าเดิม): ")
34            if new_academic_year:
35                try:
36                    course['ACADEMIC_YEAR'] = int(new_academic_year)
37                except ValueError:
38                    print("ภาคศึกษาไม่ถูกต้อง ใช้ค่าเดิม")
39
40            new_semester = input("ป้อนภาคเรียนใหม่ (Enter เพื่อใช้ค่าเดิม): ")
41            if new_semester:
42                try:
43                    course['SEMESTER'] = int(new_semester)
44                except ValueError:
45                    print("ภาคเรียนไม่ถูกต้อง ใช้ค่าเดิม")
46
47            new_active = input("ป้อนสถานะใหม่ (1 = Active, 0 = Inactive) (Enter เพื่อใช้ค่าเดิม): ")
48            if new_active:
49                try:
50                    new_active_int = int(new_active)
51                    if new_active_int in [0, 1]:
52                        course['STATUS'] = 'Active' if new_active_int == 1 else 'Inactive'
53                    else:
54                        print("สถานะไม่ถูกต้อง ใช้ค่าเดิม")
55                except ValueError:
56                    print("สถานะไม่ถูกต้อง ใช้ค่าเดิม")
57
58            updated_record = create_course_record(
59                course['COURSE_ID'],
60                course['COURSE_NAME'],
61                course['CREDIT'],
62                course['ACADEMIC_YEAR'],
63                course['SEMESTER'],
64                1 if course['STATUS'] == 'Active' else 0
65            )
66            if updated_record:
67                new_records.append(updated_record)
68
69        if found:
70            try:
71                with open(COURSE_FILE_PATH, 'wb') as f:
72                    for record in new_records:
73                        f.write(record)
74                    print("แก้ไขข้อมูลสำเร็จ!")
75            except IOError as e:
76                print(f"เกิดข้อผิดพลาดในการแก้ไขไฟล์: {e}")
77        else:
78            print("ไม่พบรหัสวิชาที่ต้องการแก้ไข")

```

ภาพที่ 4-6 โค้ดไฟล์พังก์ชันนี้ทำหน้าที่ แก้ไขข้อมูลรายวิชา

4.12 ฟังก์ชัน `delete_course` (ลบข้อมูล) ฟังก์ชันนี้ทำหน้าที่ ลบบันทึกรายวิชา ออกจากฐานข้อมูลการ หลักการทำงาน: รับ รหัสวิชา ที่ต้องการลบ จากนั้น อ่านข้อมูลทั้งหมด เข้ามาในหน่วยความจำ แล้ว คัดแยก บันทึกที่ต้องการลบออกไป เมื่อเสร็จแล้ว ระบบจะ เขียนบันทึกใบหนารีที่เหลือทั้งหมด กลับลงในไฟล์ `CourseSubject.bin` ใหม่ โดยเป็นการเขียนทับไฟล์เดิม.

4.13 ฟังก์ชัน `course_menu` (เมนูย่อยรายวิชา) ฟังก์ชันนี้ทำหน้าที่เป็น หน้าต่างควบคุมหลัก สำหรับการ จัดการข้อมูลรายวิชา

หลักการทำงาน: แสดงเมนูย่อย 6 ตัวเลือก (เพิ่ม, ดูข้อมูลรายวิชาทั้งหมด, แก้ไข, ลบ) ให้ผู้ใช้เลือก และทำ หน้าที่เป็น ตัวนำทาง เพื่อ เรียกใช้ฟังก์ชันย่อย ที่เกี่ยวข้อง (`add_course`, `update_course`, `delete_course` ฯลฯ) เพื่อดำเนินการตามคำสั่ง

```

1 def course_menu():
2     """แสดงเมนูหลักและรับตัวเลือกจากผู้ใช้"""
3     while True:
4         print("\n===== ระบบจัดการรายวิชา =====")
5         print("1. เพิ่มข้อมูลรายวิชา")
6         print("2. ดูข้อมูลรายวิชาทั้งหมด")
7         print("3. ดูข้อมูลรายวิชารายบุคคล")
8         print("4. ดูข้อมูลรายวิชาแบบกรอง")
9         print("5. แก้ไขข้อมูลรายวิชา")
10        print("6. ลบข้อมูลรายวิชา")
11        print("0. กลับสู่เมนูหลัก")
12        choice = input("กรุณาเลือกเมนู (1-0): ")
13
14        if choice == '1':
15            add_course()
16        elif choice == '2':
17            view_all_courses()
18        elif choice == '3':
19            view_single_course()
20        elif choice == '4':
21            view_filtered_courses()
22        elif choice == '5':
23            update_course()
24        elif choice == '6':
25            delete_course()
26        elif choice == '0':
27            print("ย้อนกลับสู่เมนูหลัก...")
28            break
29        else:
30            print("ตัวเลือกไม่ถูกต้อง กรุณาลองอีกครั้ง")
31
32 if __name__ == "__main__":
33     course_menu()

```

ภาพที่ 4-7 โค้ดไฟล์ ฟังก์ชัน `delete_course` (ลบข้อมูล) และ ฟังก์ชัน `course_menu`

5.ไฟล์ report.py

- 5.1 การกำหนดพาร (Path Definition):กำหนดชื่อไฟล์ฐานข้อมูลหลัก 3 ไฟล์ (student.bin, registration.bin, CourseSubject.bin).
- กำหนด ตำแหน่งที่ตั้งของไฟล์ (_FILE_PATH) โดยคำนวนพารเพื่อให้แน่ใจว่าไฟล์ฐานข้อมูลและไฟล์รายงานจะถูกบันทึกอยู่ใน โฟลเดอร์หลัก (main directory) ของโครงการ กำหนดชื่อและตำแหน่งของ ไฟล์รายงานผลลัพธ์ (_report.txt).
- 5.2 การกำหนดรูปแบบใบหน้า (Format Definition):
- กำหนด รูปแบบ (Format String) และ ขนาด (Size) ที่แน่นอนของบันทึกข้อมูลใบหน้าสำหรับ ทั้ง 3 Module (นักศึกษา, การลงทะเบียน, รายวิชา)รูปแบบเหล่านี้มีความสำคัญอย่าง
- 5.3 การกำหนดค่าคงที่อื่น ๆ:
- นำเข้า defaultdict สำหรับใช้ในการจัดการกลุ่มข้อมูลระหว่างสร้างรายงาน (เช่น การจัดกลุ่มตาม รายวิชา/นักศึกษา). กำหนด STATUS_MAPPING เพื่อใช้แปลงรหัสสถานะตัวเลข (1, 0) ให้เป็น ข้อความที่มนุษย์เข้าใจได้ (ลงทะเบียน, ถอน)

```

● ● ●
1 import os
2 import struct
3 import datetime
4 from collections import defaultdict
5
6 # -----
7 # Path และ Format
8 # -----
9 STUDENT_FILE_NAME = 'student.bin'
10 REGISTER_FILE_NAME = 'registration.bin'
11 COURSE_FILE_NAME = 'CourseSubject.bin'
12
13 current_dir = os.path.dirname(os.path.abspath(__file__))
14 main_dir = os.path.dirname(current_dir) # ขึ้นไป main/
15 STUDENT_FILE_PATH = os.path.join(main_dir, STUDENT_FILE_NAME)
16 REGISTER_FILE_PATH = os.path.join(main_dir, REGISTER_FILE_NAME)
17 COURSE_FILE_PATH = os.path.join(main_dir, COURSE_FILE_NAME)
18
19 # Report ในอุปกรณ์ main/
20 REPORT_STUDENT_FILE_PATH = os.path.join(main_dir, "student_report.txt")
21 REPORT_REGISTER_FILE_PATH = os.path.join(main_dir, "register_report.txt")
22
23 # Student Format
24 STUDENT_RECORD_FORMAT = '<16s50s50s20sBB'
25 STUDENT_RECORD_SIZE = struct.calcsize(STUDENT_RECORD_FORMAT)
26
27 # Register Format
28 REGISTER_RECORD_FORMAT = '<I16s16sdB'
29 REGISTER_RECORD_SIZE = struct.calcsize(REGISTER_RECORD_FORMAT)
30
31 # Course Format
32 COURSE_RECORD_FORMAT = '<10s50sB H B B'
33 COURSE_RECORD_SIZE = struct.calcsize(COURSE_RECORD_FORMAT)
34
35 STATUS_MAPPING = {1: 'ลงทะเบียน', 0: 'ถอน'}

```

ภาพที่ 5-1 ไฟล์โค้ด การกำหนดพาร (Path Definition) การกำหนดรูปแบบใบหน้าและ ตั้งค่าอื่นๆ

- 5.4 พังก์ชัน `read_student_record` (แกะรหัสนักศึกษา) พังก์ชันนี้ทำหน้าที่ แปลงบันทึกใบnaire ของนักศึกษาให้กลับเป็น ข้อมูล Dictionary ที่อ่านได้
- หลักการทำงาน: ใช้ `struct.unpack()` เพื่อแกะข้อมูลตามโครงสร้าง จากนั้น ถอดรหัส ข้อมูลที่เป็น ข้อความ (`.decode('utf-8')`) และลบซองว่าง (`.strip(b'\x00')`).
- ผลลัพธ์: คืนค่าเป็น Dictionary พร้อม แปลงรหัสสถานะตัวเลข ให้เป็นข้อความ (เช่น 1 → 'ลงทะเบียน') โดยใช้ `STATUS_MAPPING`.
- 5.5 พังก์ชัน `read_all_students` (ดึงข้อมูลทั้งหมด) พังก์ชันนี้ทำหน้าที่ ดึงข้อมูลนักศึกษาทั้งหมด จากไฟล์ `student.bin`
- หลักการทำงาน: เปิดไฟล์ใบnaire ('rb') และ วนลูปอ่านข้อมูล ทีละบล็อกตามขนาดบันทึก (`STUDENT_RECORD_SIZE`) แต่ละบล็อกที่อ่านได้จะถูกส่งไปให้ `read_student_record` เพื่อแกะ รหัส ก่อนจะ รวมรวม ข้อมูลทั้งหมดไว้ใน List

```

● ● ●
1 def read_student_record(record_data):
2     try:
3         unpacked_data = struct.unpack(STUDENT_RECORD_FORMAT, record_data)
4         student_id = unpacked_data[0].strip(b'\x00').decode('utf-8')
5         first_name = unpacked_data[1].strip(b'\x00').decode('utf-8')
6         last_name = unpacked_data[2].strip(b'\x00').decode('utf-8')
7         major = unpacked_data[3].strip(b'\x00').decode('utf-8')
8         year_level = unpacked_data[4]
9         status_code = unpacked_data[5]
10
11     return {
12         'STUDENT ID': student_id,
13         'FIRST NAME': first_name,
14         'LAST NAME': last_name,
15         'MAJOR': major,
16         'YEAR': year_level,
17         'STATUS': STATUS_MAPPING.get(status_code, 'ไม่ทราบ')
18     }
19 except struct.error:
20     return None
21
22 def read_all_students(file_path=STUDENT_FILE_PATH):
23     records = []
24     if not os.path.exists(file_path):
25         return records
26     with open(file_path, 'rb') as f:
27         while True:
28             record_data = f.read(STUDENT_RECORD_SIZE)
29             if not record_data:
30                 break
31             record = read_student_record(record_data)
32             if record:
33                 records.append(record)
34     return records

```

ภาพที่ 5-2 ไฟล์โค้ด พังก์ชัน `read_student_record` และ พังก์ชัน `read_all_students`

- 5.6 พังก์ชัน `read_course_record` (รหัสวิชา) พังก์ชันนี้ทำหน้าที่ แปลงบันทึกใบหนารี ของรายวิชา ให้กลับเป็น ข้อมูล Dictionary ที่อ่านและนำไปใช้ได้
- หลักการทำงาน: ใช้ `struct.unpack()` เพื่อแยกรหัสใบหนารีตามโครงสร้างที่กำหนด และ ถอดรหัส ข้อมูลที่เป็นข้อความให้เป็น String.
- ผลลัพธ์: คืนค่าเป็น Dictionary ที่มีรายละเอียดของรายวิชา.
- 5.7 พังก์ชัน `load_course_dict` (โหลดข้อมูลทั้งหมด) พังก์ชันนี้ทำหน้าที่ ดึงข้อมูลรายวิชาทั้งหมด จากไฟล์ `CourseSubject.bin` แล้วจัดเก็บในรูปแบบ Dictionary
- หลักการทำงาน: เปิดไฟล์ใบหนารีและ วนลูปอ่านข้อมูล ทีละบล็อก แต่ละบล็อกที่อ่านได้จะถูกส่งไปให้ `read_course_record` เพื่อแยกรหัส
- การจัดเก็บ: จัดเก็บข้อมูลรายวิชาที่แยกรหัสแล้วใน Dictionary โดยใช้ รหัสวิชา (`course_id`) เป็นคีย์ เพื่อให้ง่ายและรวดเร็วต่อการค้นหาในภายหลัง

```

● ● ●

1 def read_course_record(record_data):
2     try:
3         unpacked_data = struct.unpack(COURSE_RECORD_FORMAT, record_data)
4         course_id = unpacked_data[0].strip(b'\x00').decode('utf-8')
5         course_name = unpacked_data[1].strip(b'\x00').decode('utf-8')
6         return {
7             'course_id': course_id,
8             'course_name': course_name,
9             'credit': unpacked_data[2],
10            'academic_year': unpacked_data[3],
11            'semester': unpacked_data[4],
12            'is_active': unpacked_data[5]
13        }
14    except struct.error:
15        return None
16
17 def load_course_dict(file_path=COURSE_FILE_PATH):
18     courses = {}
19     if not os.path.exists(file_path):
20         return courses
21     with open(file_path, 'rb') as f:
22         while True:
23             record_data = f.read(COURSE_RECORD_SIZE)
24             if not record_data:
25                 break
26             course = read_course_record(record_data)
27             if course:
28                 courses[course['course_id']] = course
29     return courses

```

ภาพที่ 5-3 ไฟล์โค้ด พังก์ชัน `read_course_record` และ พังก์ชัน `load_course_dict`

- 5.8 พังก์ชัน `read_register_record` (รหัสการลงทะเบียน) พังก์ชันนี้ทำหน้าที่ แปลงบันทึกใบนา蕊 ของการลงทะเบียน (Registration) ให้กลับเป็น ข้อมูล Dictionary
- หลักการทำงาน: ใช้ `struct.unpack()` เพื่อแกะข้อมูลตามโครงสร้างที่กำหนด
- การแปลง: แปลงวันที่ (`unpacked_data[3]`) ที่อยู่ในรูปแบบ Timestamp ให้เป็นวัตถุ `datetime` และใช้ `STATUS_MAPPING` เพื่อแปลงรหัสสถานะตัวเลขให้เป็นข้อความที่เข้าใจง่าย (ลงทะเบียน/ ถอน).
- ผลลัพธ์: คืนค่าเป็น Dictionary ที่มีรายละเอียดการลงทะเบียนพร้อมรหัสวิชาและรหัสนักศึกษา.
- 5.9 พังก์ชัน `read_all_registrations` (ดึงข้อมูลทั้งหมด) พังก์ชันนี้ทำหน้าที่ ดึงบันทึกการลงทะเบียน ทั้งหมด จากไฟล์ `registration.bin`
- หลักการทำงาน: เปิดไฟล์ใบนา蕊 ('rb') และ วนลูปอ่านข้อมูล ที่ลับล็อกตามขนาดบันทึก (`REGISTER_RECORD_SIZE`) แต่ละบล็อกที่อ่านได้จะถูกส่งไปให้ `read_register_record` เพื่อแกะ รหัส ก่อนจะ รวบรวม ข้อมูลทั้งหมดไว้ใน List

```

1  def read_register_record(record_data):
2      try:
3          unpacked_data = struct.unpack(REGISTER_RECORD_FORMAT, record_data)
4          reg_id = unpacked_data[0]
5          student_id = unpacked_data[1].strip(b'\x00').decode('utf-8')
6          course_id = unpacked_data[2].strip(b'\x00').decode('utf-8')
7          reg_date = datetime.datetime.fromtimestamp(unpacked_data[3])
8          status = unpacked_data[4]
9          return {
10              'REGISTER_ID': reg_id,
11              'STUDENT_ID': student_id,
12              'COURSE_ID': course_id,
13              'DATE': reg_date,
14              'STATUS': STATUS_MAPPING.get(status, 'ไม่ทราบ'),
15              'STATUS_CODE': status
16          }
17      except struct.error:
18          return None
19
20 def read_all_registrations(file_path=REGISTER_FILE_PATH):
21     records = []
22     if not os.path.exists(file_path):
23         return records
24     with open(file_path, 'rb') as f:
25         while True:
26             record_data = f.read(REGISTER_RECORD_SIZE)
27             if not record_data:
28                 break
29             record = read_register_record(record_data)
30             if record:
31                 records.append(record)
32     return records

```

ภาพที่ 5-4 ไฟล์โค้ด พังก์ชัน `read_register_record` และ พังก์ชัน `read_all_registrations`

- 5.10 พัฟ์ชัน print_student_report (รายงานนักศึกษา) พัฟ์ชันนี้ทำหน้าที่ สร้างและแสดงรายงาน สรุปข้อมูลนักศึกษา อย่างละเอียด
- สร้างตาราง: นำข้อมูลนักศึกษา (List ของ Dictionary) ที่ได้รับมา จัดรูปแบบเป็น ตาราง ที่สวยงาม และจัดเรียงคอลัมน์ให้อ่านง่าย โดยมีหัวข้อหลักคือ STUDENT ID, ชื่อ, สาขา, และสถานะ
- คำนวณสถิติ: ใช้ defaultdict เพื่อ นับจำนวน นักศึกษา โดย จัดกลุ่ม ตาม สาขาวิชา และ ชั้นปี รวมถึงนับแยกตามสถานะ
- สรุปผล: พิมพ์รายงานสรุปทั้งหมดออกทางหน้าจอ โดยแสดง จำนวนรวม และสถิติที่คำนวณได้ เช่น จำนวนนักศึกษาแยกตามสาขา, แยกตามชั้นปี, และระบุ ชั้นปีที่มีนักศึกษามากที่สุด

```

● ● ●
1 def print_student_report(records):
2     report = ""
3     report += "=====\\n"
4     report += "          รายงานนักศึกษา\\n"
5     report += "=====\\n"
6
7     headers = ["STUDENT ID", "FIRST NAME", "LAST NAME", "MAJOR", "YEAR", "STATUS"]
8     col_widths = [20, 20, 20, 15, 8, 15]
9
10    header_line = " | ".join(f"{h:<{col_widths[i]}}" for i, h in enumerate(headers))
11    report += header_line + "\\n"
12    report += "-" * len(header_line) + "\\n"
13
14    for rec in records:
15        row_data = [str(rec[h]) for h in ['STUDENT ID', 'FIRST NAME', 'LAST NAME', 'MAJOR', 'YEAR', 'STATUS']]
16        row_line = " | ".join(f"{row_data[i]:<{col_widths[i]}}" for i in range(len(headers)))
17        report += row_line + "\\n"
18
19    report += "-----\\n"
20    report += f"จำนวนนักศึกษาทั้งหมด: {len(records)}\\n"
21
22    major_count = defaultdict(int)
23    year_count = defaultdict(int)
24    status_count = defaultdict(int)
25
26    for rec in records:
27        major_count[rec['MAJOR']] += 1
28        year_count[rec['YEAR']] += 1
29        status_count[rec['STATUS']] += 1
30
31    report += "\\n--- สถิตินักศึกษา ---\\n"
32
33    # สรุปตามสาขา
34    report += "นักศึกษาแยกตามสาขา:\\n"
35    for major, count in major_count.items():
36        report += f" - สาขา {major}: {count} คน\\n"
37
38    # เพิ่ม: สรุปตามชั้นปี
39    report += "\\nนักศึกษาแยกตามชั้นปี:\\n"
40    for year in sorted(year_count.keys()): # เรียงลำดับชั้นปีเพื่อความชัดเจน
41        report += f" - ชั้นปี {year}: {year_count[year]} คน\\n"
42
43    # ชั้นปีที่มีนักศึกษามากที่สุด
44    if year_count:
45        max_year = max(year_count, key=year_count.get)
46        report += f"\\n- ชั้นปีที่มีนักศึกษามากที่สุด: ชั้นปี {max_year} ({year_count[max_year]} คน)\\n"
47
48    print(report)
49    return report

```

ภาพที่ 5-5 ไฟล์ได้ พัฟ์ชัน print_student_report (รายงานนักศึกษา)

- 5.11 พัฟ์ชัน analyze_registration_statistics พัฟ์ชันนี้ทำหน้าที่ คำนวณสถิติเชิงลึก ของการลงทะเบียน:
 - ประมวลผล: วนลูปข้อมูลการลงทะเบียนทั้งหมด เพื่อ นับยอดลงทะเบียนและยอดถอน โดยแยกกลุ่ม ตาม รายวิชา, สาขา, และ ชั้นปี ของนักศึกษา
 - คำนวณ: คำนวณ อัตราการถอน (Drop Rate) ของแต่ละรายวิชา.
 - จัดอันดับ: จัดเรียงผลลัพธ์เพื่อหา รายวิชายอดนิยม (ลงทะเบียนสูงสุด) และ รายวิชาที่มีอัตราการถอนสูงสุด

```

● ● ●
1 def analyze_registration_statistics(records, courses, students):
2     """วิเคราะห์สถิติการลงทะเบียนแบบละเอียด"""
3
4     student_dict = {s['STUDENT_ID']: s for s in students}
5
6     stats = {
7         'course_stats': defaultdict(lambda: {'registered': 0, 'dropped': 0, 'students': []}),
8         'major_stats': defaultdict(lambda: {'registered': 0, 'dropped': 0}),
9         'year_stats': defaultdict(lambda: {'registered': 0, 'dropped': 0}),
10        'date_stats': defaultdict(int),
11        'popular_courses': [],
12        'drop_rates': []
13    }
14
15    for rec in records:
16        course_id = rec['COURSE_ID']
17        status = rec['STATUS_CODE']
18        student_id = rec['STUDENT_ID']
19        date_key = rec['DATE'].strftime("%Y-%m-%d")
20
21        student_info = student_dict.get(student_id, {})
22        major = student_info.get('MAJOR', 'ไม่ระบุ')
23        year = student_info.get('YEAR', 'ไม่ระบุ')
24
25        if status == 1:
26            stats['course_stats'][course_id]['registered'] += 1
27            stats['course_stats'][course_id]['students'].append(student_id)
28        else:
29            stats['course_stats'][course_id]['dropped'] += 1
30
31        if status == 1:
32            stats['major_stats'][major]['registered'] += 1
33        else:
34            stats['major_stats'][major]['dropped'] += 1
35
36        if status == 1:
37            stats['year_stats'][year]['registered'] += 1
38        else:
39            stats['year_stats'][year]['dropped'] += 1
40
41        if status == 1:
42            stats['date_stats'][date_key] += 1
43
44    for course_id, course_data in stats['course_stats'].items():
45        total = course_data['registered'] + course_data['dropped']
46        drop_rate = (course_data['dropped'] / total * 100) if total > 0 else 0
47
48        course_info = courses.get(course_id, {})
49        course_name = course_info.get('course_name', 'ไม่ระบุ')
50
51        stats['popular_courses'].append({
52            'course_id': course_id,
53            'course_name': course_name,
54            'registered': course_data['registered'],
55            'dropped': course_data['dropped'],
56            'total': total,
57            'drop_rate': drop_rate
58        })
59
60        stats['drop_rates'].append({
61            'course_id': course_id,
62            'course_name': course_name,
63            'drop_rate': drop_rate,
64            'dropped': course_data['dropped'],
65            'registered': course_data['registered']
66        })
67
68    stats['popular_courses'].sort(key=lambda x: x['registered'], reverse=True)
69    stats['drop_rates'].sort(key=lambda x: x['drop_rate'], reverse=True)
70
71    return stats

```

ภาพที่ 5-6 ไฟล์โค้ด พัฟ์ชัน analyze_registration_statistic

- 5.12 พัฟ์ชัน print_register_report (รายงานการลงทะเบียน)

ฟังก์ชันนี้ทำหน้าที่ สร้างรายงานสรุปการลงทะเบียนฉบับเต็ม โดยใช้ผลการวิเคราะห์จาก analyze_registration_statistics

ประมวลผล: เรียก analyze_registration_statistics เพื่อรับสถิติเชิงลึกทั้งหมด (ยอดลงทะเบียน/ถอน แยก ตามวิชา/สาขา/ชั้นปี)

- ส่วนที่ 1: รายละเอียดรายวิชา: วนลูป จัดกลุ่ม ข้อมูลการลงทะเบียน (เฉพาะสถานะ 'ลงทะเบียน') แยกตามรายวิชา แต่ละวิชาจะถูกแสดงในรูปแบบ ตารางย่ออย โดยระบุว่า哪กศึกษาคนใดลงทะเบียน วิชานั้นบ้าง พร้อมสรุปสถิติอย่างของวิชานั้น (เช่น อัตราการถอน, ชั้นปีที่ลงทะเบียนมากที่สุด)
- ส่วนที่ 2: การวิเคราะห์เชิงสถิติ: นำผลลัพธ์สถิติที่ได้มา แสดงผล ในรูปแบบสรุป: จัดอันดับ วิชายอดนิยม และ วิชาที่มีอัตราการถอนสูงสุด และแสดงสถิติการลงทะเบียนแยกตาม สาขา และ ชั้นปี
- แสดง วันที่มีการลงทะเบียนสูงสุด
- สรุปภาพรวม: แสดง ยอดรวม การลงทะเบียนและการถอนทั้งหมด พร้อมทั้ง อัตราการถอนโดยรวม ของระบบ

```

● ● ●
1 def print_register_report(records, courses, students):
2     report = ""
3     report += "=====รายงานการลงทะเบียน=====\n"
4     report += "  รายงานการลงทะเบียน\n"
5     report += "=====รายงานการลงทะเบียน=====\n"
6     report += f"สร้างเมื่อ: {datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n\n"
7
8     stats = analyze_registration_statistics(records, courses, students)
9     student_dict = {s['STUDENT ID']: s for s in students}
10
11    course_groups = defaultdict(list)
12    for rec in records:
13        if rec['STATUS_CODE'] == 1:
14            course_groups[rec['COURSE ID']].append(rec)
15
16    for course_id, reg_list in course_groups.items():
17        course_info = courses.get(course_id, {})
18        course_name = course_info.get('course_name', 'ไม่ระบุ')
19        academic_year = course_info.get('academic_year', 'ไม่ระบุ')
20        semester = course_info.get('semester', 'ไม่ระบุ')
21
22        report += f"วิชา: {course_id} - {course_name} [ปีการศึกษา {academic_year}, ภาคเรียน {semester}]\n"
23        report += "  จำนวน: 1\n\n"
24
25    headers = ["STUDENT ID", "FIRST NAME", "LAST NAME", "MAJOR", "YEAR", "REGISTRATION DATE", "STATUS"]
26    col_widths = [20, 20, 20, 15, 8, 20, 15]
27
28    header_line = " | ".join(f"{{h:<{col_widths[i]}}}" for i, h in enumerate(headers))
29    report += header_line + "\n"
30    report += "-" * len(header_line) + "\n"
31
32    for rec in reg_list:
33        student_info = student_dict.get(rec['STUDENT ID'], {})
34        row_data = [
35            rec["STUDENT ID"],
36            student_info.get('FIRST NAME', 'ไม่ระบุ'),
37            student_info.get('LAST NAME', 'ไม่ระบุ'),
38            student_info.get('MAJOR', 'ไม่ระบุ'),
39            str(student_info.get('YEAR', 'ไม่ระบุ')),
40            rec["DATE"].strftime("%Y-%m-%d"),
41            rec["STATUS"]
42        ]
43        row_line = " | ".join(f"{{row_data[i]}:<{col_widths[i]}}" for i in range(len(headers)))
44        report += row_line + "\n"

```

ภาพที่ 5-7 ไฟล์โค้ด พัฟ์ชัน print_register_report (รายงานการลงทะเบียน)

```

1 course_stat = stats['course_stats'][course_id]
2     total_registered = course_stat['registered']
3     total_dropped = course_stat['dropped']
4     total_students = total_registered + total_dropped
5     drop_rate = (total_dropped / total_students * 100) if total_students > 0 else 0
6
7     report += f"\nจำนวนนักศึกษาทั้งหมดในส่วนนี้: {total_registered}\n"
8
9     major_count = defaultdict(int)
10    year_count = defaultdict(int)
11    date_count = defaultdict(int)
12
13    for rec in reg_list:
14        student_info = student_dict.get(rec['STUDENT_ID'], {})
15        major = student_info.get('MAJOR', 'ไม่ระบุ')
16        year = student_info.get('YEAR', 'ไม่ระบุ')
17        date = rec["DATE"].strftime("%Y-%m-%d")
18
19        major_count[major] += 1
20        year_count[year] += 1
21        date_count[date] += 1
22
23        report += "- นักศึกษาแยกตามสาขา:\n"
24        for major, count in major_count.items():
25            report += f"  {major}: {count}\n"
26
27        report += "\n--- สรุปสถานะ ---\n"
28        report += f"- ลงทะเบียน: {total_registered}\n"
29        report += f"- ถอน: {total_dropped}\n"
30        report += f"- อัตราการถอน: {drop_rate:.1f}%\n"
31
32        if year_count:
33            max_year = max(year_count, key=year_count.get)
34            report += f"\n- ชั้นปีที่มีการลงทะเบียนมากที่สุด: ปี {max_year} [{year_count[max_year]} คน]\n"
35
36        if major_count:
37            max_major = max(major_count, key=major_count.get)
38            min_major = min(major_count, key=major_count.get)
39            report += f"- สาขาวิชาที่มีการลงทะเบียนมากที่สุด: {max_major} [{major_count[max_major]} คน]\n"
40            report += f"- สาขาวิชาที่มีการลงทะเบียนน้อยที่สุด: {min_major} [{major_count[min_major]} คน]\n"
41
42        if date_count:
43            max_date = max(date_count, key=date_count.get)
44            report += f"- วันที่ที่มีการลงทะเบียนมากที่สุด: {max_date} [{date_count[max_date]} คน]\n"
45
46        report += "\n" + "="*80 + "\n\n"
47
48 # คำนวณ total_registrations ก่อนใช้งาน
49 total_registrations = len([r for r in records if r['STATUS_CODE'] == 1])

```

ภาพที่ 5-8 ไฟล์คัตต์ฟังก์ชัน print_register_report (รายงานการลงทะเบียน)

```

● ● ●
1 report += "📊 การวิเคราะห์สถิติการลงทะเบียนแบบละเอียด\n"
2 report += "="*80 + "\n\n"
3
4 report += "💡 วิชาข้อต้นยิ่ง (เรียงตามจำนวนผู้ลงทะเบียน):\n"
5 report += "="*60 + "\n"
6 for i, course in enumerate(stats['popular_courses'][:10], 1):
7     report += f"{i}. {course['course_id']} - {course['course_name']}\n"
8     report += f"    📊 ลงทะเบียน: {course['registered']} คน, ✗ ถอน: {course['dropped']} คน, "
9     report += f"ทั้งหมด: {course['total']} คน, อัตราการถอน: {course['drop_rate']:.1f}%\n\n"
10
11 report += "⚠️ วิชาที่มีอัตราการถอนสูงที่สุด:\n"
12 report += "="*60 + "\n"
13 for i, course in enumerate(stats['drop_rates'][:5], 1):
14     if course['drop_rate'] > 0:
15         report += f"{i}. {course['course_id']} - {course['course_name']}\n"
16         report += f"    📊 อัตราการถอน: {course['drop_rate']:.1f}%" "
17         report += f"({course['dropped']} จาก {course['dropped'] + course['registered']} คน)\n\n"
18
19 report += "🌐 สถิติการลงทะเบียนแยกตามสาขา:\n"
20 report += "="*60 + "\n"
21 for major, data in stats['major_stats'].items():
22     total = data['registered'] + data['dropped']
23     drop_rate = (data['dropped'] / total * 100) if total > 0 else 0
24     report += f"- {major}: ลงทะเบียน {data['registered']} คน, ถอน {data['dropped']} คน "
25     report += f"({อัตราการถอน: {drop_rate:.1f}}%)\n"
26 report += "\n"
27
28 report += "🕒 สถิติการลงทะเบียนแยกตามปี:\n"
29 report += "="*60 + "\n"
30 for year, data in sorted(stats['year_stats'].items()):
31     total = data['registered'] + data['dropped']
32     drop_rate = (data['dropped'] / total * 100) if total > 0 else 0
33     report += f"- {year}: ลงทะเบียน {data['registered']} คน, ถอน {data['dropped']} คน "
34     report += f"({อัตราการถอน: {drop_rate:.1f}}%)\n"
35 report += "\n"
36
37 report += "📍 รันที่มีการลงทะเบียนสูงสุด (5 อันดับแรก):\n"
38 report += "="*60 + "\n"
39 sorted_dates = sorted(stats['date_stats'].items(), key=lambda x: x[1], reverse=True)
40 for i, (date, count) in enumerate(sorted_dates[:5], 1):
41     report += f"{i}. {date}: {count}\n"
42 report += "\n"
43
44 total_registered = sum(course['registered'] for course in stats['popular_courses'])
45 total_dropped = sum(course['dropped'] for course in stats['popular_courses'])
46 overall_drop_rate = (total_dropped / (total_registered + total_dropped) * 100) if (total_registered + total_dropped) > 0 else 0
47
48 report += "📝 สรุปภาพรวมทั้งหมด:\n"
49 report += "="*60 + "\n"
50 report += f"- จำนวนวิชาที่มีผลลัพธ์: {len(stats['popular_courses'])} วิชา\n"
51 report += f"- จำนวนการลงทะเบียนทั้งหมด: {total_registered} คน\n"
52 report += f"- จำนวนการถอนทั้งหมด: {total_dropped} คน\n"
53 report += f"- อัตราการถอนโดยรวม: {overall_drop_rate:.1f}%" "
54 report += f"- จำนวนนักศึกษาที่ลงทะเบียน: {len(set([r['STUDENT_ID'] for r in records if r['STATUS_CODE'] == 1]))} คน\n"
55
56 report += "\n-----\n"
57 report += f"จำนวนการลงทะเบียนทั้งหมด (เฉพาะที่ลงทะเบียน): {total_registrations}\n"
58
59 status_counts = {}
60 for rec in records:
61     status_counts[rec['STATUS']] = status_counts.get(rec['STATUS'], 0) + 1
62 for status, count in status_counts.items():
63     report += f"- {status}: {count}\n"
64
65 print(report)
66 return report

```

ภาพที่ 5-9 ไฟล์คัด พิงก์ชัน print_register_report (รายงานการลงทะเบียน)

ภาพที่ 5-9 ไฟล์โค้ด พังก์ชัน print_register_report (รายงานการลงทะเบียน)

```

● ● ●
1 def write_report(report_string, filename):
2     try:
3         with open(filename, 'w', encoding='utf-8') as f:
4             f.write(report_string)
5         print(f"✅ บันทึกรายงานลงไฟล์ {filename} เรียบร้อย")
6     except IOError as e:
7         print(f"❌ Error writing file: {e}")
8
9 # -----
10 # Main Menu
11 # -----
12 def generate_report():
13     while True:
14         print("\n--- เมนูรายงาน ---")
15         print("1. ดูรายงานนักศึกษา")
16         print("2. ดูรายงานการลงทะเบียน")
17         print("3. ย้อนกลับไปหน้าแรก")
18
19         choice = input("เลือกเมนู: ")
20
21         if choice == '1':
22             students = read_all_students()
23             if students:
24                 report = print_student_report(students)
25                 write_report(report, REPORT_STUDENT_FILE_PATH)
26             else:
27                 print("ไม่พบนักศึกษา")
28
29         elif choice == '2':
30             regs = read_all_registrations()
31             courses = load_course_dict()
32             students = read_all_students()
33             if regs:
34                 report = print_register_report(regs, courses, students)
35                 write_report(report, REPORT_REGISTER_FILE_PATH)
36             else:
37                 print("ไม่พบข้อมูลการลงทะเบียน")
38
39         elif choice == '3':
40             break
41         else:
42             print("ตัวเลือกไม่ถูกต้อง")

```

ภาพที่ 5-10 ไฟล์โค้ด พังก์ชัน print_register_report (รายงานการลงทะเบียน)

6. ไฟล์ Sample generate

6.1 ฟังก์ชัน create_registration_record (สร้างบันทึกข้อมูลการลงทะเบียน)

1. การเข้ารหัสและจัดรูปแบบสตริง:

student_id.encode('utf-8') และ course_id.encode('utf-8'): แปลงรหัสนักเรียนและรหัสวิชาให้เป็น ไบต์ [:16].ljust(16, b'\x00'): ตัด ให้มีความยาวสูงสุด 16 ไบต์ และ เติมไบต์ว่าง (b'\x00') จนครบ 16 ไบต์ เพื่อให้มีความยาวคงที่

2. การแพ็คข้อมูล: struct.pack(REGISTRATION_RECORD_FORMAT, ...): ใช้ฟังก์ชัน pack เพื่อ รวม ข้อมูล ทั้งหมด (ID, รหัสนักเรียนที่เตรียมแล้ว, รหัสวิชาที่เตรียมแล้ว, timestamp, สถานะ) เข้าด้วยกัน ข้อมูลจะถูกจัดเรียงตามรูปแบบที่กำหนดใน REGISTRATION_RECORD_FORMAT และส่งกลับเป็น บล็อก ข้อมูลในนารี

3. การจัดการข้อผิดพลาด: try...except struct.error: ดักจับข้อผิดพลาดหากชนิดข้อมูลไม่ตรงหรือไม่สามารถ แพ็คได้ จะพิมพ์ข้อความแจ้งเตือนและส่งกลับ None

```

1 import struct
2 import os
3 import random
4 from datetime import datetime, timedelta
5
6 REGISTRATION_FILE_NAME = 'registration.bin'
7 REGISTRATION_RECORD_FORMAT = '<I16s16sdB'
8 REGISTRATION_RECORD_SIZE = struct.calcsize(REGISTRATION_RECORD_FORMAT)
9 COURSE_FILE_NAME = 'CourseSubject.bin'
10 COURSE_RECORD_FORMAT = '<10s50sB H B B'
11 COURSE_RECORD_SIZE = struct.calcsize(COURSE_RECORD_FORMAT)
12
13 current_dir = os.path.dirname(os.path.abspath(__file__))
14 main_dir = os.path.dirname(current_dir)
15 REGISTRATION_FILE_PATH = os.path.join(main_dir, REGISTRATION_FILE_NAME)
16 COURSE_FILE_PATH = os.path.join(main_dir, COURSE_FILE_NAME)
17
18 def create_registration_record(register_id, student_id, course_id, registration_date, status):
19     """สร้างบันทึกข้อมูลการลงทะเบียนในรูปแบบไบนาเรィ"""
20     packed_student_id = student_id.encode('utf-8')[::16].ljust(16, b'\x00')
21     packed_course_id = course_id.encode('utf-8')[::16].ljust(16, b'\x00')
22     try:
23         record = struct.pack(
24             REGISTRATION_RECORD_FORMAT,
25             register_id,
26             packed_student_id,
27             packed_course_id,
28             registration_date,
29             status
30         )
31         return record
32     except struct.error as e:
33         print(f"เกิดข้อผิดพลาดในการแพ็คข้อมูล: {e}")
34     return None

```

ภาพที่ 6-1 โค้ดฟังก์ชันcreate_registration_record (สร้างบันทึกข้อมูลการลงทะเบียน)

6.2 พังก์ชั่น write_record_to_file (เขียนบันทึกข้อมูลลงในไฟล์)

- 1 การเปิดไฟล์ (Open File): ใช้ with open(file_path, 'ab') as f: เพื่อเปิดไฟล์ที่ระบุ 'a' (append): โหมดนี้สั่งให้ระบบ เขียนข้อมูลต่อท้าย ข้อมูลเดิมที่มีอยู่ในไฟล์ 'b' (binary): โหมดนี้ระบุว่าการดำเนินการนี้เป็นการจัดการกับ ข้อมูลในนารี (ไม่ถูกตีความเป็นข้อความ) การใช้ with open ช่วยให้แน่ใจว่าไฟล์จะถูกปิดอย่างถูกต้องเสมอ
- 2 การเขียนข้อมูล (Write Data): f.write(record): ทำการ เขียน ข้อมูลในนารีที่รับเข้ามาในพารามิเตอร์ record ลงในตำแหน่งท้ายสุดของไฟล์ทันที
3. การจัดการข้อผิดพลาด (Error Handling):
try...except IOError as e:: ดักจับข้อผิดพลาดที่เกี่ยวข้องกับการเข้าถึงไฟล์ (เช่น ไฟล์ถูกล็อก, ไม่มีสิทธิ์) และ พิมพ์ข้อความแจ้งเตือน

```
● ● ●
1 def write_record_to_file(record, file_path=REGISTRATION_FILE_PATH):
2     """เขียนบันทึกข้อมูลลงในไฟล์ในนารี"""
3     try:
4         with open(file_path, 'ab') as f:
5             f.write(record)
6     except IOError as e:
7         print(f"เกิดข้อผิดพลาดในการเขียนไฟล์: {e}")
```

ภาพที่ 6-2 โค้ดพังก์ชั่น write_record_to_file (เขียนบันทึกข้อมูลลงในไฟล์)

6.3 พังก์ชัน read_course_ids (อ่านรหัสวิชาทั้งหมดจากไฟล์)

การเตรียมและการตรวจสอบไฟล์:สร้างรายการ course_ids ว่างเปล่าสำหรับเก็บผลลัพธ์การวนอ่านไฟล์ (Loop Reading):เปิดไฟล์ในโหมด 'rb' (อ่านในนารี)การแยกและแปลงข้อมูล:

struct.unpack(COURSE_RECORD_FORMAT, record_data): แยก ข้อมูลในนารีที่อ่านได้ตามรูปแบบที่กำหนด การจัดการข้อผิดพลาด

```

1 def read_course_ids():
2     """อ่านรหัสวิชาทั้งหมดจากไฟล์ CourseSubject.bin"""
3     course_ids = []
4     try:
5         if not os.path.exists(COURSE_FILE_PATH):
6             print("ไม่พบไฟล์ CourseSubject.bin")
7             return course_ids
8         with open(COURSE_FILE_PATH, 'rb') as f:
9             while True:
10                 record_data = f.read(COURSE_RECORD_SIZE)
11                 if not record_data:
12                     break
13                 try:
14                     unpacked_data = struct.unpack(COURSE_RECORD_FORMAT, record_data)
15                     course_id = unpacked_data[0].strip(b'\x00').decode('utf-8')
16                     course_ids.append(course_id)
17                 except (struct.error, UnicodeDecodeError) as e:
18                     print(f"ข้อผิดพลาดในการอ่าน course record: {e}")
19                     continue
20             except IOError as e:
21                 print(f"เกิดข้อผิดพลาดในการอ่านไฟล์ CourseSubject.bin: {e}")
22     return course_ids

```

ภาพที่ 6-3 โค้ดฟังก์ชัน read_course_ids (อ่านรหัสวิชาทั้งหมดจากไฟล์)

ขั้นตอนการทำงานของโปรแกรม

===== ระบบจัดการข้อมูลลงทะเบียน =====

1. จัดการข้อมูลนักเรียน
2. จัดการข้อมูลรายวิชา
3. จัดการข้อมูลการลงทะเบียน
4. สร้างไฟล์รายงาน
0. ออกจากระบบ

กรุณาเลือกเมนูหลัก: []

ภาพที่ 1 หน้าหลักของโปรแกรม

1. เมื่อรันโปรแกรมจะแสดง เมนูหลักขึ้นมาให้เลือก 4 เมนู สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลขเมนู

2. เลือกเข้าใช้งานเมนู จัดการข้อมูลนักเรียน หมายเลข 1

2.1 เมนูหลักของ จัดการข้อมูลนักเรียนจะแสดงเมนูการใช้งานทั้งหมด 6 เมนูสามารถเลือกเข้าใช้งานได้โดยพิมพ์เลขเมนู

===== ระบบจัดการข้อมูลนักเรียน =====

1. เพิ่มข้อมูลนักเรียน
2. ดูข้อมูลนักเรียนทั้งหมด
3. ดูข้อมูลนักเรียนรายบุคคล
4. ดูข้อมูลนักเรียนแบบกรอง
5. แก้ไขข้อมูลนักเรียน
6. ลบข้อมูลนักเรียน
0. กลับสู่เมนูหลัก

กรุณาเลือกเมนู: []

ภาพที่ 2 เมนูหลักของหน้าจัดการข้อมูลนักเรียน

2.2 เข้าใช้งานเมนู เพิ่มข้อมูลนักเรียน สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 1

===== ระบบจัดการข้อมูลนักเรียน =====

1. เพิ่มข้อมูลนักเรียน
2. ดูข้อมูลนักเรียนทั้งหมด
3. ดูข้อมูลนักเรียนรายบุคคล
4. ดูข้อมูลนักเรียนแบบกรอง
5. แก้ไขข้อมูลนักเรียน
6. ลบข้อมูลนักเรียน
0. กลับสู่เมนูหลัก

กรุณาเลือกเมนู: 1

ป้อนรหัสนักเรียน ("ไม่เกิน 16 ตัวอักษร"): []

ภาพที่ 3 เมนูเพิ่มข้อมูลนักเรียน

2.2.1 กรอกข้อมูลตามที่โปรแกรมแจ้งเตือน ดังนี้ รหัสนักศึกษา, ชื่อจริง, นามสกุล, สาขาวิชา, ชั้นปี, สถานะ

===== ระบบคลังข้อมูลนักเรียน =====	
1.	เพิ่มข้อมูลนักเรียน
2.	ดูข้อมูลนักเรียนทั้งหมด
3.	ดูข้อมูลนักเรียนรายบุคคล
4.	ดูข้อมูลนักเรียนแบบกรอง
5.	แก้ไขข้อมูลนักเรียน
6.	ลบข้อมูลนักเรียน
0.	กลับสู่เมนูหลัก
กรุณาเลือก เมนู: 1	
ป้อนรหัสนักเรียน (ไม่ กิน 16 ตัวอักษร): 6806022510001	
ป้อนชื่อจริง (ไม่ กิน 50 ตัวอักษร): Akkarachai	
ป้อนนามสกุล (ไม่ กิน 50 ตัวอักษร): jaidee	
ป้อนสาขาวิชา (ไม่ กิน 20 ตัวอักษร): INET	
ป้อนชั้นปี (1, 2, 3, 4, ...): 1	
ป้อนสถานะ (1 = Active, 0 = Inactive): 1	
เพิ่มข้อมูลนักเรียนสำเร็จ!	

ภาพที่ 4 เพิ่มข้อมูลนักเรียน

2.3 เมนูดูข้อมูลนักเรียนทั้งหมด จะแสดงตารางรายชื่อนักเรียนทั้งหมด สามารถกดเลือกใช้ได้โดยพิมพ์หมายเลข 2

===== ระบบคลังข้อมูลนักเรียน =====	
1.	เพิ่มข้อมูลนักเรียน
2.	ดูข้อมูลนักเรียนทั้งหมด
3.	ดูข้อมูลนักเรียนรายบุคคล
4.	ดูข้อมูลนักเรียนแบบกรอง
5.	แก้ไขข้อมูลนักเรียน
6.	ลบข้อมูลนักเรียน
0.	กลับสู่ เมนูหลัก
กรุณาเลือก เมนู: 2	

ภาพที่ 5 เมนูดูข้อมูลนักเรียนทั้งหมด

2.3.1 เมื่อเลือกเมนูนี้จะแสดงตารางรายชื่อนักเรียนทั้งหมด

รายงานนักศึกษา					
STUDENT ID	FIRST NAME	LAST NAME	MAJOR	YEAR	STATUS
STU796973	Thanapon	Suwannarat	English	4	Inactive
STU705257	Supaporn	Wongyai	Computer Sci...	1	Active
STU442194	Nattapong	Phromdee	Biology	4	Inactive
STU569818	Kritsada	Meechai	Economics	2	Inactive
STU942291	Nareerat	Khamphol	Economics	2	Inactive
STU778182	Chatchai	Wongyai	Economics	2	Inactive
STU837721	Nattapong	Saengsuwan	Information ...	2	Active
STU510451	Nattapong	Wongyai	Computer Sci...	4	Active
STU911742	Supaporn	Saengsuwan	Chemistry	4	Active
STU109544	Waranya	Khamphol	Physics	3	Inactive
STU471005	Waranya	Rattanakorn	Information ...	1	Inactive
STU355431	Nattapong	Chantaras	Information ...	2	Active
STU779447	Nareerat	Khamphol	Computer Sci...	1	Active
STU669065	Nareerat	Saengsuwan	Information ...	4	Active
STU804899	Nareerat	Chantaras	Computer Sci...	3	Inactive
STU945148	Supaporn	Wongyai	Business	1	Active

ภาพที่ 6 ตารางแสดงรายชื่อนักเรียนทั้งหมด

2.4 เมนูดูข้อมูลนักเรียนรายบุคคลจะแสดงตารางรายชื่อนักเรียนตามบุคคล สามารถกดเลือกใช้ได้โดยพิมพ์หมายเลข 3

===== ระบบจัดการข้อมูลนักเรียน =====

- 1. เพิ่มนักเรียน
- 2. ดูข้อมูลนักเรียนทั้งหมด
- 3. ดูข้อมูลนักเรียนรายบุคคล
- 4. ดูข้อมูลนักเรียนแบบกรอง
- 5. แก้ไขข้อมูลนักเรียน
- 6. ลบข้อมูลนักเรียน
- 0. กลับสู่หน้าหลัก

กรุณาเลือกหมายเลข: 3
ป้อนรหัสนักเรียนที่ต้องการดู: █

ภาพที่ 7 เมนูดูข้อมูลนักเรียนแบบรายบุคคล

2.4.1 กรอกรหัสนักเรียนที่ต้องการดูข้อมูล จะแสดงข้อมูลของนักเรียนตามรหัส

กรุณาเลือกเมนู: 3

ป้อนรหัสนักเรียนที่ต้องการดู: 6806022510001

--- ข้อมูลนักเรียน ---

รหัสนักเรียน: 6806022510001

ชื่อจริง: Akkarachai

นามสกุล: jaidee

สาขาวิชา: INET

ชั้นปี: 1

สถานะ: Active

ภาพที่ 8 แสดงข้อมูลนักเรียนรายบุคคล

2.5 เมนูดูข้อมูลนักเรียนแบบกรองจะแสดงตารางรายชื่อนักเรียนตาม filter ดังนี้ กรองตามสาขาวิชา,
กรองตามชั้นปี,กรองตามสถานะ สามารถกดเลือกใช้ได้โดยพิมพ์หมายเลข 4

===== ระบบคัดกรองข้อมูลนักเรียน =====

1. เพิ่มข้อมูลนักเรียน
2. ดูข้อมูลนักเรียนทั้งหมด
3. ดูข้อมูลนักเรียนรายบุคคล
4. ดูข้อมูลนักเรียนแบบกรอง
5. แก้ไขข้อมูลนักเรียน
6. ลบข้อมูลนักเรียน
0. กลับสู่หน้าหลัก

กรุณาเลือกเมนู: 4

--- กรองข้อมูลนักเรียน ---

เลือกเงื่อนไขการกรอง:

1. กรองตามสาขาวิชา
2. กรองตามชั้นปี
3. กรองตามสถานะ

กรุณาเลือก (1-3): []

ภาพที่ 9 เมนูดูข้อมูลนักเรียนแบบกรอง

2.5.1 เมนูกรองตามสาขาวิชา กดหมายเลข 1 จะให้กรอกชื่อสาขาแล้วทำการแสดงนักเรียนตามสาขาที่กรอก

--- กรองข้อมูลนักเรียน ---					
เลือกเงื่อนไขในการกรอง:					
1. กรองตามสาขาวิชา					
2. กรองตามชื่อ					
3. กรองตามสถานะ					
กรุณาเลือก (1-3): 1					
ป้อนสาขาวิชาที่ต้องการกรอง :INET					
=====					
รายงานนักศึกษาที่กรอง					
=====					
STUDENT ID	FIRST NAME	LAST NAME	MAJOR	YEAR	STATUS
6806022510001	Akkarachai	jaidee	INET	1	Active
=====					

ภาพที่ 10 เมนูกรองนักเรียนตามสาขา

2.5.2 เมนูกรองตามชั้นปี กดหมายเลข 2 จะให้กรอกชั้นปีแล้วทำการแสดงนักเรียนตามชั้นปีที่กรอก

--- กรองข้อมูลนักเรียน ---					
เลือกเงื่อนไขในการกรอง:					
1. กรองตามสาขาวิชา					
2. กรองตามชื่อ					
3. กรองตามสถานะ					
กรุณาเลือก (1-3): 2					
ป้อนชั้นปีที่ต้องการกรอง (1, 2, 3, 4, ...): 1					
=====					
รายงานนักศึกษาที่กรอง					
=====					
STUDENT ID	FIRST NAME	LAST NAME	MAJOR	YEAR	STATUS
STU70527	Supaporn	Wongyai	Computer Sci...	1	Active
STU471005	Waranya	Rattanakorn	Information ...	1	Inactive
STU779447	Nareerat	Khamphol	Computer Sci...	1	Active
STU945148	Supaporn	Wongyai	Business	1	Active
STU197090	Pimchanok	Rattanakorn	English	1	Active
STU546039	Nattapong	Saengsuwan	English	1	Active
STU971475	Waranya	Chantaras	Computer Sci...	1	Inactive
STU892150	Somchai	Saengsuwan	Mathematics	1	Active
STU162343	Nattapong	Meechai	Economics	1	Inactive
STU116871	Pimchanok	Meechai	Computer Sci...	1	Active
STU345896	Waranya	Srisuk	Mathematics	1	Inactive
STU663268	Anuwat	Rattanakorn	Computer Sci...	1	Active

ภาพที่ 11 เมนูกรองนักเรียนตามชั้นปี

2.5.3 เมนูกรองตามสถานะ กดหมายเลข 3 จะให้กรอกสถานะ Active /Inactive เมื่อกรอก
เร็วจะทำการแสดงนักเรียนตามสถานะที่กรอก

--- กรองข้อมูลนักเรียน ---						
เลือกเงื่อนไขการกรอง:						
1. กรองตามสาขาวิชา						
2. กรองตามชั้นปี						
3. กรองตามสถานะ						
ภาษาเลือก (1-3): 3						
ป้อนสถานะที่ต้องการกรอง (Active/Inactive): Active						
รายงานนักศึกษาที่กรอง						
STUDENT ID	FIRST NAME	LAST NAME	MAJOR	YEAR	STATUS	
STU705257	Supaporn	Wongyai	Computer Sci...	1	Active	
STU837721	Nattapong	Saengsuwan	Information ...	2	Active	
STU510451	Nattapong	Wongyai	Computer Sci...	4	Active	
STU911742	Supaporn	Saengsuwan	Chemistry	4	Active	
STU355431	Nattapong	Chantaras	Information ...	2	Active	
STU779447	Nareerat	Khamphol	Computer Sci...	1	Active	
STU669065	Nareerat	Saengsuwan	Information ...	4	Active	
STU945149	Supaporn	Wongyai	Business ...	1	Active	

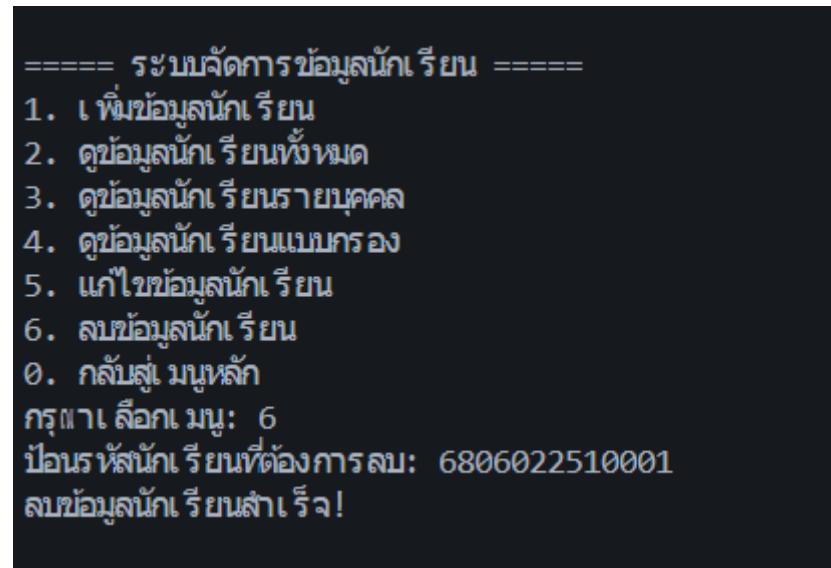
ภาพที่ 12 เมนูกรองนักเรียนตามสถานะ

2.6 เมนูแก้ไขข้อมูลนักเรียน จะสามารถกรอกรหัสนักเรียนเพื่อแก้ไขข้อมูลได้ สามารถแก้ไขข้อมูลได้
ดังนี้ ชื่อจริงใหม่ นามสกุลใหม่ สาขาวิชาใหม่ ชั้นปีใหม่ สถานะใหม่ เข้าใช้งานเมนูโดยกดหมายเลข 5

กรุณาเลือกเมนู: 5
ป้อนรหัสนักเรียนที่ต้องการแก้ไข: 6806022510001
พนักงานนักเรียนที่ต้องการแก้ไข
รหัสนักเรียน: 6806022510001
ชื่อจริง: Akkarachai
นามสกุล: jaidee
สาขาวิชา: INET
ชั้นปี: 1
สถานะ: Active
ป้อนชื่อจริงใหม่ (Enter เพื่อใช้ค่าเดิม):
ป้อนนามสกุลใหม่ (Enter เพื่อใช้ค่าเดิม):
ป้อนสาขาวิชาใหม่ (Enter เพื่อใช้ค่าเดิม):
ป้อนชั้นปีใหม่ (Enter เพื่อใช้ค่าเดิม):
ป้อนสถานะใหม่ (1=Active, 0=Inactive) (Enter เพื่อใช้ค่าเดิม):
แก้ไขข้อมูลสำเร็จ!

ภาพที่ 13 เมนูแก้ไขข้อมูลนักเรียน

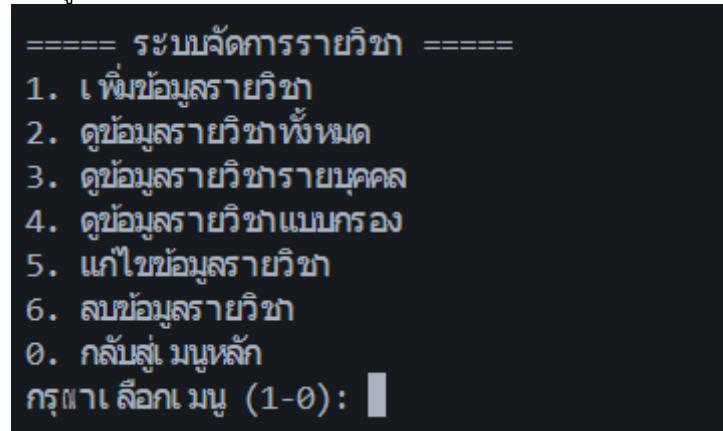
2.7 เมนูลับข้อมูลนักเรียน กรอกรหัสนักศึกษาที่ต้องการลบ โปรแกรมจะลบข้อมูลนักเรียนไปอย่างถาวร



ภาพที่ 14 เมนูลับข้อมูลนักเรียน

3.เลือกเข้าใช้งานเมนู จัดการข้อมูลรายวิชา หมายเลข 2

3.1 เมนูหลักของ จัดการข้อมูลนักเรียนจะแสดงเมนูการใช้งานทั้งหมด 6 เมนูสามารถเลือกเข้าใช้งานได้โดยพิมพ์เลขเมนูตามที่แสดง



ภาพที่ 15 เมนูหลักของ จัดการข้อมูลรายวิชา

3.2 เข้าใช้งานเมนู เพิ่มข้อมูลรายวิชา สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 1

3.2.1 กรอกข้อมูลตามที่โปรแกรมแจ้งเตือน ดังนี้ รหัสวิชา ชื่อวิชา หน่วยกิต ปีการศึกษา ภาคเรียน สถานะ

```
===== ระบบจัดการรายวิชา =====
1. เพิ่มข้อมูลรายวิชา
2. ดูข้อมูลรายวิชาทั้งหมด
3. ดูข้อมูลรายวิชารายบุคคล
4. ดูข้อมูลรายวิชาแบบกรอง
5. แก้ไขข้อมูลรายวิชา
6. ลบข้อมูลรายวิชา
0. กลับสู่ เมนูหลัก

กรุณาเลือกเมนู (1-0): 1
ป้อนรหัสวิชา (ไม่เกิน 10 ตัวอักษร): 6666666666
ป้อนชื่อวิชา (ไม่เกิน 50 ตัวอักษร): Technology
ป้อนหน่วยกิต: 3
ป้อนปีการศึกษา (เช่น 2568): 2568
ป้อนภาคเรียน (1, 2, 3): 1
ป้อนสถานะ (1 = Active, 0 = Inactive): 1
เพิ่มข้อมูลรายวิชาสำเร็จ!
```

ภาพที่ 16 เมนูเพิ่มข้อมูลรายวิชา

3.3 เข้าใช้งานเมนู ดูข้อมูลรายวิชาทั้งหมด สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 2

รายงานรายวิชา						
COURSE ID	COURSE NAME	CREDIT	ACADEMIC YEAR	SEMESTER	STATUS	
SCI569	Computer Networks 5	2	2566	1	Inactive	
IT252	Machine Learning 4	4	2567	3	Active	
SCI512	Introduction to P...	3	2567	2	Active	
IT115	Computer Networks 1	4	2568	2	Inactive	
ENG431	Computer Graphics 3	4	2567	3	Inactive	
MATH303	Data Structures 4	4	2568	3	Inactive	
IT786	Web Development 2	4	2567	2	Active	
ENG695	Computer Graphics 3	2	2568	3	Active	
MATH783	Operating Systems 3	4	2567	3	Inactive	
SCI420	Algorithms 3	3	2568	2	Inactive	

ภาพที่ 17 เมนูดูข้อมูลรายวิชาทั้งหมด

3.4 เข้าใช้งานเมนู ดูข้อมูลรายวิชาตามรายวิชา กรอกรหัสวิชาจะแสดงข้อมูลของรหัสวิชาที่กรอก
สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 3

```
===== ระบบจัดการรายวิชา =====
1. เพื่อนบ้านรายวิชา
2. ดูข้อมูลรายวิชาทั้งหมด
3. ดูข้อมูลรายวิชารายบุคคล
4. ดูข้อมูลรายวิชาแบบกรอง
5. แก้ไขข้อมูลรายวิชา
6. ลบข้อมูลรายวิชา
0. กลับสู่เมนูหลัก
กรุณาเลือกเมนู (1-0): 3
ป้อนรหัสวิชาที่ต้องการดู: CS193

=====
ข้อมูลรายวิชาที่ค้นหา
=====
รหัสวิชา: CS193
ชื่อวิชา: Operating Systems 4
หน่วยกิต: 4
ปีการศึกษา: 2567
ภาคเรียน: 2
สถานะ: Active
=====
```

ภาพที่ 18 เมนูดูข้อมูลรายวิชาตามรายวิชา

3.5 เมนูดูข้อมูลรายวิชาแบบกรองจะแสดงตารางรายชื่อวิชาตาม filter ดังนี้ กรองตามปี การศึกษา กรองตามภาคเรียน กรองตามสถานะ (Active) กรองตามปีการศึกษาและภาคเรียน สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 4

```
===== ระบบจัดการรายวิชา =====
1. เพื่อนบ้านรายวิชา
2. ดูข้อมูลรายวิชาทั้งหมด
3. ดูข้อมูลรายวิชารายบุคคล
4. ดูข้อมูลรายวิชาแบบกรอง
5. แก้ไขข้อมูลรายวิชา
6. ลบข้อมูลรายวิชา
0. กลับสู่เมนูหลัก
กรุณาเลือกเมนู (1-0): 4

--- ตัวเลือกการกรอง ---
1. กรองตามปีการศึกษา
2. กรองตามภาคเรียน
3. กรองตามสถานะ (Active)
4. กรองตามปีการศึกษาและภาคเรียน
5. กลับไปเมนูหลัก
กรุณาเลือกการกรอง (1-5): 1
```

ภาพที่ 19 เมนูดูข้อมูลรายวิชาแบบกรอง

3.5.1 เมนูกรองรายวิชาตามปีการศึกษา กรอกปีการศึกษาจะแสดงตามรายวิชาตามปีการศึกษา

--- ตัวเลือกการกรอง ---

1. กรองตามมีการศึกษา
2. กรองตามภาคเรียน
3. กรองตามสถานะ (Active)
4. กรองตามมีการศึกษาและภาคเรียน
5. กลับไปหน้าหลัก

กรุณาเลือกการกรอง (1-5): 1
ปีการศึกษาที่ต้องการกรอง: 2568

รายงานรายวิชาที่กรอง (24 รายการ)

COURSE ID	COURSE NAME	CREDIT	ACADEMIC YEAR	SEMESTER	STATUS
IT115	Computer Networks 1	4	2568	2	Inactive
MATH303	Data Structures 4	4	2568	3	Inactive
ENG695	Computer Graphics 3	2	2568	3	Active
SCI420	Algorithms 3	3	2568	2	Inactive
IT463	Operating Systems 4	3	2568	3	Inactive
CS162	Artificial Intell...	2	2568	1	Active
IT496	Web Development 1	3	2568	1	Inactive

ภาพที่ 20 เมนูกรองรายวิชาตามปีการศึกษา

3.5.2 เมนูกรองรายวิชาตามภาคเรียน กรอกภาคเรียนจะแสดงตามรายวิชาตามภาคเรียน

--- ตัวเลือกการกรอง ---

1. กรองตามมีการศึกษา
2. กรองตามภาคเรียน
3. กรองตามสถานะ (Active)
4. กรองตามมีการศึกษาและภาคเรียน
5. กลับไปหน้าหลัก

กรุณาเลือกการกรอง (1-5): 2
ปีการศึกษาที่ต้องการกรอง (1, 2, 3): 1

รายงานรายวิชาที่กรอง (19 รายการ)

COURSE ID	COURSE NAME	CREDIT	ACADEMIC YEAR	SEMESTER	STATUS
SCI569	Computer Networks 5	2	2566	1	Inactive
SCI559	Introduction to P...	2	2566	1	Active
CS162	Artificial Intell...	2	2568	1	Active
MATH198	Operating Systems 2	3	2566	1	Active
ENG250	Machine Learning 5	2	2567	1	Inactive
IT496	Web Development 1	3	2568	1	Inactive
CS346	Database Systems 4	3	2568	1	Active

ภาพที่ 21 เมนูกรองรายวิชาตามภาคเรียน

3.5.3 เมนูกรองรายวิชาตามสถานะ(Active) จะแสดงรายวิชาที่เปิดสอน

--- ตัวเลือกการกรอง ---						
1. กรองตามฝีมือการศึกษา						
2. กรองตามภาคเรียน						
3. กรองตามสถานะ (Active)						
4. กรองตามฝีมือการศึกษาและภาคเรียน						
5. กดปุ่มไปเมนูหลัก						
คุณได้เลือกการกรอง (1-5): 3						
รายงานรายวิชาที่กรอง (33 รายการ)						
COURSE ID	COURSE NAME	CREDIT	ACADEMIC YEAR	SEMESTER	STATUS	
IT252	Machine Learning 4	4	2567	3	Active	
SCI512	Introduction to P...	3	2567	2	Active	
IT786	Web Development 2	4	2567	2	Active	
ENG695	Computer Graphics 3	2	2568	3	Active	
SCI559	Introduction to P...	2	2566	1	Active	
CS162	Artificial Intell...	2	2568	1	Active	
MATH198	Operating Systems 2	3	2566	1	Active	
ENG647	Artificial Intell...	2	2567	2	Active	

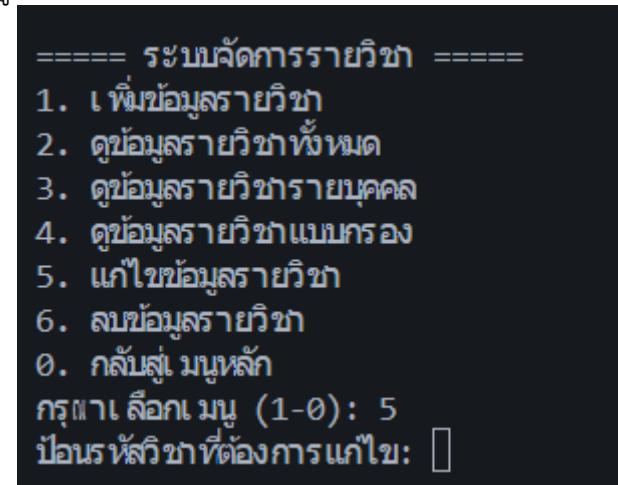
ภาพที่ 22 เมนูกรองรายวิชาตามสถานะ

3.5.4 เมนูกรองรายวิชาตามปีการศึกษาและภาคเรียน กรอกปีการศึกษาและภาคเรียนจะแสดงรายวิชาที่กรอก

--- ตัวเลือกการกรอง ---						
1. กรองตามฝีมือการศึกษา						
2. กรองตามภาคเรียน						
3. กรองตามสถานะ (Active)						
4. กรองตามฝีมือการศึกษาและภาคเรียน						
5. กดปุ่มไปเมนูหลัก						
คุณได้เลือกการกรอง (1-5): 4						
ปีของปีการศึกษาที่ต้องการกรอง: 2568						
ปีของภาคเรียนที่ต้องการกรอง (1, 2, 3): 1						
รายงานรายวิชาที่กรอง (8 รายการ)						
COURSE ID	COURSE NAME	CREDIT	ACADEMIC YEAR	SEMESTER	STATUS	
CS162	Artificial Intell...	2	2568	1	Active	
IT496	Web Development 1	3	2568	1	Inactive	
CS346	Database Systems 4	3	2568	1	Active	
CS937	Web Development 3	2	2568	1	Active	
MATH259	Database Systems 2	2	2568	1	Active	
CS991	Computer Graphics 3	2	2568	1	Inactive	

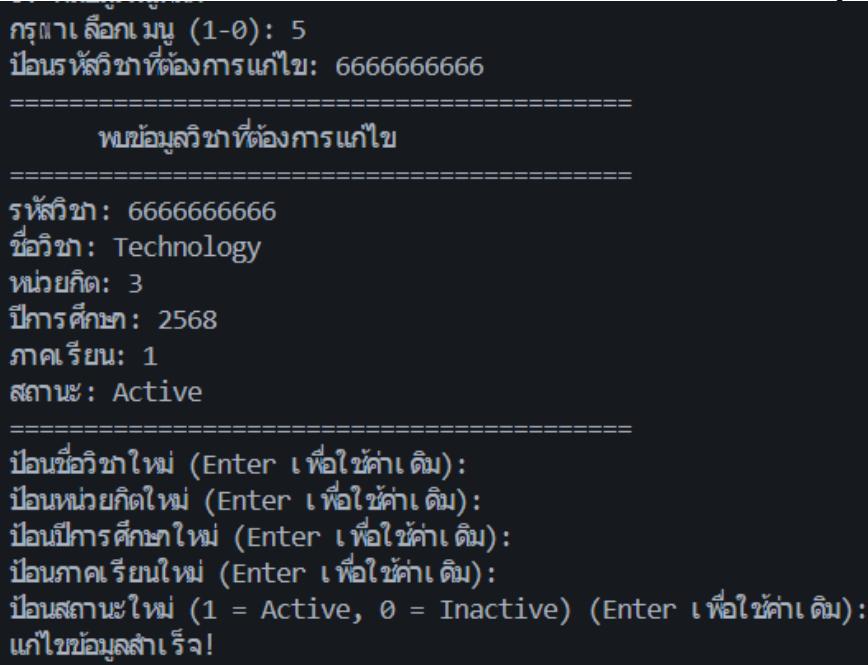
ภาพที่ 23 เมนูกรองรายวิชาตามปีการศึกษาและภาคเรียน

3.6 เมนูแก้ไขข้อมูลรายวิชา สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 5



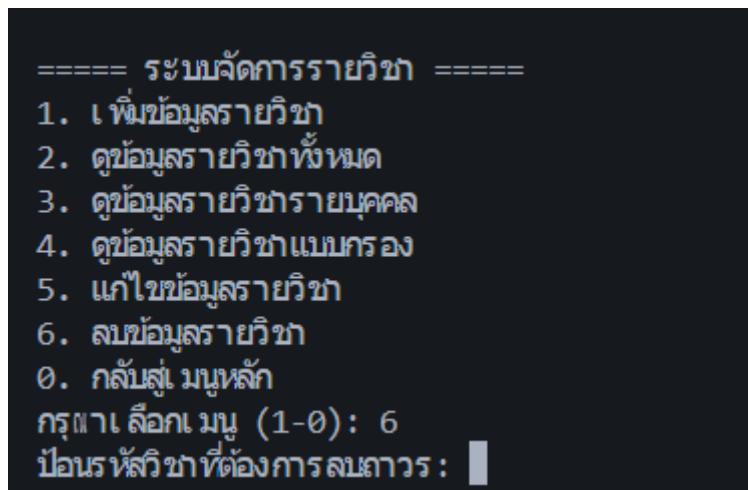
ภาพที่ 23 เมนูแก้ไขข้อมูลรายวิชา

3.6.1 แก้ไขรายวิชา โดยการกรอกรหัสวิชาที่ต้องการแก้ไข จะแสดงข้อมูลที่สามารถแก้ไขได้



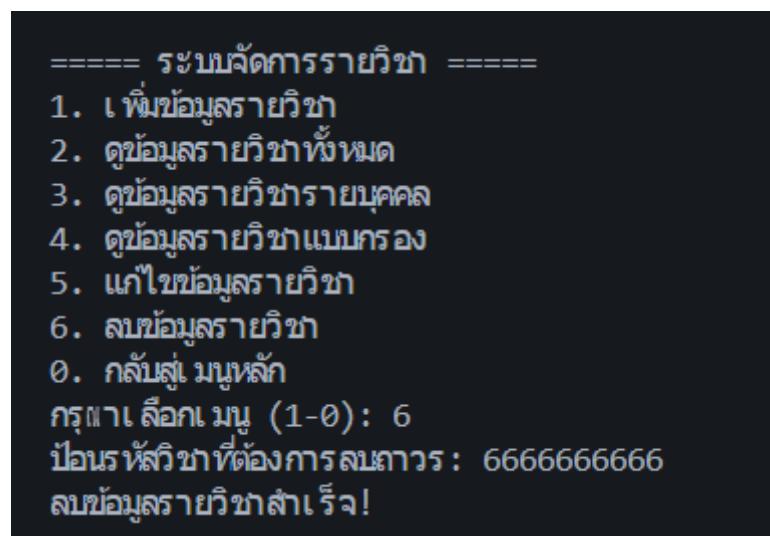
ภาพที่ 24 แก้ไขข้อมูลรายวิชาตามรหัสวิชา

3.7 เมนูลับข้อมูลรายวิชา สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 6



ภาพที่ 25 เมนูลับข้อมูลรายวิชา

3.7.1 ลบข้อมูลรายวิชา โดยการกรอกรหัสวิชาจะทำการลบข้อมูลรายการ



ภาพที่ 26 ลบข้อมูลรายวิชาอย่างถาวร

4.เลือกเข้าใช้งานเมนู จัดการข้อมูลการลงทะเบียน หมายเลข 3

```
===== ระบบจัดการข้อมูลการลงทะเบียน =====
1. เพิ่มข้อมูลการลงทะเบียน
2. ดูข้อมูลการลงทะเบียนทั้งหมด
3. ดูข้อมูลการลงทะเบียนรายการเดียว
4. ดูข้อมูลการลงทะเบียนแบบกรอง
5. แก้ไขข้อมูลการลงทะเบียน
6. ลบข้อมูลการลงทะเบียน
0. กลับสู่เมนูหลัก
กรุณาเลือกเมนู: 1
```

ภาพที่ 27 เมนูจัดการข้อมูลการลงทะเบียน

4.1 เข้าใช้งานเมนู เพิ่มข้อมูลการลงทะเบียน สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 1

```
===== ระบบจัดการข้อมูลการลงทะเบียน =====
1. เพิ่มข้อมูลการลงทะเบียน
2. ดูข้อมูลการลงทะเบียนทั้งหมด
3. ดูข้อมูลการลงทะเบียนรายการเดียว
4. ดูข้อมูลการลงทะเบียนแบบกรอง
5. แก้ไขข้อมูลการลงทะเบียน
6. ลบข้อมูลการลงทะเบียน
0. กลับสู่เมนูหลัก
กรุณาเลือกเมนู: 1
ป้อนรหัสนักเรียน: 1
```

ภาพที่ 28 เมนูเพิ่มข้อมูลการลงทะเบียน

4.1.1 กรอกข้อมูลตามที่โปรแกรมแจ้งเตือน ดังนี้

```
กรุณาเลือกเมนู: 1
ป้อนรหัสนักเรียน: STU898946

==== พนักงานนักเรียน ===
รหัสนักเรียน: STU898946
ชื่อ: Anuwat Meechai
สาขาวิชา: Engineering
ชั้นปี: 1
สถานะ: Active
=====
ใช่คนนี้ใช่หรือไม่? (y/n): y
ป้อนรหัสวิชา: CS763
ป้อนสถานะ (1 = Registered, 0 = Dropped): 1
 เพิ่มข้อมูลการลงทะเบียนแล้ว!
```

ภาพที่ 29 เพิ่มข้อมูลการลงทะเบียน

4.2 เมนูดูข้อมูลการลงทะเบียนทั้งหมด สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 2

===== รายชื่อผู้ลงทะเบียน =====				
ID	STUDENT ID	COURSE ID	REGISTRATION DATE	STATUS
1	STU788477	SCI591	2025-06-02 01:36:16	Dropped
2	STU153914	SCI591	2025-05-28 05:53:05	Dropped
3	STU818977	SCI591	2025-05-27 09:09:00	Registered
4	STU794109	SCI591	2025-07-11 00:33:28	Dropped
5	STU242278	SCI591	2025-06-16 11:19:52	Registered
6	STU535097	SCI591	2025-04-17 14:51:54	Registered

ภาพที่ 30 ดูข้อมูลการลงทะเบียนทั้งหมด

4.3 เมนูดูข้อมูลการลงทะเบียนรายการเดียวโดยกรอกรหัสการลงทะเบียน สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 3

===== รายชื่อผู้ลงทะเบียน =====				
ID	STUDENT ID	COURSE ID	REGISTRATION DATE	STATUS
3	STU818977	SCI591	2025-05-27 09:09:00	Registered

ภาพที่ 31 ดูข้อมูลการลงทะเบียนรายการเดียว

4.4 เมนูดูข้อมูลการลงทะเบียนแบบกรอง ดังนี้ กรองตามรหัสนักเรียน กรองตามรหัสวิชา กรองตามสถานะ (Registered) สามารถกดเลือกใช้ได้โดยพิมพ์หมายเลข 4

--- ตัวเลือกการกรอง ---	
1.	กรองตามรหัสนักเรียน
2.	กรองตามรหัสวิชา
3.	กรองตามสถานะ (Registered)
4.	กลับไปหน้าหลัก
กรุณาเลือกการกรอง (1-4):	[]

ภาพที่ 32 เมนูดูข้อมูลการลงทะเบียนแบบกรอง

4.4.1 เมนูกรองตามรหัสนักเรียน กดหมายเลข 1 จะให้กรอกรหัสนักเรียนแล้วทำการแสดงการลงทะเบียนของนักเรียนตามรหัสนักเรียน

--- ตัวเลือกการกรอง ---				
1. กรองตามรหัสนักเรียน				
2. กรองตามรหัสวิชา				
3. กรองตามสถานะ (Registered)				
4. กลับไปหน้าหลัก				
กรุณาเลือกการกรอง (1-4): 1				
ป้อนรหัสนักเรียนที่ต้องการกรอง: STU818977				
<hr/> <hr/> รายงานการลงทะเบียนทั่วไป (1 รายการ)				
ID	STUDENT ID	COURSE ID	REGISTRATION DATE	STATUS
3	STU818977	SCI591	2025-05-27 09:09:00	Registered

ภาพที่ 33 เมนูดูข้อมูลการลงทะเบียนของนักเรียน

4.4.2 เมนูกรองตามรหัสวิชา กดหมายเลข 2 จะให้กรอกรหัสวิชาแล้วทำการแสดงการลงทะเบียนของรายวิชาตามรหัสวิชา

--- ตัวเลือกการกรอง ---				
1. กรองตามรหัสนักเรียน				
2. กรองตามรหัสวิชา				
3. กรองตามสถานะ (Registered)				
4. กลับไปหน้าหลัก				
กรุณาเลือกการกรอง (1-4): 2				
ป้อนรหัสวิชาที่ต้องการกรอง: SCI591				
<hr/> <hr/> รายงานการลงทะเบียนทั่วไป (17 รายการ)				
ID	STUDENT ID	COURSE ID	REGISTRATION DATE	STATUS
1	STU788477	SCI591	2025-06-02 01:36:16	Dropped
2	STU153914	SCI591	2025-05-28 05:53:05	Dropped
3	STU818977	SCI591	2025-05-27 09:09:00	Registered
4	STU794109	SCI591	2025-07-11 00:33:28	Dropped
5	STU242278	SCI591	2025-06-16 11:19:52	Registered
6	STU535097	SCI591	2025-04-17 14:51:54	Registered
7	STU636498	SCI591	2024-11-28 16:51:22	Registered

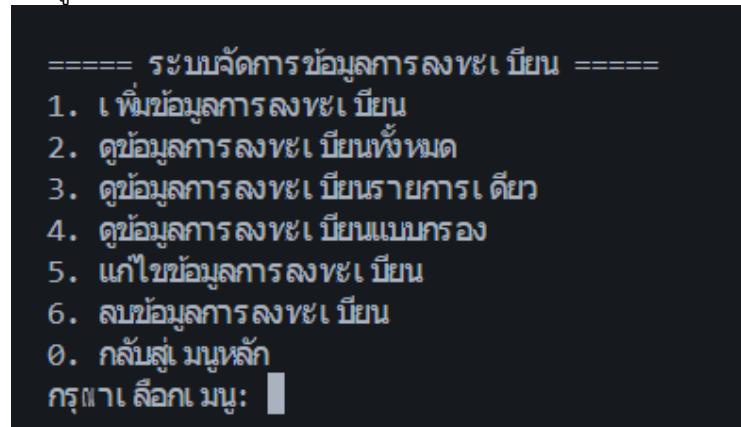
ภาพที่ 34 เมนูดูข้อมูลการลงทะเบียนของรายวิชา

4.4.3 เมนูกรองตามสถานะการลงทะเบียน กดหมายเลข 3 จะแสดงนักเรียนที่ลงทะเบียนรายวิชาต่างๆ ที่มีสถานะ Registered

--- ตัวเลือกการกรอง ---				
1. กรองตามรหัสนักเรียน				
2. กรองตามรหัสวิชา				
3. กรองตามสถานะ (Registered)				
4. กลับไปหน้าหลัก				
กรุณาเลือกการกรอง (1-4): 3				
<hr/> <hr/> รายงานการลงทะเบียนทั่วไป (44 รายการ)				
ID	STUDENT ID	COURSE ID	REGISTRATION DATE	STATUS
3	STU818977	SCI591	2025-05-27 09:09:00	Registered
5	STU242278	SCI591	2025-06-16 11:19:52	Registered
6	STU535097	SCI591	2025-04-17 14:51:54	Registered
7	STU636498	SCI591	2024-11-28 16:51:22	Registered
8	STU796973	SCI591	2024-09-28 10:29:41	Registered
9	STU820212	SCI591	2025-06-12 08:39:09	Registered
10	STU345896	SCI591	2025-02-07 19:43:36	Registered
11	STU989829	SCI591	2025-09-16 05:10:08	Registered

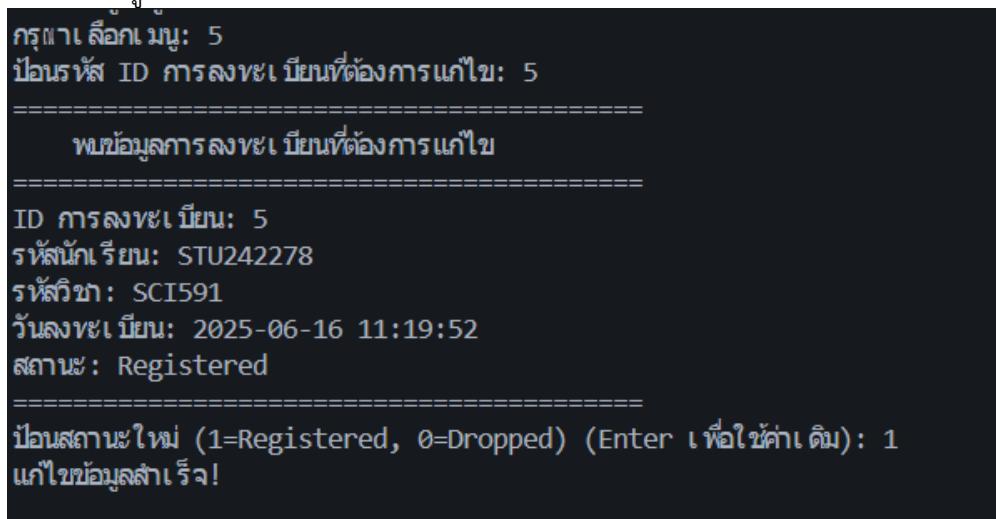
ภาพที่ 35 เมนูดูข้อมูลการลงทะเบียนของนักเรียนที่มีสถานะ Registered

4.5 เมนูแก้ไขข้อมูลการลงทะเบียน สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 5



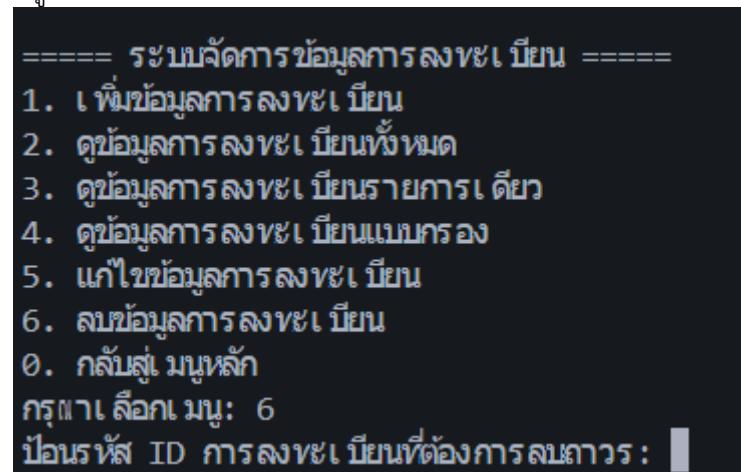
ภาพที่ 36 เมนูแก้ไขข้อมูลการลงทะเบียน

4.5.1 แก้ไขข้อมูลการลงทะเบียน โดยการกรอกรหัสลงทะเบียนที่ต้องการแก้ไข จะแสดงข้อมูลที่สามารถแก้ไขได้



ภาพที่ 37 แก้ไขข้อมูลการลงทะเบียน

4.6 เมนูลบข้อมูลการลงทะเบียน สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 6



ภาพที่ 38 เมนูลบข้อมูลการลงทะเบียน

4.6.1 ลบข้อมูลการลงทะเบียน โดยการกรอกรหัสการลงทะเบียน

```
===== ระบบจัดการข้อมูลการลงทะเบียน =====
1. เพิ่งข้อมูลการลงทะเบียน
2. ดูข้อมูลการลงทะเบียนทั้งหมด
3. ดูข้อมูลการลงทะเบียนรายการเดียว
4. ดูข้อมูลการลงทะเบียนแบบกรอง
5. แก้ไขข้อมูลการลงทะเบียน
6. ลบข้อมูลการลงทะเบียน
0. กลับไปเมนูหลัก
กรุณาเลือกเมนู: 6
ป้อนรหัส ID การลงทะเบียนที่ต้องการลบ: ■
```

ภาพที่ 38 ลบข้อมูลการลงทะเบียน

5.เลือกเข้าใช้งานเมนู สร้างไฟล์รายงาน หมายเลข 4

```
===== ระบบจัดการข้อมูลลงทะเบียนเรียน =====
1. จัดการข้อมูลนักเรียน
2. จัดการข้อมูลรายวิชา
3. จัดการข้อมูลการลงทะเบียน
4. สร้างไฟล์รายงาน
0. ออกจากโปรแกรม
กรุณาเลือกเมนูหลัก: 4

--- เมนูรายงาน ---
1. ดูรายงานนักศึกษา
2. ดูรายงานการลงทะเบียน
3. ย้อนกลับไปหน้าแรก
เลือกเมนู: ■
```

ภาพที่ 39 เมนูสร้างไฟล์รายงาน

5.1 เข้าใช้งานเมนู ดูรายงานนักเรียน จะสร้างไฟล์รายงานข้อมูลของนักเรียนและแสดงรายงานเบื้องต้น สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 1

รายงานนักศึกษา						
STUDENT ID	FIRST NAME	LAST NAME	MAJOR	YEAR	STATUS	
STU796973	Thanapon	Suwannarat	English	4	ถอน	
STU705257	Supaporn	Wongyai	Computer Science	1	ลงทะเบียน	
STU442194	Nattapong	Phromdee	Biology	4	ถอน	
STU569818	Kritsada	Meechai	Economics	2	ถอน	
STU942291	Nareerat	Khamphol	Economics	2	ถอน	
STU778182	Chatchai	Wongyai	Economics	2	ถอน	

ภาพที่ 40 ดูรายงานนักเรียน

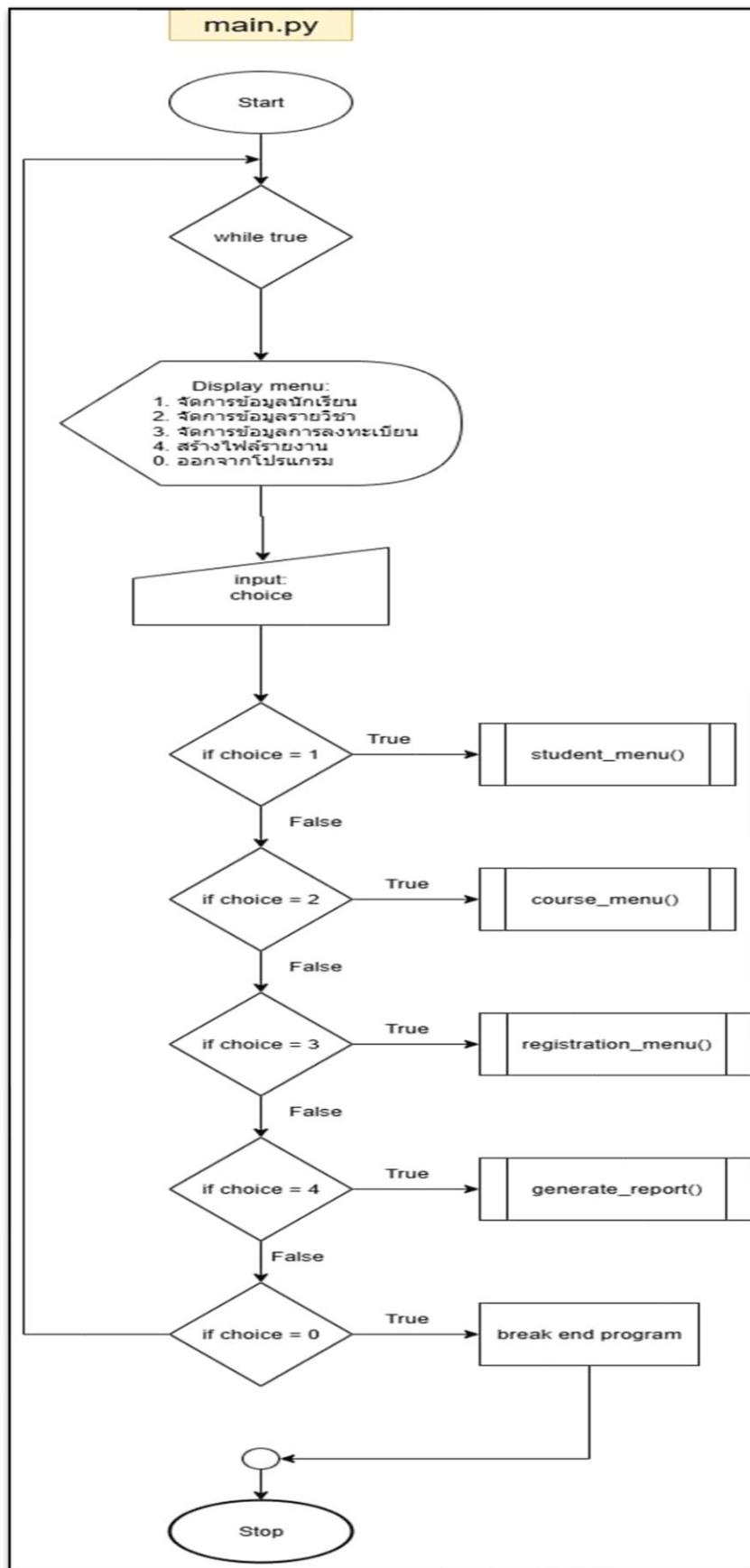
5.2 เข้าใช้งานเมนู ดูรายงานการลงทะเบียน จะสร้างไฟล์รายงานข้อมูลลงทะเบียนและแสดงรายงานการลงทะเบียนเบื้องต้น สามารถเลือกเข้าใช้งานได้โดยพิมพ์เลข 2

--- เนื้อหาผ่าน ---						
1. ดูรายงานผู้เรียน						
2. ดูรายงานการลงทะเบียน						
3. ย้อนกลับไปหน้าแรก						
เลือก: 2						
=====						
รายงานการลงทะเบียน						
=====						
สร้างเมื่อ: 2025-09-30 23:28:27						
วิชา: SCI591 - Machine Learning 4 [ปีการศึกษา 2568, ภาคเรียน 2]						
จำนวน: 1						
STUDENT ID	FIRST NAME	LAST NAME	MAJOR	YEAR	REGISTRATION DATE	STATUS
STU818977	Supaporn	Khamphol	Engineering	4	2025-05-27	ลงทะเบียน
STU242278	Nattapong	Rattanakorn	Physics	4	2025-06-16	ลงทะเบียน
STU535097	Kritsada	Wongyai	Engineering	4	2025-04-17	ลงทะเบียน

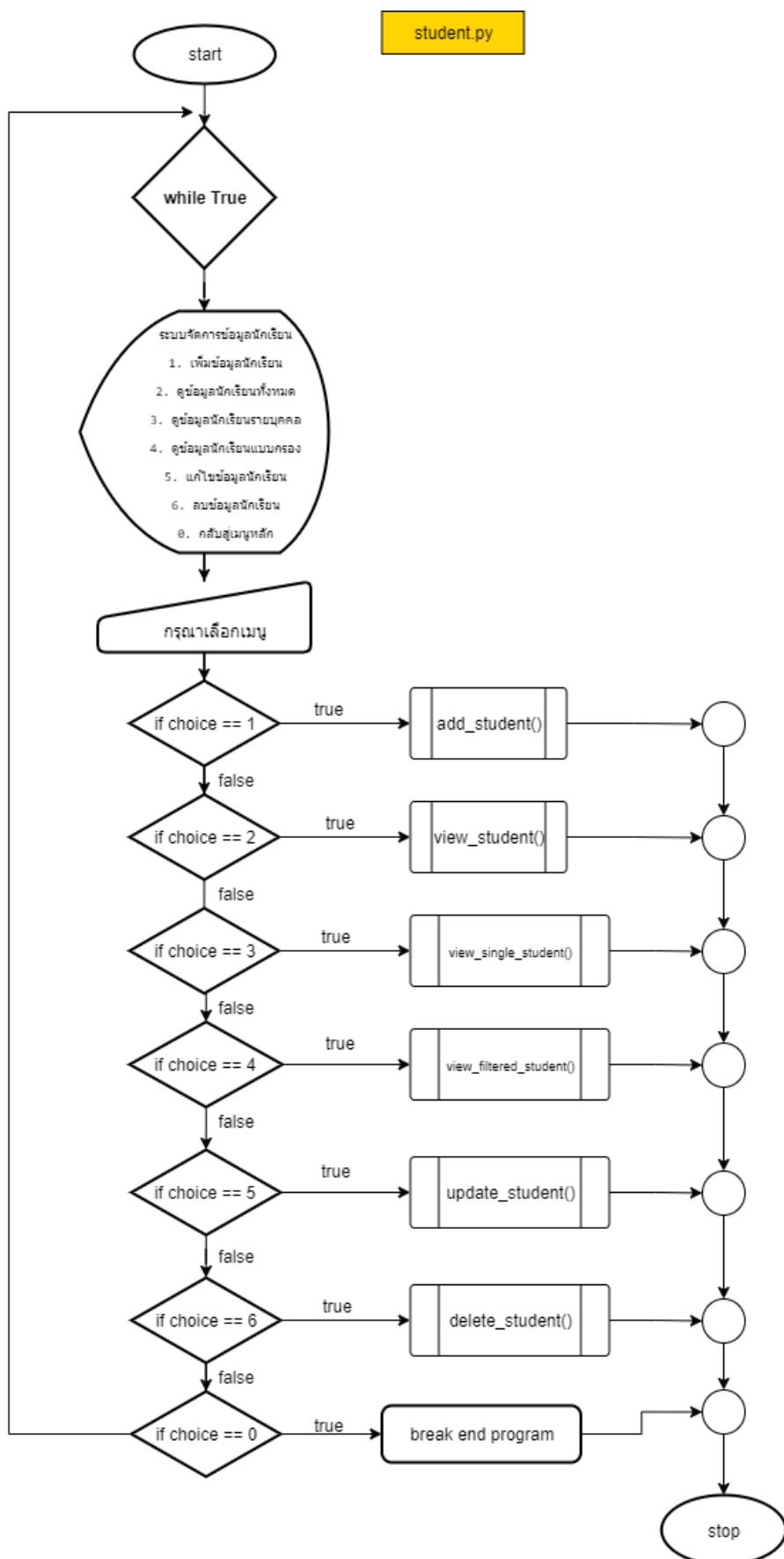
ภาพที่ 41 ดูรายงานการลงทะเบียน

ภาคผนวก

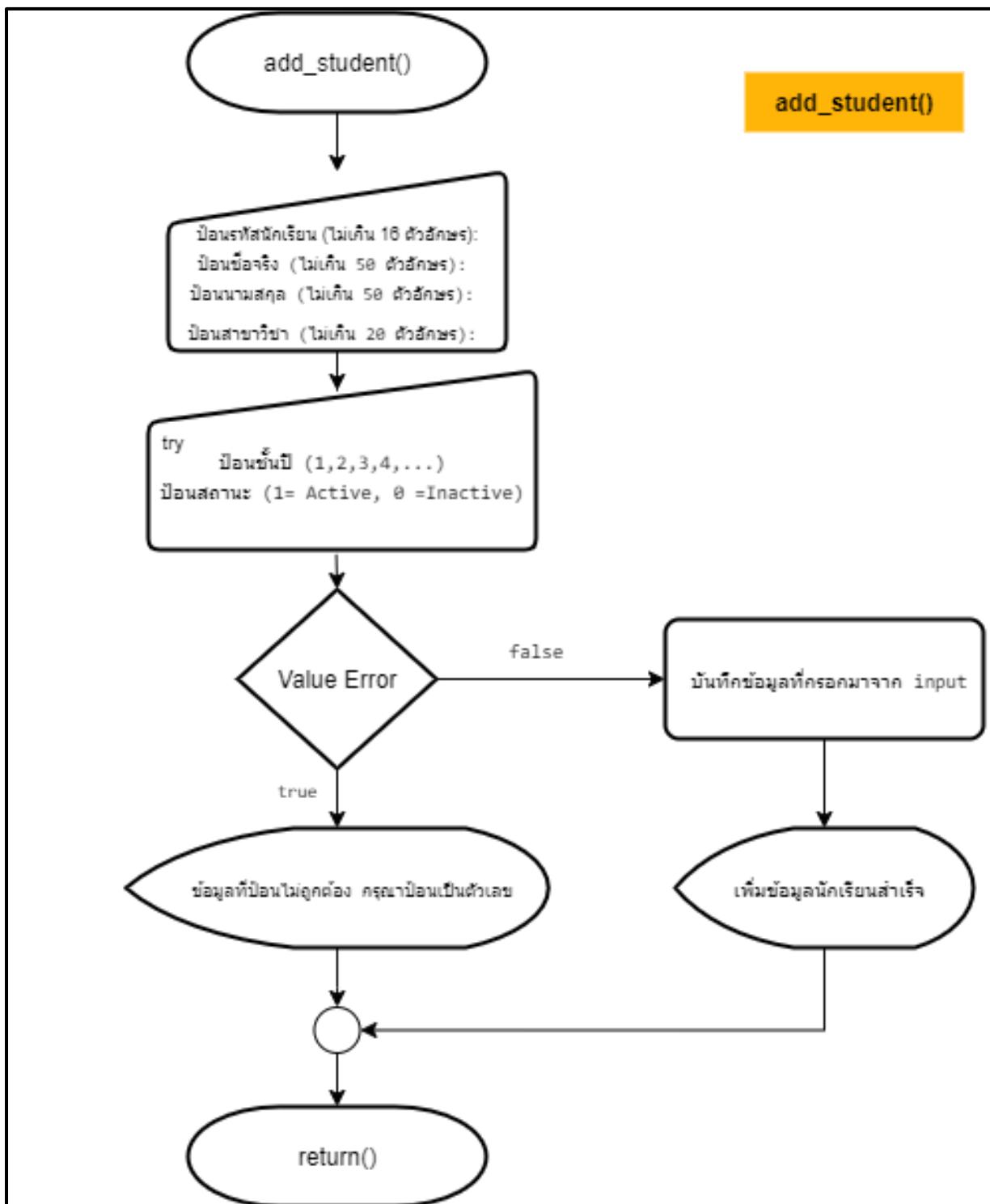
แผนผังการทำงานของโปรแกรม



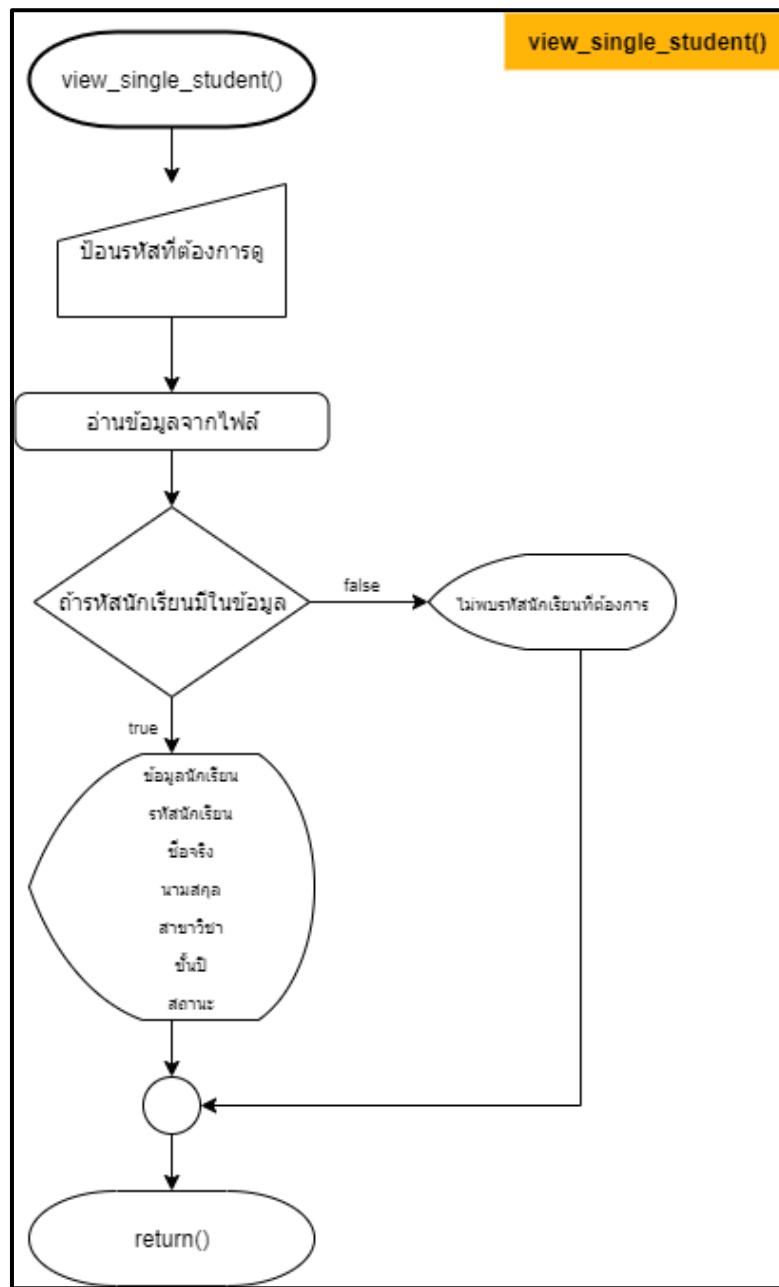
ผังการทำงานของโปรแกรม Flowchart ของ student.py



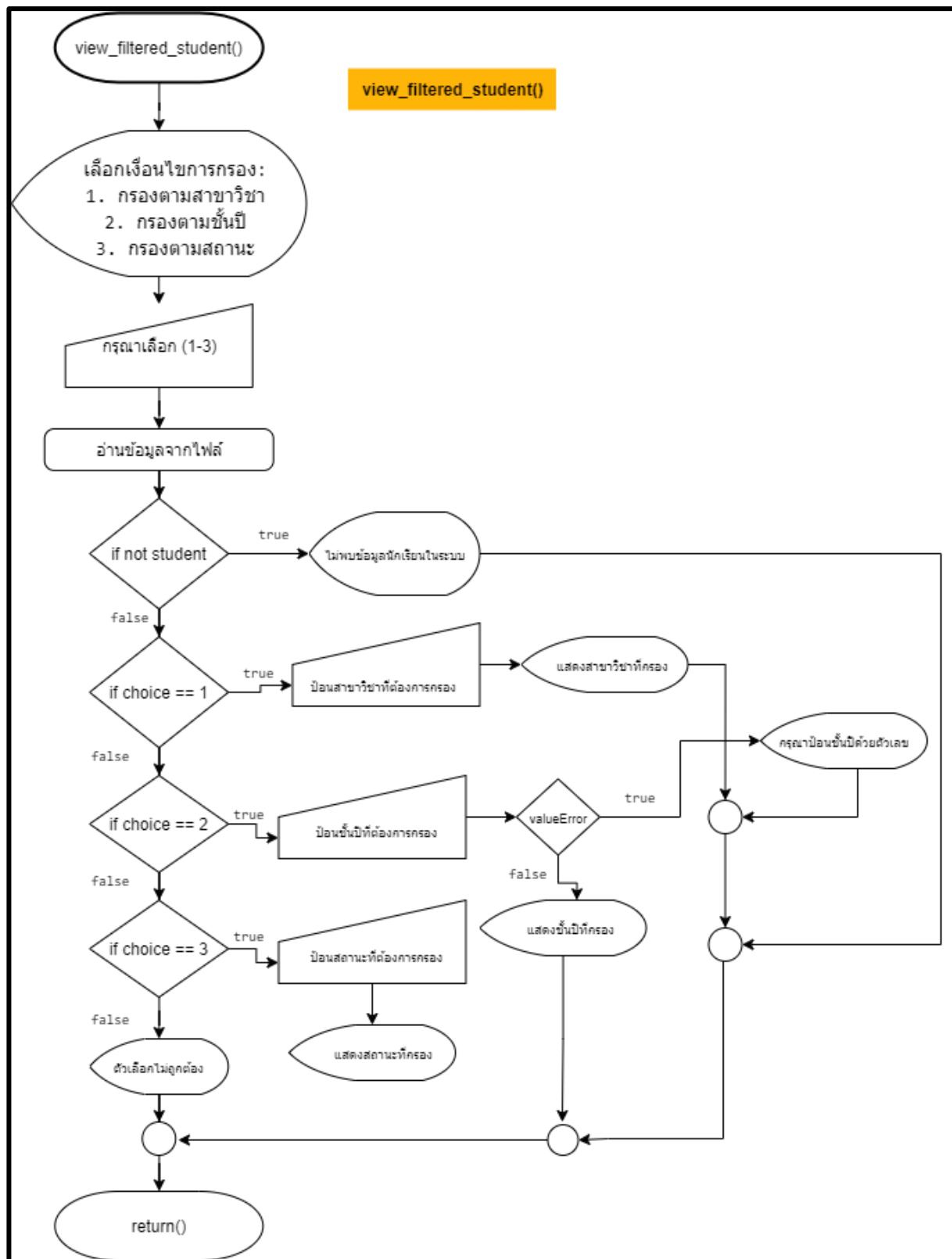
ผังการทำงานของโปรแกรม Flowchart ของ add_student ใน student.py



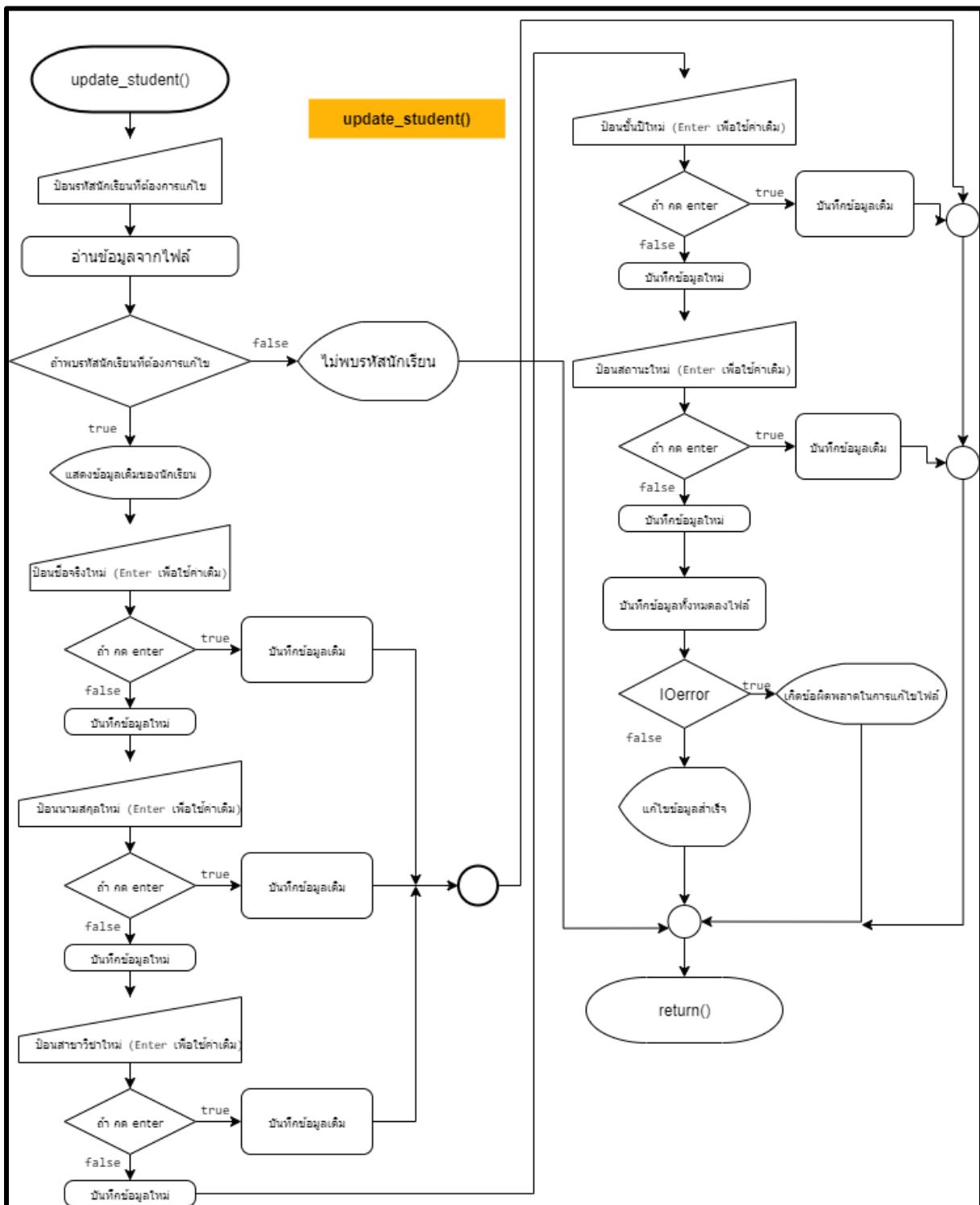
ผังการทำงานของโปรแกรม Flowchart ของ view_single_student ใน student.py



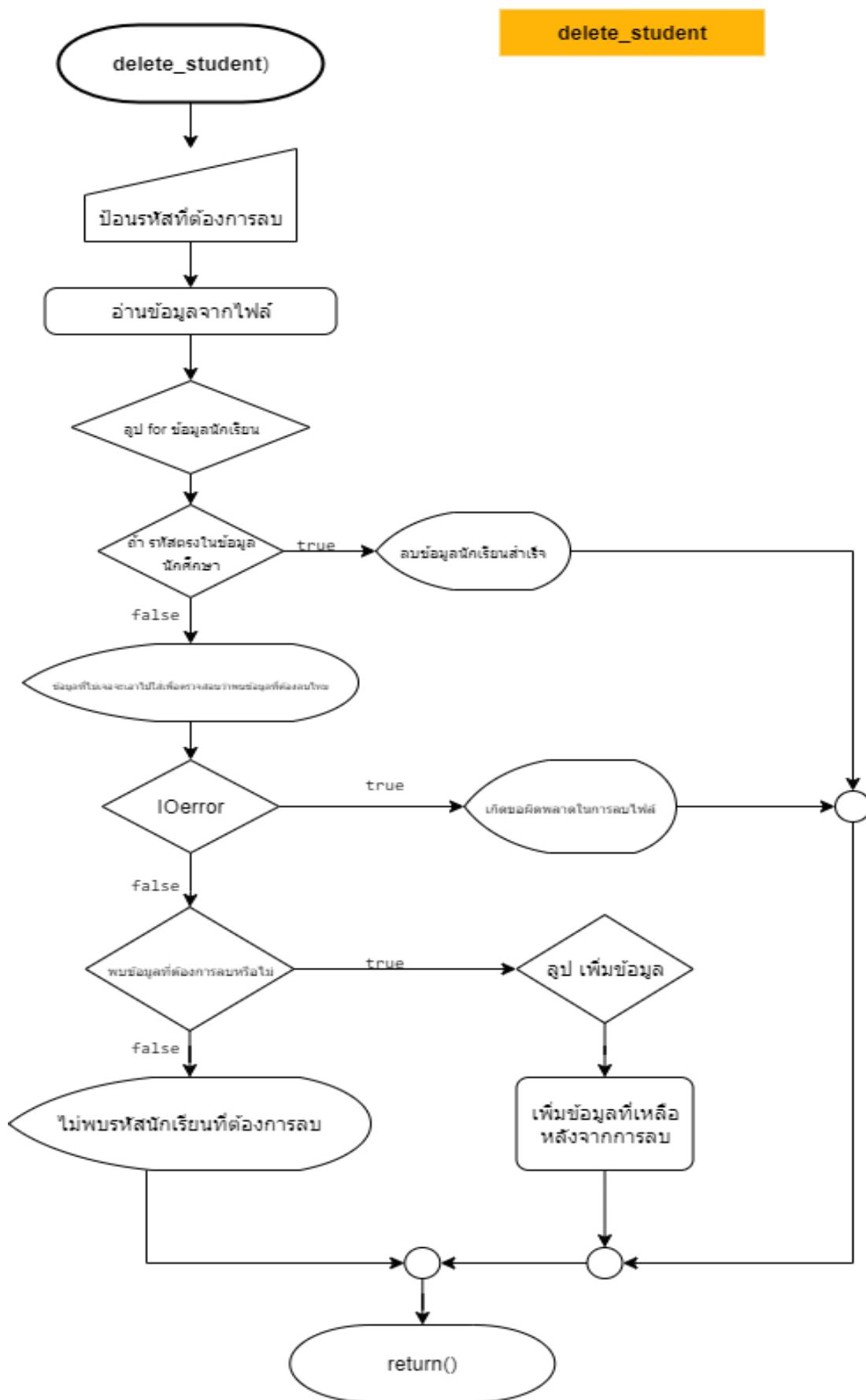
ผังการทำงานของโปรแกรม Flowchart ของ view_filtered_student ใน student.py



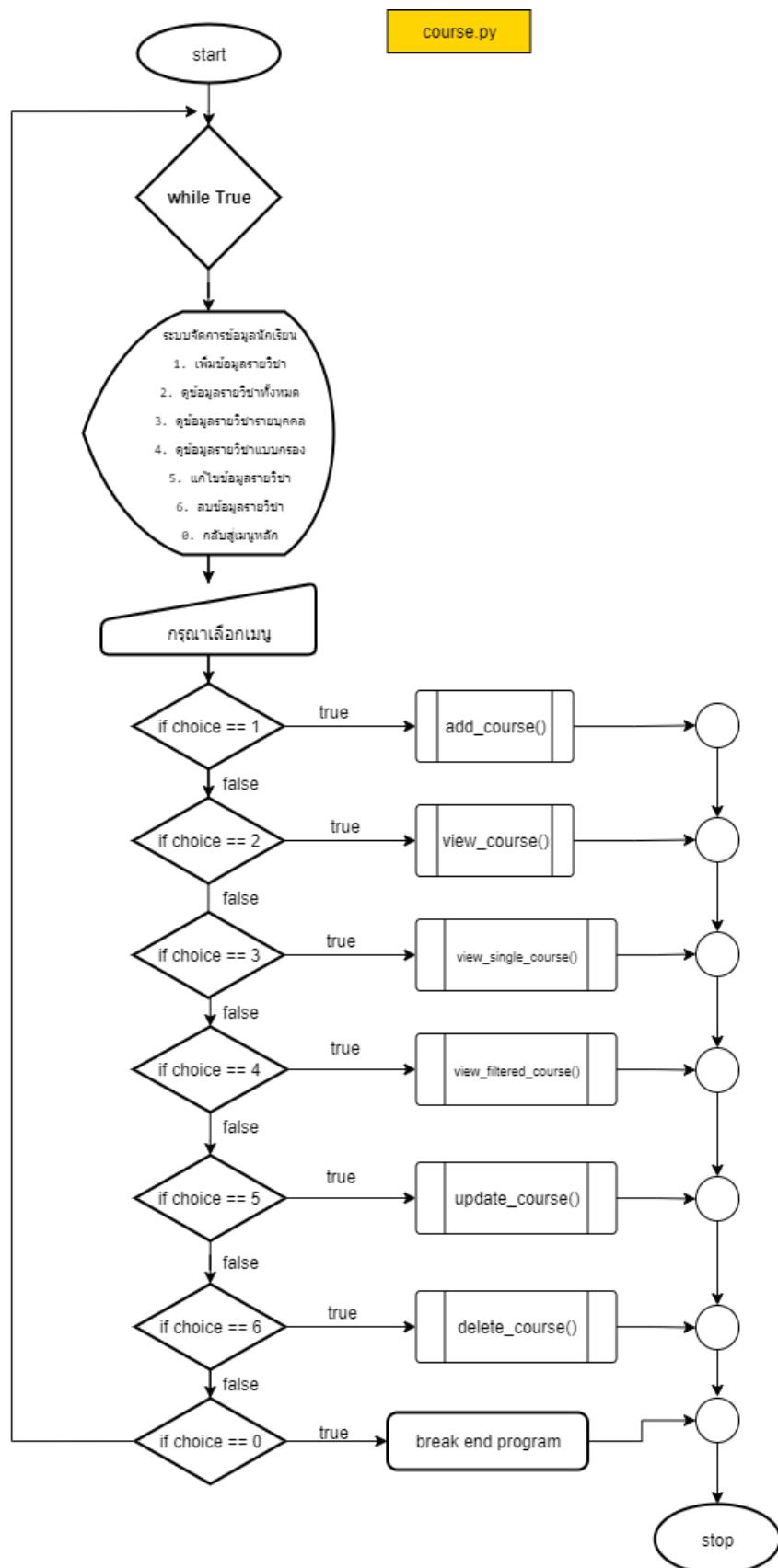
ผังการทำงานของโปรแกรม Flowchart ของ student.py



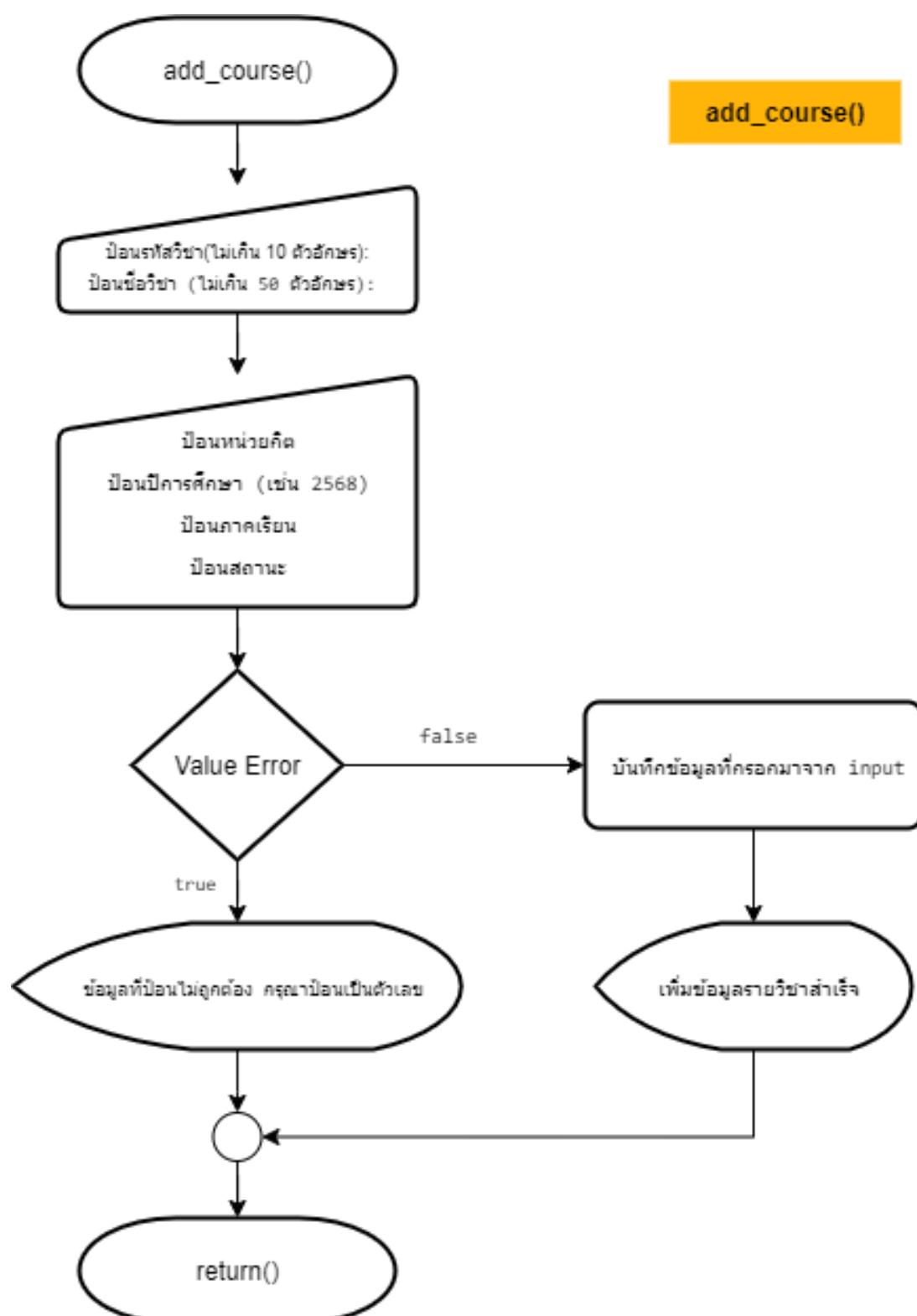
ผังการทำงานของโปรแกรม Flowchart ของ delete_student ใน student.py



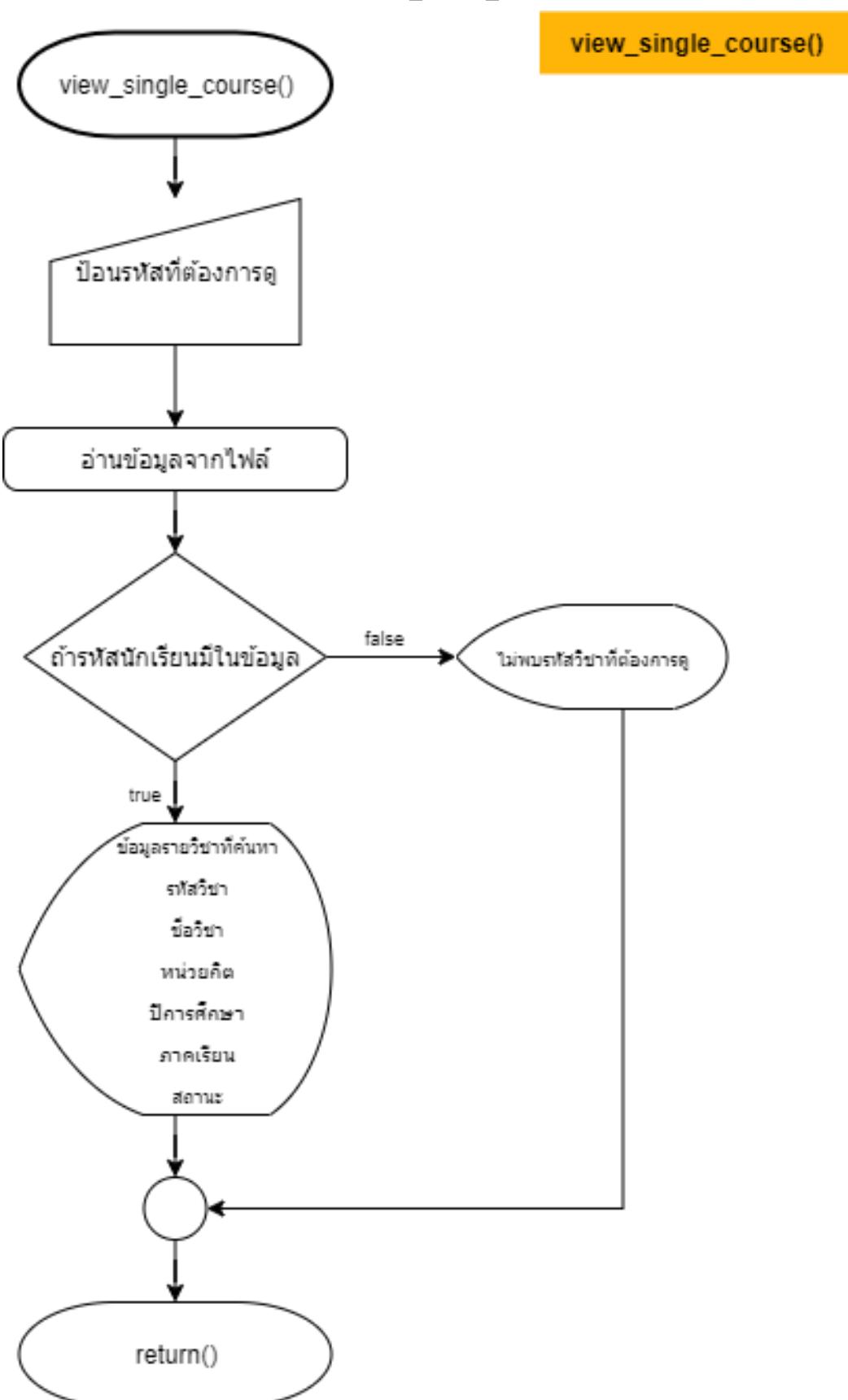
ผังการทำงานของโปรแกรม Flowchart ของ course.py



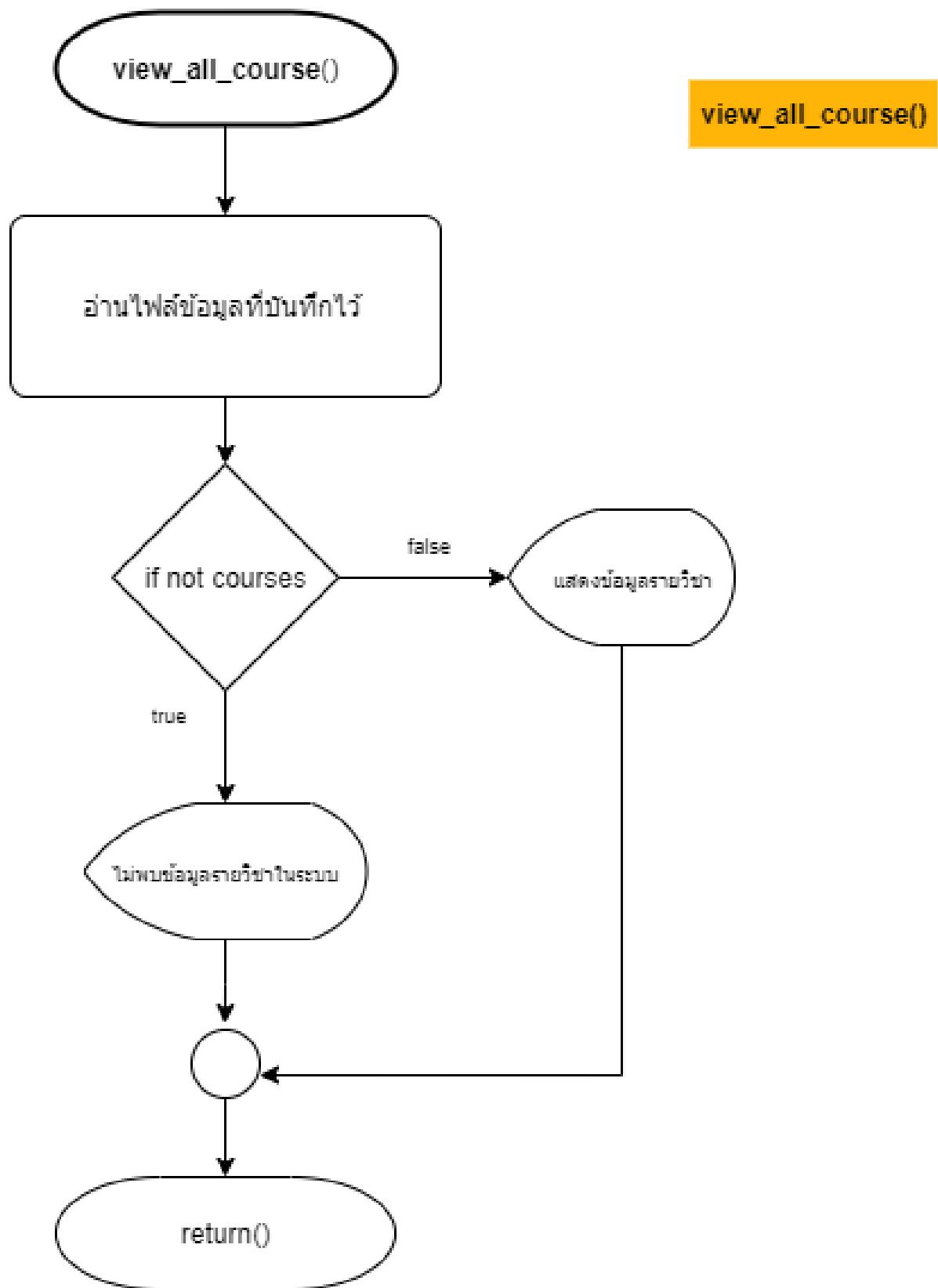
ผังการทำงานของโปรแกรม Flowchart ของ add_course ใน course.py



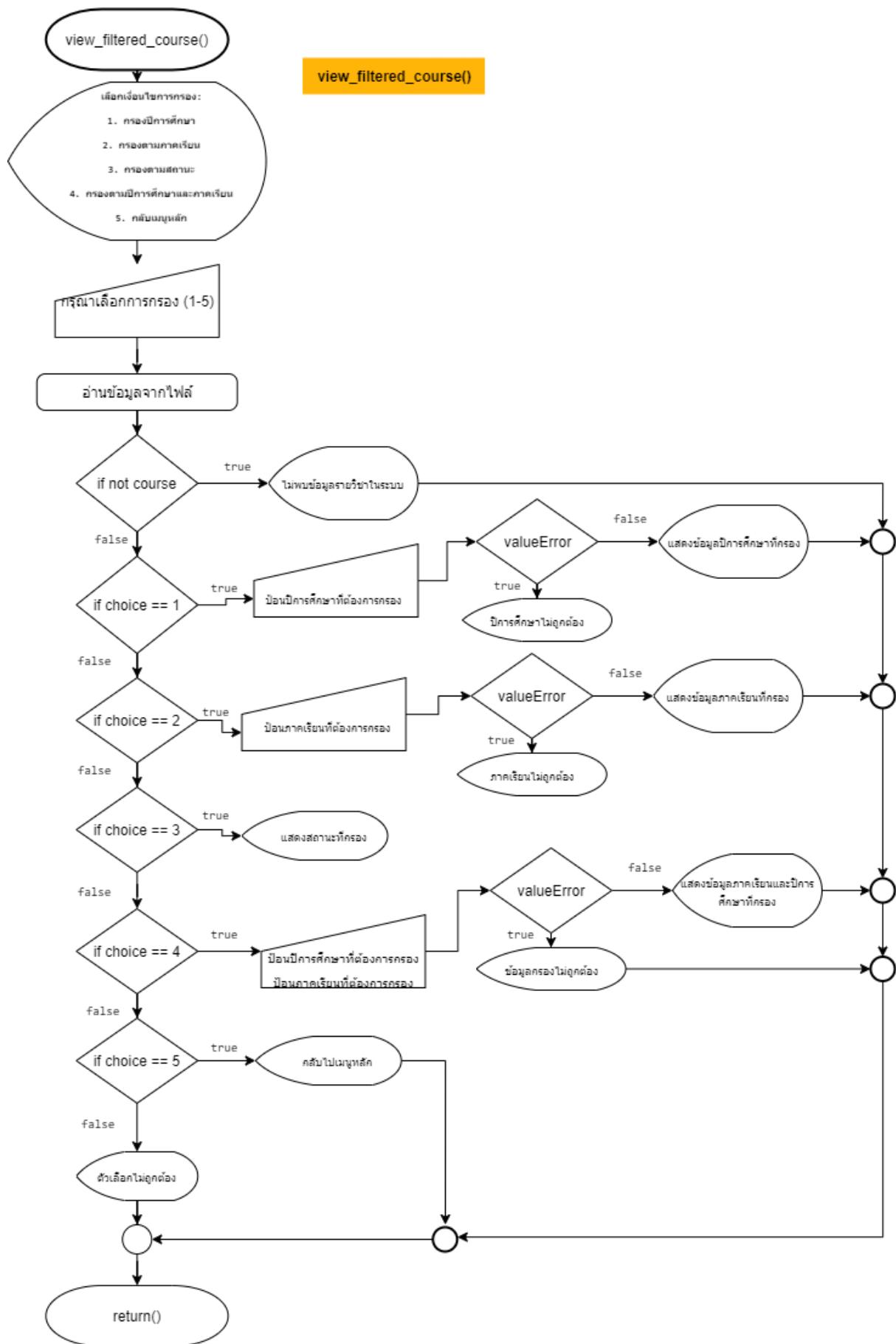
ผังการทำงานของโปรแกรม Flowchart ของ view_single_course ใน course.py



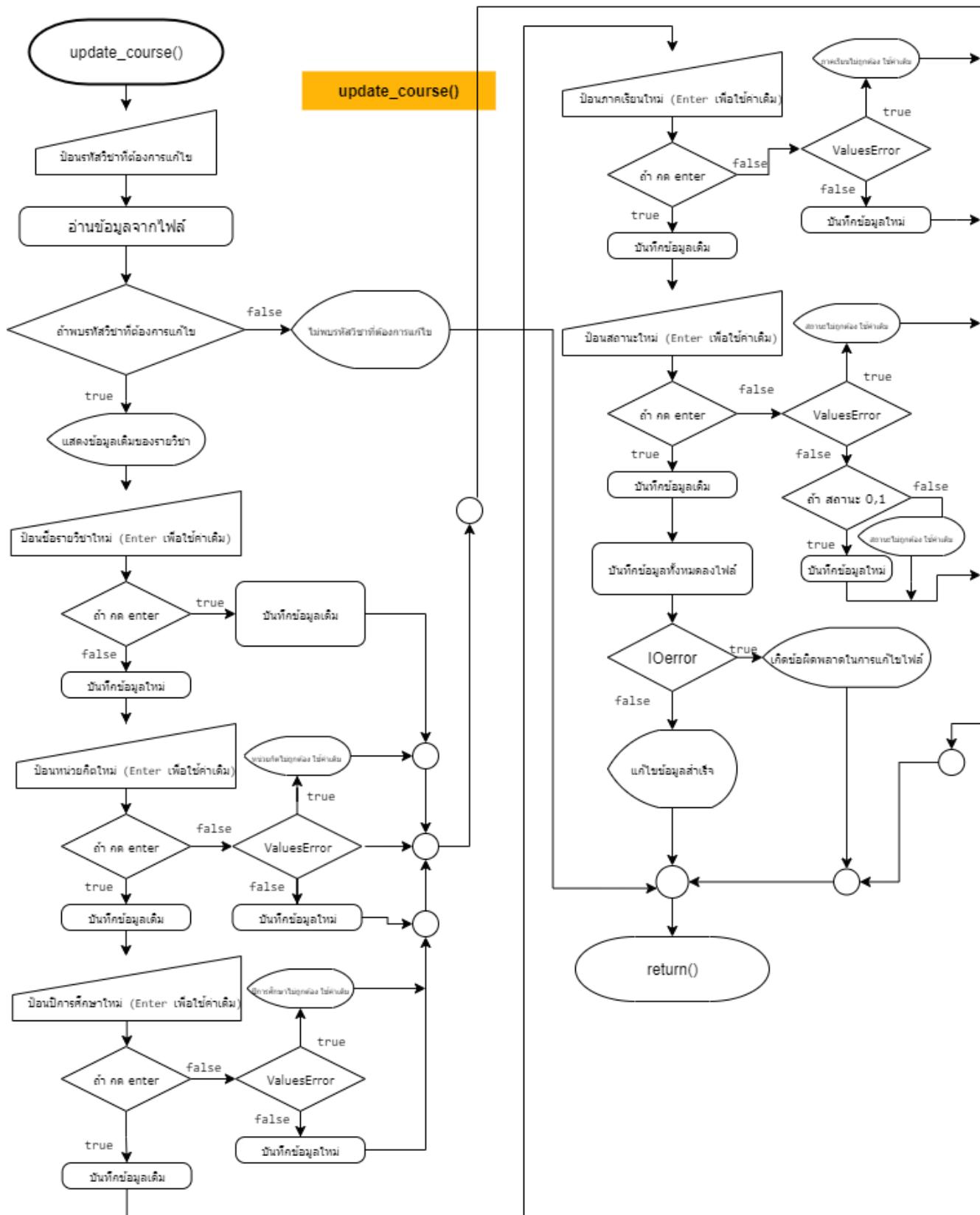
ผังการทำงานของโปรแกรม Flowchart ของ view_all_course ใน course.py



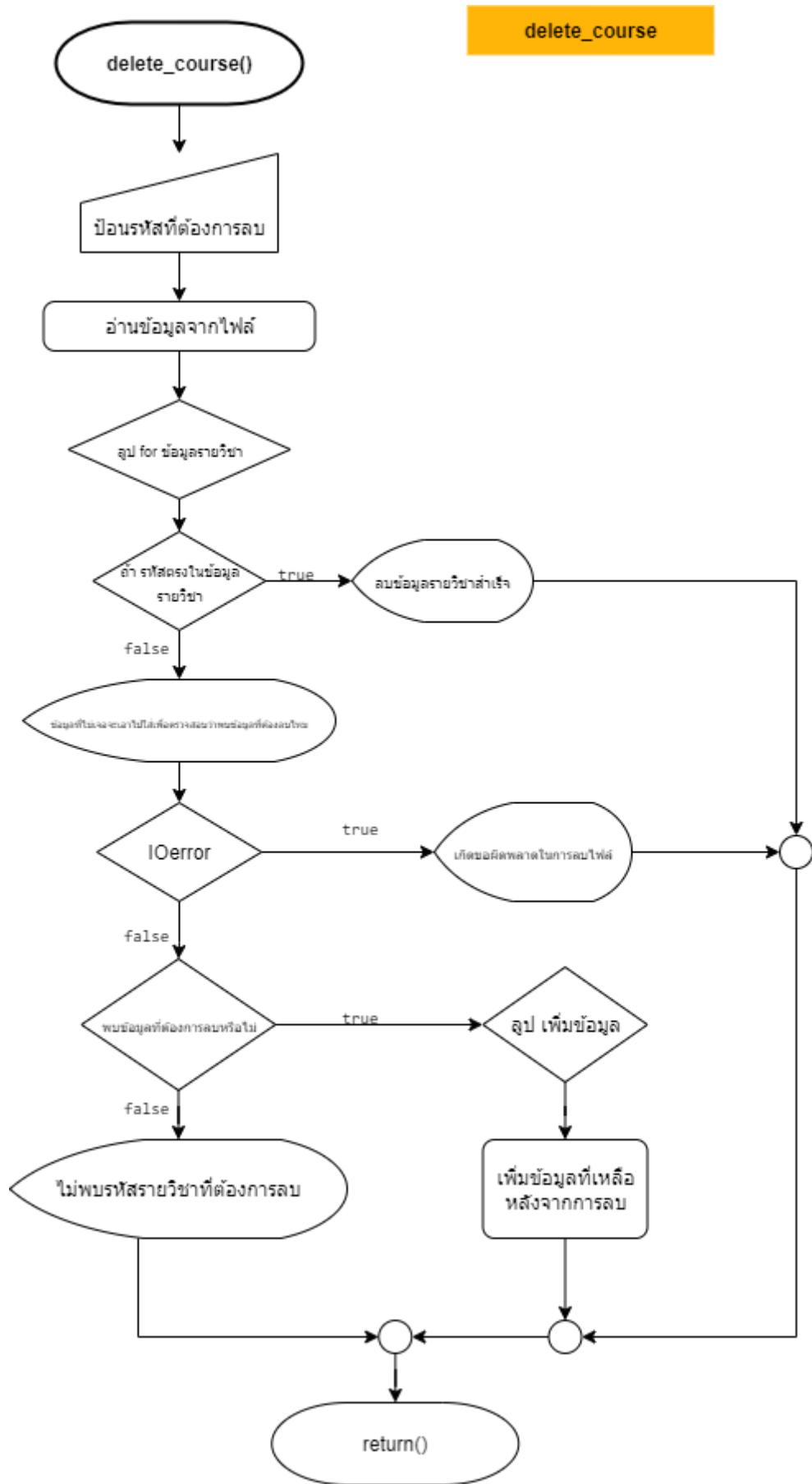
ผังการทำงานของโปรแกรม Flowchart ของ view_filtered_course ใน course.py



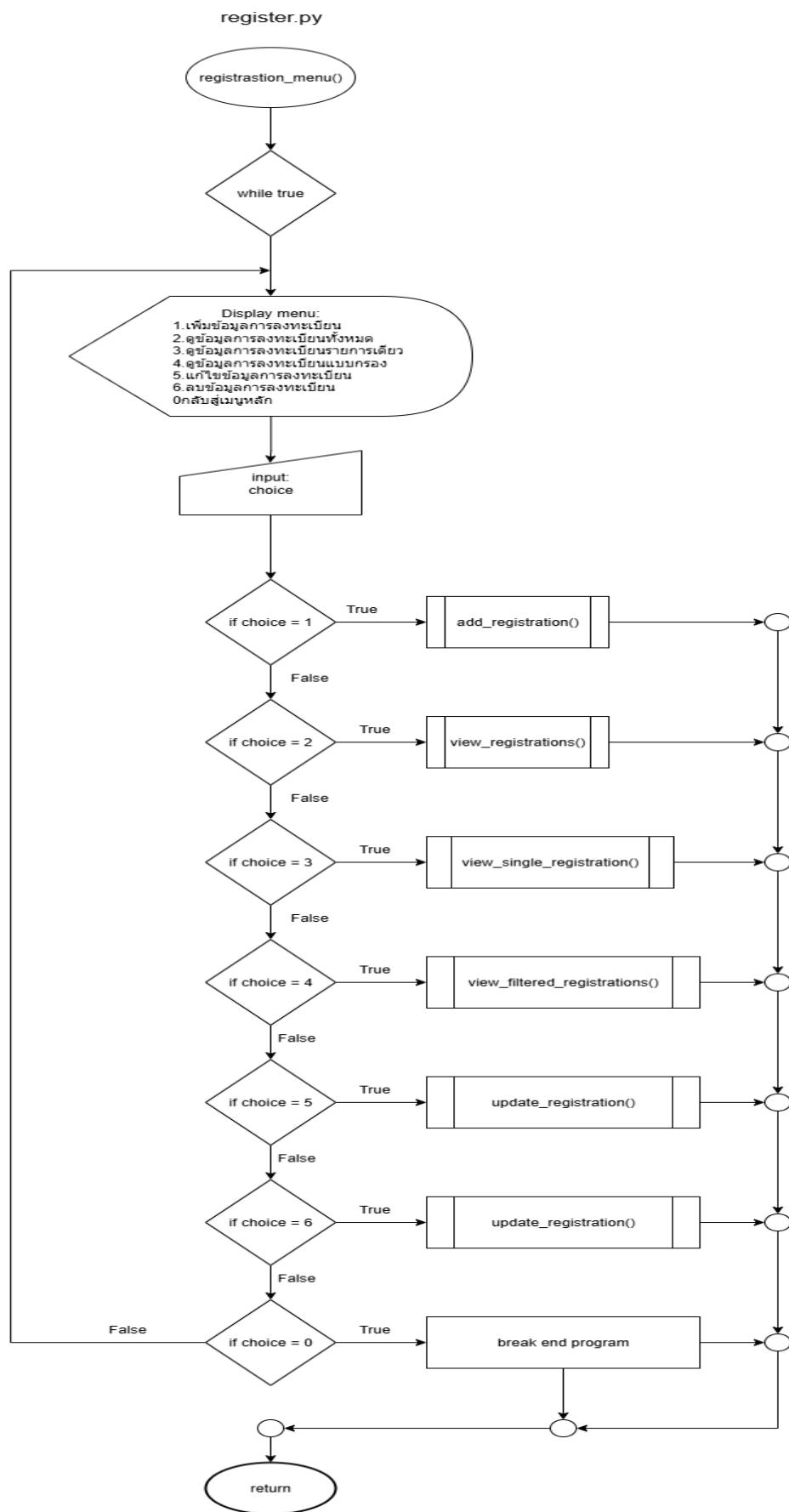
ผังการทำงานของโปรแกรม Flowchart ของ update_course ใน course.py



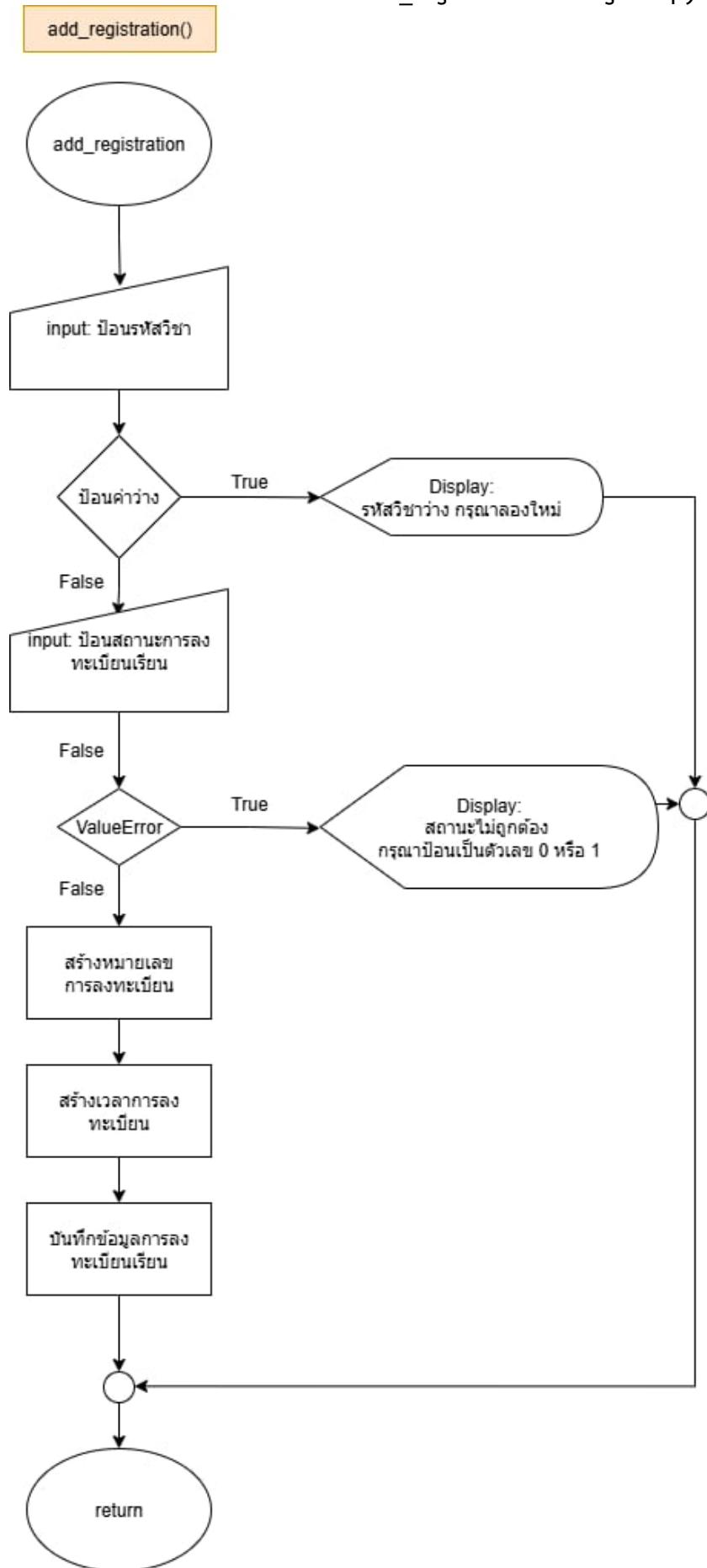
ผังการทำงานของโปรแกรม Flowchart ของ delete_course ใน course.py



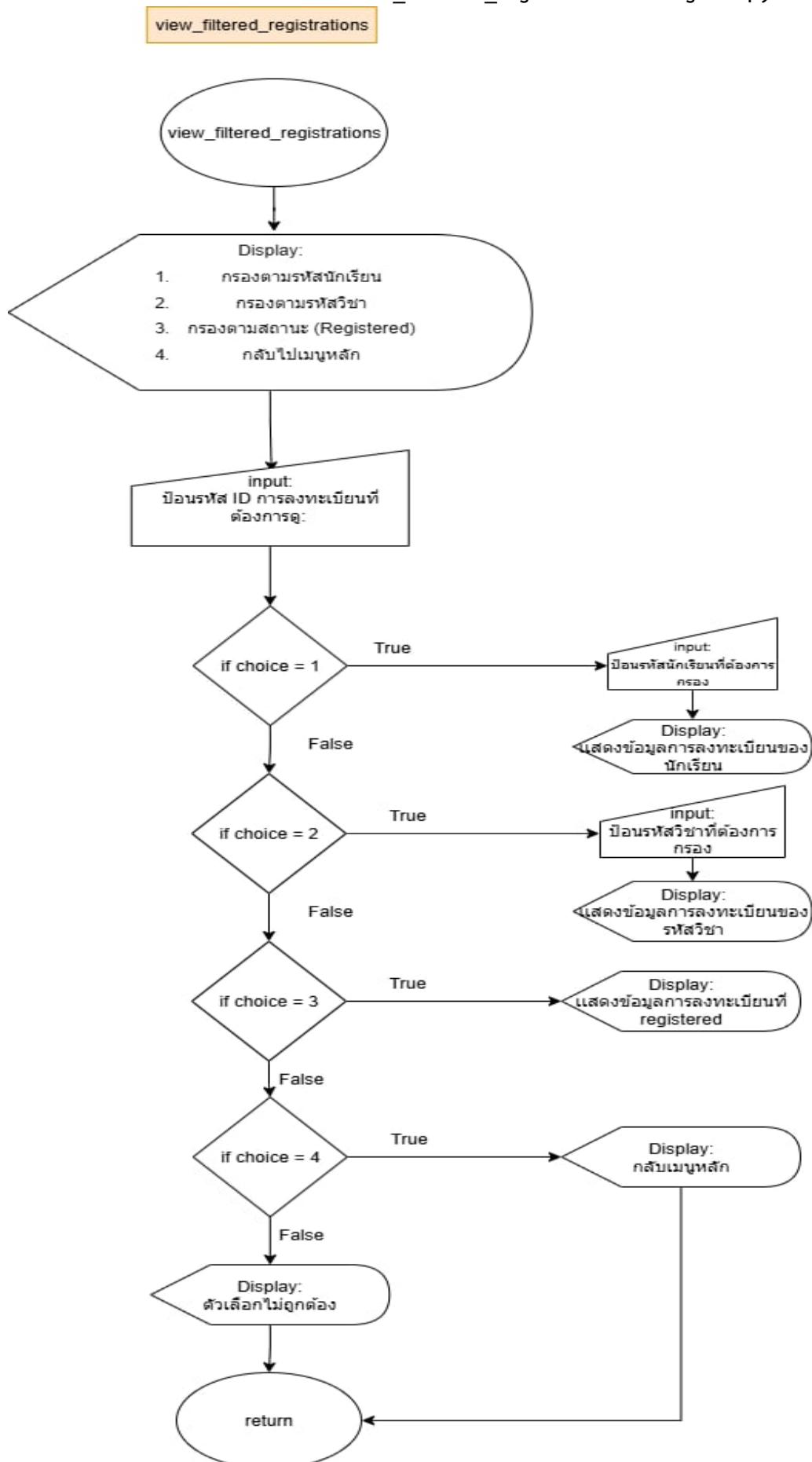
ผังการทำงานของโปรแกรม Flowchart ของ register.py



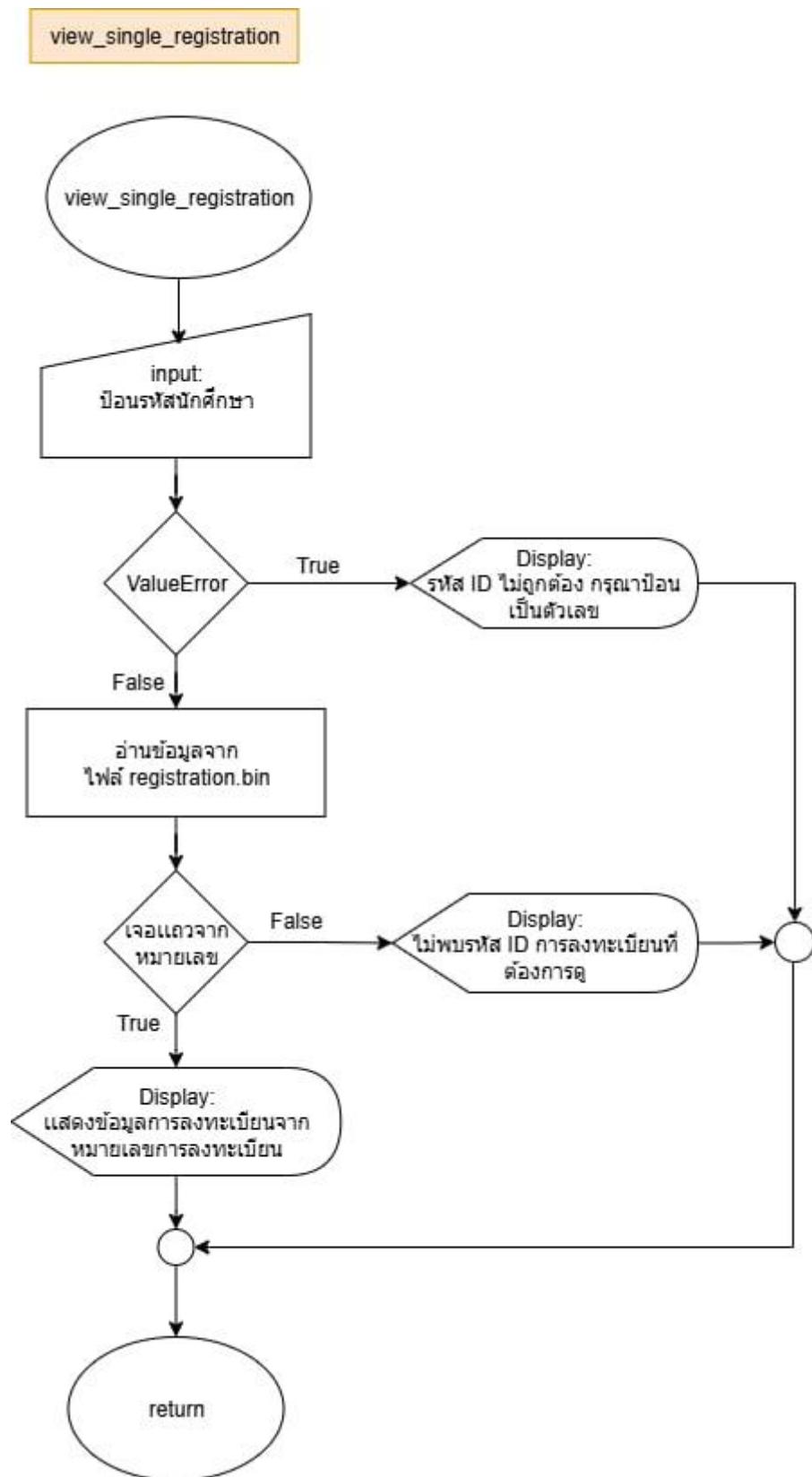
ผังการทำงานของโปรแกรม Flowchart ของ add_registration ใน register.py



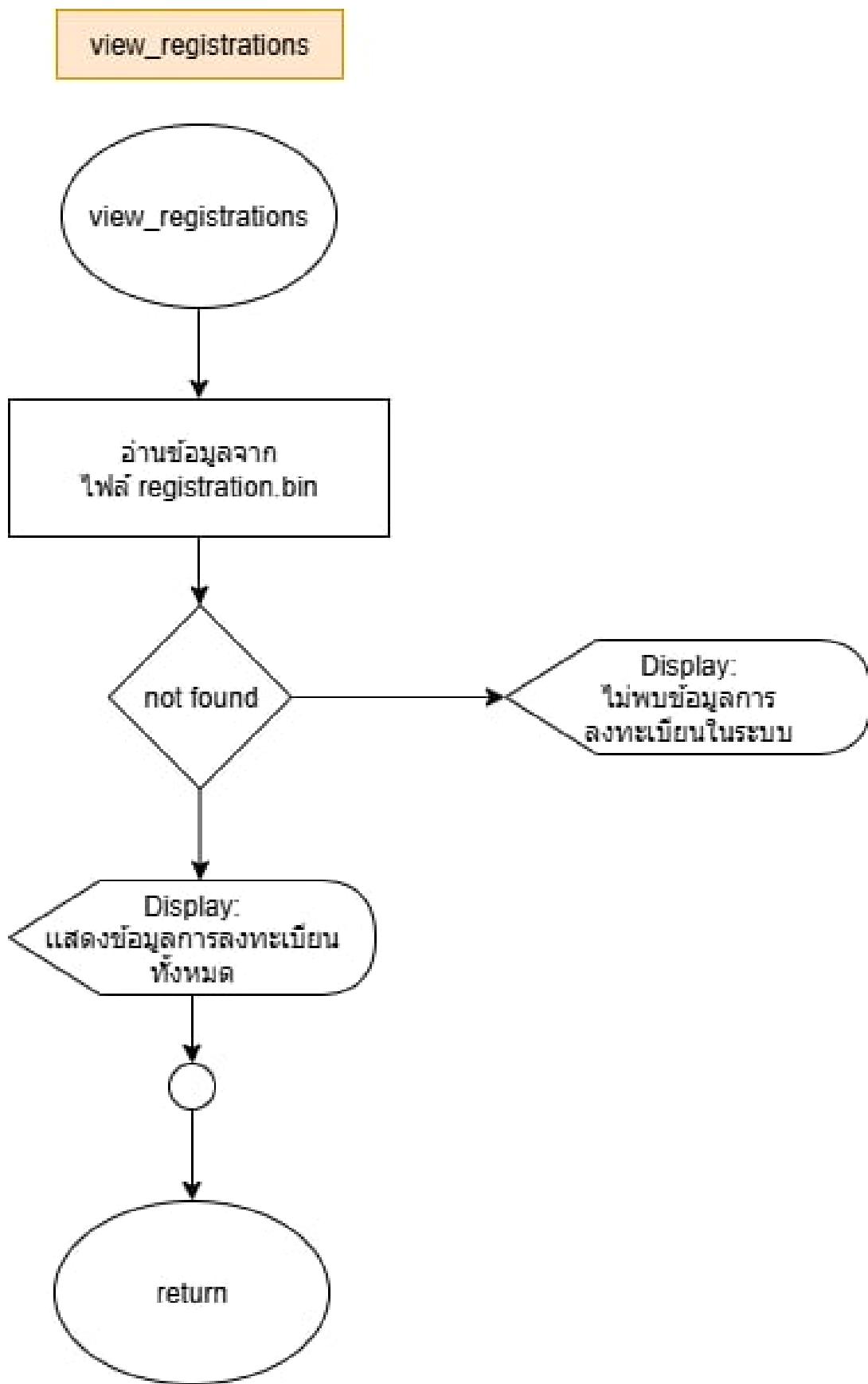
ผังการทำงานของโปรแกรม Flowchart ของ view_filtered_registrations ใน register.py



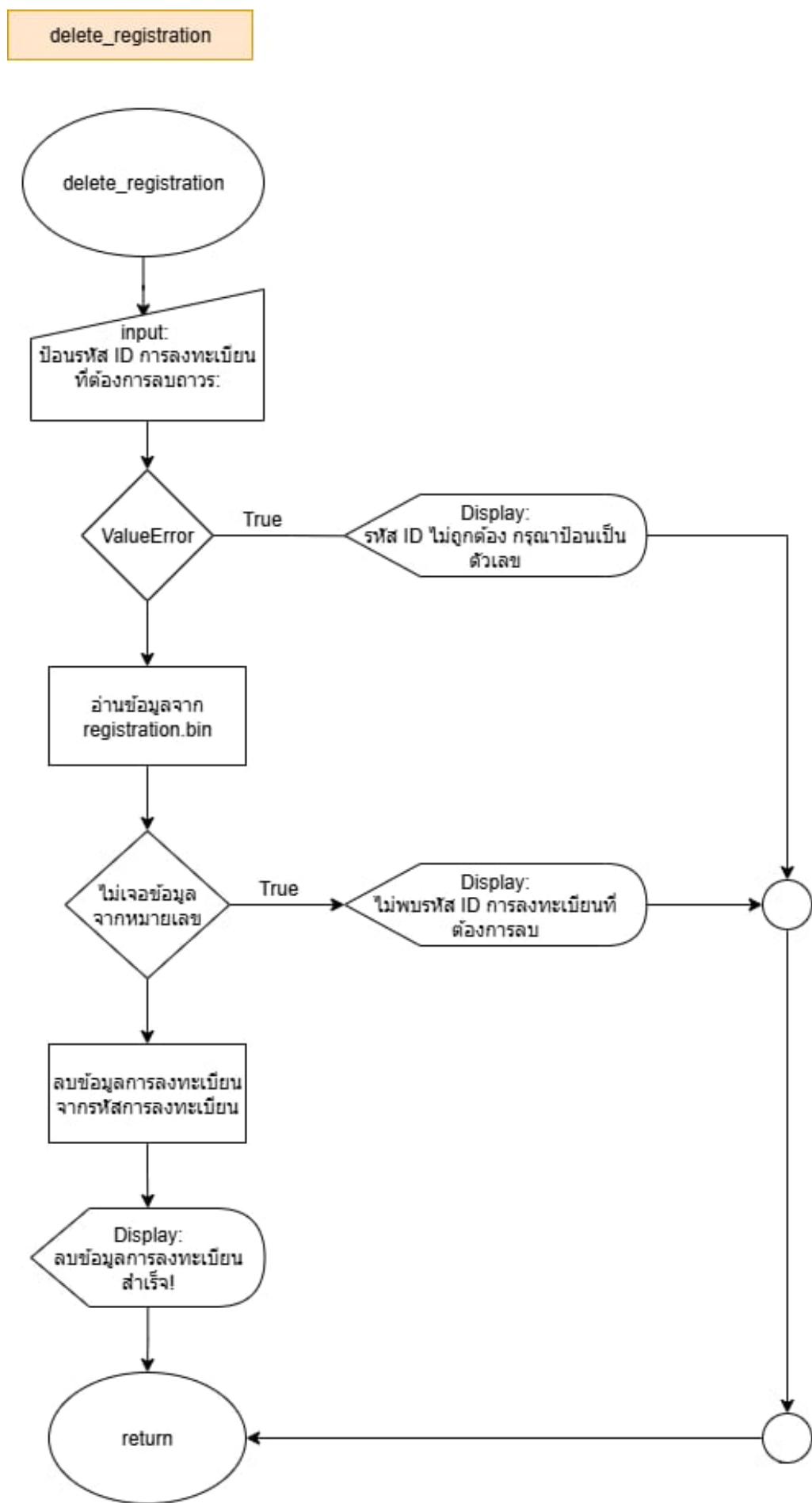
ผังการทำงานของโปรแกรม Flowchart ของ view_single_registration ใน register.py



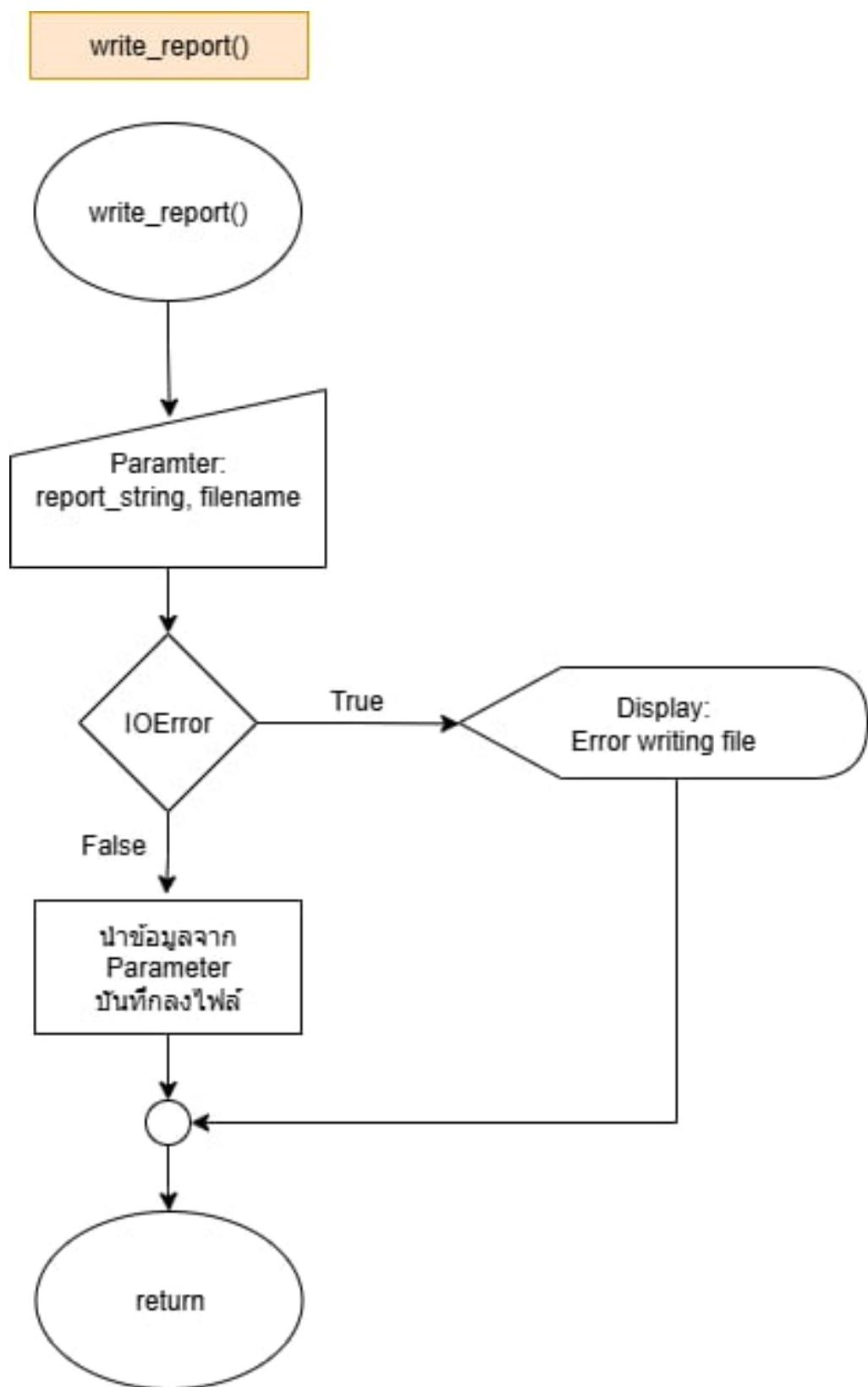
ผังการทำงานของโปรแกรม Flowchart ของ view_registrations ใน register.py



ผังการทำงานของโปรแกรม Flowchart ของ delete_registrations ใน register.py



ผังการทำงานของโปรแกรม Flowchart ของ write_report ใน report.py



ผังการทำงานของโปรแกรม Flowchart ของ generate_report ใน report.py

