

เอกสารประกอบการสอน

ปรับปรุง พ.ศ. 2567

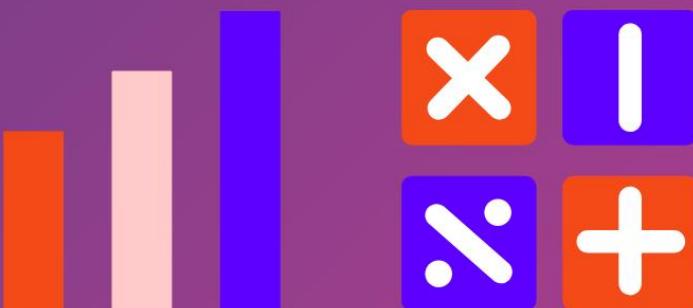
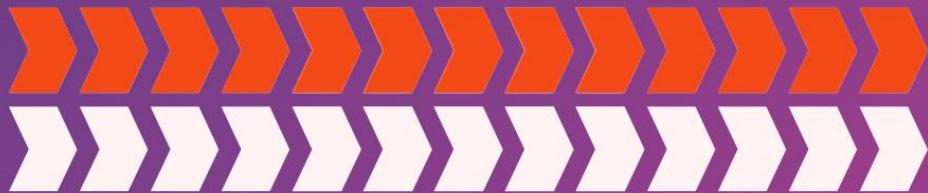
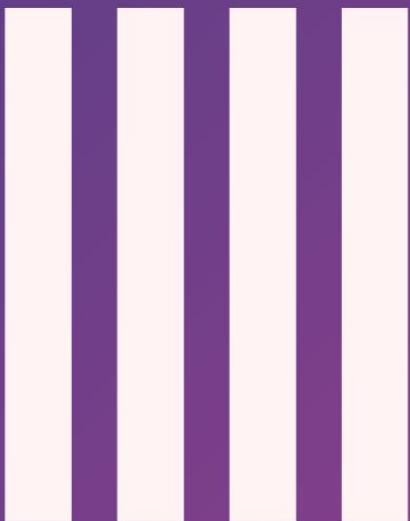
รายวิชา

การโปรแกรมคอมพิวเตอร์



EN-001-002

Computer Programming



ดร. สรายุทธ กรณ์รัตน์

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม

มหาวิทยาลัยกาฬสินธุ์

พ.ศ. 2566

รายวิชา **การโปรแกรมคอมพิวเตอร์** นี้ คือจุดเริ่มต้นที่สมบูรณ์แบบ สำหรับทุกคนที่สนใจโลกแห่งการเขียนโปรแกรม โดยใช้ภาษา PYTHON ซึ่ง เป็นภาษาที่เรียบง่ายแต่ทรงพลัง ผู้เรียนจะได้สัมผัสถึงเนื้อหาที่เข้าใจง่าย ครอบคลุมตั้งแต่พื้นฐานจนถึงเทคนิคขั้นสูง พร้อมตัวอย่างโค้ดที่หลากหลาย ไม่ว่าจะเป็นมือใหม่หรือผู้มีประสบการณ์สามารถเรียนรู้และนำไปประยุกต์ ใช้ได้จริง

นอกจากเนื้อหาพื้นฐานแล้ว รายวิชานี้ยังปูพื้นฐานการประยุกต์ใช้ PYTHON ในด้านต่างๆ เช่น

- **วิศวกรรม:** เรียนรู้การจำลองแบบระบบ และการควบคุมอัตโนมัติเบื้องต้น
- **การวิเคราะห์ข้อมูล:** ฝึกฝนการสร้างกราฟ และการวิเคราะห์แนวโน้มข้อมูล
- **IOT:** เข้าใจหลักการควบคุมอุปกรณ์ และการรับส่งข้อมูลในระบบ IOT
- **AI:** การประมวลผลข้อมูลและแสดงผล

ซึ่งจะเป็นประโยชน์อย่างยิ่งสำหรับผู้เริ่มต้นที่ต้องการต่อยอด และนำไปประยุกต์ใช้ในสายงานต่างๆ ต่อไปในอนาคต ด้วยการอธิบายที่ เป็นขั้นตอน ภาพประกอบ และตัวอย่างโค้ด ผู้เรียนจะเข้าใจเนื้อหาได้อย่าง รวดเร็ว พร้อมพัฒนาฝีมือการเขียนโปรแกรม PYTHON และก้าวสู่โลก เทคโนโลยีอย่างมั่นใจ

*“Success is the sum of small efforts,
repeated day in and day out.”*

– Robert Collier

ความสำเร็จคือผลรวมของความพยายามเล็กๆ
น้อยๆ ที่กำช้าๆ กันทุกวัน

“เช่นเดียวกัน การเรียนรู้การเขียนโปรแกรมก็เหมือนกับความสำเร็จ ทุกความ
พยายามในการฝึกฝนและแก้ไขปัญหาทีละเล็กละน้อย ช้าๆ ในแต่ละวัน คือ
เส้นทางสู่ความเชี่ยวชาญและความเป็นเลิศในอนาคต”

คำนำ

เอกสารประกอบการสอนรายวิชา การโปรแกรมคอมพิวเตอร์ รหัสวิชา EN-001-002 จัดทำขึ้นเพื่อใช้ประโยชน์ในการสอนของอาจารย์ในรายวิชาการโปรแกรมคอมพิวเตอร์ ตามหลักสูตรปริญญาตรี วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมโลจิสติกส์ และสาขาวิศวกรรมอุตสาหการ โดยเนื้อหาเอกสารนี้ได้ใช้ภาษาไทย - รอน (Python)

ผู้จัดทำเอกสารประกอบการสอนเล่มนี้มีความตั้งใจและมุ่งหวังอย่างยิ่งว่าจะสามารถตอบความรู้และความเข้าใจในหลักการเขียนโปรแกรมภาษา Python ตั้งแต่พื้นฐานจนถึงการประยุกต์ใช้งานจริง เช่น การแก้ปัญหาด้านวิศวกรรม การคำนวณหาเส้นทางที่สั้นที่สุด (TSP) และการพัฒนาระบบ Internet of Things (IoT) เพื่อใช้เป็นพื้นฐานความรู้สำหรับผู้เรียนในการต่อยอดสู่การเป็นวิศวกรที่มีความรู้ ความสามารถ และสามารถนำไปใช้ในการแก้ปัญหาในงานวิศวกรรมได้อย่างมีประสิทธิภาพ

ทั้งนี้ ผู้จัดทำหวังเป็นอย่างยิ่งว่าเอกสารประกอบการสอนเล่มนี้จะเป็นประโยชน์แก่ทั้งอาจารย์และนักศึกษา โดยอาจารย์สามารถนำไปใช้เป็นคู่มือประกอบการสอน และนักศึกษางานได้เป็นแหล่งอ้างอิงเพื่อพัฒนาความรู้ในด้านการเขียนโปรแกรมอย่างต่อเนื่อง หากเอกสารฉบับนี้มีข้อผิดพลาดประการใด ผู้จัดทำขออภัยมา ณ โอกาสนี้ และยินดีรับข้อเสนอแนะเพื่อการปรับปรุงในอนาคต

ขอแสดงความนับถือ
นายกรุ๊ป ภูรัตน์

รายละเอียดรายวิชา (Course Syllabus)

ชื่อสถาบันอุดมศึกษา	มหาวิทยาลัยกาฬสินธุ์
คณะ	วิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
หมวดที่ 1 : ข้อมูลก่อไป	
รหัสและชื่อรายวิชา	EN-001-002
	การโปรแกรมคอมพิวเตอร์ (Computer Programming)
สภาพรายวิชา	กลุ่มวิชาพื้นฐานทางวิศวกรรมศาสตร์
ระดับรายวิชา	วิศวกรรมศาสตรบัณฑิต
เวลาศึกษา	75 คาบเรียนตลอด 15 สัปดาห์ ทฤษฎี 30 คาบ ปฏิบัติ 45 คาบต่อสัปดาห์ และนักศึกษาจะต้องใช้เวลาศึกษาค้นคว้านอกเวลา 6 ชั่วโมงต่อสัปดาห์
หน่วยกิต	3(2-3-6) บรรยาย 2 ชั่วโมง ปฏิบัติ 3 ชั่วโมง นอกเวลา 6 ชั่วโมง
หมวดที่ 2 : ข้อมูลเฉพาะรายวิชา	
คำอธิบายรายวิชา	แนวคิดและองค์ประกอบของระบบคอมพิวเตอร์ การอันตรกิริยาระหว่างฮาร์ดแวร์ และซอฟต์แวร์ การเขียนโปรแกรมด้วยภาษาที่เป็นปัจจุบัน ปฏิบัติการเขียนโปรแกรม Computer concepts and computer components; hardware and software interaction; current programming language; programming practices
จุดมุ่งหมายรายวิชา	<ol style="list-style-type: none"> เข้าใจองค์ประกอบของระบบคอมพิวเตอร์และหลักการทำงานพื้นฐาน เข้าใจหลักการเขียนโปรแกรมและแนวคิดพื้นฐานของภาษา Python (Python) ใช้งานเครื่องมือสำหรับการเขียนโปรแกรมและการพัฒนาโปรแกรมด้วย Python พัฒนาโปรแกรมระบบเพื่อแก้ปัญหาทั่วไปและการออกแบบโปรแกรมเชิงโครงสร้าง ประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญหางานด้านวิศวกรรมและการคำนวณทางคณิตศาสตร์ มีวินัย ความมุ่งมั่นและความรับผิดชอบงานที่ได้รับมอบหมาย ตามเวลาที่กำหนด

ผลลัพธ์การเรียนรู้รายวิชา (Course Learning Outcomes: CLOs)

- CLO1 ผู้เรียนเข้าใจองค์ประกอบของระบบคอมพิวเตอร์ที่เป็นพื้นฐานสำหรับการเขียนโปรแกรม
- LLO1-1 อธิบายส่วนประกอบหลักของคอมพิวเตอร์ เช่น CPU หน่วยความจำ และระบบจัดเก็บข้อมูล
 - LLO1-2 อธิบายหลักการทำงานของระบบปฏิบัติการและบทบาทในระบบคอมพิวเตอร์
 - LLO1-3 อธิบายวิัฒนาการและความสำคัญของภาษาคอมพิวเตอร์ รวมถึงภาษา Python
- CLO2 ผู้เรียนสามารถพัฒนาโปรแกรมคอมพิวเตอร์พื้นฐานได้
- LLO2-1 อธิบายแนวคิดพื้นฐานของการเขียนโปรแกรม เช่น ตัวแปร ประเภทข้อมูล และโครงสร้างการควบคุม
 - LLO2-2 เขียนโปรแกรมเบื้องต้น เช่น การแสดงผลข้อความ การคำนวณทางคณิตศาสตร์ และการใช้ลิสต์และดิกชันนารี
 - LLO2-3 ใช้คำสั่งควบคุมโปรแกรม ได้แก่ if if-else loops และฟังก์ชัน เพื่อแก้ปัญหาที่กำหนด
- CLO3 ผู้เรียนสามารถพัฒนาโปรแกรมเชิงโครงสร้างเพื่อแก้ปัญหาทั่วไป
- LLO3-1 ระบุและวิเคราะห์ปัญหา พร้อมออกแบบโปรแกรมด้วยแนวคิด Top-Down Design
 - LLO3-2 ใช้งาน (Flowchart) หรือ Pseudocode ในการวางแผนและพัฒนาโปรแกรม
 - LLO3-3 เขียนโปรแกรมแก้ปัญหาโดยรับอินพุต ประมวลผล และแสดงผลลัพธ์ตามข้อกำหนด
- CLO4 ผู้เรียนสามารถจัดการข้อมูลและใช้งานไลบรารีในการวิเคราะห์และแสดงผลข้อมูล
- LLO4-1 ใช้โครงสร้างข้อมูล ได้แก่ ลิสต์ ดิกชันนารี และ DataFrame ในการจัดการข้อมูล
 - LLO4-2 เขียนโปรแกรมอ่านและเขียนไฟล์ ได้แก่ ไฟล์ข้อความ CSV และ JSON
 - LLO4-3 ใช้ไลบรารีที่ได้รับความนิยม เช่น NumPy Pandas และ Matplotlib ในการวิเคราะห์และแสดงผลข้อมูล
 - LLO4-4 สร้างกราฟแสดงผลข้อมูลเพื่อสนับสนุนการวิเคราะห์
- CLO5 ผู้เรียนสามารถประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญหางานด้านวิศวกรรม
- LLO5-1 เขียนโปรแกรมเพื่อคำนวณทางคณิตศาสตร์และแสดงผลกราฟ
 - LLO5-2 เขียนโปรแกรมเพื่อคำนวณหาเส้นทางที่สั้นที่สุด (TSP)
 - LLO5-3 เขียนโปรแกรมระบบอินเตอร์เน็ตของสรรพสิ่ง (IoT)

การแบ่งหน่วย / บทเรียน / หัวข้อ

บทกี่	รายการ	คาบเรียน	
		ก	จ
1	ความรู้พื้นฐานเกี่ยวกับระบบคอมพิวเตอร์และการเขียนโปรแกรมคอมพิวเตอร์ 1.1 ความสำคัญของการเขียนโปรแกรมคอมพิวเตอร์ 1.2 องค์ประกอบและหลักการทำงานของระบบคอมพิวเตอร์ 1.3 ระบบปฏิบัติการและการเขียนโปรแกรม 1.4 วิัฒนาการของภาษาคอมพิวเตอร์ 1.5 ความสำคัญของภาษา Python	2	3
2	การเขียนโปรแกรมเบื้องต้นด้วยภาษา Python 2.1 โปรแกรมสำหรับพัฒนาโปรแกรม (IDE) 2.2 การเริ่มต้นเขียนโปรแกรม 2.3 ตัวแปรและชนิดข้อมูล 2.4 การโปรแกรมการคำนวณทางคณิตศาสตร์ 2.5 ลิสต์ 2.6 ติกขัน Narie 2.7 ความผิดพลาดของโปรแกรม 2.8 การเขียนโปรแกรมในกระบวนการคำนวณ	4	6
3	การควบคุมโปรแกรม 3.1 การเขียนผังงาน (Flowchart) 3.2 การเขียนโปรแกรมแบบเงื่อนไข 3.3 การเขียนโปรแกรมแบบวนซ้ำ 3.4 การเขียนโปรแกรมเพิ่มเติมการสำหรับการควบคุมโปรแกรม	4	6
4	การสร้างฟังก์ชัน 4.1 ความหมายและบทบาทของฟังก์ชัน 4.2 การใช้งานฟังก์ชันเบื้องต้น ได้แก่ การสร้างและเรียกใช้ฟังก์ชัน (Function Creation and Call) 4.3 การส่งค่าและรับค่าผลลัพธ์ (Parameters and Return Values) 4.4 การใช้ฟังก์ชันแก้ไขปัญหา 4.5 การสร้างโมดูล (Create Module)	2	3
5	การเขียนโปรแกรมเชิงโครงสร้าง 5.1 การระบุปัญหาและแนวคิดการเขียนโปรแกรมเชิงโครงสร้าง		

	5.2 การออกแบบโปรแกรมด้วยแนวคิด การออกแบบจากบันลงล่าง (Top-Down Design) 5.3 การออกแบบโปรแกรมแบบจากบันลงล่าง 5.4 การพัฒนาโปรแกรมเพื่อแก้ปัญหา 5.5 ตัวอย่างและการประยุกต์ใช้งาน	6	9
6	การจัดการข้อมูลและการอ่านไฟล์ 6.1 รูปแบบโครงสร้างข้อมูล (list, dictionary, table) 6.2 การอ่านและเขียนไฟล์ใน Python 6.3 การประมวลผลข้อมูลจากไฟล์ 6.4 การติดตั้งและใช้งานไลบรารีใน Python 6.5 การใช้งานไลบรารีที่ได้รับความนิยม (NumPy, Pandas, Matplotlib)	6	9
7	การประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญหางานด้านวิศวกรรม 7.1 เขียนโปรแกรมเพื่อคำนวณด้านวิศวกรรมและแสดงผลกราฟ 7.2 เขียนโปรแกรมเพื่อคำนวณหาเส้นทางที่สั้นที่สุด (TSP) 7.3 เขียนโปรแกรมระบบอินเตอร์เน็ตของสรรพสิ่ง (IoT) 7.4 การเขียนโปรแกรมระบบการเรียนรู้ของเครื่อง (Machine Learning) 7.5 งานวิจัยเพิ่มเติม	6	9
	รวม	30	45
	สอบกลางภาคและปลายภาค	6	
	รวมกั้งสิ้น	36	45

จุดประสงค์การสอน

บทที่	รายการ	เวลาเรียน
1	ความรู้พื้นฐานเกี่ยวกับระบบคอมพิวเตอร์และการเขียนโปรแกรมคอมพิวเตอร์ 1. บอกความสำคัญของการเขียนโปรแกรมคอมพิวเตอร์ 2. บอกองค์ประกอบของระบบคอมพิวเตอร์ 3. อธิบายหลักการทำงานของระบบคอมพิวเตอร์ 4. อธิบายหลักการระบบปฏิบัติการและการเขียนโปรแกรม 5. ลำดับวิัฒนาการของภาษาคอมพิวเตอร์ 6. บอกความสำคัญของภาษา Python	5 คาบ 300 นาที
2	การเขียนโปรแกรมเบื้องต้นด้วยภาษา Python 1. ติดตั้งโปรแกรมสำหรับพัฒนาโปรแกรมได้ (IDE) 2. เขียนโปรแกรมให้แสดงผลข้อความเบื้องต้นได้ 3. บอกข้อแตกต่างของชนิดข้อมูลต่างๆ ในภาษา Pythonได้ 4. เขียนโปรแกรมการคำนวณทางคณิตศาสตร์ได้ 5. เข้าใจและเขียนโปรแกรมใช้งานชนิดข้อมูลแบบลิสต์ได้ 6. เข้าใจและเขียนโปรแกรมใช้งานชนิดติกซันนารีได้ 7. บอกประเภทของความผิดพลาดของโปรแกรมได้ 8. เขียนโปรแกรมควบกระบวนการคำนวน ได้แก่ รับอินพุต ประมวลผลและแสดงผลเอาต์พุต	15 คาบ 900 นาที
3	การควบคุมโปรแกรม 1. บอกองค์ประกอบในผังงาน (Flowchart) 2. สามารถเขียนโปรแกรมตามผังงานได้ 3. เข้าใจการควบคุมการไหลของโปรแกรม (Flow Control) เปรียบเทียบ Sequence กับ Decision และใช้งาน Boolean (and, or, not) ได้ 4. เขียนโปรแกรมด้วย if if-else if-elif และจัดการ Nested Conditions ได้อย่างเหมาะสม 5. เลือกใช้งานลูป while และ for พร้อมเปรียบเทียบความแตกต่างเพื่อแก้ปัญหา 6. ใช้คำสั่ง break และ continue ในการควบคุมการทำงานของลูปได้อย่างมีประสิทธิภาพ 7. สร้างโปรแกรมที่ใช้ลูปทำงาน (เช่น ตารางสูตรคูณ หรือ เกมไทยตัวเลข) ออกแบบโดยให้ทำงานช้าโดยไม่ช้าช้อน	15 คาบ 900 นาที
4	การสร้างฟังก์ชัน	5 คาบ

	<ol style="list-style-type: none"> 1. เข้าใจความหมายของฟังก์ชัน (Function) และบทบาทในการพัฒนาโปรแกรม 2. เขียนโปรแกรมสร้างและเรียกใช้ฟังก์ชัน 3. เขียนโปรแกรมส่งค่า (Parameters) และรับค่าผลลัพธ์ (Return Values) จากฟังก์ชัน 4. เขียนฟังก์ชันแก้ไขปัญหาได้อย่างมีประสิทธิภาพ 5. เขียนโปรแกรมเพื่อสร้างและใช้งานแบบโมดูลได้ 	300 นาที
5	<p>การเขียนโปรแกรมเชิงโครงสร้าง</p> <ol style="list-style-type: none"> 1. เข้าใจแนวคิดของการเขียนโปรแกรมเชิงโครงสร้าง รวมถึงองค์ประกอบสำคัญ ได้แก่ ลำดับการทำงาน เนื่องจากการตัดสินใจ และการวนซ้ำ 2. ออกแบบโปรแกรมด้วยแนวคิด Top-Down Design โดยสามารถแบ่งปัญหา ออกเป็นส่วนย่อยและจัดลำดับขั้นตอนการแก้ปัญหาได้ 3. เขียนโปรแกรมที่สามารถรับข้อมูลจากผู้ใช้ ประมวลผลข้อมูล และแสดงผลลัพธ์ที่ ถูกต้องตามที่กำหนด 4. ใช้ผังงาน (Flowchart) หรือ Pseudocode ในการวางแผนและพัฒนาโปรแกรมที่มี โครงสร้างชัดเจน 5. ทดสอบและแก้ไขข้อผิดพลาดในโปรแกรม เพื่อปรับปรุงคุณภาพและประสิทธิภาพของ โปรแกรมอย่างมีระบบ 	15 คาบ 900 นาที
6	<p>การจัดการข้อมูลและการอ่านไฟล์</p> <ol style="list-style-type: none"> 1. รูปแบบโครงสร้างข้อมูล ลิสต์ ติกซันนารีและการสร้างตาราง 2. เขียนโปรแกรมอ่านและเขียนไฟล์ใน Python ได้ 3. เขียนโปรแกรมเพื่อประมวลผลข้อมูลจากไฟล์ได้ 4. ติดตั้งและเขียนโปรแกรมเพื่อใช้งานไลบรารีได้ 5. เขียนโปรแกรมเพื่อใช้งานไลบรารีที่เป็นที่นิยม ได้แก่ NumPy Pandas Matplotlib ได้ 	10 คาบ 600 นาที
7	<p>การประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญหางานด้านวิศวกรรม</p> <ol style="list-style-type: none"> 1. เขียนโปรแกรมเพื่อคำนวณทางด้านวิศวกรรมและแสดงผลกราฟได้ 2. อธิบายหลักการของการหาเส้นทางที่สั้นที่สุด (TSP) ได้ 3. เขียนโปรแกรมเพื่อคำนวณหาเส้นทางที่สั้นที่สุดได้ 4. บอกองค์ประกอบของวงจรอิเล็กทรอนิกส์สำหรับงาน ระบบอินเตอร์เน็ตของ สรรพสิ่ง 5. เขียนโปรแกรมระบบอินเตอร์เน็ตของสรรพสิ่งได้ 6. บอกการประยุกต์ใช้การเรียนรู้ของเครื่องทางด้านการทำนายโดยอัตโนมัติและการ จำแนกภาพได้ 7. อธิบายหลักการงานวิจัยเพิ่มเติมได้ 	10 คาบ 600 นาที

กำหนดการสอน

วิชาการโปรแกรมคอมพิวเตอร์ รหัสวิชา EN-001-002 ตามหลักสูตรปริญญาตรี วิศวกรรมศาสตรบัณฑิต สาขาวิชาวาระมโลจิสติกส์ สาขาวิชาวาระมอุตสาหการ สาขาวิเครื่องกลและสาขาวิคอมพิวเตอร์ กำหนดให้เรียนทั้งสิ้น 15 ทดสอบกลางภาคและปลายภาค 2 สัปดาห์ รวม 17 สัปดาห์ 81 คาบเรียน

สัปดาห์ที่	LLOs	คาบที่	รายการสอน
1	LLO1-1 LLO1-2 LLO1-3	1-5	1. ความรู้พื้นฐานเกี่ยวกับระบบคอมพิวเตอร์และการเขียนโปรแกรมคอมพิวเตอร์ <ul style="list-style-type: none"> 1.1 ความสำคัญของการเขียนโปรแกรมคอมพิวเตอร์ 1.2 องค์ประกอบของระบบคอมพิวเตอร์ 1.3 ระบบปฏิบัติการและการเขียนโปรแกรม 1.4 วิัฒนาการของภาษาคอมพิวเตอร์ 1.5 ความสำคัญของภาษา Python
			2. การเขียนโปรแกรมเบื้องต้นด้วยภาษา Python <ul style="list-style-type: none"> 2.1 การติดตั้งและใช้งานโปรแกรม (IDE) 2.2 การเริ่มต้นเขียนโปรแกรม 2.3 ตัวแปรและชนิดข้อมูล 2.4 การคำนวณทางคณิตศาสตร์
			2. การเขียนโปรแกรมเบื้องต้นด้วยภาษา Python (ต่อ) <ul style="list-style-type: none"> 2.5 ลิสต์ 2.6 ดิกชันนารี 2.7 ความผิดพลาดของโปรแกรม 2.8 การเขียนโปรแกรมในกระบวนการคำนวณ
4	LLO2-3 LLO3-1 LLO3-2	16-20	3. การควบคุมโปรแกรม <ul style="list-style-type: none"> 3.1 การเขียนผังงาน (Flowchart) 3.2 การเขียนโปรแกรมแบบเงื่อนไข (if, if-else, if-elif) 3.3 การเขียนโปรแกรมแบบวนซ้ำ (while, for)
			3. การควบคุมโปรแกรม (ต่อ) <ul style="list-style-type: none"> 3.4 การเขียนโปรแกรมเพิ่มเติมการสำหรับการควบคุมโปรแกรม
			4. การสร้างฟังก์ชัน <ul style="list-style-type: none"> 4.1 ความหมายและบทบาทของฟังก์ชัน 4.2 การใช้งานฟังก์ชันเบื้องต้น ได้แก่ การสร้างและเรียกใช้ฟังก์ชัน (Function Creation and Call)

สัปดาห์ที่	LLOs	ค่าบกี่	รายการสอน
			4.3 การส่งค่าและรับค่าผลลัพธ์ 4.4 การใช้ฟังก์ชันแก้ไขปัญหา 4.5 การสร้างโมดูล (Create Module)
7	LLO3-1 LLO3-2	31-35	5. การเขียนโปรแกรมเชิงโครงสร้าง 5.1 การระบุปัญหาและแนวคิดการเขียนโปรแกรมเชิงโครงสร้าง 5.2 การออกแบบแบบโปรแกรมด้วยแนวคิด Top-Down Design
8		36-38	สอบกลางภาค
9	LLO2-3 LLO3-3	39-43	5. การเขียนโปรแกรมเชิงโครงสร้าง (ต่อ) 5.3 การพัฒนาโปรแกรมเพื่อแก้ปัญหา
10	LLO3-4	44-48	5. การเขียนโปรแกรมเชิงโครงสร้าง(ต่อ) 5.4 การพัฒนาโปรแกรมเพื่อแก้ปัญหา 5.5 ตัวอย่างและการประยุกต์ใช้งาน
11	LLO4-1 LLO4-2	49-53	6. การจัดการข้อมูลและการอ่านไฟล์ 6.1 รูปแบบโครงสร้างข้อมูล (list, dictionary, table) 6.2 การอ่านและเขียนไฟล์ใน Python
12	LLO4-3	54-58	6. การจัดการข้อมูลและการอ่านไฟล์ (ต่อ) 6.3 การประมวลผลข้อมูลจากไฟล์ 6.4 การติดตั้งและใช้งานไลบรารีใน Python
13	LLO4-4 LLO5-1	59-63	6. การจัดการข้อมูลและการอ่านไฟล์ (ต่อ) 6.5 การใช้งานไลบรารีที่ได้รับความนิยม ได้แก่ Numpy Pandas Matplotlib
14	LLO5-1 LLO5-2	64-68	7. การประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญางานด้านวิศวกรรม 7.1 การคำนวณด้านวิศวกรรมและแสดงผลกราฟ 7.2 เขียนโปรแกรมเพื่อคำนวณหาเส้นทางที่สั้นที่สุด (TSP)
15	LLO5-2 LLO5-3	69-73	7. การประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญางานด้านวิศวกรรม (ต่อ) 7.3 เขียนโปรแกรมระบบอินเตอร์เน็ตของสรรพสิ่ง (IoT) 7.4 การเขียนโปรแกรมระบบการเรียนรู้ของเครื่อง (Machine Learning)
16	LLO5-3	74-78	7. การประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญางานด้านวิศวกรรม (ต่อ) 7.4 การเขียนโปรแกรมระบบการเรียนรู้ของเครื่อง (Machine Learning) 7.5 งานวิจัยเพิ่มเติม
17		78-81	สอบปลายภาค

กลยุทธ์การสอนและการประเมินผลลัพธ์การเรียนรู้รายวิชา

1. กลยุทธ์การสอนและการประเมินผลลัพธ์การเรียนรู้รายวิชา

CLOs	น้ำหนัก CLOs* (ร้อยละ)	พฤติกรรมบ่งชี้ (Performance Criteria)	การประเมินผล		กลยุทธ์การสอน
			งาน / กิจกรรม (ที่ใช้ประเมิน)	วิธีการ/ เครื่องมือประเมินผล	
CLO1	10	<ul style="list-style-type: none"> - อธิบายส่วนประกอบหลักของระบบคอมพิวเตอร์ - อธิบายหลักการทำงานของระบบปฏิบัติการและวิัฒนาการของภาษาคอมพิวเตอร์ 	<ul style="list-style-type: none"> - แบบฝึกหัด - การสอบกลางภาค 	<ul style="list-style-type: none"> - ตรวจข้อสอบตามเกณฑ์ - การประเมินจากค่าตอบแบบอัตโนมัติ 	<ul style="list-style-type: none"> - การบรรยาย - การอภิปรายในชั้นเรียนพร้อมตัวอย่างประกอบ
CLO2	20	<ul style="list-style-type: none"> - อธิบายอธิบายแนวคิดพื้นฐานของการเขียนโปรแกรมเบื้องต้น - เขียนโปรแกรมเบื้องต้น 	<ul style="list-style-type: none"> - แบบฝึกหัด - แบบฝึกปฏิบัติ - การสอบกลางภาค - การสอบกลางภาค 	<ul style="list-style-type: none"> - การตรวจข้อสอบตามเกณฑ์ - การประเมินโดยการโปรแกรมและการทำงาน 	<ul style="list-style-type: none"> - การบรรยาย - การฝึกปฏิบัติในห้องปฏิบัติการ
CLO3	15	<ul style="list-style-type: none"> - ออกแบบบล็อกอրิทึมและผังงาน (Flowchart) - เขียนโปรแกรมที่แก้ไขปัญหาตามข้อกำหนด 	<ul style="list-style-type: none"> - แบบฝึกหัด - แบบฝึกปฏิบัติ - การสอบกลางภาค - การสอบกลางภาค 	<ul style="list-style-type: none"> - การตรวจข้อสอบตามเกณฑ์ - การประเมินโดยการโปรแกรมและการทำงาน 	<ul style="list-style-type: none"> - การบรรยาย - การฝึกปฏิบัติในห้องปฏิบัติการ - การอภิปรายกรณีศึกษา
CLO4	25	<ul style="list-style-type: none"> - ใช้โครงสร้างข้อมูล - เขียนโปรแกรมเพื่ออ่านและประมวลผลข้อมูลจากไฟล์ - ใช้เลบรารี 	<ul style="list-style-type: none"> - แบบฝึกหัด - แบบฝึกปฏิบัติ - การสอบกลางภาค - การสอบกลางภาค 	<ul style="list-style-type: none"> - การตรวจข้อสอบตามเกณฑ์ - การประเมินโดยการโปรแกรมและการทำงาน 	<ul style="list-style-type: none"> - การบรรยาย - การอภิปรายในชั้นเรียนพร้อมตัวอย่างประกอบ
CLO5	30	- ออกแบบและพัฒนาโปรแกรมเฉพาะทางวิศวกรรม	<ul style="list-style-type: none"> - โครงการวิศวกรรม - การนำเสนอผลงาน 	<ul style="list-style-type: none"> - การประเมินโครงการตามเกณฑ์ (Rubric) - การวิเคราะห์การนำเสนอ - ตรวจโค้ดโปรแกรม 	<ul style="list-style-type: none"> - การบรรยาย - การให้คำปรึกษารายบุคคล

2. แผนการประเมินผลลัพธ์การเรียนรู้รายวิชา

ลำดับ	งาน/กิจกรรม/วิธีการ (ที่ใช้ประเมิน)	น้ำหนักงาน (ร้อยละ)	น้ำหนักคงคะแนนของแต่ละงานที่รองรับแต่ละ CLOs				
			CLO1	CLO2	CLO3	CLO4	CLO5
1	การสอดคล้องภาค และปลายภาค	40	10	10	10	10	
2	การประเมินใน ห้องปฏิบัติการ	15		5	5	5	
3	การเขียนโปรแกรม การบ้าน	15		5		10	
4	การทำงานกลุ่ม	20					20
5	การนำเสนอผลงาน แล่่มรายงาน	10					10
	รวมน้ำหนักการประเมิน	100	10	20	15	25	30

3. วิธีการให้คะแนน

ดำเนินการรวบรวมข้อมูลเพื่อการประเมินผลจากคะแนนเต็มในรายวิชาทั้งหมด 100 คะแนน โดยแบ่งแยกคะแนนทั้งหมดออกเป็น 3 ส่วนคือ

1. ผลงานที่มีอยู่ 30 คะแนน หรือ 30%
2. การพิจารณาจากจิตพิสัย ความตั้งใจและการร่วมกิจกรรม 10 คะแนนหรือ 10%
3. การทดสอบแต่ละหน่วยเรียน 60 คะแนน หรือ 60% โดยจะจัดแบ่งน้ำหนัก
คะแนนในแต่ละหน่วยตามตารางกำหนดน้ำหนักคงคะแนนในหน้าถัดไป

4. เกณฑ์ค่าระดับคะแนน

1. พิจารณาเกณฑ์ผ่านรายวิชาตามค่าระดับคงคะแนนตามเกณฑ์ดังนี้

คงคะแนนร้อยละ 80 ขึ้นไป	ได้ A
คงคะแนนร้อยละ 75 - 79	ได้ B+
คงคะแนนร้อยละ 70 - 74	ได้ B
คงคะแนนร้อยละ 65 - 69	ได้ C+
คงคะแนนร้อยละ 60 - 64	ได้ C
คงคะแนนร้อยละ 55 - 59	ได้ D+
คงคะแนนร้อยละ 50 - 54	ได้ D
คงคะแนนร้อยละ 0 - 49	ได้ F

5. ตารางกำหนดน้ำหนักคะแนน

เลขที่บก	คะแนนรายหน่วย และน้ำหนักคะแนน ชื่อบก	คะแนนรายหน่วย	คะแนนผ่านรายหน่วย (%)	น้ำหนักคะแนน				
				พุทธิพิสัย				ทักษะพัฒนา
				ความรู้ความจำ	ความเข้าใจ	การนำไปใช้	สูงกว่า	
1	ความรู้พื้นฐานเกี่ยวกับระบบคอมพิวเตอร์และการเขียนโปรแกรมคอมพิวเตอร์	5	50%	3	2	-	-	-
2	การเขียนโปรแกรมเบื้องต้น Python	10	50%	2	2	2	-	4
3	การควบคุมโปรแกรม	10	50%	2	2	2	-	4
4	การสร้างฟังก์ชัน	10	50%	2	2	2	-	4
5	การเขียนโปรแกรมเชิงโครงสร้าง	10	50%	2	2	2	-	4
6	การจัดการข้อมูลและการอ่านไฟล์	5	50%	1	1	1	-	2
7	การประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญหางานด้านวิศวกรรม	10	50%	1	3	3	-	3
ก	คะแนนภาควิชาการ	60	50%	13	14	12	-	21
ข	คะแนนภาคผลงาน	30	50%					
ค	คะแนนภาคจิตพิสัย	10	80%					
	รวมทั้งสิ้น	100	50%					

สารบัญ

บทที่ 1 ความรู้พื้นฐานเกี่ยวกับระบบคอมพิวเตอร์และการเขียนโปรแกรมคอมพิวเตอร์.....	1
1.1 ความสำคัญของการเขียนโปรแกรมคอมพิวเตอร์	2
1.2 องค์ประกอบและหลักการทำงานของระบบคอมพิวเตอร์	4
1.3 ระบบปฏิบัติการและการเขียนโปรแกรม	9
1.4 วิัฒนาการของภาษาคอมพิวเตอร์	10
1.5 ความสำคัญของภาษา Python	14
บทสรุป.....	17
คำถานท้ายบท	19
บทที่ 2 การเขียนโปรแกรมเบื้องต้นด้วยภาษา Python.....	20
2.1 โปรแกรมสำหรับพัฒนาโปรแกรม (IDE)	21
2.2 การเริ่มต้นเขียนโปรแกรม	29
2.3 ตัวแปรและชนิดข้อมูล.....	35
2.4 การโปรแกรมการคำนวณทางคณิตศาสตร์	41
2.5 ลิสต์ (List)	47
2.6 ดิกชันนารี (Dictionary).....	55
2.7 ความผิดพลาดของโปรแกรม	58
2.8 การเขียนโปรแกรมในกระบวนการคำนวณ.....	61
บทสรุป.....	67
คำถานท้ายบท	68
โจทย์การเขียนโปรแกรม	69
บทที่ 3 การควบคุมโปรแกรม	71
3.1 การเขียนผังงาน (Flowchart).....	73
3.2 การเขียนโปรแกรมแบบเงื่อนไข	76
3.3 การเขียนโปรแกรมแบบวนซ้ำ (Loops).....	88
3.4 การเขียนโปรแกรมเพิ่มเติมการสำหรับการควบคุมโปรแกรม.....	99
บทสรุป.....	109
คำถานท้ายบท	110
โจทย์การเขียนโปรแกรม	111
บทที่ 4 พังก์ชัน	114
4.1 ความหมายและบทบาทของฟังก์ชัน	115
4.2 การใช้งานฟังก์ชันเบื้องต้น	117

สารบัญ (ต่อ)

4.3 การส่งค่าและรับค่าผลลัพธ์.....	122
4.4 การใช้ฟังก์ชันแก้ไขปัญหา	126
4.5 การสร้างโมดูล.....	129
บทสรุป	131
คำถามท้ายบท	133
โจทย์การเขียนโปรแกรม	134
บทที่ 5 การเขียนโปรแกรมเชิงโครงสร้าง	138
5.1 แนวคิดการเขียนโปรแกรมเชิงโครงสร้าง	139
5.2 การระบุปัญหา	142
5.3 การออกแบบแบบโปรแกรมแบบจากบนลงล่าง.....	145
5.4 การพัฒนาโปรแกรมเพื่อแก้ปัญหา.....	151
5.5 ตัวอย่างและการประยุกต์ใช้งาน	155
บทสรุป	179
คำถามท้ายบท	180
โจทย์การเขียนโปรแกรม	181
บทที่ 6 การจัดการข้อมูลและการอ่านไฟล์.....	184
6.1 รูปแบบโครงสร้างข้อมูล	185
6.2 การอ่านและเขียนไฟล์.....	193
6.3 การประมวลผลข้อมูลจากไฟล์ด้วย Pandas.....	201
6.4 การติดตั้งและใช้งานไลบรารีใน Python	204
6.5 การใช้งานไลบรารีที่ได้รับความนิยม	206
บทสรุป	220
คำถามท้ายบท	221
โจทย์การเขียนโปรแกรม	222
บทที่ 7 การประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญหางานด้านวิศวกรรม.....	227
7.1 การเขียนโปรแกรมเพื่อคำนวณด้านวิศวกรรมและแสดงผลกราฟ	228
7.2 การเขียนโปรแกรมเพื่อคำนวณหาเส้นทางที่สั้นที่สุด (TSP)	2466
7.3 การเขียนโปรแกรมระบบอินเตอร์เน็ตของสรรพสิ่ง (IoT)	252
7.4 การเขียนโปรแกรมระบบการเรียนรู้ของเครื่อง (Machine Learning).....	24666
7.5 งานวิจัยเพิ่มเติม	252

บทสรุป.....	275
คำถ้ามท้ายบท	276
โจทย์การเขียนโปรแกรม.....	277
บรรณานุกรม.....	280

บทที่ 1

ความรู้พื้นฐานเกี่ยวกับระบบคอมพิวเตอร์และ การเขียนโปรแกรมคอมพิวเตอร์

เนื้อหาบทเรียน

- ความสำคัญของการเขียนโปรแกรมคอมพิวเตอร์
- องค์ประกอบและหลักการทำงานของระบบคอมพิวเตอร์
- ระบบปฏิบัติการและการเขียนโปรแกรม
- วิัฒนาการของภาษาคอมพิวเตอร์
- ความสำคัญของภาษา Python

วัตถุประสงค์การเรียนรู้

- บอกความสำคัญของการเขียนโปรแกรมคอมพิวเตอร์
- บอกองค์ประกอบของระบบคอมพิวเตอร์
- อธิบายหลักการทำงานของระบบคอมพิวเตอร์
- อธิบายหลักการระบบปฏิบัติการและการเขียนโปรแกรม
- ลำดับวิัฒนาการของภาษาคอมพิวเตอร์
- บอกความสำคัญของภาษา Python

ในยุคที่เทคโนโลยีมีบทบาทสำคัญในทุกด้านของชีวิต การเขียนโปรแกรมคอมพิวเตอร์ได้กลายเป็นหักษะที่สำคัญและจำเป็น การเรียนรู้เกี่ยวกับการเขียนโปรแกรมไม่เพียงแต่ช่วยให้สามารถพัฒนาเทคโนโลยีใหม่ๆ แต่ยังเปิดโอกาสทางอาชีพและพัฒนาแนวคิดในการแก้ปัญหาอย่างเป็นระบบ ในบทนี้จะได้เรียนรู้เกี่ยวกับความสำคัญของการเขียนโปรแกรม องค์ประกอบของระบบคอมพิวเตอร์ และวิัฒนาการของภาษาคอมพิวเตอร์ ซึ่งจะเป็นพื้นฐานสำคัญในการพัฒนาหักษะทางเทคโนโลยี

1.1 ความสำคัญของการเขียนโปรแกรมคอมพิวเตอร์

การเขียนโปรแกรมเป็นพื้นฐานสำคัญในการพัฒนาเทคโนโลยีและนวัตกรรมใหม่ๆ ที่กำลังเปลี่ยนแปลงโลกในปัจจุบัน เช่น ปัญญาประดิษฐ์ (AI) ที่สามารถวิเคราะห์ข้อมูลและตัดสินใจในระดับที่ซับซ้อนได้ อินเตอร์เน็ตของสรรพสิ่ง (IoT) ที่เชื่อมต่ออุปกรณ์ต่างๆ เข้าด้วยกันเพื่อการทำงานอัตโนมัติและการเก็บข้อมูลแบบเรียลไทม์ และบล็อกเชน (Blockchain) ที่เพิ่มความปลอดภัยและความโปร่งใสในการทำธุรกรรมดิจิทัล โปรแกรมคอมพิวเตอร์ไม่เพียงแต่ใช้ในการพัฒนาเทคโนโลยีเหล่านี้ แต่ยังสามารถนำมาใช้ในการสร้างผลิตภัณฑ์และบริการใหม่ๆ ที่ตอบสนองความต้องการของผู้บริโภคและองค์กรธุรกิจ เช่น แอปพลิเคชันบนสมาร์ทโฟน แพลตฟอร์มการค้าออนไลน์ ระบบการจัดการข้อมูล และเครื่องมือวิเคราะห์ข้อมูลขั้นสูง การเขียนโปรแกรมจึงเป็นหักษะที่ขาดไม่ได้สำหรับการเปลี่ยนแปลงและนวัตกรรมในทุกด้านของชีวิตและการทำงาน

การเขียนโปรแกรมช่วยพัฒนาหักษะการแก้ปัญหาอย่างเป็นระบบและมีเหตุผล โดยการสร้างอัลกอริทึมเพื่อแก้ไขปัญหาที่ซับซ้อนและหลากหลาย การเขียนโปรแกรมทำให้ผู้เรียนสามารถพัฒนาแนวคิดและวิธีการในการแก้ปัญหาได้อย่างเป็นขั้นเป็นตอน โปรแกรมคอมพิวเตอร์ยังสามารถใช้ในการวิเคราะห์ข้อมูลและช่วยในการตัดสินใจอย่างมีประสิทธิภาพ เช่น การพยากรณ์ยอดขาย การวิเคราะห์การตลาด และการคาดการณ์แนวโน้มทางธุรกิจ นอกจากนี้ การเขียนโปรแกรมยังมีบทบาทสำคัญในการแก้ปัญหาทางด้านคณิตศาสตร์ วิทยาศาสตร์ และวิศวกรรม โดยสามารถสร้างเครื่องมือคำนวณที่มีความแม่นยำสูงและทำงานได้อย่างรวดเร็ว เช่น การแก้สมการทางคณิตศาสตร์ การจำลองการทดลองทางวิทยาศาสตร์ และการออกแบบระบบวิศวกรรม โปรแกรมคอมพิวเตอร์สามารถประมวลผลข้อมูลจำนวนมากและซับซ้อนได้ ทำให้นักวิจัยและวิศวกรสามารถทดสอบและปรับปรุงโมเดลทางวิทยาศาสตร์และวิศวกรรมได้อย่างมีประสิทธิภาพ ด้วยเหตุนี้การเขียนโปรแกรมจึงเป็นหักษะที่ขาดไม่ได้สำหรับการแก้ปัญหาและการตัดสินใจในหลากหลายสาขาวิชา

ความสามารถในการเขียนโปรแกรมเปิดโอกาสทางอาชีพมากมายในสาขาต่างๆ ซึ่งครอบคลุมทั้งด้านเทคโนโลยีและนอกเหนือจากเทคโนโลยี เช่น นักพัฒนาโปรแกรมที่ทำงานในองค์กรต่างๆ เพื่อสร้างและปรับปรุงซอฟต์แวร์ นักวิเคราะห์ข้อมูลที่ใช้หักษะการเขียนโปรแกรมในการจัดการและวิเคราะห์ข้อมูลเพื่อให้ข้อมูลเชิงลึกสำหรับการตัดสินใจเชิงธุรกิจ นักวิทยาศาสตร์คอมพิวเตอร์ที่ทำงานวิจัยและพัฒนาเทคโนโลยีใหม่ๆ ในการประมวลผลข้อมูล รวมถึงวิศวกรซอฟต์แวร์ที่ออกแบบและพัฒนาโครงสร้างระบบซอฟต์แวร์ที่ซับซ้อน การเขียนโปรแกรมเป็นหักษะที่มีความต้องการสูงในตลาดแรงงานปัจจุบัน ทำให้ผู้ที่มีหักษะนี้มีโอกาสในการทำงานมากขึ้น และมักมีรายได้ที่สูงกว่าค่าเฉลี่ย การเขียนโปรแกรมยังเป็นพื้นฐานสำคัญสำหรับการทำงานในอุตสาหกรรมที่กำลังเติบโตอย่างรวดเร็ว เช่น ปัญญาประดิษฐ์ (Artificial Intelligence: AI) ฐานข้อมูลขนาด

ใหญ่ (Big Data) การวิเคราะห์ข้อมูล (Data Analytics) การพัฒนาแอปพลิเคชันมือถือ และเทคโนโลยีบล็อกเชน (Blockchain)

นอกจากนี้ ความสามารถในการเขียนโปรแกรมยังเปิดโอกาสในการทำงานแบบอาชีพอิสระ หรือการสร้างสรรค์ธุรกิจส่วนตัว เช่น การพัฒนาแอปพลิเคชัน การสร้างเว็บไซต์ และการให้คำปรึกษาด้านไอที ทำให้สามารถควบคุมเวลาและสถานที่ทำงานได้อย่างยืดหยุ่น ความต้องการทักษะการเขียนโปรแกรมยังมีแนวโน้มเพิ่มขึ้นในอนาคต ทำให้การเรียนรู้และพัฒนาทักษะนี้เป็นการลงทุนที่มีคุณค่าในระยะยาวในยุคปัจจุบัน การพัฒนาซอฟต์แวร์ได้รับผลกระทบอย่างมาก



รูปที่ 1.1 แสดงเทคโนโลยีใหม่ๆ ที่เกิดจากการพัฒนาซอฟต์แวร์

แหล่งที่มา: <https://www.desuvit.com/innovative-technologies-that-have-taken-over-software-development/>

จากรูปที่ 1.1 แสดงภาพแนวคิดของเทคโนโลยีใหม่ๆ ที่เกิดขึ้นจากการพัฒนาซอฟต์แวร์ เทคโนโลยีเหล่านี้ไม่เพียงแต่เพิ่มประสิทธิภาพในการพัฒนาและการทำงานของซอฟต์แวร์เท่านั้น แต่ยังช่วยสร้างนวัตกรรมที่ตอบสนองความต้องการที่เปลี่ยนแปลงไปของผู้บริโภคและองค์กรธุรกิจอย่างรวดเร็ว นี่คือสรุปเทคโนโลยีใหม่ๆ ที่มีผลกระทบอย่างมากต่อการพัฒนาซอฟต์แวร์ ได้แก่

1. เทคโนโลยี 5G เพิ่มความเร็วอินเทอร์เน็ตและการเชื่อมต่อ ทำให้แอปพลิเคชันขึ้นสูงใน VR, เกม และการค้าปลีกดิจิทัลทำงานได้ดีขึ้น
2. อินเตอร์เน็ตของสรรพสิ่ง (Internet of Things: IoT) การเชื่อมต่ออุปกรณ์เพื่อแลกเปลี่ยนข้อมูลแบบเรียลไทม์
3. พฤติกรรมผู้บริโภค (Internet of Behaviors : IoB) วิเคราะห์ข้อมูลเพื่อเข้าใจและทำนายพฤติกรรมผู้บริโภค
4. AR/VR สร้างประสบการณ์ผู้ใช้ที่สมจริง
5. คลาวด์และเซิร์ฟเวอร์ (Everything-as-a-Service : XaaS) ให้บริการผ่านอินเทอร์เน็ต เพิ่มความยืดหยุ่นและลดต้นทุน

6. การวิเคราะห์ข้อมูลขนาดใหญ่ (Big Data Analytics) เพื่อการตัดสินใจที่ดีขึ้น

7. นวัตกรรมการป้องกันทางไซเบอร์ (Cybersecurity Innovations) รวมถึงบล็อกเชนและ VDN เพื่อเพิ่มความปลอดภัยของข้อมูล

8. บล็อกเชน (Blockchain) จัดเก็บข้อมูลแบบกระจาย ทำให้ปลอดภัยและโปร่งใส

9. ปัญญาประดิษฐ์ (AI) อัตโนมัติการทำงาน เพิ่มประสิทธิภาพ และลดข้อผิดพลาด และเทคโนโลยีอื่นๆ

จากที่กล่าวมา การเขียนโปรแกรมคอมพิวเตอร์เป็นทักษะสำคัญในการพัฒนาเทคโนโลยีและนวัตกรรมใหม่ๆ เช่น AI, IoT และ Blockchain ซึ่งเปลี่ยนแปลงโลกและธุรกิจ โปรแกรมคอมพิวเตอร์ช่วยแก้ปัญหาซับซ้อนและการตัดสินใจโดยใช้การวิเคราะห์ข้อมูล นอกจากนี้ ทักษะการเขียนโปรแกรมยังเปิดโอกาสทางอาชีพมากมายและมีความต้องการสูงในตลาดแรงงานปัจจุบัน ทำให้มีโอกาสในการทำงานและรายได้สูง รวมถึงการทำงานแบบอิสระและการสร้างธุรกิจส่วนตัวได้

ในยุคดิจิทัลปัจจุบัน การเขียนโปรแกรมคอมพิวเตอร์ได้กลายเป็นทักษะที่สำคัญอย่างยิ่ง ไม่จำกัดเฉพาะผู้ที่ทำงานในสายเทคโนโลยีสารสนเทศเท่านั้น แต่ยังรวมถึงผู้ที่ปฏิบัติงานในสาขาอื่นๆ เช่น ธุรกิจการเงิน และการตลาด เนื่องจากทักษะนี้สามารถนำไปประยุกต์ใช้ในการแก้ไขปัญหาที่ซับซ้อนได้หลากหลายรูปแบบ เช่น การวิเคราะห์ข้อมูลขนาดใหญ่ การพัฒนาแอปพลิเคชัน และการพัฒนาระบบอัตโนมัติสำหรับการปฏิบัติงานประจำวัน การมีทักษะการเขียนโปรแกรมจะช่วยเพิ่มศักยภาพในการปฏิบัติงานและเพิ่มโอกาสในการแข่งขันในตลาดแรงงานที่มีการเปลี่ยนแปลงอย่างรวดเร็ว

1.2 องค์ประกอบและหลักการทำงานของระบบคอมพิวเตอร์

ปัจจุบันคอมพิวเตอร์ได้เข้ามายืดหยุ่นมากต่อการดำเนินชีวิต กลายเป็นศาสตร์ที่ต้องเรียนรู้เพื่อให้สามารถทำงานต่างๆ ได้shedev รวดเร็ว ประหยัดเวลามากกว่าเดิม กล่าวคือ คอมพิวเตอร์ เป็น อุปกรณ์ทางอิเล็กทรอนิกส์ (electronic device) ที่มนุษย์ใช้เป็นเครื่องมือช่วยในการจัดการกับข้อมูลที่อาจเป็นได้ ทั้งตัวเลข ตัวอักษร หรือสัญลักษณ์ที่ใช้แทนความหมายในสิ่งต่างๆ โดยคุณสมบัติที่สำคัญของคอมพิวเตอร์คือการที่สามารถกำหนดชุดคำสั่งล่วงหน้าหรือโปรแกรมได้ (programmable) นั่นคือคอมพิวเตอร์สามารถทำงานได้หลากหลายรูปแบบ ขึ้นอยู่กับชุดคำสั่งที่เลือกมาใช้งาน ทำให้สามารถนำคอมพิวเตอร์ไปประยุกต์ใช้งานได้อย่างกว้างขวาง เช่น ใช้ในการตรวจสอบความถูกต้องของหัวใจ ระบบการฝาก - ถอนเจ็บในธนาคาร หุ้นยนต์ช่วยงานการผลิตในอุตสาหกรรม เป็นต้น

เนื่องจากคอมพิวเตอร์เป็นเครื่องมือที่สำคัญในการเขียนหรือพัฒนาโปรแกรม ใช้ทดสอบและรันโปรแกรม (ประมวลโปรแกรม) การมีความรู้เกี่ยวกับองค์ประกอบของระบบคอมพิวเตอร์เป็นสิ่งสำคัญในการเขียนโปรแกรม จะทำให้เข้าใจการทำงานของฮาร์ดแวร์และซอฟต์แวร์ ช่วยพัฒนาโปรแกรมที่มีประสิทธิภาพ การจัดการทรัพยากรอย่างมีประสิทธิภาพ เช่น หน่วยความจำและการประมวลผล และการแก้ปัญหาและการตรวจสอบหาข้อผิดพลาดในโปรแกรม(ดีบัก) ทำให้นักพัฒนาโปรแกรมสามารถวิเคราะห์และแก้ไขข้อผิดพลาดที่เกี่ยวข้องกับการทำงานของระบบได้อย่างรวดเร็วและถูกต้อง

1.2.1 ประวัติของคอมพิวเตอร์

การพัฒนาคอมพิวเตอร์มีการเปลี่ยนแปลงและความก้าวหน้ามาโดยตลอด ตั้งแต่การเริ่มต้นจนถึงยุคปัจจุบัน คอมพิวเตอร์ในยุคแรกๆ มีการพัฒนาอย่างช้าๆ และใช้สำหรับการคำนวณทางวิทยาศาสตร์และการทหารเป็นหลัก ดังรูปที่ 1.2 แสดงลำดับของข้อเครื่องคอมพิวเตอร์ และได้ลำดับวิวัฒนาการดังนี้



รูปที่ 1.2 ลำดับข้อเครื่องคอมพิวเตอร์ที่ถูกพัฒนาขึ้นในแต่ละปี

แหล่งที่มารูป: www.sofworld.org

ในช่วงปี ค.ศ. 1930-1940 มีการพัฒนา ENIAC ซึ่งเป็น คอมพิวเตอร์ที่สามารถโปรแกรมได้ และมีประสิทธิภาพเป็นเครื่องแรก โดยใช้หลอดสูญญากาศ และมีขนาดใหญ่มาก ต่อมาในปี ค.ศ. 1941 คอนราด ชูส (Konrad Zuse) ได้สร้าง Z1 Computer ซึ่งเป็นคอมพิวเตอร์เครื่องแรกที่สามารถตั้งโปรแกรมได้อย่างอิสระ การพัฒนาในเชิงพาณิชย์เริ่มต้นขึ้นในปี ค.ศ. 1953 เมื่อบริษัท IBM เปิดตัว EDPM และในปี ค.ศ. 1954 จอห์น แบคคัส (John Backus) และ IBM ได้สร้างภาษาโปรแกรมระดับสูงภาษาแรก คือ FORTRAN นอกจากนี้ ในปี ค.ศ. 1955 ยังมีการสร้าง ERMA และ MICR เพื่อนำมาใช้ในธุรกิจธนาคาร

ในปี ค.ศ. 1958 แจ็ค กิลบี (Jack Kilby) และ โรเบิร์ต โนยซ์ (Robert Noyce) ได้สร้างวงจรรวม (Integrated Circuit) ซึ่งช่วยให้คอมพิวเตอร์มีขนาดเล็กลงและมีประสิทธิภาพมากขึ้น ต่อมาในปี ค.ศ. 1962 สตีฟ รัสเซลล์ (Steve Russell) และทีมงานที่ MIT ได้พัฒนา เกมคอมพิวเตอร์เกมแรกของโลก ชื่อ "Spacewar" ซึ่งเป็นจุดเริ่มต้นของการเกมคอมพิวเตอร์ ในปี ค.ศ. 1964 ดักลาส เอองเกลบาร์ต (Douglas Engelbart)

ได้ประดิษฐ์ มาส์ และระบบปฏิบัติการแบบบินโดว์ซ์ ซึ่งเปลี่ยนแปลงวิธีการใช้งานคอมพิวเตอร์อย่างมาก ต่อมาในปี ค.ศ. 1969 ARPAnet ถือกำเนิดขึ้น ซึ่งนับเป็นจุดเริ่มต้นของ อินเทอร์เน็ต

ในปี ค.ศ. 1970 Intel ได้พัฒนา RAM และในปี ค.ศ. 1971 ได้พัฒนา ไมโครprocessor ตัวแรก รวมถึง พล๊อปปี้ดิสก์ ซึ่งเป็นอุปกรณ์จัดเก็บข้อมูลแบบถอดออกได้ ต่อมาในปี ค.ศ. 1973 โรเบิร์ต เมตคาล์ฟ (Robert Metcalfe) และบริษัท เชร็อกซ์ (Xerox) ได้พัฒนาระบบอีเทอร์เน็ต (Ethernet) ซึ่งเป็นเทคโนโลยีสำคัญที่ช่วยให้คอมพิวเตอร์สามารถเชื่อมต่อกันเป็นเครือข่ายได้ ในช่วงปี ค.ศ. 1974-1975 มีการวางแผนนำ คอมพิวเตอร์สำหรับผู้ใช้ทั่วไป เช่น Apple I, Apple II, TRS-80 และ Commodore Pet ต่อมาในปี ค.ศ. 1981 บริษัท ไมโครซอฟท์ (Microsoft) ได้วางจำหน่าย MS-DOS ซึ่งกล่าวเป็นระบบปฏิบัติการที่ได้รับความนิยมอย่างแพร่หลาย

ในปี ค.ศ. 1983 Apple ได้เปิดตัว Apple Lisa ซึ่งเป็นคอมพิวเตอร์เครื่องแรกที่ใช้ระบบ GUI (Graphical User Interface) และในปี ค.ศ. 1984 ได้วางจำหน่าย Apple Macintosh ซึ่งช่วยให้การใช้ คอมพิวเตอร์ส่วนบุคคลแพร่หลายมากขึ้น ต่อมาในปี ค.ศ. 1985 Microsoft ได้เปิดตัว Windows ซึ่งเป็นระบบปฏิบัติการที่มี ส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) ทำให้คอมพิวเตอร์ใช้งานได้ง่ายขึ้น การเปลี่ยนแปลงครั้ง สำคัญเกิดขึ้นในปี ค.ศ. 1991 เมื่อ ทิม เบอร์เนอร์ส-ลี (Tim Berners-Lee) ได้สร้าง World Wide Web (WWW) ซึ่งทำให้การใช้งานอินเทอร์เน็ตแพร่หลายและสะดวกยิ่งขึ้น ต่อมาในปี ค.ศ. 2007 Apple ได้เปิดตัว iPhone ซึ่งเป็นสมาร์ทโฟนที่รวมการทำงานของโทรศัพท์มือถือ คอมพิวเตอร์ และอุปกรณ์ความบันเทิงไว้ใน เครื่องเดียว

ในปี ค.ศ. 2011 Apple ได้เปิดตัว Siri ซึ่งเป็นระบบ ผู้ช่วยเสียงที่ใช้ปัญญาประดิษฐ์ (AI) และในปี ค.ศ. 2014 บริษัท Google DeepMind ได้พัฒนา AI ที่สามารถเล่นเกมและทำงานที่ซับซ้อนได้อย่างมี ประสิทธิภาพเมื่อเกิด การแพร่ระบาดของ COVID-19 ในปี ค.ศ. 2020 การใช้ เทคโนโลยีและซอฟต์แวร์ สำหรับการทำงานระยะไกล เพิ่มขึ้นอย่างมาก และในปี ค.ศ. 2021 บริษัท IBM และบริษัทอื่นๆ ได้พัฒนา Quantum Processor ซึ่งช่วยเพิ่มประสิทธิภาพในการประมวลผลข้อมูล และสามารถแก้ปัญหาที่ซับซ้อนได้ดีขึ้น

1.2.2 องค์ประกอบของระบบคอมพิวเตอร์

ระบบคอมพิวเตอร์ประกอบด้วยองค์ประกอบหลักหลายส่วนที่ทำงานร่วมกันอย่างเป็นระบบ เพื่อให้ คอมพิวเตอร์สามารถทำงานตามที่ผู้ใช้ต้องการได้ องค์ประกอบสำคัญได้แก่ ฮาร์ดแวร์ที่ทำหน้าที่ประมวลผล และจัดเก็บข้อมูล ซอฟต์แวร์ที่ควบคุมการทำงานและการประมวลผลข้อมูล ข้อมูลที่ถูกจัดเก็บและประมวลผล โดยระบบ และผู้ใช้ที่ทำการป้อนข้อมูลและรับผลลัพธ์จากคอมพิวเตอร์ องค์ประกอบเหล่านี้ทำงานร่วมกัน เพื่อให้การใช้งานคอมพิวเตอร์เป็นไปอย่างมีประสิทธิภาพ โดยระบบคอมพิวเตอร์ประกอบด้วยองค์ประกอบ หลักที่สำคัญ ดังนี้

1. ฮาร์ดแวร์ (Hardware) ประกอบด้วย

1.1) หน่วยประมวลผลกลาง (CPU) เป็นสมองของคอมพิวเตอร์ ทำหน้าที่ประมวลผลคำสั่ง และ ควบคุมการทำงานของระบบ

1.2) หน่วยความจำ (Memory) แบ่งออกเป็นหน่วยความจำหลัก (RAM) ที่ใช้เก็บข้อมูลและโปรแกรมที่กำลังใช้งาน และหน่วยความจำรอง (เช่น ฮาร์ดดิสก์, SSD) สำหรับเก็บข้อมูลถาวร

1.3) อุปกรณ์อินพุต (Input Devices) เช่น คีย์บอร์ด เม้าส์ และไมโครโฟน ใช้ในการป้อนข้อมูลเข้าสู่ระบบ

1.4) อุปกรณ์เอาต์พุต (Output Devices) เช่น หน้าจอ เครื่องพิมพ์ และลำโพง ใช้ในการแสดงผลข้อมูลจากระบบ

2. ซอฟต์แวร์ (Software) ประกอบด้วย

2.1) ระบบปฏิบัติการ (Operating System) ซอฟต์แวร์พื้นฐานที่ทำหน้าที่จัดการทรัพยากรของคอมพิวเตอร์และทำงานเป็นตัวกลางระหว่างฮาร์ดแวร์และซอฟต์แวร์อื่นๆ

2.2) ซอฟต์แวร์ประยุกต์ (Application Software) โปรแกรมที่ผู้ใช้สามารถใช้งานเพื่อทำงานเฉพาะเจาะจง เช่น โปรแกรมประมวลคำ (Word Processor) โปรแกรมตารางคำนวณ (Spreadsheet) และเว็บเบราว์เซอร์

3. ข้อมูล (Data) ข้อมูลที่คอมพิวเตอร์ประมวลผล ซึ่งอาจเป็นตัวเลข ข้อความ ภาพ เสียง หรือวิดีโอ ข้อมูลเหล่านี้จะถูกจัดเก็บและประมวลผลโดยฮาร์ดแวร์และซอฟต์แวร์

4. ผู้ใช้ (Users): บุคคลที่ใช้งานคอมพิวเตอร์ในการป้อนข้อมูล สั่งการ และรับผลลัพธ์จากระบบ

นอกจากนี้ องค์ประกอบเพิ่มเติมของระบบคอมพิวเตอร์ สามารถแบ่งได้ตามอุปกรณ์ ดังนี้

1. หน่วยอินพุต/เอาต์พุต (I/O Units) อินพุต คือ อุปกรณ์ที่ใช้ในการนำเข้าข้อมูล เช่น แป้นพิมพ์ เม้าส์ และสแกนเนอร์ และเอาต์พุต คือ อุปกรณ์ที่ใช้ในการส่งออกข้อมูล เช่น จอภาพ เครื่องพิมพ์ และลำโพง

2. ระบบเครือข่าย (Networking) การเชื่อมต่อระหว่างคอมพิวเตอร์หลายเครื่องเพื่อการสื่อสารและการแลกเปลี่ยนข้อมูล ซึ่งรวมถึงอุปกรณ์เครือข่าย เช่น เร้าเตอร์ สวิตช์ และโมเด็ม

3. หน่วยจัดเก็บข้อมูล (Storage Devices) อุปกรณ์ที่ใช้ในการจัดเก็บข้อมูลถาวร เช่น ฮาร์ดดิสก์ ไ/drฟ์โซลิดสเตต (SSD) และอุปกรณ์จัดเก็บข้อมูลแบบพกพา (USB Drive)

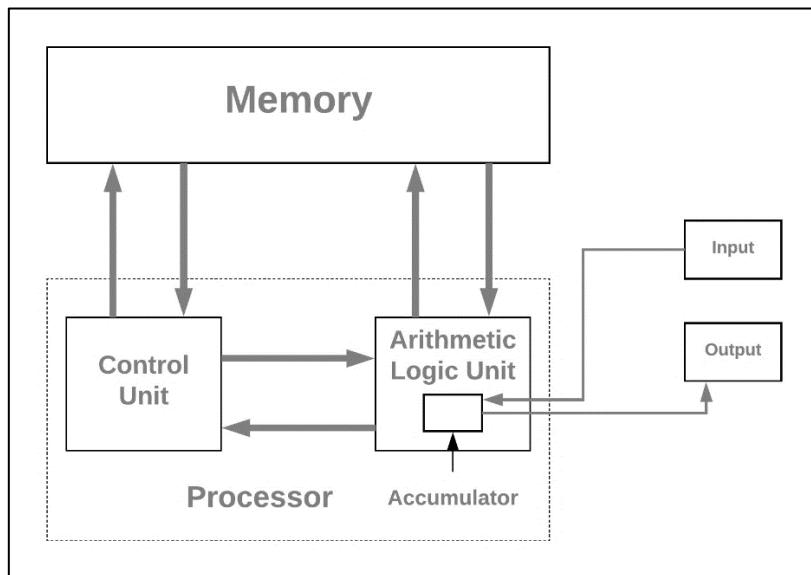
4. ระบบการจัดการฐานข้อมูล (Database Management Systems) ซอฟต์แวร์ที่ใช้ในการจัดการและจัดเก็บข้อมูลในรูปแบบฐานข้อมูล ทำให้สามารถค้นหาและดึงข้อมูลมาใช้งานได้อย่างมีประสิทธิภาพ

5. ซอฟต์แวร์ระบบ (System Software) ซอฟต์แวร์ที่ทำหน้าที่สนับสนุนการทำงานของระบบปฏิบัติการและช่วยจัดการทรัพยากรของระบบ เช่น ไดรเวอร์ระบบและเครื่องมือการจัดการระบบ

องค์ประกอบทั้งหมดนี้ทำงานร่วมกันเพื่อทำให้คอมพิวเตอร์สามารถทำงานตามที่ผู้ใช้ต้องการได้อย่างมีประสิทธิภาพ

1.2.3 หลักการทำงานของคอมพิวเตอร์

องค์ประกอบของระบบคอมพิวเตอร์รูปที่ 1.3 ที่ประกอบด้วยโครงสร้าง ของข้อมูลถูกส่งผ่านเข้ามาทางหน่วยรับข้อมูล (Input Unit) ก็จะถูกส่งต่อเพื่อนำไปจัดเก็บหรือพักข้อมูลไว้ชั่วคราวที่หน่วยความจำ (Memory Unit) ก่อน จากนั้นจึงค่อยๆ ทยอยจัดส่งข้อมูลต่าง ๆ ที่จัดเก็บไว้ไปให้หน่วยประมวลผล (Processing Unit) เพื่อประมวลผลข้อมูล แล้วส่งไปยังหน่วยสุดท้ายคือ หน่วยแสดงผล(Output Unit) เพื่อทำการแสดงผลออกทางอุปกรณ์ต่าง ๆ ต่อไป



รูปที่ 1.3 หลักการทำงานของระบบคอมพิวเตอร์

แหล่งที่มา: <https://www.geeksforgeeks.org/what-is-digital-signal-processing/>

สามารถอธิบายหลักการทำงานของแต่ละหน่วยพoSงเบปีเดีดังนี้

- หน่วยรับข้อมูล (Input Unit) หน่วยรับข้อมูลเป็นส่วนแรกที่ติดต่อกับผู้ใช้หน้าที่หลักคือ ตอบสนอง การสั่งงานจากผู้ใช้แล้วรับเป็นสัญญาณข้อมูลส่งต่อไปจัดเก็บหรือพักไว้ที่หน่วยความจำ ซึ่งอุปกรณ์ที่ ทำหน้าที่เป็นหน่วยรับข้อมูลมีมากมาย เช่น เม้าส์ (Mouse) คีย์บอร์ด (Keyboard) จอยสติ๊ก (Joystick) ทัชแพด (Touch Pad) เป็นต้น
- หน่วยประมวลผล (Processing Unit) หน่วยประมวลผลถือเป็นส่วนที่สำคัญที่สุดของเครื่อง คอมพิวเตอร์เปรียบได้กับสมองของมนุษย์หน้าที่หลักของหน่วยนี้คือ นำเอาข้อมูลที่ถูกจัดเก็บหรือพักไว้ในหน่วยความจำ มาทำการคิดคำนวณประมวลผลข้อมูลทางคณิตศาสตร์ (Arithmetic Operation) และเปรียบเทียบข้อมูลทางตรรกศาสตร์(Logical Operation) จันได้ผลลัพธ์ออกมาแล้วจึงค่อยส่งข้อมูลที่เป็นผลลัพธ์เหล่านั้นไปยังหน่วยแสดงผลต่อไป อุปกรณ์ที่ทำหน้าที่เป็นหน่วยประมวลผลในเครื่องคอมพิวเตอร์คือ ซีพียู (Central Processing Unit)
- หน่วยความจำ (Memory Unit) หน่วยความจำเป็นบทที่สำคัญ ที่จะต้องทำงานร่วมกันกับหน่วยประมวลผลอยู่โดยตลอด หน้าที่หลักคือ จดจำและบันทึกข้อมูลต่าง ๆ ที่ถูกส่งมาจากหน่วยรับข้อมูล จัดเก็บไว้ชั่วคราว ก่อนที่จะส่งต่อไปให้หน่วยประมวลผล นอกจากนี้ยังทำหน้าที่เป็นเสมือนกระดาษทด สำหรับให้หน่วยประมวลผลใช้คิดคำนวณประมวลผลข้อมูลต่าง ๆ ด้วย

- หน่วยแสดงผล (Output Unit) หน่วยแสดงผลเป็นบทที่ใช้ในการแสดงผลลัพธ์ที่ได้ออกมาในรูปแบบต่าง ๆ กันตามแต่ละอุปกรณ์ เช่น สัญญาณภาพออกสู่หน้าจอ และงานพิมพ์จากเครื่องพิมพ์ เป็นต้น

การพัฒนาโปรแกรมคอมพิวเตอร์ไม่ได้จำกัดอยู่เพียงแค่การเขียนชุดคำสั่ง (โค้ด) เท่านั้น แต่ยังเกี่ยวข้องกับความเข้าใจในองค์ประกอบต่าง ๆ ของระบบคอมพิวเตอร์ เช่น ฮาร์ดแวร์ ซอฟต์แวร์ และระบบปฏิบัติการ ซึ่งองค์ประกอบเหล่านี้ทำงานร่วมกันอย่างเป็นระบบ ความเข้าใจในองค์ประกอบเหล่านี้จะช่วยให้สามารถเขียนโปรแกรมได้อย่างมีประสิทธิภาพ รวมถึงสามารถแก้ไขปัญหาทางเทคนิคที่อาจเกิดขึ้น เช่น ปัญหาด้านการประมวลผลและการจัดสรรทรัพยากรของระบบ นอกจากนี้ ความรู้พื้นฐานเกี่ยวกับองค์ประกอบของระบบคอมพิวเตอร์ยังเป็นรากฐานที่สำคัญสำหรับการเรียนรู้ในหัวข้อที่เกี่ยวข้องอื่น ๆ เช่น เครื่อข่ายคอมพิวเตอร์ ความปลอดภัยของระบบ และการดูแลรักษาระบบ

1.3 ระบบปฏิบัติการและการเขียนโปรแกรม

ระบบปฏิบัติการ (Operating System: OS) คือซอฟต์แวร์ที่ทำหน้าที่เป็นตัวกลางระหว่างฮาร์ดแวร์ของคอมพิวเตอร์และผู้ใช้งานหรือโปรแกรมต่างๆ มันจัดการทรัพยากรของระบบ เช่น หน่วยความจำ พื้นที่จัดเก็บ และการประมวลผล และทำให้การทำงานของโปรแกรมและอุปกรณ์ต่างๆ ทำงานร่วมกันได้อย่างราบรื่น ตัวอย่างของระบบปฏิบัติการที่เป็นที่นิยมได้แก่ Windows, macOS, Linux, และ Android ความสำคัญของระบบปฏิบัติการ ได้แก่

1. การจัดการทรัพยากร ระบบปฏิบัติการทำหน้าที่จัดการทรัพยากรต่างๆ ของคอมพิวเตอร์ เช่น CPU, หน่วยความจำ และอุปกรณ์จัดเก็บข้อมูล เพื่อให้โปรแกรมต่างๆ สามารถทำงานได้อย่างมีประสิทธิภาพ และไม่เกิดปัญหาการแย่งทรัพยากรกัน
2. การจัดการไฟล์และข้อมูล ระบบปฏิบัติการจัดการโครงสร้างและการเข้าถึงไฟล์และข้อมูลในระบบ ทำให้ผู้ใช้งานสามารถจัดเก็บ เรียกดู และแก้ไขข้อมูลได้อย่างง่ายดาย
3. ความปลอดภัย ระบบปฏิบัติการมีการจัดการสิทธิ์การเข้าถึงข้อมูลและทรัพยากรต่างๆ เพื่อป้องกันการเข้าถึงโดยไม่ได้รับอนุญาตและรักษาความปลอดภัยของข้อมูล
4. การเชื่อมต่อและการสื่อสาร ระบบปฏิบัติการสนับสนุนการเชื่อมต่อและการสื่อสารระหว่างอุปกรณ์ต่างๆ เช่น การเชื่อมต่อเครือข่ายอินเทอร์เน็ต และการสื่อสารระหว่างโปรแกรม

การเขียนโปรแกรมคอมพิวเตอร์จำเป็นจะต้องเขียนบนระบบปฏิบัติการดังนั้นการเข้าใจระบบปฏิบัติการจึงมีประโยชน์ ดังนี้

1. การจัดการทรัพยากร ระบบปฏิบัติการทำหน้าที่จัดการทรัพยากรของระบบคอมพิวเตอร์ เช่น หน่วยประมวลผลกลาง (CPU), หน่วยความจำ (RAM), และอุปกรณ์จัดเก็บข้อมูล การจัดการทรัพยากรเหล่านี้ทำให้โปรแกรมสามารถทำงานได้อย่างมีประสิทธิภาพและไม่เกิดการแย่งทรัพยากรกัน

2. การจัดการไฟล์และระบบไฟล์ ระบบปฏิบัติการมีการจัดการโครงสร้างและการเข้าถึงไฟล์และระบบไฟล์ โปรแกรมต้องอาศัยระบบปฏิบัติการในการอ่าน เขียน และจัดการไฟล์ต่างๆ ซึ่งเป็นพื้นฐานสำคัญในการพัฒนาโปรแกรม

3. การจัดการกระบวนการ (Process Management) ระบบปฏิบัติการจัดการกระบวนการทำงานของโปรแกรมหลายๆ โปรแกรมในเวลาเดียวกัน โดยการแบ่งเวลาในการประมวลผลและการจัดการลำดับการทำงาน ทำให้โปรแกรมสามารถทำงานร่วมกันได้โดยไม่เกิดข้อขัดแย้ง

4. การจัดการความปลอดภัย ระบบปฏิบัติการมีการจัดการสิทธิ์การเข้าถึงทรัพยากรต่างๆ เช่น การจัดการสิทธิ์ของผู้ใช้และโปรแกรมเพื่อป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต โปรแกรมสามารถพึงพาระบบปฏิบัติการในการรักษาความปลอดภัยของข้อมูลและทรัพยากร

5. การสนับสนุนการพัฒนาโปรแกรม ระบบปฏิบัติการมีเครื่องมือและไลบรารีต่างๆ ที่ช่วยในการพัฒนาโปรแกรม เช่น คอมไพล์เตอร์ (Compiler), อินเตอร์พรีเตอร์ (Interpreter), และดีบักเกอร์ (Debugger) ซึ่งทำให้นักพัฒนาสามารถพัฒนา ทดสอบ และแก้ไขโปรแกรมได้อย่างมีประสิทธิภาพ

6. การเชื่อมต่อและการสื่อสาร ระบบปฏิบัติการสนับสนุนการเชื่อมต่อและการสื่อสารระหว่างอุปกรณ์และโปรแกรม เช่น การเชื่อมต่ออินเทอร์เน็ตและการสื่อสารผ่านเครือข่าย ซึ่งเป็นสิ่งสำคัญในการพัฒนาโปรแกรมที่ต้องการการสื่อสารระหว่างเครื่องหรือกับเซิร์ฟเวอร์

ระบบปฏิบัติการมีความสำคัญในการจัดการทรัพยากร การจัดการกระบวนการ การรักษาความปลอดภัย และการสนับสนุนเครื่องมือพัฒนาโปรแกรม ซึ่งทำให้การเขียนโปรแกรมเป็นไปได้อย่างมีประสิทธิภาพและสามารถพัฒนาโปรแกรมที่ทำงานได้ดีบนระบบคอมพิวเตอร์ต่างๆ

ระบบปฏิบัติการมีหน้าที่เป็นตัวกลางในการประสานการทำงานระหว่างฮาร์ดแวร์และซอฟต์แวร์ของคอมพิวเตอร์ ทำให้โปรแกรมต่าง ๆ สามารถเข้าถึงและใช้งานทรัพยากรของคอมพิวเตอร์ได้อย่างเต็มประสิทธิภาพ สำหรับนักพัฒนาโปรแกรม ความเข้าใจในระบบปฏิบัติการเป็นสิ่งจำเป็น เนื่องจากช่วยให้สามารถออกแบบและพัฒนาโปรแกรมที่ทำงานได้อย่างมีประสิทธิภาพบนแพลตฟอร์มที่หลากหลาย เช่น Windows macOS และ Linux นอกจากนี้ ยังช่วยในการแก้ไขปัญหาที่เกี่ยวข้องกับการจัดการหน่วยความจำ และการควบคุมกระบวนการทำงานของระบบ

1.4 วัตถุประสงค์ของการเรียนรู้

ภาษาโปรแกรมคอมพิวเตอร์(Computer Programming Language) คือ ชุดคำสั่งที่นักเขียนโปรแกรม หรือโปรแกรมเมอร์ (Programmer) เขียนโปรแกรมชอร์สโค้ด (Source Code) ที่ถูกต้องตามหลักไวยากรณ์ของภาษาโปรแกรมคอมพิวเตอร์ เพื่อให้สามารถติดต่อสื่อสาร ควบคุมการรับส่งข้อมูล และสั่งให้คอมพิวเตอร์ทำงานตามที่นักเขียนโปรแกรมต้องการได้

ชั้นภาษาโปรแกรมคอมพิวเตอร์ในปัจจุบันนี้ มีหลายภาษาให้เลือกใช้งาน ขึ้นอยู่กับความสนใจหรือความสามารถของนักพัฒนาโปรแกรม (Programmer) ที่จะเลือกใช้ภาษาโปรแกรมให้เหมาะสมกับโปรแกรมหรือหน้าที่งานที่จะนำไปใช้ เช่น ภาษา C, ภาษา ASP, ภาษา Delphi, ภาษา HTML เป็นต้น

แต่ภาษาโปรแกรมต่าง ๆ เหล่านี้ รูปแบบการเขียนคำสั่งเพื่อให้คอมพิวเตอร์ทำงานจะเป็นการเรียบเรียงไวยากรณ์ภาษาอังกฤษ ตามรูปแบบคำสั่งของภาษาโปรแกรมต่าง ๆ ซึ่งในการประมวลผลคำสั่งภาษาโปรแกรมนั้น คอมพิวเตอร์จะไม่สามารถเข้าใจในคำสั่งต่าง ๆ เหล่านี้ได้โดยตรง เนื่องจากคอมพิวเตอร์จะเข้าใจและประมวลผลด้วยภาษาเครื่อง (Machine Language) หรือในรูปแบบของเลขฐานสอง (Binary digit) คือเลข 0 และ เลข 1 เท่านั้น ดังนั้น จึงมีขั้นตอนในการแปลภาษาโปรแกรมต่าง ๆ ให้เป็นภาษาเครื่องหรือในรูปแบบเลขฐานสองก่อน โดยใช้โปรแกรมแปลภาษา (Language Translator Program) เช่น แอสเซมเบลอร์ (Assembler), คอมไพล์เตอร์ (Compiler) หรือ อินเตอร์พรีเตอร์ (Interpreter) คอมพิวเตอร์จึงจะเข้าใจและสามารถประมวลผลตามคำสั่งของภาษาโปรแกรมต่าง ๆ เหล่านี้ได้

ภาษาโปรแกรมที่ใช้ในการพัฒนาโปรแกรมขึ้นมาใช้งานนั้น มีอยู่ด้วยกันหลายภาษาซึ่งแต่ละภาษาจะมีคุณสมบัติและความสม่ำเสมอในการนำมาใช้ในการพัฒนาโปรแกรมแตกต่างกัน ชั้นภาษาโปรแกรมที่ใช้งานตั้งแต่ต่อเดือนถึงปัจจุบันนี้ สามารถแบ่งได้เป็น 5 ยุค ดังนี้

ยุคที่ 1 : ภาษาเครื่อง (Machine Language) เป็นภาษาโปรแกรมคอมพิวเตอร์ระดับต่ำที่สุด ชั้นคอมพิวเตอร์เข้าใจได้ทันทีโดยไม่ต้องผ่านตัวแปลภาษา เพราะเขียนคำสั่งและแทนข้อมูลด้วยเลขฐานสอง (Binary Code) ทั้งหมด ซึ่งเป็นการเขียนคำสั่งด้วยเลข 0 หรือ 1 ตัวต่อตัวอย่างคำสั่งภาษาเครื่อง ดังรูปที่ 1.4

```

W1-0.ram
0011000000000000 ; read n -> acc ;
1011000000001010 ; jump to Done if n < 0. ;
0101000000010000 ; add sum to the acc ;
0010000000010000 ; store the new sum ;
1001000000000000 ; go back & read in next number ;
0001000000010000 ; load the final sum ;
0100000000000000 ; output the final sum ;
0000000000000000 ; stop ;
0000000000000000 ; 2-byte location where sum is stored ;
0000000000000000 ;
0000000000000000 ;
0000000000000000 .

```

รูปที่ 1.4 ภาษาเครื่อง (Machine Language)

แหล่งที่มารูป: https://www.researchgate.net/figure/A-simple-machine-language-program_fig6_238779569

ก่อนปี ค.ศ. 1952 มีการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาเครื่องเพียงภาษาเดียวเท่านั้นที่ใช้ติดต่อกับคอมพิวเตอร์โดยตรง และคอมพิวเตอร์แต่ละเครื่องจะมีภาษาเครื่องแตกต่างกันขึ้นอยู่กับชนิดของเครื่องคอมพิวเตอร์และหน่วยประมวลผลกลาง (Central Processor Unit: CPU) โดยมีรูปแบบคำสั่งเฉพาะเครื่อง

ดังนั้นนักเขียนโปรแกรมจึงไม่นิยมที่จะเขียนโปรแกรมด้วยภาษาเครื่อง เพราะทำการแก้ไข และเขียนโปรแกรมได้ยากทำให้เกิดยุ่งยากในการจัดจำ และเขียนคำสั่งต้องใช้เวลามากในการเขียนโปรแกรม รวมทั้งการหาข้อผิดพลาดจากการทำงานของโปรแกรม และโปรแกรมที่เขียนขึ้นทำงานเฉพาะคอมพิวเตอร์ที่มีฮาร์ดแวร์เดียวกันเท่านั้น (Machine Dependent)

ข้อดีของภาษาเครื่อง คือสามารถเขียนโปรแกรมควบคุมการทำงานคอมพิวเตอร์ได้โดยตรง และสั่งงานให้คอมพิวเตอร์ทำงานได้อย่างรวดเร็ว

ยุคที่ 2 ภาษาแอสเซมบลี (Assembly Language) ภาษาแอสเซมบลี จัดอยู่ในภาษาระดับต่ำ และเป็นภาษาที่พัฒนาต่อมาจากภาษาเครื่องในปี ค.ศ. 1952 ภาษาแอสเซมบลีมีความใกล้เคียงกับภาษาเครื่องมาก คือ 1 คำสั่งของภาษาแอสเซมบลีจะเท่ากับ 1 คำสั่งของภาษาเครื่อง โดยที่ภาษาแอสเซมบลีจะเขียนคำสั่งเป็นตัวอักษรภาษาอังกฤษ เพื่อใช้แทนคำสั่งภาษาเครื่อง ทำให้นักเขียนโปรแกรมสามารถเขียนโปรแกรมได้ง่ายขึ้น โดยการจัดจารหัสคำสั่งสั้น ๆ ที่จำได้ง่าย ซึ่งเรียกว่า นิมอนิกโค้ด (Mnemonic code)

```
; Test character for match. If a match, increment count.
; NOT    R1, R1
; ADD    R1, R1, R0 ; If match, R1 = xFFFF
; NOT    R1, R1 ; If match, R1 = x0000
; BRnp   GETCHAR ; If no match, do not increment
; ADD    R2, R2, #1

; Get next character from file.
; GETCHAR ADD    R3, R3, #1 ; Point to next character.
;           LDR    R1, R3, #0 ; R1 gets next char to test
;           BRnzp TEST

; Output the count.
; OUTPUT LD     R0, ASCII ; Load the ASCII template
;           ADD    R0, R0, R2 ; Convert binary count to ASCII
;           OUT    ; ASCII code in R0 is displayed.
;           HALT   ; Halt machine
```

รูปที่ 1.5 ภาษาแอสเซมบลี (Assembly Language)

แหล่งที่มาของรูป: <https://blog.naver.com/kdhkdh0407/12019445117?photoView=15>

เมื่อนักเขียนโปรแกรม เขียนโปรแกรมด้วยภาษาแอสเซมบลีแล้ว ต้องใช้ตัวแปลภาษาที่เรียกว่า แอสเซมเบลอร์ (Assembler) เพื่อแปลภาษาแอสเซมบลีให้เป็นภาษาเครื่อง จึงจะสามารถสั่งงานคอมพิวเตอร์ให้ทำงานได้

สรุปคำสั่งที่เขียนด้วยภาษาโปรแกรมคอมพิวเตอร์ ในยุคที่ 1 และที่ 2 จะต้องใช้เทคนิคการเขียนโปรแกรมสูง เพราะมีความยืดหยุ่นในการเขียนน้อยมาก และมีความยากในการเขียนคำสั่งสำหรับผู้เขียนโปรแกรม แต่สามารถควบคุมและเข้าถึงการทำงานของเครื่องคอมพิวเตอร์ได้โดยตรง และมีความรวดเร็วกว่า การใช้ภาษาระดับอื่น ๆ

ยุคที่ 3 ภาษาระดับสูง (High-level Language) ภาษาระดับสูงถือว่าเป็นภาษาโปรแกรมคอมพิวเตอร์ในยุคที่สาม (Third-generation language) ที่มีการใช้กันอย่างแพร่หลายในปี ค.ศ. 1960 โดยมีโครงสร้างภาษาและชุดคำสั่งเหมือนกับภาษาอังกฤษ รวมทั้งสามารถใช้นิพจน์ทางคณิตศาสตร์ในการคำนวณได้ด้วย ทำให้ผู้เขียนโปรแกรมสะดวกในการเขียนคำสั่งและแสดงผลลัพธ์ได้ตามต้องการ ลดความยุ่งยากในการเขียนโปรแกรมลงได้มาก ทั้งยังทำให้เกิดการใช้งานคอมพิวเตอร์เพื่อการประมวลผลเพิ่มขึ้น เช่น การควบคุมและสั่งงานเครื่องคอมพิวเตอร์เมนเฟรม การแก้ปัญหาเฉพาะด้านทางด้านอุตสาหกรรม เช่น การควบคุมเครื่องจักรกลต่าง ๆ เป็นต้น

การเขียนโปรแกรมด้วยภาษาจะต้องใช้ตัวแปลภาษา ที่เรียกว่า คอมไพล์เลอร์ (Compiler) เพื่อแปลภาษาจะต้องทำการตรวจสอบไวยากรณ์ของภาษาจะต้องสูงไปเป็นภาษาเครื่องเพื่อส่งให้เครื่องคอมพิวเตอร์ทำงานต่อไป โดยคอมไпал์เลอร์ของภาษาจะต้องสูงแต่ละภาษาจะแปลเฉพาะภาษาของตนเอง และทำงานได้เฉพาะเครื่องคอมพิวเตอร์ชนิดเดียวกันเท่านั้น เช่น คอมไпал์เลอร์ของภาษา COBOL บนเครื่องไมโครคอมพิวเตอร์ จะแปลภาษาเฉพาะคำสั่งของภาษา COBOL และจะทำงานได้บนเครื่องคอมพิวเตอร์ที่เหมือนกันเท่านั้น ถ้าต้องการนำไปใช้กับเครื่องคอมพิวเตอร์แบบอื่น ๆ เช่น เมนเฟรม จะต้องใช้คอมไпал์เลอร์ของภาษา COBOL แบบใหม่

ตัวอย่างของภาษาคอมพิวเตอร์ระดับสูงได้แก่ ภาษา BASIC ภาษา COBOL ภาษา FORTRAN และภาษา C ที่ได้รับความนิยมมาก เช่นกัน สามารถเขียนโปรแกรมแก้ปัญหาเฉพาะด้าน เช่น การควบคุมหุ่นยนต์ การสร้างภาพกราฟิก ได้เป็นอย่างดี เพราะมีความยืดหยุ่นและเหมาะสมกับการใช้งานทั่ว ๆ ไปได้

สรุปภาษาโปรแกรมคอมพิวเตอร์ในยุคที่ 3 มีการเขียนโปรแกรมที่ง่ายกว่าในยุคที่ 2 สามารถทำงานได้บนเครื่องคอมพิวเตอร์หลายระดับ (Machine Independent) โดยต้องใช้ควบคู่กับตัวแปลภาษา (Compiler or Interpreter) สำหรับเครื่องนั้น ๆ และมีความยืดหยุ่นในการแก้ปัญหาได้มากกว่าภาษาจะต้นต่อ

ยุคที่ 4 ภาษาจะต้นสูงมาก (Very high-level Language) ภาษาจะต้นสูงมากเป็นภาษา โปรแกรมคอมพิวเตอร์ยุคที่สี่ (Fourth-generation language) ซึ่งเป็นภาษาที่ใช้ในการเขียนโปรแกรมด้วยคำสั่งสั้น ๆ และง่ายกว่าภาษาในยุคก่อน ๆ มีการทำงานแบบไม่จำเป็นต้องบอกลำดับของขั้นตอนการทำงาน (Nonprocedural language) เพียงนักเขียนโปรแกรมกำหนดว่าต้องการให้โปรแกรมทำอะไรเท่านั้นโดยไม่ต้อง ทราบว่าทำได้อย่างไร ทำให้เขียนโปรแกรมได้ง่ายและรวดเร็วกว่าภาษาจะต้นสูงในยุคที่ 3 ที่มีการเขียนโปรแกรมแบบบอกขั้นตอนการทำงาน (Procedural language) ภาษาจะต้นสูงมากทำงานเหมือนกับภาษาพูด ว่าต้องการอะไรและเขียนเหมือนภาษาอังกฤษ ดังตัวอย่าง เช่น

ข้อดีของภาษาคอมพิวเตอร์ในยุคที่ 4

- การเขียนโปรแกรมจะสั้นและง่าย เพราะเน้นที่ผลลัพธ์ของงานว่าต้องการอะไร โดยไม่สนใจว่าจะทำได้อย่างไร
- การเขียนคำสั่ง สามารถทำได้ง่ายและแก้ไข เปลี่ยนแปลงโปรแกรมได้สะดวก ทำให้พัฒนาโปรแกรมได้รวดเร็วขึ้น
- ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้เร็ว โดยไม่ต้องเสียเวลาอบรม หรือ มีความรู้ด้านการเขียนโปรแกรมหรือไม่ เพราะชุดคำสั่งเหมือนภาษาพูด
- ผู้เขียนโปรแกรมไม่จำเป็นต้องทราบถึงฮาร์ดแวร์ ของเครื่องและโครงสร้างคำสั่งของภาษาโปรแกรม

ตัวอย่างภาษาคอมพิวเตอร์ในยุคที่ 4 ประกอบด้วย Report Generators, Query Language, Application Generators และ Interactive Database Management System Programs

ยุคที่ 5 ภาษารูมชาติ (Natural Language) ภาษารูมชาติจัดเป็นภาษาโปรแกรมคอมพิวเตอร์ ยุคที่ห้า (Fifth generation language) คือ การเขียนคำสั่ง หรือสั่งงานคอมพิวเตอร์ทำงานโดยการใช้

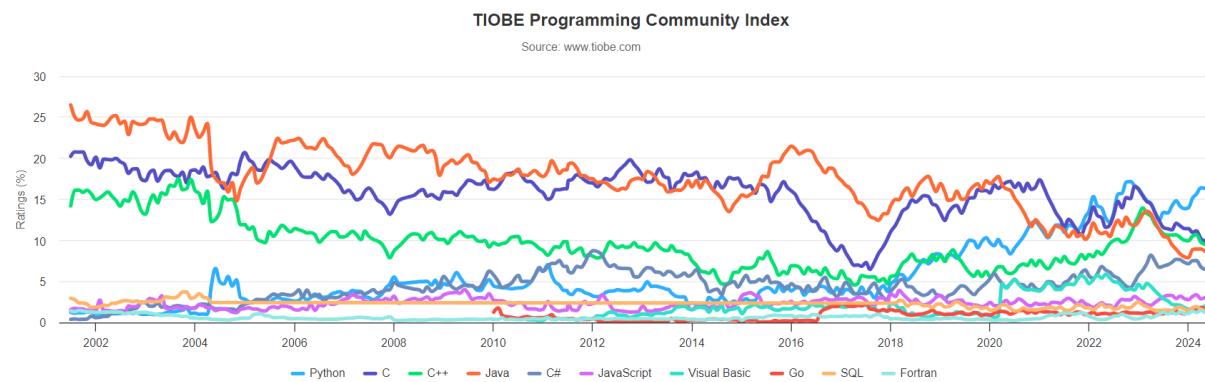
ภาษาธรรมชาติต่าง ๆ เช่น ภาพ หรือ เสียง โดยไม่สนใจรูปแบบไวยากรณ์หรือโครงสร้างของภาษาหากันนัก ซึ่งคอมพิวเตอร์จะพยายามคิดวิเคราะห์ และแปลความหมายโดยอาศัยการเรียนรู้ด้วยตนเองและระบบองค์ความรู้ (Knowledge Base System) มาช่วยแปลความหมายของคำสั่งต่าง ๆ และตอบสนองต่อผู้ใช้งาน

ข้อดีของภาษาคอมพิวเตอร์ในยุคที่ 5 คือผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้เร็ว โดยไม่ต้องมีความรู้ด้านการเขียนโปรแกรม แต่คอมพิวเตอร์ที่ใช้โปรแกรมต้องมีระบบปรับคำสั่ง และประมวลผลแบบอัจฉริยะ สามารถตอบสนองและทำงานได้หลายแบบ

วิัฒนาการของภาษาคอมพิวเตอร์สะท้อนให้เห็นถึงความพัฒนาเครื่องมือที่ช่วยให้มนุษย์สามารถสื่อสารกับคอมพิวเตอร์ได้่ายิ่งขึ้น จากยุคเริ่มต้นที่ต้องเขียนชุดคำสั่งด้วยรหัสเลขอctal สองที่ตัวช้อน ไปสู่ภาษาคอมพิวเตอร์ที่มีโครงสร้างใกล้เคียงกับภาษาอังกฤษ และปัจจุบันกำลังเข้าสู่ยุคที่คอมพิวเตอร์สามารถเข้าใจภาษามนุษย์หรือรับคำสั่งผ่านเสียงได้ การศึกษาและทำความเข้าใจวิัฒนาการของภาษาคอมพิวเตอร์จะช่วยให้เห็นถึงแนวโน้มการพัฒนาในอนาคต ซึ่งมุ่งเน้นไปที่การสร้างภาษาที่เป็นมิตรต่อผู้ใช้งานและการพัฒนาเทคโนโลยีที่สามารถทำงานร่วมกับมนุษย์ได้อย่างมีประสิทธิภาพ

1.5 ความสำคัญของภาษา Python

ภาษาคอมพิวเตอร์ระดับสูงมีหลากหลายภาษาที่ได้รับความนิยม เช่น C, C++, Java, C#, JavaScript, Python และ Go เป็นต้น ทั้งนี้ภาษา Python เป็นที่นิยมเนื่องจากเป็นภาษาที่มีความง่ายต่อการเรียนรู้และใช้งาน มีโครงสร้างที่อ่านง่าย และสามารถนำไปประยุกต์ใช้ในหลากหลายงาน ไม่ว่าจะเป็นการพัฒนาเว็บ แอปพลิเคชันทางวิทยาศาสตร์และวิศวกรรม การวิเคราะห์ข้อมูล ปัญญาประดิษฐ์ และแมชชีนเลิร์นนิง นอกจากนี้ Python ยังมีชุมชนผู้ใช้ที่ใหญ่และมีการสนับสนุนจากผู้พัฒนาอย่างต่อเนื่อง ทำให้มีไลบรารีและเครื่องมือที่หลากหลายพร้อมใช้งาน



รูปที่ 1.6 กราฟแสดงความนิยมของภาษาโปรแกรมต่างๆ ระหว่างปี 2002 ถึง 2024

แหล่งที่มา: <https://www.tiobe.com/tiobe-index/>

จากรูปที่ 1.6 กราฟที่แสดงในภาพนี้คือ "TIOBE Programming Community Index" ซึ่งแสดงความนิยมของภาษาโปรแกรมต่างๆ ตั้งแต่ปี 2002 ถึงปี 2024 โดยมีการจัดอันดับตามเปอร์เซ็นต์การใช้งานของแต่ละภาษา พบว่าภาษา Python มีความนิยมเพิ่มขึ้นอย่างต่อเนื่องและกลายเป็นภาษาที่ได้รับความนิยมสูงสุดใน

ปัจจุบัน ขณะที่ภาษา C และ Java เคยมีความนิยมสูงในอดีตแต่ลดลงในช่วงหลัง แต่ยังคงเป็นภาษาที่มีการใช้งานอย่างแพร่หลาย ภาษา Go และ JavaScript ก็แสดงให้เห็นถึงการเติบโตและความนิยมที่เพิ่มขึ้นในช่วงปีหลังๆ เช่นกัน ดังนั้น การเรียนรู้ภาษา Python มีความสำคัญและมีประโยชน์ในหลาย ๆ ด้าน ดังนี้

1. ง่ายต่อการเรียนรู้และใช้งาน Python มีไวยากรณ์ที่เรียบง่ายและชัดเจน ทำให้ง่ายต่อการเรียนรู้ สำหรับผู้เริ่มต้น นอกเหนือจาก Python ยังอ่านและเข้าใจได้ง่าย ทำให้การพัฒนาโปรแกรมเป็นไปอย่างรวดเร็วและมีประสิทธิภาพ

2. ความนิยมและการใช้งานที่แพร่หลาย Python เป็นหนึ่งในภาษาที่ได้รับความนิยมมากที่สุดในโลก มีชุมชนผู้ใช้ที่ใหญ่และมีการสนับสนุนจากผู้พัฒนาอย่างต่อเนื่อง ทำให้มีแหล่งข้อมูลและเครื่องมือมากมายที่สามารถนำไปใช้ในการพัฒนาโปรแกรม

3. การประยุกต์ใช้งานหลากหลาย Python สามารถนำไปใช้ในหลากหลายสาขา ไม่ว่าจะเป็นการพัฒนาเว็บ การวิเคราะห์ข้อมูล ปัญญาประดิษฐ์ (AI) และแมชชีนเลิร์นนิง (Machine Learning) การพัฒนา скриปต์สำหรับงานระบบ การคำนวณทางวิทยาศาสตร์ และอื่นๆ

4. เครื่องมือและไลบรารีที่มีประสิทธิภาพ Python มีไลบรารีและเฟรมเวิร์กที่หลากหลาย เช่น NumPy, Pandas, Matplotlib สำหรับการวิเคราะห์ข้อมูล Django และ Flask สำหรับการพัฒนาเว็บ และ TensorFlow และ PyTorch สำหรับการพัฒนา AI และแมชชีนเลิร์นนิง

5. โอกาสทางอาชีพ ความสามารถในการเขียนโปรแกรมด้วย Python เปิดโอกาสทางอาชีพมากมาย ในสาขาต่างๆ เช่น นักพัฒนาเว็บ นักวิเคราะห์ข้อมูล นักวิทยาศาสตร์ข้อมูล และวิศวกรซอฟต์แวร์ นอกจากนี้ ทักษะการเขียนโปรแกรม Python ยังเป็นที่ต้องการสูงในตลาดแรงงาน ทำให้มีโอกาสในการทำงานรายได้ที่ดี

การเรียนรู้ภาษา Python จึงเป็นการลงทุนที่มีคุณค่าในระยะยาว ช่วยเสริมทักษะการเขียนโปรแกรม และเปิดโอกาสในการพัฒนาอาชีพในหลายสาขาวิชา ซึ่งภาษา Python ได้ถูกนำเอามาใช้ในหลากหลายแขนง เช่น ด้านการแพทย์ วิทยาศาสตร์ มนุษยศาสตร์ และอื่นๆ

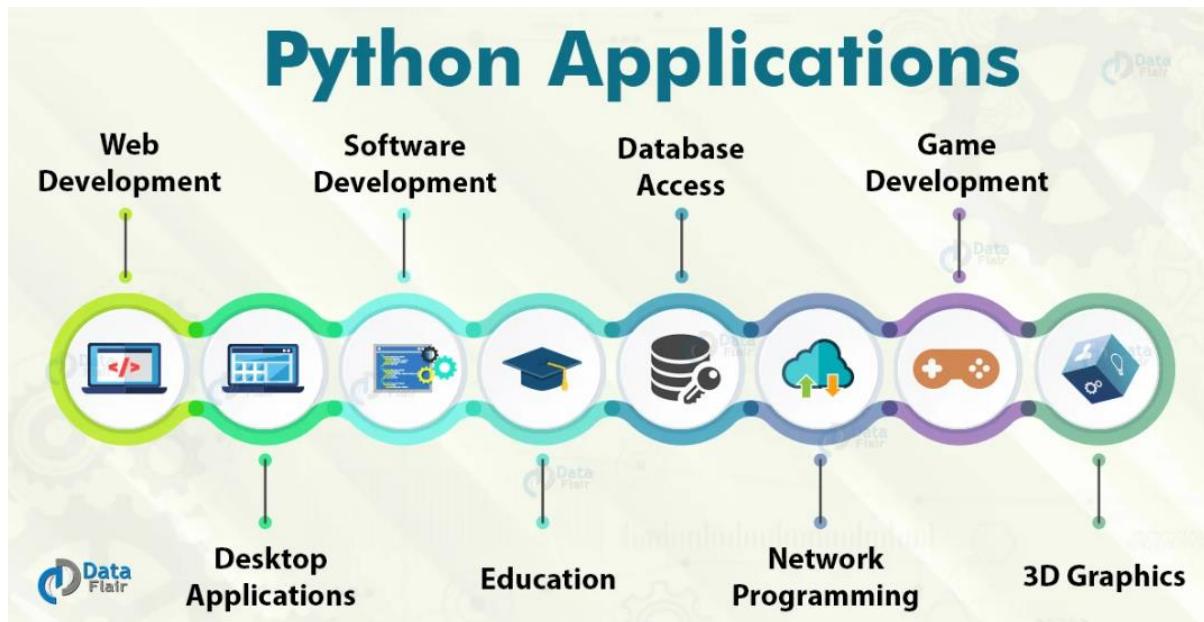
1. การพัฒนาเว็บและอินเทอร์เน็ต (Web and Internet Development) Python ถูกใช้ในการพัฒนาเว็บแอปพลิเคชันผ่านเฟรมเวิร์กต่างๆ เช่น Django และ Flask ที่ช่วยให้การพัฒนาเว็บเป็นไปอย่างรวดเร็วและมีประสิทธิภาพ นอกจากนี้ยังสามารถใช้ในการทำการดึงข้อมูลจากเว็บไซต์ (Web Scraping) โดยใช้ไลบรารีเช่น BeautifulSoup และ Scrapy เพื่อดึงข้อมูลจากเว็บไซต์ได้อย่างง่ายดาย

2. แอปพลิเคชัน GUI บนเดสก์ท็อป (Desktop GUI Applications) Python สามารถใช้ในการพัฒนาแอปพลิเคชันที่มีส่วนต่อประสานกับผู้ใช้ (GUI) โดยใช้ไลบรารีเช่น Tkinter PyQt หรือ Kivy ซึ่งช่วยให้ผู้พัฒนาสามารถสร้างโปรแกรมที่มีอินเทอร์เฟซใช้งานง่ายและตอบสนองความต้องการของผู้ใช้ได้ดี

3. วิทยาศาสตร์และการคำนวณทางตัวเลข (Science and Numeric) Python เป็นที่นิยมในการวิทยาศาสตร์และการคำนวณทางเลข เนื่องจากมีไลบรารีเช่น NumPy SciPy และ Matplotlib ที่ช่วยในการคำนวณเชิงวิทยาศาสตร์ การวิเคราะห์ข้อมูล และการสร้างกราฟอย่างมีประสิทธิภาพ

4. การพัฒนาซอฟต์แวร์ (Software Development) Python ใช้ในการพัฒนาและทดสอบซอฟต์แวร์อย่างกว้างขวาง โดยมีไลบรารีเช่น PyUnit สำหรับการทดสอบหน่วย (Unit Testing) และสามารถใช้ในการสร้างเครื่องมืออัตโนมัติและสคริปต์เพื่อเพิ่มประสิทธิภาพในการพัฒนาซอฟต์แวร์

5. การศึกษา (Education) Python เป็นภาษาที่เรียนรู้ได้ง่ายและมีความเรียบง่าย ทำให้เป็นที่นิยมในการใช้สอนการเขียนโปรแกรมในโรงเรียนและมหาวิทยาลัย มีแหล่งข้อมูลและบทเรียนมากมายที่ช่วยให้ผู้เรียนสามารถพัฒนาทักษะการเขียนโปรแกรมได้อย่างรวดเร็ว



รูปที่ 1.7 การประยุกต์ใช้งานภาษา Python

แหล่งที่มารูป: <https://data-flair.training/blogs/python-applications/>

6. การเข้าถึงฐานข้อมูล (Database Access) Python สามารถเชื่อมต่อกับฐานข้อมูลต่างๆ ได้ง่าย เช่น MySQL PostgreSQL SQLite และ MongoDB โดยใช้ไลบรารีเช่น SQLAlchemy และ PyMongo ซึ่งช่วยให้การจัดการและเข้าถึงข้อมูลเป็นไปอย่างมีประสิทธิภาพ

7. การเขียนโปรแกรมเครือข่าย (Network Programming) Python มีเครื่องมือและไลบรารีที่ช่วยในการพัฒนาโปรแกรมเครือข่าย เช่น Socket programming, Paramiko สำหรับการเชื่อมต่อ SSH และ Twisted สำหรับการพัฒนาแอปพลิเคชันที่ต้องการการสื่อสารแบบเรียลไทม์

8. การพัฒนาเกมและกราฟิก 3 มิติ (Games and 3D Graphics) Python ถูกใช้ในการพัฒนาเกม และกราฟิก 3D โดยใช้ไลบรารีเช่น Pygame สำหรับการพัฒนาเกมพื้นฐาน และ Blender สำหรับการสร้าง และปรับแต่งกราฟิก 3D ทำให้นักพัฒนาสามารถสร้างสรรค์เกมและกราฟิกที่มีความซับซ้อนได้

ภาษา Python ได้รับความนิยมอย่างรวดเร็วและต่อเนื่องในวงการเทคโนโลยีสารสนเทศ เนื่องจาก เป็นภาษาที่มีความยืดหยุ่นสูง สามารถนำไปใช้ในการพัฒนาโปรเจกต์ได้หลากหลายประเภท ไม่ว่าจะเป็น โครงการเล็กหรือระบบขนาดใหญ่ Python เหมาะสมสำหรับการพัฒนาเว็บไซต์ การวิเคราะห์ข้อมูล การพัฒนา ปัญญาประดิษฐ์และการเรียนรู้ของเครื่อง (Machine Learning) รวมถึงการพัฒนาระบบอัตโนมัติ ด้วยเหตุนี้ Python จึงกลายเป็นภาษาเริ่มต้นที่เหมาะสมสำหรับผู้ที่ต้องการเรียนรู้การเขียนโปรแกรม และเป็นเครื่องมือ หลักสำหรับนักพัฒนาที่ต้องการสร้างสรรค์นวัตกรรมในยุคดิจิทัล นอกจากนี้ Python ยังมีไลบรารีที่ หลากหลาย เช่น NumPy, Pandas และ TensorFlow ซึ่งช่วยสนับสนุนการทำงานในด้านต่าง ๆ ทำให้ Python เป็นภาษาที่มีความต้องการสูงในตลาดแรงงานปัจจุบัน

บทสรุป

ในยุคที่เทคโนโลยีเข้ามามีบทบาทสำคัญในชีวิตประจำวัน การเขียนโปรแกรมคอมพิวเตอร์ได้ กลายเป็นทักษะที่มีความสำคัญอย่างยิ่งยวด เปรียบเสมือนกุญแจสำคัญในการพัฒนาเทคโนโลยีล้ำสมัยต่างๆ ไม่ว่าจะเป็นปัญญาประดิษฐ์ (AI), อินเทอร์เน็ตของสรรพสิ่ง (IoT), เทคโนโลยีบล็อกเชน (Blockchain) และ นวัตกรรมอื่นๆ อีกมากมายที่กำลังขึ้นคลื่นโลกลสมัยใหม่ การเรียนรู้การเขียนโปรแกรมไม่เพียงแต่เปิดโอกาส ให้ได้สร้างสรรค์เทคโนโลยีเหล่านี้ แต่ยังช่วยเสริมสร้างทักษะการคิดวิเคราะห์และการแก้ปัญหาอย่างมีเหตุผล ซึ่งเป็นทักษะที่จำเป็นในทุกสาขาอาชีพ

การมีทักษะการเขียนโปรแกรมยังเปิดโอกาสทางอาชีพที่หลากหลายและน่าสนใจ เช่น นักพัฒนา โปรแกรม นักวิเคราะห์ข้อมูล วิศวกรซอฟต์แวร์ และอื่นๆ อีกมากมาย ที่กำลังเป็นที่ต้องการอย่างสูงใน ตลาดแรงงานเพื่อที่จะเขียนโปรแกรมได้อย่างมีประสิทธิภาพ จำเป็นต้องเข้าใจองค์ประกอบและหลักการ ทำงานของระบบคอมพิวเตอร์ ซึ่งประกอบด้วยส่วนประกอบหลักๆ ได้แก่ ฮาร์ดแวร์ ซอฟต์แวร์ ข้อมูล และผู้ใช้ ที่ทำงานร่วมกันอย่างเป็นระบบ องค์ประกอบสำคัญของฮาร์ดแวร์ ได้แก่ หน่วยประมวลผลกลาง (CPU) หน่วยความจำ อุปกรณ์อินพุต/เอ็ตพุต และระบบเครือข่าย โดยหลักการทำงานพื้นฐานของระบบ คอมพิวเตอร์คือการรับข้อมูล ประมวลผล และแสดงผล

ระบบปฏิบัติการ (Operating System: OS) ทำหน้าที่เป็นตัวกลางในการจัดการทรัพยากร่างๆ ของ เครื่องคอมพิวเตอร์ เช่น CPU หน่วยความจำ (RAM) ระบบไฟล์ และมีบทบาทสำคัญในการสนับสนุนการเขียน โปรแกรมให้ทำงานได้อย่างราบรื่น นอกจากนี้ ระบบปฏิบัติการยังมีเครื่องมือช่วยพัฒนาโปรแกรมต่างๆ เช่น คอมไพล์เตอร์ (Compiler) และดีบักเกอร์ (Debugger) ที่ช่วยให้นักพัฒนาสามารถเขียนโปรแกรมได้อย่างมี ประสิทธิภาพ

ภาษาคอมพิวเตอร์มีการพัฒนามาอย่างต่อเนื่อง โดยสามารถแบ่งออกเป็น 5 ยุคหลักๆ ได้แก่ ภาษาเครื่อง (Machine Language) ซึ่งเป็นภาษาที่คอมพิวเตอร์เข้าใจได้โดยตรง ภาษาแอสเซมบลี ซึ่งเป็นภาษาที่ใช้สัญลักษณ์แทนรหัสเครื่อง ภาษาระดับสูง (เช่น C, FORTRAN) ซึ่งเป็นภาษาที่มีโครงสร้างใกล้เคียงกับภาษาอังกฤษ ภาษาระดับสูงมาก (4GL) ซึ่งเป็นภาษาที่เน้นการใช้งานง่าย และภาษาธรรมชาติ ซึ่งเป็นภาษาที่ใกล้เคียงกับภาษาที่มนุษย์ใช้สื่อสาร แต่ละยุคของการพัฒนาภาษาคอมพิวเตอร์มีจุดมุ่งหมายเพื่อลดความซับซ้อนในการเขียนโปรแกรมและเพิ่มประสิทธิภาพในการใช้งาน

ในบรรดาภาษาคอมพิวเตอร์ทั้งหมด ภาษา Python ได้รับความนิยมอย่างแพร่หลายเนื่องจากมีไวยากรณ์ที่อ่านง่ายและใช้งานง่าย ทำให้เหมาะสมสำหรับการพัฒนาแอปพลิเคชันที่หลากหลาย เช่น เว็บแอปพลิเคชัน การวิเคราะห์ข้อมูล วิทยาศาสตร์ข้อมูล (Data Science) ปัญญาประดิษฐ์ (AI) การเรียนรู้ของเครื่อง (Machine Learning) และการพัฒนาเกม นอกจากนี้ Python ยังมีไลบรารี (Library) ที่หลากหลาย เช่น NumPy Pandas และ TensorFlow ซึ่งช่วยสนับสนุนการทำงานในด้านต่างๆ ทำให้ Python กลายเป็นภาษาที่ตลาดแรงงานต้องการมากที่สุดในปัจจุบันการเขียนโปรแกรมเป็นพื้นฐานสำคัญในการพัฒนาเทคโนโลยีและนวัตกรรมใหม่ๆ เช่น AI IoT และ Blockchain

โดยเนื้อหาบทนี้สามารถสรุปเป็นข้อได้ดังนี้

- โปรแกรมคอมพิวเตอร์ช่วยสร้างผลิตภัณฑ์และบริการใหม่ๆ ที่ตอบสนองความต้องการของผู้บริโภคและองค์กรธุรกิจ
- การเขียนโปรแกรมช่วยพัฒนาทักษะการแก้ปัญหาอย่างเป็นระบบและมีเหตุผล
- โปรแกรมคอมพิวเตอร์สามารถใช้ในการวิเคราะห์ข้อมูลและช่วยในการตัดสินใจอย่างมีประสิทธิภาพ
- ความสามารถในการเขียนโปรแกรมเปิดโอกาสทางอาชีพในสาขาต่างๆ เช่น นักพัฒนาโปรแกรมนักวิเคราะห์ข้อมูล และนักวิทยาศาสตร์คอมพิวเตอร์
- ทักษะการเขียนโปรแกรมเป็นที่ต้องการสูงในตลาดแรงงานปัจจุบัน
- ฮาร์ดแวร์ หน่วยประมวลผลกลาง (CPU), หน่วยความจำ (Memory), อุปกรณ์อินพุตและเอาต์พุต
- ซอฟต์แวร์ ระบบปฏิบัติการ (Operating System) และซอฟต์แวร์ประยุกต์ (Application Software)
- ระบบคอมพิวเตอร์ประกอบด้วยหน่วยรับข้อมูล หน่วยประมวลผล หน่วยความจำ และหน่วยแสดงผล
- ข้อมูลถูกส่งจากหน่วยรับข้อมูลไปยังหน่วยความจำ และถูกประมวลผลโดยหน่วยประมวลผลก่อนส่งไปยังหน่วยแสดงผล
- ระบบปฏิบัติการจัดการทรัพยากรของคอมพิวเตอร์ เช่น CPU, หน่วยความจำ และอุปกรณ์จัดเก็บข้อมูล
- ระบบปฏิบัติการมีบทบาทในการจัดการไฟล์และข้อมูล การจัดการกระบวนการ และการรักษาความปลอดภัย

- ภาษาคอมพิวเตอร์สามารถแบ่งได้เป็น 5 ยุค: ภาษาเครื่อง, ภาษาแอสเซมบลี, ภาษาระดับสูง, ภาษาระดับสูงมาก, และภาษาธรรมชาติ
- แต่ละยุค มีการพัฒนาทางด้านความสะดวกในการเขียนโปรแกรมและการใช้งานที่หลากหลายขึ้น
- Python เป็นภาษาที่ง่ายต่อการเรียนรู้และใช้งาน มีโครงสร้างที่อ่านง่ายและเข้าใจได้ง่าย
- Python มีไลบรารีและเครื่องมือที่หลากหลาย ทำให้สามารถนำไปประยุกต์ใช้ในหลายสาขา เช่น การพัฒนาเว็บ การวิเคราะห์ข้อมูล และการพัฒนา AI
- Python ใช้ในการพัฒนาเว็บแอปพลิเคชัน การพัฒนาแอปพลิเคชัน GUI บนเดสก์ท็อป และการคำนวณทางวิทยาศาสตร์
- Python ยังใช้ในการพัฒนาซอฟต์แวร์ การศึกษา การเข้าถึงฐานข้อมูล การเขียนโปรแกรมเครือข่าย และการพัฒนาเกมและกราฟิก 3D
- Python มีไลบรารีที่มีประสิทธิภาพ เช่น NumPy, Pandas, Matplotlib สำหรับการวิเคราะห์ข้อมูล
- Django และ Flask สำหรับการพัฒนาเว็บ, TensorFlow และ PyTorch สำหรับการพัฒนา AI และการเรียนรู้ของเครื่อง (Machine Learning)

คำถามก้ายบท

1. อธิบายความสำคัญของการเขียนโปรแกรมคอมพิวเตอร์ในยุคปัจจุบัน
2. การเขียนโปรแกรมสามารถเปิดโอกาสทางอาชีพใดได้บ้าง? ยกตัวอย่างอย่างน้อย 3 อาชีพ
3. หน่วยประมวลผลกลาง (CPU) มีหน้าที่อะไรในระบบคอมพิวเตอร์?
4. ระบบปฏิบัติการมีบทบาทอย่างไรในการจัดการทรัพยากรของคอมพิวเตอร์?
5. ระบบปฏิบัติการจัดการความปลอดภัยของข้อมูลในระบบคอมพิวเตอร์อย่างไร?
6. อธิบายการทำงานของหน่วยรับข้อมูล (Input Unit) และหน่วยแสดงผล (Output Unit)
7. อธิบายความแตกต่างระหว่างภาษาแอสเซมบลี (Assembly Language) กับภาษาเครื่อง (Machine Language)
8. ภาษาระดับสูง (High-level Language) มีข้อดีอย่างไรในการพัฒนาโปรแกรม?
9. Python มีการประยุกต์ใช้งานในสาขาใดบ้าง? ยกตัวอย่างอย่างน้อย 3 สาขา
10. ไลบรารีเช่น NumPy และ Pandas ใน Python ใช้สำหรับทำอะไร?

บทที่ 2

การเขียนโปรแกรมเบื้องต้นด้วยภาษา Python

เนื้อหาบทเรียน

- 2.1 โปรแกรมสำหรับพัฒนาโปรแกรม (IDE)
- 2.2 การเริ่มต้นเขียนโปรแกรม
- 2.3 ตัวแปรและชนิดข้อมูล
- 2.4 การโปรแกรมการคำนวณทางคณิตศาสตร์
- 2.5 ลิสต์
- 2.6 ติกชันนารี
- 2.7 ความผิดพลาดของโปรแกรม
- 2.8 การเขียนโปรแกรมในกระบวนการคำนวณ

วัตถุประสงค์การเรียนรู้

- ติดตั้งโปรแกรมสำหรับพัฒนาโปรแกรมได้ (IDE)
- เขียนโปรแกรมให้แสดงผลข้อความเบื้องต้นได้
- บอกข้อแตกต่างของชนิดข้อมูลต่างๆ ในภาษา Python ได้
- เขียนโปรแกรมการคำนวณทางคณิตศาสตร์ได้
- เข้าใจและเขียนโปรแกรมใช้งานชนิดข้อมูลแบบลิสต์ได้
- เข้าใจและเขียนโปรแกรมใช้งานชนิดติกชันนารีได้
- บอกระยะของความผิดพลาดของโปรแกรมได้
- เขียนโปรแกรมควบกระบวนการคำนวณ ได้แก่ รับอินพุต ประมวลผลและแสดงผลเอาต์พุต

ยุคในโลกยุคดิจิทัลที่เทคโนโลยีเข้ามามีบทบาทในชีวิตประจำวัน การเขียนโปรแกรมคอมพิวเตอร์กลายเป็นหักษะที่มีความสำคัญอย่างยิ่ง ไม่ว่าจะเป็นนักเรียน นักพัฒนา หรือผู้ที่ต้องการพัฒนาหักษะเพื่อก้าวเข้าสู่สายงานเทคโนโลยี การเขียนโปรแกรมเบื้องต้นด้วยภาษา Python ถือเป็นจุดเริ่มต้นที่ดี เนื่องจาก Python เป็นภาษาที่เรียนรู้ง่าย มีโครงสร้างง่ายขับ และถูกใช้ในงานหลากหลาย เช่น การพัฒนาแอปพลิเคชัน การวิเคราะห์ข้อมูล และปัญญาประดิษฐ์ (AI) บทเรียนนี้ประกอบด้วยของการเขียนโปรแกรม ตั้งแต่การติดตั้งเครื่องมือสำหรับพัฒนาโปรแกรม (IDE) การเริ่มต้นเขียนโค้ด Python อย่างง่าย ไปจนถึงการจัดการข้อมูล ประเภทต่าง ๆ เช่น ลิสต์ (List) และดิกชันนารี (Dictionary) รวมถึงการแก้ไขข้อผิดพลาดที่อาจเกิดขึ้น เพื่อสร้างพื้นฐานที่แข็งแรงในการพัฒนาหักษะการเขียนโปรแกรมที่ซับซ้อนในอนาคต

การเรียนรู้ Python ไม่เพียงแต่ช่วยพัฒนาหักษะด้านการเขียนโปรแกรม แต่ยังเปิดประตูสู่โอกาสในสายงานเทคโนโลยี และช่วยให้สามารถสร้างสรรค์นวัตกรรมใหม่ ๆ ได้ด้วยตัวเอง

2.1 โปรแกรมสำหรับพัฒนาโปรแกรม (IDE)

การพัฒนาโปรแกรมคอมพิวเตอร์ในปัจจุบันเป็นเรื่องที่สำคัญและซับซ้อน ภาษา Python เป็นหนึ่งในภาษาที่ได้รับความนิยมในการพัฒนาโปรแกรมเหล่านี้ การที่จะสร้างโปรแกรม Python อย่างมีประสิทธิภาพนั้น ต้องอาศัยเครื่องมือที่เรียกว่า Integrated Development Environment (IDE) โดยโปรแกรมพัฒนาโปรแกรม (IDE) คือ เครื่องมือที่รวมฟังก์ชันหลายอย่างที่จำเป็นสำหรับการพัฒนาโปรแกรมไว้ในที่เดียว โดยทั่วไป IDE จะมีเครื่องมือหลัก เช่น Text Editor สำหรับเขียนและแก้ไขโค้ด Debugger สำหรับตรวจสอบ และแก้ไขข้อผิดพลาดในโค้ด Compiler/Interpreter สำหรับแปลงโค้ดให้เป็นโปรแกรมที่สามารถรันได้ และ Version Control สำหรับจัดการการเปลี่ยนแปลงของโค้ด

ความสำคัญของ IDE คือช่วยเพิ่มประสิทธิภาพในการพัฒนาโปรแกรม ช่วยให้นักพัฒนาสามารถเขียนโค้ดได้เร็วขึ้นและมีประสิทธิภาพมากขึ้น เนื่องจากมีเครื่องมือที่ช่วยในการเขียนและตรวจสอบโค้ด ลดข้อผิดพลาดด้วยการมี Debugger และ Syntax Highlighting ที่ช่วยตรวจพบข้อผิดพลาดได้รวดเร็ว จัดการโปรเจกต์ได้ง่ายขึ้นด้วยเครื่องมือที่ช่วยในการจัดการไฟล์และโฟลเดอร์ต่างๆ ของโปรเจกต์ และรองรับการทำงานร่วมกันของนักพัฒนาหลายคนในโปรเจกต์เดียวกัน ทำให้การพัฒนาโปรแกรมเป็นไปอย่างราบรื่นและมีประสิทธิภาพมากขึ้น

สำหรับ IDE ของภาษา Python มีตัวเลือกหลายตัวที่ได้รับความนิยม เช่น PyCharm, Visual Studio Code, และ Jupyter Notebook แต่ละตัวมีคุณสมบัติและความสามารถที่ต่างกันไป ดังนี้

1. IDLE ย่อมาจาก "Integrated Development and Learning Environment" เป็นเครื่องมือที่มาพร้อมกับการติดตั้ง Python ทำให้การเริ่มต้นพัฒนาโปรแกรมด้วยภาษา Python เป็นเรื่องง่ายและสะดวก สำหรับนักพัฒนาทุกระดับ ประกอบด้วย 1) Python Shell สำหรับรันโค้ด Python แบบทันทีและทดสอบโค้ด ในขณะพัฒนา 2) Editor สำหรับเขียนและแก้ไขโค้ด Python มี Syntax Highlighting ที่ช่วยให้โค้ดอ่านง่าย

และตรวจสอบข้อผิดพลาดเบื้องต้นได้ และ 3) Debugger สำหรับตรวจสอบและแก้ไขข้อผิดพลาดในโค้ด โดยสามารถตั้ง Breakpoints และตรวจสอบค่าในตัวแปรต่างๆ ได้

2. PyCharm เป็น IDE ที่พัฒนาโดย JetBrains และได้รับความนิยมมากสำหรับการพัฒนาโปรแกรมในภาษา Python มีฟีเจอร์ครบครันทั้ง Text Editor, Debugger, และ Version Control นอกจากนี้ยังมีฟีเจอร์พิเศษสำหรับการพัฒนาเว็บแอปพลิเคชัน เช่น Django และ Flask

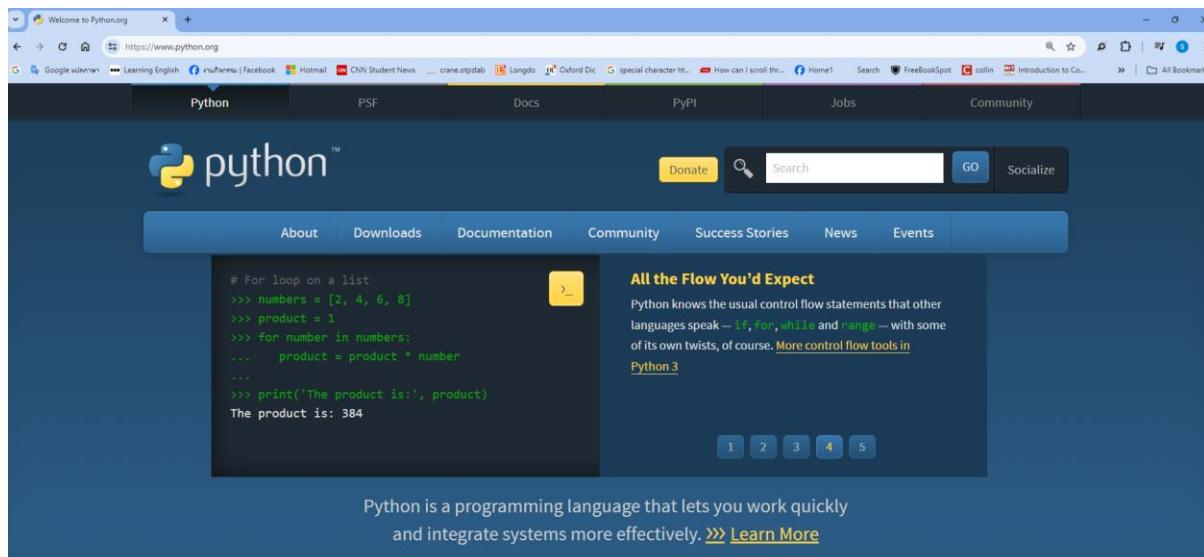
3. Visual Studio Code (VS Code) เป็น IDE ที่พัฒนาโดย Microsoft และรองรับหลายภาษา รวมถึง Python VS Code มีฟีเจอร์มากมาย เช่น IntelliSense ที่ช่วยในการเติมโค้ดอัตโนมัติ, Debugger, และ Integration กับ Git สำหรับ Version Control

4. Jupyter Notebook เป็นเครื่องมือที่เน้นการใช้งานในการวิเคราะห์ข้อมูลและการวิจัยทางวิทยาศาสตร์ โดยมีรูปแบบการทำงานเป็นโน็ตบุ๊กที่สามารถรันโค้ดที่ลีส่วนและแสดงผลลัพธ์ได้ทันที

การใช้งาน IDE สำหรับ Python ช่วยให้การพัฒนาโปรแกรมเป็นไปอย่างรวดเร็วและมีประสิทธิภาพ เนื่องจากเครื่องมือต่างๆ ที่มีใน IDE ช่วยให้นักพัฒนาสามารถเขียน ตรวจสอบ และแก้ไขโค้ดได้อย่างสะดวก และรวดเร็ว นอกจากนี้ การใช้ IDE ยังช่วยลดข้อผิดพลาดและเพิ่มความมั่นใจในการพัฒนาโปรแกรมอีกด้วย โดยหัวข้อลำดับต่อไปจะเป็นการติดตั้งเครื่องมือที่ใช้ในการเรียนในเอกสารประกอบการสอนนี้ ได้แก่ การติดตั้ง โปรแกรมภาษา Python การติดตั้ง Visual Studio Code การติดตั้ง Jupyter Notebook และการเข้าใช้งาน Google Colab

2.1.1 การติดตั้งโปรแกรมภาษา Python

การดาวน์โหลดและติดตั้งโปรแกรมภาษา Python สามารถทำได้ตามขั้นตอนดังนี้



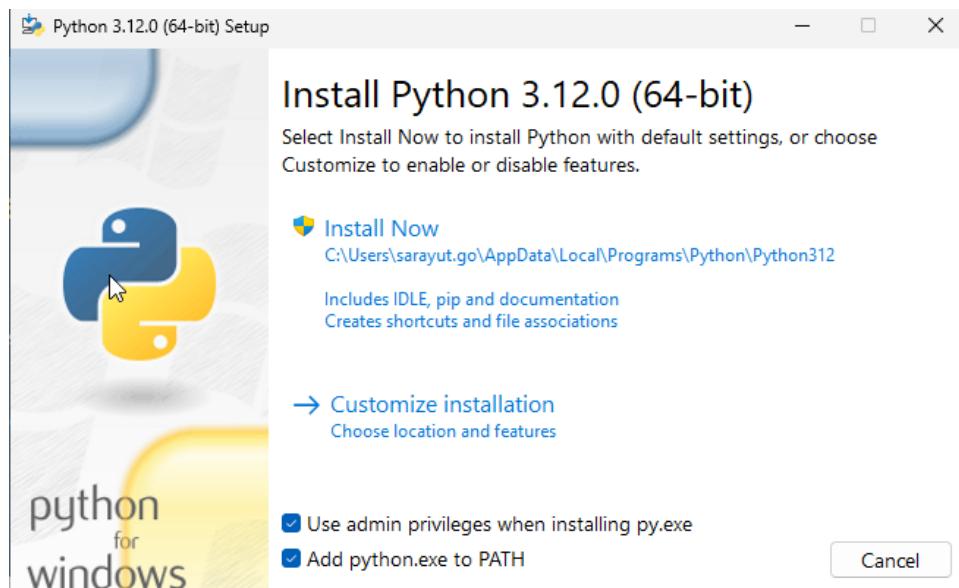
รูปที่ 2.1 หน้าเว็บไซต์ภาษา Python

แหล่งที่มารูป: <http://www.python.org>

1. การดาวน์โหลดภาษา Python ไปที่เว็บไซต์ทางการของ Python เปิดเบราว์เซอร์และไปที่เว็บไซต์ <https://www.python.org> ดังแสดงในรูปที่ 2.1 เลือกแท็บ "Downloads" คลิกที่แท็บ "Downloads" ที่

ด้านบนของหน้าเว็บเลือกเวอร์ชันที่ต้องการ โดยทั่วไปเว็บไซต์จะตรวจสอบระบบปฏิบัติการโดยอัตโนมัติและเสนอเวอร์ชันล่าสุดที่เหมาะสม คลิกที่ปุ่มดาวน์โหลด (เช่น "Download Python 3.12" เป็นเวอร์ชันที่ใช้ในเอกสารประกอบการสอนนี้)

2. การติดตั้งภาษา Python เมื่อดาวน์โหลดไฟล์เสร็จสิ้น ให้เปิดไฟล์ที่ดาวน์โหลดมาในหน้าจอการติดตั้ง เลือกตัวเลือก "Add Python to PATH" ที่มุมล่างซ้ายของหน้าจอ ดังแสดงในรูปที่ 2.2 เพื่อให้สามารถเรียกใช้ Python จาก Command Line ได้เลือก "Install Now" คลิกที่ปุ่ม "Install Now" เพื่อเริ่มการติดตั้ง



รูปที่ 2.2 หน้าต่างติดตั้งโปรแกรมภาษา Python

3. การตรวจสอบการติดตั้ง เปิด Command Prompt (cmd) หรือ Terminal: ขึ้นอยู่กับระบบปฏิบัติการ พิมพ์คำสั่งตรวจสอบ โดยพิมพ์ python --version หรือ python3 --version แล้วกด Enter หากติดตั้งสำเร็จ จะเห็นเวอร์ชันของ Python ที่ติดตั้งแสดงขึ้นมา ดังแสดงในรูปที่ 2.3 แสดงลำดับการทดสอบ การติดตั้ง

```
C:\Users\baban>python --version
Python 3.12.1

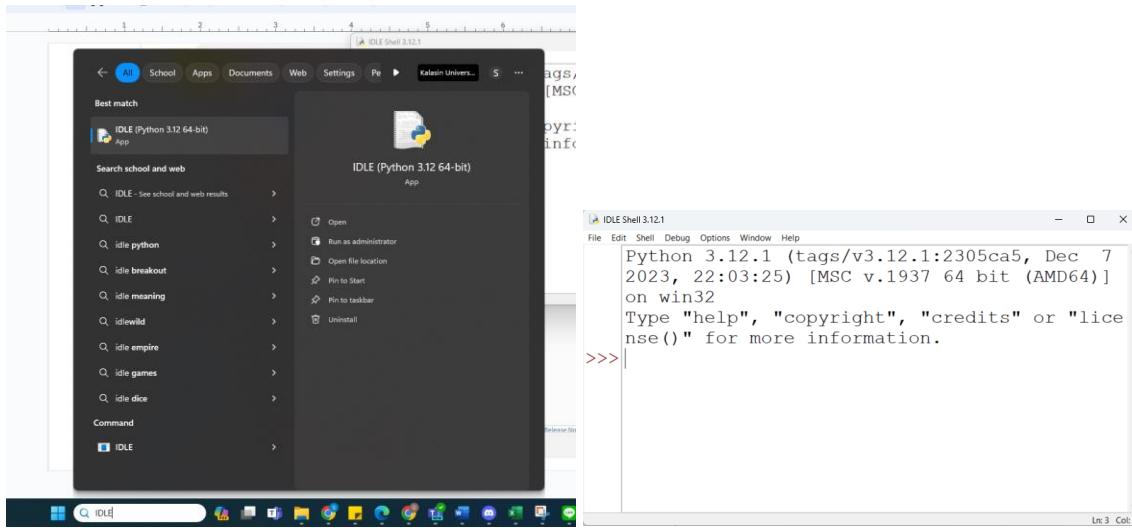
C:\Users\baban>python
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\baban>
```

รูปที่ 2.3 แสดงการทดสอบการติดตั้งบน Command Prompt

4. การใช้ IDLE โดยเปิด IDLE หลังจากติดตั้งเสร็จสิ้น จะสามารถเปิดโปรแกรม IDLE ที่ติดตั้งมาพร้อมกับ Python ได้ โดยค้นหา "IDLE" ในเมนู Start (Windows) หรือในโฟลเดอร์ Applications (macOS) ใช้

IDLE เพื่อเขียนและรันโค้ด Python ได้ทันที ดังแสดงในรูปที่ 2.4 ที่ทำการค้นหา IDLE หลังจากกดเปิด IDLE จะปรากฏหน้าจอ IDLE Shell 3.12.1

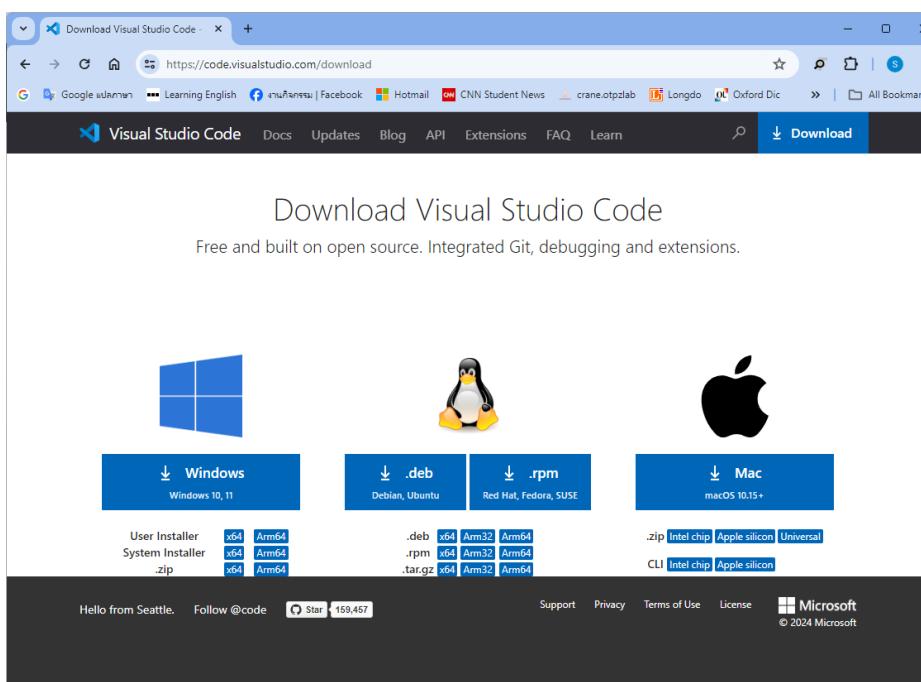


รูปที่ 2.4 แสดงการเปิดใช้ IDLE

2.1.2 การติดตั้งโปรแกรมภาษา Python

การติดตั้ง Visual Studio Code (VS Code) สามารถทำได้ตามขั้นตอนดังนี้:

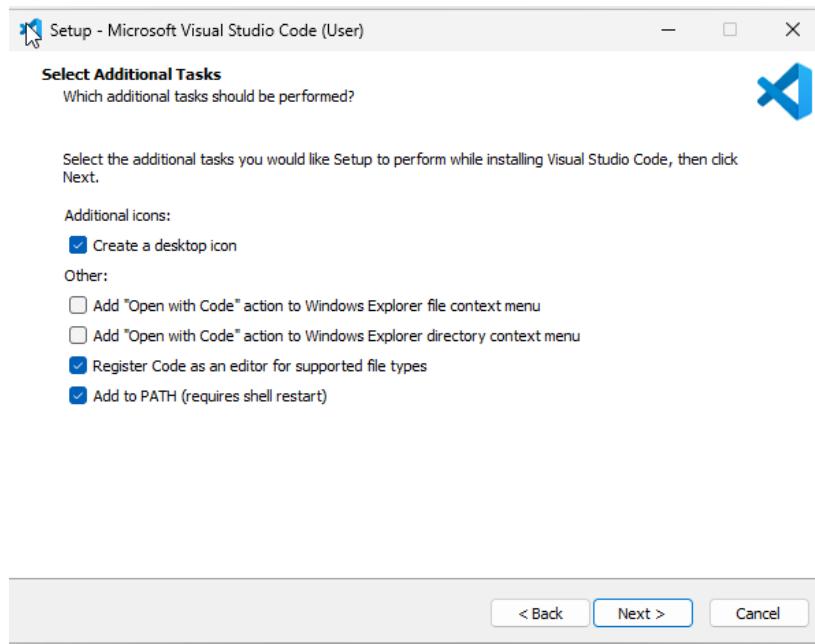
1. การดาวน์โหลด Visual Studio Code ไปที่เว็บไซต์ทางการของ Visual Studio Code เปิดเบราว์เซอร์และไปที่เว็บไซต์ <https://code.visualstudio.com/download>
2. เลือกระบบปฏิบัติการ: เลือกเวอร์ชันที่เหมาะสมกับระบบปฏิบัติการ (Windows, macOS, หรือ Linux) โดยคลิกที่ปุ่มดาวน์โหลดที่สอดคล้องดังแสดงในรูปที่ 2.5



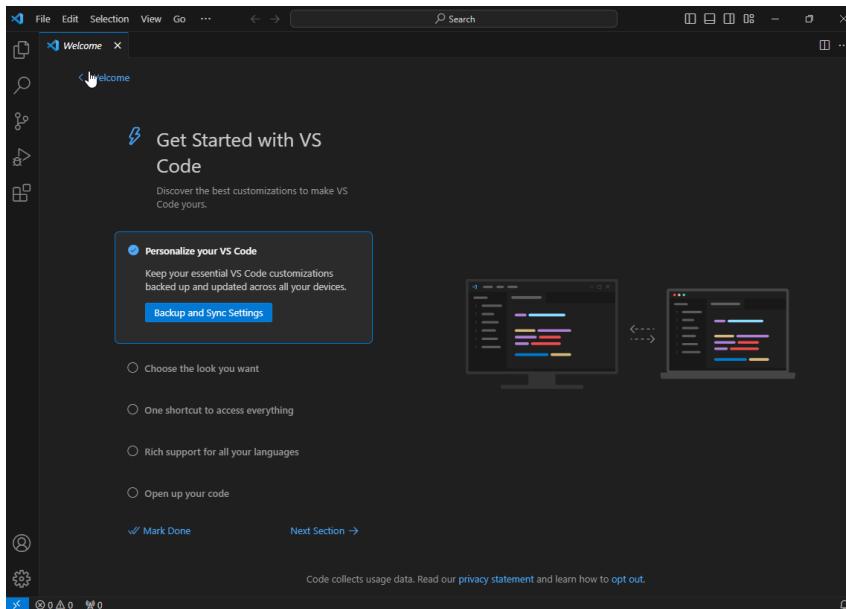
รูปที่ 2.5 แสดงหน้าต่างเว็บไซต์ที่ใช้ดาวน์โหลด VS Code

3. การติดตั้ง Visual Studio Code สำหรับ Windows

- เปิดไฟล์ติดตั้ง เมื่อดาวน์โหลดไฟล์เสร็จสิ้น ให้ดับเบิลคลิกที่ไฟล์ติดตั้ง (.exe) จะเกิดหน้าต่างเพื่อติดตั้งดังรูปที่ 2.6
- ยอมรับข้อตกลง ในหน้าต่างการติดตั้ง อ่านและยอมรับข้อตกลงในการใช้งานโดยเลือก "I accept the agreement" และคลิก "Next"
- เลือกไฟล์เดอร์การติดตั้ง เลือกไฟล์เดอร์ที่ต้องการติดตั้งโปรแกรม (หรือใช้ค่า默認) และคลิก "Next"
- ตั้งค่าเพิ่มเติม เลือกตัวเลือกที่ต้องการ เช่น การสร้าง Shortcut บน Desktop และการตั้งค่าให้สามารถเรียกใช้ VS Code จาก Command Line และคลิก "Next"
- ติดตั้งโปรแกรม คลิก "Install" เพื่อเริ่มการติดตั้ง
- เสร็จสิ้นการติดตั้ง: หลังจากการติดตั้งเสร็จสิ้น คลิก "Finish" เพื่อปิดหน้าต่างการติดตั้ง และเลือก "Launch Visual Studio Code" เพื่อเปิดโปรแกรม ดังแสดงรูปที่ 2.7



รูปที่ 2.6 แสดงหน้าต่างการติดตั้ง VS Code



รูปที่ 2.7 หน้าโปรแกรม VS Code

2.1.3 การติดตั้ง Jupyter Notebook

การติดตั้ง Jupyter สามารถติดตั้งผ่าน Anaconda ซึ่งเป็นแพลตฟอร์มสำหรับการพัฒนาและการจัดการข้อมูลขนาดใหญ่ โดยเฉพาะสำหรับงานด้านวิทยาศาสตร์ข้อมูล (Data Science) และการเรียนรู้ของเครื่อง (Machine Learning) Anaconda มาพร้อมกับเครื่องมือและแพ็คเกจต่างๆ ที่จำเป็นสำหรับการพัฒนาและวิเคราะห์ข้อมูล หรือการติดตั้งผ่าน ซึ่งเป็นตัวจัดการแพ็คเกจของ Python (pip) โดยในเอกสารนี้จะใช้รูปแบบ pip โดยมีรายละเอียดขั้นตอนดังนี้

1. ติดตั้ง Python หากยังไม่ได้ติดตั้ง Python ให้ติดตั้งก่อน ในหัวข้อ 2.1.1
2. ติดตั้ง pip โดยทั่วไป pip จะติดตั้งมาพร้อมกับ Python แล้ว แต่ถ้ายังไม่มี ให้ติดตั้งด้วยคำสั่ง

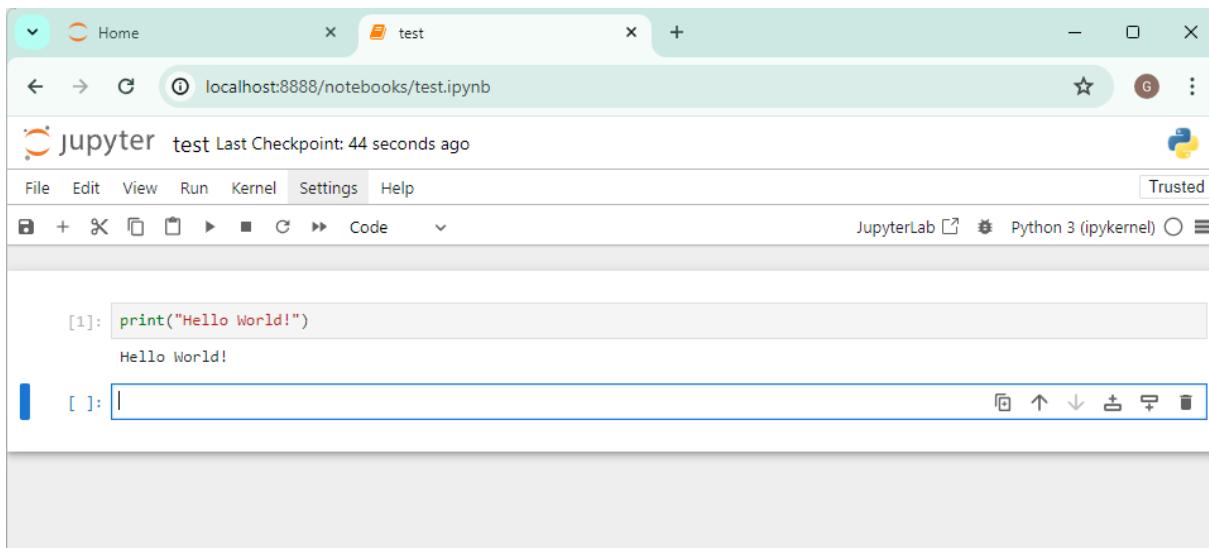
```
python -m ensurepip --upgrade
```

3. ติดตั้ง Jupyter Notebook เปิด Command Prompt (Windows) หรือ Terminal (macOS/Linux) และรันคำสั่งต่อไปนี้

```
pip install notebook
```

4. เปิด Jupyter Notebook หลังจากติดตั้งเสร็จสิ้น ให้รันคำสั่งนี้ใน Command Prompt หรือ Terminal เพื่อเปิด Jupyter Notebook ดังรูปที่ 2.8 แสดงหน้าจอโปรแกรม Jupyter Notebook

```
jupyter notebook
```



รูปที่ 2.8 หน้าโปรแกรม Jupyter Notebook

2.1.4 การเข้าใช้งาน Google Colab

อีกหนึ่งเครื่องมือของการเขียนโปรแกรมที่บริษัท Google ได้พัฒนาขึ้นมาซึ่งมีชื่อว่า Google Collaboratory หรือเรียกว่า Google Colab พัฒนามาบนพื้นฐานเดียวกัน โดย Google Colab ทำงานบนคลาวด์ของ Google และมีความสามารถในการเข้าถึงทรัพยากร เช่น GPU และ TPU ได้ฟรี ทั้งสองสามารถรันโค้ด Python และจัดการเซลล์โค้ดและ Markdown ได้เหมือนกับ Google Colab บันทึกไฟล์บน Google Drive ทำให้แชร์และเข้าถึงไฟล์ง่ายขึ้น ขณะที่ Jupyter Notebook ทำงานแบบออนไลน์และให้การควบคุมทรัพยากรที่มากกว่า โดยมีขั้นตอนการใช้งานและการเข้าใช้งาน Google Colab ดังนี้

1. เข้าเว็บไซต์ Google Colab เปิดเว็บเบราว์เซอร์ที่ใช้ พิมพ์ URL <https://colab.research.google.com> แล้วกด Enter
2. เข้าสู่ระบบด้วยบัญชี Google หากยังไม่ได้เข้าสู่ระบบ Google จะขอให้ทำการลงชื่อเข้าใช้ด้วยบัญชี Google กรอกอีเมลและรหัสผ่านของคุณ จากนั้นคลิกที่ "ติดไป" เพื่อเข้าสู่ระบบ
3. สร้างโน๊ตบุ๊กใหม่ เมื่อเข้าสู่หน้า Google Colab แล้ว ให้คลิกที่ "File" (ไฟล์) บนแถบเมนู เลือก "New notebook" (โน๊ตบุ๊กใหม่) ระบบจะสร้างโน๊ตบุ๊กใหม่ให้ใช้งาน
4. ตั้งชื่อโน๊ตบุ๊ก คลิกที่ "Untitled" (ไม่มีชื่อ) ที่มุมซ้ายบนของหน้าจอ พิมพ์ชื่อที่ต้องการตั้งให้กับโน๊ตบุ๊ก เช่น “01 My first program” แล้วกด Enter
5. การใช้งานพื้นฐานของ Google Colab ใน Google Colab สามารถเขียนโค้ดด้วยภาษา Python ได้ในเซลล์ (Cells) คลิกที่เซลล์แล้วพิมพ์โค้ด เช่น `print("Hello, Google Colab!")` กด Shift + Enter เพื่อรันโค้ดในเซลล์นั้น เมื่อดำเนินการจากข้อ 1-5 จะสามารถรันภาษา Python ได้ดังรูปที่ 2.9

6. การเพิ่มเซลล์ใหม่ สามารถเพิ่มเซลล์ใหม่ได้โดยคลิกที่ปุ่ม +Code หรือ +Text ที่อยู่ด้านบนของโน๊ตบุ๊ก

7. การจัดการและบันทึกไฟล์ Google Colab จะบันทึกโน๊ตบุ๊กอัตโนมัติใน Google Drive

8. สามารถดาวน์โหลดโน๊ตบุ๊กเป็นไฟล์ .ipynb หรือ .py โดยไปที่ "File" > "Download .ipynb" หรือ "Download .py"

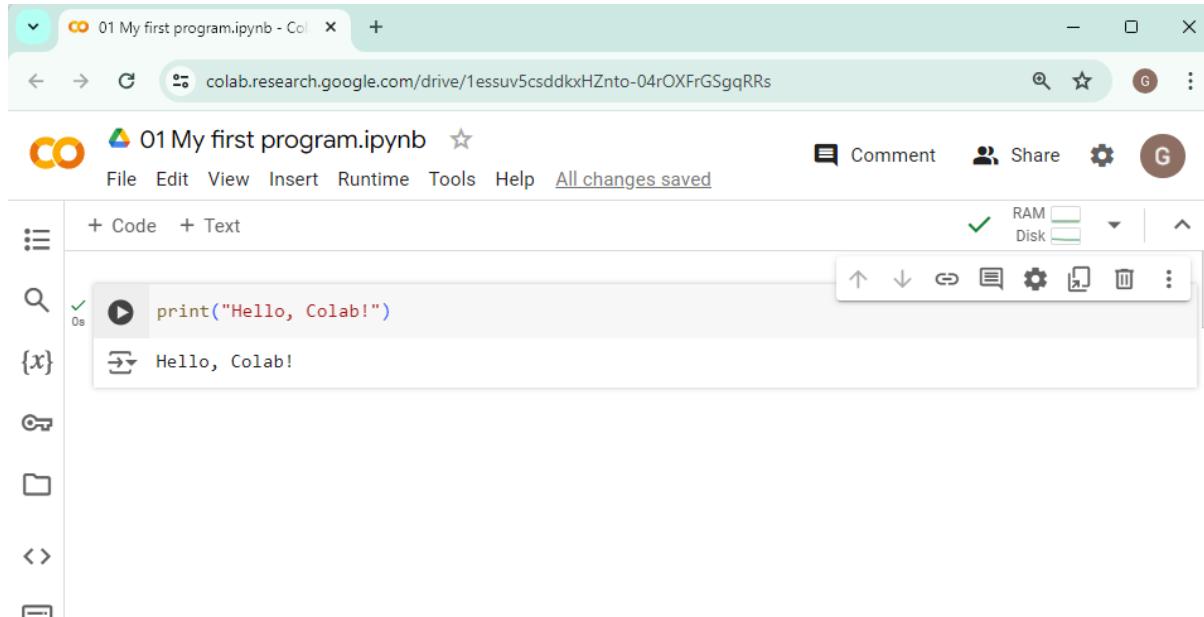
9. การติดตั้งและใช้งานแพ็คเกจเพิ่มเติม สามารถติดตั้งแพ็คเกจเพิ่มเติมได้โดยใช้คำสั่ง !pip install [ชื่อแพ็คเกจ] เช่น !pip install numpy หลังจากติดตั้งเสร็จสิ้น สามารถใช้งานแพ็คเกจนั้นๆ ได้ทันที

10. การเชื่อมต่อกับ Google Drive สามารถเชื่อมต่อ Google Colab กับ Google Drive เพื่อเข้าถึงไฟล์ต่างๆ โดยใช้คำสั่ง

```
from google.colab import drive
drive.mount('/content/drive')
```

เมื่อรันคำสั่งนี้ ระบบจะขออนุญาตเข้าถึง Google Drive ให้ออนุญาตการเข้าถึง

11. การแชร์โน๊ตบุ๊ก สามารถแชร์โน๊ตบุ๊กับผู้อื่นได้โดยคลิกที่ปุ่ม "Share" (แชร์) ที่มุมขวาบนของหน้าจอ กำหนดสิทธิ์การเข้าถึงและส่งลิงก์ให้กับผู้ที่ต้องการแชร์ด้วย



รูปที่ 2.9 หน้าต่างงานใช้งาน Google Colab

IDE เป็นเครื่องมือที่ช่วยให้กระบวนการเขียนโปรแกรมเป็นไปอย่างรวดเร็วและง่ายดายยิ่งขึ้นสำหรับนักพัฒนา ด้วยความสามารถในการตรวจสอบโค้ดแบบเรียลไทม์ การแก้ไขข้อผิดพลาด (Debugging) การ

จัดการโปรเจกต์ และการควบคุมเวอร์ชันของโค้ด IDE เปรียบเสมือนเพื่อนคู่คิดที่ช่วยลดภาระและเพิ่มประสิทธิภาพในการทำงานของโปรแกรมเมอร์ โดยเฉพาะอย่างยิ่งในยุคปัจจุบันที่การทำงานร่วมกันเป็นทีมบนแพลตฟอร์มต่างๆ เช่น Git หรือ Cloud กลายเป็นเรื่องปกติ

2.2 การเริ่มต้นเขียนโปรแกรม

โดยทั่วไปนี้จะให้ลองใช้เครื่องมือในหัวข้อที่ 2.1 ซึ่งการเขียนโปรแกรมแรกที่นิยมเขียนเมื่อเริ่มต้นเรียนรู้การเขียนโปรแกรมคือโปรแกรมที่แสดงข้อความ "Hello, World!" บนหน้าจอ ซึ่งเป็นโปรแกรมง่ายๆ ที่ช่วยให้ผู้เรียนรู้จักกับการพิมพ์โค้ด การรันโปรแกรม และการเห็นผลลัพธ์ที่ชัดเจน

การเริ่มต้นเขียนโปรแกรม Python มีหลายวิธีที่สะดวกและมีประสิทธิภาพ ทั้งการใช้ IDLE Shell สำหรับการทดลองโค้ดแบบรวดเร็ว การเขียนโปรแกรมเป็นไฟล์ใน IDLE สำหรับโครงการขนาดใหญ่ และการใช้ Visual Studio Code (VS Code) ที่มีฟีเจอร์ขั้นสูงและการจัดการโค้ดที่ครอบคลุม ในแต่ละวิธีมีขั้นตอน และข้อดีเฉพาะที่เหมาะสมกับความต้องการที่แตกต่างกันของผู้ใช้ ในหัวข้อนี้จะแสดงขั้นตอนในวิธีดังต่อไปนี้

2.2.1 การเขียนโปรแกรมด้วย IDLE Shell

เปิดโปรแกรม IDLE ที่ติดตั้งมาพร้อมกับ Python ในหัวข้อที่ 2.1.1 ดังรูปที่ 2.4 จะเห็นหน้าจอ IDLE ซึ่งจะแสดงหน้าต่าง Shell (Interactive Interpreter) ขึ้นมา

ในหน้าต่าง Shell สามารถพิมพ์โค้ด Python และรันได้ทันที เช่น พิมพ์ `print("Hello, World!")` แล้วกด Enter

```
print("Hello, World!")
```

ผลลัพธ์จะแสดงข้อความ "Hello, World!" บนหน้าจอดังรูป 2.10

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello, World!")
Hello, World!
>>> 5 + 2
7
>>>
```

รูปที่ 2.10 หน้าจอ IDLE Shell พิมพ์โปรแกรมและผลลัพธ์

นอกจากนี้ ในรูปที่ 2.10 ได้ทำการพิมพ์การคำนวน $5 + 2$ ซึ่งจะเห็นโปรแกรม IDLE Shell ตอบกลับผลลัพธ์ 7 (ตัวอักษรสีน้ำเงิน) กลับคืนให้ผู้ใช้ได้เห็นผล

2.2.2 การเขียนโปรแกรมเป็นไฟล์ใน IDLE

อีกรูปแบบการใช้ IDLE คือ การเขียนโปรแกรมแบบสร้างไฟล์ การเขียนโปรแกรมแบบ IDLE Shell นั้น หมายความว่าการทดสอบโค้ดสั้นๆ และการเรียนรู้พื้นฐานเบื้องต้น ขณะที่การเขียนโปรแกรมแบบไฟล์ หมายความว่าการพัฒนาโปรแกรมที่มีขนาดใหญ่และซับซ้อนมากขึ้น เนื่องจากสามารถบันทึกและจัดการโค้ดได้อย่างมีประสิทธิภาพ

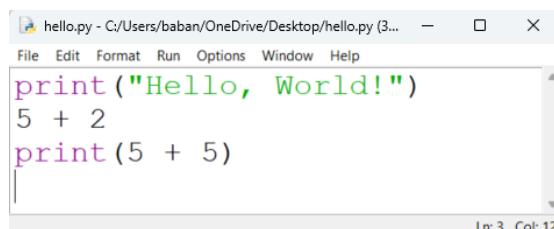
โดยการเขียนแบบไฟล์มีขั้นตอนดังนี้

- สร้างไฟล์ใหม่โดยไปที่เมนู File และเลือก New File เพื่อเปิดหน้าต่าง Editor ใหม่ จากนั้นเขียนโค้ด `print("Hello, World!")` และโค้ดเพิ่มเติมดังรูปที่ 2.11 ในหน้าต่าง Editor

- เมื่อเขียนโค้ดเสร็จแล้ว ไปที่เมนู File > Save หรือ Save As เพื่อบันทึกไฟล์ โดยตั้งชื่อไฟล์ให้มีนามสกุล .py เช่น `hello.py` จะเห็นผลลัพธ์ดังรูปที่ 2.11

- รันไฟล์โดยไปที่เมนู Run และเลือก Run Module หรือกด F5

- ผลลัพธ์จะแสดงข้อความ "Hello, World!" และ 10 ในหน้าต่าง Shell ของ IDLE ดังรูปที่ 2.12 ข้อสังเกตการเขียนแบบไฟล์จากโค้ดตัวอย่างมี 3 บรรทัด แต่ผลลัพธ์แสดงผลเพียง 2 บรรทัด คือ "Hello, World!" และ 10 นั้น ข้อแตกต่างระหว่างแบบไฟล์และ Shell คือ ถ้าไม่มีคำสั่งพิมพ์แบบไฟล์จะไม่ให้ผลลัพธ์ในบรรทัดที่ 5 + 2 ซึ่งแตกต่างจาก Shell ถ้าพิมพ์บรรทัดนี้จะแสดงผลลัพธ์ 8 อกบนหน้าจอ



```
hello.py - C:/Users/baban/OneDrive/Desktop/hello.py (3...)
```

```
File Edit Format Run Options Window Help
```

```
print("Hello, World!")
5 + 2
print(5 + 5)
```

Ln: 3 Col: 12

รูปที่ 2.11 หน้าต่าง Editor การเขียนโปรแกรมแบบไฟล์



```
IDLE Shell 3.12.1
```

```
File Edit Shell Debug Options Window Help
```

```
>>> print("Hello, World!")
Hello, World!
>>> 5 + 2
7
>>>
= RESTART: C:/Users/baban/OneDrive/Desktop/hello.py
Hello, World!
10
>>>
```

Ln: 11 Col: 0

รูปที่ 2.12 ผลลัพธ์บน IDLE Shell จากการรันโปรแกรม (กด F5) ในรูปที่ 2.11 บน

2.2.3 การเขียนโปรแกรมแบบ Visual Studio Code (VS Code)

หลังจากติดตั้ง VS Code ในหัวข้อที่ 2.1.2 ขั้นตอนการรันโปรแกรมด้วย VS Code มีดังนี้

- ตั้งค่า Python Extension ไปที่ Extensions (ไอคอนรูปสี่เหลี่ยมด้านขวาของหน้าจอ) และค้นหาและติดตั้ง Python extension จาก Microsoft ดังรูปที่ 2.13

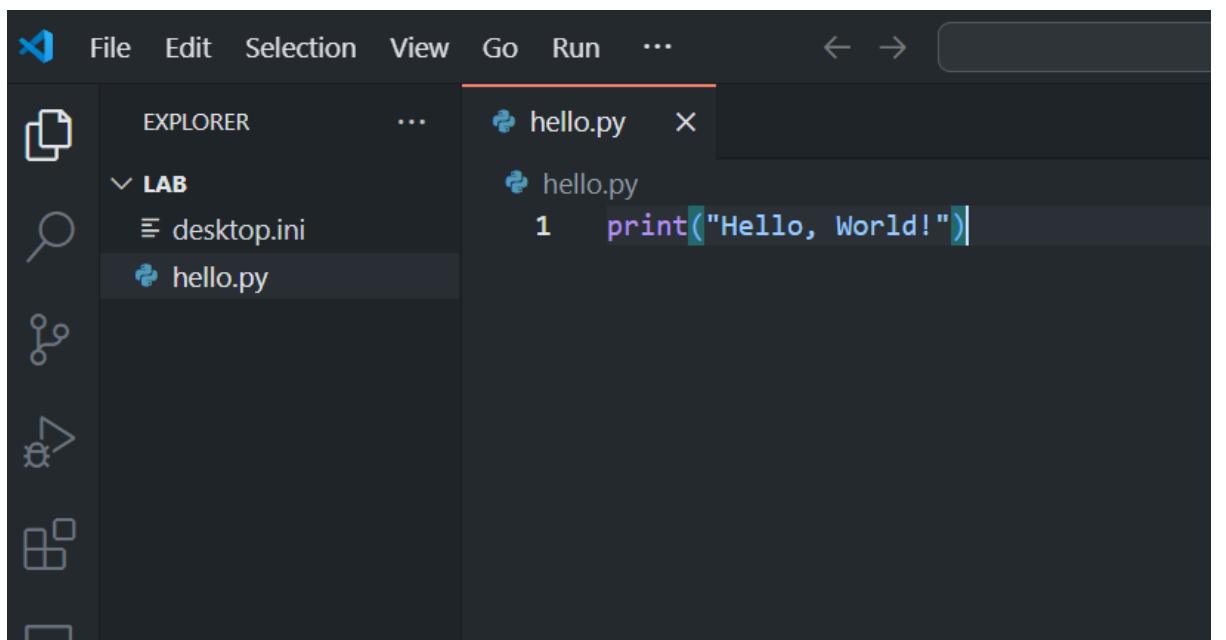


รูปที่ 2.13 การติดตั้ง Python Extension Microsoft ใน VS Code

2. สร้างไฟล์ใหม่ ไปที่เมนู File > New File แล้วพิมพ์โค้ด Python ที่ต้องการ เช่น print("Hello, World!")

3. บันทึกไฟล์ บันทึกไฟล์ด้วยนามสกุล .py เช่น hello.py

ดังรูปที่ 2.14



รูปที่ 2.14 หน้าจอโปรแกรมใน VS Code บันทึกไฟล์ hello.py

4. รันโค้ดใน VS Code เปิดไฟล์โค้ดที่เขียน แล้วคลิกขวาในพื้นที่ของไฟล์นั้น เลือก Run Python File in Terminal ผลลัพธ์จะแสดงใน Terminal ด้านล่างของหน้าต่าง VS Code ดังรูปที่ 2.15

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there's a folder named 'LAB' containing 'hello.py'. The code editor shows the same 'hello.py' file with the single-line print statement. To the right, the 'TERMINAL' tab is active, displaying the command-line output of running the script, which is 'Hello, World!'. A red arrow points from the caption text to this terminal output.

```
File Edit Selection View Go Run ... Lab
EXPLORER ... hello.py x
LAB
hello.py
1 print("Hello, World!")
TERMINAL
.
.
.
PS G:\My Drive\KSU\Lectures\2566-02\EN-01-002 Programming\Lab & C:/Programma
/a/Anaconda3/python.exe "g:/My Drive/KSU/Lectures/2566-02/EN-01-002 Programming
/Lab/hello.py"
Hello, World!
PS G:\My Drive\KSU\Lectures\2566-02\EN-01-002 Programming\Lab>
```

รูปที่ 2.15 แสดงขั้นตอนการรันโปรแกรม Python บน VS Code

2.2.4 การเขียนโปรแกรมแบบ Jupyter Notebook หรือ Google Colab

ดังได้อธิบายในหัวข้อ 2.1.3 และ 2.1.4 การติดตั้ง Jupyter Notebook และ Google Colab ตามลำดับ สามารถเพิ่มเซลล์ใหม่ คลิกที่ปุ่ม + ที่แถบเครื่องมือด้านบนเพื่อเพิ่มเซลล์ใหม่ เลือกประเภทเซลล์ระหว่าง Code (เขียนโค้ด) หรือ Markdown (เขียนข้อความ) และการจัดการเซลล์ โดย คลิกที่เซลล์ที่ต้องการจากนั้นใช้ปุ่ม Cut Copy Paste เพื่อจัดการเซลล์ ใช้ลูกศรขึ้นและลงเพื่อย้ายเซลล์ ดังแสดงการเขียนโค้ดและการรันเพิ่มเติมในรูปที่ 2.16

The screenshot shows a Google Colab notebook titled '01 My first program.ipynb'. The interface includes a toolbar with file operations like File, Edit, View, Insert, Runtime, Tools, Help, and a status bar indicating 'All changes saved'. The main area displays four cells. Cell 1 contains the code 'print("Hello, World!")' and outputs 'Hello, Colab!'. Cell 2 contains the code '5 + 2' and outputs '7'. Cell 3 contains the code 'print(5 + 5)' and outputs '10'. Cell 4 contains three lines of code: 'print("Hello, World!")', '5 + 2', and 'print(5 + 5)'. It has a play button icon and outputs 'Hello, World!', '10'.

รูปที่ 2.15 การใช้งาน Colab เพื่อรันโปรแกรม

จากการใช้เครื่องมือนี้ในการรันโปรแกรมแรกเบื้องต้นที่มีการพิมพ์ข้อความแสดงผล “Hello, World!” กับผลลัพธ์การคำนวณ

2.2.5 การเขียนโปรแกรมเบื้องต้นเพิ่มเติม

จากตัวอย่างในการทดสอบใช้เครื่องมือรันโปรแกรม “Hello, World” ในแต่ละโปรแกรม ต่อไปนี้เป็นตัวอย่างการพิมพ์เพิ่มเติมโดยใช้ `print()` และ IDLE Shell ดังรูปที่ 2.16

```

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, AMD64) [on win32]
Type "help", "copyright", "credits" or "license()"
>>> print("Hello, World!")
Hello, World!
>>> 5 + 2
7
>>> print(5+5)
10
>>> print(5*5)
25
>>> print(2567*1000)
2567000
>>> print(123456789*123456789)
15241578750190521
>>> print("dog")
dog
>>> print("dog"+"cat")
dogcat
>>> print("Hello "*5)
Hello Hello Hello Hello Hello
>>>

```

รูปที่ 2.16 โปรแกรมเพิ่มเติมใน IDLE Shell

อธิบายแต่ละบรรทัดย่อๆได้ดังนี้

`print(5+5)`

บรรทัดนี้จะคำนวณผลรวมของ $5 + 5$ (ซึ่งเท่ากับ 10) และแสดงผลลัพธ์ 10 ออกมาทางหน้าจอ

`print(5*5)`

บรรทัดนี้จะคำนวณผลคูณของ $5 * 5$ (ซึ่งเท่ากับ 25) และแสดงผลลัพธ์ 25 ออกมาทางหน้าจอ

`print(2567*1000)`

บรรทัดนี้จะคำนวณผลคูณของ $2567 * 1000$ (ซึ่งเท่ากับ 2567000) และแสดงผลลัพธ์ 2567000 ออกมาทางหน้าจอ

`print(123456789*123456789)`

บรรทัดนี้จะคำนวณผลคูณของ $123456789 * 123456789$ (ซึ่งเท่ากับ 15241578750190521) และแสดงผลลัพธ์ 15241578750190521 ออกมาทางหน้าจอ

```
print("dog")
```

บรรทัดนี้จะแสดงข้อความ "dog" ออกมาทางหน้าจอ

```
print("dog"+"cat")
```

บรรทัดนี้จะนำข้อความ "dog" และ "cat" มาต่อกัน (concatenate) แล้วแสดงผลลัพธ์ "dogcat" ออกมาทางหน้าจอ

```
print("Hello "*5)
```

บรรทัดนี้จะแสดงข้อความ "Hello " ซ้ำกัน 5 ครั้ง แล้วแสดงผลลัพธ์ "Hello Hello Hello Hello Hello " ออกมาทางหน้าจอ

2.2.6 ตัวอย่างการเขียนโปรแกรมเพื่อคำนวณ

ถ้าต้องการคำนวณจำนวนนาทีในหนึ่งสัปดาห์ โดยการคูณจำนวนวันในหนึ่งสัปดาห์ที่เท่ากับ 7 วัน จำนวนชั่วโมงในหนึ่งวันที่เท่ากับ 24 ชม. และจำนวนนาทีในหนึ่งชั่วโมงที่เท่ากับ 60 นาที สามารถคำนวณด้วยภาษา Python ได้ดังโค้ดนี้

```
print(7*24*60)
```

จะได้ออกputดังนี้

10080

2.2.7 ตัวอย่างการเขียนโปรแกรมเพื่อแสดงผลหลายบรรทัด

ถ้าต้องการแสดงผลออกบนหน้าจอเพื่อแนะนำตัวดังนี้ ที่ประกอบด้วย 3 บรรทัด

```
My name is Sarayut Gonwirat.
```

```
I was born November 9, 1986.
```

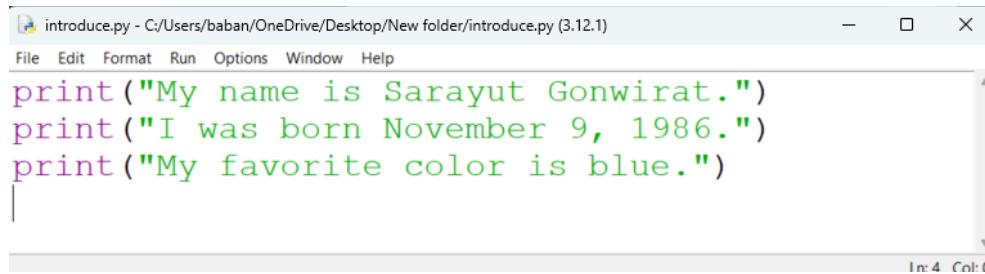
```
My favorite color is blue.
```

เพื่อที่จะให้ผลลัพธ์ออกมาเป็นหน้าจอ 3 บรรทัด ซึ่งก่อนหน้านี้ได้ใช้คำสั่งพิมพ์ print() ที่จะได้ผลลัพธ์หนึ่งบรรทัด ดังนั้นโค้ดที่ใช้ จึงเป็นดังนี้

โปรแกรมที่ 1 การพิมพ์เพื่อแนะนำตัวเอง (Introduce)

```
print("My name is Sarayut Gonwirat.")
print("I was born November 9, 1986.")
print("My favorite color is blue.")
```

โดยโปรแกรมแสดงโค้ดใน IDLE ดังรูปที่ 2.17 ที่ได้สร้างไฟล์ตามขั้นตอนที่ 2.2.2



```
introduce.py - C:/Users/baban/OneDrive/Desktop/New folder/introduce.py (3.12.1)
File Edit Format Run Options Window Help
print("My name is Sarayut Gonwirat.")
print("I was born November 9, 1986.")
print("My favorite color is blue.")

In: 4 Col: 0
```

รูปที่ 2.17 โปรแกรม introduce.py

"Hello, World!" ไม่ใช่แค่ประโยคธรรมชาติ แต่เป็นประตูบานแรกที่เปิดสู่โลกของการเขียนโปรแกรม การได้เห็นข้อความนี้ปรากฏหน้าจอเป็นครั้งแรกนั้นให้ความรู้สึกเหมือนได้ปลดล็อกศักยภาพในตัวเอง เพราะมันคือจุดเริ่มต้นของความสามารถในการสร้างสรรค์ซอฟต์แวร์ด้วยตนเอง และนั่นทำให้การเขียนโปรแกรมกลายเป็นเรื่องที่สนุกและท้าทายมากยิ่งขึ้น

2.3 ตัวแปรและชนิดข้อมูล

2.3.1 ตัวแปร

จากหัวข้อ 2.2 ที่ได้เรียนรู้การเขียนโปรแกรม Python เปื้องต้นเพื่อแสดงผลการคำนวณค่านาทีในหนึ่งสัปดาห์ และการทดสอบโปรแกรมต่างๆ นั้น จะเห็นได้ว่าการเขียนโปรแกรมมีความคล้ายคลึงกับการคำนวณทางคณิตศาสตร์ ซึ่งสามารถสร้างตัวแปร (Variable) เพื่อใช้แทนค่าโดยตรงได้ ยกตัวอย่างเช่น โปรแกรมในหัวข้อ 2.2.6 ที่ใช้คำนวณจำนวนนาที สามารถปรับปรุงโปรแกรมโดยใช้ตัวแปรที่มีชื่อมีความหมายแทนค่าต่างๆ เพื่อให้ผู้อ่านโปรแกรมเข้าใจการคำนวณได้ชัดเจนยิ่งขึ้น ทั้งนี้ ตัวแปร (Variable) ในการเขียนโปรแกรม คือสิ่งที่ใช้ในการเก็บค่าข้อมูลที่สามารถเปลี่ยนแปลงได้ในระหว่างการทำงานของโปรแกรม โดยตัวแปรมีชื่อเพื่อให้สามารถเรียกใช้งานได้ง่ายและเป็นการอ้างอิงค่าข้อมูลที่เก็บอยู่ในตัวแปรนั้น

โปรแกรมที่ 2 การคำนวณนาทีในหนึ่งสัปดาห์

```
days_per_week = 7
hours_per_day = 24
minutes_per_hour = 60
total_minutes = days_per_week * hours_per_day * minutes_per_hour
print("Total minutes in once week:")
print(total_minutes)
```

ผลลัพธ์

```
Total minutes in once week:
10080
```

ตัวแปร `days_per_week` เป็นชื่อที่มีความซัดเจนขึ้นของจำนวนวันในหนึ่งสัปดาห์ที่แทนด้วยค่า 7 แบบเดิม เช่นเดียวกับ `hours_per_day` และ `minutes_per_hour` โดยในบรรทัดต่อมาได้สร้างตัวแปรที่เกิดจากคำนวณจำนวนที่รวม

```
total_minutes = days_per_week * hours_per_day * minutes_per_hour
```

และเพื่อแสดงผลลัพธ์ได้ใช้สองบรรทัด `print()` ที่แสดงข้อความว่าโปรแกรมใช้คำนวนนาที และแสดงค่าของตัวแปรที่คำนวนนาทีรวมกอกมา ดังโค้ดด้านี้

```
print("Total minutes in once week:")
print(total_minutes)
```

2.3.2 ชนิดข้อมูล

ชนิดข้อมูล (Data Type) ชนิดข้อมูลคือประเภทของข้อมูลที่ตัวแพรสามารถเก็บได้ เช่น ตัวเลข, ข้อความ ค่าตระราก หรือกลุ่มของข้อมูล ชนิดข้อมูลช่วยกำหนดวิธีการจัดเก็บข้อมูลและการดำเนินการต่างๆ ที่สามารถทำได้กับข้อมูลนั้นๆ

ตารางที่ 2.1 ชนิดข้อมูลในภาษา Python เป็นต้น

ชื่อภาษาอังกฤษ	ชื่อภาษาไทย	ชื่อในโปรแกรมภาษา Python	คำอธิบายเพิ่มเติม
Integer	จำนวนเต็ม	int	ตัวเลขที่สามารถเขียนได้โดยไม่มีเศษส่วน: 22 10 0 -300
Boolean	บูลีน	bool	ค่าตระรากที่แสดงถึง True หรือ False
Floating-point	ทศนิยม	float	ตัวเลขที่มีส่วนประกอบเป็นทศนิยม: 3.14 2.73 10.0
String	ข้อความ	str	ลำดับของตัวอักษร: "hello world" "hey88340" '2018'
List	ลิสต์	list	ลำดับของตัวเลขที่คั่นด้วยลูกน้ำ้า, สตริง ฯลฯ: [10, '2018', 'hi']
Dictionary	ติกชั้นนารี	dict	การรวมคู่คีย์-ค่า: {"key1": "value1", "key2": "value2"}
Tuple	ทูเพิล	tuple	ลำดับของตัวเลขที่คั่นด้วยลูกน้ำ้า, สตริง ฯลฯ: (10, 20.0, "world", 5)

ชนิดข้อมูลและตัวแปรมีความสัมพันธ์กันอย่างแน่นแฟ้นในการเขียนโปรแกรม ชนิดข้อมูลกำหนดวิธีการจัดเก็บและดำเนินการกับข้อมูล ขณะที่ตัวแปรเป็นชื่อที่ใช้เก็บและอ้างอิงข้อมูล การเลือกชนิดข้อมูลที่เหมาะสมและการใช้ตัวแปรที่มีชื่อชัดเจนช่วยให้โปรแกรมมีความชัดเจน ง่ายต่อการบำรุงรักษา และป้องกันข้อผิดพลาดที่อาจเกิดขึ้นจากการใช้ข้อมูลที่ไม่ตรงกัน

2.3.3 การดำเนินการตัวแปรและชนิดข้อมูลในภาษา Python

1. การกำหนดค่าให้กับตัวแปร (Assignment)

ในภาษา Python การกำหนดค่าหรือการให้ค่า (Assignment) ให้กับตัวแปรจะใช้เครื่องหมาย = ซึ่งต่างจากการเรียบเทียบความเท่ากับที่ใช้เครื่องหมาย == การกำหนดค่านี้จะทำให้ค่าของตัวแปรถูกเก็บไว้ในหน่วยความจำของคอมพิวเตอร์

```
teacher = "Mr. Sarayut Gonwirat"
print(teacher)
```

ในบรรทัดนี้ได้กำหนดค่าตัวแปร teacher ให้มีค่าเป็น "Mr. Sarayut Gonwirat" และสั่ง print() เพื่อแสดงผลลัพธ์

Mr. Sarayut Gonwirat

2. การตรวจสอบชนิดข้อมูลของตัวแปร คือ การใช้ฟังก์ชัน type() เพื่อตรวจสอบชนิดข้อมูลของตัวแปร teacher ซึ่งผลลัพธ์ที่ได้จะเป็น

```
print(type(teacher))
```

ผลลัพธ์

<class 'str'>

3. การเปลี่ยนค่าของตัวแปร ในบรรทัดนี้ได้เปลี่ยนค่าของตัวแปร teacher เป็น "Mr. Bank"

```
teacher = "Mr. Bank"
print(teacher)
```

เมื่อรันคำสั่ง print(teacher) จะได้ผลลัพธ์

Mr. Bank

4. การรวมสตริง (String Concatenation) เนื่องด้วยชนิดข้อมูลนี้ เป็นข้อความ สามารถที่จะเพิ่มข้อมูลหรือต่อข้อความเพื่อแสดงผลได้ดังนี้

```
greeting = "Hello, " + teacher
```

```
print(greeting)
```

ในบรรทัดนี้ ได้รวมสตริง "Hello, " กับค่าของตัวแปร teacher ผลลัพธ์ที่ได้จาก print(greeting) ดังนี้

```
Hello, Mr. Bank
```

2.3.4 การดำเนินการชนิดข้อความ (String)

สตริง (String) คือชนิดข้อมูลที่ใช้ในการเก็บลำดับของตัวอักษร ซึ่งสามารถประกอบด้วยตัวอักษร ตัวเลข และสัญลักษณ์ต่างๆ โดยสตริงจะถูกประกาศด้วยเครื่องหมายคำพูดเดี่ยว (' ') หรือคำพูดคู่ ("")

```
greeting = "Hello, World!"  
name = "Sarayut Gonwirat"
```

ตารางที่ 2 พิจารณาการดำเนินการต่างๆ กับสตริงในภาษา Python พร้อมคำอธิบายและตัวอย่างโค้ด พิจารณาที่ 2 ฟังก์ชันการดำเนินการต่างๆ เช่น title(), upper(), lower(), rstrip(), lstrip(), และ strip() ใช้ในการปรับแต่งรูปแบบของ สตริง นอกจากนี้ยังมีการเข้าถึงตัวอักษรในสตริง (s[i]), การตัดส่วนของสตริง (s[a:b]), การหาความยาว (len()), และการตรวจสอบสตริงอยู่ (in)

ตารางที่ 2.2 การดำเนินการฟังก์ชันของตัวแปรชนิดข้อความ

ฟังก์ชันการดำเนินการ	คำอธิบาย	ตัวอย่างโค้ด	ผลลัพธ์
title()	ทำให้ตัวอักษรตัวแรกของแต่ละคำในสตริง เป็นตัวใหญ่ เช่นชื่อ	<pre>name = "sarayut gonwirat" print(name.title())</pre>	Sarayut Gonwirat
upper()	ทำให้ตัวอักษรทั้งหมดในสตริงเป็นตัวใหญ่	<pre>name = "sarayut gonwirat" print(name.upper())</pre>	SARAYUT GONWIRAT
lower()	ทำให้ตัวอักษรทั้งหมดในสตริงเป็นตัวเล็ก	<pre>name = "Sarayut Gonwirat" print(name.lower())</pre>	sarayut gonwirat
rstrip()	ลบช่องว่างทางด้านขวาของสตริง	<pre>name = " Sarayut Gonwirat " print(" " + name.rstrip() + " ")</pre>	Sarayut Gonwirat

ฟังก์ชันการดำเนินการ	คำอธิบาย	ตัวอย่างโค้ด	ผลลัพธ์
lstrip()	ลบซ่องว่างทางด้านซ้ายของสตริง	<pre>name = " Sarayut Gonwirat " print(" " + name.lstrip() + " ")</pre>	Sarayut Gonwirat
strip()	ลบซ่องว่างทางด้านซ้ายและขวาของสตริง	<pre>name = " Sarayut Gonwirat " print(" " + name.strip() + " ")</pre>	Sarayut Gonwirat
s[i]	เข้าถึงตัวอักษรในสตริงโดยใช้ดัชนี (index) ซึ่งเริ่มต้นที่ 0	<pre>name = "Sarayut Gonwirat" print(name[0]) print(name[3])</pre>	S a
s[a:b]	การตัดส่วนของสตริงโดยใช้ดัชนี	<pre>name = "Sarayut Gonwirat" print(name[1:5])</pre>	aray
len()	เพื่อหาความยาวของสตริง	<pre>name = "Sarayut Gonwirat" print(len(name))</pre>	16
in	ตรวจสอบว่าสตริงมีอยู่ในสตริงหลัก หรือไม่	<pre>name = "Sarayut Gonwirat" print("Sarayut" in name) print("sarayut" in name)</pre>	True False

2.3.5 การจัดรูปแบบข้อความ (String Format)

การจัดรูปแบบข้อความ (String Formatting) คือกระบวนการในการสร้างสตริงที่ประกอบด้วยข้อมูลจากตัวแปรหรือค่าคงที่ โดยที่ค่าของตัวแปรหรือค่าคงที่เหล่านั้นจะถูกแทรกเข้าไปในสตริงตามตำแหน่งที่กำหนดไว้ การจัดการสตริงทำให้สามารถสร้างข้อความที่มีการปรับแต่งและการรวมข้อมูลได้อย่างง่ายดาย และสามารถทำให้ข้อความที่สร้างขึ้นมีความชัดเจนและเข้าใจง่ายยิ่งขึ้น โดยในภาษา Python ประกอบด้วยการใช้การจัดรูปแบบ มีดังนี้

1. การใช้เครื่องหมาย % (เปอร์เซ็นต์)
2. การใช้ฟังก์ชัน str.format()
3. การใช้ f-string (รูปแบบใหม่ตั้งแต่ Python 3.6 ขึ้นไป)

โดยในเอกสารประกอบการสอนนี้ใช้รูปแบบในข้อ 3 f-string

```
first_name = "Sarayut"
last_name = "Gonwirat"
full_name = f"{first_name} {last_name}"
print(full_name)
```

จากโค้ดด้านบน สร้างตัวแปร first_name และ last_name ใช้ f-string เพื่อร่วมชื่อและนามสกุลเข้าด้วยกัน โดยใช้ {} เพื่อใส่ตัวแปรลงในสตริง แสดงผลลัพธ์ที่ได้จากการรวม ผลลัพธ์จากการพิมพ์ Sarayut Gonwirat

2.3.6 การจัดรูปแบบข้อความด้วยบรรทัดใหม่และแท็บ

1. การใช้บรรทัดใหม่ (\n)

เมื่อต้องการให้ข้อความแสดงผลในหลายบรรทัด สามารถใช้ตัวอักษรพิเศษ \n (newline) เพื่อแบ่งข้อความให้อยู่ในบรรทัดใหม่ได้ ตัวอย่างเช่น

```
print("My name is John Doe.\nI am a software developer.\nI love coding.")
```

ผลลัพธ์จากการพิมพ์

```
My name is John Doe.
I am a software developer.
I love coding.
```

2. การใช้แท็บ (\t)

เมื่อต้องการเว้นวรรคระหว่างข้อความให้เป็นระเบียบ สามารถใช้ตัวอักษรพิเศษ \t (tab) เพื่อสร้างช่องว่างระหว่างข้อความได้ ตัวอย่างเช่น

```
print("Name:\tJohn Doe\nJob:\tSoftware Developer\nSkill:\tPython")
```

ผลลัพธ์จากการพิมพ์

```
Name:      John Doe
Job:       Software Developer
Skill:     Python
```

3. การใช้ร่วมกันระหว่างบรรทัดใหม่และแท็บ

สามารถใช้ \n และ \t ร่วมกันเพื่อจัดรูปแบบข้อความที่ซับซ้อนได้ ตัวอย่างเช่น

```
print("My Profile:\n\tName:\tJohn Doe\n\tAge:\t30\n\tJob:\tSoftware Developer")
```

ผลลัพธ์จากการพิมพ์

```
My Profile:
  Name:      John Doe
  Age:       30
  Job:      Software Developer
```

การใช้ \t และ \n อย่างเหมาะสมจะช่วยให้การแสดงผลข้อความเป็นระเบียบ อ่านง่าย และเข้าใจง่ายยิ่งขึ้น

ตัวแปรในภาษา Python คือ "กล่องเก็บของ" ที่สามารถเก็บข้อมูลที่เปลี่ยนแปลงได้ตลอดเวลา การเลือกใช้ตัวแปรที่สื่อความหมายและชื่อข้อมูลที่เหมาะสมจะช่วยให้โค้ดอ่านง่ายและลดข้อผิดพลาด ซึ่งทักษะนี้จะยิ่งมีความสำคัญมากขึ้นเมื่อทำงานกับโค้ดจำนวนมาก โดยเฉพาะในโครงการขนาดใหญ่ที่มีผู้ทำงานร่วมกันหลายคน

2.4 การโปรแกรมการคำนวณทางคณิตศาสตร์

2.4.1 ตัวเลขและชนิด

ใน Python สามารถทำงานกับตัวเลขได้หลากหลายรูปแบบ ซึ่งตัวเลขในภาษา Python จะแบ่งออกเป็นสองประเภทหลัก ๆ ได้แก่

1. จำนวนเต็ม (Integer - int) จำนวนเต็มเป็นตัวเลขที่ไม่มีเศษส่วน เช่น 1, 2, -3, หรือ 100 เป็นต้น ตัวเลขประเภทนี้ใน Python ถูกจัดเก็บอยู่ในประเภท int เช่น

`type(2)`

ฟังก์ชัน `type()` ใน Python ใช้เพื่อตรวจสอบประเภทของตัวแปรหรือค่าที่อยู่ในตัวแปรนั้น ๆ โดยจะส่งคืนประเภท (type) ของข้อมูลหรือวัตถุที่สนใจ

ผลลัพธ์จากการพิมพ์

`<class 'int'>`

2. จำนวนทศนิยม (Float - float) จำนวนทศนิยมคือค่าตัวเลขที่มีจุดทศนิยม เช่น 2.0, -3.14, หรือ 100.56 เป็นต้น ตัวเลขประเภทนี้ใน Python ถูกจัดเก็บอยู่ในประเภท float เช่น

`type(2.0)`

ผลลัพธ์จากการพิมพ์

`<class 'float'>`

3. ข้อความ (String - str) แม้ว่าค่าที่อยู่ในตัวแปรจะมีลักษณะเหมือนตัวเลข แต่ถ้าถูกเขียนในรูปแบบข้อความ เช่น "2.0", "100", หรือ "abc" จะถูกจัดเก็บในประเภทข้อความ (str) เช่น

`type("2.0")`

ผลลัพธ์จากการพิมพ์

`<class 'str'>`

4. ความไม่แม่นยำของจำนวนทศนิยม Python ใช้การเก็บค่าทศนิยมในรูปแบบของระบบฐานสอง (binary floating-point) ซึ่งอาจส่งผลให้เกิดความคลาดเคลื่อนเล็กน้อยในการคำนวณ เช่น

```
print(0.1 + 0.2)
print(0.1 * 3)
```

ผลลัพธ์จากการพิมพ์

```
0.3000000000000004
0.3000000000000004
```

2.4.2 การคำนวณพื้นฐาน

Python รองรับการคำนวณพื้นฐานด้วยตัวดำเนินการ (Operator) (+, -, *, /) ดังนี้

1. การบวก (Addition: +) ใช้สำหรับการรวมค่าของตัวเลขสองจำนวน

```
print(8 + 5)
```

ผลลัพธ์

13

2. การคูณ (Multiplication: *) ใช้สำหรับการคูณค่าของตัวเลขสองจำนวน

```
print(8 * 5)
```

ผลลัพธ์

40

3. การลบ (Subtraction: -) ใช้สำหรับการหาผลต่างระหว่างตัวเลขสองจำนวน

```
print(8 - 5)
```

ผลลัพธ์

3

4. การหาร (Division: /) ใช้สำหรับการหารค่าของตัวเลขสองจำนวน ซึ่งผลลัพธ์จะเป็นจำนวนทศนิยม

เสมอ

```
print(8 / 5)
```

ผลลัพธ์

1.6

5. การบวก (Addition: +) ใช้สำหรับการรวมค่าของตัวเลขสองจำนวน

```
print(8 + 5)
```

ผลลัพธ์

13

6. การหารแบบปัดเศษลง (Floor Division: //) ใช้สำหรับการหารค่าของตัวเลขสองจำนวน โดยปัดเศษลงให้เป็นจำนวนเต็มที่ใกล้เคียงที่สุด

```
print(8 // 5)
```

ผลลัพธ์

1

7. การหารเอาเศษ (Modulo: %) ใช้สำหรับการหารค่าเศษที่เหลือจากการหารของตัวเลขสองจำนวน

```
print(8 % 5)
```

ผลลัพธ์

3

8. การยกกำลัง (Exponentiation: `**`) ใช้สำหรับการคูณตัวเลขตัวหนึ่งด้วยตัวมันเองจำนวนครั้งที่กำหนด

```
print(8 ** 2) # 8 * 8
```

ผลลัพธ์

64

ตารางที่ 2.3 สรุปการดำเนินการทางคณิตศาสตร์ใน Python

ตัวดำเนินการ (Operator)	ชื่อ (Operation)	ตัวอย่างโค้ด	ผลลัพธ์
<code>+</code>	การบวก (Addition)	<code>(8 + 5)</code>	13
<code>-</code>	การลบ (Subtraction)	<code>(8 - 5)</code>	3
<code>*</code>	การคูณ (Multiplication)	<code>(8 * 5)</code>	40
<code>/</code>	การหาร (Division)	<code>(8 / 5)</code>	1.6
<code>//</code>	การหารแบบปัดเศษลง (Floor Division)	<code>(8 // 5)</code>	1
<code>%</code>	การหารเอาเศษ (Modulo)	<code>(8 % 5)</code>	3
<code>**</code>	ยกกำลัง (Exponentiation)	<code>(8 ** 5)</code>	64

2.4.3 ลำดับการคำนวณ (Order of Operation)

ลำดับการคำนวณ หรือ Order of Operation คือ กฎที่กำหนดลำดับความสำคัญของตัวดำเนินการทางคณิตศาสตร์ในคำสั่งที่มีการคำนวณหลายขั้นตอน เพื่อให้ได้ผลลัพธ์ที่ถูกต้องและสอดคล้องตามมาตรฐานสากล โดยลำดับนี้จะช่วยลดข้อผิดพลาดที่อาจเกิดขึ้นจากการคำนวณ เช่น การคูณ การบวก และการใช้งานวงเล็บในโปรแกรม Python

ใน Python (และภาษาโปรแกรมอื่น ๆ) การคำนวณที่ประกอบไปด้วยตัวดำเนินการหลายประเภท เช่น การบวก ลบ คูณ และหาร จะไม่ได้ดำเนินการเรียงจากซ้ายไปขวาเสมอไป แต่จะปฏิบัติตามลำดับความสำคัญของตัวดำเนินการที่กำหนดไว้ ซึ่งสอดคล้องกับหลักการทางคณิตศาสตร์ที่รู้จักกันดีในชื่อ PEMDAS หรือ BODMAS ได้แก่

- **Parentheses (P)** คำนวณในวงเล็บก่อน
- **Exponents (E)** ยกกำลัง
- **Multiplication (M)** และ **Division (D)** คูณและหาร (ดำเนินการจากซ้ายไปขวา)
- **Addition (A)** และ **Subtraction (S)** บวกและลบ (ดำเนินการจากซ้ายไปขวา)

ตัวอย่างที่ 1

```
print(2 + 3 * 4)
```

ผลลัพธ์

14

ลำดับการคำนวณ

คำนวณ $3 * 4$ ก่อน (เพราะคูณมาก่อนบวก) จากนั้น คำนวณ $2 + 12$

ตัวอย่างที่ 2

```
print((2 + 3 * 4))
```

ผลลัพธ์

20

ลำดับการคำนวณ คำนวณในวงเล็บ $(2 + 3)$ ก่อน จากนั้น คำนวณ $5 * 4$

ตัวอย่างที่ 3

```
print((2 + 3 ** 2 * 4))
```

ผลลัพธ์

38

ลำดับการคำนวณ

คำนวณ $3 ** 2$ (ยกกำลังก่อน)

คำนวณ $9 * 4$ (คูณถัดมา)

คำนวณ $2 + 36$ (บวกสุดท้าย)

2.4.4 การบวกเพิ่มและลดค่า

Python มีตัวดำเนินการที่ช่วยเพิ่มหรือลดค่าของตัวแปรได้อย่างสะดวก เช่น การใช้ `+=` และ `-=` ซึ่งเป็นรูปแบบย่อของการคำนวณตัวแปรตัวเดิมกับค่าที่ต้องการเพิ่มหรือลด

1. การบวกเพิ่ม (Increment: `+=`) การบวกเพิ่มใน Python สามารถทำได้โดยใช้ `+=` ซึ่งเทียบเท่ากับการเขียน $x = x + n$ ตัวอย่างการใช้งาน

```
number = 7 # กำหนดค่าเริ่มต้น
number = number + 1 # เพิ่มค่า 1 ให้กับตัวแปร
print(f"Number is {number}.")
```

ผลลัพธ์

Number is 8.

2. การลดค่า (Decrement: `-=`) การลดค่าใน Python สามารถทำได้โดยใช้ `-=` ซึ่งเทียบเท่ากับการเขียน $x = x - n$ ตัวอย่างการใช้งาน

```
number = 9 # กำหนดค่าเริ่มต้น
number = number - 1 # ลดค่า 1 ให้กับตัวแปร
print(f"Number is {number}.")
```

ผลลัพธ์

ตารางที่ 2.4 สรุปการดำเนินการทางคณิตศาสตร์ใน Python

ตัวดำเนินการ (Operator)	ชื่อ (Operation)	ตัวอย่างโค้ด	ผลลัพธ์
$+=$	การบวกค่าเพิ่ม	<code>x += 1</code>	<code>x = x + 1</code>
$-=$	การลดค่าลง	<code>x -= 1</code>	<code>x = x - 1</code>

2.4.5 การใช้งาน E-notation

Python รองรับ E-notation (Exponential Notation) ซึ่งเป็นรูปแบบการแสดงตัวเลขที่มีค่ามากหรือน้อยมากในลักษณะของเลขยกกำลังฐาน 10 โดยใช้ตัวอักษร e หรือ E ร่วมกับเลขยกกำลัง เพื่อช่วยให้การจัดการตัวเลขง่ายและกระชับมากขึ้น

$$aEb = a \times 10^b$$

1. การแสดงค่าตัวเลขขนาดใหญ่ E-notation ใช้สำหรับแสดงค่าที่มีขนาดใหญ่มาก โดยการย้ายจุดทศนิยมไปทางขวา

```
print(9938712345656.34 * 4823459023067.456)
```

ผลลัพธ์

4.793897174132799e+25

มีค่าเท่ากับ $4.793897174132799 \times 10^{25}$

2. การบวกค่าที่ใช้ E-notation การใช้ E-notation สำหรับค่าที่แสดงในรูปเลขยกกำลังฐาน 10 ทำให้การคำนวณง่ายขึ้น

```
a = 2.5e6 # 2.5 * 10^6
b = 1.2e7 # 1.2 * 10^7
print(a + b)
```

ผลลัพธ์

14500000.0

หมายถึง $2,500,000 + 12,000,000 = 14,500,000$

3. การบวกค่าที่ยกกำลังสูง สามารถคำนวณค่าที่มีเลขยกกำลังสูงได้เช่นกัน

```
a = 2.5e75 # 2.5 * 10^75
b = 1.2e74 # 1.2 * 10^74
print(a + b)
```

ผลลัพธ์

2.72e+75

หมายถึง $2.6 \times 10^{75} + 0.12 \times 10^{75} = 2.72 \times 10^{75}$

4. การจัดการค่าทศนิยมขนาดเล็ก E-notation ยังสามารถใช้แสดงค่าทศนิยมที่เล็กมาก โดยบัญจุดทศนิยมไปทางซ้าย

```
x = 1.752e-13
print(x)
```

ผลลัพธ์

```
1.752e-13
หมายถึง 0.0000000000001752
```

2.4.6 การเปลี่ยนชนิดข้อมูล

Python รองรับการเปลี่ยนชนิดข้อมูล (Type Conversion) เพื่อให้สามารถปรับเปลี่ยนข้อมูลให้เหมาะสมกับการใช้งาน เช่น การแปลงจากจำนวนเต็ม (Integer) เป็นทศนิยม (Float) หรือจากตัวเลขเป็นข้อความ (String) โดยมีรูปแบบการเปลี่ยนชนิดข้อมูลดังนี้

1. float() ใช้เพื่อสร้างตัวเลขทศนิยม (Float) จากข้อความ (String) หรือจำนวนเต็ม (Integer)
 2. int() ใช้เพื่อสร้างจำนวนเต็ม (Integer) จากข้อความ (String) หรือตัวเลขทศนิยม (Float)
 3. str() ใช้เพื่อสร้างข้อความ (String) จากตัวเลข (Number) หรือชนิดข้อมูลอื่น ๆ
- ตัวอย่างการเปลี่ยนข้อมูล
1. แปลงจาก Float เป็น Integer

```
c = 38.0      # ตัวเลขทศนิยม
d = int(c)    # แปลงเป็นจำนวนเต็ม
print(c)      # ผลลัพธ์: 38.0
print(d)      # ผลลัพธ์: 38
```

ผลลัพธ์

```
38.0
38
2. แปลงจาก Integer เป็น Float
```

```
a = 24        # จำนวนเต็ม
b = float(a)  # แปลงเป็นตัวเลขทศนิยม
print(a)       # ผลลัพธ์: 24
print(b)       # ผลลัพธ์: 24.0
```

ผลลัพธ์

```
24
24.0
3. แปลงจาก Integer เป็น String
```

```
x = 50        # จำนวนเต็ม
y = str(x)    # แปลงเป็นข้อความ
print(x)      # ผลลัพธ์: 50
print(y)      # ผลลัพธ์: "50"
```

ผลลัพธ์

```
50
50
```

4. แปลงจาก String เป็น Integer หรือ Float

```
s = "42"      # ข้อความ
n = int(s)    # แปลงเป็นจำนวนเต็ม
f = float(s)  # แปลงเป็นตัวเลขทศนิยม
print(n)      # ผลลัพธ์: 42
print(f)      # ผลลัพธ์: 42.0
```

ผลลัพธ์

```
42
42.0
```

5. การตรวจสอบชนิดข้อมูล

```
a = '44.2'  # ข้อความ
b = 44.2     # ตัวเลขทศนิยม
print(type(a)) # ผลลัพธ์: <class 'str'>
print(type(b)) # ผลลัพธ์: <class 'float'>
```

ผลลัพธ์

```
<class 'str'>
<class 'float'>
```

6. ข้อผิดพลาดจากการแปลง การแปลงชนิดข้อมูลอาจเกิดข้อผิดพลาดได้หากค่าที่แปลงไม่สอดคล้อง กับชนิดข้อมูลเป้าหมาย

```
float('fred')
```

ผลลัพธ์

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: 'fred'
```

ตารางที่ 2.5 สรุปการแปลงชนิดข้อมูลใน Python

คำสั่ง	คำอธิบาย	ตัวอย่างโค้ด	ผลลัพธ์
int()	แปลงเป็นจำนวนเต็ม (Integer)	int(54.99)	54
float()	แปลงเป็นตัวเลขทศนิยม (Float)	float(76.3')	76.3
str()	แปลงเป็นข้อความ (String)	str(44.2)	'44.2'

Python มีความโดยเด่นในด้านการคำนวณ เนื่องจากโครงสร้างภาษาได้รับการออกแบบมาให้มีความ ใกล้เคียงกับคณิตศาสตร์ที่ใช้ในชีวิตประจำวัน ด้วยตัวดำเนินการพื้นฐานที่ครบถ้วนและความสามารถในการ

เขียนสมการได้อย่างสั้น กระชับ และเข้าใจง่าย Python จึงเป็นภาษาที่เหมาะสมอย่างยิ่งสำหรับงานด้านวิทยาศาสตร์ข้อมูล (Data Science) หรือปัญญาประดิษฐ์ (AI) ซึ่งต้องมีการคำนวณข้อมูลจำนวนมากมหาศาลอยู่เสมอ

2.5 ลิสต์ (List)

ลิสต์ (List) เป็นโครงสร้างข้อมูลชนิดหนึ่งใน Python ที่ใช้เก็บข้อมูลหลายค่าภายในตัวแปรเดียวกัน โดยลิสต์สามารถเก็บข้อมูลที่มีชนิดต่างกัน เช่น จำนวนเต็ม (int) ทศนิยม (float) และข้อความ (str) เป็นต้น นอกจากนี้ ลิสต์ยังมีคุณสมบัติในการจัดการข้อมูลที่ยืดหยุ่น เช่น การเพิ่ม ลบ หรือแก้ไขข้อมูล

คุณสมบัติของลิสต์

- สามารถเก็บข้อมูลหลายค่าได้ในตัวแปรเดียว เช่น [1, 2, 3], ["apple", "banana", "cherry"]
- สามารถเก็บข้อมูลชนิดต่างกันในลิสต์เดียวกันได้ เช่น [1, "hello", 3.14]
- รองรับการเปลี่ยนแปลง (Mutable) สามารถเพิ่ม, ลบ หรือแก้ไขค่าภายในลิสต์ได้
- สามารถเข้าถึงค่าด้วยดัชนี (Index) ดัชนีในลิสต์เริ่มต้นที่ 0

2.5.1 การสร้างลิสต์

สามารถสร้างลิสต์ได้โดยใช้เครื่องหมาย [] หรือฟังก์ชัน list()

```
my_list = [1, 2, 3, 4]           # ลิสต์ของตัวเลข
fruits = ["apple", "banana"]     # ลิสต์ของข้อความ
mixed = [1, "hello", 3.14]       # ลิสต์ที่มีข้อมูลต่างชนิด
empty_list = []                  # ลิสต์ว่าง
```

ตัวอย่างการสร้างลิสต์

- ลิสต์ที่เก็บตัวเลข

```
numbers = [1, 2, 3, 4, 5]
print(numbers)
```

ผลลัพธ์

```
[1, 2, 3, 4, 5]
2. ลิสต์ที่เก็บข้อความ
```

```
fruits = ["apple", "banana", "cherry"]
print(fruits)
```

ผลลัพธ์

```
['apple', 'banana', 'cherry']
3. ลิสต์ที่เก็บข้อมูลหลายประเภท
```

```
mixed = [1, "hello", 3.14, True]
print(mixed)
```

ผลลัพธ์

```
[1, 'hello', 3.14, True]
```

4. ลิสต์ว่าง

```
empty_list = []
print(empty_list)
```

ผลลัพธ์

```
[]
```

2.5.2 การเข้าถึงข้อมูล

Python สามารถเข้าถึงข้อมูลในลิสต์ได้โดยใช้ ดัชนี (Index) ซึ่งช่วยให้อ้างอิงค่าต่าง ๆ ที่อยู่ในตำแหน่งเฉพาะของลิสต์ ดัชนีใน Python เริ่มต้นที่ 0 สำหรับตำแหน่งแรก และใช้ค่าลบสำหรับการเข้าถึงข้อมูลจากท้ายลิสต์

```
fruits = ["apple", "banana", "cherry"]

# เข้าถึงข้อมูลโดยใช้ดัชนี
print(fruits[0]) # ผลลัพธ์: apple
print(fruits[1]) # ผลลัพธ์: banana
print(fruits[-1]) # ผลลัพธ์: cherry (เข้าถึงค่าจากท้ายลิสต์)
```

ผลลัพธ์

```
apple
banana
cherry
```

ตัวอย่างการเข้าถึงเพิ่มเติม

1. การใช้ดัชนีเพื่อเข้าถึงข้อมูลเฉพาะตำแหน่ง

```
fruits = ["apple", "banana", "cherry", "orange", "grape"]

# เข้าถึงข้อมูลในตำแหน่งต่างๆ
print(fruits[2]) # ผลลัพธ์: cherry
print(fruits[-2]) # ผลลัพธ์: orange (ตำแหน่งก่อนสุดท้าย)
```

ผลลัพธ์

```
cherry
orange
```

2. การเข้าถึงข้อมูลหลายค่าด้วยการใช้ช่วงดัชนี (Slicing) สามารถใช้ Slicing เพื่อเข้าถึงข้อมูลในช่วงตำแหน่งที่กำหนด

```
fruits = ["apple", "banana", "cherry", "orange", "grape"]

# เข้าถึงข้อมูลตั้งแต่ตำแหน่งที่ 1 ถึง 3
print(fruits[1:4]) # ผลลัพธ์: ['banana', 'cherry', 'orange']

# เข้าถึงข้อมูลตั้งแต่ตำแหน่งที่ 2 จนถึงท้ายลิสต์
```

```
print(fruits[2:]) # ผลลัพธ์: ['cherry', 'orange', 'grape']
```

เข้าถึงข้อมูลตั้งแต่เดือนที่ 2 จนถึงเดือนที่ 3

```
print(fruits[:3]) # ผลลัพธ์: ['apple', 'banana', 'cherry']
```

ผลลัพธ์

```
['banana', 'cherry', 'orange']
['cherry', 'orange', 'grape']
['apple', 'banana', 'cherry']
```

3. การตรวจสอบจำนวนข้อมูลในลิสต์ (Length of List)

```
fruits = ["apple", "banana", "cherry"]
```

```
print(len(fruits)) # ผลลัพธ์: 3
```

ผลลัพธ์

3

4. ข้อผิดพลาดจากการใช้นอกดัชนี หากพยายามเข้าถึงข้อมูลที่อยู่นอกช่วงของลิสต์ จะเกิดข้อผิดพลาด IndexError

```
fruits = ["apple", "banana", "cherry"]
```

```
print(fruits[3]) # คำแนะนำที่ 3 ไม่มีข้อมูล
```

ผลลัพธ์

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
IndexError: list index out of range
```

ข้อสังเกตเพิ่มเติม

- ผลลัพธ์ ดัชนีเริ่มต้นที่ 0 ตำแหน่งแรกของลิสต์คือ 0
- การเข้าถึงข้อมูลจากท้ายลิสต์ ใช้ดัชนีติดลบ เช่น -1 สำหรับข้อมูลสุดท้าย
- ข้อผิดพลาด IndexError เกิดเมื่อพยายามเข้าถึงตำแหน่งที่ไม่มีอยู่ในลิสต์

2.5.3 การเพิ่มข้อมูล

Python สามารถเพิ่มข้อมูลเข้าไปในลิสต์ได้หลายวิธี โดยขึ้นอยู่กับความต้องการ เช่น การเพิ่มข้อมูลใหม่ที่ท้ายลิสต์ การแทรกข้อมูลในตำแหน่งที่กำหนด หรือการเพิ่มข้อมูลหลายค่าจากลิสต์อื่น

1. การใช้ `append()` พังก์ชัน `append()` ใช้สำหรับเพิ่มข้อมูลใหม่เข้าไปที่ ท้ายลิสต์

```
fruits = ["apple", "banana"]
```

```
fruits.append("cherry") # เพิ่ม "cherry" เข้าไปท้ายลิสต์
```

ผลลัพธ์

```
['apple', 'banana', 'cherry']
```

คำสั่ง `fruits.append("cherry")` จะเพิ่มข้อมูล "cherry" เข้าไปที่ท้ายลิสต์ `fruits` ใช้ `append()` ได้มีอีกข้อมูลที่ต้องการเพิ่มมีเพียง 1 ค่า

2. การใช้ `insert()` พังก์ชัน `insert()` ใช้สำหรับเพิ่มข้อมูลใหม่เข้าไปในลิสต์ ในตำแหน่งที่ระบุ

```
fruits = ["apple", "banana"]
fruits.insert(1, "blueberry") # เพิ่ม "blueberry" ที่ตำแหน่งที่ 1
print(fruits)
ผลลัพธ์
```

['apple', 'blueberry', 'banana']

คำสั่ง fruits.insert(1, "blueberry") จะเพิ่ม "blueberry" เข้าไปที่ตำแหน่งที่ 1 (ระหว่าง "apple" และ "banana") ตำแหน่งที่ระบุจะต้องเป็นด้วยในลิสต์ เช่น

- ตำแหน่ง 0 คือก่อน "apple"
- ตำแหน่ง -1 คือก่อนข้อมูลสุดท้าย

3. การใช้ **extend()** พังก์ชัน extend() ใช้สำหรับเพิ่ม ข้อมูลหลายค่า เข้าไปในลิสต์ โดยค่าที่เพิ่มจะถูกนำมารวบกับลิสต์หรือ iterable อื่น เช่น list, tuple, หรือ set

```
fruits = ["apple", "banana"]
more_fruits = ["cherry", "orange"]
fruits.extend(more_fruits) # เพิ่มข้อมูลจาก more_fruits เข้าสู่ fruits
print(fruits)
ผลลัพธ์
```

['apple', 'banana', 'cherry', 'orange']

คำสั่ง fruits.extend(more_fruits) จะเพิ่มข้อมูลใน more_fruits ("cherry", "orange") เข้าไปต่อท้าย fruits พังก์ชัน extend() เมามะสำหรับการเพิ่มข้อมูลหลายค่าพร้อมกัน

2.5.4 การลบข้อมูล

Python มีคำสั่งหลากหลายสำหรับการลบข้อมูลในลิสต์ ขึ้นอยู่กับสถานการณ์ เช่น ลบตามค่า ลบตามตำแหน่ง หรือ ลบข้อมูลทั้งหมดในลิสต์

1. การใช้ **remove()** พังก์ชัน remove() ใช้สำหรับลบข้อมูลที่มีค่าตรงกับที่กำหนด โดยจะลบเฉพาะข้อมูลที่พบเป็นตัวแรกในลิสต์ หากค่าที่กำหนดไม่มีอยู่ในลิสต์ จะเกิดข้อผิดพลาด ValueError

```
fruits = ["apple", "banana", "cherry"]
fruits.remove("banana") # ลบ "banana" ออกจากลิสต์
print(fruits)
```

ผลลัพธ์

['apple', 'cherry']

remove("banana") จะค้นหา "banana" ในลิสต์และลบออก หาก "banana" ไม่มีอยู่ในลิสต์ จะเกิดข้อผิดพลาด

2. การใช้ **pop()** พังก์ชัน pop() ใช้สำหรับลบข้อมูลตามตำแหน่ง (ด้วย) ที่กำหนด และจะส่งคืนค่าที่ถูกลบ หากไม่ระบุตำแหน่ง พังก์ชันจะลบ ข้อมูลสุดท้าย ของลิสต์

```
fruits = ["apple", "banana", "cherry"]
```

```
fruits.pop(1) # ลบข้อมูลที่ตำแหน่งที่ 1 ("banana")
print(fruits)
```

ผลลัพธ์

```
['apple', 'cherry']
```

pop(1) ลบข้อมูลที่ตำแหน่งดังนี้ที่ 1 (เริ่มนับจาก 0) หากไม่ระบุดังนี้ เช่น pop() จะลบข้อมูลตัวสุดท้ายของลิสต์

3. การใช้ **del** คำสั่ง del ใช้สำหรับลบข้อมูลในลิสต์ตามตำแหน่งที่กำหนด หรือสามารถลบลิสต์ทั้งหมดได้

```
fruits = ["apple", "banana", "cherry"]
del fruits[0] # ลบข้อมูลที่ตำแหน่งที่ 0 ("apple")
print(fruits)
```

ผลลัพธ์

```
['banana', 'cherry']
```

del fruits[0] ลบข้อมูลที่ตำแหน่งดังนี้ที่ 0 ใช้ del fruits เพื่อลบลิสต์ทั้งหมด

2.5.5 การเรียงข้อมูล

Python มีฟังก์ชันที่ช่วยในการจัดเรียงข้อมูลในลิสต์ เช่น การเรียงลำดับข้อมูล (จากน้อยไปมาก หรือมากไปน้อย) และการกลับลำดับข้อมูลในลิสต์ ซึ่งสามารถทำได้อย่างง่ายดายด้วยฟังก์ชัน sort() และ reverse()

1. การใช้ **sort()** ฟังก์ชัน sort() ใช้สำหรับจัดเรียงข้อมูลในลิสต์ตามลำดับจากน้อยไปมาก (Ascending Order) หรือจากมากไปน้อย (Descending Order) หากกำหนดพารามิเตอร์ reverse=True

```
numbers = [3, 1, 2]
numbers.sort() # เรียงข้อมูลจากน้อยไปมาก
print(numbers)
```

ผลลัพธ์

```
[1, 2, 3]
```

sort() จะจัดเรียงข้อมูลจากน้อยไปมากโดยอัตโนมัติ หากต้องการเรียงจากมากไปน้อยให้เพิ่มพารามิเตอร์ reverse=True

2. การใช้ **reverse()** ฟังก์ชัน reverse() ใช้สำหรับ กลับลำดับข้อมูลในลิสต์ โดยไม่คำนึงถึงค่าของข้อมูลในลิสต์

```
numbers = [3, 1, 2]
numbers.sort() # เรียงข้อมูลจากน้อยไปมาก
print(numbers)
```

ผลลัพธ์

```
[3, 2, 1]
```

reverse() จะทำการกลับลำดับข้อมูลจากท้ายสุดมาอยู่ต้นลิสต์ และต้นลิสต์มาอยู่ท้ายสุด

หมายเหตุ: สำหรับการกลับลำดับข้อมูลโดยไม่จัดเรียงใหม่

2.5.6 ทูเพิล (Tuple)

ทูเพิล (Tuple) เป็นโครงสร้างข้อมูลชนิดหนึ่งใน Python ที่คล้ายกับลิสต์ (List) แต่มีคุณสมบัติที่แตกต่างกันสำคัญ คือ ไม่สามารถเปลี่ยนแปลงค่าได้ (Immutable) หลังจากที่สร้างขึ้นมาแล้ว ทำให้เหมาะสมสำหรับการเก็บข้อมูลที่ไม่ต้องการแก้ไข นอกจากนี้ ทูเพิลยังมีประสิทธิภาพในการทำงานสูงกว่าลิสต์เมื่อใช้เก็บข้อมูลที่มีขนาดใหญ่

ข้อเหมือนระหว่าง List และ Tuple

ทั้ง List และ Tuple เป็นโครงสร้างข้อมูลที่สามารถเก็บค่าหลายค่าในตัวแปรเดียว และสามารถเก็บข้อมูลชนิดต่างกันได้ในรายการเดียวกัน เช่น ตัวเลขและข้อความ นอกจากนี้ ทั้งสองยังสามารถเข้าถึงข้อมูลได้โดยใช้ดัชนี และรองรับการใช้งานร่วมกับฟังก์ชันต่าง ๆ ของ Python เช่น การวนลูปและการคำนวณความยาวด้วย `len()`

ข้อแตกต่างระหว่าง List และ Tuple

List มีคุณสมบัติ Mutable ซึ่งสามารถเปลี่ยนแปลง เพิ่ม หรือลบข้อมูลในรายการได้หลังจากสร้างขึ้น ส่วน Tuple มีคุณสมบัติ Immutable ที่ไม่สามารถเปลี่ยนแปลงค่าได้หลังจากสร้าง ทำให้ Tuple เหมาะกับการเก็บค่าคงที่ที่ไม่ต้องการแก้ไข และมีประสิทธิภาพสูงกว่าเมื่อใช้เก็บข้อมูลที่ไม่เปลี่ยนแปลง

1. การสร้างทูเพิล สามารถสร้างทูเพิลได้โดยใช้เครื่องหมายวงเล็บ () หรือใช้ฟังก์ชัน `tuple()`

```
numbers = (1, 2, 3, 4, 5)
print(numbers)

# ทูเพิลที่เก็บข้อความ
fruits = ("apple", "banana", "cherry")
print(fruits)

# ทูเพิลที่เก็บข้อมูลหลายประเภท
mixed = (1, "hello", 3.14, True)
print(mixed)

# ทูเพิลว่าง
empty_tuple = ()
print(empty_tuple)
```

ผลลัพธ์

```
(1, 2, 3, 4, 5)
('apple', 'banana', 'cherry')
(1, 'hello', 3.14, True)
()
```

2. การเข้าถึงข้อมูลในทูเพิล การเข้าถึงข้อมูลในทูเพิลทำได้เหมือนกับลิสต์ โดยใช้ดัชนี (Index) หรือการ Slicing

```
fruits = ("apple", "banana", "cherry")

# เข้าถึงข้อมูลค่าดัชนี
```

```

print(fruits[0]) # ผลลัพธ์: apple
print(fruits[-1]) # ผลลัพธ์: cherry

# ใช้ Slicing เข้าถึงข้อมูลหลายค่า
print(fruits[1:3]) # ผลลัพธ์: ('banana', 'cherry')

```

ผลลัพธ์

```

apple
cherry
('banana', 'cherry')

```

ตารางที่ 2.6 สรุปคำสั่งที่ใช้กับลิสต์

คำสั่ง	คำอธิบาย	ตัวอย่างโค้ด	ผลลัพธ์
append(x)	เพิ่มข้อมูลที่ท้ายลิสต์	<code>fruits.append("cherry")</code>	<code>['apple', 'banana', 'cherry']</code>
insert(i, x)	เพิ่มข้อมูลในตำแหน่งที่กำหนด	<code>fruits.insert(1, "blueberry")</code>	<code>['apple', 'blueberry', 'banana']</code>
remove(x)	ลบข้อมูลที่ระบุ	<code>fruits.remove("banana")</code>	<code>['apple', 'cherry']</code>
pop(i)	ลบข้อมูลตามตำแหน่งดังนี้	<code>fruits.pop(1)</code>	<code>['apple', 'cherry']</code>
extend(iterable)	เพิ่มข้อมูลจากลิสต์อื่น	<code>fruits.extend(["cherry"])</code>	<code>['apple', 'banana', 'cherry']</code>
sort()	จัดเรียงข้อมูลในลิสต์	<code>numbers.sort()</code>	<code>[1, 2, 3]</code>
reverse()	กลับลำดับข้อมูลในลิสต์	<code>numbers.reverse()</code>	<code>[3, 2, 1]</code>
append(x)	เพิ่มข้อมูลที่ท้ายลิสต์	<code>fruits.append("cherry")</code>	<code>['apple', 'banana', 'cherry']</code>

ลิสต์เปรียบเสมือน "ตะกร้าเก็บของ" ของโปรแกรมเมอร์ที่ช่วยให้สามารถจัดเก็บข้อมูลหลายค่าไว้ในตัวแปรเดียวได้อย่างเป็นระบบ ลิสต์เป็นหนึ่งในโครงสร้างข้อมูลที่มีการใช้งานบ่อยที่สุดในชีวิตจริง เช่น รายการสินค้าในตะกร้าซื้อปั๊งออนไลน์ หรือชุดข้อมูลที่นำไปวิเคราะห์ในงานวิจัย

2.6 ดิกชันนารี (Dictionary)

ดิกชันนารี (Dictionary) เป็นโครงสร้างข้อมูลชนิดหนึ่งใน Python ที่ใช้เก็บข้อมูลแบบ Key-Value โดย Key เป็นตัวระบุที่ไม่ซ้ำกัน (Unique) และ Value เป็นค่าที่กำหนดให้กับ Key นั้น ๆ ดิกชันนารีมีคุณสมบัติที่ยืดหยุ่นและเหมาะสมสำหรับการจัดการข้อมูลที่ต้องการค้นหาอย่างรวดเร็ว

คุณสมบัติของดิกชันนารี

1. จัดเก็บข้อมูลในรูปแบบ Key-Value Pair เช่น {"name": "Alice", "age": 25}
2. Key ต้องไม่ซ้ำกัน แต่ Value สามารถซ้ำกันได้
3. รองรับการเปลี่ยนแปลง (Mutable) สามารถเพิ่ม, ลบ, หรือแก้ไขข้อมูลได้
4. เข้าถึงข้อมูลด้วย Key ไม่ใช้ดัชนีเหมือนลิสต์

2.6.1 การสร้างดิกชันนารี

สร้างดิกชันนารีโดยใช้เครื่องหมาย {} หรือฟังก์ชัน dict()

1. ดิกชันนารีที่เก็บข้อมูลบุคคล

```
person = {"name": "Alice", "age": 25, "city": "New York"}  
print(person)
```

ผลลัพธ์

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

ดิกชันนารี person ประกอบด้วย Key และ Value:

- "name": "Alice"
- "age": 25
- "city": "New York"

เมื่อใช้ print(person) จะพิมพ์ดิกชันนารีออกมาในรูปแบบ Key-Value ทั้งหมด

2. ดิกชันนารีว่าง

```
empty_dict = {}  
print(empty_dict)
```

ผลลัพธ์

```
[]
```

ดิกชันนารี empty_dict ถูกสร้างขึ้นโดยไม่มี Key หรือ Value ใด ๆ การพิมพ์ empty_dict จะแสดงเครื่องหมาย {} ซึ่งหมายถึงดิกชันนารีว่าง

3. การใช้ฟังก์ชัน dict()

```
person = dict(name="Bob", age=30, city="London")  
print(person)
```

ผลลัพธ์

```
{'name': 'Bob', 'age': 30, 'city': 'London'}
```

ฟังก์ชัน dict() ใช้สร้างติกชันนารีด้วย Key และ Value ที่ระบุในรูปแบบของอาร์กิวเมนต์

- "name": "Bob"
- "age": 30
- "city": "London"

การพิมพ์ person จะแสดง Key-Value Pair ทั้งหมดในติกชันนารี

ติกชันนารีเป็นโครงสร้างข้อมูลที่ยืดหยุ่นและเหมาะสมสำหรับการเก็บข้อมูลในรูปแบบ Key-Value ที่ต้องการเข้าถึงอย่างรวดเร็ว โดยสามารถสร้างได้หลากหลายวิธี เช่น การใช้ {} หรือ dict() ซึ่งรองรับข้อมูลหลายชนิดในติกชันนารีเดียวกัน

2.6.2 การเข้าถึงข้อมูล

Python สามารถเข้าถึงข้อมูลในติกชันนารีได้โดยใช้ Key ซึ่งจะส่งคืน Value ที่สอดคล้องกับ Key นั้น ๆ การเข้าถึงข้อมูลในติกชันนารีมีความยืดหยุ่นและรวดเร็ว เนื่องจากไม่ใช้ดัชนี (Index) เหมือนลิสต์

```
person = {"name": "Alice", "age": 25, "city": "New York"}

# เข้าถึง Value ผ่าน Key
print(person["name"]) # ผลลัพธ์: Alice
print(person["age"]) # ผลลัพธ์: 25
```

ผลลัพธ์

```
Alice
25
```

person["name"] ส่งคืนค่า "Alice" ซึ่งเป็น Value ของ Key "name" และ person["age"] ส่งคืนค่า 25 ซึ่งเป็น Value ของ Key "age"

2.6.3 การเพิ่มและแก้ไขข้อมูลในติกชันนารี

ติกชันนารีใน Python มีคุณสมบัติที่เปลี่ยนแปลงได้ (Mutable) ทำให้สามารถเพิ่มข้อมูลใหม่หรือแก้ไขข้อมูลที่มีอยู่ได้อย่างง่ายดาย โดยใช้รูปแบบการกำหนดค่า (Key-Value)

1. การเพิ่มข้อมูลใหม่ สามารถเพิ่มข้อมูลใหม่ในติกชันนารีได้โดยการระบุ Key และกำหนดค่าให้กับ Key นั้น ๆ หาก Key ที่กำหนดดังไม่มีในติกชันนารี Python จะเพิ่ม Key-Value ใหม่เข้าไป

```
person = {"name": "Alice", "age": 25}
person["city"] = "New York" # เพิ่ม Key-Value ใหม่
print(person)

ผลลัพธ์

{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

คำสั่ง person["city"] = "New York" จะเพิ่ม Key "city" พร้อมกับค่า "New York" ลงในติกชันนารี person หาก Key "city" มีอยู่แล้ว จะเป็นการแก้ไขค่าเดิม

2. การแก้ไขข้อมูล หาก Key มีอยู่ใน딕ชันนารีอยู่แล้ว สามารถแก้ไขค่า (Value) ได้โดยการระบุ Key และกำหนดค่าที่ต้องการแทนค่าเดิม

```
person = {"name": "Alice", "age": 25}
person["age"] = 26 # แก้ไขค่าใน Key "age"
print(person)
```

ผลลัพธ์

```
{'name': 'Alice', 'age': 26}
```

คำสั่ง person["age"] = 26 จะเปลี่ยนค่าของ Key "age" จาก 25 เป็น 26

2.6.4 การลบข้อมูลใน딕ชันนารี

Python มีคำสั่งหลากหลายสำหรับการลบข้อมูลในลิสต์ ขึ้นอยู่กับสถานการณ์ เช่น ลบตามค่า ลบตามตำแหน่ง หรือ ลบข้อมูลทั้งหมดในลิสต์

1. การใช้ pop() พังค์ชัน pop() ใช้สำหรับลบข้อมูลโดยระบุ Key และคืนค่า (Value) ของ Key ที่ถูกลบออก หาก Key ที่ระบุไม่มีใน딕ชันนารี จะเกิดข้อผิดพลาด KeyError

```
person = {"name": "Alice", "age": 25, "city": "New York"}
person.pop("age") # ลบ Key "age"
print(person)
```

ผลลัพธ์

```
{'name': 'Alice', 'city': 'New York'}
```

คำสั่ง person.pop("age") จะลบ Key "age" และคืนค่า Value ที่ถูกลบ (25) ดิกชันนารี person จะเหลือเฉพาะ Key "name" และ "city"

2. การใช้ del คำสั่ง del ใช้สำหรับลบข้อมูลใน딕ชันนารีตาม Key ที่กำหนด หรือสามารถใช้ลบดิกชันนารีทั้งตัว

```
person = {"name": "Alice", "age": 25}
del person["age"] # ลบ Key "age"
print(person)
```

ผลลัพธ์

```
{'name': 'Alice'}
```

คำสั่ง del person["age"] จะลบ Key "age" และข้อมูลที่เกี่ยวข้องออกจากดิกชันนารี

3. การลบข้อมูลทั้งหมด พังค์ชัน clear() ใช้สำหรับลบข้อมูลทั้งหมดใน딕ชันนารี แต่ยังคงตัวดิกชันนารีอยู่ (ในรูปแบบดิกชันนารีว่าง)

```
person = {"name": "Alice", "age": 25}
person.clear()
print(person)
```

ผลลัพธ์

```
{}
```

คำสั่ง person.clear() จะลบข้อมูลทั้งหมดในติกชันนารี แต่ตัวแปร person ยังคงอยู่ในรูปแบบติกชันนารีว่าง ({}).

ตารางที่ 2.7 สรุปคำสั่งที่ใช้กับติกชันนารี

คำสั่ง	คำอธิบาย	ตัวอย่างโค้ด	ผลลัพธ์
dict()	สร้างติกชันนารี	<code>dict(name="Alice", age=25)</code>	{'name': 'Alice', 'age': 25}
[]	เข้าถึงข้อมูลโดยใช้ Key	<code>person["name"]</code>	"Alice"
get(key)	เข้าถึงข้อมูลด้วย Key	<code>person.get("age")</code>	25
pop(key)	ลบข้อมูลด้วย Key	<code>person.pop("city")</code>	{'name': 'Alice', 'age': 25}
del	ลบข้อมูลหรือทั้งติกชันนารี	<code>del person["age"]</code>	{'name': 'Alice'}
clear()	ลบข้อมูลทั้งหมดในติกชันนารี	<code>person.clear()</code>	{}
items()	คืนค่า Key-Value ทั้งหมด	<code>person.items()</code>	dict_items([('name', 'Alice')])
keys()	คืนค่า Key ทั้งหมดในรูปแบบลิสต์	<code>person.keys()</code>	dict_keys(['name', 'age', 'city'])

ติกชันนารีใน Python คือ "สมุดโทรศัพท์" ที่เก็บข้อมูลในรูปแบบคู่ของ "คีย์-ค่า" ซึ่งช่วยให้สามารถค้นหาข้อมูลได้อย่างรวดเร็วและตรงจุด ติกชันนารีเหมาะสมสำหรับการเก็บข้อมูลที่มีความสัมพันธ์กัน เช่น ชื่อ-เบอร์โทร หรือข้อมูล JSON ที่ใช้ในการพัฒนา Web API ซึ่งเป็นสิ่งที่โปรแกรมเมอร์ยุคใหม่ต้องใช้งานอยู่เป็นประจำ

2.7 ความผิดพลาดของโปรแกรม

การพัฒนาโปรแกรมด้วยภาษา Python ความผิดพลาด (Error) เป็นสิ่งที่เกิดขึ้นได้บ่อยและเป็นสิ่งที่นักพัฒนาจำเป็นต้องทำความเข้าใจและจัดการเพื่อให้โปรแกรมทำงานได้อย่างราบรื่น ความผิดพลาดในโปรแกรมสามารถแบ่งออกเป็น 3 ประเภทหลัก ดังนี้

2.7.1 Syntax Error

ความผิดพลาดทางไวยากรณ์ เกิดขึ้นเมื่อโค้ดที่เขียนไม่เป็นไปตามกฎไวยากรณ์ของภาษา Python โปรแกรมจะไม่สามารถรันได้จนกว่าจะแก้ไขข้อผิดพลาดนี้ให้ถูกต้อง

```
print("Hello World" # ขาดวงเล็บปิด
ผลลัพธ์

File "<stdin>", line 1
    print("Hello World"
          ^
SyntaxError: unexpected EOF while parsing
```

- ข้อผิดพลาดเกิดจากการลืมปิดวงเล็บ) ในคำสั่ง print() ซึ่งเป็นการละเมิดกฎไวยากรณ์ของ Python
- Python ต้องแก้ไขโดยเพิ่มวงเล็บปิดที่ขาดไป

2.7.2 Runtime Error

ความผิดพลาดขณะรันโปรแกรม เกิดขึ้นในขณะที่โปรแกรมกำลังทำงาน เช่น การหารด้วยศูนย์ การเข้าถึงดัชนีที่ไม่มีในลิสต์ หรือการใช้ตัวแปรที่ยังไม่ได้กำหนดค่า

1. การหารด้วยศูนย์

```
x = 10
y = 0
print(x / y) # การหารด้วยศูนย์
ผลลัพธ์
```

`ZeroDivisionError: division by zero`

- ข้อผิดพลาดเกิดจากการพยายามหารด้วยศูนย์ ซึ่งเป็นการดำเนินการที่ไม่สามารถทำได้
- ต้องตรวจสอบและป้องกันการหารด้วยศูนย์ เช่น ใช้คำสั่ง if y != 0 ก่อนดำเนินการหาร

2. การเข้าถึงดัชนีที่ไม่มีในลิสต์

ความผิดพลาดขณะรันโปรแกรม เกิดขึ้นในขณะที่โปรแกรมกำลังทำงาน เช่น การหารด้วยศูนย์ การเข้าถึงดัชนีที่ไม่มีในลิสต์ หรือการใช้ตัวแปรที่ยังไม่ได้กำหนดค่า

```
my_list = [1, 2, 3]
print(my_list[5]) # ดัชนีที่ไม่มีในลิสต์
ผลลัพธ์
```

`IndexError: list index out of range`

- ข้อผิดพลาดเกิดจากการพยายามเข้าถึงดัชนีที่อยู่นอกช่วงของลิสต์
- ต้องตรวจสอบดัชนีหรือความยาวของลิสต์ก่อนการเข้าถึงข้อมูล เช่น if len(my_list) > 5

2.7.3 Logical Error

ความผิดพลาดทางตรรกะ เกิดจากข้อผิดพลาดในการออกแบบหรือการเขียนโปรแกรมที่ทำให้ผลลัพธ์ไม่ตรงกับที่คาดหวัง แม้ว่าโปรแกรมจะสามารถรันได้โดยไม่มีข้อผิดพลาดใด ๆ

ตัวอย่างการคำนวณผลลัพธ์

```
x = 10
y = 20
average = x + y / 2 # คำนวณผลลัพธ์
print(average)
```

ผลลัพธ์

20.0

- ข้อผิดพลาดเกิดจากคำนวณผลลัพธ์ เนื่องจาก Python ดำเนินการหาร (/) ก่อนการบวก (+) ตามลำดับความสำคัญของตัวดำเนินการ
- ต้องแก้ไขโดยใช้วงเล็บเพื่อจัดลำดับการคำนวณให้ถูกต้อง เช่น average = (x + y) / 2

2.7.4 Type Error

ความผิดพลาดเกี่ยวกับชนิดข้อมูล เกิดจากการใช้ตัวดำเนินการกับข้อมูลที่ไม่เข้ากัน เช่น การบวกตัวเลขกับข้อความ

ตัวอย่างการบวกตัวเลขกับข้อความ

```
x = 10
y = "20"
print(x + y) # การบวกตัวเลขกับข้อความ
```

ผลลัพธ์

`TypeError: unsupported operand type(s) for +: 'int' and 'str'`

- ข้อผิดพลาดเกิดจากการพยายามรวมข้อมูลชนิด int กับ str ซึ่งไม่สามารถดำเนินการได้
- ต้องแปลงชนิดข้อมูลให้เข้ากันก่อน เช่น print(x + str(y))

2.7.5 Value Error

ความผิดพลาดเกี่ยวกับค่าที่ไม่ถูกต้อง เกิดขึ้นเมื่อฟังก์ชันได้รับค่าที่ไม่ถูกต้องหรือไม่สามารถแปลงค่าได้ ตัวอย่างการแปลงข้อความเป็นตัวเลข

```
x = "abc"
y = int(x) # การแปลงค่าที่ไม่ใช่ตัวเลข
```

ผลลัพธ์

`ValueError: invalid literal for int() with base 10: 'abc'`

- ข้อผิดพลาดเกิดจากการพยายามรวมข้อมูลชนิด int กับ str ซึ่งไม่สามารถดำเนินการได้
- ต้องแปลงชนิดข้อมูลให้เข้ากันก่อน เช่น print(x + int(y))

2.7.6 Name Error

ความผิดพลาดเกี่ยวกับตัวแปรหรือฟังก์ชันที่ไม่มีอยู่ เกิดจากการเรียกใช้ตัวแปรหรือฟังก์ชันที่ยังไม่ได้กำหนด

ตัวอย่าง การใช้ตัวแปรที่ยังไม่มีกำหนด

```
print(x) # ตัวแปร x ยังไม่มีกำหนด
```

ผลลัพธ์

```
NameError: name 'x' is not defined
```

- ข้อผิดพลาดเกิดจากการพยายามใช้ตัวแปร x ซึ่งยังไม่มีการกำหนดค่า
- ต้องตรวจสอบว่าตัวแปรได้รับการกำหนดค่าก่อนใช้งาน

ตารางที่ 2.8 สรุปประเภทข้อผิดพลาด

ประเภท ข้อผิดพลาด	คำอธิบาย	ตัวอย่างโค้ด	ผลลัพธ์
dict()	สร้างติกชันนารี	<code>dict(name="Alice", age=25)</code>	{'name': 'Alice', 'age': 25}
[]	เข้าถึงข้อมูลโดยใช้ Key	<code>person["name"]</code>	"Alice"
get(key)	เข้าถึงข้อมูลด้วย Key	<code>person.get("age")</code>	25
pop(key)	ลบข้อมูลด้วย Key	<code>person.pop("city")</code>	{'name': 'Alice', 'age': 25}
del	ลบข้อมูลหรือทั้งติกชันนารี	<code>del person["age"]</code>	{'name': 'Alice'}
clear()	ลบข้อมูลทั้งหมดในติกชันนารี	<code>person.clear()</code>	{}
items()	คืนค่า Key-Value ทั้งหมด	<code>person.items()</code>	<code>dict_items([('name', 'Alice')])</code>
keys()	คืนค่า Key ทั้งหมดในรูปแบบลิสต์	<code>person.keys()</code>	<code>dict_keys(['name', 'age', 'city'])</code>

การเขียนโปรแกรมไม่ใช่เรื่องของการหลีกเลี่ยงข้อผิดพลาด แต่เป็นการเรียนรู้และแก้ไขข้อผิดพลาด การทำความเข้าใจประเภทของข้อผิดพลาดต่างๆ เช่น Syntax Error หรือ Logic Error ตั้งแต่เนิ่นๆ จะช่วยให้พัฒนาโปรแกรมได้อย่างรวดเร็วและลดความกังวลเมื่อพบข้อผิดพลาด (Bug) ในการใช้งานจริง เพราะจะรู้วิธีรับมือและแก้ไขปัญหาได้อย่างเป็นระบบ

2.8 การเขียนโปรแกรมในกระบวนการคำนวณ

การเขียนโปรแกรมเป็นกระบวนการที่ช่วยแก้ปัญหาด้วยการใช้คอมพิวเตอร์ โดยมีขั้นตอนสำคัญที่ชัดเจน ตั้งแต่การออกแบบโปรแกรม การเขียนโค้ด การทดสอบ และการแก้ไขข้อผิดพลาด เพื่อให้ได้โปรแกรมที่ทำงานตามความต้องการอย่างมีประสิทธิภาพและแม่นยำ

การเขียนโปรแกรมในกระบวนการคำนวณมีเป้าหมายเพื่อให้ผู้อ่านสามารถสร้างโปรแกรมที่ทำงานได้ครบถ้วนใน 3 ขั้นตอนสำคัญ ได้แก่ การรับอินพุต (Input) การประมวลผล (Processing) และการแสดงผล เอาต์พุต (Output) พร้อมทั้งอธิบายองค์ประกอบเหล่านี้อย่างละเอียดเพื่อให้ผู้อ่านสามารถเข้าใจและนำไปประยุกต์ใช้ได้อย่างมีประสิทธิภาพ ตัวอย่างการนำไปใช้ อาทิเช่น

1. โปรแกรมคำนวณเกรดนักเรียน โปรแกรมนี้ช่วยในการประเมินผลการเรียนของนักเรียน โดยรับคะแนนสอบ คำนวนคะแนนรวม และกำหนดเกรดตามเกณฑ์ที่กำหนด ช่วยให้กระบวนการตรวจสอบผลการเรียนเป็นไปอย่างรวดเร็วและมีประสิทธิภาพ

2. โปรแกรมคิดเงินทอง ใช้ในงานร้านค้าเพื่อคำนวนเงินทองจากยอดชำระและเงินที่ลูกค้าจ่ายมา ช่วยลดความผิดพลาดในการคำนวนและเพิ่มความสะดวกให้กับพนักงานและลูกค้า

3. โปรแกรมแปลงหน่วยอุณหภูมิ โปรแกรมสำหรับการแปลงค่าอุณหภูมิ เช่น จากเซลเซียสเป็นฟาเรนไฮต์ หรือกลับกัน เหมาะสมสำหรับการใช้งานในบริบทที่ต้องการเปรียบเทียบหน่วยวัดในระบบที่ต่างกัน

4. โปรแกรมแสดงคำทักทายตามเวลาปัจจุบัน โปรแกรมนี้สามารถตรวจสอบเวลาปัจจุบันและแสดงคำทักทาย เช่น "สวัสดีตอนเช้า" หรือ "สวัสดีตอนเย็น" ตามช่วงเวลาของวัน เหมาะสมสำหรับระบบที่ต้องการสร้างปฏิสัมพันธ์ที่เป็นมิตรกับผู้ใช้งาน

2.8.1 การรับอินพุต (Input)

การรับอินพุตเป็นกระบวนการแรกในการเขียนโปรแกรมคำนวณ โดยโปรแกรมจะรวบรวมข้อมูลจากผู้ใช้งานหรือแหล่งข้อมูลภายนอก เช่น ค่าตัวเลข คำข้อความ หรือไฟล์ข้อมูล วิธีการรับอินพุตใน Python ตัวอย่างดังนี้

1. การรับอินพุตจากผู้ใช้งานผ่านคีย์บอร์ด ใช้ฟังก์ชัน `input()` เพื่อรับข้อมูลในรูปแบบข้อความ (String) และ สามารถแปลงข้อมูลให้เป็นชนิดที่เหมาะสม เช่น ตัวเลข ด้วยฟังก์ชัน `int()` หรือ `float()` เช่น

```
# รับค่าตัวเลขสองจำนวนจากผู้ใช้
num1 = float(input("กรุณาใส่ตัวเลขที่ 1: "))
num2 = float(input("กรุณาใส่ตัวเลขที่ 2: "))
```

โปรแกรมจะรับค่าจากผู้ใช้งานและแปลงเป็นตัวเลขทศนิยมก่อนนำไปใช้งานในกระบวนการต่อไป

2. การรับอินพุตหลายค่าในครั้งเดียว ใช้ฟังก์ชัน `input()` ร่วมกับการแยกข้อมูลโดยใช้ `split()`

```
# รับตัวเลขสองจำนวนในบรรทัดเดียว
num1, num2 = map(float, input("กรุณาใส่ตัวเลขสองจำนวนกันด้วยช่องว่าง: ").split())
```

โปรแกรมจะรับค่าตัวเลขจากผู้ใช้งานสองจำนวนในครั้งเดียวและแยกค่าโดยใช้ช่องว่างเป็นตัวแบ่ง

3. การรับอินพุตจากไฟล์ ใช้คำสั่งเปิดไฟล์ (open) เพื่ออ่านข้อมูลจากไฟล์แทนการป้อนค่าผ่านคีย์บอร์ด

```
# อ่านข้อมูลจากไฟล์
with open("input.txt", "r") as file:
    data = file.readlines()
num1 = float(data[0])
num2 = float(data[1])
```

โปรแกรมจะอ่านข้อมูลจากไฟล์ input.txt โดยสมมติว่าตัวเลขแต่ละค่าถูกเขียนในบรรทัดแยกกัน

4. การรับอินพุตนิดข้อมูลที่หลากหลาย สามารถรับข้อมูลที่ไม่ใช่ตัวเลข เช่น รายชื่อ หรือค่าที่เป็น Boolean ได้

```
# รับข้อมูลชนิดข้อความและ Boolean
name = input("กรุณาใส่ชื่อของคุณ: ")
is_member = input("คุณเป็นสมาชิกหรือไม่(yes/no): ").lower() == "yes"
```

โปรแกรมจะรับชื่อเป็นข้อความ และตรวจสอบสถานะสมาชิกโดยเปรียบเทียบคำตอบ

ข้อควรระวัง ความผิดพลาด TypeError ที่เกิดขึ้นเมื่อพยายามนำข้อมูลที่รับจาก input() ไปดำเนินการคำนวนโดยตรง เนื่องจากค่าที่ได้จาก input() จะอยู่ในรูปแบบ ข้อความ (string) โดยค่าเริ่มต้นดังนี้ หากต้องการดำเนินการทางคณิตศาสตร์ จะต้องแปลงค่าเป็นชนิดข้อมูลที่เหมาะสม เช่น int หรือ float

```
temp_string = input() # รับค่าจากผู้ใช้
temp_string + 5 # เกิด TypeError
```

เมื่อผู้ใช้ป้อนค่า เช่น 35 ค่าในตัวแปร temp_string จะเป็น '35' (string) การนำ string บวกกับจำนวนเต็ม (int) จะทำให้เกิดข้อผิดพลาด TypeError เพราะชนิดข้อมูลไม่เข้ากัน

การแก้ไขปัญหา ใช้ฟังก์ชัน float() หรือ int() เพื่อแปลงค่าจาก string เป็นชนิดข้อมูลที่สามารถคำนวนได้

```
temp_string = input() # รับค่าจากผู้ใช้
fahrenheit = float(temp_string) # แปลงเป็น float
fahrenheit + 5 # สามารถคำนวนได้
```

2.8.2 การประมวลผล (Processing)

การประมวลผลเป็นกระบวนการที่โปรแกรมจะดำเนินการคำนวนหรือดำเนินการตามคำสั่งที่ผู้พัฒนากำหนด โดยใช้ข้อมูลที่ได้รับจากขั้นตอนการรับอินพุต ขั้นตอนสำคัญในการประมวลผล ได้แก่

1. คำนวนทางคณิตศาสตร์ ใช้ตัวดำเนินการทางคณิตศาสตร์ เช่น บวก (+), ลบ (-), คูณ (*), และหาร (/) ซึ่งได้กล่าวไว้แล้วในหัวข้อที่ 2.4

2. การใช้โครงสร้างควบคุมการไหล (Flow Control) ใช้คำสั่ง if, else, หรือการวนซ้ำ (for, while) เพื่อประมวลผลข้อมูลที่ซับซ้อน โดยจะกล่าวเพิ่มเติมในบทที่ 3

3. การจัดเก็บผลลัพธ์ เก็บค่าผลลัพธ์ในตัวแปรเพื่อใช้งานในขั้นตอนถัดไป

ในตัวอย่างนี้ โปรแกรมจะคำนวณผลรวมและผลต่างของตัวเลขสองจำนวนที่ได้รับจากผู้ใช้งาน และจัดเก็บในตัวแปรเพื่อใช้งานต่อไป

```
# คำนวณผลรวมและผลต่างของตัวเลข
sum_result = num1 + num2
diff_result = num1 - num2
```

2.8.3 การแสดงผลเอาต์พุต (Output)

การแสดงผลเอาต์พุตเป็นขั้นตอนสุดท้ายในกระบวนการการเขียนโปรแกรมคำนวณ โดยโปรแกรมจะแสดงผลลัพธ์ที่ได้จากการบวกและลบให้ผู้ใช้งานรับทราบ ใช้ฟังก์ชัน print() เพื่อแสดงข้อความหรือข้อมูล และ สามารถจัดรูปแบบข้อความให้เหมาะสมด้วย f-string หรือการเชื่อมข้อความ ซึ่งได้กล่าวไว้ในหัวข้อที่ 2.3.5 – 2.3.6

```
# แสดงผลลัพธ์
print(f"ผลรวมของตัวเลขคือ: {sum_result}")
print(f"ผลต่างของตัวเลขคือ: {diff_result}")
```

ในตัวอย่างนี้ โปรแกรมจะแสดงผลรวมและผลต่างของตัวเลขที่คำนวณไว้ในรูปแบบข้อความที่เข้าใจง่าย

1. ตัวอย่างโปรแกรมกระบวนการคำนวณครบทั้ว

```
# รับอินพุตจากผู้ใช้งาน
num1 = float(input("กรุณาใส่ตัวเลขที่ 1: "))
num2 = float(input("กรุณาใส่ตัวเลขที่ 2: "))

# ประมวลผลข้อมูล
sum_result = num1 + num2
diff_result = num1 - num2

# แสดงผลลัพธ์
print(f"ผลรวม: {sum_result}")
print(f"ผลต่าง: {diff_result}")
```

ผลลัพธ์

กรุณาใส่ตัวเลขที่ 1: 10

กรุณาใส่ตัวเลขที่ 2: 5

ผลรวม: 15.0

ผลต่าง: 5.0

2. ตัวอย่างโปรแกรมทักทาย

```
# แสดงข้อความให้ผู้ใช้กรอกข้อ
print("Enter your name: ")
```

```
# รับค่าชื่อจากผู้ใช้ผ่านฟังก์ชัน input()
name = input()
# ใช้ f-string เพื่อสร้างข้อความทักทาย โดยนำชื่อที่รับมาใส่ในข้อความ
print(f"Hi {name}, how are you today?")
```

print("Enter your name: ") ใช้คำสั่ง print เพื่อแสดงข้อความบนหน้าจอว่า "Enter your name:" เพื่อบอกให้ผู้ใช้ทราบว่าต้องกรอกชื่อ

name = input() ฟังก์ชัน input() จะรอรับค่าจากผู้ใช้ผ่านคีย์บอร์ด เมื่อผู้ใช้พิมพ์ข้อความแล้วกด Enter ข้อความนั้นจะถูกเก็บในตัวแปร name

print(f"Hi {name}, how are you today?") ใช้ฟังก์ชัน print() เพื่อแสดงข้อความที่รวมค่าจากตัวแปร name ด้วยการใช้ฟอร์แมตของ f-string ("ข้อความ") โดย {name} จะถูกแทนค่าด้วยข้อความที่ผู้ใช้พิมพ์

ผลลัพธ์ เมื่อรันโปรแกรม จะแสดงข้อความ

Enter your name:

เมื่อผู้ใช้พิมพ์ชื่อ เช่น Sarayut Gonwirat และกด Enter โปรแกรมจะนำค่าที่รับมาเก็บในตัวแปร name

Sarayut Gonwirat

โปรแกรมจะแสดงผลลัพธ์

Hi Sarayut Gonwirat, how are you today?

การให้โปรแกรมแสดงผลข้อความที่งมงดในบรรทัดเดียวโดยใช้ฟังก์ชัน print() พร้อมพารามิเตอร์ end="" เพื่อไม่ให้ขึ้นบรรทัดใหม่หลังการแสดงผลแต่ละครั้ง

```
# แสดงข้อความให้ผู้ใช้กรอกชื่อ และดึงข้อความโดยไม่ขึ้นบรรทัดใหม่
print("Enter your name: ", end="")
name = input()
# ใช้ f-string เพื่อสร้างข้อความทักทาย โดยนำชื่อที่รับมาใส่ในข้อความ
print(f"Hi {name}, how are you today?")
```

เมื่อรันโปรแกรม จะแสดงข้อความ เมื่อผู้ใช้พิมพ์ชื่อ เช่น Sarayut Gonwirat และกด Enter โปรแกรมจะนำค่าที่รับมาเก็บในตัวแปร name ผลลัพธ์สุดท้าย

Enter your name: Sarayut Gonwirat

Hi Sarayut Gonwirat, how are you today?

ตัวอย่างเพิ่มเติม

```
print("My ", end="") # แสดง "My " โดยไม่ขึ้นบรรทัดใหม่
print("name ", end="") # แสดง "name " โดยไม่ขึ้นบรรทัดใหม่
print("is ", end="") # แสดง "is " โดยไม่ขึ้นบรรทัดใหม่
print("Sarayut.", end="") # แสดง "Sarayut." และไม่ขึ้นบรรทัดใหม่
ผลลัพธ์
```

My name is Sarayut.

ตัวอย่างการใช้ฟังก์ชัน `input()` แบบ single-line โดยกำหนดข้อความ `prompt` ไว้ในฟังก์ชัน ซึ่งทำให้ได้กรอบและแสดงผลในบรรทัดเดียว

```
name = input("Enter your name: ") # กำหนดข้อความใน input() เพื่อแสดง prompt
print(name) # แสดงค่าที่รับจาก input()
```

ฟังก์ชัน `input("Enter your name: ")` ใช้แสดงข้อความ "Enter your name: " และรับข้อมูลจากผู้ใช้ โดยค่าที่กรอกจะถูกเก็บในตัวแปร `name` คำสั่ง `print(name)` ใช้แสดงค่าที่ถูกเก็บในตัวแปร `name` บนหน้าจอ

3. โปรแกรมแปลงอุณหภูมิ การแปลงอุณหภูมิจากฟาเรนไฮต์ (Fahrenheit) ไปเป็นเซลเซียส (Celsius) โดยใช้สูตรดังนี้

$$\text{Celsius} = \frac{(\text{Fahrenheit} - 32) \times 5}{9}$$

```
print("This program converts Fahrenheit to Celsius.")

# รับค่าอุณหภูมิในหน่วยฟาเรนไฮต์จากผู้ใช้
fahrenheit = float(input("Type in a temperature in Fahrenheit:"))

# คำนวณอุณหภูมิในหน่วยเซลเซียส
celsius = (fahrenheit - 32) * 5.0 / 9

# แสดงผลลัพธ์
print("That is ", end="")
print(celsius, end="")
print(" degrees Celsius.")
```

- โปรแกรมแสดงข้อความเพื่อบอกผู้ใช้ว่าเป็นโปรแกรมแปลงอุณหภูมิจากฟาเรนไฮต์ (Fahrenheit) เป็นเซลเซียส (Celsius)
- รับค่าอุณหภูมิจากผู้ใช้ผ่านฟังก์ชัน `input()` และแปลงเป็นจำนวนจริง (`float`) ด้วย `float()`
- ใช้สูตร $(\text{Fahrenheit} - 32) \times 5/9$ คำนวณอุณหภูมิในเซลเซียส และเก็บในตัวแปร `Celsius`
- แสดงข้อความผลลัพธ์และค่าที่คำนวณได้ในบรรทัดเดียวทันทีโดยใช้ `end=""`
- เพิ่มข้อความ "degrees Celsius." เพื่อบอกหน่วยของผลลัพธ์อย่างชัดเจน

กระบวนการรับข้อมูลเข้า การประมวลผล และการแสดงผลลัพธ์ คือหัวใจของการเขียนโปรแกรมในทุกภาษา การฝึกฝนทักษะเหล่านี้อย่างสม่ำเสมอจะช่วยพัฒนาความสามารถในการออกแบบโครงสร้างโปรแกรมให้มีประสิทธิภาพมากยิ่งขึ้น และยังช่วยให้คิดอย่างเป็นลำดับขั้นตอน เมื่ອนกับการวางแผนงานในชีวิตประจำวัน

บทสรุป

ในโลกยุคดิจิทัลที่เทคโนโลยีเข้ามามีบทบาทอย่างลึกซึ้งในชีวิตประจำวัน การเริ่มต้นเรียนรู้การเขียนโปรแกรมด้วยภาษา Python ถือเป็นก้าวแรกที่สำคัญยิ่งสู่โลกแห่งการพัฒนาเทคโนโลยี ด้วยความที่ Python เป็นภาษาที่เรียนรู้ได้ง่าย มีโครงสร้างที่กระชับ และเป็นที่นิยมอย่างแพร่หลายในการสร้างสรรค์โปรแกรม หลากหลายประเภท ไม่ว่าจะเป็นแอปพลิเคชัน การวิเคราะห์ข้อมูล หรือแม้แต่การพัฒนาปัญญาประดิษฐ์ (AI)

เนื้อหาในบทนี้มุ่งเน้นการวางแผนฐานข้อมูลที่มั่นคงสำหรับผู้เริ่มต้น โดยครอบคลุมตั้งแต่การติดตั้งและใช้งานโปรแกรมสำหรับพัฒนาโปรแกรม (IDE) เช่น IDLE, VS Code, Jupyter Notebook และ Google Colab ไปจนถึงการเริ่มต้นเขียนโปรแกรม Python แบบพื้นฐาน เช่น การแสดงข้อความ การคำนวณเบื้องต้น และการใช้งานตัวแปรกับชนิดข้อมูลต่างๆ

นอกจากนี้ ยังจะได้เรียนรู้แนวคิดเกี่ยวกับโครงสร้างข้อมูลพื้นฐานอย่างลิสต์ (List) และดิกชันนารี (Dictionary) พร้อมทั้งทำความเข้าใจเกี่ยวกับการจัดการข้อมูลของโปรแกรมในเบื้องต้น เพื่อให้ผู้เรียนได้มีทักษะการเขียนโปรแกรมที่ครอบคลุมในกระบวนการคำนวณ ตั้งแต่การรับข้อมูลเข้า การประมวลผล และการแสดงผลลัพธ์ ซึ่งจะนำไปสู่การพัฒนาโปรแกรมที่มีความซับซ้อนมากยิ่งขึ้นในอนาคต

- IDE เช่น PyCharm, Visual Studio Code, และ Jupyter Notebook ช่วยให้การเขียนโปรแกรมสะดวกและมีประสิทธิภาพมากขึ้นด้วยเครื่องมือครบครัน เช่น Debugger และ Text Editor
- การติดตั้ง Python IDE ต้องเลือกตามความเหมาะสม เช่น Visual Studio Code สำหรับการเขียนทั่วไป และ Jupyter Notebook สำหรับงาน Data Science
- ใช้ IDLE Shell เพื่อทดลองโค้ดเบื้องต้นและแสดงผลแบบทันที
- การเขียนไฟล์ Python (.py) ช่วยเก็บและจัดการโปรแกรมที่ซับซ้อนได้ดีขึ้น
- ตัวแปรใน Python ใช้เก็บข้อมูล เช่น int, float, str, list, และ dict
- ใช้คำสั่ง type() เพื่อตรวจสอบชนิดข้อมูลของตัวแปร
- สามารถแปลงชนิดข้อมูลระหว่างกันได้ เช่น int(), float(), และ str()
- การดำเนินการทางคณิตศาสตร์ ตัวดำเนินการสำคัญ + (บวก) - (ลบ) * (คูณ) / (หาร) // (หารปัดเศษ) % (หารเอาเศษ) และ ** (ยกกำลัง)
- ลำดับการคำนวณตาม PEMDAS: วงเล็บ → ยกกำลัง → คูณ/หาร → บวก/ลบ
- การใช้ E-notation สำหรับแสดงค่าตัวเลขขนาดใหญ่หรือเล็กมาก
- ลิสต์ คือ โครงสร้างข้อมูลที่เก็บรายค่าในตัวแปรเดียว โดยใช้ดัชนีในการเข้าถึง รองรับการเพิ่ม (append) แทรก (insert) และลบข้อมูล (pop, remove, del)
- ลิสต์ sort สำหรับเรียงลำดับ และ reverse สำหรับกลับลำดับข้อมูล
- ดิกชันนารี สามารถเก็บข้อมูลในรูปแบบคู่ (Key-Value) เช่น {"name": "Alice", "age": 25}
- ดิกชันนารี ใช้ Key เพื่อเข้าถึงหรือแก้ไขข้อมูล คำสั่งสำคัญ get, pop, del, clear, และ items

- ความผิดพลาดของโปรแกรม
 - Syntax Error: เกิดจากการเขียนโค้ดผิดตามกฎไวยากรณ์ เช่น ลีมวงเล็บปิด
 - Runtime Error: เกิดจากปัญหาระหว่างรันโปรแกรม เช่น การหารด้วยศูนย์
 - Logical Error: การออกแบบหรือคำนวนผิดทำให้ผลลัพธ์ไม่ถูกต้อง
 - Type Error: การใช้ตัวดำเนินการกับข้อมูลที่ไม่เข้ากัน เช่น บวกตัวเลขกับข้อความ
 - Value Error: การแปลงค่าที่ไม่ถูกต้อง เช่น แปลงข้อความที่ไม่ใช่ตัวเลข
 - Name Error: การใช้ตัวแปรที่ไม่ได้ประกาศ
- การจัดการข้อความ (String) พังก์ชันสำหรับจัดการข้อความ: upper(), lower(), strip(), และ title() ใช้ f-string หรือ str.format() สำหรับการจัดรูปแบบข้อความ และใช้ \n และ \t สำหรับการขึ้นบรรทัดใหม่ และเว้นวรรค
- การแปลงจาก Integer เป็น Float, String และในทางกลับกัน เพื่อให้สอดคล้องกับการดำเนินการข้อผิดพลาดเกิดขึ้นเมื่อแปลงค่าที่ไม่สามารถทำได้ เช่น แปลงข้อความเป็นตัวเลขที่ไม่ใช่ตัวเลขจริง
- การเขียนโปรแกรมในกระบวนการคำนวนประกอบด้วย 3 ขั้นตอนสำคัญ ได้แก่ การรับอินพุต การประมวลผล และการแสดงผลเอาต์พุต แต่ละขั้นตอนมีความสำคัญและมีวิธีการดำเนินการที่หลากหลาย การทำความเข้าใจและปฏิบัติตามกระบวนการเหล่านี้จะช่วยให้การเขียนโปรแกรมมีประสิทธิภาพและสามารถแก้ปัญหาได้อย่างถูกต้องและรวดเร็ว

คำถามท้ายบท

1. Integrated Development Environment (IDE) คืออะไร และมีบทบาทอย่างไรในการพัฒนาโปรแกรม?
2. ยกตัวอย่างโปรแกรม IDE ที่นิยมใช้ในการเขียนภาษา Python มา 3 ตัว พร้อมอธิบายความสามารถเด่นของแต่ละโปรแกรม
3. พังก์ชัน print() ใช้ทำอะไรในภาษา Python และจะเขียนตัวอย่างการใช้งาน
4. อธิบายความแตกต่างระหว่าง ลิสต์ (List) และ ดิกชันนารี (Dictionary) ในภาษา Python
5. Syntax Error คืออะไร และเกิดขึ้นในสถานการณ์ใด?
6. อธิบายความแตกต่างระหว่างตัวแปรชนิด int และ float พร้อมตัวอย่างการใช้งาน
7. ถ้าต้องการให้ข้อความในโปรแกรมขึ้นบรรทัดใหม่ ควรใช้คำสั่งอะไร?
8. จะเขียนโปรแกรมเพื่อแสดงผลข้อความต่อไปนี้

My name is John Doe.
I love programming.
9. ถ้าต้องการสร้างตัวแปรที่มีค่าเป็น “Python Programming” ควรเขียนอย่างไร?
10. ชนิดข้อมูล Boolean ในภาษา Python มีค่าได้บ้าง? และใช้ในสถานการณ์แบบใด?
11. ตัวดำเนินการ (Operator) // และ % ใน Python ใช้ทำอะไร? พร้อมตัวอย่างการใช้งาน

12. จะเขียนโปรแกรมเพื่อคำนวณจำนวนวินาทีในหนึ่งวัน (24 ชั่วโมง)
13. ลำดับการคำนวณ (Order of Operation) ใน Python เป็นอย่างไร และตัวดำเนินการใดที่มีลำดับความสำคัญสูงสุด?
14. จะเขียนโปรแกรมเพื่อคำนวณพื้นที่ของวงกลมเมื่อรัศมีเท่ากับ 5 (พื้นที่ = πr^2 โดยใช้ math.pi)
15. พังก์ชัน append() และ extend() แตกต่างกันอย่างไร? ยกตัวอย่างการใช้งาน
16. ถ้าต้องการลบข้อมูลในลิสต์เฉพาะตำแหน่งที่ 2 ควรใช้คำสั่งใด?
17. จะเขียนโปรแกรมที่สร้างลิสต์ [10, 20, 30, 40] และเพิ่มค่าตัวเลข 50 ลงในลิสต์
18. ดิกชันนารี (Dictionary) คืออะไร และใช้งานในกรณีใดบ้าง?
19. หากต้องการเพิ่ม Key-Value คู่ใหม่ในดิกชันนารีที่มีอยู่แล้ว ควรใช้คำสั่งอย่างไร?
20. จะเขียนโปรแกรมเพื่อสร้างดิกชันนารีที่เก็บชื่อและอายุของบุคคล พร้อมเพิ่มข้อมูลของบุคคลใหม่

โจทย์การเขียนโปรแกรม

1. Python มีชนิดข้อมูลอะไรบ้าง? ยกตัวอย่างข้อมูล 1 ค่า และระบุชนิดข้อมูลนั้น
2. เขียนคำสั่ง print(type(...)) เพื่อตรวจสอบชนิดข้อมูลของตัวเลข 123.45 และระบุผลลัพธ์ที่ได้
3. เขียนโปรแกรมคำนวณผลลัพธ์ของ $3 + 4 \times 2$ และระบุลำดับการคำนวณที่ Python ใช้
4. เขียนโปรแกรมสร้างตัวแปร 3 ตัว ได้แก่ name, age, และ city พิร้อมกำหนดค่าเริ่มต้น และแสดงผลข้อมูลทั้งหมด
5. ใช้ฟังก์ชัน f-string เพื่อสร้าง "My name is [ชื่อ], I am [อายุ] years old, and I live in [เมือง]"
6. เขียนโปรแกรมเพื่อรับชื่อ อายุ และเมือง จากผู้ใช้งาน และแสดงข้อความด้วย f-string
7. เขียนโปรแกรมรับตัวเลขจำนวนเต็ม 3 ค่า และคำนวณค่าเฉลี่ยของตัวเลขเหล่านั้น
8. เขียนโปรแกรมรับค่าอุณหภูมิเป็นองศาเซลเซียส และแปลงเป็นฟาเรนไฮต์ ($F = C \times 9/5 + 32$)
9. เขียนโปรแกรมคำนวณเส้นรอบรูปของสี่เหลี่ยมโดยรับค่าความกว้างและความยาวจากผู้ใช้
10. เขียนโปรแกรมรับค่ารายการอาหาร คำนวนภาษี 7% และรับค่าทิป แสดงผลรวมค่าใช้จ่ายทั้งหมด
11. เขียนโปรแกรมเพื่อคำนวณการถอนเงินเป็นธนาบัตรและเหรียญไทย เช่น 500, 100, 50, 20, 10, 5 และ 1 บาท
12. จงระบุผลลัพธ์ของคำสั่งต่อไปนี้ และอธิบายลำดับการคำนวณใน Python

```
print(5 + 3 * 2 - 8 / 4)
```

13. ระบบข้อผิดพลาดในโค้ดต่อไปนี้ และแก้ไขให้ถูกต้อง

```
print("Hello World")
```

14. อธิบายเหตุผลที่เกิดข้อผิดพลาดในโค้ดนี้ และวิธีแก้ไข

```
x = 10
y = 0
print(x / y)
```

15. ในโค้ดนี้ ผลลัพธ์ที่ได้มีตรงกับที่คาดหวัง อธิบายเหตุผล

```
x = 5  
y = 10  
average = x + y / 2  
print(average)
```

16. เขียนโปรแกรมสร้างลิสต์ของตัวเลข 5 ตัว และแสดงผลลัพธ์รวมของตัวเลขในลิสต์
17. ใช้ฟังก์ชัน append() เพิ่มข้อมูลใหม่ในลิสต์เดิม และแสดงผลลัพธ์
18. เขียนโค้ดสร้างติกซันนารีของข้อมูลส่วนตัว และลบข้อมูลใน Key age
19. จงอธิบายผลลัพธ์ของคำสั่งต่อไปนี้

```
print(1.2e3 + 4.5e2)
```

20. เขียนโปรแกรมแสดงข้อความต่อไปนี้โดยใช้คำสั่ง print() และ \n

My Profile:

Name: John
Age: 25
Favorite Color: Blue

บทที่ 3

การควบคุมโปรแกรม

เนื้อหาบทเรียน

- 3.1 การเขียนผังงาน (Flowchart)
- 3.2 การเขียนโปรแกรมแบบเงื่อนไข
- 3.3 การเขียนโปรแกรมแบบวนซ้ำ
- 3.4 การเขียนโปรแกรมเพิ่มเติมการสำหรับการควบคุมโปรแกรม

วัตถุประสงค์การเรียนรู้

- บอกร่องรอยในผังงาน (Flowchart)
- สามารถเขียนโปรแกรมตามผังงานได้
- เข้าใจการควบคุมการไหลของโปรแกรม (Flow Control) เปรียบเทียบ Sequence กับ Decision และใช้งาน Boolean (and, or, not) ได้
- เขียนโปรแกรมด้วย if, if-else, if-elif และจัดการ Nested Conditions ได้อย่างเหมาะสม
- เลือกใช้งานลูป while และ for พร้อมเปรียบเทียบความแตกต่างเพื่อแก้ปัญหาต่าง ๆ
- ใช้คำสั่ง break และ continue ในการควบคุมการทำงานของลูปได้อย่างมีประสิทธิภาพ
- สร้างโปรแกรมที่ใช้ลูปซ้ำงาน (เช่น ตารางสูตรคูณ หรือ เกมไทยตัวเลข) ออกแบบโดยไม่ซ้ำซ้อน

ในการพัฒนาโปรแกรมคอมพิวเตอร์ การควบคุมการไหลของโปรแกรม (Flow Control) ถือเป็นหัวใจสำคัญที่ทำให้โปรแกรมสามารถทำงานได้อย่างถูกต้องและมีประสิทธิภาพสูงสุด โดยช่วยกำหนดลำดับการทำงานของคำสั่งต่าง ๆ เพื่อให้โปรแกรมตอบสนองต่อข้อมูลหรือสถานการณ์ที่หลากหลายได้อย่างยืดหยุ่น ตัวอย่างเช่น หากผู้ใช้กรอกข้อมูลผิด โปรแกรมสามารถแสดงคำเตือนเพื่อให้แก้ไข หรือเมื่อคำนวณรายการสินค้าในตะกร้า โปรแกรมจะวนซ้ำเพื่อนับจำนวนรายการทั้งหมดอย่างถูกต้อง การควบคุมการไหลของโปรแกรมสามารถแบ่งออกเป็น 2 ส่วนสำคัญ ดังนี้

1. การตัดสินใจหรือเงื่อนไข (Decision หรือ Condition) การตัดสินใจเป็นกระบวนการที่โปรแกรมเลือกเส้นทางการทำงานที่เหมาะสมตามเงื่อนไข เช่น หากเงื่อนไขเป็นจริง (True) โปรแกรมจะดำเนินการชุดคำสั่งที่กำหนดไว้ และหากเงื่อนไขเป็นเท็จ (False) จะเลือกอีกเส้นทางหนึ่ง โครงสร้างคำสั่งที่ใช้บ่อยได้แก่ if-else หรือ switch ตัวอย่างเช่น การตรวจสอบรหัสผ่านผู้ใช้ หากข้อมูลที่ป้อนถูกต้อง โปรแกรมจะอนุญาตให้เข้าถึงระบบ แต่หากไม่ถูกต้อง โปรแกรมจะแสดงข้อความแจ้งเตือนทันที ดังตัวอย่างรูปไฟที่ต้องเลือกเส้นทางในรูป



รูปที่ 3.1 เงื่อนไขหรือทางเลือกของการควบคุมโปรแกรม

ที่มา Sande, W., & Sande, C. (2020). Hello World



รูปที่ 3.2 การวนซ้ำเมื่อมีรถที่วิ่งในรอบที่กำหนด

ที่มา Sande, W., & Sande, C. (2020). Hello World

2. การทำซ้ำ (Repetition) การทำซ้ำช่วยลดความซ้ำซ้อนของโค้ด และทำให้โปรแกรมสามารถทำงานซ้ำในลักษณะเดียวกันหลาย ๆ ครั้งได้จนกว่าจะครบเงื่อนไขที่กำหนด โครงสร้างคำสั่งที่นิยมใช้ เช่น for, while หรือ do-while ตัวอย่างเช่น การวนลูปเพื่อคำนวณยอดรวมของสินค้าในตะกร้าสินค้าหรือการแสดงผลข้อมูลที่มีหลายรายการในรูปแบบของรายงาน

โครงสร้างพื้นฐานของโปรแกรมการให้ผลของชุดคำสั่ง ช่วยกำหนดขั้นตอนการทำงานของโปรแกรมในลักษณะที่ชัดเจนและยึดหยุ่น โดยสามารถนำไปประยุกต์ใช้กับงานต่าง ๆ ได้อย่างเหมาะสม ดังนี้

1. โครงสร้างที่ทำงานเรียงลำดับจากบันลุงล่าง (Sequence) ไม่มีการตัดสินใจหรือวนซ้ำ เหมาะกับกระบวนการที่เรียบง่ายและไม่ซับซ้อน เช่น รับข้อมูล คำนวณผลลัพธ์ และแสดงผลลัพธ์ตามลำดับ

2. การตัดสินใจ (Decision) คือ โครงสร้างที่ให้โปรแกรมตัดสินใจตามเงื่อนไข เช่น เป็นจริง (True) หรือ เป็นเท็จ (False) เพื่อเลือกขั้นตอนถัดไปที่เหมาะสม เช่น รับข้อมูล ตรวจสอบเงื่อนไข แล้วเลือกคำนวนแบบที่ 1 หากเงื่อนไขเป็น True หรือคำนวนแบบที่ 2 หากเป็น False

3. ลูปหรือการวนซ้ำ คือ โครงสร้างที่ใช้ทำซ้ำกระบวนการเดิมหลายครั้งจนกว่าเงื่อนไขจะเป็น False เหมาะสำหรับงานที่ต้องทำงานซ้ำ เช่น รับข้อมูล ตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็น True จะทำซ้ำขั้นตอนเดิม และถ้าเป็น False จะออกจากลูปและดำเนินการขั้นตอนถัดไป

3.1 การเขียนผังงาน (Flowchart)

Flowchart หรือ ผังงาน หมายถึง แผนภาพ (Diagram) ที่ใช้แสดงลำดับขั้นตอนการทำงานของกระบวนการ (Process) อย่างเป็นระบบและมีโครงสร้างชัดเจน โดยใช้สัญลักษณ์รูปทรงเรขาคณิตมาตรฐานร่วมกับลูกศรหรือเส้นเชื่อม เพื่อสะท้อนถึงการให้ผลของข้อมูล การตัดสินใจ และการดำเนินงานในแต่ละขั้นตอน การจัดทำ Flowchart ช่วยให้ผู้เกี่ยวข้องสามารถเข้าใจภาพรวมของกระบวนการได้ง่ายขึ้น ลดความคลุมเครือในการสื่อสาร และสนับสนุนการวิเคราะห์ปรับปรุงกระบวนการอย่างเป็นรูปธรรม

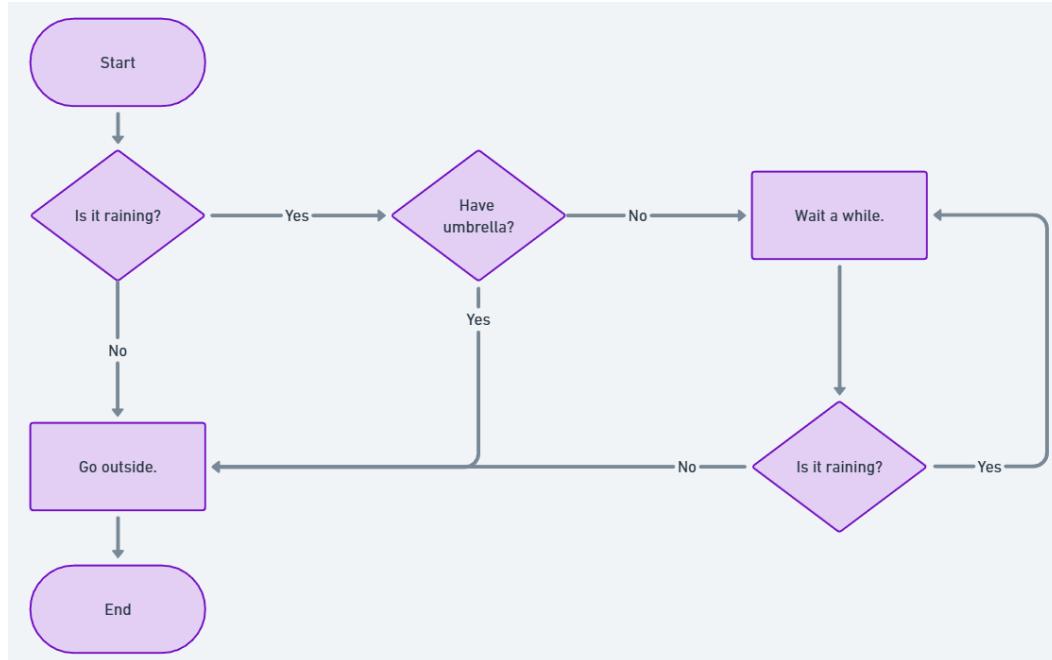
ประโยชน์ของการใช้ Flowchart

1. เพิ่มประสิทธิภาพในการบริหารจัดการ การมองเห็นภาพรวมของกระบวนการผ่าน Flowchart ช่วยให้ผู้บริหารวางแผน แบ่งทรัพยากร และกำหนดแนวทางการดำเนินงานได้อย่างรอบคอบ ลดค่าใช้จ่ายและเวลาในการผลิตหรือพัฒนางาน

2. ส่งเสริมการทำงานเป็นทีม เมื่อทุกคนมีความเข้าใจกระบวนการในทิศทางเดียวกันด้วย Flowchart ยอมช่วยลดความขัดแย้งและข้อโต้แย้งในทีมงาน โดยเฉพาะโครงการขนาดใหญ่ที่มีผู้เกี่ยวข้องหลายฝ่าย

3. เป็นเครื่องมือช่วยสอนและฝึกอบรม Flowchart สามารถใช้เป็นสื่อการเรียนรู้ในการอธิบายแนวทางปฏิบัติงาน หรือการทำงานของระบบได้อย่างมีประสิทธิภาพ เหมาะสมสำหรับการฝึกอบรมบุคลากรใหม่ เพื่อให้เข้าใจขั้นตอนการทำงานได้เร็วขึ้น

4. ต่อยอดสู่การปรับปรุงและพัฒนาวัตกรรม เมื่อสามารถระบุโครงสร้างและขั้นตอนของกระบวนการได้อย่างละเอียด ผู้มีส่วนร่วมจะเห็นโอกาสในการปรับปรุง เพิ่มฟังก์ชัน หรือสร้างวัตกรรมใหม่ได้ง่ายขึ้น เป็นประโยชน์ต่อการพัฒนาวัตกรรมเชิงองค์กรและการเพิ่มมูลค่าในระยะยาว

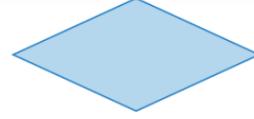


รูปที่ 3.3 ตัวอย่างภาพ Flowchart

Flowchart นี้ช่วยแนะนำวิธีการตัดสินใจอย่างง่ายในกรณีที่ฝนตก โดยให้คำแนะนำทั้งในกรณีที่มีร่มหรือไม่มีร่ม เพื่อความสะดวกและปลอดภัยในการออกไปข้างนอก Flowchart นี้แสดงกระบวนการตัดสินใจว่าจะออกไปข้างนอกหรือไม่ โดยมีขั้นตอนดังนี้

1. เริ่มต้น (Start) เริ่มต้นกระบวนการตรวจสอบสภาพอากาศ
2. ตรวจสอบว่าฝนตกหรือไม่ (Is it raining?)
 - หาก "ไม่": ออกไปข้างนอกได้เลย (Go outside)
 - หาก "ใช่": ดำเนินการตรวจสอบต่อไป
3. ตรวจสอบว่ามีร่มหรือไม่ (Have umbrella?)
 - หาก "ใช่": สามารถออกไปข้างนอกได้ (Go outside)
 - หาก "ไม่": ต้องรอ (Wait a while)
4. รอสักครู่ (Wait a while) หลังจากฯ ตรวจสอบว่าฝนยังตกอยู่หรือไม่อีกครั้ง (Is it raining?)
5. ตรวจสอบฝนอีกครั้ง (Is it raining?)
 - หาก "ยังตกอยู่": กลับไปที่ขั้นตอนรอ (Wait a while)
 - หาก "หยุดตกแล้ว": สามารถออกไปข้างนอกได้ (Go outside)
6. สิ้นสุดกระบวนการ (End) เมื่อออกไปข้างนอกแล้ว กระบวนการสิ้นสุด

ตารางที่ 3.1 องค์ประกอบของผังงาน

ภาพองค์ประกอบ	คำอธิบาย
	Terminator (Start/End) รูปวงรี (Oval) ใช้แสดงจุดเริ่มต้น (Start) หรือ จุดสิ้นสุด (End) ของผังงาน ควรระบุ “Start” หรือ “End” ไว้อย่างชัดเจน ภายในวงรี เพื่อป้องกันความสับสน
	Input/Output (I/O) รูปสี่เหลี่ยมคงที่ (Parallelogram) ใช้แทนการรับข้อมูลเข้า (Input) หรือการแสดงผล (Output) ของกระบวนการ เช่น การรับค่าจากคีย์บอร์ด การอ่านข้อมูลจากไฟล์ การพิมพ์ข้อความออกหน้าจอ เป็นต้น การเขียนคำอธิบายสั้น ๆ เช่น “Input X” หรือ “Print Y” เพื่อระบุว่าข้อมูลที่รับหรือแสดงคืออะไร
	Process รูปสี่เหลี่ยมผืนผ้า (Rectangle) ใช้แทนการประมวลผล การคำนวณ หรือการปฏิบัติงานใด ๆ ในกระบวนการ เช่น การคำนวณสูตร การอัปเดตข้อมูลในระบบ การประมวลผลเพื่อสร้างข้อมูลใหม่ เป็นต้น ควรสรุปข้อความให้กระชับ เช่น “Calculate Total” เพื่อให้เข้าใจว่าขั้นตอนการทำงานคืออะไร
	Decision รูปข้าวหลามตัดหรือรูปสี่เหลี่ยมขนมเปียกปูน (Diamond) ใช้แสดงจุดตัดสินใจ ซึ่งมักมีเส้นทางแตกแขนงอย่างน้อย 2 เส้นทาง (เช่น Yes/No, True/False) หลังจากเงื่อนไข (Condition) ถูกตรวจสอบ ควรระบุเงื่อนไขสั้น ๆ ไว้ในสัญลักษณ์ เช่น “X > 0?” โดยมีการทำกับเส้นทางที่ออกจาก Decision เพื่อความชัดเจน
	Arrow/Flow line ลูกศรหรือเส้นพร้อมหัวลูกศร (Arrow) ใช้เชื่อมโยงสัญลักษณ์แต่ละขั้นตอน แสดงทิศทางการไหลของกระบวนการ ควรเขียนให้ต่อเนื่องและชัดเจน โดยไม่ให้เส้นลูกศรไขว้กันมากเกินไป เพราะอาจทำให้ผังงานดูสับสน
	Connector (บางครั้งเรียกเป็น On-page/Off-page Connector) รูปวงกลมเล็ก (On-page) หรือรูปตัว “บ้าน” (Off-page) ใช้เชื่อมขั้นตอนที่อยู่ไกลกัน หรือเชื่อมผังงานข้ามหน้า (ในกรณีที่ผังงานมีหลายหน้า) ควรระบุชื่อหรือเลขกำกับคู่กันเพื่อให้ง่ายต่อการติดตาม
	Subprocess/Function (ในบางกรณี) สี่เหลี่ยมผืนผ้าที่มีเส้นขอบสองชั้น (Double-lined Rectangle) หรือบางครั้งใช้สัญลักษณ์อื่นตามมาตรฐานของแต่ละองค์กรใช้แทนกระบวนการย่อยที่อาจมีรายละเอียดซับซ้อนในตัวเอง แต่สรุปมาใช้ในผังงานหลักแบบย่อ ควรตั้งชื่อ Subprocess ให้เข้าใจง่าย และจัดทำ Flowchart แยกต่างหากสำหรับรายละเอียดใน Subprocess นั้น

ผังงานเปรียบเสมือน "แผนที่นำทาง" ของโปรแกรม ที่ช่วยให้เข้าใจกระบวนการทำงานได้อย่างเป็นระบบและลดความสับสน โดยเฉพาะในการทำงานเป็นทีม ผังงานจะช่วยให้ทุกคนในทีมเห็นภาพรวมตรงกันลดข้อผิดพลาดที่เกิดจากความเข้าใจที่ไม่ตรงกัน การฝึกภาคผังงานตั้งแต่เริ่มต้นเขียนโปรแกรมจะช่วยให้การออกแบบโปรแกรมเป็นไปอย่างมีเหตุผลและง่ายต่อการปรับปรุงในอนาคต

3.2 การเขียนโปรแกรมแบบเงื่อนไข

การเขียนโปรแกรมแบบเงื่อนไข (Conditional Programming) มีบทบาทสำคัญในการควบคุมการทำงานของโปรแกรม (Flow Control) เพื่อให้โปรแกรมสามารถตัดสินใจและเลือกทิศทางการทำงานตามสถานการณ์ต่าง ๆ ได้ เช่น ในโปรแกรมถูกกำหนดโดย Boolean Expression ซึ่งมีค่าผลลัพธ์เป็น True หรือ False ร่วมกับการใช้ตัวดำเนินการเปรียบเทียบ (Comparison Operators) เช่น ==, <, > และตัวดำเนินการทางตรรกية (Logical Operators) ได้แก่ and, or, not เพื่อสร้างเงื่อนไขที่ซับซ้อนยิ่งขึ้น โครงสร้างเงื่อนไข เช่น if-else และ if-elif-else ช่วยให้โปรแกรมเลือกดำเนินการตามแต่ละเงื่อนไขได้อย่างเหมาะสมรวมถึงสามารถจัดการกรณีซ้อนซับด้วย Nested Conditions นอกจากนี้ การเขียนโปรแกรมแบบเงื่อนไขยังช่วยเพิ่มความยืดหยุ่นและประสิทธิภาพให้โปรแกรม ลดการประมวลผลที่ไม่จำเป็น และรองรับการตรวจสอบหรือจัดการข้อผิดพลาดได้อย่างดีเยี่ยม

ประโยชน์ของการเขียนโปรแกรมแบบเงื่อนไข (Conditional Programming)

1. โปรแกรมสามารถเลือกทำงานตามสถานการณ์ที่กำหนด เช่น ตรวจสอบสิทธิ์การเข้าถึงระบบหรือแสดงข้อมูลที่เหมาะสมตามข้อมูลที่ได้รับ
2. เงื่อนไขช่วยให้โปรแกรมตอบสนองต่อความเปลี่ยนแปลงของอินพุตหรือสถานการณ์ต่าง ๆ ได้ เช่น การตรวจสอบค่าที่ผู้ใช้ป้อน
3. ช่วยลดการเขียนโค้ดซ้ำ โดยโปรแกรมจะทำเฉพาะคำสั่งที่จำเป็นเท่านั้น
4. ใช้ตรวจสอบและจัดการกรณีที่ไม่ปกติ เช่น ข้อมูลที่ผิดพลาด หรือการป้องกันการทำงานที่อาจส่งผลกระทบต่อระบบ
5. การใช้เงื่อนไขร่วมกับลูปและฟังก์ชันช่วยให้สามารถออกแบบโปรแกรมที่มีโครงสร้างชัดเจนและจัดการกรณีที่ซับซ้อนได้ง่ายขึ้น

3.2.1 เงื่อนไขพื้นฐาน (Boolean Logic)

ความหมายของ True และ False ในภาษา Python และโปรแกรมมิ่งอื่น ๆ ค่าความจริง (Boolean) เป็นชนิดข้อมูลพื้นฐานที่ใช้ในการตัดสินใจ ค่าความจริงจะมีเพียง 2 ค่าเท่านั้น ได้แก่

- True ใช้แทนค่าที่เป็น "จริง"
- False ใช้แทนค่าที่เป็น "เท็จ"

3.2.2 การดำเนินการเปรียบเทียบ (Comparison Operations)

การดำเนินการตรรกศาสตร์ใน Python ใช้ Boolean Operators เพื่อตรวจสอบเงื่อนไขหรือเปรียบเทียบค่าต่าง ๆ โดยผลลัพธ์ที่ได้จะเป็น True หรือ False เท่านั้น ซึ่ง Boolean Operations มีความสำคัญอย่างยิ่งในการเขียนโปรแกรมเพื่อควบคุมการไหลของคำสั่ง เช่น การใช้ในโครงสร้างเงื่อนไข (if, elif, else) ที่จะกล่าวในหัวข้อต่อไป Python รองรับการดำเนินการตรรกศาสตร์ ดังนี้

1. == (Equal to) ใช้ตรวจสอบว่าสองค่ามีค่าเท่ากันหรือไม่ หากเท่ากันจะให้ผลลัพธ์เป็น True

```
print(42 == 42.0) # True (ตัวเลขเท่ากัน แม้ชนิดข้อมูลต่างกัน)
print(42 == '42') # False (ชนิดข้อมูลต่างกัน)
```

ผลลัพธ์

```
True
False
```

Python สามารถเปรียบเทียบข้อความโดยใช้ Boolean Operators เช่นเดียวกับตัวเลข โดยพิจารณาตัวอักษรเล็ก-ใหญ่เป็นส่วนสำคัญ

```
print('hello' == 'hello') # True (ข้อความเหมือนกัน)
print('hello' == 'Hello') # False (ตัวพิมพ์เล็ก-ใหญ่ต่างกัน)
```

ผลลัพธ์

```
True
False
```

หากต้องการเปรียบเทียบข้อความโดยไม่สนใจตัวพิมพ์เล็ก-ใหญ่ สามารถใช้ .lower() หรือ .upper()

```
print('Hello'.lower() == 'hello') # True (แปลงข้อความเป็นตัวพิมพ์เล็กเหมือนกัน)
```

ผลลัพธ์

```
True
```

การเปรียบเทียบระหว่างตัวเลขและข้อความ (String) จะให้ผลลัพธ์เป็น False เสมอ เนื่องจากชนิดข้อมูลต่างกัน

```
print(42 == '42') # False (ตัวเลขและข้อความไม่เท่ากัน)
print(42 == 42.0) # True (ค่าทางเลขเท่ากัน แม้ชนิดต่างกัน)
```

ผลลัพธ์

```
False
True
```

ข้อควรระวัง การเปรียบเทียบค่าที่มีชนิดข้อมูลต่างกัน เช่น ตัวเลขและข้อความ จะให้ผลลัพธ์เป็น False เสมอ การใช้ฟังก์ชัน เช่น .lower() หรือ .upper() กับข้อความช่วยลดข้อผิดพลาดจากการเปรียบเทียบตัวพิมพ์เล็ก-ใหญ่

2. != (Not equal to) ใช้ตรวจสอบว่าสองค่าไม่เท่ากันหรือไม่ หากไม่เท่ากันจะให้ผลลัพธ์เป็น True

```
print(2 != 3) # True ( เพราะ 2 ไม่เท่ากับ 3)
print(2 != 2) # False ( เพราะ 2 เท่ากับ 2)
```

ผลลัพธ์

```
True
False
```

3. <, <=, >, >= ใช้เปรียบเทียบค่าตัวเลข เช่น น้อยกว่า (<), มากกว่า (>), น้อยกว่าหรือเท่ากับ (<=), และมากกว่าหรือเท่ากับ (>=)

```
a = 42
print(a <= 42) # True ( เพราะ a เท่ากับ 42 )
print(a > 50) # False ( เพราะ a น้อยกว่า 50 )
```

ผลลัพธ์

True

False

ตารางที่ 3.2 สรุปการดำเนินการเปรียบเทียบใน Python

ตัวดำเนินการ (Operator)	ชื่อ (Operation)	ตัวอย่างโค้ด	ผลลัพธ์
==	เท่ากับ	42 == 42.0	True
!=	ไม่เท่ากับ	2 != 3	True
<	น้อยกว่า	5 < 10	True
<=	น้อยกว่าหรือเท่ากับ	5 <= 5	True
>	มากกว่า	10 > 5	True
>=	มากกว่าหรือเท่ากับ	10 >= 10	True

3.2.3 คำสั่งเงื่อนไข if

คำสั่ง if ใช้เพื่อ ตรวจสอบ เงื่อนไขที่กำหนด หากเงื่อนไขเป็นจริง (True) โปรแกรมจะดำเนินการ คำสั่งในบล็อกของ if ต่อไป หากเงื่อนไขเป็นเท็จ (False) โปรแกรมจะข้ามการทำงานของบล็อก if และ ดำเนินการคำสั่งถัดไป ตัวอย่างเช่น การตรวจสอบคำตอบของผู้เล่นในเกม หากคำตอบตรงกับคำตอบที่ถูกต้อง จะเพิ่มคะแนนและพิมพ์ข้อความยืนยัน แต่หากไม่ตรง จะข้ามคำสั่งใน if และแสดงข้อความอื่นแทน

```
if timsAnswer == correctAnswer:
    print("You got it right!")
    score = score + 1

print("Thanks for playing.")
```

เงื่อนไข if timsAnswer == correctAnswer: ใช้เครื่องหมาย == เพื่อตรวจสอบว่าคำตอบของ timsAnswer ตรงกับ correctAnswer หรือไม่ หากเงื่อนไขเป็น True (คำตอบถูกต้อง) พิมพ์ข้อความ "You got it right!" เพิ่มค่าคะแนน (score) ขึ้น 1 หากเงื่อนไขเป็น False (คำตอบไม่ถูกต้อง) บล็อกคำสั่งใน if จะไม่ถูกดำเนินการ ข้อความ "Thanks for playing."

ผลลัพธ์ กรณีคำตอบถูกต้อง ถ้า timsAnswer มีค่าเท่ากับ correctAnswer

You got it right!

Thanks for playing.

ผลลัพธ์ กรณีคำตอบไม่ถูกต้อง ถ้า timsAnswer มีค่าไม่เท่ากับ correctAnswer

Thanks for playing.

- การใช้ Colon (:) หลังเงื่อนไข if ต้องใช้ : เพื่อระบุว่าเงื่อนไขนี้มีบล็อกคำสั่งที่เกี่ยวข้อง
- Indentation (การเยื่อง) คำสั่งที่อยู่ภายใต้บล็อก if ต้องเยื่อง (Indent) เพื่อแสดงว่าเป็นส่วนหนึ่งของคำสั่ง if
- การเปรียบเทียบด้วย == ใช้ตรวจสอบความเท่าเทียมระหว่างสองค่า เช่น คำตอบของผู้เล่นและคำตอบที่ถูกต้อง

3.2.4 คำสั่งเงื่อนไข if – else

คำสั่ง if – else ใช้สำหรับตรวจสอบเงื่อนไขที่กำหนด หากเงื่อนไขเป็นจริง (True) โปรแกรมจะดำเนินการคำสั่งในบล็อกของ if แต่หากเงื่อนไขเป็นเท็จ (False) โปรแกรมจะดำเนินการคำสั่งในบล็อกของ else แทน การใช้ if – else เหมาะสำหรับสถานการณ์ที่มีการเลือกทางเดินอย่างใดอย่างหนึ่ง (Either-Or) เช่น การตรวจสอบว่าคะแนนของนักเรียนผ่านเกณฑ์หรือไม่

ตรวจสอบเกณฑ์ผ่านคะแนน

```
score = 75
passing_score = 60

if score >= passing_score:
    print("Congratulations! You passed.")
else:
    print("Sorry, you did not pass.")
```

เงื่อนไข if score >= passing_score: ใช้ตรวจสอบว่าคะแนน score มากกว่าหรือเท่ากับคะแนนผ่าน passing_score หรือไม่บล็อกคำสั่งใน if: หากเงื่อนไขเป็นจริง (True) โปรแกรมจะแสดงข้อความ Congratulations! You passed. บล็อกคำสั่งใน else: หากเงื่อนไขเป็นเท็จ (False) โปรแกรมจะแสดงข้อความ Sorry, you did not pass.

ผลลัพธ์ หาก score = 75 และ passing_score = 60

Congratulations! You passed.

ผลลัพธ์ หาก score = 50 และ passing_score = 60

Sorry, you did not pass.

จุดสำคัญในการใช้ if – else

- Colon (:): ต้องใช้ : หลังคำสั่ง if และ else เพื่อระบุว่ามีบล็อกคำสั่งที่เกี่ยวข้อง
- Indentation (การเยื่อง): คำสั่งในบล็อกของ if และ else ต้องเยื่อง (Indent) เพื่อบ่งบอกว่าเป็นส่วนหนึ่งของเงื่อนไข

- การกำหนดทางเลือกสองทาง: if – else ใช้สำหรับสถานการณ์ที่ต้องเลือกว่าจะทำงานแบบใดแบบหนึ่ง ไม่สามารถเลือกทั้งสองได้

ตัวอย่างการประยุกต์ใช้งานในสถานการณ์จริง ในกระบวนการตัดสินใจเกี่ยวกับสิทธิของบุคคล เช่น การตรวจสอบว่าใครสามารถสมัครเลือกตั้งได้ การกำหนดเงื่อนไขที่ชัดเจน เช่น อายุขั้นต่ำ มีความสำคัญอย่างยิ่ง โดยการเขียนโปรแกรม Python ด้วยคำสั่ง if – else ช่วยให้สามารถสร้างเงื่อนไขและแยกการทำงานของโปรแกรมตามสถานะของอายุได้อย่างง่ายดายและแม่นยำ

```
age = 20

if age >= 18:
    print("คุณสามารถสมัครเลือกตั้งได้")
else:
    print("คุณยังไม่สามารถสมัครเลือกตั้งได้")

ผลลัพธ์กรณีอายุ 20
คุณสามารถสมัครเลือกตั้งได้

ผลลัพธ์กรณีอายุ 15
คุณยังไม่สามารถสมัครเลือกตั้งได้
```

3.2.5 การดำเนินการทางตรรกศาสตร์ (Comparison Operations)

ตรรกศาสตร์ Boolean ใช้ในการตรวจสอบค่าความจริง (True หรือ False) ผ่านตัวดำเนินการ (Logical Operators) เช่น and, or, not โดยสามารถอธิบายตัวดำเนินการดังนี้

ตารางที่ 3.3 ตารางความจริงการดำเนินการทางตรรกศาสตร์

P	Q	P and Q	P or Q	$P \rightarrow Q$	$P \leftrightarrow Q$	not P
True	True	True	True	True	True	False
True	False	False	True	True	False	
False	True	False	True	False	True	True
False	False	False	False	False	False	

1. and (และ) เนื่องไปจะเป็น True ก็ต่อเมื่อ ทั้งสองเงื่อนไข เป็น True

```
p = True
q = False

print(p and q)      # False
print(True and True) # True
```

ผลลัพธ์

```
False
True
```

ตัวอย่างใช้งานในสถานการณ์จริง

```
age = 25
is_employed = True

if age > 18 and is_employed:
    print("คุณมีสิทธิ์เงิน")
else:
    print("คุณไม่มีสิทธิ์เงิน")
```

ผลลัพธ์

คุณมีสิทธิ์เงิน

2. or (หรือ) เงื่อนไขจะเป็น True ก็ต่อเมื่อ มีเงื่อนไขใดเงื่อนไขหนึ่ง เป็น True หากทุกเงื่อนไขเป็น False จะได้ผลลัพธ์เป็น False

```
p = True
q = False

print(p or q)          # True
print(False or False) # False
```

ผลลัพธ์

```
True
False
```

ตัวอย่างใช้งานในสถานการณ์จริง

```
has_passport = False
has_id_card = True

if has_passport or has_id_card:
    print("อนุญาตให้เดินทาง")
else:
    print("ไม่อนุญาตให้เดินทาง")
```

ผลลัพธ์

อนุญาตให้เดินทาง

3. not (ไม่) ตัวดำเนินการ not ใช้สำหรับกลับค่าความจริง เช่น จาก True เป็น False และจาก False เป็น True

```
print(not p)          # False
print(not False)
```

ผลลัพธ์

```
False
True
```

ตัวอย่างใช้งานในสถานการณ์จริง

```
is_sunny = False

if not is_sunny:
    print("อย่าลืมพกร่ม!")
else:
```

ผลลัพธ์

อย่าลืมพก_rém!

4. $P \rightarrow Q$ (Implication) เงื่อนไขจะเป็น False เมื่อ P เป็น True และ Q เป็น False ใช้สำหรับการสร้างความสัมพันธ์ที่ระบุว่าหาก P เป็นจริง Q ต้องเป็นจริง

```
p = True
q = False
implication = not p or q # จำลอง P → Q
print("P → Q:", implication)
```

ผลลัพธ์

$P \rightarrow Q: \text{False}$

ตัวอย่างใช้งานในสถานการณ์จริง

```
is_traffic_light_green = False
is_traffic_light_yellow = True

if is_traffic_light_green or (is_traffic_light_yellow and not
is_traffic_light_green):
    print("ยานพาหนะสามารถเคลื่อนที่ได้")
else:
    print("ยานพาหนะต้องหยุด")
```

ผลลัพธ์

ยานพาหนะสามารถเคลื่อนที่ได้

5. $P \leftrightarrow Q$ (Equivalence) เงื่อนไขจะเป็น True ก็ต่อเมื่อ P และ Q มีค่าเท่ากัน (True ทั้งคู่ หรือ False ทั้งคู่)

```
equivalence = (p and q) or (not p and not q)
print("P ↔ Q:", equivalence)
```

ผลลัพธ์

$P \leftrightarrow Q: \text{False}$

ตัวอย่างใช้งานในสถานการณ์จริง

```
# ตัวอย่าง Equivalence ในระบบเข้าสู่ระบบ
username_correct = True      # ชื่อผู้ใช้ถูกต้อง
password_correct = True      # รหัสผ่านถูกต้อง

# ตรวจสอบว่าชื่อผู้ใช้และรหัสผ่านมีสถานะตรงกัน
equivalence = ( (username_correct and password_correct) or
(not username_correct and not password_correct) )
```

```

if equivalence:
    print("ทั้งชื่อผู้ใช้และรหัสผ่านอยู่ในสถานะเดียวกัน")
else:
    print("ชื่อผู้ใช้และรหัสผ่านอยู่ในสถานะไม่ตรงกัน")

```

ผลลัพธ์

ทั้งชื่อผู้ใช้และรหัสผ่านอยู่ในสถานะเดียวกัน

คำสั่ง if – else ช่วยเพิ่มความยืดหยุ่นและความแม่นยำในการเขียนโปรแกรมเมื่อต้องตัดสินใจในสถานการณ์ที่มีตัวเลือกสองทาง

3.2.6 การเขียนผังงานสำหรับเงื่อนไข

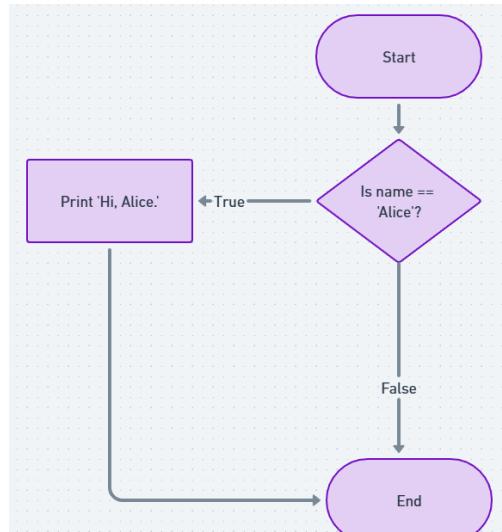
- ตัวอย่างการร่าง Flowchart สำหรับโค้ด เงื่อนไขแบบ if

```

if name == "Alice":
    print("Hi, Alice.")

```

ตัวอย่างภาพ Flowchart นี้แสดงกระบวนการตรวจสอบว่าชื่อตั้งกับ "Alice" หรือไม่ โดยเริ่มต้นจาก การรับค่า name หากเงื่อนไขเป็นจริงจะพิมพ์ข้อความ "Hi, Alice." และจบการทำงาน หากเงื่อนไขเป็นเท็จจะ ข้ามการพิมพ์และจบโปรแกรมทันที.



รูปที่ 3.4 Flowchart ของโปรแกรมแบบเงื่อนไข if

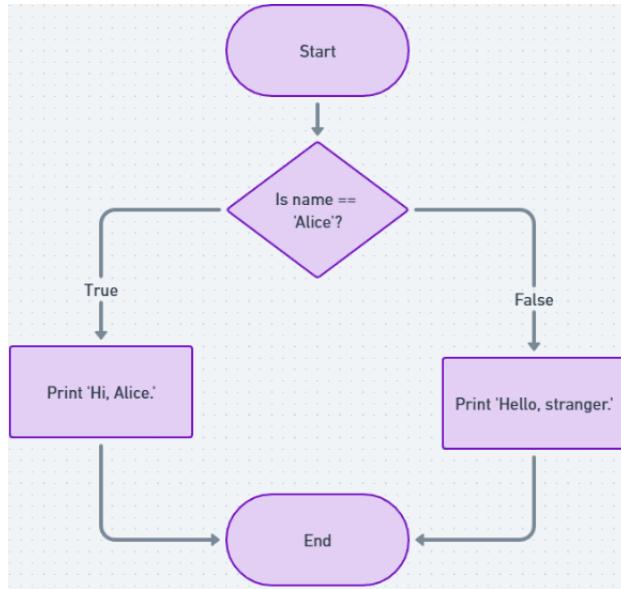
- ตัวอย่างการร่าง Flowchart สำหรับโค้ด เงื่อนไขแบบ if-else

```

if name == "Alice":
    print("Hi, Alice.")
else:
    print("Hello, stranger.")

```

ตัวอย่างภาพ Flowchart นี้แสดงกระบวนการตรวจสอบว่าชื่อตั้งกับ "Alice" หรือไม่ โดยเริ่มต้นจาก การรับค่า name หากเงื่อนไขเป็นจริงจะพิมพ์ข้อความ "Hi, Alice." และจบการทำงาน หากเงื่อนไขเป็นเท็จจะ ข้ามการพิมพ์และจบโปรแกรมทันที.



รูปที่ 3.5 Flowchart ของโปรแกรมแบบเงื่อนไข if-else

3.2.7 คำสั่งเงื่อนไข if – elif – else

บางครั้งการเขียนโปรแกรมจำเป็นต้องใช้ หลาย if และ การซ้อน if (Nested if) เพื่อจัดการกับเงื่อนไขที่ซับซ้อนหรือกรณีที่ต้องตรวจสอบเงื่อนไขอย่างภายในเงื่อนไขหลัก เช่น การตรวจสอบสถานะหลายระดับ หรือการทำงานที่ต้องพิจารณาข้อมูลหลายชุด การซ้อน if ช่วยให้สามารถแยกตระกูลของโปรแกรมออกเป็นชั้นตอนย่อยที่ชัดเจนและเป็นระบบ แต่ควรใช้อย่างระมัดระวังเพื่อไม่ให้โค้ดซับซ้อนเกินไปและยากต่อการอ่านตัวอย่างเช่นปัญหาการตัดเกรด

โปรแกรมตัดเกรดเป็นโปรแกรมที่ใช้ในการประเมินผลคะแนนของนักเรียนหรือนักศึกษา โดยเปรียบเทียบคะแนนที่ได้กับเกณฑ์ที่กำหนดไว้ เพื่อกำหนดรั้ดับเกรด (Grade) ที่เหมาะสม เช่น A, B, C, D หรือ F โปรแกรมนี้มักใช้ในระบบการจัดการการศึกษา หรือการพัฒนาซอฟต์แวร์เพื่อช่วยในกระบวนการประเมินผลตัวอย่างเกณฑ์การตัดเกรด

คะแนน >= 80: Grade A

คะแนน >= 70: Grade B

คะแนน >= 60: Grade C

คะแนน >= 50: Grade D

คะแนน < 50: Grade F

การเขียนโค้ดแบบเงื่อนไขแบบหลายเงื่อนไข

```

# ตรวจสอบเกณฑ์การตัดเกรด
if score >= 80:
    print("Grade A")
if score >= 70 and score < 80:
    print("Grade B")
if score >= 60 and score < 70:
    print("Grade C")
  
```

```

if score >= 50 and score < 60:
    print("Grade D")
if score < 50:
    print("Grade F")

```

การเขียนโค้ดแบบเงื่อนไขซ้อน (Nested-if)

```

if score >= 80:
    print("Grade A")
else:
    if score >= 70:
        print("Grade B")
    else:
        if score >= 60:
            print("Grade C")
        else:
            if score >= 50:
                print("Grade D")
            else:
                print("Grade F")

```

การเขียนโค้ดแบบเงื่อนไขหลาย if มีข้อดีคือโครงสร้างดูเรียบง่ายและเหมาะสมกับเงื่อนไขที่แยกกันอิสระ แต่เมื่อข้อเสียคือทุกเงื่อนไขจะถูกตรวจสอบซ้ำซ้อน ทำให้โปรแกรมทำงานไม่ประยุกต์เวลา ควรปรับปรุงโดยใช้ elif เพื่อเพิ่มประสิทธิภาพและลดความซ้ำซ้อน สำหรับการใช้ Nested if ช่วยลดการตรวจสอบเงื่อนไขซ้ำซ้อน แต่โครงสร้างซ้อนกันมากเกินไปอาจทำให้โค้ดดูซับซ้อนและยากต่อการอ่าน ควรปรับเป็น if – elif – else เพื่อทำให้โค้ดกระชับและอ่านง่ายขึ้น

```

if score >= 80:
    print("Grade A")
elif score >= 70:
    print("Grade B")
elif score >= 60:
    print("Grade C")
elif score >= 50:
    print("Grade D")
else:
    print("Grade F")

```

เงื่อนไขแรก (if score >= 80) โปรแกรมจะตรวจสอบว่า score มากกว่าหรือเท่ากับ 80 หรือไม่

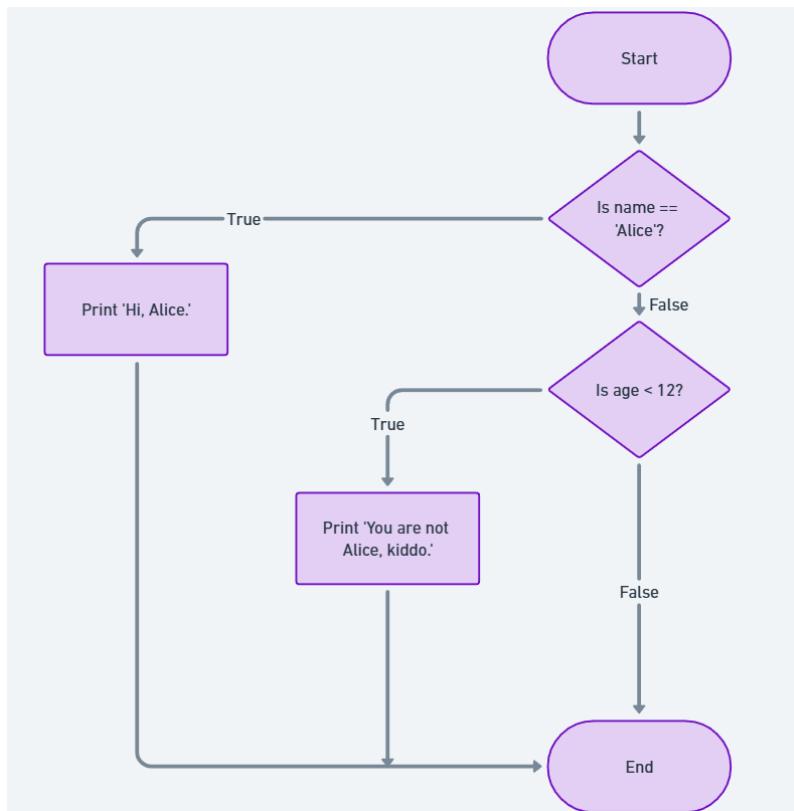
- หากเป็นจริง (True) แสดงข้อความ Grade A และสิ้นสุดการตรวจสอบเงื่อนไขอื่น ๆ เงื่อนไขเพิ่มเติม (elif)

- หากเงื่อนไขแรกไม่เป็นจริง (False) โปรแกรมจะตรวจสอบเงื่อนไขใน elif ตามลำดับ
 - elif score >= 70: หากเป็นจริง แสดงข้อความ Grade B
 - elif score >= 60: หากเป็นจริง แสดงข้อความ Grade C
 - elif score >= 50: หากเป็นจริง แสดงข้อความ Grade D

โปรแกรมจะหยุดตรวจสอบเมื่อพบเงื่อนไขแรกที่เป็นจริง

- กรณีที่ไม่มีเงื่อนไขใดเป็นจริง (else)
- ตัวอย่างเพิ่มเติมการใช้เงื่อนไขแบบ if-elif-else

1. ถ้าต้องการโปรแกรม ตรวจสอบเงื่อนไขโดยเริ่มจากการเช็คว่าชื่อ (name) เป็น "Alice" หรือไม่ และพิมพ์ "Hi, Alice." หากเป็นจริง หากชื่อไม่ใช่ "Alice" จะตรวจสอบว่าอายุ (age) น้อยกว่า 12 หรือไม่และพิมพ์ "You are not Alice, kiddo." หากเป็นจริง หากไม่มีเงื่อนไขใดเป็นจริง จะพิมพ์ข้อความเริ่มต้น "Hello, stranger."



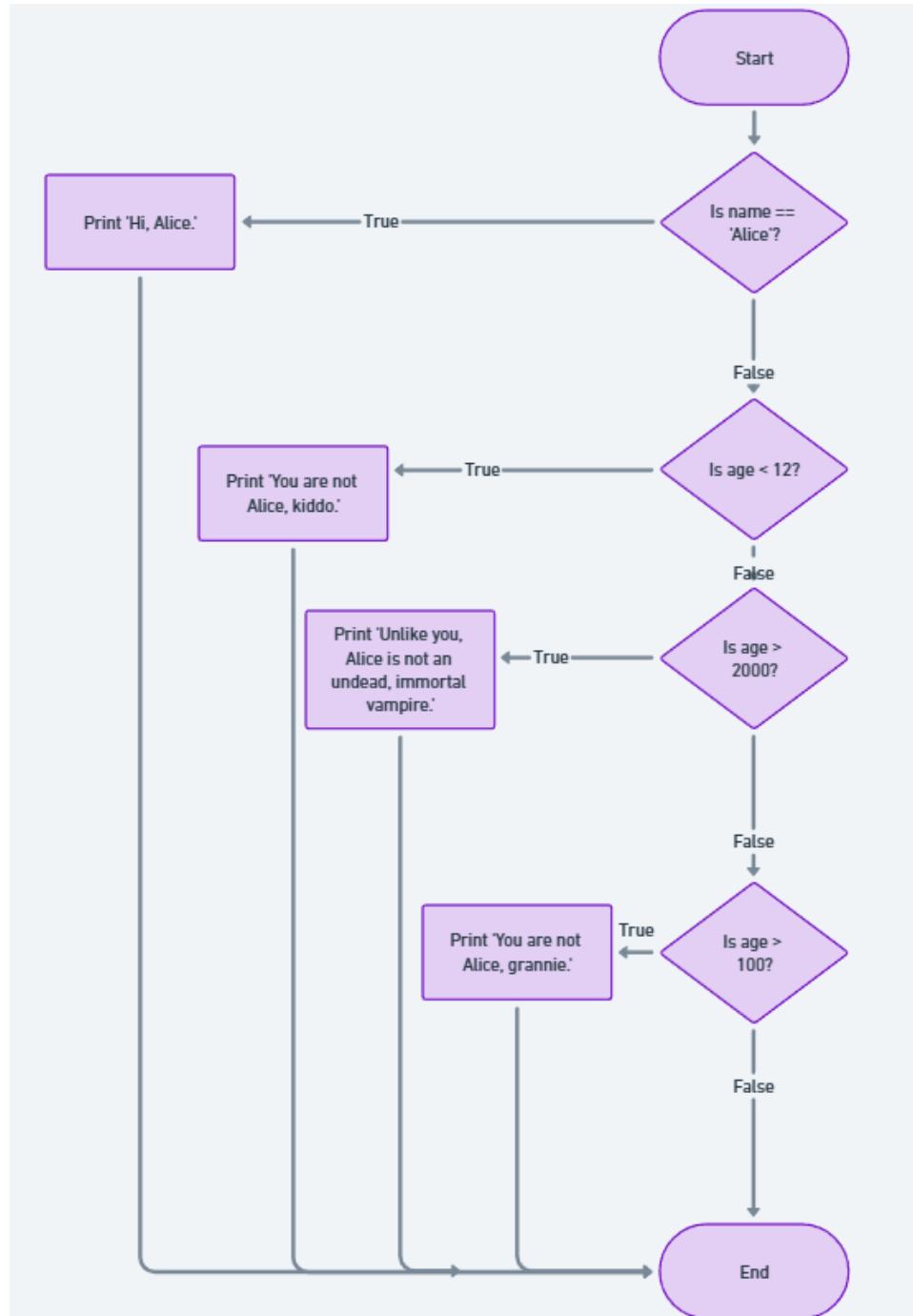
รูปที่ 3.6 Flowchart ของโปรแกรมแบบเงื่อนไข if-elif-else

```

if name == "Alice":
    print("Hi, Alice.")
elif age < 12:
    print("You are not Alice, kiddo.")
else:
    print("Hello, stranger.")
  
```

โค้ดนี้ตรวจสอบเงื่อนไขโดยเริ่มจากการเช็คชื่อ (name) เป็น "Alice" หรือไม่ และพิมพ์ "Hi, Alice." หากเป็นจริง หากชื่อไม่ใช่ "Alice" จะตรวจสอบว่าอายุ (age) น้อยกว่า 12 หรือไม่และพิมพ์ "You are not Alice, kiddo." หากเป็นจริง หากไม่มีเงื่อนไขใดเป็นจริง จะพิมพ์ข้อความเริ่มต้น "Hello, stranger."

2. สร้างโปรแกรมเพิ่มเติมจากข้อที่แล้ว โดยให้ตรวจสอบชื่อและอายุของผู้ใช้งาน โดยเริ่มจากการเช็คว่าชื่อ (name) เป็น "Alice" หรือไม่ หากใช่ ให้พิมพ์ "Hi, Alice." หากไม่ใช่ ให้ตรวจสอบอายุตามเงื่อนไขต่อไปนี้: หากอายุน้อยกว่า 12 ให้พิมพ์ "You are not Alice, kiddo." หากอายุมากกว่า 2000 ให้พิมพ์ "Unlike you, Alice is not an undead, immortal vampire." หากอายุมากกว่า 100 ให้พิมพ์ "You are not Alice, grannie." และหากไม่เข้าเงื่อนไขใด ๆ ให้พิมพ์ "Hello, stranger."



รูปที่ 3.7 Flowchart ของโปรแกรมแบบเงื่อนไข if-elif-else

```

if name == "Alice":
    print("Hi, Alice.")
elif age < 12:
  
```

```

    print("You are not Alice, kiddo.")
elif age > 2000:
    print("Unlike you, Alice is not an undead, immortal vampire.")
elif age > 100:
    print("You are not Alice, grannie.")
else:
    print("Hello, stranger.")

```

โค้ดนี้ตรวจสอบเงื่อนไขโดยเริ่มจากการเช็คชื่อ (name) เป็น "Alice" หรือไม่ และพิมพ์ "Hi, Alice." หากเป็นจริง หากชื่อไม่ใช่ "Alice" จะตรวจสอบว่าอายุ (age) น้อยกว่า 12 หรือไม่และพิมพ์ "You are not Alice, kiddo." หากเป็นจริง หากอายุมากกว่า 2000 จะพิมพ์ "Unlike you, Alice is not an undead, immortal vampire." และหากอายุมากกว่า 100 จะพิมพ์ "You are not Alice, grannie." หากไม่มีเงื่อนไขใดเป็นจริง จะพิมพ์ข้อความเริ่มต้น "Hello, stranger." โดย Flowchart แสดงในรูปที่ 3.7

โครงสร้าง if-else ไม่ใช่แค่เครื่องมือพื้นฐาน แต่เป็นเหมือน "กระดูกสันหลัง" ของการควบคุมโปรแกรม การที่ Python ทำให้การเขียนเงื่อนไขเป็นเรื่องง่ายและซัดเจนนั้นมีประโยชน์อย่างมาก เงื่อนไขเหล่านี้ทำให้โปรแกรมสามารถตอบสนองต่อข้อมูลหรือเหตุการณ์ที่เปลี่ยนแปลงไปได้ เช่น การตรวจสอบข้อมูลที่ผู้ใช้ป้อนเข้ามา หรือการสร้างเกมที่มีหลายเส้นทางให้เลือก ยิ่งเงื่อนไขมีความซับซ้อนมากเท่าไหร่ ความสามารถในการเขียนโค้ดให้กระชับและเข้าใจง่ายก็ยิ่งสำคัญมากขึ้นเท่านั้น

3.3 การเขียนโปรแกรมแบบวนซ้ำ (Loops)

การใช้ลูป (Loops) เป็นหนึ่งในโครงสร้างการควบคุมที่สำคัญที่สุดในโปรแกรมมิ่ง เนื่องจากช่วยลดความซับซ้อนและเพิ่มประสิทธิภาพในการเขียนโค้ดในสถานการณ์ที่ต้องทำงานซ้ำ ๆ การเขียนคำสั่งที่มีรูปแบบเดิมหลายครั้งด้วยลูปช่วยให้ ดังนี้

1. ลดโค้ดซ้ำซ้อน (Reduce Code Duplication) เขียนคำสั่งเพียงครั้งเดียวแต่สามารถนำไปใช้ซ้ำได้หลายรอบ เช่น การพิมพ์ข้อความ 10 ครั้ง

2. เพิ่มความยืดหยุ่น (Increase Flexibility) หากจำนวนรอบการทำงานทำซ้ำเปลี่ยนแปลง สามารถแก้ไขได้ง่ายเพียงปรับค่าตัวแปรหรือเงื่อนไขในลูป

3. ลดข้อผิดพลาด (Minimize Errors) การเขียนโค้ดแบบซ้ำ ๆ ด้วยมือลดโอกาสการพิมพ์ผิด แต่การใช้ลูปทำให้มั่นใจว่าโค้ดทำงานได้ตามที่ตั้งใจ

4. ช่วยในการควบคุมการประมวลผลข้อมูลขนาดใหญ่ ลูปเหมาะสมสำหรับการประมวลผลข้อมูลจำนวนมาก เช่น การวิเคราะห์ข้อมูลในไฟล์หรือฐานข้อมูล

การวนทำซ้ำสามารถแบ่งตามคำสั่งที่ใช้เขียน ได้แก่

1. while loop ใช้เมื่อไม่ทราบจำนวนรอบที่ต้องการวนซ้ำล่วงหน้า ลูปจะทำงานต่อไปตราบใดที่เงื่อนไขที่กำหนดยังเป็น True นิยมใช้ในกรณีที่ต้องการรอเหตุการณ์หรือค่าบางอย่างเกิดขึ้น เช่น การตรวจสอบข้อมูลจากผู้ใช้

2. for loop ใช้เมื่อทราบจำนวนรอบการวนซ้ำล่วงหน้า หรือเมื่อ wan ช้าผ่านรายการข้อมูล เช่น list tuple หรือช่วงตัวเลข (range) มีประสิทธิภาพและอ่านง่ายเมื่อต้องการประมวลผลข้อมูลที่มีโครงสร้างแบบที่สามารถวนซ้ำได้ (iterable)

3.3.1 While Loop

while loop ถูกออกแบบมาเพื่อใช้ในสถานการณ์ที่ต้องการทำซ้ำคำสั่งใด ๆ โดยไม่ทราบจำนวนรอบที่แน่นอนล่วงหน้า การวนซ้ำจะดำเนินต่อไปจนกว่าเงื่อนไขที่กำหนดจะเป็น False เหมาะสำหรับกระบวนการที่ต้องการตรวจสอบเงื่อนไขอย่างต่อเนื่อง เช่น การรับข้อมูลจากผู้ใช้จนกว่าจะป้อนคำตอบที่ถูกต้อง หรือการตรวจสอบสถานะของระบบ

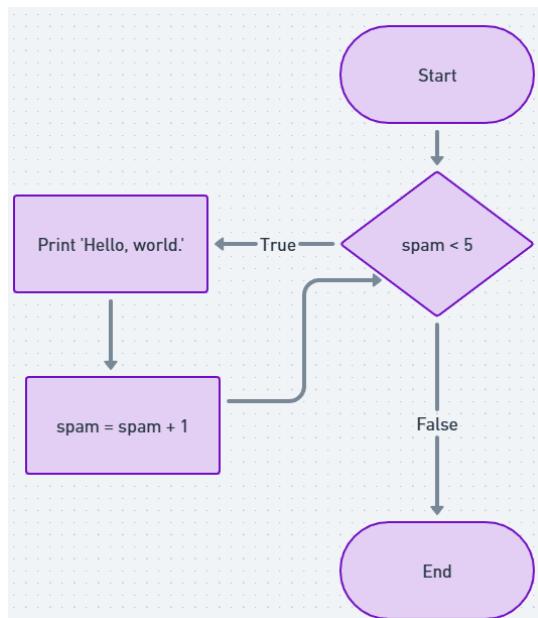
หลักการทำงาน

- ตรวจสอบเงื่อนไขก่อน (condition)
 - ถ้าเงื่อนไขเป็น True -> โปรแกรมจะทำคำสั่งในบล็อกของ while
 - ถ้าเงื่อนไขเป็น False -> โปรแกรมจะหยุดการทำงานของลูปทันที
- คำสั่งในบล็อกลูปจะถูกทำงานซ้ำๆ ทุกครั้งที่เงื่อนไขยังคงเป็น True
- จำเป็นต้องมีการเปลี่ยนแปลงค่าที่เกี่ยวข้องกับเงื่อนไขในลูป มิฉะนั้นจะเกิด ลูปที่ไม่สิ้นสุด (infinite loop)

โครงสร้างพื้นฐาน

```
while condition:
    # คำสั่งที่ต้องการให้ทำซ้ำ
```

เงื่อนไข (Condition) เป็นการตรวจสอบเงื่อนไขที่ต้องเป็น True เพื่อให้คำสั่งภายในลูปทำงาน คำสั่งในบล็อกลูปจะถูกเย็บ (indentation) เพื่อแยกส่วนโค้ดในลูป



รูปที่ 3.8 Flowchart ของโปรแกรมวนทำซ้ำ

รูปที่ 3.8 แสดง Flowchart การวนทำซ้ำ เริ่มต้นโปรแกรมด้วยการกำหนดค่าเริ่มต้นให้ spam = 0 และตรวจสอบเงื่อนไข spam < 5 หากเป็นจริง พิมพ์ข้อความ "Hello, world." และเพิ่มค่า spam ทีละ 1 จากนั้นวนกลับมาตรวจสอบเงื่อนไขใหม่ หากเงื่อนไขเป็นเท็จ (ค่า spam มากกว่าหรือเท่ากับ 5) โปรแกรมจะสิ้นสุดการทำงาน สามารถเขียนเป็นโค้ดแบบ while loop ได้ดังนี้

```
spam = 0

while spam < 5:
    print("Hello, world.")
    spam = spam + 1
```

ผลลัพธ์

```
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
```

เริ่มต้นด้วยตัวแปร spam ที่มีค่าเริ่มต้นเป็น 0 จากนั้น ลูป while จะทำงานตราบใดที่ค่าของ spam น้อยกว่า 5 ในแต่ละครอบลูป พิมพ์ข้อความ "Hello, world." และ เพิ่มค่าของ spam ขึ้นทีละ 1

เมื่อ spam มีค่าเท่ากับ 5 หรือมากกว่า ลูปจะหยุดทำงาน

ข้อแตกต่างระหว่าง while และ if

```
spam = 0

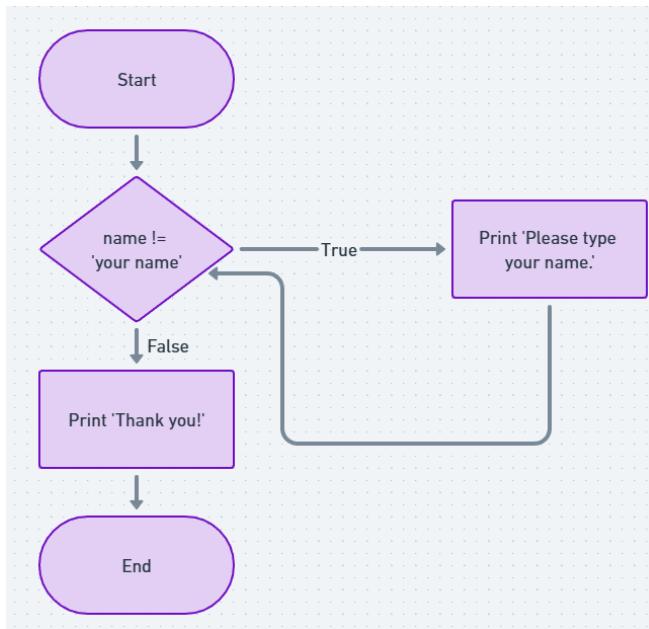
if spam < 5:
    print("Hello, world.")
    spam = spam + 1
```

ผลลัพธ์

```
Hello, world.
```

if statement ตรวจสอบเงื่อนไขเพียงครั้งเดียว หากเงื่อนไขเป็นจริง จะทำการสั่งในบล็อก if เพียงครั้งเดียว และดำเนินการต่อไปยังคำสั่งถัดไปในโปรแกรม หากเงื่อนไขเป็นเท็จ จะข้ามบล็อกคำสั่ง if และดำเนินการต่อ

ตัวอย่างโปรแกรมตรวจสอบชื่อ จากรูปที่ 3.9 แสดง Flowchart นี้แสดงกระบวนการตรวจสอบการป้อนชื่อจากผู้ใช้ โดยโปรแกรมจะวนลูปเพื่อแจ้งให้ป้อนชื่อจนกว่าผู้ใช้จะป้อน "your name" จึงจะแสดงข้อความ "Thank you!" และจบการทำงาน.



รูปที่ 3.9 Flowchart ของโปรแกรมวนทำซ้ำเพื่อตรวจสอบชื่อ

```

while name != "your name":
    print("Please type your name.")
    name = input()
  
```

ผลลัพธ์

```

Please type your name.
Alice
Please type your name.
John
Please type your name.
your name
Thank you!
  
```

แสดงกระบวนการที่โปรแกรมวนลูปเพื่อให้ผู้ใช้ป้อนชื่อจนกว่าจะพิมพ์ "your name" โดยในแต่ละรอบ โปรแกรมพิมพ์ข้อความ "Please type your name." และรับค่าจากผู้ใช้ เมื่อตรงตามเงื่อนไข (name == "your name"), โปรแกรมแสดงข้อความ "Thank you!" และสิ้นสุดการทำงาน.

การใช้คำสั่ง break

break ใช้ควบคุมการทำงานของลูปให้หยุดทันทีเมื่อถึงเงื่อนไขที่กำหนด โดยไม่ต้องรอให้ลูปทำงานจนจบ ช่วยเพิ่มความยืดหยุ่นและลดการเขียนโค้ดซ้ำซ้อน โดยเฉพาะในลูปที่ไม่มีเงื่อนไขสิ้นสุด (while True) หรือสถานการณ์ซับซ้อน นอก จาก นี้ ยังช่วยเพิ่มประสิทธิภาพและความซัดเจนของโค้ด ทำให้ผู้อ่านเข้าใจการทำงานได้ง่ายขึ้น

```

while True:
    print("Please type your name.")
    name = input()
    if name == "your name":
        break

print("Thank you!")

```

โปรแกรมใช้ลูป while True เพื่อให้ทำงานอย่างต่อเนื่องและพิมพ์ข้อความ "Please type your name." พร้อมรับค่าป้อนจากผู้ใช้และบันทึกในตัวแปร name ในแต่ละรอบ โปรแกรมตรวจสอบว่าผู้ใช้ป้อน "your name" หรือไม่ หากใช่ จะใช้คำสั่ง break เพื่อหยุดลูปและสิ้นสุดการทำงาน เมื่อออกจากลูปได้สำเร็จ โปรแกรมจะแสดงข้อความ "Thank you!"

การใช้คำสั่ง continue

continue ใช้ควบคุมการทำงานของลูปให้ข้ามการทำงานในรอบปัจจุบันและกลับไปเริ่มต้นรอบใหม่ ทันที โดยไม่ต้องรันคำสั่งที่เหลือในลูป ช่วยเพิ่มความยืดหยุ่นในการจัดการเงื่อนไขที่ไม่ต้องการประมวลผลในรอบนั้น ๆ โดยเฉพาะในลูปที่มีหลายเงื่อนไขหรือขั้นตอนการทำงานที่ซับซ้อน continue ช่วยลดความซับซ้อน และทำให้ได้ดีอ่านง่ายขึ้น ตัวอย่างโปรแกรมใช้สำหรับจำกัดการเข้าถึง โดยตรวจสอบว่าผู้ใช้คือ "Joe" และใส่รหัสผ่าน "swordfish" อย่างถูกต้อง จึงจะได้รับสิทธิ์เข้าถึง

```

while True:
    print('Who are you?')
    name = input()
    if name != 'Joe':
        continue
    print('Hello, Joe. What is the password? (It is a fish.)')
    password = input()
    if password == 'swordfish':
        break

print('Access granted.')

```

โปรแกรมนี้ใช้ลูป while True เพื่อทำงานอย่างต่อเนื่อง โดยในแต่ละรอบ จะถามชื่อผู้ใช้ (name) และตรวจสอบว่าชื่อนั้นเป็น "Joe" หรือไม่ หากชื่อไม่ใช่ "Joe", โปรแกรมจะใช้คำสั่ง continue เพื่อข้ามคำสั่งที่เหลือในลูปและกลับไปเริ่มต้นลูปรอบใหม่ทันที หากชื่อเป็น "Joe", โปรแกรมจะแสดงข้อความ "Hello, Joe. What is the password? (It is a fish.)" และขอรหัสผ่าน (password) หากรหัสผ่านเป็น "swordfish", โปรแกรมจะใช้คำสั่ง break เพื่อหยุดลูปและแสดงข้อความ "Access granted."

break ใช้สำหรับหยุดการทำงานของลูปทั้งหมดทันทีและออกจากลูป ไม่ว่าลูปจะเป็น for หรือ while continue ใช้สำหรับข้ามคำสั่งที่เหลือในรอบปัจจุบันของลูป และกลับไปเริ่มต้นรอบใหม่ทันที break จะลูปทันทีเมื่อเงื่อนไขเป็นจริง ขณะที่ continue ให้ลูปดำเนินต่อไปตามเงื่อนไขที่กำหนด

3.3.2 For Loop

for loop ถูกออกแบบมาเพื่อใช้ในสถานการณ์ที่ทราบจำนวนรอบการทำซ้ำล่วงหน้า หรือเมื่อต้องการวนซ้ำตามลำดับของค่าที่กำหนดไว้ เช่น การวนซ้ำในรายการ (list) ช่วงตัวเลข (range) หรือข้อความ (string) for loop มีความสะดวกในการจัดการข้อมูลที่มีโครงสร้าง เช่น การประมวลผลข้อมูลในรายการ หรือการนับจำนวนรอบที่แน่นอน

หลักการทำงาน

- ทำงานโดยการวนซ้ำในแต่ละค่าของ iterable เช่น list tuple range หรือ string
 - โปรแกรมจะนำค่าจาก iterable แต่ละค่ามาใส่ในตัวแปรที่กำหนด และดำเนินการคำสั่งในบล็อกๆ
 - เมื่อวนซ้ำครบทุกค่าใน iterable โปรแกรมจะหยุดการทำงานของลูปโดยอัตโนมัติ
- โครงสร้างพื้นฐาน**

```
for variable in iterable:  
    # คำสั่งที่ต้องการให้ทำซ้ำ
```

ตัวแปร (variable) ใช้เก็บค่าจาก iterable ในแต่ละรอบของการวนซ้ำ iterable คือ ที่เก็บหลายค่า (คอลเลกชัน) ของค่าที่สามารถวนซ้ำได้ เช่น list range หรือ string

ตัวอย่างที่ 1 การวนซ้ำด้วย for loop จากรูปเดิมที่ 3.8 Flowchart ของโปรแกรมการวนทำซ้ำ พิมพ์ข้อความ "Hello, world." จำนวน 5 รอบ โปรแกรมจะสิ้นสุดการทำงาน สามารถเขียนเป็นโค้ดแบบ for loop การวนซ้ำในช่วงตัวเลขด้วย range()

```
for i in range(5):  
    print("Hello, world.")
```

ผลลัพธ์

```
Hello, world.  
Hello, world.  
Hello, world.  
Hello, world.  
Hello, world.
```

โปรแกรมนี้ใช้ range(5) เพื่อกำหนดช่วงตัวเลขตั้งแต่ 0 ถึง 4 โดยในแต่ละรอบตัวแปร i จะเก็บค่าตัวเลข และพิมพ์ข้อความ "Hello, world." จนครบ 5 รอบ

ตัวอย่างที่ 2 การวนซ้ำในรายการ (List) ของชื่อผลไม้หลายชนิด fruits และแสดงข้อความว่า "I like [ชื่อผลไม้]" สำหรับแต่ละผลไม้ในรายการ fruits โดยใช้ลูป for เพื่อเข้าถึงค่าภายในทีล็อกตัว

```
fruits = ['apple', 'banana', 'cherry']  
  
for fruit in fruits:  
    print(f"I like {fruit}.")
```

ผลลัพธ์

```
I like apple.  
I like banana.  
I like cherry.
```

ในตัวอย่างนี้ for loop วนซ้ำในแต่ละค่าของรายการ fruits และนำค่ามาใส่ในตัวแปร fruit เพื่อพิมพ์ข้อความ

ตัวอย่างที่ 3 การวนซ้ำในข้อความ (String) ตัวอย่างของข้อความ "Python" ที่จะวน โดยใช้ลูป for และพิมพ์ตัวอักษรแต่ละตัวออกมากีฬบรรทัด

```
for char in "Python":  
    print(char)
```

ผลลัพธ์

```
P  
y  
t  
h  
o  
n
```

โปรแกรมจะวนซ้ำตัวอักษรแต่ละตัวในข้อความ "Python" และพิมพ์ออกมากีฬบรรทัด

ตัวอย่างที่ 4 การวนซ้ำใน字典ชันนารี (Dictionary) เป็นโครงสร้างข้อมูลที่สำคัญใน Python ซึ่งเก็บข้อมูลในรูปแบบคู่ของ Key:Value โดยที่ Key ทำหน้าที่เป็นตัวระบุหรือชื่อที่ไม่ซ้ำกัน (unique) และ Value คือค่าที่สัมพันธ์กับ Key นั้น ซึ่งสามารถเป็นข้อมูลประเภทใดก็ได้ เช่น ตัวเลข ข้อความ หรือแม้แต่โครงสร้างข้อมูลอื่น ดังนี้การวนซ้ำใน Dictionary ช่วยให้สามารถเข้าถึงข้อมูลแต่ละคู่ Key:Value เพื่อประมวลผลหรือแสดงผลข้อมูลได้ง่าย

```
data = {"name": "Alice", "age": 25}  
  
for key in data:  
    print(key, ":", data[key])
```

ผลลัพธ์

```
name : Alice  
age : 25
```

ใช้ for key in data เพื่อเข้าถึง Key ในแต่ละรอบของลูป ใช้ data[key] เพื่อเข้าถึง Value ที่สัมพันธ์กับ Key นั้น

หรือสามารถที่จะดึงทั้ง key และ value ของแต่ละคู่ข้อมูล

```
data = {"name": "Alice", "age": 25}  
  
for key, value in data.items():  
    print(key, ":", value)
```

ผลลัพธ์

```
name : Alice
age : 25
```

key จะเก็บ Key ของ Dictionary value จะเก็บ Value ที่สัมพันธ์กับ Key นั้น
นอกจากนี้ ยังสามารถ การแสดงเฉพาะ Key หรือ Value ผ่านฟังก์ชัน keys() และ values()

```
for key in data.keys():
    print(key)
```

ผลลัพธ์

```
name
age
```

สำหรับ Value

```
for key in data.values():
    print(value)
```

ผลลัพธ์

```
Alice
25
```

การใช้ range()

range() เป็นฟังก์ชันใน Python ที่ใช้สร้างลำดับตัวเลข ซึ่งหมายความว่าการทำงานร่วมกับลูป for หรือการประมวลผลที่ต้องการการนับจำนวนรอบ โดย range() สามารถกำหนดค่าเริ่มต้น (start) ค่าจบ (stop) และค่าก้าว (step) เพื่อสร้างลำดับตัวเลขตามที่ต้องการได้ ฟังก์ชัน range() มี 3 รูปแบบการใช้งาน หลักๆ ดังนี้

- range(stop) สร้างลำดับตัวเลขตั้งแต่ 0 ถึงก่อน stop (ไม่นับ stop)

```
for i in range(5):
    print(i)
```

ผลลัพธ์

```
0
1
2
3
4
```

- range(start, stop) สร้างลำดับตัวเลขตั้งแต่ start ถึงก่อน stop (ไม่นับ stop)

```
for i in range(2, 6):
    print(i)
```

ผลลัพธ์

```
2
3
4
5
```

3. range(start, stop, step) สร้างลำดับตัวเลขตั้งแต่ start ถึงก่อน stop (ไม่นับ stop) โดยเพิ่มค่า ด้วย step

```
for i in range(1, 10, 2):
    print(i)
```

ผลลัพธ์

```
1
3
5
7
9
```

คุณสมบัติของ range() ประยุกต์หน่วยความจำ range() ไม่สร้างรายการตัวเลขทั้งหมดในหน่วยความจำ แต่จะสร้างค่าตามที่ต้องการในแต่ละรอบลูป ทำให้มีประสิทธิภาพสูง และรองรับค่าก้าวเป็นลบสามารถสร้างลำดับตัวเลขแบบถอยหลังได้

```
for i in range(10, 0, -2):
    print(i)
```

ผลลัพธ์

```
10
8
6
4
2
```

range() สามารถนำไปใช้ร่วมกับฟังก์ชันอื่น เช่น list() เพื่อแปลงเป็นรายการ

```
print(list(range(5)))
```

ผลลัพธ์

```
[0, 1, 2, 3, 4]
```

การใช้คำสั่ง break

คำสั่ง break ใน Python ใช้สำหรับ หยุดการทำงานของลูปทันที เมื่อถึงเงื่อนไขที่กำหนดโดยไม่ต้องรอให้ลูปทำงานจนครบรอบ การใช้ break ในลูป for ช่วยเพิ่มความยืดหยุ่นในการควบคุมลูปโดยเฉพาะในสถานการณ์ที่ต้องการหยุดการวนซ้ำทันทีที่พบค่าหรือเงื่อนไขที่ต้องการ

เมื่อโปรแกรมพบคำสั่ง break ในลูป for การวนซ้ำจะหยุดทันที และโปรแกรมจะออกจากลูปโดยไม่รันคำสั่งที่เหลือในลูป และใช้ในกรณีที่ต้องการค้นหาค่าที่ต้องการในลูป และหยุดเมื่อพบค่าโดยไม่ต้องวนซ้ำจนจบทั้งหมด ดังตัวอย่างที่ 1 หยุดลูปเมื่อเจอค่าที่ต้องการ

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    if num == 3:
        print(f"Found {num}! Stopping the loop.")
        break
    print(f"Checking {num}...")
```

ผลลัพธ์

```
Checking 1...
Checking 2...
Found 3! Stopping the loop.
```

ในตัวอย่างนี้ ลูปจะหยุดทันทีเมื่อ num มีค่าเท่ากับ 3 โดยไม่ตรวจสอบค่าที่เหลือในรายการ
ตัวอย่างที่ 2 ค้นหาข้อมูลในข้อความ

```
text = "Python programming"

for char in text:
    if char == 'g':
        print(f"Found '{char}' in the text!")
        break
    print(f"Checking '{char}'...")
```

ผลลัพธ์

```
Checking 'P'...
Checking 'y'...
Checking 't'...
Checking 'h'...
Checking 'o'...
Checking 'n'...
Checking ' '...
Checking 'p'...
Checking 'r'...
Checking 'o'...
Checking 'g'...
Found 'g' in the text!
```

ลูปจะหยุดทันทีเมื่อพบตัวอักษร 'g' ในข้อความ และออกจากลูป
ตัวอย่างที่ 3 หยุดการค้นหาในช่วงตัวเลข

```
for i in range(10):
    if i == 5:
        print(f"Stopping at {i}")
        break
    print(f"Processing {i}...")
```

ผลลัพธ์

```
Processing 0...
Processing 1...
Processing 2...
Processing 3...
Processing 4...
Stopping at 5
```

ในตัวอย่างนี้ break ใช้หยุดลูปทันทีเมื่อค่าของ i เป็น 7

คำสั่ง break ในลูป for เป็นเครื่องมือสำคัญที่ช่วยให้สามารถหยุดการวนซ้ำได้ทันทีตามเงื่อนไขที่กำหนด ช่วยลดระยะเวลาการประมวลผลและทำให้โค้ดมีประสิทธิภาพมากขึ้น โดยเฉพาะในกรณีที่ต้องค้นหาหรือจับคู่ข้อมูลในรายการ หรือหยุดการทำงานเมื่อพบข้อผิดพลาดเฉพาะในลูป

การใช้คำสั่ง continue

คำสั่ง continue ใน Python ใช้สำหรับ ข้ามคำสั่งที่เหลือในรอบปัจจุบันของลูป และกลับไปเริ่มรอบถัดไปของลูปทันที โดยไม่หยุดการทำงานของลูปทั้งหมด การใช้ continue ช่วยจัดการเงื่อนไขที่ไม่ต้องการให้ดำเนินการต่อในรอบนั้น เช่น การข้ามค่าที่ไม่ตรงตามเงื่อนไข หรือการกรองข้อมูลในลูป

หลักการทำงานของ continue กับ For Loop เมื่อโปรแกรมพบคำสั่ง continue ในลูป for จะข้ามคำสั่งที่เหลือในรอบปัจจุบันทันที และโปรแกรมจะวนซ้ำกลับไปเริ่มต้นรอบถัดไปตามลำดับของ iterable

ตัวอย่างที่ 1 ข้ามค่าที่ไม่ต้องการ

```
numbers = [1, 2, 3, 4, 5]

for num in numbers:
    if num % 2 == 0: # หากเป็นเลขคู่
        continue      # ข้ามเลขคู่นั้น
    print(num)
```

ผลลัพธ์

```
1
3
5
```

ในตัวอย่างนี้ คำสั่ง continue จะข้ามค่าที่เป็นเลขคู่ (2 และ 4) และดำเนินการต่อไปในรอบถัดไป
ตัวอย่างที่ 2 กรองข้อมูลจากข้อความ

```
text = "Python programming"

for char in text:
    if char in "aeiou": # หากตัวอักษรเป็นสระ
        continue          # ข้ามตัวอักษรนั้น
    print(char, end="")
```

ผลลัพธ์

```
Pytn prgrmmng
```

คำสั่ง continue ในตัวอย่างนี้จะข้ามตัวอักษรที่เป็นสระ (a, e, i, o, u) และแสดงเฉพาะตัวอักษรที่ไม่ใช่สระ

ตัวอย่างที่ 3 การข้ามข้อผิดพลาดในข้อมูล

```
data = [10, "error", 20, "error", 30]
for item in data:
    if item == "error": # หากเจอกำลัง 'error'
        continue          # ข้ามไปยังรอบถัดไป
    print(item)
```

ผลลัพธ์

10
20
30

คำสั่ง continue ในลูป for ช่วยเพิ่มความยืดหยุ่นในการควบคุมลูป โดยเฉพาะในกรณีที่ต้องการข้ามค่าหรือเงื่อนไขที่ไม่ต้องการประมวลผลในรอบนั้น ๆ คำสั่งนี้ช่วยลดความซับซ้อนของโค้ด และทำให้โปรแกรมสามารถจัดการข้อมูลได้อย่างมีประสิทธิภาพและซัดเจนมากขึ้น

3.4 การเขียนโปรแกรมเพิ่มเติมการสำหรับการควบคุมโปรแกรม

การเขียนโปรแกรมสำหรับการควบคุมโปรแกรม เช่น การสร้างตารางสูตรคูณ การแสดงรูปประจำ และเกมทายตัวเลข เป็นเครื่องมือสำคัญในการเรียนรู้พื้นฐานการเขียนโค้ดใน Python โปรแกรมเหล่านี้ช่วยเสริมสร้างทักษะการใช้ลูป (Loop) การจัดการตัวแปร (Variables) และการสร้างโปรแกรมเชิงโต้ตอบ โปรแกรมสูตรคูณช่วยให้เข้าใจการประมวลผลข้อและการแสดงผลในรูปแบบตาราง ส่วนโปรแกรมสร้างรูปประจำเน้นการใช้ลูปซ้อน (Nested Loop) เพื่อสร้างรูปแบบต่าง ๆ เช่น สี่เหลี่ยมหรือสามเหลี่ยม และสุดท้ายโปรแกรมเกมทายตัวเลขนำเสนอการใช้เงื่อนไข (if-else) และการจัดการโมดูล random เพื่อพัฒนาเกมที่โต้ตอบได้

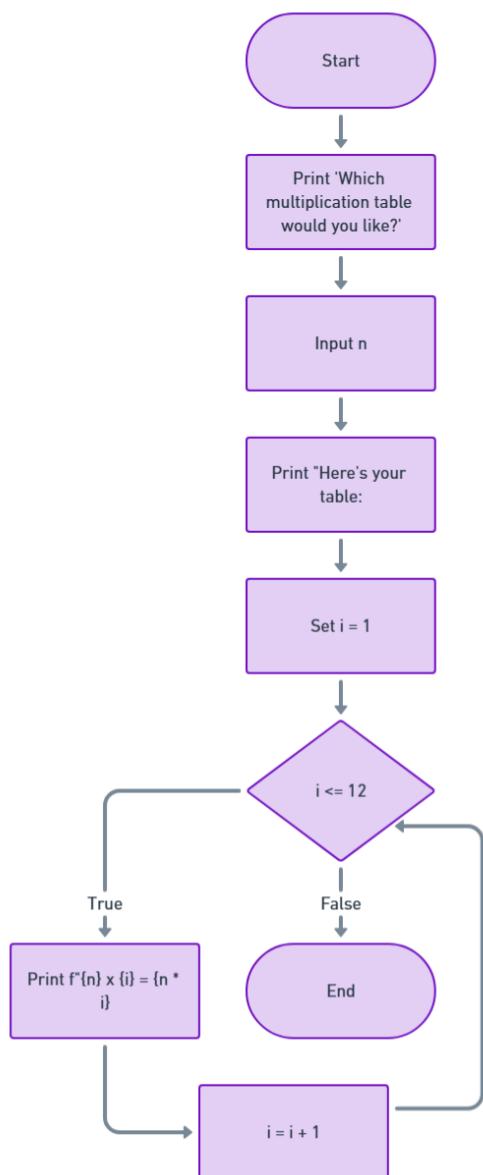
3.4.1 การเขียนโปรแกรมลักษณะตารางแบบสูตรคูณ

การเขียนโปรแกรมเพื่อแสดงตารางสูตรคูณเป็นตัวอย่างที่ดีสำหรับการเริ่มต้นเรียนรู้ การวนซ้ำ (Loop) และการทำงานกับ ตัวแปร ใน Python โปรแกรมลักษณะนี้มีความเรียบง่าย แต่ทรงพลัง เพราะช่วยให้ผู้เรียนเข้าใจแนวคิดพื้นฐาน เช่น การรับค่าจากผู้ใช้ การใช้ลูป for หรือ while เพื่อทำซ้ำคำสั่ง และการจัดรูปแบบผลลัพธ์ให้อ่านง่ายเหมือนตาราง

โปรแกรมตารางสูตรคูณมีความสำคัญในการช่วยผู้เริ่มต้นเรียนรู้การประมวลผลข้อและการคำนวณพร้อมทั้งแสดงผลข้อมูลในรูปแบบตารางที่เข้าใจง่ายและเป็นระเบียบ นอกจากนี้ยังสามารถปรับใช้ในงานที่เกี่ยวข้องกับการคำนวณข้อ เช่น ตารางดอกเบี้ยหรือแผนการผ่อนชำระ

องค์ประกอบของโปรแกรม ตามการทำงาน Flowchart รูปที่ 3.10

- รับค่าจากผู้ใช้ เพื่อให้ผู้ใช้กำหนดตัวเลขที่ต้องการสร้างตาราง
- การใช้ลูป ใช้ลูป for หรือ while เพื่อคำนวณค่าผลคูณในแต่ละรอบ
- การแสดงผลในรูปแบบตาราง ใช้คำสั่ง print() และการจัดรูปแบบผลลัพธ์ เช่น การใช้ f-string เพื่อให้อ่านง่ายและเป็นระเบียบ



รูปที่ 3.10 Flowchart ของโปรแกรมสร้างตารางสูตรคูณ

โค้ดการทำงาน

```

print("Which multiplication table would you like?")
n = int(input("Enter a number: "))

print("Here's your table:")
for i in range(1, 13):
    print(f"{n} x {i} = {n * i}")
  
```

ผลลัพธ์

```

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
  
```

```
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
5 x 12 = 60
```

คำอธิบายโค้ด

- รับค่าจากผู้ใช้

```
n = int(input())
```

ผู้ใช้ป้อนตัวเลขที่ต้องการ (เช่น 5) เพื่อสร้างตารางสูตรคูณของตัวเลขนั้น ใช้ int() เพื่อแปลงข้อมูลที่รับมาเป็นจำนวนเต็ม (integer)

- แสดงข้อความบอกตารางสูตรคูณ

```
print("Here's your table:")
```

- สร้างตารางสูตรคูณด้วยลูป for

```
for i in range(1, 13):
    a = n * i
    print(f"{n} x {i} = {a}")
```

for i in range(1, 13): วนซ้ำค่าของ i ตั้งแต่ 1 ถึง 12 (12 รอบ) a = n * i คำนวณผลคูณของ n (ค่าที่ผู้ใช้ป้อน) และ i และ print(f"{n} x {i} = {a}") พิมพ์ผลลัพธ์ในรูปแบบตารางสูตรคูณ

3.4.2 การเขียนโปรแกรมสร้างรูปร่างด้วยลูป

การเขียนโปรแกรมเพื่อสร้างรูปร่างต่าง ๆ เช่น เส้นตรง สี่เหลี่ยม และสามเหลี่ยม เป็นการฝึกการใช้ลูปซ้อน (Nested Loop) และการจัดการเงื่อนไขใน Python การสร้างรูปร่างช่วยให้ผู้เรียนเข้าใจการควบคุมการทำงานของลูปและการจัดรูปแบบผลลัพธ์ให้เป็นระเบียบ

โปรแกรมสร้างรูปร่างช่วยเสริมทักษะการเขียนโปรแกรมพื้นฐาน โดยเน้นการใช้ลูปและตัวแปรอย่างมีประสิทธิภาพ รวมถึงฝึกการควบคุมลูปซ้อนและการจัดการเงื่อนไขเพื่อสร้างรูปแบบต่าง ๆ ที่ซับซ้อนยิ่งขึ้น นอกจากนี้ยังสามารถปรับแต่งเพื่อนำไปประยุกต์ใช้งาน เช่น สร้างกรอบข้อมูลหรือการแสดงผลกราฟิกในคอมโฉล

- พิมพ์เครื่องหมาย # ในแนวตั้ง

```
for i in range(5):
    print("#")
```

ผลลัพธ์

```
#  
#  
#  
#  
#
```

2. พิมพ์เครื่องหมาย # ในแนวตั้ง

```
for i in range(5):
    print("#", end="")
```

ผลลัพธ์

#####

ข้อสังเกต แนวตั้ง ใช้ค่าเริ่มต้นของ print() ที่ขึ้นบรรทัดใหม่ทุกครั้ง ทำให้ผลลัพธ์เรียงลงในแนวตั้ง แนวอน ใช้ end="" เพื่อควบคุมไม่ให้ขึ้นบรรทัดใหม่ ทำให้ผลลัพธ์ต่อเนื่องในบรรทัดเดียวกัน

3. พิมพ์เครื่องหมาย # แบบสี่เหลี่ยม

```
row = 5
col = 7
for i in range(row):
    for j in range(col):
        print("#", end="")
    print()
```

ผลลัพธ์

#####
#####

อธิบายโค้ด

- ตัวแปร row และ col row กำหนดจำนวนແລກ (5 ແລກ) col กำหนดจำนวนຄອລັມນ໌ (7 ຄອລັມນ໌ ในແຕ່ລະແລກ)
- ລູປ້ອນກັນ (Nested Loop) ລູປແຮກ (for i in range(row)) ຄວບຄຸມຈຳນວນແລກ (row) ໂດຍວນຊ້າ 5 ຮອບ ລູປທີ່ສອງ (for j in range(col)) ຄວບຄຸມຈຳນວນຄອລັມນ໌ (col) ໃນແຕ່ລະແລກ ໂດຍວນຊ້າ 7 ຮອບ
- คำสั่ง print("#", end="") พิมพ์ເຄື່ອງໝາຍ # ໃນແຕ່ລະຄອລັມນ໌ໃນบรรທັດເດືອນ ໂດຍໄມ່ຂຶ້ນบรรທັດໃໝ່ ໃຫ້ຜົນໄຟໃນລູປທີ່ສອງຈະຕ່ອນໄຟໃນແນວອນ
- คำสั่ง print() (ໄມ່ມີຂໍອຄວາມ) ໃຊ້ສໍາຮັບຂຶ້ນบรรທັດໃໝ່ໜ່າງຈາກລູປທີ່ສອງ (j) ທຳກຳຈຳນວນ ຄອລັມນ໌ໃນແຕ່ລະແລກ

ໃນ แนวตั้ง ໂດຍໃຫ້ລູປຄວບຄຸມຈຳນວນແລກ (row) ເພື່ອແສດງຜົນແຕ່ລະບຣທັດ ໃນ ແນວອນ ໂດຍໃຫ້ ລູປຄວບຄຸມຈຳນວນຄອລັມນ໌ (col) ເພື່ອແສດງຜົນຕ່ອນໄຟໃນບຣທັດເດືອນ ເນື່ອຜົນການທຳກຳຈຳນວນຂອງລູປ້ອນ (row ແລກ col) ຈະໄດ້ຜົນໄຟແບບ ຕາຮາງສື່ເລື່ອຍ່າມ ທີ່ມີຈຳນວນແລກແລກຄອລັມນ໌ຕາມທີ່ກຳນົດ

4. ຕ້ອງຢ່າງເພີ່ມເຕີມການທຳຽຸປສາມເຫີ່ມທັງ 4 ທິດທາງ ດັ່ງຕາຮາງທີ່ 3.4 ປັບການໃຫ້ລູປ້ອນ (for) ແລກ ເພີ່ມການພິມພົມ ຊົ່ວໂມງ (" ") ກ່ອນຫຼັງເຄື່ອງໝາຍ # ເພື່ອສ້າງສາມເຫີ່ມໃນທິດທາງຕ່າງໆ ໄດ້ ທັ້ງເອີ່ນຫ້າຍ

เอียงขวา กลับหัว และกลับหัวเอียงขวา โดยโค้ดที่ปรับปรุงนี้ทำให้ครอบคลุมการสร้างรูปแบบสามเหลี่ยมครบถ้วนทิศทาง การเขียนรูปแบบ สี่เหลี่ยม และ สามเหลี่ยม มีความสัมพันธ์ในเรื่องของการใช้ ลูปซ้อน (Nested Loop) โดยลูปแรกควบคุมจำนวนแถว และลูปที่สองควบคุมจำนวน colum หรือการพิมพ์สัญลักษณ์ในแต่ละแถว สำหรับสี่เหลี่ยม ลูป colum จะมีจำนวนคงที่ในทุกแถว แต่สำหรับสามเหลี่ยม ลูป colum จะเปลี่ยนตามเงื่อนไข (เพิ่มหรือลด) ในแต่ละแถวเพื่อสร้างลักษณะเฉพาะของสามเหลี่ยม

ตารางที่ 3.4 ตารางโค้ดตัวอย่างสามเหลี่ยมทั้ง 4 ทิศทาง

โค้ด	ผลลัพธ์
<p>1. สามเหลี่ยมแบบเอียงซ้าย (Left-Aligned Triangle)</p> <pre>row = 5 for i in range(row): for j in range(i + 1): print("#", end="") print()</pre>	<pre># ## ### #### #####</pre>
<p>2. สามเหลี่ยมแบบกลับหัว (Inverted Triangle)</p> <pre>row = 5 for i in range(row): for j in range(row - i): print("#", end="") print()</pre>	<pre>##### #### ### ## #</pre>
<p>3. สามเหลี่ยมแบบเอียงขวา (Right-Aligned Triangle)</p> <pre>row = 5 for i in range(row): for j in range(row - i - 1): print(" ", end="") for j in range(i + 1): print("#", end="") print()</pre>	<pre> # ## ### #### #####</pre>
<p>4. สามเหลี่ยมแบบกลับหัวเอียงขวา (Inverted Right-Aligned Triangle)</p> <pre>row = 5 for i in range(row): for j in range(i): print(" ", end="") for j in range(row - i): print("#", end="") print()</pre>	<pre>##### #### ### ## #</pre>

3.4.3 การเขียนโปรแกรมคำนวณหาค่าผลรวม หรือค่าทางคณิตศาสตร์

การเขียนโปรแกรมสำหรับคำนวณค่าผลรวม หรือการคำนวณทางคณิตศาสตร์อื่น ๆ เป็นหัวข้อสำคัญที่ช่วยให้ผู้เรียนเข้าใจ ลูป (Loop) และการจัดการ ตัวแปร (Variables) ได้อย่างมีประสิทธิภาพ โปรแกรมในลักษณะนี้สามารถประยุกต์ใช้กับโจทย์ทางคณิตศาสตร์หลากหลายรูปแบบ เช่น การหาผลรวมของตัวเลข การคำนวณค่าเฉลี่ย การหาค่าผลคูณ หรือการหาค่าเลขยกกำลัง

การเขียนโปรแกรมคำนวณทางคณิตศาสตร์ช่วยเสริมความเข้าใจพื้นฐานการเขียนโปรแกรม เช่น การใช้ลูป ตัวแปร และการจัดการค่าทางคณิตศาสตร์ในโปรแกรม นอกจากนี้ยังฝึกการประยุกต์ใช้ในสถานการณ์จริง เช่น การคำนวณยอดขายหรือการวิเคราะห์ข้อมูล พร้อมทั้งช่วยลดขั้นตอนการคำนวณที่ซับซ้อนและเพิ่มความรวดเร็วในการแก้ปัญหา

1. โปรแกรมคำนวณค่าผลรวม คำนวณผลรวมของตัวเลขตั้งแต่ 1 ถึง n

```
n = int(input("Enter a number: "))
total = 0

for i in range(1, n + 1):
    total += i

print(f"The sum of numbers from 1 to {n} is {total}")
```

ผลลัพธ์

```
The sum of numbers from 1 to 5 is 15
```

รับค่าตัวเลข n จากผู้ใช้ ใช้ตัวแปร total สำหรับเก็บค่าผลรวม และเริ่มต้นที่ 0 ใช้ลูป for วนซ้ำตั้งแต่ 1 ถึง n และเพิ่มค่าตัวเลขที่ลงทะเบียนไว้ใน total แสดงผลลัพธ์ค่าผลรวม

2. โปรแกรมคำนวณค่าผลคูณ (Factorial)

```
n = int(input("Enter a number: "))
factorial = 1

for i in range(1, n + 1):
    factorial *= i

print(f"The factorial of {n} is {factorial}")
```

ผลลัพธ์

```
The factorial of 5 is 120
```

รับค่าตัวเลข n จากผู้ใช้ ใช้ตัวแปร factorial สำหรับเก็บผลคูณ และเริ่มต้นที่ 1 และใช้ลูป for วนซ้ำตั้งแต่ 1 ถึง n และคูณค่าที่ลงทะเบียนไว้ใน factorial

4. คำนวณหาค่าเลขยกกำลัง

```
base = int(input("Enter the base: "))
exponent = int(input("Enter the exponent: "))
result = 1
```

```

for i in range(exponent):
    result *= base

print(f"{base} to the power of {exponent} is {result}")

```

ผลลัพธ์

2 to the power of 3 is 8

รับค่าฐาน (base) และเลขยกกำลัง (exponent) จากผู้ใช้ ใช้ตัวแปร result สำหรับเก็บค่าผลลัพธ์ และเริ่มต้นที่ 1 ใช้ลูป for วนซ้ำเท่ากับจำนวนของเลขยกกำลัง และคุณค่าฐานกับ result ในแต่ละรอบ และแสดงผลลัพธ์ค่าเลขยกกำลัง

3.4.4 การเขียนโปรแกรมภาษาตัวเลข

การเขียนโปรแกรมเกมทายตัวเลขเป็นตัวอย่างที่สนุกและท้าทายสำหรับการฝึกเขียนโปรแกรมเชิงโต้ตอบ โปรแกรมลักษณะนี้ช่วยเสริมทักษะการเขียนโค้ดพื้นฐาน เช่น การใช้คำสั่ง if-else, การใช้ลูป, และการจัดการโมดูล random เพื่อสุ่มตัวเลข โปรแกรมเกมทายตัวเลขสามารถปรับแต่งให้เหมาะสมกับระดับความยาก และช่วยให้ผู้เรียนเข้าใจแนวคิดการพัฒนาเกมเบื้องต้น

โปรแกรมเกมทายตัวเลขช่วยฝึกการใช้เงื่อนไข if-else และลูป for เพื่อควบคุมจำนวนครั้งในการทายพร้อมทั้งเรียนรู้การใช้โมดูล random สำหรับการสุ่มตัวเลข นอกจากนี้ยังเสริมทักษะการสร้างโปรแกรมเชิงโต้ตอบด้วย input() เพื่อโต้ตอบและให้คำแนะนำสำหรับผู้เล่นได้อย่างเหมาะสม

องค์ประกอบของโปรแกรม ตามการทำงาน Flowchart รูปที่ 3.11

- เริ่มต้น (Start) โปรแกรมเริ่มต้นและสุ่มตัวเลขลับ (secretNumber) จาก 1 ถึง 20

- พิมพ์ข้อความแสดงข้อความแจ้งผู้เล่นว่าโปรแกรมกำลังคิดเลขระหว่าง 1 ถึง 20

- ลูปสำหรับการทาย (7 ครั้ง) ให้ผู้เล่นทายเลขในแต่ละรอบ (สูงสุด 7 ครั้ง)

- เปรียบเทียบการทายกับตัวเลขลับ (secretNumber)

- หากทายต่ำเกินไป พิมพ์ "Your guess is too low." และไปยังรอบถัดไป

- หากทายสูงเกินไป พิมพ์ "Your guess is too high." และไปยังรอบถัดไป

- หากทายถูก หยุดลูปด้วยคำสั่ง break

- ตรวจสอบคำตอบหลังจบลูป หากผู้เล่นทายถูก แสดงข้อความแสดงความยินดีพร้อมจำนวนครั้งที่ใช้ทาย หากทายไม่ถูกภายใน 7 ครั้ง แสดงตัวเลขลับและจบเกม

- จบโปรแกรม (End) สิ้นสุดการทำงานของเกม. โค้ดการทำงาน

```

import random

# สุ่มตัวเลขลับ
secretNumber = random.randint(1, 20)
print('I am thinking of a number between 1 and 20.')

# ให้ผู้เล่นทาย 7 ครั้ง

```

```

for guessesTaken in range(1, 7):
    print(f"Take a guess, round {guessesTaken}")
    guess = int(input("Enter number: "))

    if guess < secretNumber:
        print('Your guess is too low.')
    elif guess > secretNumber:
        print('Your guess is too high.')
    else:
        break # ทายถูกแล้วออกจากลูป

# ตรวจสอบคำตอบ
if guess == secretNumber:
    print(f'Good job! You guessed my number in {guessesTaken} guesses!')
else:
    print(f'Nope. The number I was thinking of was {secretNumber}')

```

ผลลัพธ์ กรณีทายถูก

```

I am thinking of a number between 1 and 20.
Take a guess, round 1
Enter number: 10
Your guess is too low.
Take a guess, round 2
Enter number: 15
Your guess is too low.
Take a guess, round 3
Enter number: 17
Your guess is too high.
Take a guess, round 4
Enter number: 16
Good job! You guessed my number in 4 guesses!

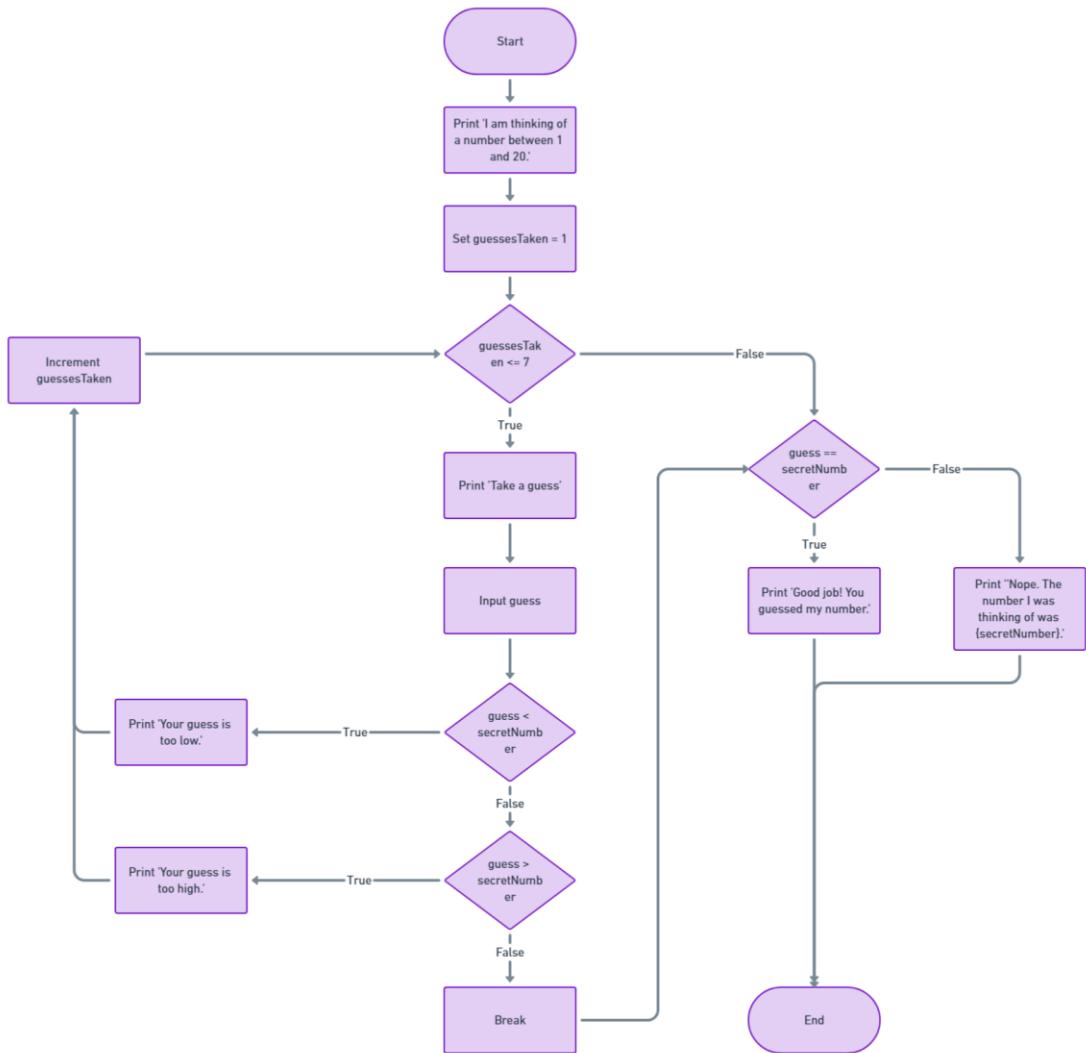
```

ผลลัพธ์ กรณีทายผิด

```

I am thinking of a number between 1 and 20.
Take a guess, round 1
Enter number: 8
Your guess is too low.
Take a guess, round 2
Enter number: 18
Your guess is too high.
Take a guess, round 3
Enter number: 12
Your guess is too low.
Take a guess, round 4
Enter number: 15
Your guess is too high.
Take a guess, round 5
Enter number: 13
Your guess is too low.
Take a guess, round 6
Enter number: 14
Nope. The number I was thinking of was 16.

```



รูปที่ 3.11 Flowchart ของโปรแกรมเกมทายตัวเลข

คำอธิบายโค้ดโครงสร้างโปรแกรม

- การสุ่มตัวเลขลับ ใช้โมดูล random เพื่อสุ่มตัวเลขลับในช่วงที่กำหนด เช่น 1 ถึง 20

```

import random

# สรุมตัวเลขลับ
secretNumber = random.randint(1, 20)
print('I am thinking of a number between 1 and 20.')

```

import random คือ การนำเข้าไลบรารีเพื่อการใช้งานการสุ่ม (random) หรือ คำสั่งสำเร็จรูปที่ถูกพัฒนาช่วยการทำงานง่ายขึ้น

random.randint(1, 20) พังก์ชันนี้ใช้สำหรับสุ่มตัวเลขจำนวนเต็มในช่วงที่กำหนด (รวมค่าเริ่มต้นและค่าสุดท้าย) ในตัวอย่างนี้ สุ่มตัวเลขระหว่าง 1 ถึง 20 ผลลัพธ์ที่ได้คือ 9 ซึ่งเป็นตัวเลขหนึ่งในช่วงที่กำหนด

ตัวอย่างการสุ่มอื่นๆ random.random() ฟังก์ชันนี้ใช้สำหรับสุ่มตัวเลขแบบทศนิยมในช่วง [0.0, 1.0) (ตั้งแต่ 0.0 ถึงน้อยกว่า 1.0) เช่นตัวอย่าง ผลลัพธ์ที่ได้คือ 0.4688785588458374 ซึ่งเป็นค่าทศนิยมในช่วงดังกล่าว

การใช้งานเพิ่มเติม

random.randint ใช้ในกรณีที่ต้องการสุ่มตัวเลขจำนวนเต็ม เช่น เกมทายตัวเลข, การสุ่มตำแหน่ง หรือการสุ่มตัวเลขในช่วงที่กำหนด

random.random ใช้ในกรณีที่ต้องการสุ่มตัวเลขทศนิยม เช่น การคำนวณทางสถิติ การสุ่มเบอร์เซ็นต์ หรือการสร้างค่าแบบสุ่มเพื่อจำลองสถานการณ์

ตัวอย่างโค้ด

```
# สุ่มตัวเลขจำนวนเต็มระหว่าง 1 ถึง 100
random_integer = random.randint(1, 100)
print(f"Random integer: {random_integer}")

# สุ่มตัวเลขทศนิยมระหว่าง 0.0 ถึง 1.0
random_decimal = random.random()
print(f"Random decimal: {random_decimal}")
```

ผลลัพธ์ กรณีทายถูก

```
integer: 57
Random decimal: 0.762348754838475
```

2. การกำหนดลูปทายตัวเลข ใช้ลูป for เพื่อให้ผู้เล่นทายตัวเลขสูงสุด 7 ครั้ง และรับค่าจากผู้ใช้ด้วย input()

```
for guessesTaken in range(1, 7):
    print(f"Take a guess, round {guessesTaken}")
    guess = int(input("Enter number: "))
```

3. การตรวจสอบคำตอบ ใช้คำสั่ง if-elif-else เพื่อตรวจสอบว่าตัวเลขที่ผู้เล่นทาย สูงเกินไป, ต่ำเกินไป, หรือ ถูกต้อง

```
if guess < secretNumber:
    print('Your guess is too low.')
elif guess > secretNumber:
    print('Your guess is too high.')
else:
    break # หากถูกแล้วออกจากลูป
```

4. การแสดงผลลัพธ์ หลังจากจบลูป ตรวจสอบว่าผู้เล่นทายถูกหรือไม่ และแสดงผลลัพธ์ที่เหมาะสม

```
# ตรวจสอบคำตอบ
if guess == secretNumber:
    print(f'Good job! You guessed my number in {guessesTaken} guesses!')
else:
    print(f'Nope. The number I was thinking of was {secretNumber}')
```

โปรแกรมเหล่านี้ช่วยเสริมความเข้าใจพื้นฐานการเขียนโค้ด ไม่ว่าจะเป็นการใช้ลูป การจัดการตัวแปร หรือการสร้างโปรแกรมที่มีปฏิสัมพันธ์กับผู้ใช้ โปรแกรมสูตรคูณและรูปร่างช่วยฝึกการจัดการโครงสร้างลูปอย่างมีประสิทธิภาพ ในขณะที่เกมไทยตัวเลขช่วยฝึกการใช้เงื่อนไขและโมดูลใน Python โปรแกรมทั้งหมดนี้ เป็นพื้นฐานที่สำคัญสำหรับการพัฒนาทักษะการเขียนโปรแกรมและการประยุกต์ใช้ในงานที่ซับซ้อนยิ่งขึ้น

การนำโครงสร้างควบคุมมาใช้กับโปรแกรมง่ายๆ อย่างตารางสูตรคูณหรือเกมไทยตัวเลขเป็นวิธีที่ดีในการฝึกประยุกต์ใช้ความรู้เรื่องเงื่อนไขและลูปในสถานการณ์จริง นอกจากนี้ การเขียนโปรแกรมสร้างรูปร่างด้วยลูปซ้อนยังช่วยให้เข้าใจหลักการควบคุมการทำซ้ำในระดับที่ลึกขึ้น ซึ่งเป็นการเตรียมความพร้อมไปสู่การเขียนโปรแกรมในงานที่ซับซ้อนกว่า เช่น การสร้างส่วนติดต่อผู้ใช้ (UI) หรือการเขียนเกมในอนาคต

บทสรุป

การควบคุมการทำงานของโปรแกรม (Flow Control) เป็นหัวใจสำคัญที่ช่วยให้โปรแกรมสามารถทำงานได้อย่างมีประสิทธิภาพและปรับตัวเข้ากับสถานการณ์หรือข้อมูลที่หลากหลายได้อย่างยืดหยุ่น บทนี้จะนำไปสู่การเรียนรู้โครงสร้างควบคุมพื้นฐาน ไม่ว่าจะเป็นการตัดสินใจด้วยคำสั่งเงื่อนไข (Conditional Statements) การทำซ้ำ (Loops) และการนำไปประยุกต์ใช้กับสถานการณ์จริง

การใช้คำสั่ง if, if-else และ if-elif-else ช่วยให้โปรแกรมสามารถเลือกเส้นทางการทำงานได้อย่างเหมาะสมตามเงื่อนไขที่กำหนดไว้ ในขณะที่การใช้ลูป while และ for ช่วยลดความซ้ำซ้อนของโค้ด ทำให้โปรแกรมสามารถทำซ้ำกระบวนการได้หลายครั้งจนกว่าจะตรงตามเงื่อนไข นอกจากนี้ ยังจะได้เรียนรู้คำสั่ง break และ continue เพื่อเพิ่มความยืดหยุ่นและประสิทธิภาพในการควบคุมการทำงานของลูปให้เหมาะสมกับสถานการณ์ต่างๆ

บทนี้จะนำเสนอการเขียนโปรแกรมในรูปแบบที่ประยุกต์ใช้โครงสร้างควบคุม เช่น การสร้างตารางสูตรคูณ การวาดรูปร่างด้วยลูปซ้อน และเกมไทยตัวเลข ซึ่งจะช่วยเสริมสร้างทักษะในการนำแนวคิดเรื่อง Flow Control ไปพัฒนาโปรแกรมเชิงโต้ตอบที่สมบูรณ์ยิ่งขึ้น

- Flow Control เป็นหัวใจสำคัญในการกำหนดลำดับการทำงานของโปรแกรมให้เหมาะสมและมีประสิทธิภาพสูงสุด
- โปรแกรมสามารถตอบสนองต่อข้อมูลและสถานการณ์ที่หลากหลายด้วยเงื่อนไข (Condition) และการทำซ้ำ (Loop)
- การเขียนโปรแกรมแบบเงื่อนไข (Conditional Programming) ช่วยให้โปรแกรมตัดสินใจตามสถานการณ์ได้อย่างยืดหยุ่น
- เงื่อนไขในโปรแกรมใช้ Boolean Expression ที่ผลลัพธ์เป็น True หรือ False
- ตัวดำเนินการเปรียบเทียบ เช่น <, >, ==, และ != ช่วยตรวจสอบค่าต่าง ๆ ในเงื่อนไขได้อย่างแม่นยำ
- if-else และ if-elif-else ช่วยในการเลือกเส้นทางการทำงานตามเงื่อนไขที่ซับซ้อน
- การใช้ nested if ช่วยเพิ่มความละเอียดในการตัดสินใจในกรณีที่มีเงื่อนไขย่อย

- while loop เมามากับสถานการณ์ที่ไม่รู้จำนวนรอบล่วงหน้า แต่ต้องการให้ทำงานจนกว่าเงื่อนไขจะเป็น False
- for loop เมามากับสถานการณ์ที่รู้จำนวนรอบล่วงหน้า เช่น การวนซ้ำใน list หรือ range
- คำสั่ง break ช่วยหยุดลูปทันทีเมื่อถึงเงื่อนไขที่กำหนด
- คำสั่ง continue ช่วยข้ามคำสั่งที่เหลือในรอบปัจจุบันของลูปและไปเริ่มต้นรอบใหม่ทันที
- Flowchart ช่วยแสดงกระบวนการของโปรแกรมในรูปแบบภาพและลดความคลุมเครือในการสื่อสาร
- สัญลักษณ์ใน Flowchart เช่น Oval, Rectangle, และ Diamond ช่วยแยกແຍະลำดับการทำงานในโปรแกรมได้ง่ายขึ้น
- การใช้ลูปซ้อน (Nested Loop) ช่วยสร้างรูปร่างหรือรูปแบบที่ซับซ้อน เช่น ตารางสี่เหลี่ยมหรือสามเหลี่ยม
- โปรแกรมวนซ้ำสามารถปรับแต่งเพื่อสร้างรายงานหรือประมวลผลข้อมูลขนาดใหญ่ได้อย่างง่ายดาย
- การใช้ Boolean Logic เช่น and, or, not ช่วยสร้างเงื่อนไขที่ซับซ้อนและมีประสิทธิภาพ
- คำสั่ง elif ลดการตรวจสอบเงื่อนไขเข้าช้อนในโปรแกรม และเพิ่มประสิทธิภาพในการตัดสินใจ
- โปรแกรมที่ดีควรมีโครงสร้างที่ชัดเจน อ่านง่าย และสามารถขยายหรือปรับปรุงได้ในอนาคต
- Flow Control เป็นเครื่องมือสำคัญในการสร้างโปรแกรมที่ตอบสนองต่อสถานการณ์ได้หลากหลาย
- การเรียนรู้และฝึกฝนการเขียนโปรแกรมควบคู่กับคุณช่วยสร้างพื้นฐานที่มั่นคงสำหรับการพัฒนาโปรแกรมขั้นสูงในอนาคต

คำถามท้ายบท

1. อธิบายความหมายของ Flow Control และความสำคัญในการเขียนโปรแกรม
2. เงื่อนไขในโปรแกรมใช้ Boolean Expression อย่างไร? ยกตัวอย่างประกอบ
3. อธิบายการทำงานของคำสั่ง if-else พร้อมตัวอย่างโค้ด
4. ในกรณีเดียวกับใช้ if-elif-else แทน if-else?
5. อธิบายความแตกต่างระหว่าง while loop และ for loop
6. ให้ตัวอย่างการใช้ break ในลูป และอธิบายประโยชน์ของคำสั่งนี้
7. คำสั่ง continue มีหน้าที่อะไร? และต่างจาก break อย่างไร?
8. ให้เขียนโปรแกรมที่ใช้ลูป for เพื่อหาผลรวมของตัวเลขตั้งแต่ 1 ถึง 100
9. อธิบายความสำคัญของการใช้ Nested Loop พร้อมตัวอย่างการสร้างรูปสามเหลี่ยม
10. อธิบายการใช้โมดูล random และการสุ่มตัวเลขใน Python
11. อะไรคือความแตกต่างระหว่าง random.randint() และ random.random()?
12. ให้เขียนโปรแกรมเกม tahyตัวเลขโดยใช้ random.randint() และ if-else

13. Flowchart ช่วยอะไรในการเขียนโปรแกรม และสัญลักษณ์สำคัญใน Flowchart มีอะไรบ้าง?
14. อธิบายขั้นตอนการเขียนโปรแกรมสูตรคูณ และประโยชน์ที่ได้จากโปรแกรมนี้
15. หากต้องการแสดงผลตัวเลขในลักษณะตาราง จะใช้โครงสร้างโปรแกรมแบบใด?
16. ให้ตัวอย่างโปรแกรมที่ใช้ลูปซ้อน (Nested Loop) เพื่อสร้างรูปสี่เหลี่ยม
17. หากต้องการสร้างโปรแกรมที่ตรวจสอบค่าจากผู้ใช้ เช่น ชื่อหรือรหัสผ่าน จะใช้โครงสร้างโปรแกรมแบบใด?
18. Boolean Operators เช่น and, or, และ not ทำงานอย่างไรในโปรแกรมเงื่อนไข?
19. หากผู้ใช้ต้องการสร้างโปรแกรมที่ทำขึ้นกว่าจะเจอเงื่อนไขที่ต้องการ ควรใช้ while loop หรือ for loop?
20. จากเนื้อหาในบทนี้ ผู้เรียนคิดว่าโครงสร้างการควบคุมใดเหมาะสมที่สุดสำหรับการเขียนเกม และ เพราะเหตุใด?

โจทย์การเขียนโปรแกรม

- เขียนโปรแกรม เพื่อรับตัวเลขจากผู้ใช้ และตรวจสอบว่าตัวเลขนั้นเป็นเลขคู่หรือเลขคี่

```
Enter a number: 7  
7 is odd.
```

- เขียนโปรแกรม คำนวณวันในอนาคต โดยให้ผู้ใช้ป้อนจำนวนวันที่ต้องการคำนวณจากวันที่ปัจจุบัน

```
Enter the number of days: 5  
Today is Monday.  
The day in 5 days will be Saturday.
```

- เขียนโปรแกรม เพื่อค้นหาตัวเลขที่มากที่สุดในชุดตัวเลข 5 ตัวที่ผู้ใช้ป้อนเข้ามา

```
Enter number 1: 10  
Enter number 2: 25  
Enter number 3: 5  
Enter number 4: 30  
Enter number 5: 20  
The largest number is: 30
```

- เขียนโปรแกรม สร้างรูปแบบพิрамิดตามจำนวนชั้นที่ผู้ใช้กำหนด

```
Enter the number of levels: 4  
#  
###  
#####  
#####
```

- เขียนโปรแกรม เกมทายตัวเลขที่มีช่วงตั้งแต่ 1 ถึง 100 โดยให้ผู้เล่นทายไม่เกิน 7 ครั้ง

```
I am thinking of a number between 1 and 100.  
Take a guess: 50  
Your guess is too low.
```

Take a guess: 75

Your guess is too high.

Take a guess: 63

Good job! You guessed my number in 3 guesses!

- เขียนโปรแกรม คำนวณเกรดจากคะแนนที่ผู้ใช้ป้อนเข้าไป และแสดงข้อความแนะนำ (A, B, C, D, F)

Enter your score: 85

Your grade is: A

- เขียนโปรแกรม ตรวจสอบว่าข้อความที่ผู้ใช้ป้อนเข้าไปเป็นพอลินโตร姆 (palindrome) หรือไม่

Enter a word: radar

radar is a palindrome.

- เขียนโปรแกรม รับชื่อของผู้ใช้แล้วตรวจสอบว่าเป็นชื่อที่กำหนดไว้ล่วงหน้าหรือไม่

Enter your name: Alice

Hello, Alice! Welcome back.

- เขียนโปรแกรม หาค่า Factorial ของตัวเลขที่ผู้ใช้ป้อน

Enter a number: 5

The factorial of 5 is 120.

- เขียนโปรแกรม สร้างตารางสูตรคูณของแม่ที่ผู้ใช้กำหนด (เช่นแม่ 3)

Enter a number: 3

$3 \times 1 = 3$

$3 \times 2 = 6$

...

$3 \times 12 = 36$

- เขียนโปรแกรม รับข้อมูลอายุของบุคคล และแสดงว่าเขามีความสามารถลงคะแนนเสียงเลือกตั้งได้หรือไม่

Enter your age: 17

You are not eligible to vote.

- เขียนโปรแกรม สร้างสามเหลี่ยมกลับหัวโดยให้ผู้ใช้กำหนดจำนวนแถว

#####

###

##

#

- เขียนโปรแกรม รับคำจากผู้ใช้และตรวจสอบว่าคำนั้นเป็นคำที่ซ้ำกัน (Duplicate) ในลิสต์หรือไม่

Enter a word: apple

The word 'apple' is not in the list. Adding it now.

- เขียนโปรแกรม แปลงค่าอุณหภูมิจากเซลเซียสเป็นฟาร์นไฮต์

Enter temperature in Celsius: 25

Temperature in Fahrenheit is: 77.0

- เขียนโปรแกรม สร้างรูปแบบกรอบข้อความตามความยาวของข้อความที่ผู้ใช้ป้อน

Enter a message: Hello

+-----+

| Hello |

+-----+

- เขียนโปรแกรม รับตัวเลข 3 ตัว และบอกว่ามีตัวเลขใดบ้างที่เป็นเลขเฉพาะ (Prime Number)

Enter three numbers: 3 4 5

Prime numbers: 3, 5

17. เขียนโปรแกรม ตรวจสอบว่าปีที่ผู้ใช้ป้อนเป็นปีอัลกสูรทิน (Leap Year) หรือไม่

Enter a year: 2024

2024 is a leap year.

18. เขียนโปรแกรม เกมจำลองล็อกอิน โดยผู้ใช้ต้องป้อนชื่อผู้ใช้และรหัสผ่านให้ถูกต้อง

Enter username: admin

Enter password: 1234

Login successful. Welcome!

19. เขียนโปรแกรม แปลงตัวเลขที่ผู้ใช้ป้อนจากเลขฐานสิบไปเป็นเลขฐานสอง

Enter a decimal number: 10

Binary representation: 1010

20. เขียนโปรแกรม เพื่อรับจำนวนรายการในลิสต์ และหาผลรวมและค่าเฉลี่ยของตัวเลขทั้งหมด

Enter numbers separated by space: 10 20 30

Sum = 60

Average = 20.0

บทที่ 4

ฟังก์ชัน

เนื้อหาบทเรียน

- ความหมายและบทบาทของฟังก์ชัน
- การใช้งานฟังก์ชันเบื้องต้น ได้แก่ การสร้างและเรียกใช้ฟังก์ชัน (Function Creation and Call)
- การส่งค่าและรับค่าผลลัพธ์ (Parameters and Return Values)
- การใช้ฟังก์ชันแก้ไขปัญหา
- การสร้างโมดูล (Create Module)

วัตถุประสงค์การเรียนรู้

- เข้าใจความหมายของฟังก์ชัน (Function) และบทบาทในการพัฒนาโปรแกรม
- เขียนโปรแกรมสร้างและเรียกใช้ฟังก์ชัน
- เขียนโปรแกรมส่งค่า (Parameters) และรับค่าผลลัพธ์ (Return Values) จากฟังก์ชัน
- เขียนฟังก์ชันแก้ไขปัญหาได้อย่างมีประสิทธิภาพ
- เขียนโปรแกรมเพื่อสร้างและใช้งานแบบโมดูลได้

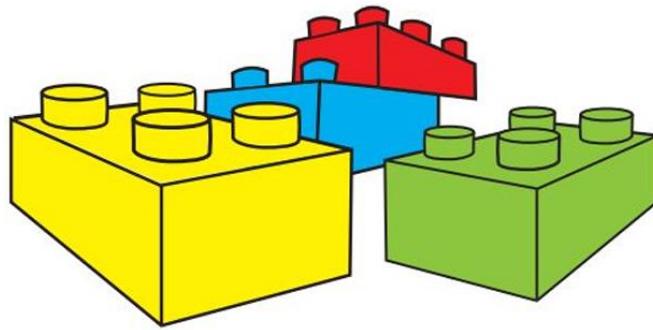
การเขียนโปรแกรมในปัจจุบันมีความสำคัญอย่างยิ่งในการพัฒนาซอฟต์แวร์ที่ซับซ้อน พังก์ชัน (Function) เป็นเครื่องมือสำคัญที่ช่วยให้โปรแกรมมีโครงสร้างที่ชัดเจน และลดความซับซ้อนในการพัฒนา พังก์ชันช่วยให้การทำงานของโปรแกรมถูกแบ่งออกเป็นส่วนย่อย ๆ ที่สามารถนำกลับมาใช้ซ้ำ (Reusability) ได้ ทำให้โค้ดมีความยืดหยุ่นและง่ายต่อการดูแลรักษา นอกจากนี้ พังก์ชันยังช่วยสนับสนุนการทำงานร่วมกันในทีมพัฒนา สามารถแบ่งภาระงานที่ต้องดูแลรักษาให้กับผู้พัฒนาคนอื่นได้โดยไม่รบกวนงานส่วนอื่นของโปรแกรม แนวคิดของการสร้างพังก์ชันย่ออย่างช่วยส่งเสริมกระบวนการออกแบบแบบโปรแกรมที่เป็นระบบ เช่น การเขียนโปรแกรมเชิงโมดูลาร์ (Modular Programming) ที่ช่วยลดปัญหาโค้ดซ้ำซ้อนและเพิ่มประสิทธิภาพในการพัฒนา

เมื่อโครงสร้างการพัฒนาซอฟต์แวร์มีความซับซ้อน การจัดการโค้ดให้อยู่ในรูปแบบโมดูล (Module) ยิ่งมีความสำคัญมากขึ้น โมดูลช่วยรวบรวมฟังก์ชันหรือคลาสที่เกี่ยวข้องกันไว้ในไฟล์เดียว ทำให้โปรแกรมมีความเป็นระเบียบ และสามารถนำโมดูลไปใช้ซ้ำในโครงการอื่นได้อย่างง่ายดาย การสร้างและใช้งานฟังก์ชันร่วมกับโมดูลยังช่วยเพิ่มความสามารถในการออกแบบโปรแกรมให้รองรับการเปลี่ยนแปลงในอนาคต ไม่ว่าจะเป็นการเพิ่มฟีเจอร์ใหม่ การแก้ไขข้อผิดพลาด หรือการปรับปรุงประสิทธิภาพ

ในบทเรียนนี้ จะสำรวจแนวคิดพื้นฐานของฟังก์ชัน ตั้งแต่การสร้างฟังก์ชัน การใช้งาน ไปจนถึงการสร้างโมดูลที่ช่วยให้การพัฒนาโปรแกรมเป็นไปอย่างมีประสิทธิภาพ พร้อมตัวอย่างที่เข้าใจง่ายและสามารถนำไปประยุกต์ใช้ในงานจริงได้ หัวข้อนี้จะช่วยให้สามารถพัฒนาทักษะการเขียนโปรแกรม และออกแบบโครงสร้างโปรแกรมที่มีความยืดหยุ่นและตอบโจทย์การใช้งานได้อย่างมืออาชีพ

4.1 ความหมายและบทบาทของฟังก์ชัน

ฟังก์ชัน (Function) หมายถึงกลุ่มของคำสั่งหรือโค้ดที่ถูกเขียนขึ้นมาให้ทำงานเฉพาะเจาะจง (คล้ายกับ LEGO) ดังรูปที่ 4.1 คล้ายกับ “หน่วยย่อย” ของโปรแกรม (Building Block) ซึ่งสามารถเรียกใช้ซ้ำได้หลายครั้งโดยไม่ต้องเขียนโค้ดเดิมซ้ำ การเขียนฟังก์ชันในลักษณะโมดูลยังช่วยให้การตั้งชื่อกลุ่มโค้ดเป็นไปอย่างชัดเจนและง่ายต่อการเรียกใช้งาน นอกจากนี้ การจัดกลุ่มคำสั่งในฟังก์ชันยังช่วยให้การพัฒนาโปรแกรมเป็นไปอย่างมีระบบระเบียบ และสามารถเพิ่มหรือปรับปรุงความสามารถของโปรแกรมในภายหลังได้โดยสะดวก ฟังก์ชันที่ถูกจัดให้อยู่ในโมดูลสามารถแยกออกจากเป็นไฟล์ย่อย ทำให้การเรียกใช้งานในโปรแกรมอื่น ๆ สะดวกขึ้น อีกทั้งยังส่งเสริมความสามารถในการนำโมดูลกลับมาใช้ใหม่ในโปรเจกต์อื่น ๆ ได้อย่างมีประสิทธิภาพ ฟังก์ชันช่วยให้โปรแกรมมีโครงสร้างที่ชัดเจน และสามารถดูแล ปรับปรุง หรือเพิ่มเติมความสามารถใหม่ ๆ ได้ง่ายขึ้น



รูปที่ 4.1 "หน่วยอยู่" ที่สามารถประกอบเข้าด้วยกัน ตามแนวคิดการเขียนโปรแกรมแบบโมดูลาร์

ที่มา Sande, W., & Sande, C. (2020). Hello World

ตัวอย่างเช่น ในทางคณิตศาสตร์ ถ้ากำหนดฟังก์ชัน $f(x, y) = 3x - 2y + 5$ เมื่อส่งค่าของ x และ y เข้าไปในฟังก์ชัน f จะได้ผลลัพธ์ (Output) เป็นค่าตัวเลขหนึ่งค่าเสมอ ในการเขียนโปรแกรมก็คล้ายกัน สามารถกำหนดฟังก์ชันขึ้นมาเพื่อกำหนดค่าหรือทำงานบางอย่าง แล้วเรียกใช้ได้ตามต้องการ นอกจากนี้ ในชีวิตประจำวัน ฟังก์ชันสามารถเปรียบได้กับการทำอาหารตามสูตร โดยในสูตรอาหารจะมีการกำหนดส่วนผสม (Parameters) และขั้นตอนการทำงาน (Processes) ที่ชัดเจน เช่น การทำเค้ก สูตรอาจจะระบุว่าต้องใช้แป้ง 2 ถ้วย ไข่ 3 พอง น้ำตาล 1 ถ้วย และนม 1 ถ้วย (Input) จากนั้นจึงนำส่วนผสมทั้งหมดมาผัดกันในลำดับที่กำหนด และอบในเตาอบที่อุณหภูมิ 180 องศาเซลเซียสเป็นเวลา 30 นาที (Process) ผลลัพธ์ที่ได้คือเค้ก 1 ก้อน (Output)

Pseudocode ตัวอย่าง

```
function makeCake(flour, eggs, sugar, milk):
    mixIngredients(flour, eggs, sugar, milk)
    bake(temperature=180, time=30)
    return "Cake"

result = makeCake(2, 3, 1, 1)
print(result) # Output: Cake
```

ฟังก์ชันเป็นเครื่องมือสำคัญในการเขียนโปรแกรมที่ช่วยให้การจัดระเบียบโค้ดเป็นไปอย่างมีประสิทธิภาพ โดยฟังก์ชันทำหน้าที่รวบรวมคำสั่งที่มีเป้าหมายเฉพาะไว้ด้วยกัน ทำให้ผู้เขียนโปรแกรมสามารถแยกแยะส่วนต่าง ๆ ของโค้ดออกจากกันได้อย่างเป็นระบบและชัดเจน นอกจากนี้ ฟังก์ชันยังมีคุณสมบัติที่โดดเด่นในด้านการทำงานซ้ำ (Reusability) ซึ่งช่วยลดความยุ่งยากในการเขียนคำสั่งเดิมซ้ำหลายครั้ง ผู้เขียนโปรแกรมสามารถซ่อนรายละเอียดของการทำงานภายใต้ฟังก์ชัน และเรียกใช้งานได้เมื่อจำเป็น

ด้วยความสามารถเหล่านี้ ฟังก์ชันจึงมีบทบาทสำคัญในการลดความซับซ้อนของโปรแกรมขนาดใหญ่ ซึ่งมักเต็มไปด้วยโครงสร้างและกระบวนการที่ซับซ้อน นอกจากนี้ ฟังก์ชันยังช่วยให้การบำรุงรักษา (Maintenance) เป็นเรื่องง่ายขึ้น การแก้ไขในฟังก์ชันเพียงจุดเดียวสามารถส่งผลไปยังส่วนอื่น ๆ ของโปรแกรมได้ทันที

ประโยชน์ของการเขียนฟังก์ชัน

1. การปรับปรุงความปลอดภัยของโค้ด ฟังก์ชันช่วยให้สามารถจำกัดการเข้าถึงหรือแก้ไขข้อมูลในส่วนต่าง ๆ ของโปรแกรมได้ผ่านการกำหนดขอบเขตของฟังก์ชัน
2. เพิ่มประสิทธิภาพการทำงาน ฟังก์ชันช่วยแบ่งงานออกเป็นส่วนเล็ก ๆ ที่สามารถทดสอบหรือปรับปรุงได้อย่างง่ายดาย
3. สนับสนุนการทำงานเป็นทีม ด้วยโครงสร้างที่แยกส่วนชัดเจน สมาชิกทีมสามารถพัฒนาฟังก์ชันของตนเองได้โดยไม่ส่งผลกระทบต่อส่วนอื่น
4. เพิ่มความยืดหยุ่นในการออกแบบโปรแกรม ฟังก์ชันช่วยให้การออกแบบโปรแกรมมีความยืดหยุ่น และสามารถปรับเปลี่ยนได้ง่าย

ฟังก์ชันเปรียบเสมือน "เครื่องมือจัดระเบียบ" ที่ช่วยให้โค้ดมีโครงสร้างที่ชัดเจนขึ้น การแบ่งโปรแกรมออกเป็นฟังก์ชันช่วยให้การทำงานเป็นทีมมีประสิทธิภาพมากขึ้น ตัวอย่างเช่น ในโปรเจกต์หนึ่ง คนหนึ่งอาจรับผิดชอบฟังก์ชันคำนวณ ส่วนอีกคนดูแลฟังก์ชันแสดงผล ซึ่งจะช่วยลดความเสี่ยงจากการแก้ไขโค้ดผิดส่วน และยังทำให้โค้ดสามารถนำกลับมาใช้ซ้ำได้ในอนาคต

4.2 การใช้งานฟังก์ชันเบื้องต้น

ฟังก์ชัน (Function) เป็นเครื่องมือสำคัญในการเขียนโปรแกรมที่ช่วยลดความซ้ำซ้อนของโค้ด เพิ่มความยืดหยุ่น และทำให้โค้ดอ่านง่ายขึ้น การใช้งานฟังก์ชันเบื้องต้นประกอบด้วยการสร้างและการเรียกใช้งานฟังก์ชัน พร้อมตัวอย่างการประยุกต์ใช้ในโปรแกรมเบื้องต้น

4.2.1 การสร้างฟังก์ชันแบบพื้นฐาน

```
def function_name():
    # คำสั่งที่ต้องการให้ทำงาน Indentation
    การสร้างฟังก์ชันใน Python มีส่วนประกอบพื้นฐานเป็น แบบฟังก์ชันไม่มีพารามิเตอร์ ที่สำคัญ ดังนี้
    1. def คำสั่ง def เป็นจุดเริ่มต้นของการนิยามฟังก์ชันใน Python โดยบอกให้โปรแกรมทราบว่า
        กำลังสร้างฟังก์ชันใหม่ ตัวคำสั่งนี้จำเป็นต้องอยู่ที่บรรทัดแรกของการสร้างฟังก์ชัน
    2. function_name ชื่อฟังก์ชัน (function_name) เป็นตัวกำหนดเอกลักษณ์ของฟังก์ชันที่สร้างขึ้น
        ชื่อฟังก์ชันควรเป็นคำที่สื่อความหมายถึงสิ่งที่ฟังก์ชันทำ และควรเขียนในรูปแบบตัวพิมพ์เล็กโดยใช้
        เครื่องหมายขีดล่าง _ หากมีหลายคำ เช่น calculate_sum หรือ display_menu เพื่อให้สอดคล้องกับ
        มาตรฐานการเขียนโปรแกรม
```

3. : เครื่องหมายโคลอน (:) ที่ต่อท้ายชื่อฟังก์ชัน ใช้เพื่อบอก Python ว่ามีคำสั่งที่ต้องการให้ทำงานอยู่ในบล็อกถัดไปของฟังก์ชัน

4. คำสั่งในฟังก์ชัน คำสั่งในฟังก์ชันคือโค้ดที่ต้องการให้ทำงานเมื่อฟังก์ชันถูกเรียกใช้งาน คำสั่งทั้งหมดต้องถูกเย็บบรรทัด (Indentation) โดยใช้ 4 ช่องว่างตามมาตรฐาน Python เพื่อแสดงว่าโค้ดเหล่านี้เป็นส่วนหนึ่งของฟังก์ชัน

ตัวอย่างการสร้างฟังก์ชัน

```
def greet():
    print("Hello, world!")
```

def ใช้เริ่มต้นนิยามฟังก์ชัน greet เป็นชื่อของฟังก์ชันที่กำหนดขึ้น : ระบุว่าฟังก์ชันมีคำสั่งในบล็อกถัดไป คำสั่ง print("Hello, world!") เป็นโค้ดที่ต้องการให้ทำงานเมื่อเรียกใช้ฟังก์ชัน

4.2.2 การเรียกใช้ฟังก์ชัน

การเรียกใช้งานฟังก์ชันใน Python หมายถึงการสั่งให้ฟังก์ชันที่สร้างไว้ทำงาน โดยใช้ชื่อฟังก์ชันตามด้วยวงเล็บ () หากฟังก์ชันมีพารามิเตอร์ จะต้องส่งค่าที่จำเป็นเข้าไปในวงเล็บ การเรียกใช้งานฟังก์ชันช่วยให้โค้ดทำงานตามที่ได้กำหนดไว้ในฟังก์ชัน วิธีการเรียกใช้ฟังก์ชัน ดังนี้

- ใช้ชื่อฟังก์ชันที่นิยามไว้ ตามด้วยเครื่องหมายวงเล็บ ()
- หากฟังก์ชันมีพารามิเตอร์ ต้องส่งค่าที่ต้องการเข้าไปในวงเล็บ โดยจะกล่าวในหัวข้อที่ 4.3 เพิ่มเติม
- ฟังก์ชันจะดำเนินการคำสั่งภายใน และคืนค่าผลลัพธ์ (ถ้ามี) กลับมา โดยจะกล่าวในหัวข้อที่ 4.3 เพิ่มเติม

```
def greet():
    print("Hello, world!")

# เรียกใช้งานฟังก์ชัน
greet() # Output: Hello, world!
ผลลัพธ์
Hello, world.
```

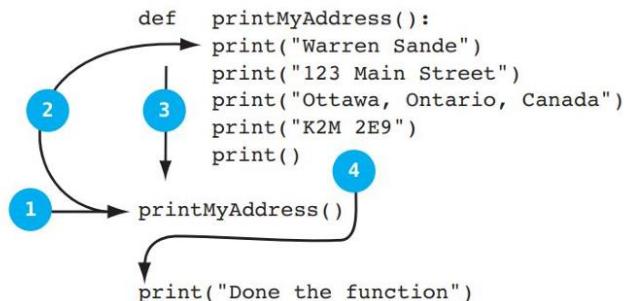
4.2.3 ลำดับการรับของโค้ดฟังก์ชัน

ตัวอย่างฟังก์ชันพิมพ์ที่อยู่ การสร้างฟังก์ชัน printMyAddress() โดยใช้คำสั่ง def เพื่อกำหนดคำสั่งที่ต้องการให้ทำงาน ดังนี้

```
def printMyAddress():
    print("Warren Sande")
    print("123 Main Street")
    print("Ottawa, Ontario, Canada")
    print("K2M 2E9")
    print()
```

- def ใช้เพื่อบอก Python ว่านี่คือการสร้างฟังก์ชัน
- print() ใช้แสดงข้อความ เช่น ชื่อ ที่อยู่ และข้อมูลอื่น ๆ
- ชื่อฟังก์ชัน printMyAddress ใช้เพื่อเรียกใช้งานในภายหลัง ลำดับการเรียกใช้จากรูปที่ 4.2 แสดงขั้นตอนการทำงาน ดังนี้
 - เริ่มนั่งโปรแกรมและเจอฟังก์ชัน printMyAddress()

2. Python จะไปบังคับสั่ง def และเริ่มทำงานตามคำสั่งในฟังก์ชัน (พิมพ์ทีอยู่)
3. เมื่อทำงานเสร็จ ฟังก์ชันจะคืนการควบคุมกลับไปยังส่วนที่เรียกใช้งาน (เรียกจบแล้ว กลับมา)
4. โปรแกรมแสดงข้อความเพิ่มเติมหลังจากการเรียกฟังก์ชัน print("Done the function")



รูปที่ 4.2 แสดงการสร้าง การเรียกใช้ฟังก์ชัน และลำดับการทำงาน

ที่มา Sande, W., & Sande, C. (2020). Hello World

ตัวอย่างเพิ่มเติมแสดงลำดับการเรียกใช้ได้ดังตัวอย่างนี้

```

def function_a():
    print("function_a starts")
    function_b() # เรียก function_b
    function_d() # เรียก function_d
    print("function_a ends")

def function_b():
    print("function_b starts")
    function_c() # เรียก function_c
    print("function_b ends")

def function_c():
    print("function_c starts")
    print("function_c ends")

def function_d():
    print("function_d starts")
    print("function_d ends")

```

เรียกใช้ฟังก์ชันหลัก

function_a()

ผลลัพธ์

```

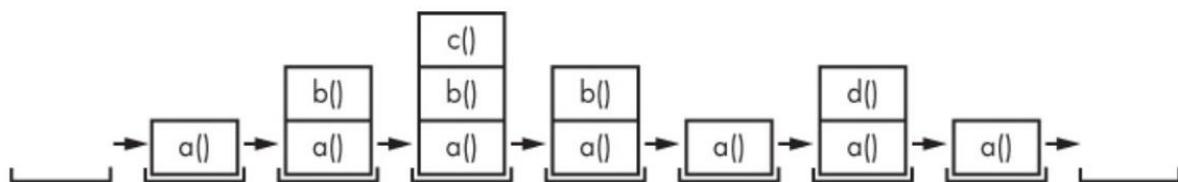
function_a starts
function_b starts
function_c starts
function_c ends
function_b ends

```

```
function_d starts
function_d ends
function_a ends
```

อธิบายการทำงานของ Call Stack ตามรูปที่ 4.3

- function_a ถูกเรียกและเริ่มต้นการทำงาน: Call Stack: function_a
- function_a เรียก function_b: Call Stack: function_a -> function_b
- function_b เรียก function_c: Call Stack: function_a -> function_b -> function_c
- เมื่อ function_c จบการทำงาน (returns): Call Stack: function_a -> function_b
- เมื่อ function_b จบการทำงาน (returns): Call Stack: function_a
- function_a เรียก function_d: Call Stack: function_a -> function_d
- เมื่อ function_d จบการทำงาน (returns): Call Stack: function_a
- เมื่อ function_a จบการทำงาน (returns): Call Stack: empty



รูปที่ 4.3 แสดงการเรียกใช้งานแบบ call stack

ที่มา Sweigart, A. (2015). Automate the Boring Stuff with Python: Practical Programming for Total Beginners

4.2.4 การลดความซ้ำซ้อนของโค้ด

โค้ดเริ่มต้นด้านล่างมีการพิมพ์ข้อมูลซ้ำ ๆ ซึ่งทำให้เกิดความยุ่งยากในการจัดการและแก้ไข หากมีการเปลี่ยนแปลงข้อมูล เช่น ชื่อ อายุ หรือข้อความใด ๆ จำเป็นต้องแก้ไขในทุกจุดที่โค้ดถูกเขียนซ้ำ

```
print("Hello -----")
print("My name is Sarayut Gonwirat")
print("I'm 20 years old.")
print("My favorite color is Green.")
print()
print("Can you introduce yourself? -----")
print("My name is Sarayut Gonwirat")
print("I'm 20 years old.")
print("My favorite color is Green.")
print()
print("Again, can you introduce yourself? -----")
print("My name is Sarayut Gonwirat")
print("I'm 20 years old.")
print("My favorite color is Green.")
print()
```

ผลลัพธ์

```
Hello -----
My name is Sarayut Gonwirat
I'm 20 years old.
My favorite color is Green.

Can you introduce yourself? -----
My name is Sarayut Gonwirat
I'm 20 years old.
My favorite color is Green.

Again, can you introduce yourself? -----
My name is Sarayut Gonwirat
I'm 20 years old.
My favorite color is Green.
```

โค้ดหลังลดความซ้ำซ้อนโดยใช้ฟังก์ชัน โดยมีขั้นตอนการปรับปรุง ดังนี้

- สร้างฟังก์ชัน `introduce` สร้างฟังก์ชันที่รวบรวมข้อมูลที่ต้องการพิมพ์มาในหลาย ๆ จุด ฟังก์ชันนี้สามารถเรียกใช้ได้ทุกครั้งที่ต้องการพิมพ์ข้อมูลดังกล่าว
- การเรียกใช้งานฟังก์ชัน (`introduce()`) เรียกใช้ฟังก์ชันแทนการเขียนข้อความซ้ำ ๆ ทำให้โค้ดกระชับและอ่านง่ายขึ้น
- ประโยชน์ที่ได้รับ ลดความซ้ำซ้อน หากต้องการเปลี่ยนแปลงข้อมูล เช่น ชื่อหรืออายุ สามารถแก้ไขที่ฟังก์ชันเพียงครั้งเดียว และเพิ่มความชัดเจน โค้ดอ่านง่ายขึ้นและแยกส่วนการทำงานชัดเจน
- ปรับปรุงง่าย หากต้องการเปลี่ยนรูปแบบข้อความ เช่น เพิ่มข้อความใหม่ สามารถแก้ไขได้ที่ฟังก์ชันโดยไม่ต้องแก้ไขในทุกจุด

```
def introduce():
    print("My name is Sarayut Gonwirat")
    print("I'm 20 years old.")
    print("My favorite color is Green.")
    print()

print("Hello -----")
introduce()
print("Can you introduce yourself? -----")
introduce()
print("Again, can you introduce yourself? -----")
introduce()
```

การสร้างและเรียกใช้ฟังก์ชันเป็นทักษะพื้นฐานที่สำคัญในการเขียนโปรแกรม เพราะช่วยลดความซ้ำซ้อนของโค้ด เช่น ไม่ต้องเขียนโค้ดเดิมซ้ำหลายจุดในโปรแกรม เมื่อเจอบัญหาใหม่ ก็สามารถเรียกใช้ฟังก์ชันเดิมได้ทันที โดยไม่ต้องเขียนโค้ดใหม่ทั้งหมด ซึ่งช่วยลดเวลาและลดโอกาสเกิดข้อผิดพลาด

4.3 การส่งค่าและรับค่าผลลัพธ์

การส่งค่าและรับค่าผลลัพธ์ในฟังก์ชันเป็นอีกหนึ่งกระบวนการสำคัญที่ช่วยให้ฟังก์ชันสามารถประมวลผลข้อมูลที่ส่งเข้ามาและส่งผลลัพธ์กลับออกไปยังส่วนอื่นของโปรแกรมได้ การทำงานในลักษณะนี้ช่วยเพิ่มความยืดหยุ่นและประสิทธิภาพในการออกแบบโปรแกรม

4.3.1 การส่งค่า (Parameters)

พารามิเตอร์ (Parameters) คือ ตัวแปรที่กำหนดในวงเล็บหลังชื่อฟังก์ชัน เพื่อรับข้อมูลที่ส่งเข้ามาจากภายนอกเมื่อเรียกใช้งานฟังก์ชัน พารามิเตอร์ช่วยให้ฟังก์ชันสามารถทำงานกับข้อมูลที่แตกต่างกันในแต่ละครั้งที่เรียกใช้

```
def function_name(parameter1, parameter2):
```

```
# คำสั่งที่ต้องการให้ทำงาน
```

1. การส่งพารามิเตอร์ 1 ค่า

```
def greet_user(name):
    print(f"Hello, {name}!")
```

```
# การเรียกใช้งานฟังก์ชัน
```

```
greet_user("Alice") # Output: Hello, Alice!
greet_user("Bob")  # Output: Hello, Bob!
```

ผลลัพธ์

Hello, Alice!

Hello, Bob!

- ฟังก์ชันชื่อ `greet_user` ถูกสร้างขึ้นเพื่อทำงานพิมพ์ข้อความทักทาย พารามิเตอร์ `name` เป็นตัวแปรที่รับค่าข้อมูลจากผู้เรียกใช้งานฟังก์ชัน ข้อความ `Hello, {name}!` ใช้ฟอร์แมต (f-string) เพื่อแทนที่ `{name}` ด้วยค่าที่ส่งเข้ามาในพารามิเตอร์ `name`
- `greet_user("Alice")` ฟังก์ชันจะพิมพ์ข้อความ `Hello, Alice!` เนื่องจากค่า `Alice` ถูกส่งเข้าไปแทนพารามิเตอร์ `name`

2. การส่งพารามิเตอร์ 2 ค่า

```
def add_numbers(a, b):
    print(f"ผลรวมของ {a} และ {b} คือ {a + b}")
```

```
# การเรียกใช้งานฟังก์ชัน
```

```
add_numbers(5, 10) # Output: ผลรวมของ 5 และ 10 คือ 15
add_numbers(2, 3)  # Output: ผลรวมของ 2 และ 3 คือ 5
```

ผลลัพธ์

ผลรวมของ 5 และ 10 คือ 15

ผลรวมของ 2 และ 3 คือ 5

- ฟังก์ชันชื่อ `add_numbers` ถูกสร้างขึ้นเพื่อทำงานคำนวณผลรวมของตัวเลขสองตัวที่รับเข้ามา และพิมพ์ผลลัพธ์ออกมา

- พารามิเตอร์ a และ b เป็นตัวแปรที่รับค่าข้อมูลจากผู้เรียกใช้งานฟังก์ชัน ข้อความ ผลรวมของ {a} และ {b} คือ {a + b} ใช้ฟอร์แมต (f-string) เพื่อแทนที่ {a} และ {b} ด้วยค่าที่ส่งเข้ามาในพารามิเตอร์
 - add_numbers(5, 10) ฟังก์ชันจะพิมพ์ข้อความ ผลรวมของ 5 และ 10 คือ 15 เนื่องจากค่าของ a คือ 5 และ b คือ 10
 - add_numbers(2, 3) ฟังก์ชันจะพิมพ์ข้อความ ผลรวมของ 2 และ 3 คือ 5 เนื่องจากค่าของ a คือ 2 และ b คือ 3
3. การส่งพารามิเตอร์หลายค่า (ใช้ *args)

```
def calculate_sum(*numbers):
    total = sum(numbers)
    print(f"ผลรวมของตัวเลขทั้งหมดคือ {total}")

# การเรียกใช้งานฟังก์ชัน
calculate_sum(1, 2, 3, 4, 5) # Output: ผลรวมของตัวเลขทั้งหมดคือ 15
calculate_sum(10, 20, 30)     # Output: ผลรวมของตัวเลขทั้งหมดคือ 60
ผลลัพธ์
```

ผลรวมของตัวเลขทั้งหมดคือ 15

ผลรวมของตัวเลขทั้งหมดคือ 60

- ฟังก์ชันชื่อ calculate_sum ถูกสร้างขึ้นเพื่อกำหนณผลรวมของตัวเลขจำนวนไม่จำกัดที่ส่งเข้ามา โดยใช้ *numbers ซึ่งเป็นการรับพารามิเตอร์แบบหลายค่าในรูปแบบของ tuple
- คำสั่ง sum(numbers) ใช้สำหรับคำนวนผลรวมของค่าทั้งหมดที่อยู่ใน numbers และพิมพ์ผลลัพธ์ออกมายังรูปแบบ f-string
- การเรียกใช้งานฟังก์ชัน
 - calculate_sum(1, 2, 3, 4, 5) ค่าที่ส่งเข้ามา: 1, 2, 3, 4, 5 ฟังก์ชันคำนวนผลรวม: $1 + 2 + 3 + 4 + 5 = 15$ ผลลัพธ์ที่ได้: ผลรวมของตัวเลขทั้งหมดคือ 15
 - calculate_sum(10, 20, 30) ค่าที่ส่งเข้ามา: 10, 20, 30 ฟังก์ชันคำนวนผลรวม: $10 + 20 + 30 = 60$ ผลลัพธ์ที่ได้: ผลรวมของตัวเลขทั้งหมดคือ 60
- ข้อดีของการใช้ *args สามารถส่งพารามิเตอร์จำนวนใด ๆ ก็ได้ ทำให้ฟังก์ชันมีความยืดหยุ่นสูง และเหมาะสมสำหรับสถานการณ์ที่จำนวนตัวเลขหรือข้อมูลที่ต้องการประมวลผลไม่แน่นอน

4.3.2 การรับค่าผลลัพธ์ (Return Values)

การรับค่าผลลัพธ์หมายถึงการที่ฟังก์ชันส่งค่ากลับมายังส่วนที่เรียกใช้งานฟังก์ชัน โดยใช้คำสั่ง return คำสั่งนี้ช่วยให้โปรแกรมสามารถนำค่าที่ได้จากการฟังก์ชันไปใช้งานในส่วนอื่น ๆ ได้

```
def function_name():
    # คำสั่งที่ต้องการให้ทำงาน
    return value
```

1. การส่งพารามิเตอร์ 1 ค่า

```
def add_numbers(a, b):
    return a + b

result = add_numbers(5, 7)
print(result) # Output: 12
```

ผลลัพธ์

12

- ฟังก์ชันชื่อ add_numbers ถูกสร้างขึ้นเพื่อคำนวณผลรวมของตัวเลขสองตัวที่ส่งเข้ามาผ่านพารามิเตอร์ a และ b
- คำสั่ง return a + b ส่งค่าผลลัพธ์ของการบวกตัวเลขทั้งสองกลับไปยังส่วนที่เรียกใช้งานฟังก์ชัน
- เมื่อเรียกใช้งานฟังก์ชัน add_numbers(5, 7) ค่าที่ส่งกลับคือ 12 และถูกเก็บในตัวแปร result
- ข้อมูลใน result สามารถนำไปใช้งานต่อ เช่น แสดงผลด้วยคำสั่ง print(result)

2. การส่งพารามิเตอร์ 2 ค่า

```
def calculate_values(x, y):
    sum_result = x + y
    product_result = x * y
    return sum_result, product_result

# การเรียกใช้งานฟังก์ชัน
sum_val, product_val = calculate_values(3, 4)
print(f"ผลรวม: {sum_val}, ผลคูณ: {product_val}")
```

ผลลัพธ์

ผลรวม: 7, ผลคูณ: 12

- ฟังก์ชัน calculate_values คำนวณผลรวมและผลคูณของตัวเลขสองตัวที่ส่งเข้ามาในพารามิเตอร์ x และ y
- ใช้คำสั่ง return sum_result, product_result เพื่อส่งค่าผลลัพธ์สองค่าออกจากฟังก์ชัน
- ค่าที่คืนกลับสามารถแยกเก็บในตัวแปรสองตัว (sum_val และ product_val) เพื่อใช้งานต่อ

3. การคืนค่าผลลัพธ์ที่คำนวณจากลิสต์

```
def find_max_and_min(numbers):
    max_val = max(numbers)
    min_val = min(numbers)
    return max_val, min_val

# การเรียกใช้งานฟังก์ชัน
max_num, min_num = find_max_and_min([10, 20, 30, 40, 50])
print(f"ค่าสูงสุด: {max_num}, ค่าต่ำสุด: {min_num}")
```

ผลลัพธ์

ค่าสูงสุด: 50, ค่าต่ำสุด: 10

- พังก์ชัน `find_max_and_min` รับพารามิเตอร์เป็นลิสต์ของตัวเลข
- ใช้ฟังก์ชัน `max` และ `min` เพื่อหาค่าสูงสุดและค่าต่ำสุดของลิสต์นั้น
- ค่าผลลัพธ์สองค่าถูกส่งกลับผ่านคำสั่ง `return max_val, min_val`
- ค่าที่คืนกลับสามารถเก็บในตัวแปรสองตัวเพื่อใช้งานต่อ เช่น การพิมพ์ค่าสูงสุดและต่ำสุด

4.3.3 ตัวแปรระดับโปรแกรมและภายนอกฟังก์ชัน

ตัวแปรระดับโปรแกรม (Global Variable) และ ตัวแปรภายในฟังก์ชัน (Local Variable) เป็นแนวคิดพื้นฐานที่เกี่ยวข้องกับการจัดการขอบเขต (Scope) ของตัวแปรในโปรแกรม การทำความเข้าใจความแตกต่างระหว่างตัวแปรทั้งสองชนิดนี้จะช่วยให้สามารถเขียนโปรแกรมที่มีโครงสร้างดีและปลอดภัยมากขึ้น

Global Variable ตัวแปรที่ประกาศภายนอกฟังก์ชัน สามารถเข้าถึงและใช้งานได้จากทุกส่วนของโปรแกรม และมักใช้สำหรับค่าที่ไม่เปลี่ยนแปลง หรือค่าที่ต้องการใช้งานร่วมกันในหลายฟังก์ชัน เช่น ค่าคงที่ หรือการตั้งค่าทั่วไป

Local Variable ตัวแปรที่ประกาศภายในฟังก์ชัน มีช่วงชีวิตอยู่เฉพาะในฟังก์ชันนั้น และไม่สามารถเข้าถึงได้จากฟังก์ชันอื่น และช่วยลดผลกระทบของตัวแปรต่อส่วนอื่นของโปรแกรม และเพิ่มความปลอดภัยของข้อมูล โดยป้องกันไม่ให้ข้อมูลถูกเปลี่ยนแปลงโดยไม่ตั้งใจ การแก้ไขค่าของ Global Variable ภายนอกฟังก์ชัน

การแก้ไขค่า Global Variable ภายนอกฟังก์ชัน จะเป็นต้องใช้คำสั่ง `global` เพื่อบอก Python ว่าต้องการแก้ไขตัวแปรที่อยู่นอกฟังก์ชัน มิฉะนั้น Python จะสร้าง Local Variable ขึ้นมาแทน

```
# การใช้งาน
salary = 50000

# กรณีไม่มี global
print("กรณีไม่มี global:")
print(f"อัตราภาษีเริ่มต้น: {TAX_RATE}")
net_salary = salary - calculate_tax(salary)
print(f"เงินเดือนสุทธิ: {net_salary}")

update_tax_rate_without_global(0.15)
net_salary = salary - calculate_tax(salary)
print(f"เงินเดือนสุทธิหลังแก้ไขภาษี (ไม่มี global): {net_salary}")

# กรณีใช้ global
print("\nกรณีใช้ global:")
print(f"อัตราภาษีเริ่มต้น: {TAX_RATE}")
net_salary = salary - calculate_tax(salary)
print(f"เงินเดือนสุทธิ: {net_salary}")
update_tax_rate_with_global(0.15)
net_salary = salary - calculate_tax(salary)
print(f"เงินเดือนสุทธิหลังแก้ไขภาษี (ใช้ global): {net_salary}")
```

ผลลัพธ์ กรณีไม่มี global

ภาษีเริ่มต้น: 0.1

เงินเดือนสุทธิ: 45000.0

อัตราภาษีในฟังก์ชัน (ไม่มี global): 0.15

เงินเดือนสุทธิหลังแก้ไขภาษี (ไม่มี global): 45000.0

- ค่า TAX_RATE ภายในอกฟังก์ชันยังคงเป็น 0.10 เนื่องจากในฟังก์ชัน update_tax_rate_without_global ไม่มีคำสั่ง global
- Python สร้าง Local Variable ชื่อ TAX_RATE ภายในฟังก์ชัน ซึ่งไม่ส่งผลกระทบต่อ Global Variable ที่อยู่นอกฟังก์ชัน
- การคำนวณภาษียังคงใช้ค่า TAX_RATE = 0.10 จาก Global Variable

ผลลัพธ์ กรณีใช้ global

อัตราภาษีเริ่มต้น: 0.1

เงินเดือนสุทธิ: 45000.0

อัตราภาษีในฟังก์ชัน (ใช้ global): 0.15

เงินเดือนสุทธิหลังแก้ไขภาษี (ใช้ global): 42500.0

- ค่า TAX_RATE ถูกเปลี่ยนเป็น 0.15 เนื่องจากในฟังก์ชัน update_tax_rate_with_global มีคำสั่ง global
 - Global Variable TAX_RATE ถูกอัปเดต ส่งผลให้การคำนวณภาษีใช้ค่าใหม่
 - การคำนวณภาษีจึงเปลี่ยนแปลงตามค่าที่ถูกอัปเดต
- การส่งค่าและรับค่าผลลัพธ์ทำให้ฟังก์ชันมีความ "ยืดหยุ่น" มาขึ้น เช่น สามารถนำข้อมูลที่แตกต่างกันเข้าไปประมวลผลในฟังก์ชันเดียวกันได้ โดยไม่ต้องสร้างฟังก์ชันใหม่หลายๆ ตัว ทักษะนี้มีความสำคัญต่อการสร้างโปรแกรมขนาดใหญ่ เพราะจะช่วยให้โค้ดมีประสิทธิภาพและง่ายต่อการดูแลในระยะยาว

4.4 การใช้ฟังก์ชันแก้ไขปัญหา

การนำฟังก์ชันมาใช้แก้ไขปัญหาในโปรแกรมเป็นวิธีการที่ช่วยลดความซับซ้อน ทำให้โค้ดมีความยืดหยุ่น และง่ายต่อการพัฒนา โดยฟังก์ชันช่วยให้สามารถจัดการกับปัญหาได้อย่างมีประสิทธิภาพผ่านการแยกส่วนของงาน และการประมวลผลข้อมูลอย่างเป็นระบบ

4.4.1 แนวคิดการใช้ฟังก์ชันแก้ไขปัญหา

- การวิเคราะห์ปัญหา เริ่มต้นด้วยการวิเคราะห์ปัญหาและแยกเป็นส่วนย่อย (Sub-problems) และจัดลำดับความสำคัญของปัญหาและกำหนดว่าแต่ละส่วนต้องการฟังก์ชันใด

2. การกำหนดหน้าที่ของฟังก์ชัน ออกแบบฟังก์ชันให้มีความชัดเจนในหน้าที่ (Single Responsibility Principle) ตัวอย่างเช่น หากต้องการคำนวณค่าเฉลี่ย ควรสร้างฟังก์ชันที่ทำหน้าที่เฉพาะการคำนวณเท่านั้น เช่น calculate_average()

3. การใช้พารามิเตอร์และผลลัพธ์ (Parameters and Return Values) ใช้พารามิเตอร์เพื่อส่งข้อมูลที่จำเป็นให้กับฟังก์ชัน และใช้ return เพื่อคืนค่าผลลัพธ์ที่สามารถนำไปใช้งานต่อในโปรแกรม

4. การรวมฟังก์ชันเข้ากับโปรแกรมหลัก สร้างฟังก์ชันหลัก (Main Function) เพื่อควบคุมลำดับการทำงานของฟังก์ชันย่อย และ ฟังก์ชันหลักควรมีความกระชับ และเน้นการเรียกใช้ฟังก์ชันย่อยเพื่อแก้ปัญหา

4.4.2 ตัวอย่างการใช้ฟังก์ชันแก้ไขปัญหา

ตัวปัญหาการเขียนโปรแกรม คือ การคำนวณเงินเดือนสุทธิของพนักงาน โดยมีรายละเอียดดังนี้

- รับข้อมูลเงินเดือนพื้นฐาน (Basic Salary)
 - หักค่าภาษี 10%
 - เพิ่มโบนัส 5%
 - แสดงผลเงินเดือนสุทธิ (Net Salary)
- ขั้นตอนการแก้ไขปัญหา วิเคราะห์และออกแบบฟังก์ชัน กำหนดพารามิเตอร์
- ฟังก์ชันสำหรับรับข้อมูล: get_basic_salary()
 - ฟังก์ชันคำนวณภาษี: calculate_tax(salary)
 - ฟังก์ชันคำนวณโบนัส: calculate_bonus(salary)
 - ฟังก์ชันคำนวณเงินเดือนสุทธิ: calculate_net_salary(salary, tax, bonus)
 - ฟังก์ชันหลัก: main() เพื่อควบคุมลำดับการทำงาน

```
# ตัวแปร Global สำหรับอัตราภาษีและโบนัส
TAX_RATE = 0.10 # ภาษี 10%
BONUS_RATE = 0.05 # โบนัส 5%

# ฟังก์ชันรับข้อมูลเงินเดือนพื้นฐาน
def get_basic_salary():
    salary = float(input("Enter the basic salary: "))
    return salary

# ฟังก์ชันคำนวณภาษี
def calculate_tax(salary):
    tax = salary * TAX_RATE
    return tax

# ฟังก์ชันคำนวณโบนัส
def calculate_bonus(salary):
```

```

bonus = salary * BONUS_RATE
return bonus

# พิมพ์ชั้นคำนวณเงินเดือนสุทธิ
def calculate_net_salary(salary, tax, bonus):
    net_salary = salary - tax + bonus
    return net_salary

# พิมพ์ชั้นหลัก
def main():
    # รับข้อมูล
    basic_salary = get_basic_salary()

    # คำนวณภาษีและโบนัส
    tax = calculate_tax(basic_salary)
    bonus = calculate_bonus(basic_salary)

    # คำนวณเงินเดือนสุทธิ
    net_salary = calculate_net_salary(basic_salary, tax, bonus)

    # แสดงผล
    print("\n--- Salary Summary ---")
    print(f"Basic Salary: {basic_salary:.2f}")
    print(f"Tax Deduction: {tax:.2f}")
    print(f"Bonus: {bonus:.2f}")
    print(f"Net Salary: {net_salary:.2f}")

# เรียกใช้งานโปรแกรม
main()

```

อธิบายโค้ด

- กำหนดอัตราภาษีและโบนัสเป็นตัวแปร Global เพื่อปรับเปลี่ยนได้ง่าย
- พิมพ์ชั้นสำคัญ
 - get_basic_salary: รับเงินเดือนพื้นฐานจากผู้ใช้
 - calculate_tax: คำนวณภาษีจากเงินเดือนพื้นฐาน
 - calculate_bonus: คำนวณโบนัสจากเงินเดือนพื้นฐาน
 - calculate_net_salary: คำนวณเงินเดือนสุทธิ (เงินเดือน - ภาษี + โบนัส)
- พิมพ์ชั้นหลัก main ทำหน้าที่รวมกระบวนการทั้งหมด ได้แก่ รับข้อมูล คำนวณภาษี โบนัส และเงินเดือนสุทธิ จากนั้นแสดงผล

ผลลัพธ์

```
Enter the basic salary: 50000
```

```
--- Salary Summary ---
```

Basic Salary: 50000.00

Tax Deduction: 5000.00

Bonus: 2500.00

Net Salary: 47500.00

คำอธิบายผลลัพธ์

- เงินเดือนพื้นฐาน = 50000
- ภาษี = $50000 \times 0.10 = 5000$
- โบนัส = $50000 \times 0.05 = 2500$
- เงินเดือนสุทธิ = $50000 - 5000 + 2500 = 47500$

4.4.3 ข้อดีของการประยุกต์ใช้ฟังก์ชัน

1. ลดความซับซ้อนของโค้ด ทำให้โปรแกรมแบ่งเป็นส่วนย่อย เช้าใจง่าย
2. นำกลับมาใช้ซ้ำ (Reusability): สามารถเรียกฟังก์ชันในส่วนต่าง ๆ ของโปรแกรมหรือโปรแกรมอื่น ๆ ได้
3. เพิ่มประสิทธิภาพในการทำงานร่วมกัน: สามารถแบ่งงานเป็นฟังก์ชันย่อย ให้สมาชิกแต่ละคนรับผิดชอบคละฟังก์ชัน โดยไม่กระทบกันมาก
4. บำรุงรักษาง่าย (Maintenance): เมื่อมีการเปลี่ยนแปลง แก้ไขโค้ดเฉพาะฟังก์ชันที่เกี่ยวข้อง การใช้ฟังก์ชันช่วยให้สามารถ "แยกแยะ" ปัญหาออกเป็นส่วนย่อยๆ และจัดการได้อย่างเป็นระบบ ยิ่งงานมีความซับซ้อนมากเท่าไหร่ การใช้ฟังก์ชันย่อยๆ มาจัดการในแต่ละส่วนยิ่งช่วยลดความซับซ้อนและความผิดพลาดของโปรแกรมได้ดี ยิ่งกว่านั้นยังช่วยให้การแก้ไขหรือปรับปรุงในอนาคตเป็นเรื่องง่าย

4.5 การสร้างโมดูล

การสร้างโมดูล (Create Module) เป็นขั้นตอนที่ต้องอดจากแนวคิดการแยกโค้ดเป็นฟังก์ชันย่อย เมื่อโปรแกรมมีฟังก์ชันจำนวนมาก หรือโครงการมีขนาดใหญ่ การรวมฟังก์ชันเป็นไฟล์แยก (Module) แต่ละไฟล์จะทำหน้าที่เป็น “คลังฟังก์ชัน” สำหรับงานเฉพาะด้าน ช่วยให้การจัดการโค้ดเป็นระบบมากขึ้น และสามารถนำโมดูลนั้นไปใช้ซ้ำในโปรแกรมอื่น ๆ ได้ง่าย

4.5.1 ความหมายของโมดูล

โมดูล (Module) คือ ไฟล์ Python (.py) ที่รวบรวมฟังก์ชัน คลาส หรือค่าคงที่ (Constant) ที่เกี่ยวข้องกันไว้ด้วยกัน เช่น math.py, string_utils.py, file_handler.py

การใช้งาน สามารถนำเข้า (Import) โมดูลเหล่านี้เข้ามาในโปรแกรมอื่น เพื่อเรียกใช้ฟังก์ชันและคลาสที่อยู่ในโมดูล ตัวอย่างการใช้งานโมดูล random โดยที่โมดูล random ใน Python เป็นโมดูลมาตรฐานที่ช่วยจัดการเกี่ยวกับการสุ่มตัวเลขและข้อมูล

1. นำเข้าโมดูลทั้งหมด

```
import random

random_number = random.randint(1, 100) # สรุมตัวเลขระหว่าง 1 ถึง 100
print(f"Random Number: {random_number}")
```

2. นำเข้าเฉพาะฟังก์ชัน

```
from random import choice, shuffle

# เลือกแบบสุ่มจากลิสต์
items = ["Apple", "Banana", "Cherry"]
selected_item = choice(items)
print(f"Selected Item: {selected_item}")
```

4.5.2 ขั้นตอนการสร้างโมดูล

1. สร้างไฟล์ Python ใหม่ ตั้งชื่อไฟล์สี่อความหมายถึงหน้าที่หรือเนื้อหาที่อยู่ภายใน เช่น math_utils.py, data_processing.py

2. กำหนดฟังก์ชันหรือคลาสที่ต้องการใช้งานร่วมกัน เขียนฟังก์ชันที่เกี่ยวข้องกันและเป็นกลุ่มงานเดียวกันลงในไฟล์โมดูลนั้น

3. ทดสอบการทำงานของโมดูล อาจทดสอบภายในไฟล์โมดูลด้วยการเรียกใช้งานฟังก์ชันผ่าน if __name__ == "__main__": เพื่อให้แน่ใจว่าแต่ละฟังก์ชันทำงานถูกต้องก่อนนำไปใช้งาน ตัวอย่างโค้ดโมดูล (math_utils.py)

```
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b != 0:
        return a / b
    else:
        return None

# ส่วนทดสอบการทำงานในโมดูล
if __name__ == "__main__":
    print(add(3, 4))      # 7
    print(subtract(10, 2)) # 8
    print(multiply(5, 5)) # 25
    print(divide(10, 0))  # None
```

4.5.3 การนำโมดูลไปใช้งาน (Import Module)

เมื่อสร้างโมดูลแล้ว สามารถนำโมดูลนี้ไปใช้งานในไฟล์ Python อื่น ๆ ได้โดยใช้คำสั่ง import หรือ from ... import ... ตามความเหมาะสม

1. นำเข้าทั้งโมดูล

```
import math_utils

result = math_utils.add(5, 10)
print(result) # 15
```

เรียกฟังก์ชันในโมดูลโดยระบุชื่อโมดูล math_utils ตามด้วยเครื่องหมายจุด(.) math_utils.add(5, 10)

2. นำเข้าทั้งโมดูล

```
from math_utils import multiply, divide

product = multiply(4, 5)
quotient = divide(20, 4)
print(product) # 20
print(quotient) # 5
```

เรียกใช้งานฟังก์ชันได้โดยตรงโดยไม่ต้องพิมพ์ชื่อโมดูล

3. นำเข้าฟังก์ชันทั้งหมดโดยกำหนดชื่อย่อ

```
from math_utils import *

result = add(5, 10)
print(result)
```

เรียกใช้งานฟังก์ชันได้โดยตรงโดยไม่ต้องพิมพ์ชื่อโมดูลระหว่างชื่อฟังก์ชันซ้ำ (Name Conflict) ถ้าในโปรแกรมหลักมีฟังก์ชันชื่อเดียวกับในโมดูล

โมดูลเปรียบเสมือน "คลังเครื่องมือ" ที่เก็บฟังก์ชันที่เกี่ยวข้องกันไว้ในที่เดียว ทำให้โค้ดดูเป็นระเบียบมากขึ้น และสามารถนำโมดูลนั้นไปใช้ช้าในโปรแกรมอื่นๆ ได้สะดวก การเรียนรู้การสร้างโมดูลตั้งแต่ขั้นพื้นฐานจะช่วยเตรียมความพร้อมสำหรับการทำโปรเจกต์ขนาดใหญ่ ที่ต้องการจัดการกับโค้ดจำนวนมากให้มีประสิทธิภาพ

บทสรุป

ฟังก์ชัน (Function) ถือเป็นหัวใจสำคัญในการพัฒนาโปรแกรมคอมพิวเตอร์ยุคปัจจุบัน เนื่องจากช่วยให้โค้ดมีโครงสร้างที่ชัดเจน ลดความซับซ้อน และส่งเสริมการนำโค้ดกลับมาใช้ซ้ำ (Reusability) การใช้ฟังก์ชันช่วยแบ่งโปรแกรมขนาดใหญ่ออกเป็นส่วนย่อยๆ ที่สามารถจัดการและดูแลรักษาได้ง่ายขึ้น นอกจากนี้ ยังช่วยเพิ่มประสิทธิภาพในการทำงานเป็นทีม เนื่องจากแต่ละคนสามารถรับผิดชอบพัฒนาฟังก์ชันที่แตกต่างกันได้โดยไม่กระทบกับการทำงานของผู้อื่น

บทนี้จะนำไปสู่ความเข้าใจแนวคิดพื้นฐานเกี่ยวกับฟังก์ชัน ตั้งแต่การสร้างฟังก์ชัน การเรียกใช้งาน การส่งค่า (Parameters) และการรับค่าผลลัพธ์ (Return Values) ไปจนถึงการจัดการกับตัวแปรแบบโกลบอล (Global) และโลคอล (Local) เพื่อให้สามารถเขียนโค้ดที่ปลอดภัยและมีประสิทธิภาพ นอกจากนี้ ยังจะแนะนำการใช้ฟังก์ชันในการแก้ไขปัญหาในโปรแกรมจริง และการสร้างโมดูล (Module) เพื่อร่วบรวมฟังก์ชันที่เกี่ยวข้องกันไว้ในไฟล์เดียว ซึ่งจะช่วยให้โปรแกรมมีความเป็นระเบียบและสามารถนำไปใช้ซ้ำในโปรเจกต์อื่นๆ ได้อย่างสะดวก

- ฟังก์ชัน (Function) หมายถึงกลุ่มของคำสั่งที่ทำงานเพื่อวัตถุประสงค์เฉพาะ คล้ายกับ “หน่วยย่อย” หรือ “LEGO” ที่สามารถนำไปประกอบใช้ซ้ำได้
- การเขียนโปรแกรมแบบโมดูลาร์ เน้นการแบ่งโปรแกรมเป็นส่วนย่อย (Building Block) ทำให้ได้มีโครงสร้างชัดเจนและบำรุงรักษาได้ง่าย
- ตัวอย่างทางคณิตศาสตร์ เช่น $f(x, y) = 3x - 2y + 5$ แสดงแนวคิดว่าฟังก์ชันจะรับข้อมูล (Input) และประมวลผลจนได้ผลลัพธ์ (Output)
- ประโยชน์ของฟังก์ชัน ช่วยลดความซ้ำซ้อนของโค้ด สร้าง Reusability ลดความซับซ้อน และทำให้การบำรุงรักษาง่ายขึ้น
- บทบาทฟังก์ชัน ช่วยเพิ่มความปลอดภัยของโค้ด (จำกัดการเข้าถึงข้อมูล) เพิ่มประสิทธิภาพ (แยกเป็นส่วนเล็ก ๆ) สนับสนุนการทำงานเป็นทีม และปรับเปลี่ยนโค้ดได้ง่าย
- การสร้างฟังก์ชันขั้นพื้นฐานใน Python ใช้คำสั่ง `def` ตามด้วยชื่อฟังก์ชันและเครื่องหมายวงเล็บ (): จำนวนนี้ยื่งโค้ดที่เป็นคำสั่งภายใน
- ลำดับการรับฟังก์ชัน เมื่อเจอคำสั่ง `def` จะเป็นการประกาศฟังก์ชันไว้ก่อน จนกว่าจะถูกเรียกใช้จะประมวลผลคำสั่งภายใน
- ลดความซ้ำซ้อนของโค้ดด้วยฟังก์ชัน หากมีการพิมพ์หรือคำนวนซ้ำหลายจุด ควรรวบรวมไว้ในฟังก์ชันเดียว และเรียกใช้แทนการคัดลอกโค้ดซ้ำ
- พารามิเตอร์ (Parameters) คือ ตัวแปรที่ประกาศหลังชื่อฟังก์ชัน เช่น `def add_numbers(a, b):` เพื่อรับค่าจากภายนอกเข้าไปประมวลผล
- *การส่งพารามิเตอร์หลายค่า (args) ทำให้ฟังก์ชันยอมรับข้อมูลจำนวนไม่จำกัด เช่น `def calculate_sum(*numbers):` และคือยนำมาประมวลผลภายใน
- การรับค่าผลลัพธ์ (Return Values) ใช้คำสั่ง `return` เพื่อส่งค่ากลับไปยังส่วนที่เรียกฟังก์ชัน เช่น `return a + b`
- การส่งคืนหลายค่า สามารถ `return` ได้มากกว่าหนึ่งค่าในฟังก์ชันเดียว เช่น `return sum_result, product_result` และรับด้วยตัวแปรหลายตัว

- Global Variable คือ ตัวแปรที่ประกาศนอกฟังก์ชัน ทุกฟังก์ชันสามารถเข้าถึงได้ แต่ต้องระวังการแก้ไขค่าโดยไม่ตั้งใจ
- Local Variable คือ ตัวแปรที่ประกาศภายในฟังก์ชัน มีช่วงชีวิต (Scope) อยู่เฉพาะฟังก์ชันนั้น ช่วยลดโอกาสสับสนหรือการแก้ไขข้ามฟังก์ชัน
- การใช้ฟังก์ชันแก้ไขปัญหา เริ่มจากการวิเคราะห์ปัญหา แยกเป็นส่วนย่อย และกำหนดฟังก์ชันแต่ละส่วนให้มีหน้าที่ชัดเจน
- แนวคิดการออกแบบ ตามหลัก Single Responsibility Principle ฟังก์ชันหนึ่งฟังก์ชันควรทำหน้าที่เฉพาะ เช่น การคำนวน การอ่านอินพุต หรือการแสดงผล
- ข้อดีของการประยุกต์ใช้ฟังก์ชัน ลดความซับซ้อน นำฟังก์ชันกลับมาใช้ซ้ำได้ง่าย ทำงานร่วมกันเป็นทีมได้ดี และบำรุงรักษาง่าย
- โมดูล (Module) คือ ไฟล์ .py ที่รวมฟังก์ชันหรือคลาสที่เกี่ยวข้องไว้ด้วยกัน เช่น math_utils.py เพื่อให้สะดวกต่อการนำเข้าไปใช้ที่อื่น
- วิธีการนำเข้า (Import) ทำได้หลายแบบ เช่น import module_name, from module_name import function_name หรือ from module_name import *
- ประโยชน์ของโมดูล ทำให้โค้ดเป็นระบบ จัดการง่าย นำไปใช้ซ้ำในโปรแกรมอื่นได้สะดวก และลดปัญหาความซ้ำซ้อนของฟังก์ชันในโปรเจกต์ใหญ่ ๆ

คำถามก้ายบท

1. อธิบายความหมายของฟังก์ชัน (Function) และประโยชน์ที่สำคัญในการพัฒนาโปรแกรม
2. อะไรมีคือพารามิเตอร์ (Parameters) และค่าที่ส่งคืน (Return Values) ในฟังก์ชัน? ยกตัวอย่าง
3. ลำดับการรันของฟังก์ชันใน Python ทำงานอย่างไร? และ Call Stack มีความสำคัญอย่างไร?
4. อธิบายความแตกต่างระหว่างตัวแปร Global และ Local พร้อมตัวอย่างการใช้งาน
5. ในกรณีได้คราวใช้ฟังก์ชันเพื่อแก้ไขปัญหา แทนการเขียนโค้ดซ้ำ ๆ?
6. อะไรมีคือโมดูล (Module) และมีบทบาทสำคัญอย่างไรในโปรแกรมขนาดใหญ่?
7. อธิบายการนำเข้าโมดูลใน Python และวิธีการใช้งาน import และ from ... import ...
8. ฟังก์ชัน *args และ **kwargs ใช้ประโยชน์ในสถานการณ์ใด? พร้อมตัวอย่าง
9. การออกแบบฟังก์ชันตามหลัก Single Responsibility Principle หมายถึงอะไร?
10. ให้ตัวอย่างสถานการณ์ที่ควรใช้ฟังก์ชันคืนค่ามากกว่าหนึ่งค่า และอธิบายการทำงาน
11. อธิบายขั้นตอนการสร้างฟังก์ชันที่มีพารามิเตอร์และคืนค่าผลลัพธ์
12. การใช้ฟังก์ชันเพื่อปรับปรุงประสิทธิภาพของทีมงานพัฒนาโปรแกรม มีความสำคัญอย่างไร?
13. เปรียบเทียบข้อดีและข้อเสียของการใช้ Global Variables ในโปรแกรม
14. ให้ตัวอย่างการสร้างฟังก์ชันที่ใช้การส่งค่าพารามิเตอร์หลายค่า (Multiple Parameters)

15. อธิบายการใช้คำสั่ง `return` ในฟังก์ชัน และเหตุผลที่ควรใช้คำสั่งนี้
16. ฟังก์ชันแบบไม่มีพารามิเตอร์ (No-Parameter Function) มีประโยชน์ในสถานการณ์ใด?
17. อธิบายการสร้างฟังก์ชันในโมดูลและการทดสอบการทำงานของฟังก์ชันในโมดูล
18. ให้ตัวอย่างการเขียนฟังก์ชันที่ส่งค่ากลับมาเป็น `list` และอธิบายสถานการณ์ที่ควรใช้งาน
19. ให้ตัวอย่างโปรแกรมที่ใช้ฟังก์ชันเพื่อคำนวณค่าภาษีและโบนัสในระบบเงินเดือน
20. การนำฟังก์ชันจากโมดูลมาตรฐานใน Python เช่น `math` หรือ `random` มาใช้ในโปรแกรม

โจทย์การเขียนโปรแกรม

21. เขียนโปรแกรมสร้างฟังก์ชัน เพื่อรับชื่อ อายุ และสีโปรด จากนั้นแสดงข้อความแนะนำตัว

```
Enter your name: Alice
Enter your age: 25
Enter your favorite color: Blue
Hello, Alice! You are 25 years old and your favorite color is Blue.
```

22. เขียนโปรแกรมสร้างฟังก์ชัน คำนวณเลขยกกำลัง

```
Enter the base number: 2
Enter the exponent: 3
2 raised to the power of 3 is 8
```

23. เขียนโปรแกรมสร้างฟังก์ชัน เพื่อค้นหาตัวเลขที่มากที่สุดในชุดตัวเลข 5 ตัวที่ผู้ใช้ป้อนเข้ามา

```
Enter number 1: 10
Enter number 2: 25
Enter number 3: 5
Enter number 4: 30
Enter number 5: 20
The largest number is: 30
```

24. เขียนโปรแกรมสร้างฟังก์ชัน สร้างรูปแบบปริมิตตามจำนวนชั้นที่ผู้ใช้กำหนด

```
Enter the number of levels: 4
#
###
#####
#####
#####
```

25. เขียนโปรแกรมสร้างฟังก์ชัน คำนวณพื้นที่สี่เหลี่ยมผืนผ้า

```
Enter the length: 5
Enter the width: 3
The area of the rectangle is 15.00
```

26. เขียนโปรแกรมสร้างฟังก์ชัน คำนวณเกรดจากคะแนนที่ผู้ใช้ป้อนเข้าไป และแสดงข้อความแนะนำ (`A`, `B`, `C`, `D`, `F`)

```
Enter your score: 85
Your grade is: A
```

27. เขียนโปรแกรมสร้างฟังก์ชัน ตรวจสอบว่าข้อความที่ผู้ใช้ป้อนเข้าไปเป็นพอลินโตรม (palindrome) หรือไม่

```
Enter a word: radar
radar is a palindrome.
```

28. เขียนโปรแกรมสร้างฟังก์ชัน รับชื่อของผู้ใช้แล้วตรวจสอบว่าเป็นชื่อที่กำหนดไว้ล่วงหน้าหรือไม่

```
Enter your name: Alice
Hello, Alice! Welcome back.
```

29. เขียนโปรแกรมสร้างฟังก์ชัน หาค่า Factorial ของตัวเลขที่ผู้ใช้ป้อน

```
Enter a number: 5
The factorial of 5 is 120.
```

30. เขียนโปรแกรมสร้างฟังก์ชัน สร้างตารางสูตรคูณของแม่ที่ผู้ใช้กำหนด (เช่นแม่ 3)

```
Enter a number: 3
3 x 1 = 3
3 x 2 = 6
...
3 x 12 = 36
```

31. เขียนโปรแกรมสร้างฟังก์ชัน รับข้อมูลอายุของบุคคล และแสดงว่าเขาสามารถลงคะแนนเสียงเลือกตั้งได้หรือไม่

```
Enter your age: 17
You are not eligible to vote.
```

32. เขียนโปรแกรมสร้างฟังก์ชัน สร้างสามเหลี่ยมกลับหัวโดยให้ผู้ใช้กำหนดจำนวนแถว

```
#####
####
##
#
```

33. เขียนโปรแกรมสร้างฟังก์ชัน รับคำจากผู้ใช้และตรวจสอบว่าคำนั้นเป็นคำที่ซ้ำกัน (Duplicate) ในลิสต์ หรือไม่

```
Enter a word: apple
The word 'apple' is not in the list. Adding it now.
```

34. เขียนโปรแกรมสร้างฟังก์ชัน แปลงค่าอุณหภูมิจากเซลเซียสเป็นฟาเรนไฮต์

```
Enter temperature in Celsius: 25
Temperature in Fahrenheit is: 77.0
```

35. เขียนโปรแกรมสร้างฟังก์ชัน คำนวณ Factorial ของตัวเลขที่ผู้ใช้ป้อน

```
Enter a number: 5
The factorial of 5 is 120. เขียนโปรแกรมสร้างฟังก์ชัน คำนวณเงินเดือนสุทธิจากค่าภาษีและ
โบนัส
```

```
Enter the basic salary: 50000
-- Salary Summary ---
Basic Salary: 50000.00
Tax Deduction: 5000.00
Bonus: 2500.00
Net Salary: 47500.00
```

36. เขียนโปรแกรมสร้างฟังก์ชัน สร้างกรอบข้อความจากข้อความที่ผู้ใช้ป้อน

```
Enter a message: Hello
+-----+
| Hello |
+-----+
```

37. เขียนโปรแกรมสร้างฟังก์ชัน เกมจำลองล็อกอิน โดยผู้ใช้ต้องป้อนชื่อผู้ใช้และรหัสผ่านให้ถูกต้อง

```
Enter username: admin
Enter password: 1234
Login successful. Welcome!
```

38. เขียนโปรแกรมสร้างฟังก์ชัน ตรวจสอบว่าเลขที่ผู้ใช้ป้อนเป็นเลขคู่หรือเลขคี่

```
Enter a number: 7
7 is odd.
```

39. เขียนโปรแกรมสร้างฟังก์ชัน เพื่อสร้างรูปแบบปริมาמידตามจำนวนชั้นที่ผู้ใช้กำหนด

```
Enter the number of levels: 4
```

```
#  
###  
#####  
######
```

40. โปรแกรมคำนวณพื้นที่รูปทรงเรขาคณิตโดยใช้โมดูล สร้างโมดูล geometry.py ที่มีฟังก์ชัน:

- circle_area(radius) สำหรับคำนวณพื้นที่วงกลม
- rectangle_area(length, width) สำหรับคำนวณพื้นที่สี่เหลี่ยมผืนผ้า
- triangle_area(base, height) สำหรับคำนวณพื้นที่สามเหลี่ยม

```
import geometry

print("Choose a shape to calculate area:")
print("1. Circle\n2. Rectangle\n3. Triangle")
choice = int(input("Enter your choice: "))

if choice == 1:
    radius = float(input("Enter radius: "))
    print(f"Area of circle: {geometry.circle_area(radius):.2f}")
elif choice == 2:
    length = float(input("Enter length: "))
    width = float(input("Enter width: "))
    print(f"Area of rectangle: {geometry.rectangle_area(length, width):.2f}")
elif choice == 3:
    base = float(input("Enter base: "))
    height = float(input("Enter height: "))
    print(f"Area of triangle: {geometry.triangle_area(base, height):.2f}")
else:
    print("Invalid choice!")
```

Choose a shape to calculate area:

1. Circle

```
2. Rectangle  
3. Triangle  
Enter your choice: 1  
Enter radius: 5  
Area of circle: 78.54
```

บทที่ 5

การเขียนโปรแกรมเชิงโครงสร้าง

เนื้อหาบทเรียน

- การระบุปัญหาและแนวคิดการเขียนโปรแกรมเชิงโครงสร้าง
- การออกแบบโปรแกรมด้วยแนวคิด การออกแบบจากบนลงล่าง (Top-Down Design)
- การออกแบบโปรแกรมแบบจากบนลงล่าง
- การพัฒนาโปรแกรมเพื่อแก้ปัญหา
- ตัวอย่างและการประยุกต์ใช้งาน

วัตถุประสงค์การเรียนรู้

- เข้าใจแนวคิดของการเขียนโปรแกรมเชิงโครงสร้าง รวมถึงองค์ประกอบสำคัญ เช่น ลำดับการทำงาน (Sequence) เงื่อนไขการตัดสินใจ (Selection) และการวนซ้ำ (Iteration)
- ออกแบบโปรแกรมด้วยแนวคิด Top-Down Design โดยสามารถแบ่งปัญหาออกเป็นส่วนย่อย และจัดลำดับขั้นตอนการแก้ปัญหาได้
- เขียนโปรแกรมที่สามารถรับข้อมูลจากผู้ใช้ ประมวลผลข้อมูล และแสดงผลลัพธ์ที่ถูกต้องตามข้อกำหนด
- ใช้ผังงาน (Flowchart) หรือ Pseudocode ในการวางแผนและพัฒนาโปรแกรมทีมีโครงสร้างชัดเจน
- ทดสอบและแก้ไขข้อผิดพลาดในโปรแกรม เพื่อปรับปรุงคุณภาพและประสิทธิภาพของโปรแกรมอย่างมีระบบ

ในยุคปัจจุบันที่เทคโนโลยีดิจิทัลเข้ามามีบทบาทสำคัญในชีวิตประจำวัน การเขียนโปรแกรมได้กลายเป็นทักษะพื้นฐานที่ช่วยให้สามารถสร้างซอฟต์แวร์เพื่อแก้ไขปัญหาและพัฒนาวัตกรรมใหม่ ๆ อย่างต่อเนื่อง การเขียนโปรแกรมเชิงโครงสร้าง (Structured Programming) เป็นหนึ่งในแนวทางสำคัญที่ช่วยให้นักพัฒนาสามารถจัดการโค้ดได้อย่างเป็นระบบ ลดความซับซ้อน และเพิ่มประสิทธิภาพในการพัฒนาโปรแกรม

บทเรียนนี้จะพาผู้อ่านทำความเข้าใจกับแนวคิดพื้นฐานของการเขียนโปรแกรมเชิงโครงสร้าง ตั้งแต่การระบุปัญหา การออกแบบโครงสร้างโปรแกรม ไปจนถึงการพัฒนาโปรแกรมที่สามารถแก้ปัญหาได้อย่างมีประสิทธิภาพ จะเรียนรู้แนวคิดการออกแบบจากบนลงล่าง (Top-Down Design) ซึ่งช่วยให้การแก้ปัญหามีลำดับขั้นตอนที่ชัดเจน และสามารถนำไปประยุกต์ใช้กับการพัฒนาโปรแกรมในโลกจริงได้ นอกจากนี้ ตัวอย่างโปรแกรมและการประยุกต์ใช้งานจะช่วยให้ผู้อ่านมองเห็นภาพรวมของการนำแนวคิดไปใช้ในสถานการณ์ต่าง ๆ ได้ชัดเจนยิ่งขึ้น

วัตถุประสงค์ของบทเรียนนี้ไม่เพียงแต่ช่วยให้ผู้อ่านเข้าใจแนวคิดการเขียนโปรแกรมเชิงโครงสร้าง เท่านั้น แต่ยังมุ่งเน้นให้ผู้อ่านสามารถนำแนวคิดนี้ไปปรับใช้กับการพัฒนาโปรแกรมที่ตอบโจทย์ความต้องการได้อย่างมีคุณภาพและมีความเป็นมืออาชีพ

5.1 แนวคิดการเขียนโปรแกรมเชิงโครงสร้าง

ในโลกปัจจุบันที่ซอฟต์แวร์มีความซับซ้อนมากขึ้น การพัฒนาโปรแกรมที่มีขนาดใหญ่และประกอบด้วยคำสั่งหลายพันหรือหลายหมื่นบรรทัดมักเผชิญกับปัญหาต่าง ๆ เช่น ความยุ่งยากในการตรวจสอบข้อผิดพลาด ความลำบากในการแบ่งงานสำหรับทีมพัฒนา และการบำรุงรักษาที่ใช้เวลามาก การเขียนโปรแกรมเชิงโครงสร้างจึงเข้ามามีบทบาทสำคัญ เพราะช่วยให้สามารถควบคุมการทำงานของโปรแกรมได้อย่างมีระบบ ลดความซับซ้อนในการเข้าใจโปรแกรม และสนับสนุนการพัฒนาแบบทีม

การเขียนโปรแกรมเชิงโครงสร้างเป็นแนวทางที่สำคัญในการพัฒนาซอฟต์แวร์ โดยมุ่งเน้นการจัดโครงสร้างโค้ดให้มีลำดับการทำงานที่ชัดเจนและเข้าใจง่าย การจัดการโปรแกรมด้วยวิธีนี้ช่วยลดความซับซ้อนของโค้ด โดยการแบ่งส่วนโปรแกรมออกเป็นโครงสร้างย่อย ๆ ที่มีหน้าที่เฉพาะเจาะจง การแบ่งโครงสร้างดังกล่าวช่วยให้โปรแกรมสามารถดูแลและปรับปรุงได้ง่าย ทั้งยังช่วยเพิ่มประสิทธิภาพในการค้นหาและแก้ไขข้อผิดพลาดได้อย่างรวดเร็ว

การออกแบบโปรแกรมจากบนลงล่าง (Top-Down Design) เป็นอีกแนวคิดสำคัญของการเขียนโปรแกรมเชิงโครงสร้าง โดยเริ่มจากการมองภาพรวมของปัญหาและแบ่งออกเป็นส่วนย่อย ๆ ที่สามารถจัดการได้ง่าย การแบ่งปัญหานี้ช่วยให้การพัฒนาโปรแกรมมีโครงสร้างชัดเจนและสามารถแก้ไขปัญหาได้ตรงจุด ตัวอย่างเช่น ในการพัฒนาโปรแกรมคำนวณพื้นที่ของรูปทรงเรขาคณิต สามารถออกแบบโดยเริ่มจากขั้นตอน

หลัก เช่น การเลือกประเภทของรูปทรง จากนั้นพัฒนาโมดูลหรือฟังก์ชันที่เกี่ยวข้อง เช่น การคำนวณค่าพื้นที่ของรูปทรงลูกบาศก์หรือทรงกลม

การเขียนโปรแกรมเชิงโครงสร้างยังเน้นการสร้างโมดูลหรือฟังก์ชันเฉพาะที่แก้ปัญหาได้อย่างมีประสิทธิภาพ การแยกการทำงานออกเป็นฟังก์ชันช่วยลดการเขียนโค้ดซ้ำซ้อนและเพิ่มความชัดเจนในกระบวนการทำงาน ตัวอย่างเช่น การสร้างฟังก์ชันสำหรับการรับค่าข้อมูลจากผู้ใช้ ฟังก์ชันสำหรับคำนวณ และฟังก์ชันสำหรับการแสดงผลลัพธ์ แต่ละฟังก์ชันจะรับผิดชอบเฉพาะงานของตัวเอง ทำให้โค้ดสามารถอ่านและพัฒนาได้ง่ายขึ้น

5.1.1 โครงสร้างพื้นฐานของโปรแกรมเชิงโครงสร้าง

เมื่อโปรแกรมมีขนาดใหญ่ขึ้น การจัดการโปรแกรมแบบไม่มีโครงสร้างมักทำให้เกิดปัญหา เช่น การใช้คำสั่ง "goto" ซึ่งทำให้โปรแกรมมีลำดับการทำงานที่กระโดดไปมาและยากต่อการติดตาม การเขียนโปรแกรมเชิงโครงสร้างช่วยแก้ไขปัญหานี้โดยการสนับสนุนการใช้โครงสร้างควบคุมที่ชัดเจน ได้แก่

- **Block** การเขียนคำสั่งแบบต่อเนื่อง (Sequential Block) เป็นพื้นฐานของโปรแกรมที่ดำเนินการตามลำดับที่กำหนดไว้ เช่น การรับค่า ประมวลผล และแสดงผลลัพธ์
- **Selection** การตัดสินใจโดยใช้โครงสร้างเงื่อนไข เช่น if-else เพื่อแยกการทำงานตามเงื่อนไขที่กำหนด
- **Iteration** การวนซ้ำ เช่น for หรือ while loop ช่วยให้โปรแกรมสามารถทำงานซ้ำ ๆ ได้จนกว่าจะตรงตามเงื่อนไขที่กำหนด
- **Nesting** การซ้อนโครงสร้าง เช่น การใช้ลูปภายในเงื่อนไขหรือเงื่อนไขภายในลูป เพื่อเพิ่มความยืดหยุ่นและตอบสนองต่อความซับซ้อนของปัญหา
- **Subroutines** การสร้างฟังก์ชันหรือโมดูลย่อยที่แยกการทำงานออกเป็นส่วนย่อย เช่น การรับค่าข้อมูล การคำนวณ หรือการแสดงผล เพื่อเพิ่มความชัดเจนและลดการเขียนโค้ดซ้ำซ้อน
- **Encapsulation** การซ่อนรายละเอียดภายในฟังก์ชันหรือโมดูล เพื่อให้โค้ดส่วนอื่นสามารถใช้งานได้โดยไม่ต้องรู้รายละเอียดการทำงาน
- **Control Flow** การควบคุมลำดับการทำงานของโปรแกรม เช่น การเรียกใช้ฟังก์ชันหรือการจัดลำดับคำสั่งอย่างมีระบบ
-

5.1.2 ประเภทของโปรแกรมเชิงโครงสร้าง

โปรแกรมเชิงโครงสร้างสามารถแบ่งออกได้เป็นหลายประเภท โดยแต่ละประเภทมีลักษณะเฉพาะและ การใช้งานที่เหมาะสมกับลักษณะของปัญหา ดังนี้

- **Procedural Programming** เป็นแนวทางการเขียนโปรแกรมที่มุ่งเน้นการจัดลำดับขั้นตอนของการทำงานในโปรแกรม เริ่มจากการกำหนดคำสั่งในลำดับที่แน่นอน และดำเนินการตามขั้นตอนที่กำหนดไว้ หลักการนี้เหมาะสมสำหรับการพัฒนาระบบที่มีลำดับการทำงานที่ชัดเจนและไม่ซับซ้อน ตัวอย่างเช่น การพัฒนาโปรแกรมคำนวณคะแนนเฉลี่ยของนักเรียน โดยมีขั้นตอนต่อไปนี้
 - รับค่าคะแนนจากผู้ใช้ คำนวณค่าเฉลี่ย และแสดงผลลัพธ์
- **Object-oriented Programming (OOP)** เป็นแนวคิดการเขียนโปรแกรมที่จัดการข้อมูลและการทำงานในรูปแบบของวัตถุ (Objects) และคลาส (Classes) ซึ่งวัตถุแต่ละตัวจะมีคุณสมบัติ (Attributes) และพฤติกรรม (Methods) ที่เป็นเอกลักษณ์ แนวคิดนี้ช่วยเพิ่มความยืดหยุ่นและสามารถนำกลับมาใช้ใหม่ได้ ตัวอย่างเช่น การสร้างระบบบริหารจัดการนักศึกษา โดยอาจมีคลาส "Student" ที่มีคุณสมบัติ เช่น ชื่อและคะแนน และพฤติกรรม เช่น การคำนวณเกรดหรือการแสดงข้อมูลนักศึกษา
- **Model-based Programming** เป็นการเขียนโปรแกรมที่เน้นการใช้แบบจำลองหรือโมเดลในการพัฒนา โดยโปรแกรมจะถูกออกแบบและพัฒนาตามโครงสร้างหรือกระบวนการที่ระบุไว้ในโมเดล ตัวอย่างที่พัฒนาโดยคือ การเขียนโปรแกรมฐานข้อมูลโดยใช้ SQL ซึ่งเป็นภาษาสำหรับจัดการข้อมูลในฐานข้อมูล หรือการใช้โมเดลทางฟิสิกส์ในโปรแกรมวิศวกรรมศาสตร์ เช่น การคำนวณแรง การวิเคราะห์โครงสร้าง หรือการแก้ปัญหาคณิตศาสตร์ที่ซับซ้อน

5.1.3 เหตุผลที่ควรใช้โปรแกรมเชิงโครงสร้าง

โปรแกรมเชิงโครงสร้างช่วยให้การพัฒนาซอฟต์แวร์เป็นไปอย่างมีระเบียบแบบแผน ลดความซับซ้อนในกระบวนการพัฒนาและบำรุงรักษา โดยเหตุผลหลักที่ควรใช้โปรแกรมเชิงโครงสร้าง ได้แก่

- การเพิ่มความเข้าใจในโปรแกรม การจัดโครงสร้างโปรแกรมให้มีลำดับการทำงานที่ชัดเจนช่วยให้ผู้พัฒนาและผู้ที่ดูแลโปรแกรมในอนาคตสามารถเข้าใจและติดตามการทำงานของโปรแกรมได้ง่ายขึ้น แม้จะไม่ได้เป็นผู้เขียนโปรแกรมด้วยเดิม
- การลดข้อผิดพลาดในโปรแกรม การแบ่งโปรแกรมออกเป็นโมดูลอย่างหรือฟังก์ชันที่มีความรับผิดชอบเฉพาะช่วยวัดข้อผิดพลาดที่อาจเกิดขึ้น เนื่องจากแต่ละโมดูลสามารถพัฒนาและตรวจสอบได้แยกกัน
- การเพิ่มประสิทธิภาพในการพัฒนา การใช้โครงสร้างควบคุมที่ชัดเจน เช่น ลำดับการทำงาน การตัดสินใจ และการวนซ้ำ ช่วยลดเวลาในการเขียนโค้ดและทดสอบโปรแกรม
- การสนับสนุนการทำงานเป็นทีม ในกรณีที่มีนักพัฒนาหลายคน การจัดโครงสร้างโปรแกรมช่วยให้สามารถแบ่งงานกันทำได้ง่ายขึ้น แต่ละคนสามารถพัฒนาโมดูลเฉพาะของตนโดยไม่กระทบกับส่วนอื่นของโปรแกรม

5.1.4 วิธีการช่วยให้โปรแกรมง่ายต่อการเข้าใจและบำรุงรักษา

- การใช้โครงสร้างควบคุมที่ชัดเจน การเขียนโค้ดโดยใช้โครงสร้างควบคุมที่กำหนดไว้ เช่น การทำงานตามลำดับ (Sequence) การใช้เงื่อนไข (Selection) และการวนซ้ำ (Iteration) ทำให้โปรแกรมมีความเรียบง่ายและเข้าใจง่าย
- การแบ่งโปรแกรมออกเป็นโมดูลหรือฟังก์ชัน การแยกโค้ดออกเป็นส่วนย่อยที่มีหน้าที่เฉพาะ เช่น ฟังก์ชันสำหรับการรับข้อมูล การประมวลผล และการแสดงผล ทำให้สามารถบำรุงรักษาและแก้ไขได้ง่าย
- การใช้ชื่อที่สื่อความหมาย การตั้งชื่อตัวแปร ฟังก์ชัน หรือโมดูลที่สื่อความหมายชัดเจนช่วยให้ผู้พัฒนาสามารถเข้าใจหน้าที่ของแต่ละส่วนได้ทันที
- การใช้คำอธิบายโค้ด (Comments) การใส่คำอธิบายในส่วนที่สำคัญของโค้ดช่วยให้ผู้พัฒนาหรือผู้อื่นและโปรแกรมในอนาคตสามารถเข้าใจแนวคิดหรือเหตุผลเบื้องหลังการเขียนโค้ดในลักษณะนั้น ๆ
- การทดสอบและการตรวจสอบโค้ด การจัดโครงสร้างโปรแกรมให้เป็นระเบียบช่วยให้สามารถทดสอบแต่ละส่วนได้อย่างอิสระ ทำให้การตรวจสอบและแก้ไขข้อผิดพลาดทำได้ง่ายและรวดเร็ว ว่าโปรแกรมเชิงโครงสร้างเป็นเหมือน "แม่แบบ" ของการเขียนโปรแกรมในยุคใหม่ ที่ช่วยให้โค้ดมีระเบียบมากขึ้น และง่ายต่อการอ่านหรือปรับปรุงในอนาคต โดยเฉพาะในโปรเจกต์ขนาดใหญ่ที่มีนักพัฒนาหลายคนร่วมมือกัน ความเป็นระบบของโค้ดช่วยลดปัญหาการสื่อสารผิดพลาด และทำให้การแก้ไขหรือเพิ่มฟีเจอร์ใหม่ทำได้รวดเร็วขึ้น

5.2 การระบุปัญหา

การพัฒนาโปรแกรมที่มีประสิทธิภาพและตอบโจทย์ความต้องการของผู้ใช้งานเริ่มต้นจากการบูรณาการวางแผนที่ชัดเจนและเป็นระบบ การระบุปัญหาและข้อกำหนดของระบบเป็นขั้นตอนสำคัญที่ช่วยให้โปรแกรมเมอร์สามารถเข้าใจขอบเขตของปัญหาและเป้าหมายที่ต้องการได้อย่างชัดเจน สำหรับผู้เริ่มต้นศึกษาการเริ่มต้นด้วยปัญหานาดเล็ก เช่น การคำนวณค่าทางคณิตศาสตร์หรือการแก้สมการเบื้องต้น เป็นวิธีที่ดีในการเรียนรู้กระบวนการพัฒนาโปรแกรม กระบวนการนี้ยังรวมถึงการเก็บรวบรวมข้อมูลที่เกี่ยวข้อง การวิเคราะห์ความต้องการ และการออกแบบโครงสร้างโปรแกรมที่เหมาะสม ซึ่งช่วยลดความซับซ้อนและเพิ่มประสิทธิภาพในการพัฒนาโปรแกรม ในเนื้อหาต่อไปนี้จะอธิบายถึงกระบวนการสำคัญตั้งแต่การระบุปัญหา การเก็บข้อมูล ไปจนถึงการออกแบบแนวทางแก้ปัญหาอย่างเป็นระบบ

5.2.1 การระบุปัญหาและข้อกำหนดของระบบ

ขั้นตอนการกำหนดขอบเขตและเป้าหมายของปัญหา

การระบุปัญหาเริ่มต้นด้วยการเข้าใจความต้องการของผู้ใช้งานหรือองค์กร สำหรับผู้เริ่มต้นศึกษาและผู้ฝึกเขียนโปรแกรมเบื้องต้น ควรมุ่งเน้นที่ปัญหานาดเล็ก เช่น การคำนวณค่าพื้นที่ของรูปทรงเรขาคณิต การ

หาก่าเฉลี่ยของตัวเลข หรือการแก้สมการง่าย ๆ กระบวนการนี้รวมถึงการเก็บข้อมูลเบื้องต้นและตรวจสอบความต้องการผ่านการพูดคุยกับผู้ใช้งานหรือการศึกษาเอกสารที่เกี่ยวข้อง

เมื่อเข้าใจปัญหาแล้ว เป้าหมายของโปรแกรมควรกำหนดให้ชัดเจน เช่น การแสดงผลลัพธ์ที่เข้าใจง่าย การคำนวณที่แม่นยำ และการตอบสนองที่รวดเร็ว เป้าหมายที่ชัดเจนช่วยให้ผู้พัฒนาสามารถวางแผนและออกแบบโครงสร้างโปรแกรมได้อย่างมีประสิทธิภาพ

การกำหนดขอบเขตของระบบช่วยระบุงานที่ระบบต้องทำ เช่น การรับค่าจากผู้ใช้งาน การประมวลผลข้อมูล และการแสดงผลลัพธ์ การจำกัดขอบเขตนี้ช่วยลดความซับซ้อนและเหมาะสมสำหรับการฝึกเขียนโปรแกรมที่มีโครงสร้างชัดเจน ข้อมูลทั้งหมดควรบันทึกในรูปแบบที่ง่าย เช่น ลำดับขั้นตอนการทำงานหรือผังงาน (Flowchart) เพื่อช่วยในการพัฒนาและแก้ปัญหา

ขั้นตอนการวิเคราะห์ข้อกำหนดของระบบ

การวิเคราะห์ข้อกำหนดของระบบเริ่มจากการระบุฟังก์ชันหลักที่โปรแกรมต้องมี เช่น การคำนวณตัวเลข การแก้สมการ หรือการสร้างรายงานผลลัพธ์ สำหรับผู้เริ่มต้นศึกษา การวิเคราะห์นี้ควรมุ่งเน้นที่การทำความเข้าใจกระบวนการการทำงานพื้นฐานและการแบ่งงานออกเป็นส่วนย่อยที่สามารถจัดการได้ง่าย

ข้อกำหนดที่ไม่ใช่หน้าที่ เช่น ความเร็วในการประมวลผลและความง่ายในการใช้งาน ควรถูกระบุอย่างชัดเจน เช่น โปรแกรมต้องแสดงผลลัพธ์ภายใน 1 วินาที หรืออินเทอร์เฟซของระบบควรเข้าใจได้ทันทีโดยไม่ต้องมีคำอธิบายเพิ่มเติม

ขั้นตอนต่อมาคือการสร้าง Flowchart เพื่อแสดงกระบวนการทำงานของโปรแกรม เช่น การรับค่าจากผู้ใช้ การคำนวณ และการแสดงผล การใช้ Flowchart จะช่วยให้เห็นภาพรวมของการประมวลผลและลดความผิดพลาดในการเขียนโปรแกรม

สุดท้ายคือการตรวจสอบความถูกต้องของข้อกำหนด โดยอาจทำงานร่วมกับผู้ใช้งานหรือผู้ที่เกี่ยวข้องเพื่อยืนยันว่าโปรแกรมที่พัฒนาจะตอบโจทย์ได้ตามความต้องการ การปรับปรุงข้อกำหนดในขั้นตอนนี้ช่วยลดปัญหาที่อาจเกิดขึ้นและทำให้โปรแกรมเหมาะสมสำหรับผู้เริ่มต้นศึกษา

ตัวอย่างกรณีศึกษา

- กรณีศึกษา 1 โปรแกรมคำนวณพื้นที่และปริมาตรของรูปทรงเรขาคณิต
 - การระบุปัญหา นักเรียนมักประสบปัญหาในการคำนวณค่าพื้นที่และปริมาตรของรูปทรง เช่น ลูกบาศก์ ทรงกลม หรือทรงกรวย ซึ่งอาจเกิดข้อผิดพลาดทางคณิตศาสตร์
 - ข้อกำหนดของระบบ โปรแกรมต้องสามารถรับค่าพารามิเตอร์ เช่น ความยาว รัศมี หรือความสูง จากผู้ใช้งาน และคำนวณพื้นที่หรือปริมาตรของรูปทรงเหล่านั้น พร้อมแสดงผลลัพธ์ในรูปแบบที่เข้าใจง่าย
- กรณีศึกษา 2 โปรแกรมแก้สมการเชิงเส้น

- การระบุปัญหา วิศวกรและนักเรียนต้องแก้สมการเชิงเส้นหลายตัวแปรด้วยมือ ซึ่งใช้เวลานาน และอาจเกิดข้อผิดพลาด
- ข้อกำหนดของระบบ โปรแกรมต้องสามารถรับค่าสัมประสิทธิ์และค่าคงที่ของสมการจากผู้ใช้งาน จากนั้นคำนวณคำตอบและแสดงผลลัพธ์อย่างแม่นยำ
- กรณีศึกษา 3 โปรแกรมคำนวณแรงในระบบกลศาสตร์
 - การระบุปัญหา นักศึกษาวิศวกรรมศาสตร์ต้องคำนวณแรงและโมเมนต์ในระบบกลศาสตร์ ซึ่งต้องใช้เวลามากในการคำนวณด้วยมือ
 - ข้อกำหนดของระบบ โปรแกรมต้องสามารถรับข้อมูลเกี่ยวกับแรง ระยะทาง และมุม จากนั้นคำนวณแรงสูตรหรือโมเมนต์และแสดงผลลัพธ์ในรูปแบบตัวเลขและการพิกอย่างง่าย

5.2.2 การเก็บรวบรวมข้อมูลและวิเคราะห์ข้อมูล

การเก็บข้อมูลจากผู้ใช้งานหรือแหล่งข้อมูลอื่น

การเก็บข้อมูลเป็นขั้นตอนสำคัญในการพัฒนาโปรแกรม โดยเฉพาะสำหรับผู้เริ่มต้นศึกษาและผู้ฝึกเขียนโปรแกรมเบื้องต้น การเก็บข้อมูลเริ่มต้นจากการระบุว่าใครคือผู้ใช้งานหลักของโปรแกรม และข้อมูลใดที่จำเป็นต่อการพัฒนา ตัวอย่างเช่น หากต้องการพัฒนาโปรแกรมคำนวณพื้นที่ของรูปทรงเรขาคณิต ผู้พัฒนาควรสอบถามผู้ใช้งานเกี่ยวกับรูปทรงที่ต้องการคำนวณ เช่น ลูกบาศก์ ทรงกลม หรือทรงกระบอก และพารามิเตอร์ที่เกี่ยวข้อง เช่น ความยาว รัศมี หรือความสูง

นอกจากนี้ อาจมีการเก็บข้อมูลจากแหล่งข้อมูลอื่น เช่น เอกสารทางวิชาการ เว็บไซต์ หรือฐานข้อมูลที่เกี่ยวข้อง เพื่อช่วยสร้างความเข้าใจในปัญหาที่ต้องการแก้ไข ข้อมูลเหล่านี้จะช่วยสนับสนุนการออกแบบโปรแกรมที่ตอบโจทย์ความต้องการของผู้ใช้งาน

การวิเคราะห์ข้อมูลเพื่อพัฒนาแนวทางแก้ปัญหา

หลังจากเก็บรวบรวมข้อมูลแล้ว ข้อมูลเหล่านี้จะถูกนำมาวิเคราะห์เพื่อสร้างแนวทางแก้ปัญหา ขั้นตอนนี้เริ่มต้นจากการจัดหมวดหมู่ข้อมูลที่เก็บมา เช่น ข้อมูลที่เกี่ยวข้องกับการคำนวณ ข้อมูลที่เกี่ยวข้องกับการแสดงผล หรือข้อมูลที่เกี่ยวข้องกับการป้อนค่าจากผู้ใช้งาน จากนั้น ผู้พัฒนาจะทำการสรุปความต้องการของระบบและแปลงความต้องการเหล่านี้เป็นข้อกำหนดที่ชัดเจน

ตัวอย่างการวิเคราะห์ข้อมูลสำหรับปัญหานำเด็ก

- โปรแกรมคำนวณพื้นที่และปริมาตร วิเคราะห์ข้อมูลรูปทรงที่ต้องการคำนวณ และกำหนดสูตรทางคณิตศาสตร์ที่เหมาะสมสำหรับแต่ละรูปทรง
- โปรแกรมแก้สมการเชิงเส้น วิเคราะห์จำนวนตัวแปรที่โปรแกรมต้องรองรับ และกำหนดขั้นตอนวิธี (Algorithm) ที่จะใช้แก้สมการ เช่น การใช้เมทริกซ์
- โปรแกรมคำนวณแรงในระบบกลศาสตร์ วิเคราะห์ประเภทของแรง เช่น แรงดัน แรงเสียดทาน และโมเมนต์ และกำหนดวิธีการคำนวณที่เหมาะสม เช่น การใช้สมการแรงสมดุล

5.2.3 การออกแบบแนวการทำงานแก้ปัญหาเบื้องต้น

การวางแผนลำดับขั้นตอนการแก้ปัญหา การวางแผนลำดับขั้นตอนการแก้ปัญหาเป็นกระบวนการสำคัญในการพัฒนาโปรแกรม โดยเริ่มจากการแยกปัญหาใหญ่ออกเป็นส่วนย่อย ๆ เพื่อให้สามารถจัดการและพัฒนาได้ง่ายขึ้น ตัวอย่างเช่น ในการพัฒนาโปรแกรมคำนวณค่าพื้นที่ของรูปทรงเรขาคณิต ขั้นตอนการวางแผนอาจประกอบด้วย

1. กำหนดประเภทของรูปทรงที่ต้องการคำนวณ เช่น ลูกบาศก์ ทรงกลม หรือทรงกระบอก
2. รับข้อมูลที่จำเป็นจากผู้ใช้งาน เช่น ความยาว รัศมี หรือความสูง
3. คำนวณพื้นที่หรือปริมาตรของรูปทรงโดยใช้สูตรที่เหมาะสม
4. แสดงผลลัพธ์ในรูปแบบที่เข้าใจง่าย เช่น ตัวเลขหรือข้อความที่ชัดเจน

การเขียนลำดับขั้นตอนสามารถทำได้ในรูปแบบ Pseudocode หรือ Flowchart ซึ่งช่วยให้ผู้พัฒนาเห็นภาพรวมของกระบวนการและลดข้อผิดพลาดในการเขียนโค้ดจริง

การเลือกเครื่องมือและเทคนิคที่เหมาะสม

หลังจากการวางแผนลำดับขั้นตอนแล้ว ขั้นตอนต่อมาคือการเลือกเครื่องมือและเทคนิคที่เหมาะสมสำหรับการพัฒนาโปรแกรม ผู้เริ่มต้นควรเลือกใช้เครื่องมือที่เข้าใจง่ายและสอดคล้องกับวัตถุประสงค์ของโปรแกรม เช่น การใช้ภาษา Python สำหรับการคำนวณทางคณิตศาสตร์ การสร้าง Flowchart ด้วย Microsoft Visio เพื่อวางแผนร่าง และการใช้แนวคิดการเขียนโปรแกรมเชิงโครงสร้าง โดยแบ่งโปรแกรมเป็นส่วนย่อยที่ชัดเจน เช่น การรับค่าข้อมูล การประมวลผล และการแสดงผลลัพธ์

ผู้แต่งมองว่าการเริ่มต้นพัฒนาโปรแกรมจากการ "เข้าใจปัญหา" และ "วางแผนเป็น" ที่ชัดเจนเป็นเรื่องสำคัญมาก เพราะหากมองข้ามขั้นตอนนี้ไป อาจทำให้โปรแกรมที่พัฒนาออกมาไม่ตอบโจทย์ความต้องการจริง ๆ ของผู้ใช้งาน นอกจากนี้ การเขียนข้อกำหนดอย่างละเอียดตั้งแต่แรกยังช่วยลดการแก้ไขข้อผิดพลาดในภายหลัง ซึ่งเป็นสิ่งที่เกิดขึ้นบ่อยในโปรเจกต์จริง

5.3 การออกแบบโปรแกรมแบบจำกบลงล่าง

เป็นกระบวนการวางแผนและพัฒนาโปรแกรมโดยการแบ่งปัญหาใหญ่ออกเป็นส่วนย่อย ๆ ที่ง่ายต่อการจัดการและแก้ไข วิธีนี้ช่วยให้การเขียนโปรแกรมมีโครงสร้างชัดเจนและเป็นระบบ ลดความซับซ้อนของปัญหา และช่วยให้นักพัฒนาสามารถจัดลำดับขั้นตอนการแก้ปัญหาได้อย่างมีประสิทธิภาพ โดยเริ่มจากการมองภาพรวมของระบบ และค่อย ๆ แยกย่อยเป็นรายละเอียดในแต่ละขั้นตอนเพื่อพัฒนาโปรแกรมให้สมบูรณ์ และตรวจสอบได้ง่ายในภายหลัง

5.3.1 การแบ่งปัญหาออกเป็นส่วนย่อย

แบ่งปัญหาใหญ่ออกเป็นส่วนย่อย หรือ "Decomposition" เป็นแนวทางสำคัญใน Top-Down Design ซึ่งช่วยให้การพัฒนาโปรแกรมมีโครงสร้างที่ชัดเจน และง่ายต่อการจัดการและแก้ไขในภายหลัง โดยแนวคิดนี้สามารถนำไปใช้กับปัญหาทั่วไปในชีวิตประจำวันหรือการเขียนโปรแกรมได้

หลักการของการแบ่งปัญหา

- แยกปัญหาออกเป็นงานย่อย แต่ละงานควรมีขอบเขตชัดเจนและสามารถจัดการได้ง่าย
- สร้างลำดับขั้นตอนที่มีโครงสร้าง: เมื่อแต่ละส่วนทำงานเสร็จสมบูรณ์ สามารถประกอบเข้าด้วยกันเพื่อสร้างโปรแกรมที่สมบูรณ์
- มุ่งเน้นที่การแก้ปัญหาย่อยทีละส่วน ลดความซับซ้อนของปัญหาใหญ่ให้เหลือปัญหาย่อยที่เข้าใจง่ายกว่า

โดยตัวอย่างกับปัญหาเกมทายตัวเลขในบทที่ 3 หัวข้อ 3.4.4 ดังนี้

- การเริ่มต้นโปรแกรม สุ่มตัวเลขลับ (secretNumber) จากช่วง 1 ถึง 20 และแสดงข้อความแนะนำเกมให้ผู้เล่น
- การรับข้อมูลจากผู้เล่น รับค่าที่ผู้เล่นทายในแต่ละรอบ
- การตรวจสอบเงื่อนไข เปรียบเทียบค่าที่ผู้เล่นทายกับตัวเลขลับ
 - หากทายต่ำเกินไป แจ้งผู้เล่นว่า "ต่ำเกินไป"
 - หากทายสูงเกินไป แจ้งผู้เล่นว่า "สูงเกินไป"
 - หากทายถูก ออกจากลูปทันที
- การจบเกม และแสดงผลลัพธ์
 - หากทายถูก แสดงข้อความยินดีพร้อมจำนวนครั้งที่ใช้
 - หากทายไม่ถูกภายในรอบที่กำหนด แจ้งหมายเลขอีกครั้งและจบเกม

5.3.2 การจัดลำดับขั้นตอนการแก้ปัญหา

การจัดลำดับขั้นตอนการแก้ปัญหา (Stepwise Refinement) เป็นขั้นตอนสำคัญในกระบวนการ Top-Down Design เพื่อให้โปรแกรมทำงานได้อย่างถูกต้องตามลำดับที่กำหนดไว้ การกำหนดลำดับนี้ช่วยให้การทำงานมีความเป็นระบบและง่ายต่อการตรวจสอบ

หลักการของการจัดลำดับขั้นตอนการแก้ปัญหา

- ลำดับที่ชัดเจนและเป็นเหตุเป็นผล กำหนดขั้นตอนที่มีลำดับชัดเจน เรียงจากขั้นตอนเริ่มต้นไปจนถึงการจบโปรแกรม
- ลำดับที่ครอบคลุมทุกรถี ควรตรวจสอบเงื่อนไขทุกประการ เพื่อป้องกันข้อผิดพลาดที่อาจเกิดขึ้น
- เน้นการวนซ้ำที่มีขอบเขต (Loops) ลดความซับซ้อนของลำดับการทำงานด้วยการใช้ลูปหรือโครงสร้างควบคุมที่เหมาะสม

โดยสามารถจัดลำดับขั้นตอนจากตัวอย่างเกมทายตัวเลขใน หัวข้อ 3.4.4 ได้ดังนี้

1. การเริ่มต้นโปรแกรม

ลำดับที่ 1 สุ่มตัวเลขลับ (secretNumber) โดยใช้ค่าช่วงระหว่าง 1 ถึง 20

ลำดับที่ 2 แสดงข้อความแนะนำเพื่อแจ้งผู้เล่นเกี่ยวกับติกาของเกม เช่น จำนวนครั้งที่สามารถพยายามได้ และช่วงของตัวเลขที่ต้องเดา

2. การกำหนดลูปสำหรับการหา

ลำดับที่ 3 ตั้งค่าลูป for เพื่อให้ผู้เล่นสามารถหาได้สูงสุด 7 ครั้ง

ลำดับที่ 4 ในแต่ละครอบของลูป ให้รับค่าหมายเลขจากผู้เล่นผ่าน input()

3. การตรวจสอบค่าที่ผู้เล่นหา

ลำดับที่ 5 เปรียบเทียบค่าที่ผู้เล่นหา (guess) กับตัวเลขลับ (secretNumber)

- หากตัวเลขต่ำเกินไป (guess < secretNumber) แสดงข้อความ "ต่ำเกินไป"
- หากตัวเลขสูงเกินไป (guess > secretNumber) แสดงข้อความ "สูงเกินไป"
- หากตัวเลขถูกต้อง (guess == secretNumber) แสดงข้อความยินดีและหยุดลูปทันที

4. การตรวจสอบผลลัพธ์หลังจบลูป

ลำดับที่ 6 หลังจากลูปสิ้นสุด ให้ตรวจสอบว่าผู้เล่นสามารถหาถูกหรือไม่

- หากหาถูก แสดงข้อความยินดีและแจ้งจำนวนครั้งที่ใช้
- หากหาไม่ถูก เปิดเผยตัวเลขลับที่โปรแกรมสุ่มไว้ และแจ้งว่าจบเกม

5. การจบการทำงานของเกม

ลำดับที่ 7 โปรแกรมจบการทำงาน โดยแสดงข้อความปิดท้ายหรือปล่อยให้โปรแกรมสิ้นสุด

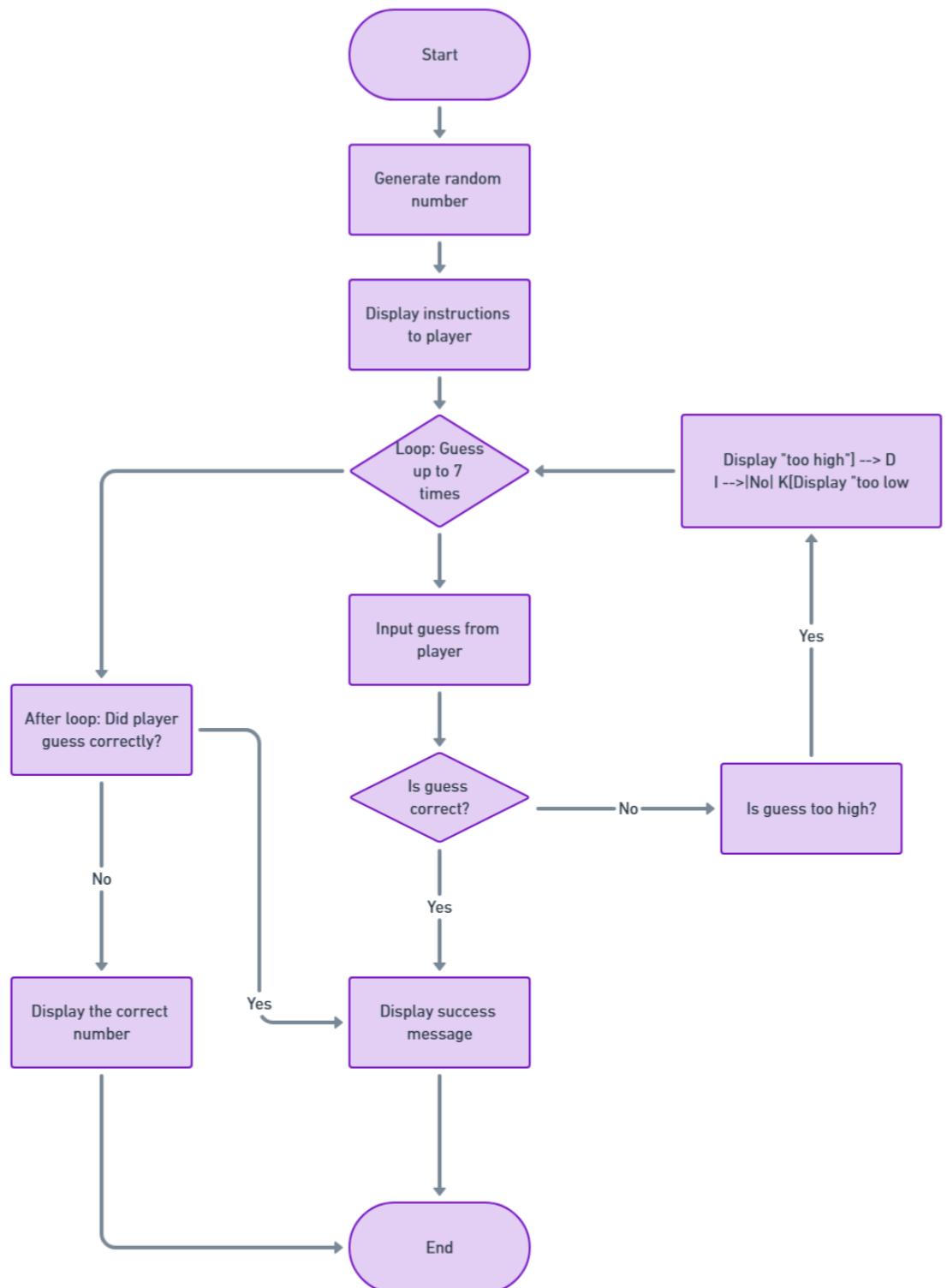
5.3.3 การสร้างผังงานเพื่อแสดงลำดับขั้นตอนของโปรแกรม

จากหัวข้อที่ 3.1 การสร้างผังงาน (Flowchart) เป็นกระบวนการที่ช่วยให้เข้าใจลำดับขั้นตอนของโปรแกรมได้ง่ายขึ้น โดยแสดงโครงสร้างการทำงานในรูปแบบแผนภาพที่ชัดเจนและเข้าใจง่าย ผังงานจะใช้สัญลักษณ์มาตรฐานในการแสดงขั้นตอนต่าง ๆ เช่น การเริ่มต้น การรับข้อมูล การตัดสินใจ และการจบโปรแกรม

อธิบาย Flowchart

1. [Start] จุดเริ่มต้นของการทำงานทั้งหมด เป็นการเริ่มต้นโปรแกรมเมามาเดาเลข
2. [Generate random number] ระบบจะสุ่มตัวเลข (random number) ซึ่งเป็นตัวเลขที่ผู้เล่นต้องเดา ตัวเลขนี้จะถูกเก็บไว้ในระบบ และผู้เล่นจะต้องพยายามเดาให้ถูกต้อง
3. [Display instructions to player] แสดงคำอธิบายติกาของเกมให้ผู้เล่นทราบ เช่น เดาว่าตัวเลขเป็นอะไร มีโอกาสเดาได้สูงสุด 7 ครั้ง
4. [Loop: Guess up to 7 times] เริ่มลูปของการเล่น โดยผู้เล่นสามารถเดาได้สูงสุด 7 ครั้ง หากครบ 7 ครั้งแล้ว ยังเดาไม่ถูก ลูปจะสิ้นสุด

5. [Input guess from player] ผู้เล่นจะใส่คำตอบหรือตัวเลขที่คิดว่าเป็นคำตอบที่ถูกต้อง
6. [Is guess correct?] ระบบจะตรวจสอบว่าคำตอบที่ผู้เล่นใส่มาถูกต้องหรือไม่:
 - Yes (ใช่): หากคำตอบถูกต้อง ระบบจะไปยัง [Display success message]
 - No (ไม่): หากคำตอบไม่ถูกต้อง ระบบจะไปตรวจสอบต่อที่ [Is guess too high?]
7. [Is guess too high?] ระบบตรวจสอบว่าคำตอบของผู้เล่นมากเกินไปหรือไม่:
 - Yes (ใช่): แสดงข้อความว่า "ตัวเลขสูงเกินไป" และวนกลับไปลูปให้ผู้เล่นเดาใหม่
 - No (ไม่): ระบบจะไปที่ [Display "too low"]



รูปที่ 5.1 Flowchart พังก์ชันการทำงานในโปรแกรมไทยตัวเลข

8. [Display "too low"] หากตัวเลขที่ผู้เล่นเดามาน้อยเกินไป ระบบจะแสดงข้อความว่า "ตัวเลขต่ำเกินไป" หลังจากนั้นจะวนกลับไปที่ลูป ให้ผู้เล่นเดาใหม่

9. [After loop: Did player guess correctly?] เมื่อคุณปุ่มบลัง (หลังจากเดาครบ 7 ครั้ง) ระบบจะตรวจสอบอีกครั้งว่าผู้เล่นเดาถูกต้องหรือไม่:

- Yes (ใช่): แสดงข้อความสำเร็จ [Display success message]
- No (ไม่): หากยังผิดอยู่ จะไปแสดงคำตอบที่ถูกต้อง [Display the correct number]

10. [Display success message]

หากผู้เล่นเดาถูกต้องในลูป ระบบจะแสดงข้อความยินดี เช่น "คุณเดาถูกต้องแล้ว!"

11. [Display the correct number] หากผู้เล่นไม่สามารถเดาได้ถูกต้องภายใน 7 ครั้ง ระบบจะแสดงคำตอบที่ถูกต้อง เช่น "คุณแพ้! ตัวเลขที่ถูกต้องคือ X"

12. [End] สิ้นสุดกระบวนการของเกม ไม่ว่าจะชนะหรือแพ้

5.3.4 การเขียนโค้ดจากแบบจำลอง

คือการเขียนลำดับขั้นตอนของโปรแกรมในรูปแบบกึ่งรหัส (Semi-code) ที่ผสมผสานระหว่างภาษามนุษย์และโครงสร้างของโปรแกรมที่เรียกว่า Pseudocode เป็นวิธีที่ช่วยอธิบายกระบวนการทำงานของโปรแกรมอย่างเข้าใจง่าย โดยไม่ต้องยึดติดกับไวยากรณ์เฉพาะของภาษาโปรแกรมใด ๆ

ประโยชน์ของ Pseudocode

- ช่วยวางแผนและออกแบบโครงสร้างโปรแกรมก่อนการเขียนโค้ดจริง
- ทำให้มีพัฒนาเข้าใจแนวคิดและขั้นตอนการทำงานร่วมกันได้ง่ายขึ้น
- ลดความผิดพลาดในการเขียนโปรแกรมโดยการทบทวนตรวจสอบล่วงหน้า
- สามารถแปลงเป็นโค้ดโปรแกรมจริงได้ง่ายในภาษาที่ต้องการ

วิธีการเขียน Pseudocode

1. ระบุปัญหาและเป้าหมายของโปรแกรม ริมต้นด้วยการเข้าใจว่าโปรแกรมต้องการทำอะไร และผลลัพธ์ที่คาดหวังคืออะไร เช่น การแก้ปัญหา หรือการประมวลผลข้อมูล

2. กำหนดโครงสร้างและลำดับขั้นตอนการทำงาน แบ่งการทำงานออกเป็นลำดับขั้นตอนที่ชัดเจนและเป็นเหตุเป็นผล โดยเริ่มจากจุดเริ่มต้น (Start) ไปจนถึงจุดสิ้นสุด (End) ใช้ลำดับที่มีตระกูล เช่น การแสดงข้อความ การรับข้อมูล การประมวลผล และการแสดงผล

3. ใช้คำที่อ่านง่ายและเป็นธรรมชาติ ใช้ภาษาเรียบง่าย เช่น "Input" "Display" "Calculate" หรือ "Check" เพื่อให้เข้าใจได้ง่าย เช่น Input age from user หรือ Display result to screen

4. ใช้โครงสร้างควบคุมที่เข้าใจง่าย หากต้องมีการตัดสินใจ (Decision) ใช้คำเช่น IF ELSE หรือ WHILE เพื่อแสดงการทำงานตามเงื่อนไข ตัวอย่างดังนี้

```
IF score >= 50 THEN
    Display "Pass"
```

```
ELSE
```

```
    Display "Fail"
```

5. แยกส่วนการทำงานออกเป็นโมดูลหรือฟังก์ชัน (ถ้าจำเป็น) หากโปรแกรมมีความซับซ้อน สามารถแบ่งงานออกเป็นส่วนย่อย ๆ และใช้ฟังก์ชัน เช่น Call CalculateTax() เพื่อทำให้ง่ายต่อการอ่าน

6. ตรวจสอบความถูกต้องและสมบูรณ์ของตรรกะ ทบทวน Pseudocode ว่าครอบคลุมทุกเงื่อนไข หรือสถานการณ์หรือไม่ เช่น กรณิช้อมูลผิดพลาด

ตัวอย่าง Pseudocode แบบง่าย

1. Start
2. Input two numbers from the user
3. Calculate the sum of the two numbers
4. Display the result
5. End

ตัวอย่าง Pseudocode ที่มีเงื่อนไขและลูป ดังของโปรแกรม tahyตัวเลข

1. Start
2. Generate random_number between 1 and 100
3. Display "Welcome to the Guessing Game!"
4. Display "You have 7 attempts to guess the correct number."
5. Set attempts = 7
6. WHILE attempts > 0:
 7. Input guess from player
 8. IF guess == random_number THEN:


```
Display "Congratulations! You guessed the correct number."
          Exit loop
```
 9. ELSE:


```
IF guess > random_number THEN:
            Display "Too high! Try again."
          ELSE:
            Display "Too low! Try again."
```
 10. Decrease attempts by 1
 11. IF attempts == 0 THEN:


```
Display "Sorry, you've used all your attempts."
          Display "The correct number was: " + random_number
```
- 12 End

แนวคิดการออกแบบโปรแกรมแบบจากบนลงล่าง ทำให้มองเห็นภาพรวมของโปรแกรมก่อนเริ่มลงมือเขียนโค้ดจริง ๆ และช่วยให้การพัฒนาเป็นไปอย่างมีลำดับขั้นตอน ไม่กระโดดข้ามไปมาหรือเขียนแบบ "แก้ไปทีละจุด" โดยเฉพาะเมื่อทำโปรเจกต์ที่ซับซ้อน เช่น เกม หรือโปรแกรมคำนวณต่าง ๆ การออกแบบที่ดีตั้งแต่ต้นจะช่วยลดเวลาการแก้ไขดับกพร่อง (debug) และแก้ไขโค้ดได้มาก

5.4 การพัฒนาโปรแกรมเพื่อแก้ปัญหา

ในขั้นตอนการพัฒนาโปรแกรมเพื่อแก้ปัญหา การทำงานจะเริ่มจากการรับข้อมูลจากผู้ใช้ การประมวลผลข้อมูลตามข้อกำหนด การแสดงผลลัพธ์ในรูปแบบที่เข้าใจง่าย และการตรวจสอบเพื่อปรับปรุงคุณภาพโปรแกรม ต่อไปนี้เป็นคำอธิบายและตัวอย่างที่เกี่ยวข้อง

5.4.1 การรับข้อมูลจากผู้ใช้

การรับข้อมูลจากผู้ใช้เป็นจุดเริ่มต้นสำคัญของการพัฒนาโปรแกรม โดยมีเป้าหมายเพื่อเก็บข้อมูลที่จำเป็นสำหรับการประมวลผล

- ส่วนติดตอแบบข้อความ (Text-based Interface): ใช้คำสั่งเช่น `input()` ใน Python เพื่อรับค่าข้อมูลจากผู้ใช้ ดังตัวอย่างโค้ดนี้

```
# รับค่าจากผู้ใช้
name = input("กรุณากรอกชื่อของคุณ: ")
age = int(input("กรุณากรอกอายุของคุณ: "))
print(f"สวัสดี {name}, อายุ {age} ปี")
```

ผลลัพธ์

```
กรุณากรอกชื่อของคุณ: อลิส
กรุณากรอกอายุของคุณ: 25
สวัสดี อลิส, อายุ 25 ปี
```

โปรแกรมรับข้อมูลชื่อและอายุจากผู้ใช้ โดยแสดงข้อความเพื่อให้ผู้ใช้ป้อนข้อมูล เช่น "อลิส" และ "25" ตามลำดับ จากนั้นแสดงผลลัพธ์: "สวัสดี อลิส, อายุ 25 ปี"

- ส่วนติดตอแบบกราฟิก (GUI) ใช้เครื่องมือ GUI เช่น Tkinter (Python) หรือ Java Swing เพื่อสร้างฟอร์มให้ผู้ใช้กรอกข้อมูล ตัวอย่างใน Python ดังตัวอย่างโค้ดนี้

```
import tkinter as tk

def submit():
    name = entry_name.get()
    age = entry_age.get()
    print(f"ชื่อ: {name}, อายุ: {age}")

root = tk.Tk()
tk.Label(root, text="ชื่อ").pack()
entry_name = tk.Entry(root)
entry_name.pack()
tk.Label(root, text="อายุ").pack()
entry_age = tk.Entry(root)
entry_age.pack()
tk.Button(root, text="บันทึก", command=submit).pack()
root.mainloop()
```

ผลลัพธ์



5.4.2 การประมวลผลข้อมูลตามข้อกำหนด

ขั้นตอนนี้เกี่ยวกับการคำนวณหรือการดำเนินการตามต้องการของโปรแกรม โดยใช้ฟังก์ชัน อัลกอริทึม และโครงสร้างควบคุม (Control Structures)

```
def calculate_bmi(weight, height):
    return weight / (height ** 2)

weight = float(input("กรอกน้ำหนัก(kg): "))
height = float(input("กรอกส่วนสูง(m): "))
bmi = calculate_bmi(weight, height)
print(f"BMI ของคุณคือ {bmi:.2f}")
```

ผลลัพธ์

กรอกน้ำหนัก (kg): 70

กรอกส่วนสูง (m): 1.75

BMI ของคุณคือ 22.86

โปรแกรมรับข้อมูลน้ำหนักและส่วนสูงจากผู้ใช้ เช่น น้ำหนัก "70" และส่วนสูง "1.75" เมตร จากนั้น คำนวณค่า BMI ด้วยสูตร $\text{weight} / (\text{height} ** 2)$ ซึ่งได้ผลลัพธ์เป็น 22.857142857142858 และแสดงผลลัพธ์ในรูปแบบทศนิยม 2 ตำแหน่งว่า "BMI ของคุณคือ 22.86"

5.4.3 การแสดงผลลัพธ์ที่เหมาะสม

การแสดงผลลัพธ์ให้ผู้ใช้ในรูปแบบที่เข้าใจง่าย เป็นขั้นตอนสุดท้ายที่สำคัญ โปรแกรมที่ดีควรมีรูปแบบการแสดงผลที่ชัดเจนและใช้งานสะดวก

การแสดงผลลัพธ์เป็นข้อความ หากผลลัพธ์เป็นตัวเลข ให้จัดรูปแบบ เช่น ตำแหน่งทศนิยม

```
print(f"ผลลัพธ์: {bmi:.2f} (BMI)")
```

การแสดงผลลัพธ์แบบกราฟิก: ใช้ Matplotlib ใน Python

```
import matplotlib.pyplot as plt
```

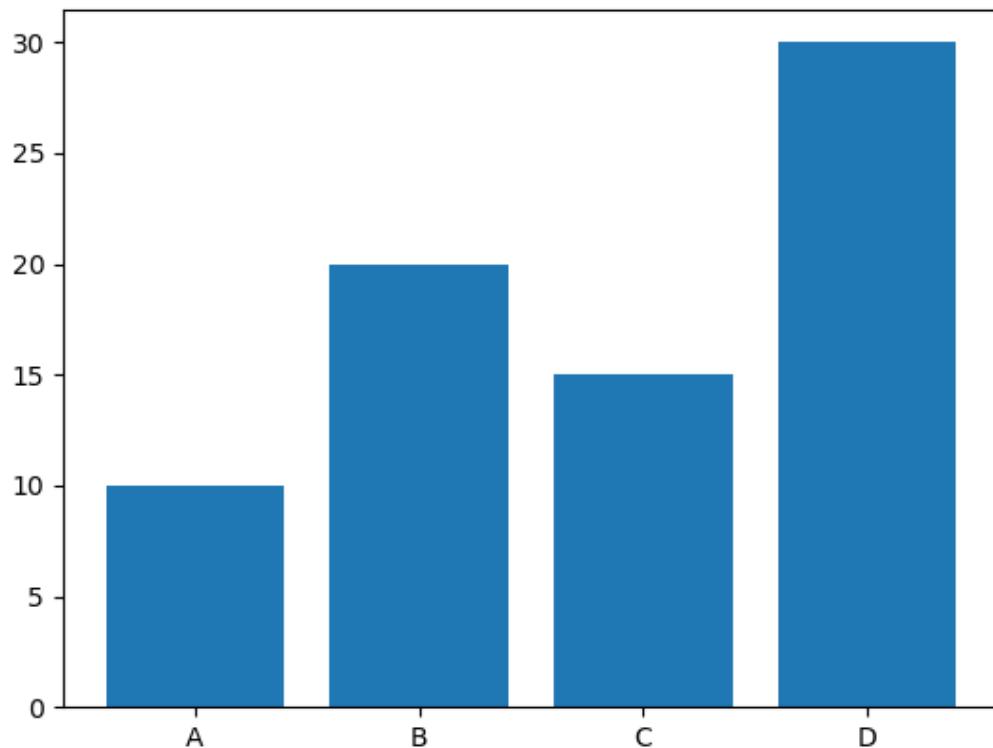
```
data = [10, 20, 15, 30]
```

```
labels = ['A', 'B', 'C', 'D']
plt.bar(labels, data)
plt.title('Bar Chart Example')
plt.show()
```

ผลลัพธ์



Bar Chart Example



5.4.4 การทดสอบและแก้ไขข้อผิดพลาดของโปรแกรม

การทดสอบโปรแกรมช่วยระบุข้อผิดพลาด (Bugs) และปรับปรุงคุณภาพ

- การทดสอบด้วยกรณีตัวอย่าง (Test Cases) เช่น Test Cases เพื่อครอบคลุมทุกสถานการณ์
- การแก้ไขข้อผิดพลาด ใช้เครื่องมือ Debugger หรือ print() เพื่อตรวจสอบค่าในแต่ละขั้นตอน
- ตัวอย่างการปรับปรุง หากโปรแกรมรับตัวเลขติดลบจากผู้ใช้ ควรเพิ่มการตรวจสอบก่อน

ประเมินผล

```
weight = float(input("กรอกน้ำหนัก(kg): "))
while weight <= 0:
    print("น้ำหนักต้องมากกว่า 0")
    weight = float(input("กรอกน้ำหนัก(kg): "))
```

ในการเขียนโปรแกรมจริง ผู้แต่งมีความเชื่อว่าการแบ่งงานออกเป็นขั้นตอนย่อย เช่น การรับข้อมูล การประมวลผล และการแสดงผล ช่วยให้ได้ดู "เป็นระบบ" และ "อ่านง่าย" มากขึ้น และเมื่อสมกับการใช้พังก์ชันย่อย ๆ หรือโมดูล ก็จะช่วยลดการเขียนโค้ดซ้ำซ้อน เหมาะกับการพัฒนาโปรแกรมที่ต้องมีการอัปเดต หรือปรับปรุงในอนาคต

5.5 ตัวอย่างและการประยุกต์ใช้งาน

หัวข้อนี้ศึกษาตัวอย่างการพัฒนาโปรแกรมด้วยแนวคิด Top-Down Design ซึ่งเป็นกระบวนการแบ่งปัญหาใหญ่ออกเป็นส่วนย่อย เพื่อให้ง่ายต่อการวางแผนและพัฒนาโปรแกรม โดยครอบคลุมหลากหลายแง่มุม ตั้งแต่ปัญหาทางคณิตศาสตร์ การคำนวนค่าใช้จ่ายในชีวิตประจำวัน ไปจนถึงโปรแกรมเฉพาะทางในอุตสาหกรรม ตัวอย่างในหัวข้อนี้ประกอบด้วย

1. โปรแกรมแก้สมการกำลังสอง (Quadratic Equation Solver) ตัวอย่างโปรแกรมที่ช่วยคำนวนรากของสมการกำลังสอง โดยใช้สูตรทางคณิตศาสตร์และตรวจสอบผลลัพธ์ทั้งกรณีค่าตอบจริงและคำตอบเชิงซ้อน พร้อมแสดงผลลัพธ์ในรูปแบบที่เข้าใจง่าย

2. โปรแกรมคำนวนค่าไฟฟ้ารายเดือน การคำนวนค่าใช้จ่ายไฟฟ้าแบบขั้นบันไดที่อิงกับการใช้งานในชีวิตประจำวัน โดยใช้เงื่อนไขการตรวจสอบและการคำนวนที่เหมาะสม เพื่อแสดงค่าใช้จ่ายรวมได้อย่างแม่นยำ

3. โปรแกรมคำนวนค่าดัชนีมวลกาย (BMI) และประเมินสุขภาพ การประยุกต์ใช้โปรแกรมสำหรับอุตสาหกรรมการดูแลสุขภาพ โดยคำนวนค่า BMI และวิเคราะห์สถานะสุขภาพ เช่น น้ำหนักต่ำกว่าเกณฑ์น้ำหนักปกติ น้ำหนักเกิน หรือโรคอ้วน พร้อมแนะนำสถานะสุขภาพของผู้ใช้งาน

4. โปรแกรมเกม Tic-Tac-Toe (OX) ตัวอย่างโปรแกรมเกมเชิงตัดตอบที่สร้างความสนุกและฝึกการคิดเชิงตรรกะสำหรับผู้เล่นสองคน โดยโปรแกรมสามารถตรวจสอบผลการเล่นในทุกรอบนี้ เช่น ผู้ชนะ เกมเสมอ หรือความผิดพลาดของการป้อนข้อมูล

การนำเสนอในบทนี้จะแสดงถึงกระบวนการออกแบบโปรแกรมด้วย Pseudocode การแบ่งพังก์ชันย่อย และตัวอย่างโค้ด Python ที่สมบูรณ์ ซึ่งเหมาะสมสำหรับการเรียนรู้การพัฒนาซอฟต์แวร์ในระดับเริ่มต้นและการประยุกต์ใช้ในสถานการณ์จริง.

5.5.1 ตัวอย่างโปรแกรมการแก้สมการกำลังสอง (Quadratic Equation Solver)

ขั้นตอนการแก้ปัญหา

1. รับค่าสัมประสิทธิ์ a , b และ c จากผู้ใช้
2. คำนวนค่า Discriminant

$$D = b^2 - 4ac$$

3. ใช้ค่า D ตรวจสอบประเภทของค่าตอบ:
 - $D > 0$ มีค่าตอบจริง 2 คำตอบ

○ $D=0$ มีคำตอบจริง 1 คำตอบ

○ $D<0$ ไม่มีคำตอบจริง

4. คำนวณคำตอบและแสดงผลลัพธ์

โดยสามารถเขียนเป็น Pseudocode ได้ดังนี้

START

1. Input coefficients a , b , c from the user

 INPUT a

 INPUT b

 INPUT c

2. Check if a is not equal to 0 (since it must be a quadratic equation)

 IF $a == 0$ THEN

 PRINT "Coefficient 'a' cannot be 0"

 EXIT

 ENDIF

3. Calculate the Discriminant (D)

$D = b^2 - 4 * a * c$

4. Check the type of solution based on D:

 IF $D > 0$ THEN

$root_1 = (-b + \sqrt{D}) / (2 * a)$

$root_2 = (-b - \sqrt{D}) / (2 * a)$

 PRINT "There are 2 real solutions"

 PRINT "Root 1: $root_1$ "

 PRINT "Root 2: $root_2$ "

 ELSE IF $D == 0$ THEN

$root = -b / (2 * a)$

 PRINT "There is 1 real solution"

 PRINT "Root: $root$ "

 ELSE

 real_part = $-b / (2 * a)$

 imaginary_part = $\sqrt{-D} / (2 * a)$

 PRINT "There are no real solutions (complex roots)"

 PRINT "Root 1: $real_part + imaginary_part i$ "

 PRINT "Root 2: $real_part - imaginary_part i$ "

 ENDIF

END

สร้างฟังก์ชันย่อยในการทำงาน

1. ฟังก์ชันคำนวณค่า Discriminant

```
# พัฒนาคำนวณค่า Discriminant
```

```
def calculate_discriminant(a, b, c):
    return b**2 - 4 * a * c
```

2. พิจารณาตรวจสอบ โดยใช้ค่า Discriminant ตรวจสอบประเภทของคำตอบ และคืนค่าผลลัพธ์

```
# พิจารณาแก้สมการกำลังสอง
def solve_quadratic(a, b, c):
    discriminant = calculate_discriminant(a, b, c)
    if discriminant >= 0:
        root1 = (-b + discriminant**0.5) / (2 * a)
        root2 = (-b - discriminant**0.5) / (2 * a)
    else:
        root1 = (-b + cmath.sqrt(discriminant)) / (2 * a)
        root2 = (-b - cmath.sqrt(discriminant)) / (2 * a)
    return root1, root2
```

โปรแกรมสมบูรณ์

```
import cmath # สำหรับคำนวณกรณี D < 0
# พิจารณาคำนวณค่า Discriminant
def calculate_discriminant(a, b, c):
    return b**2 - 4 * a * c
# พิจารณาแก้สมการกำลังสอง
def solve_quadratic(a, b, c):
    discriminant = calculate_discriminant(a, b, c)
    if discriminant >= 0:
        root1 = (-b + discriminant**0.5) / (2 * a)
        root2 = (-b - discriminant**0.5) / (2 * a)
    else:
        root1 = (-b + cmath.sqrt(discriminant)) / (2 * a)
        root2 = (-b - cmath.sqrt(discriminant)) / (2 * a)
    return root1, root2
# โปรแกรมหลัก
def main():
    # รับค่าสัมประสิทธิ์จากผู้ใช้
    print("แก้สมการกำลังสองในรูปแบบ ax^2 + bx + c = 0")
    a = float(input("กรอกค่า a: "))
    b = float(input("กรอกค่า b: "))
    c = float(input("กรอกค่า c: "))
    # ตรวจสอบว่า a ต้องไม่เท่ากับ 0
    if a == 0:
        print("ค่า a ต้องไม่เป็น 0")
        return
    # คำนวณคำตอบ
    root1, root2 = solve_quadratic(a, b, c)
    # แสดงผลลัพธ์
    print(f"\nคำตอบของสมการคือ: \n根号ที่ 1: {root1}\n根号ที่ 2: {root2}")

# เรียกโปรแกรมหลัก
if __name__ == "__main__":
    main()
```

ผลลัพธ์ กรณี $D > 0$

สมการกำลังสองในรูปแบบ $ax^2 + bx + c = 0$

กรอกค่า a: 1

กรอกค่า b: -3

กรอกค่า c: 2

คำตอบของสมการคือ:

รากที่ 1: 2.0

รากที่ 2: 1.0

ผลลัพธ์ กรณี $D = 0$

แก้สมการกำลังสองในรูปแบบ $ax^2 + bx + c = 0$

กรอกค่า a: 1

กรอกค่า b: -2

กรอกค่า c: 1

คำตอบของสมการคือ:

รากที่ 1: 1.0

รากที่ 2: 1.0

ผลลัพธ์ กรณี $D < 0$

แก้สมการกำลังสองในรูปแบบ $ax^2 + bx + c = 0$

กรอกค่า a: 1

กรอกค่า b: 2

กรอกค่า c: 5

คำตอบของสมการคือ:

รากที่ 1: $(-1+2j)$

รากที่ 2: $(-1-2j)$

5.5.2 ตัวอย่างการโปรแกรมคำนวณค่าใช้จ่ายไฟฟ้าต่อเดือน

ขั้นตอนการแก้ปัญหา

- รับข้อมูลการใช้ไฟฟ้า (จำนวนหน่วยไฟฟ้าใน kWh) จากผู้ใช้

- ใช้อัตราค่าไฟฟ้าแบบขั้นบันได:

- 0–50 หน่วย 3 บาทต่อหน่วย
- 51–150 หน่วย 4 บาทต่อหน่วย
- 151+ หน่วย 5 บาทต่อหน่วย

3. คำนวณค่าไฟฟ้าโดยรวมตามอัตราดังกล่าว

4. แสดงผลค่าใช้จ่ายไฟฟ้าทั้งหมด

โดยสามารถเขียนเป็น Pseudocode ได้ดังนี้

START

1. Input the number of electricity units used (kWh) from the user
INPUT units

2. Initialize total_cost = 0

3. Calculate electricity cost based on units:

```
IF units <= 50 THEN
    total_cost = units * 3
ELSE IF units <= 150 THEN
    total_cost = (50 * 3) + ((units - 50) * 4)
ELSE
    total_cost = (50 * 3) + (100 * 4) + ((units - 150) * 5)
ENDIF
```

4. Display the total electricity cost

PRINT "Total electricity cost: total_cost Baht"

END

สร้างฟังก์ชันย่อยในการทำงาน

1. ฟังก์ชันคำนวณค่าไฟฟ้า

```
def calculate_electricity_cost(units):
    if units <= 50:
        return units * 3
    elif units <= 150:
        return (50 * 3) + ((units - 50) * 4)
    else:
        return (50 * 3) + (100 * 4) + ((units - 150) * 5)
```

โปรแกรมสมบูรณ์

```
def calculate_electricity_cost(units):
    if units <= 50:
        return units * 3
    elif units <= 150:
        return (50 * 3) + ((units - 50) * 4)
    else:
        return (50 * 3) + (100 * 4) + ((units - 150) * 5)
```

```

# พิมพ์ชั้นหลัก
def main():
    print("โปรแกรมคำนวณค่าใช้จ่ายไฟฟ้าต่อเดือน")
    units = float(input("กรอกจำนวนหน่วยไฟฟ้าที่ใช้(kWh): "))

    if units < 0:
        print("กรุณากรอกจำนวนหน่วยไฟฟ้าให้มากกว่า 0")
        return

    total_cost = calculate_electricity_cost(units)
    print(f"\nค่าใช้จ่ายไฟฟ้ารวม: {total_cost:.2f} บาท")

# เรียกใช้งานโปรแกรมหลัก
if __name__ == "__main__":
    main()

# คำนวณคำตอบ
root1, root2 = solve_quadratic(a, b, c)

# แสดงผลลัพธ์
print(f"\nค่าตอบของสมการคือ: \nรากที่ 1: {root1}\nรากที่ 2: {root2}")

# เรียกโปรแกรมหลัก
if __name__ == "__main__":
    main()

```

ผลลัพธ์ กรณีหน่วยไฟฟ้า น้อยกว่าเท่ากับ 50

โปรแกรมคำนวณค่าใช้จ่ายไฟฟ้าต่อเดือน

กรอกจำนวนหน่วยไฟฟ้าที่ใช้ (kWh): 30

ค่าใช้จ่ายไฟฟ้ารวม: 90.00 บาท

ผลลัพธ์ กรณีหน่วยไฟฟ้า 51 - 150

โปรแกรมคำนวณค่าใช้จ่ายไฟฟ้าต่อเดือน

กรอกจำนวนหน่วยไฟฟ้าที่ใช้ (kWh): 120

ค่าใช้จ่ายไฟฟ้ารวม: 430.00 บาท

ผลลัพธ์ กรณีหน่วยไฟฟ้า มากกว่า 150

โปรแกรมคำนวณค่าใช้จ่ายไฟฟ้าต่อเดือน

กรอกจำนวนหน่วยไฟฟ้าที่ใช้ (kWh): 200

ค่าใช้จ่ายไฟฟ้ารวม: 950.00 บาท

5.5.3 โปรแกรมคำนวณค่าดัชนีมวลกาย (BMI) และประเมินสุขภาพในอุตสาหกรรมการดูแลสุขภาพ

วัตถุประสงค์

- รับข้อมูลส่วนสูง (Height) และน้ำหนัก (Weight) ของบุคคลจากผู้ใช้ 2. ใช้อัตราค่าไฟฟ้าแบบขั้นบันได
- ค่า BMI ด้วยสูตร

$$\text{BMI} = \frac{\text{Weight (kg)}}{\text{Height (m)}^2}$$

- ประเมินสถานะสุขภาพตามค่า BMI

- $\text{BMI} < 18.5$: น้ำหนักน้อย (Underweight)
- $18.5 \leq \text{BMI} < 24.9$: น้ำหนักปกติ (Normal weight)
- $25.0 \leq \text{BMI} < 29.9$: น้ำหนักเกิน (Overweight)
- $\text{BMI} \geq 30$: โรคอ้วน (Obesity)

โดยสามารถเขียนเป็น Pseudocode ได้ดังนี้

START

- Input height (in meters) and weight (in kilograms) from the user
 INPUT height
 INPUT weight
- Validate that height > 0 and weight > 0
 IF height ≤ 0 OR weight ≤ 0 THEN
 PRINT "Height and weight must be positive numbers"
 EXIT
 ENDIF
- Calculate BMI using the formula:
 BMI = weight / (height^2)
- Determine health status based on BMI:
 IF BMI < 18.5 THEN
 PRINT "Underweight"
 ELSE IF BMI < 24.9 THEN
 PRINT "Normal weight"
 ELSE IF BMI < 29.9 THEN
 PRINT "Overweight"
 ELSE
 PRINT "Obesity"
 ENDIF

5. Display the BMI value and health status

```
PRINT "Your BMI is BMI_value"
PRINT "Health Status: status"
```

END

สร้างฟังก์ชันย่อยในการทำงาน

1. ฟังก์ชันคำนวณ BMI

```
def calculate_bmi(weight, height):
    return weight / (height ** 2)
```

2. ฟังก์ชันประเมินสถานะสุขภาพ

```
def evaluate_health_status(bmi):
    if bmi < 18.5:
        return "Underweight"
    elif bmi < 24.9:
        return "Normal weight"
    elif bmi < 29.9:
        return "Overweight"
    else:
        return "Obesity"
```

โปรแกรมสมบูรณ์

```
def calculate_bmi(weight, height):
    return weight / (height ** 2)

# ฟังก์ชันประเมินสถานะสุขภาพ
def evaluate_health_status(bmi):
    if bmi < 18.5:
        return "Underweight"
    elif bmi < 24.9:
        return "Normal weight"
    elif bmi < 29.9:
        return "Overweight"
    else:
        return "Obesity"

# โปรแกรมหลัก
def main():
    print("โปรแกรมคำนวณค่าดัชนีมวลกาย(BMI) และประเมินสถานะสุขภาพ")

    # รับค่าจากผู้ใช้
    try:
        height = float(input("กรุณากรอกส่วนสูง(เมตร): "))
        weight = float(input("กรุณากรอกน้ำหนัก(กิโลกรัม): "))
    except ValueError:
        print("กรุณากรอกตัวเลขที่ถูกต้อง")
        return
```

```

# ตรวจสอบค่าที่ผู้ใช้ป้อน
if height <= 0 or weight <= 0:
    print("ส่วนสูงและน้ำหนักต้องเป็นค่าน้ำหนักที่มากกว่า 0")
    return
# คำนวณ BMI
bmi = calculate_bmi(weight, height)
# ประเมินสถานะสุขภาพ
status = evaluate_health_status(bmi)

# แสดงผลลัพธ์
print(f"\nค่าดัชนีมวลกาย (BMI) ของคุณคือ: {bmi:.2f}")
print(f"สถานะสุขภาพ: {status}")

# เรียกโปรแกรมหลัก
if __name__ == "__main__":
    main()

```

ผลลัพธ์ กรณี BMI < 18.5

โปรแกรมคำนวณค่าดัชนีมวลกาย (BMI) และประเมินสถานะสุขภาพ

กรุณารอกส่วนสูง (เมตร): 1.75

กรุณารอกน้ำหนัก (กิโลกรัม): 50

ค่าดัชนีมวลกาย (BMI) ของคุณคือ: 16.33

สถานะสุขภาพ: Underweight

ผลลัพธ์ กรณี $8.5 \leq \text{BMI} < 24.9$

โปรแกรมคำนวณค่าดัชนีมวลกาย (BMI) และประเมินสถานะสุขภาพ

กรุณารอกส่วนสูง (เมตร): 1.75

กรุณารอกน้ำหนัก (กิโลกรัม): 70

ค่าดัชนีมวลกาย (BMI) ของคุณคือ: 22.86

สถานะสุขภาพ: Normal weight

ผลลัพธ์ กรณี BMI มากกว่าเท่ากับ 30

โปรแกรมคำนวณค่าดัชนีมวลกาย (BMI) และประเมินสถานะสุขภาพ

กรุณารอกส่วนสูง (เมตร): 1.6

กรุณารอกน้ำหนัก (กิโลกรัม): 90

ค่าดัชนีมวลกาย (BMI) ของคุณคือ: 35.16

สถานะสุขภาพ: Obesity

5.5.4 โปรแกรมเกม Tic-Tac-Toe (OX)

พัฒนาโปรแกรม Tic-Tac-Toe ใช้หลักการ Top-Down Design โดยเริ่มจากภาพรวมของการทำงานหลัก แล้วแบ่งปัญหาออกเป็นส่วนย่อย ๆ เพื่อให้การพัฒนาง่ายต่อการจัดการ และช่วยให้โค้ดมีความชัดเจนและอ่านง่าย

โครงสร้างโปรแกรม

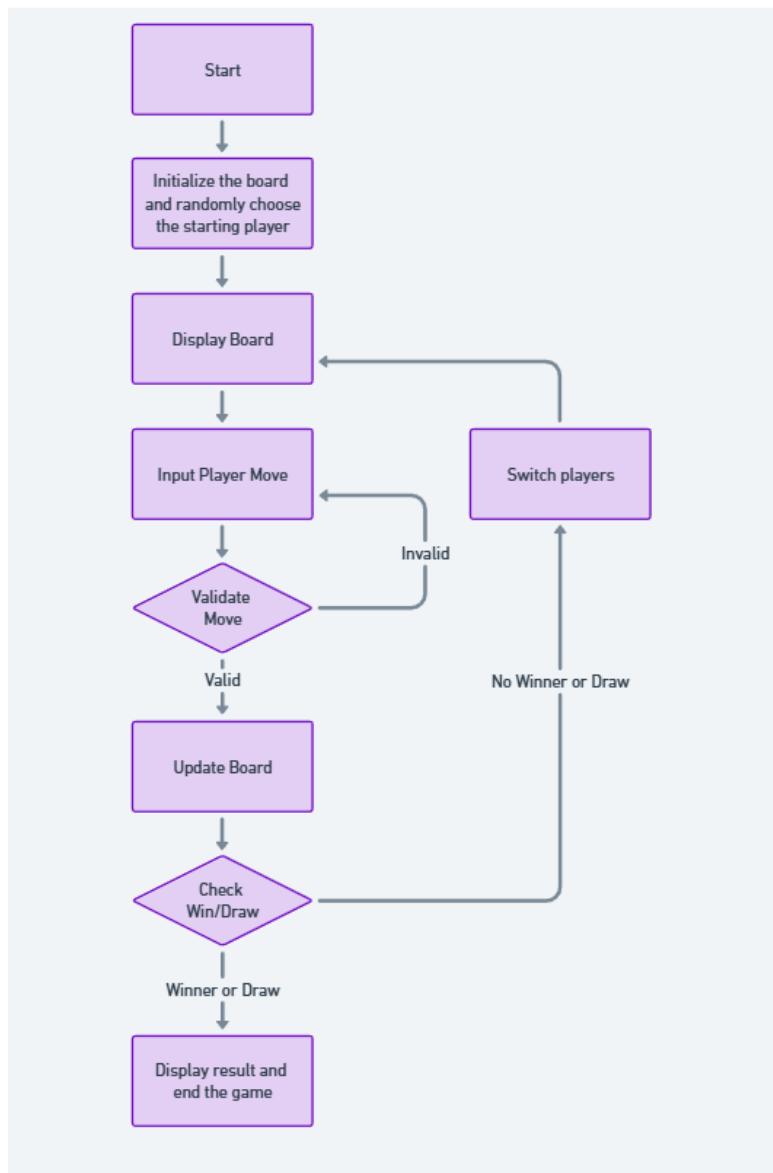
โปรแกรมแบ่งเป็นส่วนย่อยดังนี้

1. การสร้างกระดานเกม (Initialize the game board) สร้างตารางขนาด 3×3 สำหรับกระดานเกม
2. สุ่มผู้เล่นเริ่มต้น (Randomly choose the starting player) ใช้ฟังก์ชันสุ่มเพื่อเลือกว่า Player X หรือ O จะเริ่มเล่นก่อน
3. การวนลูปสำหรับแต่ละรอบเกม (Loop for the game turns) ผู้เล่นสลับกันเล่นในแต่ละรอบให้ผู้เล่นป้อนตำแหน่งที่ต้องการวางสัญลักษณ์ และอัปเดตสถานะกระดาน
4. ตรวจสอบผลลัพธ์ (Check for a win or a draw) หากผู้เล่นคนใดชนะ แสดงผลและสิ้นสุดเกม หรือหากกระดานเต็ม (ครบ 9 ตา) แต่ไม่มีผู้ชนะ ถือว่าเสมอ
5. ตรวจสอบเงื่อนไขการชนะ (Check for a win condition) ตรวจสอบว่าผู้เล่นคนใดสร้างแทริกในแนวอน แนวตั้ง หรือแนวทแยงสำเร็จหรือไม่

การออกแบบ Flowchart

แสดงกระบวนการทำงานของเกมแบบลำดับขั้นตอน โดยมีรายละเอียดดังนี้

1. Start จุดเริ่มต้นของกระบวนการ
2. Initialize the board and randomly choose the starting player เตรียมบอร์ดเกมเริ่มต้น (ว่างทั้งหมด) เลือกผู้เล่นคนแรกแบบสุ่ม
3. Display Board แสดงสถานะปัจจุบันของบอร์ดให้ผู้เล่นเห็น
4. Input Player Move ให้ผู้เล่นที่ถึงรอบของตนกรอกการเคลื่อนไหว (ระบุตำแหน่งที่ต้องการเล่น)
5. Validate Move ตรวจสอบความถูกต้องของตำแหน่งที่ผู้เล่นเลือก
 - Valid หากตำแหน่งถูกต้อง (ตำแหน่งยังว่างอยู่) ให้ไปที่ขั้นตอนถัดไป
 - Invalid หากตำแหน่งไม่ถูกต้อง (ตำแหน่งถูกใช้ไปแล้ว) ให้กลับไปที่ Input Player Move
6. Update Board บันทึกตำแหน่งของผู้เล่นในบอร์ด (เช่น ใส่เครื่องหมาย X หรือ O)
7. Check Win/Draw ตรวจสอบว่า
 - Winner ผู้เล่นคนปัจจุบันชนะหรือไม่ (เงื่อนไข เช่น มี 3 ช่องเรียงกัน)
 - Draw บอร์ดเต็มและไม่มีผู้ชนะหรือไม่
8. End or Continue หากมีผู้ชนะหรือเสมอ แสดงผลลัพธ์เกม (Winner หรือ Draw) และจบเกม หากไม่มีผู้ชนะหรือเสมอ เปลี่ยนผู้เล่น (Switch Players) และวนกลับไปที่ Display Board เพื่อเริ่มรอบใหม่
9. จบเกม กระบวนการจะวนลูปจนกว่าจะมีผู้ชนะหรือเสมอ และเกมจะสิ้นสุดลง



รูปที่ 5.2 Flowchart ฟังก์ชันการทำงานในโปรแกรมเกม Tic-Tac-Toe

Pseudocode

```

START 1. Initialize an empty 3x3 game board.
2. Randomly choose the starting player (Player X or O).
3. WHILE the game is not over:
   3.1 Display the current game board.
   3.2 Prompt the current player to input their move (row, column).
   3.3 Validate the move:
      - Ensure the input is within bounds.
      - Ensure the position is not already taken.
   3.4 Update the board with the player's move.
   3.5 Check for a win or a draw:
      - IF a player wins, display the winner and end the game.
      - IF it's a draw, display "Draw" and end the game.
   3.6 Switch to the other player.
4. END
  
```

การสร้างฟังก์ชันย่อย

1. โค้ดโครงสร้างหลัก

```
# Initialize the game board
def create_board():
    pass

# Display the board
def display_board(board):
    pass

# Check if a player has won
def check_win(board):
    pass

# Check if the board is full
def is_draw(board):
    pass

# Make a move on the board
def make_move(board, player, position):
    pass

# Randomly choose the first player
def random_start():
    pass

# Main game function
def play_game():
    # Initialize the board and first player
    while True:
        # Display the board
        # Get the player's move
        # Validate the move
        # Make the move
        # Check for a win
        # Check for a draw
        # Switch player

    # Start the game
if __name__ == "__main__":
    play_game()
```

2. การสุ่มเลือกผู้เล่นที่เริ่มต้นเกม random_start()

วัตถุประสงค์ คือ การสุ่มเลือกผู้เล่นที่เริ่มต้นเกม ("X" หรือ "O") ผลลัพธ์ คืนค่า "X" หรือ "O"

```
import random

# Randomly choose the first player
def random_start():
    return random.choice(["X", "O"])
```

การทดสอบ

```
print("Test: Random Start")
for _ in range(5):
    print(random_start())
```

ผลลัพธ์

```
Test: Random Start
```

```
X  
0  
X  
0  
X
```

1. การแสดงบอร์ด display_board(board)

วัตถุประสงค์ คือ การแสดงกระดานเกม Tic-Tac-Toe ในรูปแบบที่อ่านง่าย โดยใช้ตัวแบ่งเส้นแนวอนอน (---+---+---) เพื่อแยกแถว และแสดงตำแหน่งที่ผู้เล่นวางสัญลักษณ์ ("X" หรือ "O") บนกระดาน

```
# Display the current game board
def display_board(board):
    print("\n")
    rcount = 0
    for row in board:
        print(" " + " | ".join(row))
        if rcount < 2:
            print("----+----+----")
        rcount += 1
    print("\n")
```

- การเพิ่มบรรทัดว่างด้านบนและล่างกระดาน ใช้ `print("\n")` เพื่อเพิ่มช่องว่างก่อนแสดงกระดาน ทำให้การแสดงผลของกระดานดูมีระเบียบห่างและอ่านง่ายขึ้น
- การวนลูปเพื่อแสดงแต่ละแถวของกระดาน ใช้ `for row in board:` เพื่อวนลูปแสดงข้อมูลในแต่ละแถว โดยแต่ละตำแหน่งในแถวจะถูกเชื่อมด้วย " | " โดยใช้คำสั่ง " | ".join(row) เพื่อสร้างรูปแบบกระดานที่เหมาะสม
- การแสดงเส้นแบ่งระหว่างแถว ใช้ `if rcount < 2:` เพื่อตรวจสอบว่าแถวปัจจุบันเป็นแถวที่ 1 หรือ 2 ถ้าใช่ จะแสดงเส้นแบ่ง ---+---+--- ระหว่างแถว
- การนับจำนวนแถว ตัวแปร `rcount` ใช้สำหรับติดตามลำดับของแถวปัจจุบัน โดยเพิ่มค่า (`rcount += 1`) ทุกครั้งที่วนลูปแสดงผล

การทดสอบ

```
# สร้างกระดานตัวอย่าง
board1 = [["X", "O", "X"], [" ", "X", "O"], ["O", " ", "X"]]
board2 = [[ " ", " ", " "], [ " ", " ", " "], [ " ", " ", " "]]

print("Test: Display Board - Example 1")
display_board(board1)

print("Test: Display Board - Example 2 (Empty Board)")
display_board(board2)
```

ผลลัพธ์

```
Test: Display Board - Example 1
X | O | X
---+---+---
| X | O
---+---+---
O |   | X
Test: Display Board - Example 2 (Empty Board)
|   |
---+---+---
|   |
---+---+---
|   |
```

2. การสร้างกระดานเริ่มต้น create_board()

วัตถุประสงค์ คือ พงก์ชันนี้ใช้สำหรับสร้างกระดานเกม Tic-Tac-Toe เริ่มต้นในรูปแบบ 3×3 โดยกำหนดให้แต่ละตำแหน่งของกระดานเริ่มต้นเป็นช่องว่าง (" ")

```
# Initialize the game board
def create_board():
    return [ [ " " for _ in range(3) ] for _ in range(3) ]
```

- สร้างกระดานเปล่า 3×3 ใช้ list comprehension เพื่อสร้างกระดาน (2D list) ที่มี 3 แถว และแต่ละแถวมี 3 คอลัมน์ โดยแต่ละช่องจะถูกกำหนดค่าเริ่มต้นเป็นช่องว่าง (" ")
- คืนค่ากระดาน พงก์ชันจะคืนค่าเป็น 2D list ที่แสดงกระดานเปล่า. การทดสอบ

```
print("Test: Create Board")
board = create_board()
display_board(board)
```

ผลลัพธ์

```
Test: Create Board
|   |
---+---+---
|   |
---+---+---
|   |
```

3. การวางแผน make_move(board, player, position)

ฟังก์ชันนี้ใช้สำหรับวางแผนของผู้เล่น ("X" หรือ "O") บนกระดาน Tic-Tac-Toe ที่ตำแหน่งที่ผู้เล่นเลือก หากตำแหน่งนั้นว่างอยู่

```
# Make a move on the board
def make_move(board, player, position):
    row, col = position
    if board[row][col] == " ":
        board[row][col] = player
        return True
    else:
        print("Position already taken. Try again.")
        return False
```

รับพารามิเตอร์ board กระดานเกม (2D list), player: สัญลักษณ์ของผู้เล่น ("X" หรือ "O") และตำแหน่ง position ตำแหน่งที่ผู้เล่นต้องการวางแผนในรูปแบบ (row, col)

ตรวจสอบว่าตำแหน่งว่างหรือไม่ ใช้เงื่อนไข if board[row][col] == " " เพื่อตรวจสอบว่าช่องที่เลือกว่างอยู่ (" ") หากว่างอยู่ ให้วางหมายเลขของผู้เล่นลงไป (board[row][col] = player) และคืนค่า True

แจ้งเตือนเมื่อช่องไม่ว่าง หากช่องที่เลือกไม่ว่าง (board[row][col] != " ") และแสดงข้อความ "Position already taken. Try again." และคืนค่า False

การทดสอบ

```
# สร้างกระดานเริ่มต้น
board = create_board()

print("Test: Make Move")

# วางหมายเลขตำแหน่งที่ว่าง
print("Move 1 (0,0):", make_move(board, "X", (0, 0))) # Expected: True
display_board(board)

# พยายามวางหมายเลขตำแหน่งที่มีหมายเลขแล้ว
print("Move 2 (0,0):", make_move(board, "O", (0, 0))) # Expected: False
display_board(board)

# วางหมายเลขตำแหน่งใหม่
print("Move 3 (1,1):", make_move(board, "O", (1, 1))) # Expected: True
display_board(board)
```

ผลลัพธ์

Test: Make Move

Move 1 (0,0): True

```
X |   |
---+---+---
|   |
---+---+---
|   |
```

Position already taken. Try again.

Move 2 (0,0): False

```
X |   |
---+---+---
|   |
---+---+---
|   |
```

Move 3 (1,1): True

```
X |   |
---+---+---
| O |
---+---+---
|   |
```

อธิบายผลลัพธ์

- Move 1 (0,0) ผู้เล่น "X" วางหมากในตำแหน่ง (0,0) ได้สำเร็จ
 - Move 2 (0,0) ผู้เล่น "O" พยายามวางหมากในตำแหน่ง (0,0) ซึ่งมีหมาก "X" อยู่แล้ว จึงไม่สำเร็จ และแสดงข้อความเตือน
 - Move 3 (1,1) ผู้เล่น "O" วางหมากในตำแหน่ง (1,1) ได้สำเร็จ
4. การตรวจสอบผู้ชนะ check_win(board)

ฟังก์ชันนี้ตรวจสอบว่าผู้เล่นคนใดชนะเกมโดยดูจากการเรียงสัญลักษณ์เดียวกัน ("X" หรือ "O") ในแถว (row) คอลัมน์ (column) หรือ เส้นทแยงมุม (diagonal)

```
# Check if a player has won
def check_win(board):
    # Check rows
    for row in board:
        if row[0] == row[1] == row[2] and row[0] != " ":
            return True

    # Check columns
    for col in range(3):
        if board[0][col] == board[1][col] == board[2][col] and board[0][col] != " ":
            return True

    # Check diagonals
    if board[0][0] == board[1][1] == board[2][2] and board[0][0] != " ":
        return True
    if board[0][2] == board[1][1] == board[2][0] and board[0][2] != " ":
        return True

    return False
```

ตรวจสอบแถว (Rows) ใช้การวนลูป for row in board เพื่อตรวจสอบแต่ละแถว (row) หากทุกช่องในแถว (row[0], row[1], row[2]) มีสัญลักษณ์เดียวกัน และไม่ใช่ช่องว่าง (" "): คืนค่า True.

ตรวจสอบคอลัมน์ (Columns) ใช้ for col in range(3) เพื่อตรวจสอบแต่ละคอลัมน์ หากทุกช่องในคอลัมน์ (board[0][col], board[1][col], board[2][col]) มีสัญลักษณ์เดียวกัน และไม่ใช่ช่องว่าง คืนค่า True

ตรวจสอบเส้นทแยงมุม (Diagonals) ตรวจสอบเส้นทแยงมุมจากซ้ายบนไปขวาล่าง (board[0][0], board[1][1], board[2][2]) ตรวจสอบเส้นทแยงมุมจากขวาบนไปซ้ายล่าง (board[0][2], board[1][1], board[2][0]) หากมีสัญลักษณ์เดียวกัน และไม่ใช่ช่องว่าง คืนค่า True

คืนค่า False หากไม่มีเงื่อนไขใดที่เป็นจริง (ไม่มีผู้ชนะ) คืนค่า False

การทดสอบ

```
# กระดาษทดสอบ
board1 = [["X", "X", "X"], [" ", "O", " "], ["O", " ", "O"]] # แถวที่ 1 ชนะ
board2 = [["X", "O", "X"], [" ", "X", "O"], ["O", " ", "O"]] # ไม่มีผู้ชนะ
board3 = [["X", "O", " "], [" ", "X", " "], ["O", " ", "X"]]] # เส้นทแยงมุมจากซ้ายบนชนะ

# ทดสอบฟังก์ชัน check_win
print("Test: Check Win")
print("Board 1 Winner:", check_win(board1)) # Expected: True (Row Win)
print("Board 2 Winner:", check_win(board2)) # Expected: False (No Win)
print("Board 3 Winner:", check_win(board3)) # Expected: True (Diagonal Win)
```

ผลลัพธ์

```
Test: Check Win
Board 1 Winner: True
Board 2 Winner: False
Board 3 Winner: True
```

อธิบายผลลัพธ์

- Board 1 ผู้เล่น "X" ชนะจากการเรียงแถวที่ 1
- Board 2 ไม่มีผู้เล่นคนใดชนะ
- Board 3 ผู้เล่น "X" ชนะจากการเรียงเส้นทแยงมุมซ้ายบนถึงขวาล่าง

5. การตรวจสอบว่าเกมเสมอ `is_draw(board)`

ฟังก์ชันนี้ตรวจสอบว่าเกมเสมอหรือไม่ โดยพิจารณาจากกระดานที่เต็มทุกช่อง (ไม่มีช่องว่าง " ") และไม่มีผู้ชนะ

```
# Check if the board is full
def is_draw(board):
    for row in board:
        if " " in row:
            return False
    return True
```

ตรวจสอบแต่ละแถวในกระดาน ใช้ `for row in board:` เพื่อวนซ้ำในแต่ละแถวของกระดาน
ตรวจสอบว่าแถวปัจจุบันมีช่องว่าง (" ") อยู่หรือไม่ หากพบช่องว่าง คืนค่า `False` (เกมยังไม่เสมอ) หากไม่มีช่องว่างในทุกแถว คืนค่า `True` (กระดานเต็มและเกมเสมอ)

คืนค่า `True` หากไม่มีช่องว่างในกระดานเลย แสดงว่ากระดานเต็มและเกมจบลงด้วยผลเสมอ

การทดสอบ

```
board1 = [[ "X", "O", "X"], [ "O", "X", "O"], [ "O", "X", "O]] # กระดานเต็มเสมอ
board2 = [[ "X", "O", "X"], [ "O", "X", " "], [ "O", "X", "O]] # กระดานยังไม่เต็ม

# ทดสอบฟังก์ชัน is_draw
print("Test: Is Draw")
print("Board 1 Draw:", is_draw(board1)) # Expected: True
print("Board 2 Draw:", is_draw(board2)) # Expected: False
```

ผลลัพธ์

```
Test: Check Win Test: Is Draw
Board 1 Draw: True
Board 2 Draw: False
```

อธิบายผลลัพธ์

- Board 1 กระดานเต็มทุกช่อง ไม่มีช่องว่าง (" ") ผลลัพธ์ `True` (เกมเสมอ).
- Board 2 กระดานยังมีช่องว่างในตำแหน่ง `board[1][2]` ผลลัพธ์ `False` (เกมยังไม่จบ)

6. การตรวจสอบผู้ชนะ check_win(board)

ฟังก์ชันนี้ตรวจสอบว่าผู้เล่นคนใดชนะเกมโดยดูจากการเรียงสัญลักษณ์เดียวกัน ("X" หรือ "O") ในแถว (row) คอลัมน์ (column) หรือ เส้นทแยงมุม (diagonal)

```
# Check if a player has won
def check_win(board):
    # Check rows
    for row in board:
        if row[0] == row[1] == row[2] and row[0] != " ":
            return True

    # Check columns
    for col in range(3):
        if board[0][col] == board[1][col] == board[2][col] and board[0][col] != " ":
            return True

    # Check diagonals
    if board[0][0] == board[1][1] == board[2][2] and board[0][0] != " ":
        return True
    if board[0][2] == board[1][1] == board[2][0] and board[0][2] != " ":
        return True

    return False
```

ตรวจสอบแถว (Rows) ใช้การวนลูป for row in board เพื่อตรวจสอบแต่ละแถว (row) หากทุกช่องในแถว (row[0], row[1], row[2]) มีสัญลักษณ์เดียวกัน และไม่ใช่ช่องว่าง (" ") คืนค่า True.

ตรวจสอบคอลัมน์ (Columns) ใช้ for col in range(3) เพื่อตรวจสอบแต่ละคอลัมน์ หากทุกช่องในคอลัมน์ (board[0][col], board[1][col], board[2][col]) มีสัญลักษณ์เดียวกัน และไม่ใช่ช่องว่าง คืนค่า True

ตรวจสอบเส้นทแยงมุม (Diagonals) ตรวจสอบเส้นทแยงมุมจากซ้ายบนไปขวาล่าง (board[0][0], board[1][1], board[2][2]) ตรวจสอบเส้นทแยงมุมจากขวาบนไปซ้ายล่าง (board[0][2], board[1][1], board[2][0]) หากมีสัญลักษณ์เดียวกัน และไม่ใช่ช่องว่าง คืนค่า True

คืนค่า False หากไม่มีเงื่อนไขใดที่เป็นจริง (ไม่มีผู้ชนะ) คืนค่า False

การทดสอบ

```
# กระดาษทดสอบ
board1 = [["X", "X", "X"], [" ", "O", " "], ["O", " ", "O"]] # แถวที่ 1 ชนะ
board2 = [["X", "O", "X"], [" ", "X", "O"], ["O", " ", "O"]] # ไม่มีผู้ชนะ
board3 = [["X", "O", " "], [" ", "X", " "], ["O", " ", "X"]]] # เส้นทแยงมุมจากซ้ายบนขวาล่าง

# ทดสอบฟังก์ชัน check_win
print("Test: Check Win")
print("Board 1 Winner:", check_win(board1)) # Expected: True (Row Win)
print("Board 2 Winner:", check_win(board2)) # Expected: False (No Win)
print("Board 3 Winner:", check_win(board3)) # Expected: True (Diagonal Win)
```

ผลลัพธ์

```
Test: Check Win
Board 1 Winner: True
Board 2 Winner: False
Board 3 Winner: True
```

อธิบายผลลัพธ์

- Board 1 ผู้เล่น "X" ชนะจากการเรียงแถวที่ 1
- Board 2 ไม่มีผู้เล่นคนใดชนะ
- Board 3 ผู้เล่น "X" ชนะจากการเรียงเส้นทแยงมุมซ้ายบนถึงขวาล่าง

7. พัฒนา play_game()

พัฒนาหลักสำหรับการเล่นเกม Tic-Tac-Toe โดยดำเนินการตามลำดับการเล่นจนจบเกม (มีผู้ชนะหรือเสมอ)

```
def play_game():
    # Initialize the board and first player
    board = create_board()
    current_player = random_start()
    print(f"Player {current_player} goes first!")

    turns = 0

    while True:
        # Display the board
        display_board(board)

        # Get the player's move
        try:
            position = input(f"Player {current_player}, enter your move (row,col): ")
            row, col = map(int, position.split(","))
        except ValueError:
            print("Invalid input. Please enter row,col (e.g., 1,2).")
            continue

        # Validate the move
        if row < 1 or row > 3 or col < 1 or col > 3:
            print("Invalid position. Please choose between 1 and 3 for both row and column.")
            continue

        # Make the move
        if make_move(board, current_player, (row - 1, col - 1)):
            turns += 1
```

```

# Check for a win
if check_win(board):
    display_board(board)
    print(f"Player {current_player} wins!")
    break

# Check for a draw
if is_draw(board):
    display_board(board)
    print("It's a draw!")
    break

# Switch player
current_player = "O" if current_player == "X" else "X"

```

- การเริ่มต้นเกม พิมพ์ข้อความต้อนรับ `print("Welcome to Tic-Tac-Toe!")` และสร้างกระดานว่างใหม่โดยเรียก `create_board()` และแสดงกระดานโดยเรียก `display_board(board)`
 - สุ่มเลือกผู้เล่นเริ่มต้น ใช้ `random_start()` เพื่อสุ่มเลือกว่า "X" หรือ "O" จะเริ่มเล่นก่อน
 - ลูปการเล่นเกม
 - แสดงผู้เล่นที่ต้องเล่นในรอบปัจจุบัน (`current_player`)
 - รับตำแหน่งจากผู้เล่น (`row` และ `col`) โดยใช้ `input()` และตรวจสอบว่าค่าที่กรอกนั้นถูกต้องหรือไม่ถ้าค่าไม่ถูกต้องหรืออยู่นอกขอบเขต ให้แจ้งข้อความและให้กรอกใหม่
 - เรียกใช้ `make_move(board, current_player, (row, col))` เพื่อตรวจสอบและวางสัญลักษณ์
 - ตรวจสอบผลลัพธ์หลังจากแต่ละรอบ ตรวจสอบว่าผู้เล่นชนะโดยเรียก `check_win(board)` ถ้าชนะแสดงข้อความแสดงความยินดีและหยุดลูป ตรวจสอบว่ากระดานเต็มและเสมอโดยเรียก `is_draw(board)` ถ้าเสมอ แสดงข้อความแจ้งและหยุดลูป
 - สลับผู้เล่น หากยังไม่มีผลลัพธ์ (ไม่มีผู้ชนะหรือเสมอ) ให้สลับผู้เล่น `current_player = "O" if current_player == "X" else "X"`
- โปรแกรมสมบูรณ์

```

import random

# Randomly choose the first player
def random_start():
    return random.choice(["X", "O"])

print("Test: Random Start")
for _ in range(5):
    print(random_start())

```

```

# Initialize the game board
def create_board():
    return [[" " for _ in range(3)] for _ in range(3)]

# Display the board
def display_board(board):
    print("\n")
    rcount = 0
    for row in board:
        print(" "+" ".join(row))
        if rcount < 2:
            print("----+----+----")
        rcount += 1
    print("\n")

# Check if a player has won
def check_win(board):
    # Check rows
    for row in board:
        if row[0] == row[1] == row[2] and row[0] != " ":
            return True

    # Check columns
    for col in range(3):
        if board[0][col] == board[1][col] == board[2][col] and board[0][col]
!= " ":
            return True

    # Check diagonals
    if board[0][0] == board[1][1] == board[2][2] and board[0][0] != " ":
        return True
    if board[0][2] == board[1][1] == board[2][0] and board[0][2] != " ":
        return True

    return False

# Check if the board is full
def is_draw(board):
    for row in board:
        if " " in row:
            return False
    return True

# Make a move on the board
def make_move(board, player, position):
    row, col = position
    if board[row][col] == " ":
        board[row][col] = player

```

```

        return True
    else:
        print("Position already taken. Try again.")
        return False

# Main game function
def play_game():
    # Initialize the board and first player
    board = create_board()
    current_player = random_start()
    print(f"Player {current_player} goes first!")

    turns = 0

    while True:
        # Display the board
        display_board(board)

        # Get the player's move
        try:
            position = input(f"Player {current_player}, enter your move "
(row,col): ")
            row, col = map(int, position.split(","))
        except ValueError:
            print("Invalid input. Please enter row,col (e.g., 1,2).")
            continue

        # Validate the move
        if row < 1 or row > 3 or col < 1 or col > 3:
            print("Invalid position. Please choose between 1 and 3 for both "
row and column.")
            continue

        # Make the move
        if make_move(board, current_player, (row - 1, col - 1)):
            turns += 1

        # Check for a win
        if check_win(board):
            display_board(board)
            print(f"Player {current_player} wins!")
            break

        # Check for a draw
        if is_draw(board):
            display_board(board)
            print("It's a draw!")
            break

```

```

# Switch player
current_player = "O" if current_player == "X" else "X"

# Start the game
if __name__ == "__main__":
    # กระดาษทดสอบ
    play_game()

```

ตัวอย่างผลลัพธ์

```
Welcome to Tic-Tac-Toe!

|   |
---+---+---
|   |
---+---+---
|   |

X will start the game.
X's turn. Enter row (0-2): 0
Enter column (0-2): 0
```

```
X |   |
---+---+---
|   |
---+---+---
|   |
```

```
O's turn.
Enter row (0-2): 1
Enter column (0-2): 1
```

```
X |   |
---+---+---
| O |
---+---+---
|   |
```

```
X's turn.
Enter row (0-2): 0
Enter column (1-2): 1
```

```
X | X |
---+---+---
| O |
---+---+---
|   |
```

```
O's turn.
Enter row (0-2): 2
Enter column (0-2): 0
```

```
X | X |
---+---+---
| O |
---+---+---
O |   |
```

```
X's turn.
Enter row (0-2): 0
Enter column (2-2): 2
```

```
X | X | X
---+---+---
| O |
---+---+---
O |   |
```

Congratulations! X wins!

ตัวอย่างโปรแกรมในบทนี้ เช่น โปรแกรมคำนวณ BMI หรือเกม Tic-Tac-Toe จะช่วยให้เห็นภาพจริงของการนำแนวคิดเชิงโครงสร้างมาสร้างสรรค์โปรแกรมที่มีปฏิสัมพันธ์กับผู้ใช้งาน (Interactive) ได้อย่างมีประสิทธิภาพ ยิ่งไปกว่านั้นยังเป็นพื้นฐานที่สามารถต่อยอดไปสู่การพัฒนาแอปพลิเคชันที่ซับซ้อนและมีความน่าใช้มากขึ้นในอนาคต เช่น เกมแบบหลายผู้เล่นหรือแอปพลิเคชันที่มีฐานข้อมูลเชื่อมต่อกับผู้ใช้จริง

บทสรุป

ในยุคที่ซอฟต์แวร์มีความซับซ้อนมากขึ้น แนวคิดการเขียนโปรแกรมเชิงโครงสร้าง (Structured Programming) จึงกลายเป็นรากฐานสำคัญที่ช่วยให้การพัฒนาโปรแกรมมีความเป็นระเบียบ เข้าใจง่าย และลดความซับซ้อนของโค้ดลง โปรแกรมที่พัฒนาด้วยแนวคิดนี้จะมีโครงสร้างที่ชัดเจน โดยใช้หลักการทำงานแบบลำดับ (Sequence), การตัดสินใจแบบมีเงื่อนไข (Selection) และการวนซ้ำ (Iteration) รวมไปถึงการแบ่งปัญหาออกเป็นฟังก์ชันหรือโมดูลย่อย ซึ่งช่วยให้โปรแกรมมีความยืดหยุ่นและง่ายต่อการดูแลรักษา

บทนี้ยังเน้นการใช้แนวคิดการออกแบบโปรแกรมจากภาระรวมลงสู่รายละเอียด (Top-Down Design) ซึ่งเริ่มจากการมองภาพรวมของปัญหา แล้วค่อยๆ แยกออกเป็นงานย่อย พร้อมกับการวางแผนลำดับขั้นตอนที่เป็นระบบ เช่น การเขียนรหัสจำลอง (Pseudocode) หรือผังงาน (Flowchart) เพื่อช่วยให้ระบบการพัฒนาโปรแกรมมีความชัดเจนและลดข้อผิดพลาด นอกจากนี้ ยังมีตัวอย่างการประยุกต์ใช้งานจริง เช่น โปรแกรมแก้

สมการ โปรแกรมคำนวณค่าไฟฟ้า โปรแกรมคำนวณดัชนีมวลกาย (BMI) และเกม Tic-Tac-Toe ที่แสดงให้เห็นถึงความสามารถในการนำแนวคิดเชิงโครงสร้างไปแก้ไขปัญหาที่หลากหลายได้อย่างมีประสิทธิภาพ

- แนวคิดการเขียนโปรแกรมเชิงโครงสร้าง ช่วยลดความซับซ้อนของโค้ด และหมายเหตุกับการพัฒนาทีม มุ่งเน้นการจัดโครงสร้างโค้ดให้มีลำดับชัดเจน (Sequence, Selection, Iteration)
- การออกแบบแบบ Top-Down Design เริ่มจากภาพรวมและแบ่งเป็นปัญหาย่อย ๆ เพื่อแก้ไขทีละส่วน
- โครงสร้างพื้นฐานของโปรแกรมเชิงโครงสร้าง ใช้ลำดับคำสั่งที่ชัดเจน เช่น Sequential, Selection (if-else), Iteration (loops)
- การแยกโค้ดเป็นโมดูลและฟังก์ชัน ช่วยลดการเขียนโค้ดซ้ำและเพิ่มความชัดเจน
- ประโยชน์ของโปรแกรมเชิงโครงสร้าง ลดข้อผิดพลาด เพิ่มประสิทธิภาพ และสนับสนุนการทำงานเป็นทีม
- การใช้ Flowchart หรือ Pseudocode ช่วยวางแผนลำดับขั้นตอนการทำงานอย่างชัดเจน
- การระบุปัญหาและข้อกำหนดของระบบ เน้นการวิเคราะห์ความต้องการของผู้ใช้งานก่อนเริ่มพัฒนา
- การเก็บรวบรวมข้อมูล รวบรวมข้อมูลจากผู้ใช้งานหรือแหล่งข้อมูลอื่น ๆ เพื่อวางแผนระบบ
- การวิเคราะห์ข้อกำหนด ระบุฟังก์ชันหลักที่โปรแกรมต้องมี เช่น การคำนวณหรือการแสดงผล
- การวางแผนการแก้ปัญหา ใช้การแบ่งปัญหาย่อยและจัดลำดับขั้นตอนการแก้ปัญหาอย่างชัดเจน
- การสร้าง Flowchart ใช้แผนภาพแสดงกระบวนการทำงาน เช่น การเริ่มต้น การรับข้อมูล การวนลูป
- การเขียน Pseudocode ใช้รหัสกึ่งภาษาในการวางแผนโค้ด ช่วยลดข้อผิดพลาด
- การรับข้อมูลจากผู้ใช้งาน ใช้คำสั่ง input() ใน Python หรือ GUI เช่น Tkinter เพื่อรับข้อมูล
- การประมวลผลข้อมูล ใช้ฟังก์ชันและโครงสร้างควบคุม เช่น if-else และ loops
- การแสดงผลลัพธ์ แสดงข้อมูลในรูปแบบที่เข้าใจง่าย เช่น ข้อความหรือกราฟ
- การทดสอบและแก้ไขข้อผิดพลาด ใช้ Test Cases และ Debugger เพื่อตรวจสอบความถูกต้อง
- ออกแบบโปรแกรมด้วย Top-Down Design โดยแบ่งปัญหาเป็นส่วนย่อย ๆ เช่น สร้างบอร์ด, ตรวจเงื่อนไขการชนะ
- การแบ่งฟังก์ชันย่อยในโค้ด เช่น ฟังก์ชัน create_board() สร้างบอร์ด make_move() วางหมาก check_win() ตรวจผู้ชนะ

คำถามก้ายบทก

1. อธิบายความหมายของฟังก์ชัน (Function) และประโยชน์ที่สำคัญในการพัฒนาโปรแกรมการเขียนโปรแกรมเชิงโครงสร้างคืออะไร และมีความสำคัญอย่างไร?

2. โครงสร้างพื้นฐาน 3 อย่างของโปรแกรมเชิงโครงสร้างคืออะไร? ยกตัวอย่าง
3. การออกแบบโปรแกรมด้วยแนวคิด Top-Down Design ช่วยลดความซับซ้อนได้อย่างไร?
4. อธิบายความหมายของ Pseudocode และข้อดีของการใช้ Pseudocode
5. การใช้ Flowchart มีความสำคัญอย่างไรในการออกแบบโปรแกรม?
6. การจัดลำดับขั้นตอนการแก้ปัญหา (Stepwise Refinement) มีหลักการอย่างไร?
7. ทำไงการใช้โครงสร้างควบคุมที่ซับเจน (Control Structures)
8. จึงช่วยให้โปรแกรมเข้าใจง่ายขึ้น? ขั้นตอนสำคัญของการ ระบุปัญหาและข้อกำหนดของระบบ มีอะไรบ้าง?
9. การเก็บรวบรวมข้อมูลจากผู้ใช้งานมีความสำคัญต่อการออกแบบโปรแกรมอย่างไร?
10. ยกตัวอย่างกระบวนการวิเคราะห์ข้อกำหนดของระบบในโปรแกรมคำนวนพื้นที่.
11. การออกแบบโครงสร้างโปรแกรมเพื่อลดข้อผิดพลาดในการพัฒนาการเริ่มจากอะไร?
12. การใช้ Top-Down Design ช่วยจัดการกับโปรแกรมขนาดใหญ่ได้อย่างไร?
13. การแบ่งปัญหาออกเป็นส่วนย่อย (Decomposition) มีหลักการใดที่ควรคำนึงถึง?
14. การกำหนดข้อกำหนดที่ไม่ใช่ฟังก์ชัน (Non-functional Requirements) เช่น ความเร็ว มีความสำคัญอย่างไร?
15. เขียน Pseudocode สำหรับโปรแกรมคำนวนค่า BMI และประเมินสถานะสุขภาพ
16. อธิบายโครงสร้างโปรแกรม Tic-Tac-Toe ด้วยแนวคิด Top-Down Design
17. เขียนตัวอย่างโค้ดการรับข้อมูลจากผู้ใช้งานใน Python
18. การแสดงผลลัพธ์ในรูปแบบกราฟิก เช่น Matplotlib มีประโยชน์ในกรณีใด?
19. อธิบายขั้นตอนการเขียน Pseudocode สำหรับโปรแกรมสุ่มเลขและเกมเดาตัวเลข
20. ยกตัวอย่างการออกแบบโปรแกรมแก้สมการกำลังสองด้วย Pseudocode

โจทย์การเขียนโปรแกรม

1. เขียนโปรแกรมโปรแกรมตรวจสอบจำนวนคู่หรือคี่ รับตัวเลขจากผู้ใช้และตรวจสอบว่าเป็นเลขคู่หรือคี่

Enter your number: 8

It is even.

2. เขียนโปรแกรมหาค่ามากที่สุดใน 3 จำนวน รับค่าจำนวน 3 ตัว และแสดงค่ามากที่สุด

Enter the 1st number : 10

Enter the 2nd number: 20

Enter the 3rd number: 15

The maximum value is 20

3. เขียนโปรแกรมโปรแกรมคำนวนค่าดัชนีมวลกาย (BMI)

Enter weight (kg): 70

Enter height (m): 1.5

BMI: 22.86

Health status : Normal

4. เขียนโปรแกรมแปลงเวลา (ชั่วโมง -> นาที -> วินาที)

Enter hour : 2

The total seconds is 7200

5. เขียนโปรแกรมคำนวณค่าไฟฟ้าใช้อัตราค่าไฟแบบขั้นบันได (0-50 หน่วย 3 บาท/หน่วย, 51-150 หน่วย 4 บาท/หน่วย, 151+ หน่วย 5 บาท/หน่วย)

Enter electric units: 120

The total cost is 430 baht

6. เขียนโปรแกรมสุ่มรหัสผ่าน (Password Generator) สร้างรหัสผ่านความยาว 8 ตัวที่มีตัวอักษรและตัวเลขแบบสุ่ม

Your password : a3b7Xz9P

7. เขียนฟังก์ชันวาดเส้น (line_base) สร้างฟังก์ชันรับ nspace และ ncol และวาดเส้น # ที่มีช่องว่างด้านหน้า ตัวอย่าง line_base(2, 4)

####

8. วาดสามเหลี่ยมดาว (top_leaf) เขียนฟังก์ชันวาดสามเหลี่ยมดาว 3 บรรทัด โดยเรียกใช้ line_base ในข้อ 7

*

9. เขียนโปรแกรมวาดต้นไม้ สร้างโปรแกรมวาดต้นไม้ที่ใช้ฟังก์ชัน line_base และ top_leaf

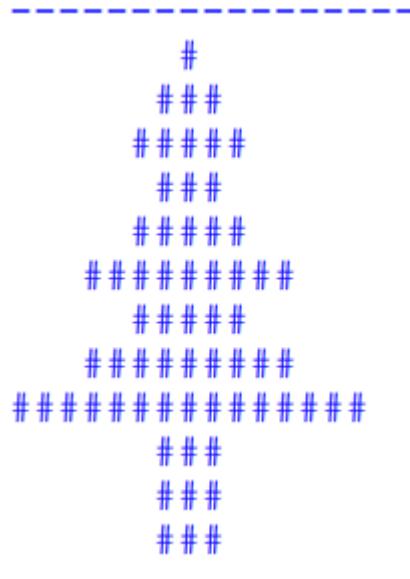
*

#

#

10. เขียนโปรแกรมสร้างเกม Tic-Tac-Toe

11. จากข้อ 9 สร้างต้นไม้ดังรูป โดยใช้หลักการ Top Dow Design



บทที่ 6

การจัดการข้อมูลและการอ่านไฟล์

เนื้อหาบทเรียน

- 6.1 รูปแบบโครงสร้างข้อมูล (list, dictionary, table)
- 6.2 การอ่านและเขียนไฟล์ใน Python
- 6.3 การประมวลผลข้อมูลจากไฟล์
- 6.4 การติดตั้งและใช้งานไลบรารีใน Python
- 6.5 การใช้งานไลบรารีที่ได้รับความนิยม (NumPy, Pandas, Matplotlib)

วัตถุประสงค์การเรียนรู้

- รูปแบบโครงสร้างข้อมูล ลิสต์ ดิกชันนารีและการสร้างตาราง
- เขียนโปรแกรมอ่านและเขียนไฟล์ใน Python ได้
- เขียนโปรแกรมเพื่อประมวลผลข้อมูลจากไฟล์ได้
- ติดตั้งและเขียนโปรแกรมเพื่อใช้งานไลบรารีได้
- เขียนโปรแกรมเพื่อใช้งานไลบรารีที่เป็นที่นิยม ได้แก่ NumPy Pandas Matplotlib ได้

ในยุคปัจจุบัน ข้อมูลถือเป็นทรัพยากรที่สำคัญในการขับเคลื่อนองค์กรและกระบวนการตัดสินใจในหลากหลายสาขาวิชา ไม่ว่าจะเป็นธุรกิจ การศึกษา วิทยาศาสตร์ หรืออุตสาหกรรม การจัดการและการวิเคราะห์ข้อมูลอย่างมีประสิทธิภาพจึงกลายเป็นสิ่งจำเป็นสำหรับการพัฒนาและเพิ่มขีดความสามารถในการแข่งขัน การเลือกใช้โครงสร้างข้อมูลและเครื่องมือที่เหมาะสมมีบทบาทสำคัญต่อกระบวนการดังกล่าว

ภาษาโปรแกรม Python เป็นเครื่องมือที่ได้รับความนิยมในงานด้านข้อมูล เนื่องจากมีความยืดหยุ่น และมีไลบรารีที่ทรงพลัง เช่น NumPy สำหรับการคำนวณเชิงตัวเลข Pandas สำหรับการจัดการข้อมูลในรูปแบบตาราง และ Matplotlib สำหรับการสร้างกราฟและการแสดงผลข้อมูลในรูปแบบต่าง ๆ ไลบรารีเหล่านี้ช่วยให้การจัดเก็บ การประมวลผล และการนำเสนอข้อมูลสามารถทำได้อย่างมีประสิทธิภาพและสอดคล้องกับความต้องการในเชิงวิชาการและวิชาชีพ

เนื้อหาในบทนี้ครอบคลุมการใช้งานโครงสร้างข้อมูลพื้นฐาน เช่น List และ Dictionary ซึ่งเหมาะสมสำหรับการจัดเก็บข้อมูลเบื้องต้น และต่อยอดสู่การใช้งานไลบรารี NumPy, Pandas และ Matplotlib ผ่านตัวอย่างและกรณีศึกษาที่เชื่อมโยงกับการวิเคราะห์ข้อมูลในสถานการณ์จริง นอกจากนี้ ยังมุ่งเน้นการเสริมสร้างความเข้าใจในการจัดการข้อมูลเชิงโครงสร้าง การคำนวณเชิงตัวเลข และการแสดงผลข้อมูลด้วยวิธีการเชิงวิทยาศาสตร์

บทนี้จึงเป็นแนวทางสำคัญสำหรับผู้เรียนในการพัฒนาทักษะด้านการจัดการและวิเคราะห์ข้อมูล โดยให้ความสำคัญกับการเลือกใช้เครื่องมือที่เหมาะสมเพื่อเพิ่มประสิทธิภาพของกระบวนการทำงาน และสามารถประยุกต์ใช้ความรู้ที่ได้กับงานวิจัยหรือโครงการที่เกี่ยวข้องในสาขาต่าง ๆ อย่างเหมาะสมและมีประสิทธิผล

6.1 รูปแบบโครงสร้างข้อมูล

ในกระบวนการวิเคราะห์ข้อมูล การจัดเก็บและจัดการข้อมูลในรูปแบบที่มีโครงสร้างชัดเจนเป็นสิ่งสำคัญ เพื่อให้ง่ายต่อการเข้าถึง การประมวลผล และการวิเคราะห์ข้อมูล Python มีโครงสร้างข้อมูลที่หลากหลาย เช่น List และ Dictionary ซึ่งแต่ละโครงสร้างมีคุณสมบัติและประโยชน์ที่แตกต่างกัน เหมาะสำหรับการใช้งานในบริบทต่าง ๆ

ในหัวข้อนี้ จะนำเสนอการจัดการข้อมูลในรูปแบบ Table ซึ่งเป็นการเก็บข้อมูลแบบมีลำดับชัดเจนโดยใช้โครงสร้าง List และ Dictionary พร้อมทั้งเปรียบเทียบข้อดีของแต่ละวิธี ตัวอย่างการใช้งานที่นำเสนอจะมุ่งเน้นไปที่การเก็บคะแนนสอบของนักเรียนในรายวิชา เช่น คณิตศาสตร์ วิทยาศาสตร์ การอ่าน และการสะกดคำ โดยใช้โครงสร้างข้อมูลเหล่านี้เพื่อแสดงถึงวิธีการจัดเก็บ วิเคราะห์ และเข้าถึงข้อมูลได้อย่างมีประสิทธิภาพ

เนื้อหานี้จะช่วยให้ผู้เรียนเข้าใจการจัดการข้อมูลที่ซับซ้อนมากขึ้น และสามารถเลือกใช้โครงสร้างข้อมูลที่เหมาะสมสำหรับการจัดเก็บข้อมูลในโปรเจกต์ได้อย่างมีประสิทธิผล

6.1.1 การใช้งาน list เพื่อจัดเก็บข้อมูลแบบลำดับ

จากลิสต์ (List) ใน Python หัวข้อที่ 2.5 เป็นโครงสร้างข้อมูลที่สามารถเก็บข้อมูลหลายค่าไว้ในตัวแปรเดียวกัน โดยรองรับข้อมูลที่หลากหลายชนิด เช่น ตัวเลข ข้อความ และค่าอื่น ๆ อีกทั้งยังสามารถจัดการข้อมูลได้ยืดหยุ่น เช่น เพิ่ม ลบ หรือแก้ไขข้อมูลในลิสต์ผ่านฟังก์ชันต่าง ๆ เช่น append() remove() และ sort() ลิสต์เหมาะสมสำหรับการใช้งานที่ต้องการจัดเก็บข้อมูลเป็นลำดับและเข้าถึงข้อมูลด้วยดัชนี (index) ที่เริ่มต้นจาก 0

ตัวอย่างการใช้ลิสต์ในการเก็บคะแนนสอบ ในกรณีที่ต้องการวิเคราะห์คะแนนของนักเรียนหลายคน เช่น การคำนวณคะแนนเฉลี่ย ค่าน้ำหนักเรียนที่ได้คะแนนสูงสุดและต่ำสุด รวมถึงแสดงข้อมูลที่เชื่อมโยงระหว่างคะแนนกับรายชื่อนักเรียน การจัดการข้อมูลด้วย ลิสต์ (List) เป็นวิธีที่เรียบง่ายและสะดวก โค้ดตัวอย่างนี้จะแสดงวิธีการจัดการข้อมูลนักเรียนและคะแนนเพื่อให้ได้ผลลัพธ์ที่ต้องการ โค้ดดังนี้

```
# รายชื่อนักเรียนและคะแนน
students = ["Alice", "Bob", "Charlie", "David", "Eve"]
scores = [85, 90, 78, 92, 88]

# คำนวณคะแนนเฉลี่ย
average_score = sum(scores) / len(scores)
print(f"คะแนนเฉลี่ย: {average_score:.2f}") # Output: 86.6

# ค้นหาคะแนนสูงสุดและต่ำสุด
max_score = max(scores)
min_score = min(scores)

# หาชื่อนักเรียนที่ได้คะแนนสูงสุดและต่ำสุด
max_score_student = students[scores.index(max_score)]
min_score_student = students[scores.index(min_score)]

print(f"คะแนนสูงสุด: {max_score} (โดย {max_score_student})")
print(f"คะแนนต่ำสุด: {min_score} (โดย {min_score_student})")
```

อธิบายโค้ดได้ดังนี้

- กำหนดรายชื่อนักเรียนและคะแนน students คือ ลิสต์ที่เก็บรายชื่อนักเรียน scores คือ ลิสต์ที่เก็บคะแนนของนักเรียน โดยแต่ละค่าของคะแนนจะตรงกับนักเรียนในลิสต์ students ตามตำแหน่งดัชนี (index)

```
students = ["Alice", "Bob", "Charlie", "David", "Eve"]
scores = [85, 90, 78, 92, 88]
```

- คำนวณคะแนนเฉลี่ย ใช้ฟังก์ชัน sum(scores) เพื่อหาผลรวมของคะแนนทั้งหมด ใช้ len(scores) เพื่อหาจำนวนของนักเรียน (จำนวนของคะแนนในลิสต์) จากนั้นหาค่าเฉลี่ยโดยการนำผลรวมของคะแนนหารด้วยจำนวนคะแนนทั้งหมด และแสดงผลลัพธ์โดยใช้รูปแบบ .2f เพื่อแสดงคะแนนเฉลี่ยในรูปแบบทศนิยม 2 ตำแหน่ง

```
# คำนวณคะแนนเฉลี่ย
average_score = sum(scores) / len(scores)
print(f"คะแนนเฉลี่ย: {average_score:.2f}") # Output: 86.6
```

3. ค้นหาคะแนนสูงสุดและต่ำสุด ใช้ฟังก์ชัน `max(scores)` เพื่อหาค่าคะแนนที่มากที่สุดในลิสต์ ใช้ฟังก์ชัน `min(scores)` เพื่อหาค่าคะแนนที่น้อยที่สุดในลิสต์

```
# ค้นหาคะแนนสูงสุดและต่ำสุด
max_score = max(scores)
min_score = min(scores)
```

4. หาชื่อนักเรียนที่ได้คะแนนสูงสุดและต่ำสุด ใช้ `scores.index(max_score)` เพื่อหาดัชนีของคะแนนสูงสุดในลิสต์ `scores` และใช้ `scores.index(min_score)` เพื่อหาดัชนีของคะแนนต่ำสุดในลิสต์ `scores` จากนั้นใช้ดัชนีที่ได้ไปอ้างอิงในลิสต์ `students` เพื่อดึงชื่อนักเรียนที่ตรงกับคะแนนสูงสุดและต่ำสุด

```
# หาชื่อนักเรียนที่ได้คะแนนสูงสุดและต่ำสุด
max_score_student = students[scores.index(max_score)]
min_score_student = students[scores.index(min_score)]
```

5. แสดงผลลัพธ์ แสดงผลคะแนนสูงสุดพร้อมชื่อของนักเรียนที่ได้คะแนนนั้น และแสดงผลคะแนนต่ำสุดพร้อมชื่อของนักเรียนที่ได้คะแนนนั้น

การสร้างฟังก์ชันรับอินพุต

การจัดการข้อมูลนักเรียนและคะแนนสามารถทำได้อย่างมีประสิทธิภาพผ่านการใช้ ลิสต์ (List) ซึ่งเป็นโครงสร้างข้อมูลที่มีความยืดหยุ่นและสามารถเก็บข้อมูลหลายค่าได้ในตัวแปรเดียวทั้งนี้ โค้ดตัวอย่างนี้แสดงวิธีการสร้างฟังก์ชันสำหรับรับข้อมูลจากผู้ใช้ เช่น รายชื่อนักเรียนและคะแนนสอบ โดยการเพิ่มข้อมูลทีละรายการจนกว่าผู้ใช้จะพิมพ์คำว่า "exit" เพื่อหยุดกระบวนการรับข้อมูล

```
def input_students_and_scores():
    students = []
    scores = []
    print("โปรดป้อนข้อมูลนักเรียนและคะแนน (พิมพ์ 'exit' เพื่อหยุด):")

    while True:
        name = input("ชื่อนักเรียน: ")
        if name.lower() == "exit":
            break
        try:
            score = float(input(f"คะแนนของ {name}: "))
            students.append(name)
            scores.append(score)
        except ValueError:
```

```

print("กรุณาป้อนคะแนนในรูปแบบตัวเลข")

return students, scores

# เรียกใช้งานฟังก์ชัน
students, scores = input_students_and_scores()

```

อธิบายโค้ด

1. การสร้างฟังก์ชัน input_students_and_scores

- วัตถุประสงค์ ฟังก์ชันนี้ออกแบบมาเพื่อรับข้อมูลรายชื่อนักเรียนและคะแนนจากผู้ใช้ และเก็บข้อมูลในลิสต์สองลิสต์ ได้แก่ students สำหรับรายชื่อ และ scores สำหรับคะแนน
- การทำงาน ฟังก์ชันเริ่มต้นด้วยการสร้างลิสต์ว่างสองลิสต์คือ students และ scores และแสดงข้อความแนะนำผู้ใช้ว่า สามารถป้อนข้อมูลนักเรียนและคะแนนได้ และพิมพ์ "exit" เพื่อหยุดการป้อนข้อมูล
- ใช้ ลูป while เพื่อให้ผู้ใช้ป้อนข้อมูลซ้ำจนกว่าจะพิมพ์ "exit"

2. กระบวนการรับข้อมูล

- ผู้ใช้ป้อนชื่อของนักเรียน หากผู้ใช้พิมพ์ "exit" (ทั้งตัวพิมพ์ใหญ่และเล็ก) จะหยุดการทำงานของลูปหากซื้อไม่ใช่ "exit" ฟังก์ชันจะขอให้ผู้ใช้ป้อนคะแนนของนักเรียน คะแนนจะถูกแปลงเป็นชนิดข้อมูล float เพื่อรับคะแนนที่มีทศนิยม และเพิ่มชื่อนักเรียนลงในลิสต์ students และคะแนนในลิสต์ scores
- หากผู้ใช้ป้อนคะแนนในรูปแบบที่ไม่ถูกต้อง (เช่น ตัวอักษรแทนตัวเลข) ฟังก์ชันจะแสดงข้อความเตือนให้กรอกใหม่

3. การส่งค่ากลับ เมื่อผู้ใช้พิมพ์ "exit" ฟังก์ชันจะส่งคืนลิสต์ students และ scores กลับไปยังตัว

โปรแกรม

4. การเรียกใช้งานฟังก์ชัน ฟังก์ชันถูกเรียกใช้งานด้วยคำสั่ง students, scores = input_students_and_scores() โดยผลลัพธ์ที่ได้จะถูกเก็บไว้ในตัวแปร students และ scores ผลการรันโปรแกรม

โปรดป้อนข้อมูลนักเรียนและคะแนน (พิมพ์ 'exit' เพื่อหยุด):

ชื่อนักเรียน: Alice

คะแนนของ Alice: 85

ชื่อนักเรียน: Bob

คะแนนของ Bob: 90

ชื่อนักเรียน: Charlie

คะแนนของ Charlie: 78

```

ชื่อนักเรียน: David
คะแนนของ David: 92
ชื่อนักเรียน: Eve
คะแนนของ Eve: 88
ชื่อนักเรียน: exit

```

6.1.2 การใช้งาน dictionary เพื่อจัดเก็บข้อมูลแบบคู่คีย์-ค่า

จาก Dictionary ใน Python หัวข้อที่ 2.6 เป็นโครงสร้างข้อมูลใช้เก็บข้อมูลแบบจับคู่ระหว่าง Key และ Value โดย Key ใช้สำหรับการอ้างอิงข้อมูล และ Value คือตัวข้อมูลที่เก็บไว้ Dictionary มีคุณสมบัติที่สำคัญ เช่น การเข้าถึงข้อมูลด้วย Key ได้อย่างรวดเร็ว และรองรับข้อมูลที่หลากหลายชนิด ทั้งตัวเลข ข้อความ ลิสต์ หรือแม้แต่ Dictionary ข้อนั้นเอง ในการจัดการข้อมูลรายชื่อนักเรียนและคะแนน การใช้ Dictionary ใน Python มีความสะดวกและเหมาะสมสำหรับข้อมูลที่มีความสัมพันธ์แบบคู่ เช่น ชื่อนักเรียนและคะแนนสอบ เพราะสามารถเข้าถึงข้อมูลได้โดยตรงผ่าน Key โดยไม่ต้องใช้ดัชนีเหมือนกับ List อีกทั้งยังช่วยลดความซับซ้อนในการจัดการข้อมูล เช่น การเพิ่ม ลบ หรือค้นหาค่าต่าง ๆ อย่างมีประสิทธิภาพ

โค้ดตัวอย่างนี้แสดงการใช้ Dictionary ในการจัดเก็บและวิเคราะห์ข้อมูลนักเรียนและคะแนนสอบ พร้อมคำนวณคะแนนเฉลี่ย ค้นหานักเรียนที่ได้คะแนนสูงสุดและต่ำสุด และแสดงผลลัพธ์ที่เชื่อมโยงระหว่างชื่อและคะแนน

```

student_scores = {
    "Alice": 85,
    "Bob": 90,
    "Charlie": 78,
    "David": 92,
    "Eve": 88
}
# คำนวณคะแนนเฉลี่ย
average_score = sum(student_scores.values()) / len(student_scores)
print(f"คะแนนเฉลี่ย: {average_score:.2f}") # Output: 86.6

# ค้นหาคะแนนสูงสุดและต่ำสุด
max_score = max(student_scores.values())
min_score = min(student_scores.values())

# นักเรียนที่ได้คะแนนสูงสุดและต่ำสุด
max_score_student = [name for name, score in student_scores.items() if score == max_score]
min_score_student = [name for name, score in student_scores.items() if score == min_score]

print(f"คะแนนสูงสุด: {max_score} (โดย {' , '.join(max_score_student)})")
print(f"คะแนนต่ำสุด: {min_score} (โดย {' , '.join(min_score_student)})")

```

อธิบายโค้ด

- เก็บข้อมูลรายชื่อนักเรียนและคะแนน ใช้ Dictionary ในการจับคู่ระหว่างชื่อนักเรียน (Key) และคะแนน (Value) เช่น {"Alice": 85} ซึ่งช่วยลดความซับซ้อนของการอ้างอิงข้อมูลเมื่อเทียบกับการใช้ List

```
student_scores = {
    "Alice": 85,
    "Bob": 90,
    "Charlie": 78,
    "David": 92,
    "Eve": 88
```

- คำนวณคะแนนเฉลี่ย ใช้ sum(student_scores.values()) เพื่อคำนวณผลรวมของคะแนนทั้งหมด และ len(student_scores) เพื่อหาจำนวนนักเรียน จากนั้นนำหารเพื่อหาค่าเฉลี่ย และแสดงผลลัพธ์ด้วยทศนิยม 2 ตำแหน่งด้วย .2f

```
# คำนวณคะแนนเฉลี่ย
average_score = sum(student_scores.values()) / len(student_scores)
print(f"คะแนนเฉลี่ย: {average_score:.2f}") # Output: 86.6
```

- ค้นหาคะแนนสูงสุดและต่ำสุด ใช้ max(student_scores.values()) และ min(student_scores.values()) เพื่อดึงค่าคะแนนสูงสุดและต่ำสุดจาก Dictionary

```
# ค้นหาคะแนนสูงสุดและต่ำสุด
max_score = max(student_scores.values())
min_score = min(student_scores.values())
```

- หานักเรียนที่ได้คะแนนสูงสุดและต่ำสุด ใช้ List Comprehension [name for name, score in student_scores.items() if score == max_score] เพื่อตรวจสอบชื่อ (Key) ของนักเรียนที่มีคะแนนตรงกับค่าคะแนนสูงสุด (max_score) และต่ำสุด (min_score)

```
# หานักเรียนที่ได้คะแนนสูงสุดและต่ำสุด
max_score_student = [name for name, score in student_scores.items() if score == max_score]
min_score_student = [name for name, score in student_scores.items() if score == min_score]
```

- หานักเรียนที่ได้คะแนนสูงสุดและต่ำสุด ใช้ List Comprehension [name for name, score in student_scores.items() if score == max_score] เพื่อตรวจสอบชื่อ (Key) ของนักเรียนที่มีคะแนนตรงกับค่าคะแนนสูงสุด (max_score) และต่ำสุด (min_score)

```
print(f"คะแนนสูงสุด: {max_score} (โดย {' '.join(max_score_student)})")
print(f"คะแนนต่ำสุด: {min_score} (โดย {' '.join(min_score_student)})")
```

การสร้างฟังก์ชันรับอินพุตส่งค่า Dictionary กลับ

```
# พังก์ชันรับอินพุตข้อมูลรายชื่อนักเรียนและคะแนน
def input_student_scores():
    student_scores = {}
    print("โปรดป้อนข้อมูลนักเรียนและคะแนน (พิมพ์ 'exit' เพื่อหยุด):")

    while True:
        name = input("ชื่อนักเรียน: ")
        if name.lower() == "exit":
            break
        try:
            score = float(input(f"คะแนนของ {name}: "))
            student_scores[name] = score
        except ValueError:
            print("กรุณาป้อนคะแนนในรูปแบบตัวเลข")

    return student_scores
```

6.1.3 การจัดการข้อมูลในรูปแบบ table โดยใช้โครงสร้างข้อมูล

การจัดการข้อมูลในรูปแบบ Table เป็นการจัดเก็บข้อมูลแบบมีโครงสร้าง เพื่อให้สามารถเข้าถึงหรือวิเคราะห์ข้อมูลได้อย่างมีประสิทธิภาพ ใน Python มีหลายแนวทางในการจัดการข้อมูลแบบ Table ขึ้นอยู่กับความต้องการของการใช้งาน โดยสามารถใช้โครงสร้างข้อมูลที่ยืดหยุ่น เช่น List Dictionary หรือแม้แต่โครงสร้างข้อมูลขั้นสูงในไลบรารีอย่าง Pandas (ซึ่งจะกล่าวเพิ่มเติมในหัวข้อต่อไปในบทนี้)

ตัวอย่างการจัดการข้อมูลในรูปแบบ Table ด้วย List ในกรณีที่ข้อมูลถูกจัดเก็บเป็นลำดับ เช่น คะแนนของนักเรียนในแต่ละวิชา สามารถใช้ List ได้สองแนวทางหลัก แสดงตัวอย่างดังรูป

	Math	Science	Reading	Spelling
Joe	55	63	77	81
Tom	65	61	67	72
Beth	97	95	92	88

รูปที่ 6.1 แสดงตารางคะแนน

1. การเก็บข้อมูลสำหรับแต่ละคน และเป็นรายวิชา เหมาะสำหรับการดึงข้อมูลของบุคคลใดบุคคลหนึ่งพร้อมทุกวิชา และข้อมูลในแต่ละลิสต์สอดคล้องกับลำดับของวิชา

```
# คะแนนของนักเรียนแต่ละคนในแต่ละวิชา
joeMarks = [55, 63, 77, 81] # Math, Science, Reading, Spelling
tomMarks = [65, 61, 67, 72]
bethMarks = [97, 95, 92, 88]
```

```
print(f"คะแนนของ Joe ในวิชาคณิตศาสตร์: {joeMarks[0]}") # Output: 55
```

2. การเก็บข้อมูลตามวิชา แยกเป็นรายคน หมายความว่าสำหรับการวิเคราะห์ข้อมูลในแต่ละวิชาข้อมูลในแต่ละลิสต์สอดคล้องกับลำดับของนักเรียน

```
mathMarks = [55, 65, 97]
scienceMarks = [63, 61, 95]
readingMarks = [77, 67, 92]
spellingMarks = [81, 72, 88]

# การเข้าถึงคะแนนวิชาคณิตศาสตร์ของนักเรียนคนที่ 1
print(f"คะแนนวิชาคณิตศาสตร์ของนักเรียนคนที่ 1: {mathMarks[0]}") # Output: 55
```

จากตัวอย่างก่อนหน้า การจัดการข้อมูลในรูปแบบ Table โดยใช้โครงสร้างข้อมูลแบบลิสต์ (List) สามารถนำไปใช้เก็บและแสดงผลข้อมูลนักเรียนและคะแนนในแต่ละวิชาได้ ตัวอย่างดังที่แสดงในภาพมีการเก็บข้อมูลเป็น List ซ้อน List โดยแต่ละรายการในลิสต์ชั้นนอกแทนข้อมูลของนักเรียนแต่ละคน และลิสต์ชั้นในแทนคะแนนในแต่ละวิชาของนักเรียนคนนั้น

1. การสร้างลิสต์ของคะแนนนักเรียน โครงสร้างนี้เก็บคะแนนในแต่ละวิชาของนักเรียนเป็นลิสต์แยกกัน (Math, Science, Reading, Spelling) แต่ละรายการในลิสต์ classMarks แทนคะแนนของนักเรียนคนหนึ่ง

```
classMarks = [
    [55, 63, 77, 81], # คะแนนของ Joe
    [65, 61, 67, 72], # คะแนนของ Tom
    [97, 95, 92, 88] # คะแนนของ Beth
]
print(classMarks)
```

ผลลัพธ์

```
[[55, 63, 77, 81], [65, 61, 67, 72], [97, 95, 92, 88]]
```

การพิมพ์แบบวนลูป

```
or studentMarks in classMarks:
    print(studentMarks)
```

ผลลัพธ์

```
[55, 63, 77, 81]
[65, 61, 67, 72]
[97, 95, 92, 88]
```

การเข้าถึงข้อมูลของนักเรียนคนแรก (Joe)

```
print(classMarks[0])
```

ผลลัพธ์

[55, 63, 77, 81]

การเข้าถึง โดยเลือกค่าลำดับที่ 2 จากลิสต์ของ Joe ซึ่งแทนคะแนนในวิชา Reading

```
print(classMarks[0][2])
```

ผลลัพธ์

77

ผู้แต่งเห็นว่าโครงสร้างข้อมูลเป็น "เครื่องมือพื้นฐาน" ที่ช่วยจัดระเบียบข้อมูลให้อยู่ในรูปแบบที่เหมาะสมต่อการใช้งาน โครงสร้างแต่ละแบบ เช่น ลิสต์หรือ딕ษันนารี มีข้อดีที่แตกต่างกัน ขึ้นอยู่กับลักษณะของข้อมูลที่ต้องจัดการ การเลือกใช้โครงสร้างข้อมูลที่เหมาะสมตั้งแต่แรกจึงช่วยลดปัญหาในการพัฒนาและวิเคราะห์ข้อมูลในระยะยาว

6.2 การอ่านและเขียนไฟล์

Python มีฟังก์ชันและโมดูลในตัวที่ทำให้การจัดการไฟล์เป็นเรื่องง่าย ไม่ว่าจะเป็นการอ่านหรือเขียนไฟล์ข้อมูล ไฟล์ CSV หรือไฟล์ JSON การจัดการไฟล์อย่างมีประสิทธิภาพถือเป็นสิ่งสำคัญสำหรับงานที่เกี่ยวข้องกับข้อมูลในรูปแบบต่าง ๆ

ในหัวข้อนี้ ผู้เรียนจะได้เรียนรู้เกี่ยวกับการอ่านและเขียนไฟล์ใน Python โดยจะเน้นทั้งไฟล์ข้อมูล (.txt) ไฟล์ CSV และไฟล์ JSON พร้อมตัวอย่างโค้ดที่สามารถนำไปประยุกต์ใช้งานได้จริง

6.2.1 การอ่านไฟล์ข้อมูล (.txt) ด้วย open() และ read()

การอ่านไฟล์ข้อมูลใน Python สามารถทำได้โดยใช้ฟังก์ชัน `open()` เพื่อเปิดไฟล์ และใช้ `read()` หรือ `readlines()` เพื่ออ่านข้อมูลในไฟล์ สมมติว่ามีไฟล์ชื่อ `example.txt` มีเนื้อหาดังนี้

ตัวอย่างไฟล์ `example.txt`

```
Alice,90
Bob,85
Charlie,78
David,92
Eve,88
```

1. การอ่านไฟล์ข้อมูลทั้งหมด `open("example.txt", "r")` เปิดไฟล์ `example.txt` ในโหมดอ่าน `read()` อ่านเนื้อหาในไฟล์ทั้งหมดและเก็บไว้ในตัวแปร `content` และใช้ `print(content)` เพื่อแสดงข้อมูลทั้งหมดในไฟล์

```
# การอ่านไฟล์ข้อมูลทั้งหมด
with open("example.txt", "r") as file:
    content = file.read()
    print("เนื้อหาในไฟล์ทั้งหมด:")
    print(content)
```

ผลลัพธ์

เนื้อหาในไฟล์ทั้งหมด:

Alice,90

Bob,85

Charlie,78

David,92

Eve,88

2. การอ่านทีละบรรทัดการอ่านไฟล์ข้อความทั้งหมด for line in file: ใช้ลูปอ่านข้อมูลในไฟล์ทีละบรรทัด line.strip() ลบช่องว่างหรือ newline (\n) ที่อยู่ท้ายข้อความในแต่ละบรรทัด เพื่อให้ข้อความมีรูปแบบที่สะอาด และข้อมูลในแต่ละบรรทัดจะถูกพิมพ์ออกมาตามลำดับ

```
# การอ่านไฟล์ทีละบรรทัด
```

```
with open("example.txt", "r") as file:
    print("เนื้อหาในไฟล์ทีละบรรทัด:")
    for line in file:
        print(line.strip())
```

ผลลัพธ์

เนื้อหาในไฟล์ทีละบรรทัด:

Alice,90

Bob,85

Charlie,78

David,92

Eve,88

3. การใช้งานร่วมกับการประมวลผลข้อมูล เมื่ออ่านข้อมูลจากไฟล์แล้ว สามารถนำข้อมูลนั้นไปประมวลผลเพิ่มเติมได้ เช่น การแยกชื่อและคะแนน

```
# อ่านไฟล์และแปลงข้อมูลเป็นรายชื่อและคะแนน
```

```
with open("example.txt", "r") as file:
    students = {}
    for line in file:
        name, score = line.strip().split(",")
        students[name] = int(score)

print("รายชื่อนักเรียนและคะแนน:")
for name, score in students.items():
    print(f"{name}: {score}")
```

- ใช้ line.strip().split(",") แยกชื่อและคะแนนในแต่ละบรรทัด (โดยใช้ , เป็นตัวแบ่ง)
- แปลงคะแนนเป็นจำนวนเต็ม (int(score))
- เก็บข้อมูลในรูปแบบ Dictionary (students) โดยชื่อเป็น Key และคะแนนเป็น Value

ผลลัพธ์

เนื้อหาในไฟล์ทั้งหมด:

Alice : 90

Bob : 85

Charlie : 78

David : 92

Eve : 88

6.2.2 การเขียนไฟล์ข้อความ (.txt) และบันทึกข้อมูล

ใน Python การเขียนไฟล์ข้อความสามารถทำได้โดยใช้ฟังก์ชัน open() เพื่อเปิดไฟล์ในโหมดเขียน ("w" สำหรับเขียนใหม่ หรือ "a" สำหรับเขียนต่อ) และใช้ฟังก์ชัน write() หรือ writelines() เพื่อเขียนข้อมูลลงในไฟล์ สมมติว่าต้องการบันทึกข้อมูลรายชื่อนักเรียนและคะแนนลงในไฟล์ชื่อ output.txt

```
# รายชื่อนักเรียนและคะแนน
students = {
    "Alice": 90,
    "Bob": 85,
    "Charlie": 78,
    "David": 92,
    "Eve": 88
}

# การเขียนไฟล์ข้อความ
with open("output.txt", "w") as file:
    for name, score in students.items():
        file.write(f"{name},{score}\n")

print("ข้อมูลถูกบันทึกลงในไฟล์output.txt แล้ว")
```

คำอธิบาย

1. เปิดไฟล์ในโหมดเขียน open("output.txt", "w") จะเปิดไฟล์ชื่อ output.txt ในโหมดเขียนใหม่ (ถ้าไฟล์นี้มีอยู่แล้ว ข้อมูลเดิมจะถูกลบ) ใช้ with เพื่อจัดการไฟล์ให้ปิดอัตโนมัติหลังจากการเขียนเสร็จ

2. เขียนข้อมูลลงในไฟล์ ใช้ลูป for เพื่อวนซ้ำข้อมูลใน students และแต่ละรอบลูป ใช้ file.write(f"{name},{score}\n") เพื่อเขียนข้อมูลของนักเรียนและคะแนน โดยใช้เครื่องหมาย , คั่นระหว่างชื่อ และคะแนน

3. แสดงข้อความยืนยัน พิมพ์ข้อความแจ้งให้ผู้ใช้ทราบว่าข้อมูลถูกบันทึกลงไฟล์เรียบร้อยแล้ว ตัวอย่างไฟล์ output.txt

Alice,90

Bob,85

Charlie,78

David,92

Eve,88

การเขียนข้อมูลต่อท้ายไฟล์

หากต้องการเพิ่มข้อมูลใหม่ลงในไฟล์เดิมโดยไม่ลบข้อมูลเดิม สามารถใช้โหมด "a" (append) ได้

```
# เพิ่มข้อมูลใหม่ลงในไฟล์
new_students = {
    "Frank": 80,
    "Grace": 95
}

with open("output.txt", "a") as file:
    for name, score in new_students.items():
        file.write(f"{name},{score}\n")

print("ข้อมูลเพิ่มเติมถูกเพิ่มลงในไฟล์ output.txt แล้ว")
```

ใช้โหมด "a" เพื่อเปิดไฟล์ในโหมดเขียนต่อ (append) และข้อมูลใหม่ (new_students) จะถูกเพิ่มลง "เปทิ่ห้ายไฟล์โดยไม่ลบข้อมูลเดิม"

ตัวอย่างไฟล์ที่ได้หลังการเพิ่มข้อมูล output.txt

```
Alice,90
Bob,85
Charlie,78
David,92
Eve,88
Frank,80
Grace,95
```

6.2.3 การอ่านและเขียนไฟล์ CSV โดยใช้โมดูล csv

ไฟล์ CSV (Comma-Separated Values) เป็นรูปแบบไฟล์ที่ใช้จัดเก็บข้อมูลในรูปแบบตาราง โดยค่าต่าง ๆ ในแต่ละแถวถูกคั่นด้วยเครื่องหมายจุลภาค (,) Python มีโมดูล csv ซึ่งช่วยให้การอ่านและเขียนไฟล์ CSV ทำได้ง่ายและสะดวก ตัวอย่างด้านล่างจะแสดงวิธีการใช้งาน

1. การอ่านไฟล์ CSV สมมติว่ามีไฟล์ชื่อ students.csv ที่มีเนื้อหาดังนี้

ตัวอย่างไฟล์ students.csv

```
Name,Math,Science,English
Alice,90,85,88
Bob,78,80,82
Charlie,92,88,85
David,85,91,89
Eve,88,84,90
```

มุมมองไฟล์ในโปรแกรม Excel

	A	B	C	D	E	
1	Name	Math	Science	English		
2	Alice	90	85	88		
3	Bob	78	80	82		
4	Charlie	92	88	85		
5	David	85	91	89		
6	Eve	88	84	90		
7						
8						

ตัวอย่างโค้ดสำหรับอ่านไฟล์ CSV

```
# การอ่านไฟล์ข้อความทั้งหมด
with open("example.txt", "r") as file:
    content = file.read()
    print("เนื้อหาในไฟล์ทั้งหมด:")
    print(content)
```

อธิบายโค้ด

- เปิดไฟล์ CSV `open("students.csv", "r")` เปิดไฟล์ในโหมดอ่าน และใช้ `with` เพื่อให้ไฟล์ถูกปิดอัตโนมัติเมื่ออ่านเสร็จ
- ใช้ `csv.reader` โดยโมดูล `csv.reader` ช่วยแปลงแต่ละบรรทัดในไฟล์เป็นรายการ (list) โดยแยกค่าด้วยเครื่องหมาย逗號 (ในที่นี้คือ ,)
- วนลูปอ่านแต่ละแถว ใช้ `for row in reader` เพื่ออ่านแต่ละแถวในไฟล์ และแสดงผลออกมาทีละแถวผลลัพธ์

```
['Name', 'Math', 'Science', 'English']
['Alice', '90', '85', '88']
['Bob', '78', '80', '82']
['Charlie', '92', '88', '85']
['David', '85', '91', '89']
['Eve', '88', '84', '90']
```

2. การเขียนไฟล์ CSV ตัวอย่างการเขียนข้อมูลลงไฟล์ CSV ดังนี้

```
import csv

# ข้อมูลนักเรียน
students = [
    ["Name", "Math", "Science", "English"],
    ["Alice", 90, 85, 88],
    ["Bob", 78, 80, 82],
    ["Charlie", 92, 88, 85],
    ["David", 85, 91, 89],
    ["Eve", 88, 84, 90]
]
```

```
# การเขียนไฟล์ CSV
with open("output.csv", "w", newline="") as csvfile:
    writer = csv.writer(csvfile)
    writer.writerows(students)

print("ข้อมูลถูกบันทึกลงในไฟล์ output.csv แล้ว")
```

อธิบายโค้ด

- เปิดไฟล์ในโหมดเขียน open("output.csv", "w", newline="") เปิดไฟล์ในโหมดเขียน ("w") พร้อมระบุ newline="" เพื่อป้องกันการเพิ่มบรรทัดว่าง
- ใช้ csv.writer โมดูล csv.writer ใช้เขียนข้อมูลลงในไฟล์ CSV
- เขียนหลายแถวพร้อมกัน ใช้ writer.writerows(students) เพื่อเขียนข้อมูลจากลิสต์ของลิสต์ลงในไฟล์ ตัวอย่างไฟล์ output.csv

```
Name,Math,Science,English
Alice,90,85,88
Bob,78,80,82
Charlie,92,88,85
David,85,91,89
Eve,88,84,90
```

6.2.4 การจัดการไฟล์ JSON ด้วยโมดูล json

ไฟล์ JSON (JavaScript Object Notation) เป็นรูปแบบไฟล์ที่ใช้จัดเก็บข้อมูลในโครงสร้างแบบ key-value หรือ array ที่สามารถอ่านและเขียนได้ง่ายใน Python ซึ่ง JSON เป็นรูปแบบการแลกเปลี่ยนข้อมูลที่เรียบง่ายและรองรับหลายภาษา ทำให้เหมาะสมสำหรับการใช้งานใน RESTful APIs เพื่อเชื่อมต่อระหว่างเซิร์ฟเวอร์และคลาวน์ต่อย่างมีประสิทธิภาพ นอกจากนี้ JSON ยังมีขนาดเล็ก ประมวลผลได้รวดเร็ว และรองรับโครงสร้างข้อมูลที่ซับซ้อน เช่น objects และ arrays ได้อย่างยืดหยุ่น โดยใช้โมดูล json ซึ่งเป็นโมดูลมาตรฐานใน Python ที่ช่วยให้สามารถจัดการข้อมูล JSON ได้อย่างมีประสิทธิภาพ ตัวอย่างต่อไปนี้จะแสดงวิธีการใช้งาน

1. การเขียนไฟล์ JSON สมมติว่ามีข้อมูลของนักเรียนที่ต้องการบันทึกลงในไฟล์ JSON

```
import json

# ข้อมูลนักเรียน
students = {
    "students": [
        {"name": "Alice", "math": 90, "science": 85, "english": 88},
        {"name": "Bob", "math": 78, "science": 80, "english": 82},
        {"name": "Charlie", "math": 92, "science": 88, "english": 85},
        {"name": "David", "math": 85, "science": 91, "english": 89},
```

```

        {"name": "Eve", "math": 88, "science": 84, "english": 90}
    ]
}

# การเขียนไฟล์ JSON
with open("students.json", "w") as jsonfile:
    json.dump(students, jsonfile, indent=4)

print("ข้อมูลถูกบันทึกลงในไฟล์ students.json แล้ว")

```

การนำเข้าโมดูล json: ใช้โมดูล json สำหรับจัดการข้อมูลในรูปแบบ JSON ใช้ json.dump(): เขียนข้อมูลในรูปแบบ JSON ลงในไฟล์ โดย indent=4 ช่วยให้ข้อมูลจัดรูปแบบ (pretty print) เพื่อให้อ่านง่ายตัวอย่างไฟล์ students.json

```

{
    "students": [
        {
            "name": "Alice",
            "math": 90,
            "science": 85,
            "english": 88
        },
        {
            "name": "Bob",
            "math": 78,
            "science": 80,
            "english": 82
        },
        {
            "name": "Charlie",
            "math": 92,
            "science": 88,
            "english": 85
        },
        {
            "name": "David",
            "math": 85,
            "science": 91,
            "english": 89
        },
        {
            "name": "Eve",
            "math": 88,
            "science": 84,
            "english": 90
        }
    ]
}

```

2. การอ่านไฟล์ JSON โดยทำการอ่านอ่านข้อมูลในไฟล์ students.json

```
import json

# การอ่านไฟล์ JSON
with open("students.json", "r") as jsonfile:
    data = json.load(jsonfile)

# แสดงข้อมูล
print("ข้อมูลนักเรียนจากไฟล์ JSON:")
for student in data["students"]:
    print(f"ชื่อ: {student['name']}, คณิตศาสตร์: {student['math']}, วิทยาศาสตร์: {student['science']}, ภาษาอังกฤษ: {student['english']}")
```

ใช้ json.load() อ่านข้อมูล JSON จากไฟล์และแปลงเป็นโครงสร้างข้อมูล Python เช่น dictionary หรือ list และการใช้ลูป เข้าถึงข้อมูลแต่ละรายการใน dictionary เพื่อแสดงผล

ผลลัพธ์

ข้อมูลนักเรียนจากไฟล์ JSON:

```
ชื่อ: Alice, คณิตศาสตร์: 90, วิทยาศาสตร์: 85, ภาษาอังกฤษ: 88
ชื่อ: Bob, คณิตศาสตร์: 78, วิทยาศาสตร์: 80, ภาษาอังกฤษ: 82
ชื่อ: Charlie, คณิตศาสตร์: 92, วิทยาศาสตร์: 88, ภาษาอังกฤษ: 85
ชื่อ: David, คณิตศาสตร์: 85, วิทยาศาสตร์: 91, ภาษาอังกฤษ: 89
ชื่อ: Eve, คณิตศาสตร์: 88, วิทยาศาสตร์: 84, ภาษาอังกฤษ: 90
```

3. การแก้ไขข้อมูลในไฟล์ JSON โดยเพิ่มข้อมูลนักเรียนใหม่ในไฟล์ students.json

```
import json
# การอ่านข้อมูลจากไฟล์ JSON
with open("students.json", "r") as jsonfile:
    data = json.load(jsonfile)

# เพิ่มข้อมูลใหม่
new_student = {"name": "Frank", "math": 82, "science": 87, "english": 84}
data["students"].append(new_student)

# บันทึกข้อมูลกลับลงไฟล์
with open("students.json", "w") as jsonfile:
    json.dump(data, jsonfile, indent=4)

print("ข้อมูลนักเรียนใหม่ถูกเพิ่มลงในไฟล์ students.json แล้ว")
```

เพิ่มข้อมูลใหม่โดยใช้ append() เพิ่ม dictionary ใหม่เข้าไปใน list ของ students และบันทึกข้อมูลกลับลงไฟล์โดยใช้ json.dump() เพื่อเขียนข้อมูลที่แก้ไขกลับลงไฟล์

ตัวอย่างไฟล์ students.json หลังการแก้ไข

```
{
    "students": [
        {
            "name": "Alice",
            "math": 90,
            "science": 85,
            "english": 88
        },
        ...
        {
            "name": "Frank",
            "math": 82,
            "science": 87,
            "english": 84
        }
    ]
}
```

การจัดการไฟล์ JSON ใน Python ใช้โมดูล json เพื่ออ่าน เขียน และแก้ไขข้อมูลได้อย่างสะดวก พงก์ชั้นหลักที่ใช้คือ json.dump() สำหรับการเขียนข้อมูล และ json.load() สำหรับการอ่านข้อมูล JSON เหมาะสมสำหรับการจัดเก็บข้อมูลโครงสร้าง เช่น key-value และ list ที่เข้าใจง่ายและมีประสิทธิภาพในการใช้งาน

ในการทำโปรเจกต์จริง ข้อมูลจำนวนมากมักถูกเก็บอยู่ในไฟล์ เช่น CSV หรือ JSON ดังนั้นการเข้าใจวิธีอ่านและเขียนไฟล์ใน Python เป็นสิ่งจำเป็น ผู้แต่งมีความชอบที่ Python ออกแบบให้การจัดการไฟล์เป็นเรื่องง่ายและตรงไปตรงมา เช่น พงก์ชั้น open() หรือโมดูล csv และ json ช่วยให้ทำงานกับข้อมูลได้อย่างรวดเร็วและมีประสิทธิภาพมากขึ้น

6.3 การประมวลผลข้อมูลจากไฟล์ด้วย Pandas

Pandas เป็นไลบรารีสำหรับการจัดการและประมวลผลข้อมูลในรูปแบบตารางที่ทรงพลังและใช้งานง่าย โดยเฉพาะเมื่อต้องทำงานกับข้อมูลจำนวนมากหรือข้อมูลที่ซับซ้อน Pandas ช่วยให้การอ่าน เขียน และประมวลผลข้อมูลจากไฟล์ เช่น CSV Excel หรือ JSON มีประสิทธิภาพยิ่งขึ้น

6.3.1 การอ่านข้อมูลจำนวนมาจากการไฟล์และประมวลผลเบื้องต้น

ตัวอย่างไฟล์ student.csv

Name	Math	Science	English
Alice	90	85	88
Bob	78	80	82
Charlie	92	88	85
David	85	91	89

โค้ดการอ่านไฟล์

```
import pandas as pd

# อ่านข้อมูลจากไฟล์ CSV
df = pd.read_csv("students.csv")

# แสดงข้อมูล 5 แถวแรก
print(df.head())

# ตรวจสอบข้อมูลเบื้องต้น
print("ข้อมูลทั้งหมด:")
print(df.info())
```

- pd.read_csv() ใช้สำหรับอ่านข้อมูลจากไฟล์ CSV
- df.head() แสดงข้อมูลตัวอย่าง 5 แถวแรก
- df.info() ให้ข้อมูลสรุปเกี่ยวกับโครงสร้างของ DataFrame เช่น จำนวนคอลัมน์และชนิดข้อมูล

ผลลัพธ์

```
ข้อมูลทั้งหมด:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
--- 
 0   Name      4 non-null      object  
 1   Math       4 non-null      int64   
 2   Science    4 non-null      int64   
 3   English    4 non-null      int64   
dtypes: int64(3), object(1)
memory usage: 256.0+ bytes
```

6.3.2 การกรองข้อมูลที่สนใจจากไฟล์

โดยการกรองคะแนนคณิตศาสตร์มากกว่า 80

```
import pandas as pd

# อ่านข้อมูลจากไฟล์ CSV
df = pd.read_csv("students.csv")

filtered_df = df[df["Math"] > 80]

print("นักเรียนที่ได้คะแนน Math มากกว่า 80:")
print(filtered_df)
```

- ใช้การกรองข้อมูลผ่านเงื่อนไข (`df["Math"] > 80`) เพื่อเลือกเฉพาะข้อมูลที่ต้องการ
- สามารถประยุกต์ใช้เงื่อนไขเพิ่มเติม เช่น `&` (and) หรือ `|` (or) ข้อมูลทั้งหมด: นักเรียนที่ได้คะแนน Math มากกว่า 80:

	Name	Math	Science	English
0	Alice	90	85	88
2	Charlie	92	88	85
3	David	85	91	89

6.3.3 การประมวลผลข้อมูลด้วยการคำนวณและสรุปผล

โค้ดการหาค่าเฉลี่ย

```
import pandas as pd
# อ่านข้อมูลจากไฟล์ CSV
df = pd.read_csv("students.csv")
# คำนวณค่าเฉลี่ยของคะแนนในแต่ละวิชา
average_scores = df[["Math", "Science", "English"]].mean()
print("ค่าเฉลี่ยของคะแนนแต่ละวิชา:")
print(average_scores)

# นับจำนวนแถวของข้อมูล
student_count = len(df)
print(f"จำนวนนักเรียนทั้งหมด: {student_count}")
```

- `df[["Math", "Science", "English"]].mean()` คำนวณค่าเฉลี่ยของแต่ละคอลัมน์ในกลุ่มที่เลือก
- `len(df)` ใช้สำหรับนับจำนวนแถวใน DataFrame

ค่าเฉลี่ยของคะแนนแต่ละวิชา:

```
Math      86.25
Science   86.00
English   86.00
dtype: float64
จำนวนนักเรียนทั้งหมด: 4
```

6.3.4 การเขียนข้อมูลผลลัพธ์กลับลงในไฟล์ใหม่

โค้ดการหาค่าเฉลี่ย

```
import pandas as pd
# อ่านข้อมูลจากไฟล์ CSV
df = pd.read_csv("students.csv")
# คำนวณค่าเฉลี่ยของคะแนนในแต่ละวิชา
average_scores = df[["Math", "Science", "English"]].mean()
print("ค่าเฉลี่ยของคะแนนแต่ละวิชา:")
print(average_scores)

# นับจำนวนแถวของข้อมูล
student_count = len(df)
print(f"จำนวนนักเรียนทั้งหมด: {student_count}")
```

- `to_csv()` ใช้สำหรับบันทึกข้อมูล DataFrame ลงในไฟล์ CSV
- `index=False` ใช้เพื่อไม่บันทึกชื่อ (index) ลงในไฟล์

ตัวอย่างไฟล์ filtered_students.csv

```
Name,Math,Science,English
```

```
Alice,90,85,88
```

```
Charlie,92,88,85
```

```
David,85,91,89
```

สำหรับผู้ตั้ง การประมวลผลข้อมูลไม่ใช่แค่การอ่านและแสดงผลข้อมูลเท่านั้น แต่เป็นการ "วิเคราะห์ข้อมูล" เพื่อค้นหาข้อมูลเชิงลึกที่ซ่อนอยู่ เช่น การคำนวณค่าเฉลี่ย การกรองข้อมูล หรือการสรุปผลข้อมูล การใช้ Pandas ช่วยให้การจัดการข้อมูลปริมาณมากทำได้ง่ายขึ้นมาก เมื่อเทียบกับการใช้ลูปใน Python แบบดั้งเดิม

6.4 การติดตั้งและใช้งานไลบรารีใน Python

Python มีระบบจัดการไลบรารีที่ทรงพลัง เช่น pip ซึ่งช่วยให้ผู้ใช้งานสามารถติดตั้ง อัปเดต และจัดการไลบรารีที่ต้องการได้อย่างสะดวก ในหัวข้อนี้จะอธิบายวิธีการติดตั้งและจัดการไลบรารี พร้อมตัวอย่างการใช้งานจริง

6.4.1 การติดตั้งไลบรารีด้วย pip

pip เป็นโปรแกรมจัดการแพ็คเกจจำมาตรฐานสำหรับ Python ที่ใช้ติดตั้งไลบรารีจาก Python Package Index (PyPI) ซึ่งเป็นแหล่งรวมไลบรารีริมماยกายสำหรับการพัฒนาโปรแกรม

- ใช้คำสั่ง pip install ตามด้วยชื่อไลบรารีที่ต้องการ แสดงหน้าต่างการติดตั้งผ่าน Command Prompt หรือ Power Shell ในระบบปฏิบัติการ Windows ดังรูปที่ 6.2

```
pip install requests
```

```
# clear sys_audit_hooks
PS C:\Users\baban> python --version
Python 3.12.8
PS C:\Users\baban> pip install pandas
WARNING: Ignoring invalid distribution ~ip (C:\Python312\lib\site-packages)
WARNING: Ignoring invalid distribution ~umpy (C:\Python312\lib\site-packages)
WARNING: Ignoring invalid distribution ~ip (C:\Python312\lib\site-packages)
WARNING: Ignoring invalid distribution ~umpy (C:\Python312\lib\site-packages)
Requirement already satisfied: pandas in c:\python312\lib\site-packages (2.2.0)
Requirement already satisfied: numpy<2,>=1.26.0 in c:\python312\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil<=2.8.2 in c:\python312\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\python312\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in c:\python312\lib\site-packages (from pandas) (2023.4)
Requirement already satisfied: six>=1.5 in c:\python312\lib\site-packages (from python-dateutil<=2.8.2->pandas) (1.16.0)
WARNING: Ignoring invalid distribution ~ip (C:\Python312\lib\site-packages)
WARNING: Ignoring invalid distribution ~umpy (C:\Python312\lib\site-packages)
WARNING: Ignoring invalid distribution ~ip (C:\Python312\lib\site-packages)
WARNING: Ignoring invalid distribution ~umpy (C:\Python312\lib\site-packages)
PS C:\Users\baban> |
```

รูปที่ 6.2 แสดงหน้าต่างการติดตั้ง

- สำหรับการติดตั้งไลบรารีเฉพาะเวอร์ชัน

```
install requests==2.28.0
```

- การติดตั้งหลายไลบรารีพร้อมกันจากไฟล์ requirements.txt

```
pip install -r requirements.txt
```

ไฟล์ requirements.txt อาจมีเนื้อหาดังนี้

```
requests==2.28.0
numpy>=1.22
pandas
```

6.4.2 การอัปเดตและจัดการเวอร์ชันของไลบรารี

บางครั้ง ไลบรารีที่ใช้อาจมีการอัปเดตเพื่อเพิ่มฟีเจอร์หรือแก้ไขข้อผิดพลาด การจัดการเวอร์ชันอย่างเหมาะสมช่วยให้ได้ทำงานได้อย่างมีประสิทธิภาพและปลอดภัย

- การอัปเดตไลบรารีเป็นเวอร์ชันล่าสุด ด้วย --upgrade

```
pip install --upgrade requests
```

- การตรวจสอบเวอร์ชันของไลบรารีที่ติดตั้ง ด้วยคำสั่ง show

```
pip show requests
```

ผลลัพธ์

```
Name: requests
```

```
Version: 2.28.0
```

```
Summary: Python HTTP for Humans.
```

- การถอนการติดตั้งไลบรารี ด้วยคำสั่ง uninstall

```
pip uninstall requests
```

- การแก้ไขเวอร์ชันไลบรารีเมื่อมีความขัดแย้ง หากไลบรารีสองตัวมีข้อขัดแย้งของเวอร์ชัน อาจต้องแก้ไข requirements.txt เพื่อรับเวอร์ชันที่เข้ากันได้ เช่น

```
flask>=2.0.0,<3.0.0
```

ไฟล์ requirements.txt อาจมีเนื้อหาดังนี้

6.4.3 การตรวจสอบไลบรารีที่ติดตั้งในระบบ

เมื่อต้องการดูรายการไลบรารีทั้งหมดที่ติดตั้งในระบบ สามารถใช้คำสั่ง pip เพื่อช่วยจัดการและตรวจสอบได้

- การแสดงรายการไลบรารีทั้งหมดที่ติดตั้ง

```
pip list
```

ผลลัพธ์

Package	Version
numpy	1.22.0
pandas	1.4.0
requests	2.28.0

- การตรวจสอบว่ามีการอัปเดตสำหรับไลบรารีหรือไม่

```
pip list --outdated
```

ผลลัพธ์

Package	Version	Latest	Type
numpy	1.22.0	1.23.0	wheel

3. การบันทึกรายการไลบรารีลงในไฟล์

```
pip freeze > requirements.txt
```

ไฟล์ requirements.txt จะมีเนื้อหาที่แสดงเวอร์ชันไลบรารีทั้งหมดในโปรเจกต์

```
numpy==1.22.0
```

```
pandas==1.4.0
```

```
requests==2.28.0
```

4. การตรวจสอบความเข้ากันของไลบรารีในโปรเจกต์ ใช้คำสั่งนี้เพื่อตรวจสอบความเข้ากันของไลบรารีใน requirements.txt

```
pip check
```

ในหัวข้อนี้ ผู้เรียนได้เรียนรู้วิธีติดตั้งไลบรารีด้วย pip วิธีอัปเดตและจัดการเวอร์ชันของไลบรารี และการตรวจสอบไลบรารีที่ติดตั้งในระบบ ทักษะเหล่านี้ช่วยให้สามารถจัดการสภาพแวดล้อมของโปรเจกต์ Python ได้อย่างมีประสิทธิภาพ

ผู้แต่งเห็นว่า pip เป็น "ผู้ช่วย" ที่ดีสำหรับโปรแกรมเมอร์ Python เพราะช่วยให้การติดตั้งไลบรารีใหม่ๆ ได้ภายในไม่กี่วินาที ยิ่งไปกว่านั้น การจัดการไลบรารีใหม่เวอร์ชันที่เหมาะสมกับโปรเจกต์ก็ช่วยลดปัญหาเรื่องการทำงานขัดแย้งระหว่างไลบรารีได้ดี นอกจากนี้ การรู้จักวิธีตรวจสอบและการ requirements.txt ยังเป็นทักษะที่จำเป็นมากในการทำงานเป็นทีม

6.5 การใช้งานไลบรารีที่ได้รับความนิยม

Python มีไลบรารีที่ทรงพลังและได้รับความนิยมอย่างมากในงานวิเคราะห์และจัดการข้อมูล รวมถึงการสร้างกราฟและการแสดงผลข้อมูล ไลบรารีเหล่านี้ช่วยให้นักพัฒนาสามารถทำงานได้อย่างมีประสิทธิภาพ โดยในหัวข้อนี้จะอธิบายถึงการใช้งานไลบรารีหลัก ได้แก่ NumPy Pandas และ Matplotlib พร้อมตัวอย่างการทดสอบการใช้งานในโปรเจกต์จริง

ก่อนเริ่มต้นใช้งานหัวข้อนี้ การใช้งานไลบรารีที่ได้รับความนิยม จะต้องติดตั้งไลบรารีที่เกี่ยวข้องเพื่อให้ Python สามารถใช้งานไลบรารี NumPy Pandas และ Matplotlib ได้ หากยังไม่ได้ติดตั้ง สามารถใช้คำสั่งต่อไปนี้

คำสั่งติดตั้ง

```
pip install numpy
pip install pandas
pip install matplotlib
หรือติดตั้งพร้อมกัน
```

```
pip install numpy pandas matplotlib
```

6.5.1 ใช้งานไลบรารี NumPy สำหรับการจัดการข้อมูลเชิงตัวเลข

NumPy (Numerical Python) เป็นไลบรารีที่ออกแบบมาสำหรับการคำนวณและจัดการข้อมูลเชิงตัวเลขที่มีประสิทธิภาพสูง รองรับการทำงานกับ arrays และฟังก์ชันทางคณิตศาสตร์ต่าง ๆ เช่น การบวก ลบ คูณ หาร เมทริกซ์ และฟังก์ชันทางสถิติ

การใช้งาน NumPy Array ดังนี้

1. การสร้างอาร์เรย์ (Array)

```
import numpy as np
array = np.array([1, 2, 3, 4, 5])
```

ใช้ฟังก์ชัน np.array() เพื่อสร้าง อาร์เรย์หนึ่งมิติ ที่เก็บข้อมูลตัวเลข [1, 2, 3, 4, 5] และ NumPy Array รองรับการคำนวณทางคณิตศาสตร์และฟังก์ชันเชิงตัวเลขได้อย่างมีประสิทธิภาพ

1. การคำนวณพื้นฐานในอาร์เรย์

```
print("อาร์เรย์:", array)
print("ผลรวม:", np.sum(array))
print("ค่าเฉลี่ย:", np.mean(array))
print("ค่าสูงสุด:", np.max(array))
print("ค่าน้อยสุด:", np.min(array))
```

- np.sum(array) คือ คำนวณผลรวมของค่าทั้งหมดในอาร์เรย์ ($1+2+3+4+5 = 15$)
- np.mean(array) คือ คำนวณค่าเฉลี่ยของค่าทั้งหมดในอาร์เรย์ (ผลรวม \div จำนวนสมาชิก $= 15 \div 5 = 3.0$)
- np.max(array) คือ หาค่าสูงสุดในอาร์เรย์ (5)
- np.min(array) คือ หาค่าน้อยสุดในอาร์เรย์ (1)

3. การสร้างเมทริกซ์ (Matrix) ใช้ฟังก์ชัน np.array() สร้าง อาร์เรย์สองมิติ (เมทริกซ์) ขนาด 3×3

```
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("เมทริกซ์:")
print(matrix)
```

ผลลัพธ์

```
[[1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]]
```

4. การคำนวณในเมทริกซ์

```
print("ผลรวมในแต่ละคอลัมน์:", np.sum(matrix, axis=0))
print("ผลรวมในแต่ละแถว:", np.sum(matrix, axis=1))
```

- `np.sum(matrix, axis=0)` คำนวณผลรวมของแต่ละคอลัมน์ในเมทริกซ์
 - คอลัมน์ที่ 1 ผลรวมคือ $1+4+7=12$
 - คอลัมน์ที่ 2 ผลรวมคือ $2+5+8=15$
 - คอลัมน์ที่ 3 ผลรวมคือ $3+6+9=18$
 - ผลลัพธ์ ผลรวมคือ `[12, 15, 18]`
- `np.sum(matrix, axis=1)` คำนวณผลรวมของแต่ละแถวแนวอนในเมทริกซ์
 - แถวแนวอนที่ 1 ผลรวมคือ $1+2+3=6$
 - แถวแนวอนที่ 2 ผลรวมคือ $4+5+6=15$
 - แถวแนวอนที่ 3 ผลรวมคือ $7+8+9=24$
 - ผลลัพธ์ ผลรวมคือ `[6, 15, 24]`

โดยทั้งนี้ List และ Numpy สามารถที่จะเก็บข้อมูลหลายค่า แต่ NumPy เหมาะกับงานที่ต้องการประสิทธิภาพสูง และการคำนวณตัวเลขในปริมาณมาก ส่วน List เหมาะสำหรับงานที่ไม่ได้นำเนื่องประสิทธิภาพหรือการคำนวณเชิงตัวเลข

ข้อแตกต่างระหว่าง List และ NumPy Array ดังนี้

1. โครงสร้างพื้นฐาน List สามารถเก็บข้อมูลได้หลายชนิดในลิสต์เดียว (heterogeneous) เช่น ตัวเลขและสตริงในการการเดียว แต่ NumPy Array เก็บข้อมูลชนิดเดียวกันทั้งหมด (homogeneous) เช่น ตัวเลขทั้งหมดในอาร์เรย์เดียว

```
my_list = [1, 2, 'a', 3.5] # เก็บข้อมูลหลายชนิดในลิสต์เดียว

import numpy as np
my_array = np.array([1, 2, 3, 4]) # ข้อมูลในอาร์เรย์ต้องเป็นชนิดเดียวกัน เช่น ตัวเลข
```

2. ประสิทธิภาพ List ทำงานช้ากว่า เพราะไม่มีการจัดการหน่วยความจำที่มีประสิทธิภาพ ขณะที่ NumPy Array ทำงานเร็วกว่า เพราะออกแบบมาสำหรับการจัดการข้อมูลจำนวนมาก

3. การคำนวณทางคณิตศาสตร์ List ใช้ลูปหรือฟังก์ชันคำนวณค่าทีละตัว ขณะที่ NumPy Array รองรับการคำนวณแบบเบกเตอร์ ทำให้คำนวณทั้งอาร์เรย์ได้พร้อมกัน

```
my_list = [1, 2, 3, 4]
doubled_list = [x * 2 for x in my_list] # ใช้ list comprehension
print(doubled_list)

my_array = np.array([1, 2, 3, 4])
doubled_array = my_array * 2 # การคำนวณแบบเบกเตอร์
```

4. ฟังก์ชันที่รองรับ List มีฟังก์ชันพื้นฐาน เช่น `len()` และ `sum()` ขณะที่ NumPy Array มีฟังก์ชันที่ซับซ้อน เช่น `mean()`, `sum()`, `std()`

```
import numpy as np
my_array = np.array([1, 2, 3, 4])
print("ค่าเฉลี่ย:", np.mean(my_array))
print("ผลรวม:", np.sum(my_array))
print("ค่าความเบี่ยงเบนมาตรฐาน:", np.std(my_array))
```

5. การจัดการหลายมิติ List ใช้ลิสต์ซ้อนลิสต์ ซึ่งการเข้าถึงข้อมูลหลายมิติทำได้ยากกว่า ขณะที่ NumPy Array รองรับข้อมูลหลายมิติโดยตรง ทำให้การจัดการสะดวกขึ้น

```
my_list = [[1, 2], [3, 4]]
print(my_list[1][1]) # ผลลัพธ์: 4
```

```
import numpy as np
my_array = np.array([[1, 2], [3, 4]])
print(my_array[1, 1]) # ผลลัพธ์: 4
```

6. การจัดการหน่วยความจำ List ใช้หน่วยความจำมากกว่า เพราะข้อมูลถูกจัดเก็บแยกกัน ขณะที่ NumPy Array ใช้หน่วยความจำน้อยกว่า เพราะข้อมูลถูกจัดเก็บในบล็อกหน่วยความจำเดียวกัน

ตารางที่ 6.1 ตารางสรุปคำสั่ง NumPy เป็นต้น

ฟังก์ชันการดำเนินการ	คำอธิบาย	ตัวอย่างโค้ด	ผลลัพธ์
np.array()	สร้างอาร์เรย์ (Array) จากลิสต์หรือข้อมูลอื่น ๆ	np.array([1, 2, 3])	[1 2 3]
np.zeros()	สร้างอาร์เรย์ที่มีค่าเริ่มต้นเป็นศูนย์	np.zeros((2, 3))	[[0. 0. 0.] [0. 0. 0.]]
np.ones()	สร้างอาร์เรย์ที่มีค่าเริ่มต้นเป็นหนึ่ง	np.ones((2, 2))	[[1. 1.] [1. 1.]]
np.arange(start, stop, step)	สร้างอาร์เรย์ของตัวเลขในช่วงที่กำหนด (คล้าย range() ใน Python)	np.arange(0, 10, 2)	[0 2 4 6 8]
np.linspace(start, stop, num)	สร้างอาร์เรย์ของตัวเลขที่แบ่งช่วงเป็นกุณ ค่าเท่า ๆ กัน	np.linspace(0, 1, 5)	[0. 0.25 0.5 0.75 1.]
np.reshape()	ปรับรูปร่างของอาร์เรย์ (เปลี่ยนขนาดแฉ-คอลัมน์)	np.array([1, 2, 3, 4]).reshape(2, 2)	[[1 2] [3 4]]
np.sum()	คำนวณผลรวมของค่าทั้งหมดในอาร์เรย์	np.sum([1, 2, 3])	6
np.mean()	คำนวณค่าเฉลี่ยของอาร์เรย์	np.mean([1, 2, 3])	2

ฟังก์ชันการดำเนินการ	คำอธิบาย	ตัวอย่างโค้ด	ผลลัพธ์
np.max()	ค้นหาค่าสูงสุดในอาร์เรย์	np.max([1, 2, 3])	3
np.min()	ค้นหาค่าน้อยสุดในอาร์เรย์	np.min([1, 2, 3])	1
np.std()	คำนวณค่าเบี่ยงเบนมาตรฐาน	np.std([1, 2, 3])	0.816497
np.sqrt()	คำนวณค่ารากที่สองของอาร์เรย์	np.sqrt([1, 4, 9])	[1. 2. 3.]
np.dot()	คำนวณจุดรวมระหว่างสองอาร์เรย์ (Dot Product)	np.dot([1, 2], [3, 4])	11
np.transpose()	transpose (Transpose) หรือสลับแถว เป็นคอลัมน์ในอาร์เรย์สมมติ	np.transpose([[1, 2], [3, 4]])	[[1 3] [2 4]]
np.random.rand()	สร้างอาร์เรย์ตัวเลขสุ่มในช่วง [0, 1)	np.random.rand(2, 2)	ตัวเลขสุ่ม เช่น [[0.1, 0.3]...]
np.random.randint()	สร้างตัวเลขสุ่มแบบจำนวนเต็มในช่วงที่กำหนด	np.random.randint(0, 10, 5)	ตัวเลขสุ่ม เช่น [3, 7, 2, 9, 1]
np.concatenate()	รวมอาร์เรย์หลายอันเข้าด้วยกัน (ต่อແລກ หรือคอลัมน์)	np.concatenate(([1, 2], [3, 4]))	[1 2 3 4]
np.shape	แสดงขนาดของอาร์เรย์ในรูป (rows, columns)	np.array([[1, 2], [3, 4]]).shape	(2, 2)
np.size	แสดงจำนวนสมาชิกทั้งหมดในอาร์เรย์	np.array([[1, 2], [3, 4]]).size	4
np.eye()	สร้างเมทริกซ์หน่วย (Identity Matrix)	np.eye(3)	[[1. 0. 0.] [0. 1. 0.] [0. 0. 1.]]

6.5.2 การใช้งานไลบรารี Pandas สำหรับการวิเคราะห์และประมวลผลข้อมูลในรูปแบบตาราง

จากหัวข้อที่ 6.3 ได้มีการกล่าวถึง Pandas เป็นต้น ที่เป็นไลบรารีใน Python ที่มีเครื่องมือสำหรับจัดการข้อมูลในรูปแบบ DataFrame และ Series ได้อย่างมีประสิทธิภาพ เหมาะสำหรับการประมวลผลและวิเคราะห์ข้อมูล เช่น การกรอง การรวมข้อมูล การคำนวณทางสถิติ และการแสดงผลข้อมูล ในหัวข้อนี้แสดงการใช้งาน Pandas เพิ่มเติม โดยข้อมูลที่ใช้ในการอ่านเป็นข้อมูลเกี่ยวกับอำเภอในจังหวัดกาฬสินธุ์ ดังนี้

ไฟล์ kalasin.csv แสดงในรูปแบบตารางใน Excel ได้ดังนี้

1	รหัสอำเภอ	อำเภอ	รหัสไปรษณีย์	จำนวนประชากร (ประมาณ)	LAT	LNG	
2	4601	เมืองกาฬสินธุ์	46000	150000	16.43	103.5078	
3	4602	นาหมก	46230	40000	16.25	103.8193	
4	4603	กุจินารายณ์	46110	120000	16.51	104.0536	
5	4604	กมลาไสย	46130	70000	16.3	103.5886	
6	4605	ยางตลาด	46120	110000	16.4	103.2922	
7	4606	สมเด็จ	46150	60000	16.62	103.3812	
8	4607	หัวยผึ้ง	46240	35000	16.48	103.0447	
9	4608	สหัสขันธ์	46140	50000	16.73	103.4639	
10	4609	คำม่วง	46180	30000	16.68	103.0621	
11	4610	ท่าคันโถ	46190	40000	16.79	103.1167	
12	4611	หนองกุงศรี	46220	45000	16.54	103.0064	
13	4612	สามชัย	46180	25000	16.7	103.2141	
14	4613	นาคู	46160	20000	16.48	103.2613	
15	4614	ดอนจาน	46000	15000	16.43	103.3472	
16	4615	น้องชัย	46130	20000	16.6	103.1939	

แสดงไฟล์ kalasin.csv ใน Text Editor เช่น Notepad ได้ดังนี้

รหัสอำเภอ,อำเภอ,รหัสไปรษณีย์,จำนวนประชากร (ประมาณ),LAT,LNG

4601,เมืองกาฬสินธุ์,46000,150000,16.4336,103.5078

4602,นาหมก,46230,40000,16.2511,103.8193

4603,กุจินารายณ์,46110,120000,16.5127,104.0536

4604,กมลาไสย,46130,70000,16.2966,103.5886

4605,ยางตลาด,46120,110000,16.4019,103.2922

4606,สมเด็จ,46150,60000,16.6195,103.3812

4607,หัวยผึ้ง,46240,35000,16.4798,103.0447

4608,สหัสขันธ์,46140,50000,16.7283,103.4639

4609,คำม่วง,46180,30000,16.6777,103.0621

4610,ท่าคันโถ,46190,40000,16.7928,103.1167

4611,หนองกุงศรี,46220,45000,16.5435,103.0064

4612,สามชัย,46180,25000,16.7011,103.2141

4613,นาคู,46160,20000,16.4792,103.2613

4614,ดอนจาน,46000,15000,16.4289,103.3472

4615,น้องชัย,46130,20000,16.5952,103.1939

การอ่านไฟล์ kalasin.csv ด้วย pandas

```
import pandas as pd
df = pd.read_csv("kalasin.csv")
print(df)
```

ตัวอย่างการใช้งาน Pandas ด้วยข้อมูลอำเภอในจังหวัดกาฬสินธุ์

```
import pandas as pd

# สร้าง DataFrame จากข้อมูลอำเภอในจังหวัดกาฬสินธุ์
data = {
    "รหัสอำเภอ": [
        4601, 4602, 4603, 4604, 4605, 4606, 4607, 4608, 4609, 4610, 4611,
        4612, 4613, 4614, 4615
    ],
    "อำเภอ": [
        "เมืองกาฬสินธุ์", "นาหมื่น", "กุฉินารายณ์", "กมลาไสย", "ยางคลาด",
        "สมเด็จ", "ห้วยตึง", "สหสันต์", "คำเมือง", "ท่าคันโภ",
        "หนองกุงศรี", "สามชัย", "นาฎ", "ดอนchan", "เมืองชัย"
    ],
    "รหัสไปรษณีย์": [
        46000, 46230, 46110, 46130, 46120,
        46150, 46240, 46140, 46180, 46190,
        46220, 46180, 46160, 46000, 46130
    ],
    "จำนวนประชากร (ประมาณ)": [
        150000, 40000, 120000, 70000, 110000,
        60000, 35000, 50000, 30000, 40000,
        45000, 25000, 20000, 15000, 20000
    ]
}
df = pd.DataFrame(data)

# แสดงข้อมูล
print("ข้อมูลทั้งหมด:")
print(df)
```

ผลลัพธ์

	รหัสอำเภอ	อำเภอ	รหัสไปรษณีย์	จำนวนประชากร (ประมาณ)	LAT	LNG
0	4601	เมืองกาฬสินธุ์	46000	150000	16.4336	103.5078
1	4602	นาหมื่น	46230	40000	16.2511	103.8193
2	4603	กุฉินารายณ์	46110	120000	16.5127	104.0536

3	4604	กมลาไสย	46130	70000	16.2966	103.5886
4	4605	ยางตลาด	46120	110000	16.4019	103.2922
5	4606	สมเด็จ	46150	60000	16.6195	103.3812
6	4607	ห้วยผึ้ง	46240	35000	16.4798	103.0447
7	4608	สหสันธ์	46140	50000	16.7283	103.4639
8	4609	คำม่วง	46180	30000	16.6777	103.0621
9	4610	ท่าคันโถ	46190	40000	16.7928	103.1167
10	4611	หนองกุ่งศรี	46220	45000	16.5435	103.0064
11	4612	สามชัย	46180	25000	16.7011	103.2141
12	4613	นาคุ	46160	20000	16.4792	103.2613
13	4614	ดอนจาน	46000	15000	16.4289	103.3472
14	4615	ฉ่องชัย	46130	20000	16.5952	103.1939

การประมวลผลและวิเคราะห์ข้อมูลโดยใช้ Pandas ดังนี้

1. กรองข้อมูล กรองข้อมูลเพื่อหาข้อมูลอำเภอที่มีประชากรเกิน 50,000 คน

```
filtered_df = df[df["จำนวนประชากร (ประมาณ)"] > 50000]
print("อำเภอที่มีประชากรมากกว่า 50,000 คน:")
print(filtered_df)
```

ผลลัพธ์

รหัสอำเภอ	อำเภอ	รหัสไปรษณีย์	จำนวนประชากร (ประมาณ)	LAT	LNG
0	4601	เมืองกาฬสินธุ์	46000	150000	16.4336
2	4603	กุฉินารายณ์	46110	120000	16.5127
3	4604	กมลาไสย	46130	70000	16.2966
4	4605	ยางตลาด	46120	110000	16.4019
5	4606	สมเด็จ	46150	60000	16.6195

2. คำนวณค่าเฉลี่ย คำนวณค่าเฉลี่ยของจำนวนประชากรในแต่ละอำเภอ

```
average_population = df["จำนวนประชากร (ประมาณ)"].mean()
print(f"ค่าเฉลี่ยของจำนวนประชากร: {average_population:.0f}")
```

ผลลัพธ์

ค่าเฉลี่ยของจำนวนประชากร: 54,333

3. เรียงลำดับข้อมูล เรียงลำดับอำเภอจากจำนวนประชากรมากที่สุดไปหน้าอยู่ที่สุด

```
sorted_df = df.sort_values(by="จำนวนประชากร (ประมาณ)", ascending=False)
print("เรียงลำดับอำเภอตามจำนวนประชากร:")
print(sorted_df)
```

ผลลัพธ์

รหัสอำเภอ	อำเภอ	รหัสไปรษณีย์	จำนวนประชากร (ประมาณ)	LAT	LNG
0	4601	เมืองกาฬสินธุ์	46000	150000	16.4336 103.5078
2	4603	กุดน้ำรายณ์	46110	120000	16.5127 104.0536
4	4605	ยางตลาด	46120	110000	16.4019 103.2922
3	4604	กมลาไสย	46130	70000	16.2966 103.5886
5	4606	สมเด็จ	46150	60000	16.6195 103.3812
...					

4. หาข้อมูลสถิติของจำนวนประชากร คำนวณค่าสูงสุด (Max) ค่าต่ำสุด (Min) และส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation) ของประชากร

```
max_population = df[ "จำนวนประชากร (ประมาณ)" ].max()
min_population = df[ "จำนวนประชากร (ประมาณ)" ].min()
std_population = df[ "จำนวนประชากร (ประมาณ)" ].std()

print(f"จำนวนประชากรมากที่สุด: {max_population:,}")
print(f"จำนวนประชากรน้อยที่สุด: {min_population:,}")
print(f"ส่วนเบี่ยงเบนมาตรฐานของประชากร: {std_population:.2f}")
```

ผลลัพธ์

จำนวนประชากรมากที่สุด: 150,000
จำนวนประชากรน้อยที่สุด: 15,000
ส่วนเบี่ยงเบนมาตรฐานของประชากร: 44,258.21

5. การเพิ่มคอลัมน์ใหม่ เพิ่มคอลัมน์ "ประชากร (หน่วยพัน)" โดยแสดงจำนวนประชากรในหน่วยพัน

```
df[ "ประชากร (หน่วยพัน)" ] = df[ "จำนวนประชากร (ประมาณ)" ] / 1000
print("เพิ่มคอลัมน์ประชากรในหน่วยพัน:")
print(df)
```

ผลลัพธ์

รหัสอำเภอ	อำเภอ	รหัสไปรษณีย์	จำนวนประชากร (ประมาณ)	LAT	LNG	ประชากร (หน่วยพัน)
0	4601	เมืองกาฬสินธุ์	46000	150000	16.4336 103.5078	150.0
1	4602	นามน	46230	40000	16.2511 103.8193	40.0
2	4603	กุดน้ำรายณ์	46110	120000	16.5127 104.0536	120.0
3	4604	กมลาไสย	46130	70000	16.2966 103.5886	70.0
4	4605	ยางตลาด	46120	110000	16.4019 103.2922	110.0
...						

6. การจัดกลุ่มข้อมูล นับจำนวนอำเภอที่มีรหัสไปรษณีย์เหมือนกัน

```
grouped_data = df.groupby("รหัสไปรษณีย์").size()
print("จำนวนอำเภอต่อรหัสไปรษณีย์:")
print(grouped_data)
```

ผลลัพธ์

จำนวนอำเภอต่อรหัสไปรษณีย์:

รหัสไปรษณีย์

46000 2

46110 1

46120 1

46130 2

46140 1

46150 1

46160 1

46180 2

46190 1

46220 1

46230 1

46240 1

dtype: int64

Pandas ช่วยให้สามารถวิเคราะห์ข้อมูลในรูปแบบตารางได้อย่างง่ายดาย เช่น การกรอง การเรียงลำดับ การคำนวณสถิติ และการสร้างคอลัมน์ใหม่ สามารถจัดการข้อมูลขนาดใหญ่ได้อย่างมีประสิทธิภาพ และปรับแต่งผลลัพธ์ตามความต้องการ และการใช้ Pandas ช่วยลดความซับซ้อนของโค้ดและเพิ่มความสะดวกในการประมวลผลข้อมูลสำหรับการวิเคราะห์และแสดงผล

6.5.3 การใช้งานไลบรารี Matplotlib สำหรับการสร้างกราฟและการแสดงผลข้อมูล

Matplotlib เป็นไลบรารีที่ทรงพลังสำหรับการสร้างกราฟและการแสดงผลข้อมูลใน Python โดยรองรับการสร้างกราฟหลากหลายประเภท เช่น กราฟเส้น กราฟแท่ง กราฟวงกลม และกราฟการกระจาย (scatter plot) ในหัวข้อนี้จะแสดงตัวอย่างการสร้างกราฟจากข้อมูลเกี่ยวกับอำเภอในจังหวัดกาฬสินธุ์ ดังนี้

1. การสร้างกราฟแท่ง (Bar Chart)

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("kalasin.csv")

plt.figure(figsize=(10, 6))
plt.bar(df["อำเภอ"], df["จำนวนประชากร (ประมาณ)", color='skyblue')
plt.title("Number of Population by District", fontsize=16)

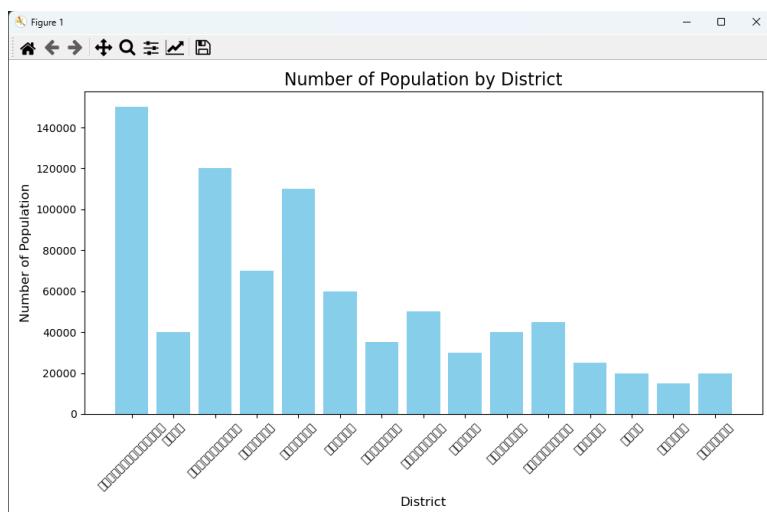
plt.xlabel("District", fontsize=12)
plt.ylabel("Number of Population", fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
```

```
plt.show()
```

อธิบายโค้ด

- plt.figure(figsize=(10, 6)) กำหนดขนาดของกราฟเป็นกว้าง 10 นิ้ว สูง 6 นิ้ว
- plt.bar(x, height, color='skyblue') สร้างกราฟแท่งโดย x เป็นข้อมูลชื่ออำเภอ (df["อำเภอ"]) และ height เป็นจำนวนประชากร (df["จำนวนประชากร (ประมาณ)"])
- plt.title() เพิ่มชื่อกราฟที่ด้านบน โดยกำหนดขนาดตัวอักษร (fontsize=16)
- plt.xlabel() และ plt.ylabel() เพิ่มป้ายชื่อแกน x และ y พร้อมกำหนดขนาดตัวอักษร
- plt.xticks(rotation=45) หมุนข้อความที่แกน x ให้อายุ 45 องศา เพื่อความชัดเจน
- plt.tight_layout() จัดการระยะห่างในกราฟให้เหมาะสม
- plt.show() แสดงกราฟ

ผลลัพธ์



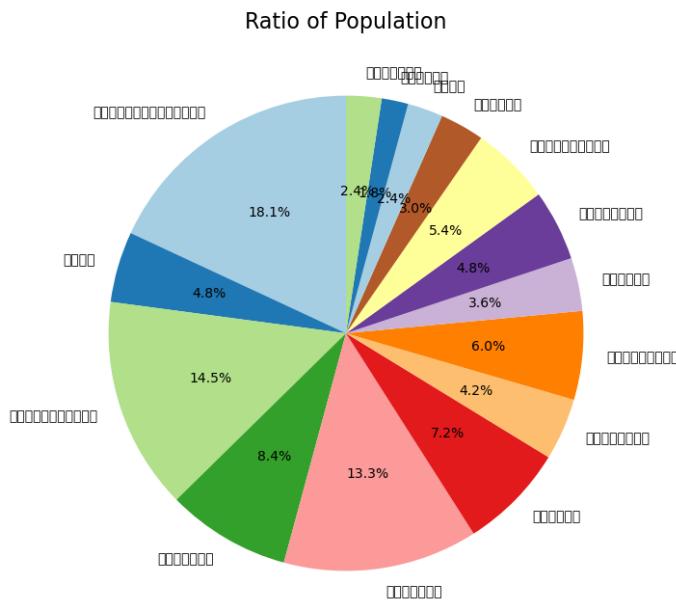
2. การสร้างกราฟวงกลม (Pie Chart)

```
plt.figure(figsize=(8, 8))
plt.pie(df["จำนวนประชากร (ประมาณ)"], labels=df["อำเภอ"], autopct='%1.1f%%',
startangle=90, colors=plt.cm.Paired.colors)
plt.title("Ratio of Population", fontsize=16)
plt.show()
```

อธิบายโค้ด

- plt.figure(figsize=(8, 8)) กำหนดขนาดกราฟเป็นสี่เหลี่ยมจัตุรัส (กว้างและสูง 8 นิ้ว)
- plt.pie(values, labels, autopct, startangle, colors) รายละเอียดดังนี้
 - values ข้อมูลจำนวนประชากร
 - labels ชื่ออำเภอ
 - autopct='%1.1f%%' แสดงเปอร์เซ็นต์ในแต่ละส่วนของวงกลม (1 ทศนิยม)
 - startangle=90 เริ่มต้นหมุนวงกลมจากมุม 90 องศา

- colors ใช้ชุดสี plt.cm.Paired.colors เพื่อเพิ่มสีที่หลากหลาย
- ผลลัพธ์



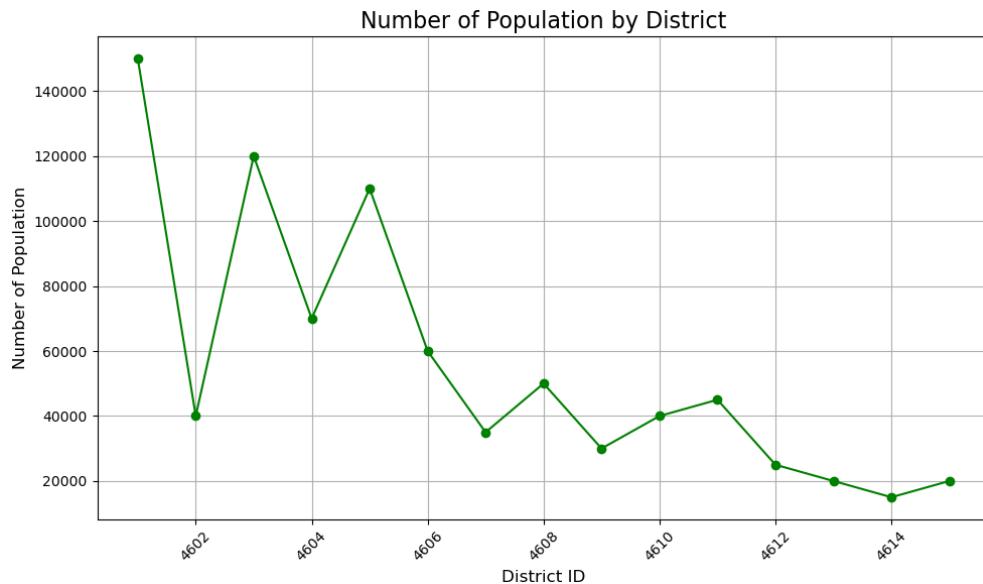
3. การสร้างกราฟเส้น (Line Chart)

```
plt.figure(figsize=(10, 6))
plt.plot(df["รหัสอำเภอ"], df["จำนวนประชากร (ประมาณ)", marker='o', linestyle='--',
color='green')
plt.title("Number of Population by District", fontsize=16)
plt.xlabel("District ID", fontsize=12)
plt.ylabel("Number of Population", fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

อธิบายโค้ด

- plt.plot(x, y, marker, linestyle, color): สร้างกราฟเส้นโดย
 - x: ข้อมูลชื่ออำเภอ
 - y: ข้อมูลจำนวนประชากร
 - marker='o': ใช้เครื่องหมายวงกลมที่จุดข้อมูล
 - linestyle='--': เส้นเข้มๆ มีจุดเป็นเส้นตรง
 - color='green': ใช้สีเขียว
- plt.grid(True): เพิ่มเส้นตารางในพื้นหลังกราฟ

ผลลัพธ์

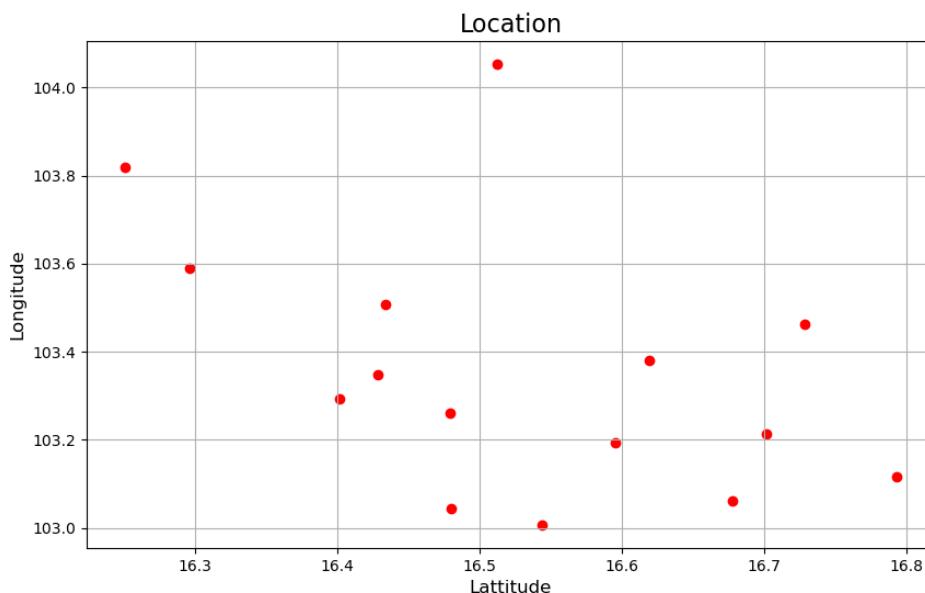


4. การสร้างกราฟการกระจาย (Scatter Plot)

```
plt.figure(figsize=(10, 6))
plt.scatter(df["LAT"], df["LNG"], color='red')
plt.title("Location", fontsize=16)
plt.xlabel("Latitude", fontsize=12)
plt.ylabel("Longitude", fontsize=12)
plt.grid(True)
plt.show()
```

อธิบายโค้ด plt.scatter(x, y, color): สร้างกราฟการกระจาย (scatter plot) โดยที่ x: ข้อมูลรหัสอำเภอ y: ข้อมูลจำนวนประชากร และ color='red': ใช้จุดสีแดง

ผลลัพธ์

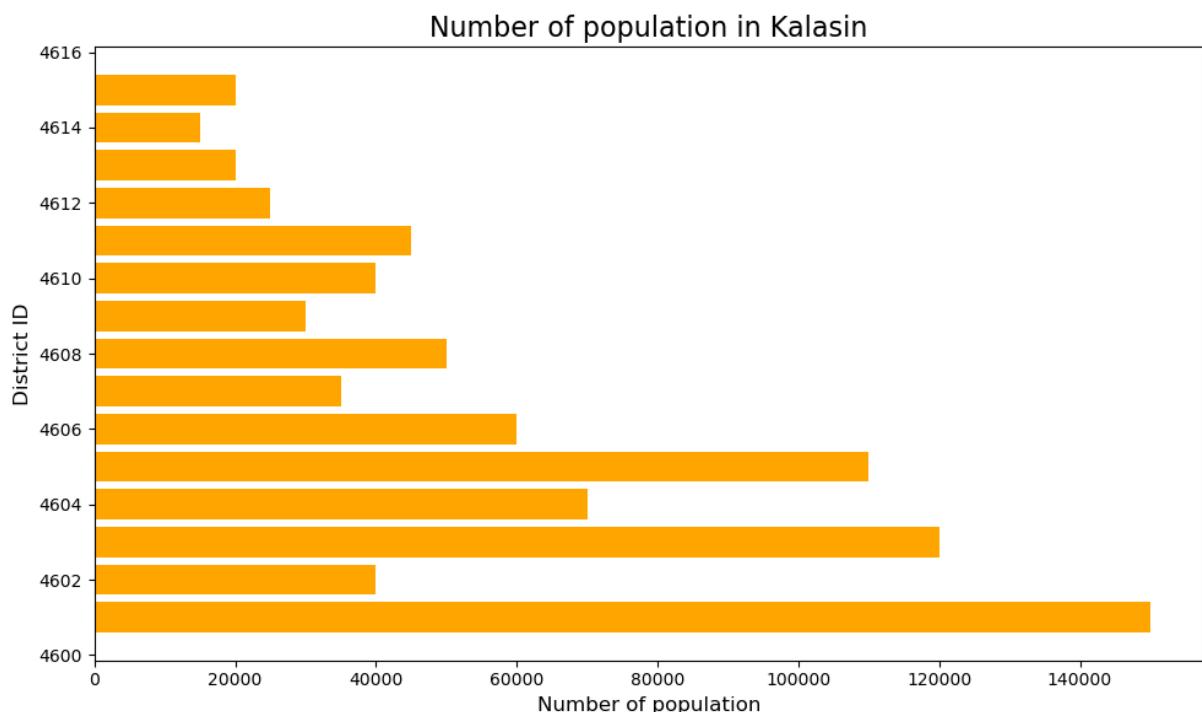


5. การเปรียบเทียบกราฟแท่งแนวตั้งและแนวนอน

```
plt.figure(figsize=(10, 6))
plt.barh(df["รหัสอำเภอ"], df["จำนวนประชากร (ประมาณ)", color='orange')
plt.title("Number of population in Kalasin", fontsize=16)
plt.xlabel("Number of population", fontsize=12)
plt.ylabel("District ID", fontsize=12)
plt.tight_layout()
plt.show()
```

อธิบายโค้ด plt.barh(y, width, color) สร้างกราฟแท่งแนวนอนโดย: y: ข้อมูลชื่ออำเภอ width: ข้อมูลจำนวนประชากร และ color='orange' ใช้สีส้ม

ผลลัพธ์



การแสดงผลข้อมูลด้วยกราฟช่วยให้วิเคราะห์ข้อมูลได้ง่ายขึ้น กราฟแท่ง เหมาะสำหรับเปรียบเทียบ ข้อมูลระหว่างหมวดหมู่ กราฟวงกลม ใช้แสดงสัดส่วนของแต่ละหมวดหมู่ และ กราฟเส้น แสดงแนวโน้มหรือ การเปลี่ยนแปลง กราฟการกระจาย (Scatter) ใช้แสดงความสัมพันธ์ระหว่างตัวแปร ส่วน กราฟแท่งแนวนอน เหมาะสำหรับข้อมูลที่หมวดหมู่มีข้อความยาวและต้องการความชัดเจน

ไลบรารีอย่าง NumPy, Pandas และ Matplotlib เป็น "ชุมพลัง" ของนักวิเคราะห์ข้อมูลและนักวิทยาศาสตร์ข้อมูลทุกคน ผู้แต่งเชื่อว่าไลบรารีเหล่านี้ช่วยให้สามารถจัดการข้อมูลและนำเสนอผลลัพธ์ได้อย่างมืออาชีพ ไม่ว่าจะเป็นการคำนวนเมทริกซ์ การวิเคราะห์ข้อมูลใน DataFrame หรือการสร้างกราฟแสดงผลที่ดูเข้าใจง่าย ยิ่งผู้เรียนเชี่ยวชาญการใช้ไลบรารีเหล่านี้มากเท่าไหร่ โอกาสต่อยอดไปสู่งานด้านวิทยาศาสตร์ข้อมูลหรือปัญญาประดิษฐ์ยิ่งมากขึ้น

บทสรุป

ในยุคที่ข้อมูลมีบทบาทสำคัญในการตัดสินใจและการวิเคราะห์ การจัดการข้อมูลอย่างเป็นระบบจึงกลายเป็นทักษะพื้นฐานที่จำเป็นในทุกสาขา บทนี้จะพาคุณไปทำความเข้าใจเกี่ยวกับโครงสร้างข้อมูลพื้นฐานใน Python เช่น ลิสต์ (List) ดิกชันนารี (Dictionary) และการจัดการข้อมูลในรูปแบบตาราง เพื่อรองรับงานวิเคราะห์ข้อมูลในสถานการณ์จริง

นอกจากนี้ ยังจะได้เรียนรู้วิธีการอ่านและเขียนไฟล์ในรูปแบบต่างๆ เช่น ไฟล์ข้อความ (.txt), ไฟล์ CSV และไฟล์ JSON พร้อมตัวอย่างการใช้งานที่สามารถนำไปใช้ได้จริงในโปรเจกต์ การเรียนรู้การใช้ไลบรารีสำคัญของ Python เช่น NumPy, Pandas และ Matplotlib ยังเป็นหัวใจสำคัญของบทนี้ เพื่อช่วยในการจัดการข้อมูลเชิงตัวเลข การประมวลผลข้อมูลขนาดใหญ่ และการสร้างกราฟนำเสนอบล็อก

บทนี้จึงเป็นการเตรียมความพร้อมที่ดีสำหรับการทำางกับข้อมูลในโครงการจริง ช่วยให้สามารถนำความรู้ไปประยุกต์ใช้กับงานด้านวิทยาศาสตร์ข้อมูล (Data Science), การเรียนรู้ของเครื่อง (Machine Learning) หรือระบบสารสนเทศได้อย่างมีประสิทธิภาพ

- โครงสร้างข้อมูลพื้นฐาน Python มีโครงสร้างข้อมูลที่หลากหลาย เช่น List Dictionary และ Table ซึ่งเหมาะสมสำหรับการจัดเก็บและประมวลผลข้อมูลในรูปแบบต่าง ๆ
- การจัดการลิสต์ ใช้สำหรับจัดเก็บข้อมูลแบบลำดับที่เข้าถึงด้วยตัวชี้ (Index) มีฟังก์ชันเช่น append() remove() และ sort() ที่ช่วยจัดการข้อมูลในลิสต์
- การจัดการดิกชันนารี เก็บข้อมูลแบบคู่คี่-ค่า (Key-Value) เพื่อการเข้าถึงข้อมูลที่รวดเร็วและยืดหยุ่น
- การประมวลผลข้อมูลในรูปแบบตาราง ใช้โครงสร้าง List และ Dictionary เพื่อจัดการข้อมูลเชิงโครงสร้างสำหรับการวิเคราะห์ เช่น คะแนนนักเรียนในแต่ละวิชา
- การอ่านไฟล์ข้อความ ใช้ฟังก์ชัน open() และ read() หรือ readlines() เพื่ออ่านข้อมูลจากไฟล์ .txt
- การเขียนไฟล์ข้อความ ใช้ฟังก์ชัน write() สำหรับเขียนข้อมูลใหม่ลงไฟล์ และ append() สำหรับเพิ่มข้อมูลต่อท้ายไฟล์เดิม
- การจัดการไฟล์ CSV ใช้โมดูล csv เพื่ออ่านและเขียนข้อมูลในรูปแบบ CSV ที่สะดวกต่อการประมวลผลในรูปแบบตาราง
- การจัดการไฟล์ JSON ใช้โมดูล json เพื่อจัดเก็บข้อมูลในโครงสร้าง Key-Value และ Array ที่สามารถอ่านและแก้ไขได้ง่าย
- การใช้ไลบรารี Pandas สำหรับจัดการข้อมูลในรูปแบบตาราง DataFrame เช่น การอ่านข้อมูลจากไฟล์ CS
- การประมวลผลข้อมูลด้วย Pandas กรองข้อมูลด้วยเงื่อนไขและการคำนวนค่าเฉลี่ยหรือสถิติเชิงปริมาณ

- การใช้ไลบรารี NumPy สำหรับการคำนวณเชิงตัวเลข เช่น การจัดการอาเรย์และเมทริกซ์ พร้อมทั้ง การคำนวณค่าทางคณิตศาสตร์
- การเปรียบเทียบ NumPy กับ List NumPy เมมเบอร์กับงานที่ต้องการประสิทธิภาพสูงในขณะที่ List เหมาะสมสำหรับการใช้งานทั่วไป
- การติดตั้งไลบรารีด้วย pip สามารถติดตั้งไลบรารี เช่น NumPy Pandas และ Matplotlib ผ่านคำสั่ง pip install
- การอัปเดตและจัดการเวอร์ชันไลบรารี ใช้คำสั่ง pip install --upgrade และ pip list เพื่อตรวจสอบ และอัปเดตเวอร์ชันไลบรารี
- การวิเคราะห์ข้อมูลด้วย Matplotlib ใช้สำหรับสร้างกราฟและการแสดงผลข้อมูล เช่น การสร้าง แผนภูมิแท่งหรือแผนภูมิวงกลม
- การจัดการข้อมูลในรูปแบบตารางด้วย Pandas ใช้ DataFrame ในการกรองและจัดการข้อมูลจำนวนมาก มาก
- การอ่านและเขียนข้อมูลขนาดใหญ่ Pandas ช่วยให้ง่ายต่อการจัดการข้อมูลในไฟล์ขนาดใหญ่ เช่น CSV และ Excel
- การประมวลผลเมทริกซ์ NumPy ช่วยให้การคำนวณในเมทริกซ์มีประสิทธิภาพ เช่น การบวกและการ หาผลรวมในแต่ละแถวหรือคอลัมน์
- การสร้างข้อมูลสุ่มด้วย NumPy ใช้ฟังก์ชัน np.random สำหรับการสร้างข้อมูลตัวเลขสุ่มในงานวิจัย หรือการจำลอง
- การใช้งาน Pandas กับข้อมูลในชีวิตจริง Pandas มีความสามารถในการจัดการข้อมูลoba เก่าใน จังหวัดกาฬสินธุ์ในรูปแบบ DataFrame
- การคำนวณสถิติ NumPy และ Pandas ช่วยให้สามารถคำนวณค่าเฉลี่ย ค่ามัธยฐาน และค่าความ เปี่ยงเบนมาตรฐานได้อย่างรวดเร็ว
- การจัดการหน่วยความจำ NumPy ใช้หน่วยความจำน้อยกว่า List ทำให้เหมาะสมกับการจัดการข้อมูล จำนวนมาก
- ความสามารถของโครงสร้างข้อมูล การเลือกใช้โครงสร้างข้อมูลและไลบรารีที่เหมาะสมช่วยเพิ่ม ประสิทธิภาพและลดความซับซ้อนของกระบวนการทำงาน

คำถามท้ายบท

1. อธิบายความแตกต่างระหว่างโครงสร้างข้อมูล List และ Dictionary พร้อมยกตัวอย่างการใช้งาน
2. ในกรณีใดที่ควรเลือกใช้ NumPy Array แทน List ในการจัดเก็บข้อมูล และ เพราะเหตุใด
3. อธิบายวิธีการอ่านและเขียนไฟล์ .txt โดยใช้ฟังก์ชัน open() ใน Python พร้อมยกตัวอย่างโค้ด

4. การใช้ฟังก์ชัน sum() และ mean() ใน NumPy ต่างจากการใช้งานใน Python แบบปกติอย่างไร
5. ระบุขั้นตอนในการสร้าง DataFrame จากข้อมูลในรูปแบบ Dictionary โดยใช้ Pandas พร้อมยกตัวอย่าง
6. อธิบายวิธีการกรองข้อมูลใน DataFrame ของ Pandas โดยใช้เงื่อนไข เช่น ค่าที่มากกว่าหรือเท่ากับ
7. แสดงตัวอย่างการสร้างกราฟแท่ง (Bar Chart) โดยใช้ Matplotlib เพื่อแสดงข้อมูลจำนวนประชากรในแต่ละอำเภอ
8. ความแตกต่างระหว่างการจัดการข้อมูลในไฟล์ CSV และ JSON ใน Python คืออะไร
9. อธิบายการใช้งานฟังก์ชัน groupby() ใน Pandas และระบุสถานการณ์ที่เหมาะสมสำหรับการใช้ฟังก์ชันนี้
10. ใน NumPy ฟังก์ชัน reshape() ใช้สำหรับอะไร และสามารถใช้งานอย่างไรในงานวิเคราะห์ข้อมูล
11. อธิบายความสำคัญของฟังก์ชัน plt.tight_layout() ใน Matplotlib
12. แสดงตัวอย่างการใช้ฟังก์ชัน json.dump() และ json.load() ในการเขียนและอ่านข้อมูล JSON
13. การใช้ np.random.rand() และ np.random.randint() แตกต่างกันอย่างไร ยกตัวอย่างการใช้งาน
14. อธิบายวิธีการเพิ่มคอลัมน์ใหม่ใน DataFrame ของ Pandas และยกตัวอย่างการสร้างคอลัมน์ที่มีค่าคำนวนจากคอลัมน์อื่น
15. เปรียบเทียบ两种การเปิดไฟล์ใน Python ระหว่าง "r" (อ่าน) และ "w" (เขียนใหม่) พร้อมตัวอย่าง
16. อธิบายการคำนวณค่าความเบี่ยงเบนมาตรฐาน (Standard Deviation) ของข้อมูลโดยใช้ NumPy
17. แสดงตัวอย่างการเขียนไฟล์ CSV จาก DataFrame โดยใช้ Pandas พร้อมคำอธิบายโค้ด
18. ใน Matplotlib การสร้างกราฟวงกลม (Pie Chart) ใช้พารามิเตอร์ autopct และ startangle เพื่อปรับอะไรในกราฟ
19. อธิบายวิธีการอัปเดตเวอร์ชันของไลบรารี Python โดยใช้ pip พร้อมยกตัวอย่างคำสั่ง
20. อธิบายการใช้งานฟังก์ชัน filter() ใน Pandas พร้อมแสดงตัวอย่างการกรองข้อมูลที่ซับซ้อน

โจทย์การเขียนโปรแกรม

1. เขียนโปรแกรมที่รับรายชื่อนักเรียนและคะแนน 3 วิชา (คณิตศาสตร์, วิทยาศาสตร์, ภาษาอังกฤษ) จากผู้ใช้ แล้วคำนวณคะแนนเฉลี่ยของแต่ละนักเรียนและแสดงผลลัพธ์

ชื่อนักเรียน: Alice

คะแนนวิชาคณิตศาสตร์: 85

คะแนนวิชาวิทยาศาสตร์: 90

คะแนนวิชาภาษาอังกฤษ: 88

ชื่อนักเรียน: Bob

คะแนนวิชาคณิตศาสตร์: 78

คะแนนวิชาวิทยาศาสตร์: 80

คะแนนวิชาภาษาอังกฤษ: 82

ชื่อนักเรียน: exit

ผลลัพธ์:

Alice: คะแนนเฉลี่ย 87.67

Bob: คะแนนเฉลี่ย 80.00

2. เขียนโปรแกรมอ่านข้อมูลจากไฟล์ kalasin.csv และสร้างกราฟแท่ง (Bar Chart) เพื่อแสดงจำนวนประชากรในแต่ละอำเภอ แกน X: ชื่ออำเภอ แกน Y: จำนวนประชากร

3. เขียนโปรแกรมที่อ่านข้อมูล kalasin.csv และแสดงชื่ออำเภอที่มีประชากรมากที่สุดและน้อยที่สุด พร้อมจำนวนประชากร

อำเภอที่มีประชากรมากที่สุด: เมืองกาฬสินธุ์ (150,000 คน)

อำเภอที่มีประชากรน้อยที่สุด: ดอนจาน (15,000 คน)

4. เขียนโปรแกรมอ่านข้อมูล kalasin.csv และกรองข้อมูลเฉพาะอำเภอที่มีประชากรมากกว่า 50,000 คน

อำเภอที่มีประชากรมากกว่า 50,000 คน:

- เมืองกาฬสินธุ์ (150,000 คน)
- กุฉินารายณ์ (120,000 คน)
- ยางตลาด (110,000 คน)
- กลมลาไ洒 (70,000 คน)
- สมเด็จ (60,000 คน)

5. เขียนโปรแกรมที่อ่านข้อมูลจาก kalasin.csv และสร้างไฟล์ใหม่ filtered_population.csv ซึ่งเก็บข้อมูลเฉพาะอำเภอที่มีประชากรมากกว่า 50,000 คน

ตัวอย่างไฟล์ filtered_population.csv

รหัสอำเภอ,อำเภอ,รหัสไปรษณีย์,จำนวนประชากร (ประมาณ),LAT,LNG

4601,เมืองกาฬสินธุ์,46000,150000,16.4336,103.5078

4603,กุฉินารายณ์,46110,120000,16.5127,104.0536

...

6. เขียนโปรแกรมที่อ่านข้อมูล kalasin.csv และคำนวณข้อมูลสถิติ เช่น ค่าเฉลี่ย, ค่าสูงสุด, ค่าต่ำสุด, และส่วนเบี่ยงเบนมาตรฐานของจำนวนประชากร

ค่าเฉลี่ยประชากร: 54,333 คน

จำนวนประชากรมากที่สุด: 150,000 คน

จำนวนประชากรน้อยที่สุด: 15,000 คน

ส่วนเบี่ยงเบนมาตรฐาน: 44,258.21

7. เขียนโปรแกรมที่รับข้อมูลรายชื่อนักเรียนและคะแนน จากนั้นแสดงชื่อนักเรียนที่ได้คะแนนสูงสุด

ชื่อนักเรียน: Alice

คะแนน: 85

ชื่อนักเรียน: Bob

คะแนน: 90

ชื่อนักเรียน: exit

ผลลัพธ์:

นักเรียนที่ได้คะแนนสูงสุดคือ Bob (90 คะแนน)

8. เขียนโปรแกรมอ่านข้อมูล kalasin.csv และสร้างกราฟวงกลม (Pie Chart) เพื่อแสดงสัดส่วนประชากรในแต่ละอำเภอ

9. เขียนโปรแกรมที่อ่านข้อมูล kalasin.csv และเพิ่มคอลัมน์ใหม่ชื่อ "ประชากร (หน่วยพัน)" โดยแสดงจำนวนประชากรในหน่วยพัน จากนั้นแสดงผลลัพธ์

รหัสอำเภอ, อำเภอ, รหัสไปรษณีย์, จำนวนประชากร (ประมาณ), ประชากร (หน่วยพัน)

4601, เมืองกาฬสินธุ์, 46000, 150000, 150.0

4602, นาหม่วง, 46230, 40000, 40.0

...

10. เขียนโปรแกรมที่อ่านข้อมูล kalasin.csv และนับจำนวนอำเภอที่มีรหัสไปรษณีย์ซ้ำกัน จากนั้นแสดงผลลัพธ์

จำนวนอำเภอที่มีรหัสไปรษณีย์ซ้ำกัน:

46000: 2 อำเภอ

46130: 2 อำเภอ

46180: 2 อำเภอ

...

11. เขียนโปรแกรมอ่านข้อมูลจากไฟล์ JSON (students.json) และแปลงข้อมูลเป็น DataFrame โดยใช้ Pandas จากนั้นแสดงผลข้อมูลทั้งหมด

	name	math	science	english
0	Alice	90	85	88

1	Bob	78	80	82
2	Charlie	92	88	85
3	David	85	91	89
4	Eve	88	84	90

12. เขียนโปรแกรมที่อ่านข้อมูล students.json และให้ผู้ใช้ป้อนชื่อวิชา จากนั้นแสดงชื่อและคะแนนของนักเรียนในวิชานั้นเรียงจากมากไปน้อย

วิชาที่ต้องการค้นหา: math

ผลลัพธ์:

Charlie: 92

Alice: 90

Eve: 88

David: 85

Bob: 78

13. สร้างโปรแกรมที่สร้างเมทริกซ์ 3×3 โดยให้ผู้ใช้ป้อนตัวเลขแต่ละค่า จากนั้นคำนวณผลรวมของแต่ละแถวและแต่ละคอลัมน์

ป้อนค่าในเมทริกซ์ 3×3 :

1 2 3

4 5 6

7 8 9

ผลรวมของแต่ละแถว: [6, 15, 24]

ผลรวมของแต่ละคอลัมน์: [12, 15, 18]

14. เขียนโปรแกรมอ่านข้อมูล students.json และแสดงชื่อนักเรียนที่ได้คะแนนสูงสุดในแต่ละวิชา

ผลลัพธ์:

คณิตศาสตร์: Charlie (92 คะแนน)

วิทยาศาสตร์: David (91 คะแนน)

ภาษาอังกฤษ: David (89 คะแนน)

15. เขียนโปรแกรมสร้างอาร์เรย์ตัวเลขสุ่มจำนวน 10 ค่าในช่วง [1, 100] จากนั้นคำนวณค่าเฉลี่ย ค่าสูงสุด และค่าต่ำสุด

อาร์เรย์สุ่ม: [15, 87, 23, 45, 66, 78, 12, 90, 34, 56]

ค่าเฉลี่ย: 50.60

ค่าสูงสุด: 90

ค่าต่ำสุด: 12

16. เขียนโปรแกรมอ่านข้อมูล kalasin.csv และสร้างกราฟเส้น (Line Chart) เพื่อแสดงแนวโน้มจำนวนประชากรเรียงตามรหัสอำเภอ (กราฟเส้นแสดงจำนวนประชากรตามรหัสอำเภอ)

17. เขียนโปรแกรมที่รับข้อมูลนักเรียน (ชื่อ, คะแนนแต่ละวิชา) จากผู้ใช้ จากนั้นแสดงผลในรูปแบบตารางโดยใช้ Pandas DataFrame

	Name	Math	Science	English
0	Alice	85	88	90
1	Bob	80	85	83
2	Charlie	92	89	88

18. เขียนโปรแกรมอ่านข้อมูล kalasin.csv และคำนวณเปอร์เซ็นต์ของประชากรในแต่ละอำเภอเทียบกับประชากรทั้งหมด

เมืองกาฬสินธุ์: 27.6%

กุจินารายณ์: 22.1%

ยางตลาด: 20.3%

...

19. เขียนโปรแกรมที่รับเกณฑ์คะแนนขั้นต่ำ จากนั้นกรองข้อมูลนักเรียนใน students.json ที่ได้คะแนนเฉลี่ยต่ำกว่าเกณฑ์

เกณฑ์คะแนนขั้นต่ำ: 85

นักเรียนที่ได้คะแนนเฉลี่ยต่ำกว่า 85:

Bob: 80.0

Charlie: 83.3

20. เขียนโปรแกรมอ่านข้อมูล kalasin.csv และสร้างกราฟการกระจาย (Scatter Plot) เพื่อแสดงพิกัดต่ำเหน่งของแต่ละอำเภอ (LAT และ LNG) (กราฟ Scatter Plot แสดงพิกัดต่ำเหน่งของแต่ละอำเภอ)

บทที่ 7

การประยุกต์การเขียนโปรแกรมเพื่อแก้ปัญหา งานด้านวิศวกรรม

เนื้อหาบทเรียน

- 7.1 การเขียนโปรแกรมเพื่อคำนวณด้านวิศวกรรมและแสดงผลกราฟ
- 7.2 การเขียนโปรแกรมเพื่อคำนวณหาเส้นทางที่สั้นที่สุด (TSP)
- 7.3 การเขียนโปรแกรมระบบอินเตอร์เน็ตของสรรพสิ่ง (IoT)
- 7.4 การเขียนโปรแกรมระบบการเรียนรู้ของเครื่อง (Machine Learning)
- 7.5 งานวิจัยเพิ่มเติม

วัตถุประสงค์การเรียนรู้

- เขียนโปรแกรมเพื่อคำนวณทางด้านวิศวกรรมและแสดงผลกราฟได้
- อธิบายหลักการของการหาเส้นทางที่สั้นที่สุด (TSP) ได้
- เขียนโปรแกรมเพื่อคำนวณหาเส้นทางที่สั้นที่สุดได้
- บอกองค์ประกอบของวงจรอิเล็กทรอนิกส์สำหรับงาน ระบบอินเตอร์เน็ตของสรรพสิ่ง (IoT)
- เขียนโปรแกรมระบบอินเตอร์เน็ตของสรรพสิ่งได้
- บอกการประยุกต์ใช้การเรียนรู้ของเครื่องทางด้านการทำนายคาดถอยและการจำแนกภาพได้
- อธิบายหลักการงานวิจัยเพิ่มเติมได้

การเขียนโปรแกรมเพื่อแก้ปัญหาทางวิศวกรรมเป็นกระบวนการที่สำคัญในการพัฒนาระบบหรืออุปกรณ์ที่ตอบสนองต่อความต้องการเฉพาะด้านของงานวิศวกรรม กระบวนการนี้ประกอบด้วยขั้นตอนหลัก ๆ ดังนี้

การวิเคราะห์ปัญหา ทำความเข้าใจปัญหาที่ต้องการแก้ไข กำหนดวัตถุประสงค์ ขอบเขต และข้อกำหนดต่าง ๆ ของระบบหรืออุปกรณ์ที่ต้องการพัฒนา

การออกแบบขั้นตอนวิธี (Algorithm Design) วางแผนลำดับขั้นตอนการทำงานของโปรแกรมอย่างเป็นระบบ เพื่อให้สามารถแก้ปัญหาได้อย่างมีประสิทธิภาพ

การเขียนโปรแกรม (Coding) นำขั้นตอนวิธีที่ออกแบบไว้มาพัฒนาเป็นโปรแกรมด้วยภาษาคอมพิวเตอร์ที่เหมาะสม

การทดสอบและแก้ไขโปรแกรม (Testing and Debugging) ตรวจสอบการทำงานของโปรแกรมด้วยข้อมูลทดสอบ เพื่อหาข้อผิดพลาดและปรับปรุงให้โปรแกรมทำงานได้อย่างถูกต้อง

การจัดทำเอกสารประกอบ (Documentation) บันทึกรายละเอียดเกี่ยวกับโปรแกรม เช่น วัตถุประสงค์ วิธีการทำงาน และวิธีการใช้งาน เพื่อให้ผู้อื่นสามารถเข้าใจและนำไปใช้หรือพัฒนาต่อได้

การปฏิบัติตามขั้นตอนเหล่านี้อย่างเป็นระบบจะช่วยให้การพัฒนาโปรแกรมเพื่อแก้ปัญหาทางวิศวกรรมเป็นไปอย่างมีประสิทธิภาพและประสบความสำเร็จ

7.1 การเขียนโปรแกรมเพื่อคำนวณด้านวิศวกรรมและแสดงผลกราฟ

ในยุคปัจจุบัน การเขียนโปรแกรมไม่เพียงแค่กู้ใช้เพื่อพัฒนาซอฟต์แวร์ทั่วไป แต่ยังถูกนำมาใช้แก้ปัญหาในงานด้านวิศวกรรมเพื่อช่วยคำนวณ วิเคราะห์ และแสดงผลกราฟที่ช่วยสื่อสารข้อมูลเชิงลึกได้อย่างซัคเจน การใช้ Pygame และ Matplotlib ในบทนี้ถือเป็นเครื่องมือสำคัญที่ช่วยให้นักพัฒนาสามารถสร้างจำลองระบบ (Simulation) แบบเรียลไทม์ และแสดงผลข้อมูลที่ได้ในรูปแบบกราฟิก เช่น เส้นทางการเคลื่อนที่ของลูกบล็อก การชน การสูญเสียพลังงาน และพฤติกรรมทางฟิสิกส์อื่น ๆ

บทนี้เริ่มต้นด้วยการแนะนำ Pygame ซึ่งเป็นไลบรารีสำหรับสร้างภาพเคลื่อนไหว 2D ที่สามารถนำไปใช้ในการจำลองปรากฏการณ์ทางฟิสิกส์ เช่น แรงโน้มถ่วงและการชนของลูกบล็อก นอกจากนี้ยังมีการแสดงผลข้อมูลด้วย Matplotlib ซึ่งช่วยให้วิเคราะห์ข้อมูลผ่านกราฟง่ายขึ้น เช่น เส้นทางการเคลื่อนที่ของวัตถุ การเปลี่ยนแปลงพลังงาน และพฤติกรรมในเวลาต่าง ๆ

การทดสอบ Pygame และ Matplotlib ช่วยเพิ่มความเข้าใจต่อระบบที่จำลอง และยังช่วยให้การวิเคราะห์ผลลัพธ์เป็นไปอย่างมีประสิทธิภาพ เหมาะสำหรับการศึกษาและการพัฒนาโครงการในด้านวิศวกรรม หรือวิทยาศาสตร์ข้อมูลโดยการแก้ปัญหานี้ได้ใช้

Pygame ซึ่งเป็นไลบรารีใน Python ที่ช่วยจำลองการเคลื่อนที่ของลูกบล็อกแบบเรียลไทม์ โดยใช้ในการคำนวณตำแหน่ง ความเร็ว และแรงโน้มถ่วง รวมถึงการจัดการการชนกับขอบและลูกบล็อกอื่น ๆ พร้อมทั้งวัดภาพเคลื่อนไหวที่อัปเดตทุกเฟรมเพื่อแสดงผลสมจริง และ Matplotlib ใช้วิเคราะห์ผลการจำลองใน

รูปแบบกราฟ เช่น เส้นทางการเคลื่อนที่ของลูกบอล ความเร็ว ก่อนและหลังชน และการเปรียบเทียบระยะเวลาที่ผ่านไป ช่วยให้เข้าใจผลลัพธ์เชิงพิสิกส์ได้ดียิ่งขึ้น

การเริ่มต้น Pygame เป็นต้น

- ติดตั้งผ่าน pip คำสั่งสำหรับติดตั้ง Pygame

```
pip install pygame
```

- การสร้างหน้าจอแสดงผล Pygame ใช้คำสั่งต่อไปนี้เพื่อเริ่มต้นหน้าจอ

```
import pygame

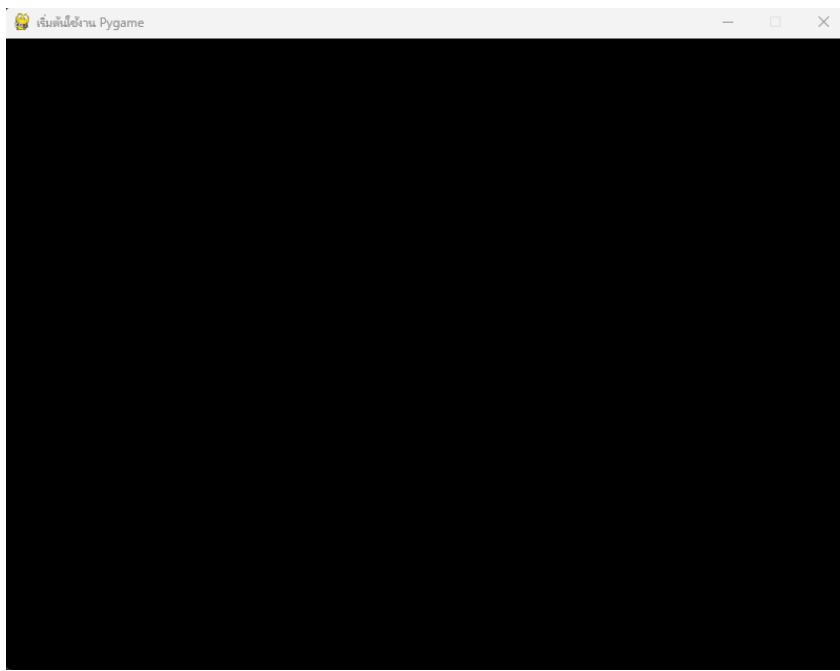
# เริ่มต้น Pygame
pygame.init()
# กำหนดขนาดหน้าจอ
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
# ตั้งชื่อหน้าต่าง
pygame.display.set_caption("เริ่มต้นใช้งาน Pygame")
# สีพื้นหลัง
BLACK = (0, 0, 0)
# วาดหน้าจอพื้นฐาน
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    # เติมสีพื้นหลัง
    screen.fill(BLACK)
    # อัปเดตหน้าจอ
    pygame.display.flip()

pygame.quit()
```

อธิบายโค้ดเบื้องต้น

- pygame.init() เริ่มต้นโมดูลต่าง ๆ ใน Pygame
- pygame.display.set_mode((WIDTH, HEIGHT)) สร้างหน้าต่างขนาด 800x600 พิกเซล
- pygame.display.set_caption() ตั้งชื่อหน้าต่าง
- screen.fill() เติมสีพื้นหลังให้หน้าจอ
- pygame.display.flip() อัปเดตการแสดงผล
- pygame.event.get() จัดการเหตุการณ์ เช่น การคลิกหรือปิดหน้าต่าง
- pygame.quit() ปิดการทำงานของ Pygame

ผลลัพธ์



หลังจากรันโค้ด จะเห็นหน้าต่างสีดำขนาด 800×600 พร้อมชื่อ "เริ่มต้นใช้งาน Pygame" ซึ่งเป็นหน้าจอพื้นฐานสำหรับการพัฒนาเกมหรือการจำลองภาพเคลื่อนไหว

3. การวาดรูปทรง Pygame มีฟังก์ชันสำหรับการวาดรูปทรงต่าง ๆ รวมถึงวงกลม สี่เหลี่ยม และเส้นตรง โดยสามารถปรับตำแหน่ง สี และขนาดได้ตามต้องการ ได้แก่

3.1) การวาดวงกลม ใช้ฟังก์ชัน `pygame.draw.circle()`

```
pygame.draw.circle(surface, color, center, radius)
```

`surface` คือ พื้นที่วิด (ในที่นี้คือ `screen`) `color` คือ สีของวงกลม (เช่น `RED`) `center` คือ ตำแหน่งศูนย์กลางของวงกลมในรูปแบบ (x, y) และ `radius` คือ รัศมีของวงกลม

3.2) การวาดสี่เหลี่ยม ใช้ฟังก์ชัน `pygame.draw.rect()`

```
pygame.draw.rect(surface, color, (x, y, width, height))
```

`x, y` คือ ตำแหน่งมุมบนซ้ายของสี่เหลี่ยม และ `width, height` คือ ความกว้างและความสูงของสี่เหลี่ยม

3.3) การวาดเส้นตรง ใช้ฟังก์ชัน `pygame.draw.line()`

```
pygame.draw.line(surface, color, start_pos, end_pos, width)
```

`start_pos` คือ จุดเริ่มต้นของเส้น `end_pos` คือ จุดสิ้นสุดของเส้น และ `width` ความหนาของเส้น

3.4) ตัวอย่างโค้ดการวาดวงกลม สี่เหลี่ยม และเส้นตรง โดยวัด

วงกลมสีแดง ตำแหน่งศูนย์กลาง $(200,300)$ และรัศมี 50

สี่เหลี่ยมสีฟ้า มุมบนซ้ายอยู่ที่ $(400,300)$ และมีขนาด 200×100

เส้นตรงสีเขียว เริ่มต้นที่ $(100,100)$ และสิ้นสุดที่ $(700,500)$ พร้อมความหนา 5

```
import pygame
```

```
# เริ่มต้น Pygame
```

```

pygame.init()

# กำหนดขนาดหน้าจอ
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("การวาดวงกลม สี่เหลี่ยม และเส้นตรงใน Pygame")

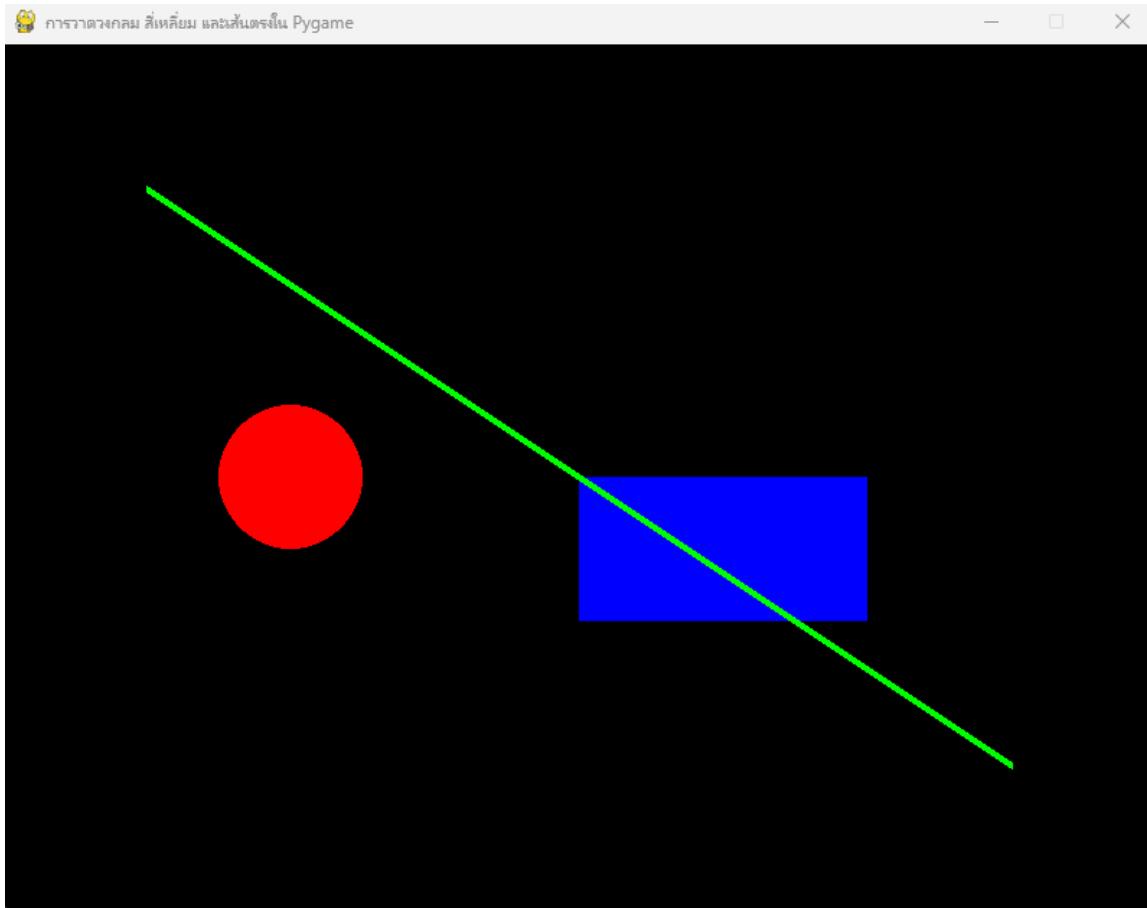
# กำหนดสี
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
BLUE = (0, 0, 255)
GREEN = (0, 255, 0)
# กำหนดตำแหน่งและขนาด
circle_x, circle_y = 200, 300 # ตำแหน่งของวงกลม
circle_radius = 50 # รัศมีของวงกลม
rect_x, rect_y = 400, 300 # ตำแหน่งของสี่เหลี่ยม
rect_width, rect_height = 200, 100 # ขนาดของสี่เหลี่ยม
line_start = (100, 100) # จุดเริ่มต้นของเส้น
line_end = (700, 500) # จุดสิ้นสุดของเส้น
line_width = 5 # ความหนาของเส้น

# ลูปหลัก
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # เดิมพื้นหลังสีดำ
    screen.fill(BLACK)
    # วาดวงกลม
    pygame.draw.circle(screen, RED, (circle_x, circle_y), circle_radius)
    # วาดสี่เหลี่ยม
    pygame.draw.rect(screen, BLUE, (rect_x, rect_y, rect_width, rect_height))
    # วาดเส้นตรง
    pygame.draw.line(screen, GREEN, line_start, line_end, line_width)
    # อัปเดตหน้าจอ
    pygame.display.flip()
# ปิด Pygame
pygame.quit()

```

ผลลัพธ์



7.1.1 การจำลองแรงโน้มถ่วงของลูกบอลและแสดงผล

ปัญหานี้เกี่ยวกับการจำลองการเคลื่อนที่ของลูกบอลในกรอบสี่เหลี่ยม โดยคำนึงถึงแรงโน้มถ่วงและการชนกับขอบของกรอบ พร้อมทั้งบันทึกตำแหน่งลูกบอลในแต่ละเฟรมเพื่อนำไปพล็อตกราฟเส้นทางการเคลื่อนที่ (Trajectory) ของลูกบอลด้วย Matplotlib

หลักการเคลื่อนที่ของลูกบอลตามหลักแรงโน้มถ่วง

- แรงโน้มถ่วง (Gravity)

$$v_y = v_y + g \cdot t$$

โดยที่ $g=0.1\text{pixels/frame}^2$ ค่าความเร่งจากแรงโน้มถ่วง และ $t = 1/\text{FPS}$ เวลาในแต่ละเฟรม
DAMPING_COEFFICIENT (ค่าสมมประสิทธิ์ลดแรง R)

$$v_x = Rv_x, v_y = Rv_y$$

2. การชนกับขอบจอ เมื่อเกิดการชนกับขอบจอ ความเร็วในแกนนั้นจะเปลี่ยนทิศทาง (สะท้อนกลับ)
E คือค่าสมมประสิทธิ์พลังงานที่สูญเสีย (energy loss factor)

$$v_x = -Ev_x$$

และ การชนกับขอบแนวตั้ง y

$$v_y = -Ev_y$$

ขั้นตอนการแก้ปัญหา

1. การตั้งค่าหน้าจอและตัวแปรเริ่มต้น ใช้ Pygame เพื่อสร้างหน้าจอขนาด 800×600 พิกเซล และกำหนดค่าคงที่ เช่น รัศมีลูกบอล (15) จำนวนลูกบอล (20) อัตราเฟรม (30FPS) พลังงานที่ลดลงหลังจากการชน(0.5) อัตราลดลงของความเร็ว (0.99) แรงโน้มถ่วง (0.1) และจำนวนเฟรมที่จำกัด (600)

```
import pygame
import matplotlib.pyplot as plt

# กำหนดค่าต่างๆ
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
BALL_RADIUS = 15
GRAVITY = 0.1
BOUNCE_DAMPING = 0.9
DAMPING_COEFFICIENT = 0.999
FPS = 60
FRAME_LIMIT = 600

# สี
BLACK = (0, 0, 0)
RED = (255, 0, 0)
```

2. การสร้างลูกบอล ใช้ลูปสร้างลูกบอลในรูปแบบของ dictionary โดยกำหนด ตำแหน่งเริ่มต้น (x,y) ความเร็ว (v_x,v_y) และสี

```
# ลูกบอลและตำแหน่ง
ball = {"x": 400, "y": 100, "vx": 3, "vy": 0, "color": RED}
positions = [] # เก็บตำแหน่งเพื่อผลลัพธ์
```

3. การเคลื่อนที่ของลูกบอล คำนวณตำแหน่งใหม่ของลูกบอลในแต่ละเฟรม การอัปเดตความเร็วแกนตั้ง

$$\begin{aligned} v_y &= v_y + g \\ v_x &= Rv_x, v_y = Rv_y \end{aligned}$$

และ การย้ายตำแหน่งแนวนอนและแนวตั้ง

$$x = x + v_x \text{ และ } y = y + v_y$$

เพิ่มแรงโน้มถ่วงในแกนตั้ง

ตรวจสอบการชนกับขอบจอ หากชนขอบแนวนอน (x) สะท้อนกลับด้วย

$$v_x = -Ev_x, v_y = -Ev_y$$

หากชนขอบแนวตั้ง

3.1) อัปเดตตำแหน่งของลูกบอล

```
ball["x"] += ball["vx"]
ball["y"] += ball["vy"]
```

3.2) ลดความเร็วในทุกเฟรม

```
ball["vx"] *= DAMPING_COEFFICIENT
ball["vy"] *= DAMPING_COEFFICIENT
```

3.3) เพิ่มแรงโน้มถ่วง

```
ball["vy"] += GRAVITY # Apply gravity
```

3.4) การชนกับขอบจอแนวอน (x)

```
# การชนกับขอบจอ
if ball["x"] - BALL_RADIUS <= 0 or ball["x"] + BALL_RADIUS >= SCREEN_WIDTH:
    ball["vx"] = -ball["vx"] * BOUNCE_DAMPING
```

3.5) การชนกับขอบจอแนวตั้ง (y)

```
if ball["y"] + BALL_RADIUS >= SCREEN_HEIGHT:
    ball["vy"] = -ball["vy"] * BOUNCE_DAMPING
    ball["y"] = SCREEN_HEIGHT - BALL_RADIUS
```

4. การอัปเดตตำแหน่งลูกบอล ใช้ฟังก์ชัน move_ball และบันทึกตำแหน่งลูกบอลใน positions

```
move_ball(ball)
positions.append((ball["x"], ball["y"])) # บันทึกตำแหน่ง
```

5. การแสดงผลลูกบอล ใช้ pygame.draw.circle() เพื่อวาดลูกบอลตามตำแหน่งที่คำนวณได้ในแต่ละเฟรม ในฟังก์ชัน draw_ball(ball)

```
def draw_ball(ball):
    pygame.draw.circle(screen, ball["color"], (int(ball["x"]),
int(ball["y"])),
BALL_RADIUS)
```

6. การอัปเดตหน้าจอ ใช้ pygame.display.flip() เพื่ออัปเดตหน้าจอหลังจากวาดลูกบอล

7. การพล็อตกราฟเส้นทางการเคลื่อนที่ หลังจากสิ้นสุด Simulation นำตำแหน่งลูกบอลใน positions มาพล็อตกราฟโดยใช้ Matplotlib โดยแกน x ตำแหน่งแนวอน และแกน y ตำแหน่งแนวตั้ง (กลับด้านให้เหมือนในโลกจริง)

```
# พล็อตกราฟ
positions_x = [p[0] for p in positions]
positions_y = [p[1] for p in positions]
plt.plot(positions_x, positions_y, label="Ball Path")
plt.xlabel("X Position")
plt.ylabel("Y Position")
plt.title("Ball Trajectory")
plt.legend()
plt.grid()
```

```

# Invert แก้ไข y
plt.gca().invert_yaxis()
plt.show()

โปรแกรมสมบูรณ์

import pygame
import matplotlib.pyplot as plt

# กำหนดค่าต่างๆ
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
BALL_RADIUS = 15
GRAVITY = 0.1
BOUNCE_DAMPING = 0.9
DAMPING_COEFFICIENT = 0.999
FPS = 60
FRAME_LIMIT = 600

# สี
BLACK = (0, 0, 0)
RED = (255, 0, 0)

# ลูกบอลและตำแหน่ง
ball = {"x": 400, "y": 100, "vx": 3, "vy": 0, "color": RED}
positions = [] # เก็บตำแหน่งเพื่อผลักดัน

ball["x"] += ball["vx"]
ball["y"] += ball["vy"]

ball["vx"] *= DAMPING_COEFFICIENT
ball["vy"] *= DAMPING_COEFFICIENT

ball["vy"] += GRAVITY # Apply gravity

# การชนกับขอบจอ
if ball["x"] - BALL_RADIUS <= 0 or ball["x"] + BALL_RADIUS >= SCREEN_WIDTH:
    ball["vx"] = -ball["vx"] * BOUNCE_DAMPING

if ball["y"] + BALL_RADIUS >= SCREEN_HEIGHT:
    ball["vy"] = -ball["vy"] * BOUNCE_DAMPING
    ball["y"] = SCREEN_HEIGHT - BALL_RADIUS

move_ball(ball)
positions.append((ball["x"], ball["y"])) # บันทึกตำแหน่ง

```

```

def draw_ball(ball):
    pygame.draw.circle(screen, ball["color"], (int(ball["x"]),
                                                int(ball["y"])),
                        BALL_RADIUS)

# พล็อตกราฟ
positions_x = [p[0] for p in positions]
positions_y = [p[1] for p in positions]
plt.plot(positions_x, positions_y, label="Ball Path")
plt.xlabel("X Position")
plt.ylabel("Y Position")
plt.title("Ball Trajectory")
plt.legend()
plt.grid()
# Invert แก้ y
plt.gca().invert_yaxis()
plt.show()

import pygame
import matplotlib.pyplot as plt

# กำหนดค่าต่างๆ
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
BALL_RADIUS = 15
GRAVITY = 0.1
BOUNCE_DAMPING = 0.9
DAMPING_COEFFICIENT = 0.999
FPS = 60
FRAME_LIMIT = 600

# สี
BLACK = (0, 0, 0)
RED = (255, 0, 0)

# เริ่มต้น Pygame
pygame.init()
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("ลูกบอลเด้ง")
clock = pygame.time.Clock()

# ลูกบอลและตำแหน่ง
ball = {"x": 400, "y": 100, "vx": 3, "vy": 0, "color": RED}

```

```

positions = [] # เก็บตำแหน่งเพื่อพิจารณา

def move_ball(ball):
    ball["x"] += ball["vx"]
    ball["y"] += ball["vy"]
    ball["vy"] += GRAVITY
    ball["vx"] *= DAMPING_COEFFICIENT
    ball["vy"] *= DAMPING_COEFFICIENT

    # การชนกับขอบจอ
    if ball["x"] - BALL_RADIUS <= 0 or ball["x"] + BALL_RADIUS >=
SCREEN_WIDTH:
        ball["vx"] = -ball["vx"] * BOUNCE_DAMPING
    if ball["y"] + BALL_RADIUS >= SCREEN_HEIGHT:
        ball["vy"] = -ball["vy"] * BOUNCE_DAMPING
        ball["y"] = SCREEN_HEIGHT - BALL_RADIUS

def draw_ball(ball):
    pygame.draw.circle(screen, ball["color"], (int(ball["x"]),
int(ball["y"])), BALL_RADIUS)

# Simulation loop
running = True
time_steps = 0
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    screen.fill(BLACK)
    move_ball(ball)
    draw_ball(ball)
    positions.append((ball["x"], ball["y"])) # บันทึกตำแหน่ง
    pygame.display.flip()
    clock.tick(FPS)
    time_steps += 1
    if time_steps > FRAME_LIMIT: # จำกัดจำนวนเฟรม
        running = False

pygame.quit()

# พิจารณา
positions_x = [p[0] for p in positions]
positions_y = [p[1] for p in positions]

plt.plot(positions_x, positions_y, label="Ball Path")
plt.xlabel("X Position")
plt.ylabel("Y Position")

```

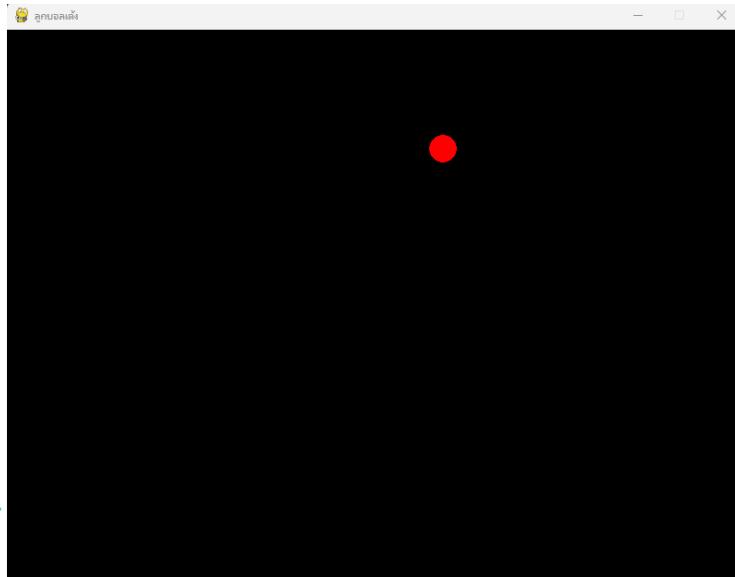
```

plt.title("Ball Trajectory")
plt.legend()
plt.grid()

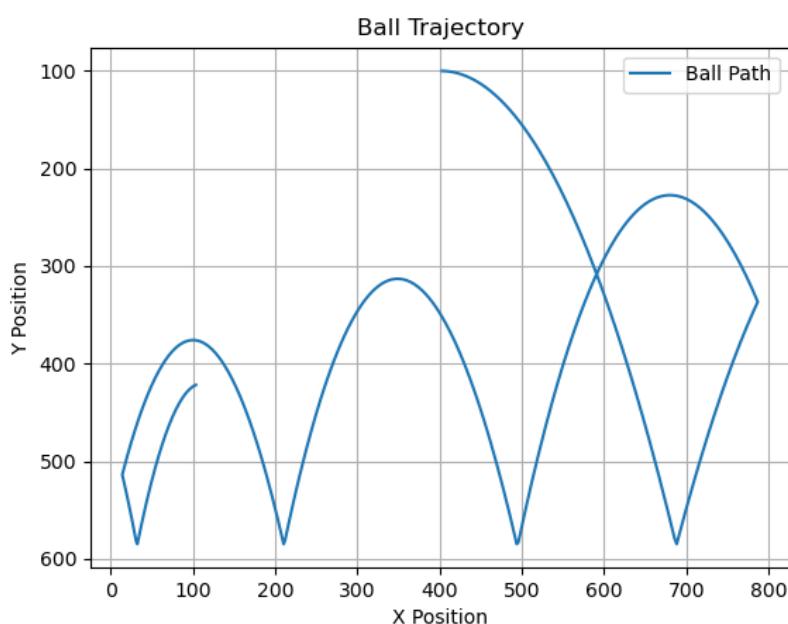
# Invert แก้ y
plt.gca().invert_yaxis()
plt.show()

```

ผลลัพธ์ การจำลองการเคลื่อนที่ ลูกบอลจะเคลื่อนที่ในกรอบหน้าจอ มีแรงโน้มถ่วงส่งผลให้ลูกบอลเคลื่อนที่ลง ด้านล่าง และลูกบอลชนขอบแล้วเด้งกลับ พร้อมสูญเสียพลังงานตามค่าสัมประสิทธิ์ BOUNCE_DAMPING



หลังจากการจำลอง ผลลัพธ์ กราฟเส้นทางการเคลื่อนที่ แสดงเส้นทางการวิ่งของลูกบอลในแต่ละ เพرم สามารถวิเคราะห์การสูญเสียพลังงานและเส้นทางการเคลื่อนที่ได้



7.1.2 การจำลองการชนและแรงโน้มถ่วงของลูกบอล

ปัญหานี้เกี่ยวกับการจำลองการเคลื่อนที่ของลูกบอลที่ได้รับผลกระทบจากแรงโน้มถ่วง และสามารถเกิดการชนกันทั้งกับขอบของกรอบสี่เหลี่ยมและระหว่างลูกบอลด้วยกันเอง โดยต้องคำนวณการเด้งกลับและปรับตำแหน่งของลูกบอลหลังชนเพื่อความสมจริง

หลักการเคลื่อนที่ของลูกบอลตามหลักแรงโน้มถ่วง

1. แรงโน้มถ่วง (Gravity)

$$v_y = v_y + g \cdot t$$

โดยที่ $g=0.1\text{pixels/frame}^2$ ค่าความเร่งจากแรงโน้มถ่วง และ $t = 1/\text{FPS}$ เวลาในแต่ละเฟรม DAMPING_COEFFICIENT (ค่าสัมประสิทธิ์ลดแรง R)

$$v_x = Rv_x, v_y = Rv_y$$

2. การชนกับขอบจอ เมื่อเกิดการชนกับขอบจอ ความเร็วในแกนนั้นจะเปลี่ยนทิศทาง (สะท้อนกลับ) E คือค่าสัมประสิทธิ์พลังงานที่สูญเสีย (energy loss factor)

$$v_x = -Ev_x$$

และ การชนกับขอบแนวตั้ง y

$$v_y = -Ev_y$$

ขั้นตอนการแก้ปัญหา

1. การตั้งค่าหน้าจอและตัวแปรเริ่มต้น ใช้ Pygame เพื่อสร้างหน้าจอขนาด 800×600 พิกเซล และกำหนดค่าคงที่ เช่น รัศมีลูกบอล (15) จำนวนลูกบอล (20) อัตราเฟรม (30FPS) พลังงานที่ลดลงหลังจากการชน(0.5) อัตราลดลงของความเร็ว (0.99) และแรงโน้มถ่วง (0.1)

```
import pygame
import random
import math

# กำหนดค่าคงที่
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
BALL_RADIUS = 15
NUM_BALLS = 20
FPS = 60
GRAVITY = 0.1
BOUNCE_DAMPING = 0.5
VELOCITY_REDUCTION = 0.999
# สี
BLACK = (0, 0, 0)
COLORS = [
    (255, 0, 0), (0, 255, 0), (0, 0, 255),
    (255, 255, 0), (0, 255, 255), (255, 0, 255)
]
```

2. การสร้างลูกбол ให้ลุปสร้างลูกบลในรูปแบบของ dictionary โดยกำหนด ตำแหน่งเริ่มต้น (x, y) ความเร็ว (v_x, v_y) สี และมวล

```
balls = []
for _ in range(NUM_BALLS):
    ball = {
        "x": random.randint(BALL_RADIUS, SCREEN_WIDTH - BALL_RADIUS),
        "y": random.randint(BALL_RADIUS, SCREEN_HEIGHT - BALL_RADIUS),
        "vx": random.uniform(-4, 4),
        "vy": random.uniform(-4, 4),
        "color": random.choice(COLORS),
        "mass": 1 # มวลของลูกบล
    }
    balls.append(ball)
```

3. การเคลื่อนที่ของลูกบล คำนวณตำแหน่งใหม่ของลูกบลในแต่ละเฟรม การอัพเดทความเร็วแกนตั้ง

$$\begin{aligned}v_y &= v_y + g \\v_x &= Rv_x, v_y = Rv_y\end{aligned}$$

และ การย้ายตำแหน่งแนวนอนและแนวตั้ง

$$x = x + v_x \text{ และ } y = y + v_y$$

เพิ่มแรงโน้มถ่วงในแกนตั้ง

ตรวจสอบการชนกับขอบจอ หากชนขอบแนวนอน (x) สะท้อนกลับด้วย

$$v_x = -Ev_x, v_y = -Ev_y$$

หากชนขอบแนวตั้ง

3.1) อัปเดตตำแหน่งของลูกบล

```
ball["x"] += ball["vx"]
ball["y"] += ball["vy"]
```

3.2) ลดความเร็วในทุกเฟรม

```
ball["vx"] *= DAMPING_COEFFICIENT
ball["vy"] *= DAMPING_COEFFICIENT
```

3.3) หยุดลูกบลเมื่อความเร็วต่ำมาก

```
if abs(ball["vx"]) <= 0.1:
    ball["vx"] = 0
if ball["y"] + BALL_RADIUS >= SCREEN_HEIGHT - 2 and abs(ball["vy"]) <= 0.01:
    ball["vy"] = 0
```

3.4) เพิ่มแรงโน้มถ่วง

```
ball["vy"] += GRAVITY # Apply gravity
```

3.5) การชนกับขอบจอแนวอน (x)

```
# ตรวจสอบการชนกับขอบจอ
if ball["x"] - BALL_RADIUS <= 0 or ball["x"] + BALL_RADIUS >= SCREEN_WIDTH:
    ball["vx"] = -ball["vx"]*BOUNCE_DAMPING # ลดทื่องความเร็วแกน X
    if ball["x"] - BALL_RADIUS <= 0:
        ball["x"] = BALL_RADIUS
    else:
        ball["x"] = SCREEN_WIDTH - BALL_RADIUS
```

3.6) การชนกับขอบจอแนวตั้ง (y)

```
if ball["y"] - BALL_RADIUS <= 0 or ball["y"] + BALL_RADIUS >= SCREEN_HEIGHT:
    ball["vy"] = -ball["vy"]*BOUNCE_DAMPING # ลดทื่องความเร็วแกน y
    if ball["y"] - BALL_RADIUS <= 0:
        ball["y"] = BALL_RADIUS
    else:
        ball["y"] = SCREEN_HEIGHT - BALL_RADIUS
```

4. การจัดการการชนระหว่างลูกบอล

4.1) คำนวณระยะห่างระหว่างลูกบอลสองลูก

$$\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```
dx = ball2["x"] - ball1["x"]
dy = ball2["y"] - ball1["y"]
distance = math.sqrt(dx**2 + dy**2)
```

4.2) เสื่อนไขการชน

$$\text{distance} < 2 \cdot \text{BALL_RADIUS}$$

```
# ตรวจสอบว่าลูกบอลชนกันหรือไม่
if distance < 2 * BALL_RADIUS:
```

4.3) การคำนวณทิศทางการชน (เวกเตอร์ปกติ)

$$n_x = \frac{x_2 - x_1}{\text{distance}}, \quad n_y = \frac{y_2 - y_1}{\text{distance}}$$

```
nx = dx / distance
ny = dy / distance
```

4.4) การคำนวณความเร็วใหม่ของลูกบอลหลังชน

```
v_x1 = -|v_1| \cdot n_x, \quad v_y1 = -|v_1| \cdot n_y
abs_velocity1 = math.sqrt(ball1["vx"]**2 + ball1["vy"]**2)
abs_velocity2 = math.sqrt(ball2["vx"]**2 + ball2["vy"]**2)
```

```
# อัปเดตความเร็ว
ball1["vx"] = - abs_velocity1 * nx * BOUNCE_DAMPING
ball1["vy"] = - abs_velocity1 * ny * BOUNCE_DAMPING
ball2["vx"] = abs_velocity2 * nx * BOUNCE_DAMPING
ball2["vy"] = abs_velocity2 * ny * BOUNCE_DAMPING
```

4.5) การปรับตำแหน่งลูกบอลเพื่อป้องกันการซ้อนทับ

$$x_1 = \frac{\text{overlap}}{2} \cdot n_x, \quad y_1 = \frac{\text{overlap}}{2} \cdot n_y$$

```
overlap = 2 * BALL_RADIUS - distance
separation = overlap / 2
ball1["x"] -= separation * nx
ball1["y"] -= separation * ny
ball2["x"] += separation * nx
ball2["y"] += separation * ny
```

5. การแสดงผลลูกบอล ใช้ pygame.draw.circle() เพื่อวาดลูกบอลตามตำแหน่งที่คำนวณได้ในแต่ละเฟรม

6. การอัปเดตหน้าจอ ใช้ pygame.display.flip() เพื่ออัปเดตหน้าจอหลังจากวาดลูกบอล

โปรแกรมสมบูรณ์

```
import pygame
import random
import math
# กำหนดค่าต่างๆ
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
BALL_RADIUS = 15
NUM_BALLS = 20
FPS = 60
GRAVITY = 0.1
BOUNCE_DAMPING = 0.75
DAMPING_COEFFICIENT = 0.99
# สี
BLACK = (0, 0, 0)
COLORS = [
    (255, 0, 0), (0, 255, 0), (0, 0, 255),
    (255, 255, 0), (0, 255, 255), (255, 0, 255)
]
# เริ่มต้น Pygame
pygame.init()
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("ลูกบอลชนกันในกล่อง")
clock = pygame.time.Clock()
# สร้างลูกบอลหลายลูก (ตีบเข้มสุกคุณลักษณะในรูปแบบ dictionary)
```

```

balls = []
for _ in range(NUM_BALLS):
    ball = {
        "x": random.randint(BALL_RADIUS, SCREEN_WIDTH - BALL_RADIUS),
        "y": random.randint(BALL_RADIUS, SCREEN_HEIGHT - BALL_RADIUS),
        "vx": random.uniform(-4, 4),
        "vy": random.uniform(-4, 4),
        "color": random.choice(COLORS),
        "mass": 1 # มวลของลูกบอล
    }
    balls.append(ball)
# พังก์ชั่นเคลื่อนที่ลูกบอล
def move_ball(ball):
    # อัปเดตตำแหน่ง
    ball["x"] += ball["vx"]
    ball["y"] += ball["vy"]
    ball["vx"] *= DAMPING_COEFFICIENT
    ball["vy"] *= DAMPING_COEFFICIENT

    if abs(ball["vx"]) <= 0.1:
        ball["vx"] = 0
    if ball["y"] + BALL_RADIUS >= SCREEN_HEIGHT-2 and abs(ball["vy"]) <=
0.01:
        ball["vy"] = 0
    else:
        ball["vy"] += GRAVITY # Apply gravity

    # ตรวจสอบการชนกับขอบจอ
    if ball["x"] - BALL_RADIUS <= 0 or ball["x"] + BALL_RADIUS >=
SCREEN_WIDTH:
        ball["vx"] = -ball["vx"]*BOUNCE_DAMPING # สะท้อนความเร็วแทน X
        if ball["x"] - BALL_RADIUS <= 0:
            ball["x"] = BALL_RADIUS
        else:
            ball["x"] = SCREEN_WIDTH - BALL_RADIUS
    if ball["y"] - BALL_RADIUS <= 0 or ball["y"] + BALL_RADIUS >=
SCREEN_HEIGHT:
        ball["vy"] = -ball["vy"]*BOUNCE_DAMPING # สะท้อนความเร็วแทน Y
        if ball["y"] - BALL_RADIUS <= 0:
            ball["y"] = BALL_RADIUS
        else:
            ball["y"] = SCREEN_HEIGHT - BALL_RADIUS

# พังก์ชั่นวาดลูกบอล
def draw_ball(ball):
    pygame.draw.circle(screen, ball["color"], (int(ball["x"]),
int(ball["y"])), BALL_RADIUS)

```

```

# ฟังก์ชันจัดการการชนระหว่างลูกนอล
def handle_collision(ball1, ball2):
    # คำนวณระยะห่างระหว่างลูกนอลสองลูก
    dx = ball2["x"] - ball1["x"]
    dy = ball2["y"] - ball1["y"]
    distance = math.sqrt(dx**2 + dy**2)

    # ตรวจสอบว่าลูกนอลชนกันหรือไม่
    if distance < 2 * BALL_RADIUS:
        # คำนวณเวกเตอร์ปั๊ดในแนว X และ Y
        nx = dx / distance
        ny = dy / distance

        abs_velocity1 = math.sqrt(ball1["vx"]**2 + ball1["vy"]**2)
        abs_velocity2 = math.sqrt(ball2["vx"]**2 + ball2["vy"]**2)

        # อัปเดตความเร็ว
        ball1["vx"] = -abs_velocity1 * nx * BOUNCE_DAMPING
        ball1["vy"] = -abs_velocity1 * ny * BOUNCE_DAMPING
        ball2["vx"] = abs_velocity2 * nx * BOUNCE_DAMPING
        ball2["vy"] = abs_velocity2 * ny * BOUNCE_DAMPING

        # แยกตำแหน่งลูกนอลออกจากกันเพื่อป้องกันการซ้อนทับ
        overlap = 2 * BALL_RADIUS - distance
        separation = overlap / 2
        ball1["x"] -= separation * nx
        ball1["y"] -= separation * ny
        ball2["x"] += separation * nx
        ball2["y"] += separation * ny

# เกมสูง
running = True
while running:
    # จัดการเหตุการณ์
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # ตรวจสอบการชนกันระหว่างลูกนอล
    for i in range(len(balls)):
        for j in range(i + 1, len(balls)):
            handle_collision(balls[i], balls[j])

    # อัปเดตตำแหน่งลูกนอล
    screen.fill(BLACK) # ล้างหน้าจอ
    for ball in balls:
        move_ball(ball)
        draw_ball(ball)

```

```
# แสดงผล
pygame.display.flip()
clock.tick(FPS)

# ออกจากรีบ
pygame.quit()
```

คำอธิบายฟังก์ชันในโค้ด

1. ฟังก์ชัน move_ball(ball) ฟังก์ชันนี้ใช้สำหรับอัปเดตตำแหน่งของลูกบอลในแต่ละเฟรม รวมถึงการจัดการแรงโน้มถ่วง การลดความเร็ว และการชนกับขอบจอ
2. ฟังก์ชัน draw_ball(ball) วาดลูกบอลบนหน้าจอที่ตำแหน่งปัจจุบันของลูกบอลโดยใช้ข้อมูลใน dictionary ball
3. ฟังก์ชัน handle_collision(ball1, ball2) จัดการการชนกันระหว่างลูกบอลสองลูก รวมถึงการคำนวณการสะท้อนกลับของความเร็วและการปรับตำแหน่งเพื่อป้องกันการซ้อนทับ
4. ควบคุมการทำงานของเกมโดยอัปเดตตำแหน่งลูกบอล จัดการการชนกัน และแสดงผลทุกเฟรม
 - 4.1) จัดการเหตุการณ์ ตรวจสอบว่าผู้ใช้ปิดหน้าต่างเกมหรือไม่

```
# จัดการเหตุการณ์
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        running = False
```

4.2) จัดการการชนระหว่างลูกบอล

```
# ตรวจสอบการชนกันระหว่างลูกบอล
for i in range(len(balls)):
    for j in range(i + 1, len(balls)):
        handle_collision(balls[i], balls[j])
```

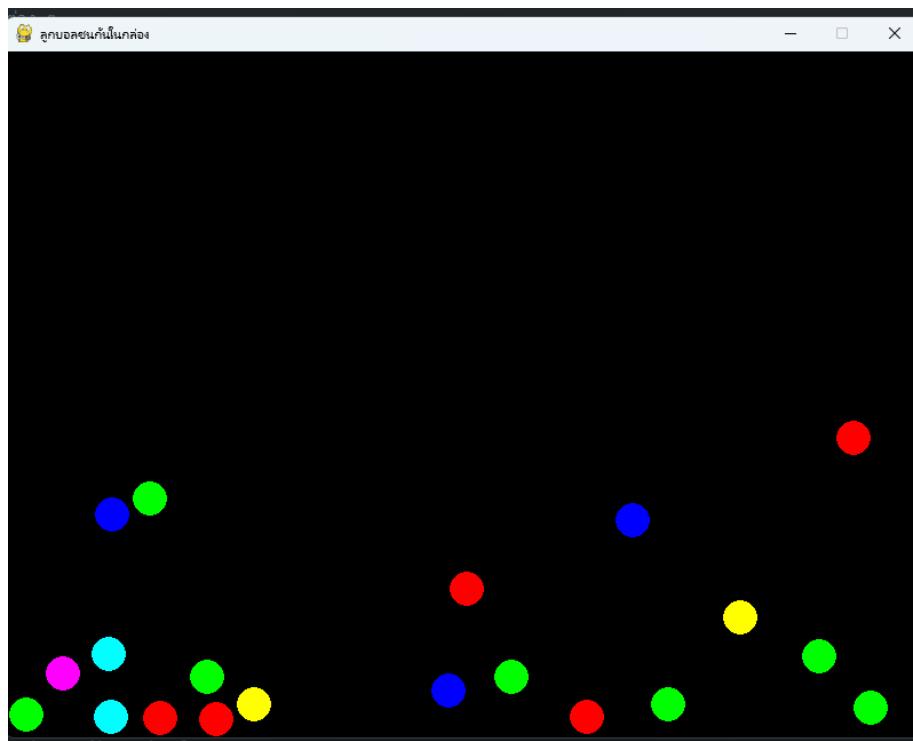
4.3) อัปเดตและวาดลูกบอล อัปเดตตำแหน่งลูกบอลแต่ละลูกด้วย move_ball และวาดลูกบอลใหม่ ด้วย draw_ball

```
# อัปเดตตำแหน่งลูกบอล
screen.fill(BLACK) # ล้างหน้าจอ
for ball in balls:
    move_ball(ball)
    draw_ball(ball)
```

4.4) ควบคุมอัตราเฟรม และแสดงผล

```
# แสดงผล
pygame.display.flip()
clock.tick(FPS)
```

ผลลัพธ์



ผู้แต่งมีความเห็นว่าการใช้ Pygame และ Matplotlib ร่วมกันในงานจำลองทางวิศวกรรมช่วยให้ "เห็นภาพ" ของข้อมูลและพฤติกรรมของวัตถุในโลกจริงได้ชัดเจนยิ่งขึ้น การสร้างการจำลอง เช่น การเคลื่อนที่ของลูกบอลที่ได้รับผลจากแรงโน้มถ่วงและแรงกระแทกของขอบจอ เป็นตัวอย่างที่ดีที่ทำให้เข้าใจหลักการทำงานพิสิกส์พร้อมกับฝึกฝนทักษะการเขียนโปรแกรมไปพร้อมกัน ถือเป็นจุดเริ่มต้นที่ดีสำหรับผู้ที่ต้องการต่อยอดไปทำงานด้านวิศวกรรมหรือการพัฒนาเกม

7.2 การเขียนโปรแกรมเพื่อคำนวนหาเส้นทางที่สั้นที่สุด (TSP)

การหาวิธีแก้ปัญหาเส้นทางที่สั้นที่สุด (Traveling Salesman Problem - TSP) เป็นปัญหาในงานคำนวนที่เกี่ยวข้องกับการหาลำดับของการเดินทางที่ครอบคลุมทุกจุดอย่างน้อยหนึ่งครั้งและมีระยะทางรวมสั้นที่สุด ปัญหานี้มีความสำคัญในด้านโลจิสติกส์ การขนส่ง และการจัดการเส้นทาง โดยในบทนี้จะใช้ไฟล์ข้อมูลที่เกี่ยวข้องกับระยะทางระหว่างจุดในจังหวัดกาฬสินธุ์เป็นตัวอย่างในการประยุกต์ใช้แนวคิด TSP

7.2.1 การเตรียมข้อมูล

ข้อมูลในไฟล์จะประกอบไปด้วยรายชื่ออำเภอและพิกัด (Latitude , Longitude) ของแต่ละอำเภอในจังหวัดกาฬสินธุ์ ซึ่งจะใช้เพื่อคำนวนระยะทางระหว่างจุดต่างๆ

ข้อมูลไฟล์ kalasin_districts_english.csv

```
District Code,District,Postal Code,Estimated
Population,Latitude,Longitude
4601,Mueang Kalasin,46000,150000,16.4336,103.5078
4602,Na Mon,46230,40000,16.2511,103.8193
4603,Kuchinarai,46110,120000,16.5127,104.0536
4604,Kamalasai,46130,70000,16.2966,103.5886
```

```

4605,Yang Talat,46120,110000,16.4019,103.2922
4606,Somdet,46150,60000,16.6195,103.3812
4607,Huai Phueng,46240,35000,16.4798,103.0447
4608,Sahatsakhan,46140,50000,16.7283,103.4639
4609,Kham Muang,46180,30000,16.6777,103.0621
4610,Tha Khantho,46190,40000,16.7928,103.1167
4611,Nong Kung Si,46220,45000,16.5435,103.0064
4612,Sam Chai,46180,25000,16.7011,103.2141
4613,Na Khu,46160,20000,16.4792,103.2613
4614,Don Chan,46000,15000,16.4289,103.3472
4615,Khong Chai,46130,20000,16.5952,103.1939

```

1. การอ่านไฟล์ CSV

```

import pandas as pd

# อ่านไฟล์ CSV
file_path = 'kalasin_districts_english.csv'
data = pd.read_csv(file_path)

# แสดงข้อมูลตัวอย่าง
print(data.head())

```

ผลลัพธ์

	District Code	District	Postal Code	Estimated Population	Latitude	Longitude
0	4601	Mueang Kalasin	46000	150000	16.4336	103.5078
1	4602	Na Mon	46230	40000	16.2511	103.8193
2	4603	Kuchinarai	46110	120000	16.5127	104.0536
3	4604	Kamalasai	46130	70000	16.2966	103.5886
4	4605	Yang Talat	46120	110000	16.4019	103.2922

2. การวัดพร้อมตำแหน่งโดยใช้ LAT LNG

```

import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
file_path = 'kalasin_districts_english.csv'
data = pd.read_csv(file_path)

# Plot the locations with names
plt.figure(figsize=(12, 8))
plt.scatter(data['Longitude'], data['Latitude'], c='blue', label='Districts',
           s=100)

# Annotate district names
for i, row in data.iterrows():
    plt.text(row['Longitude'], row['Latitude'], row['District'], fontsize=12,
             ha='right', va='bottom')

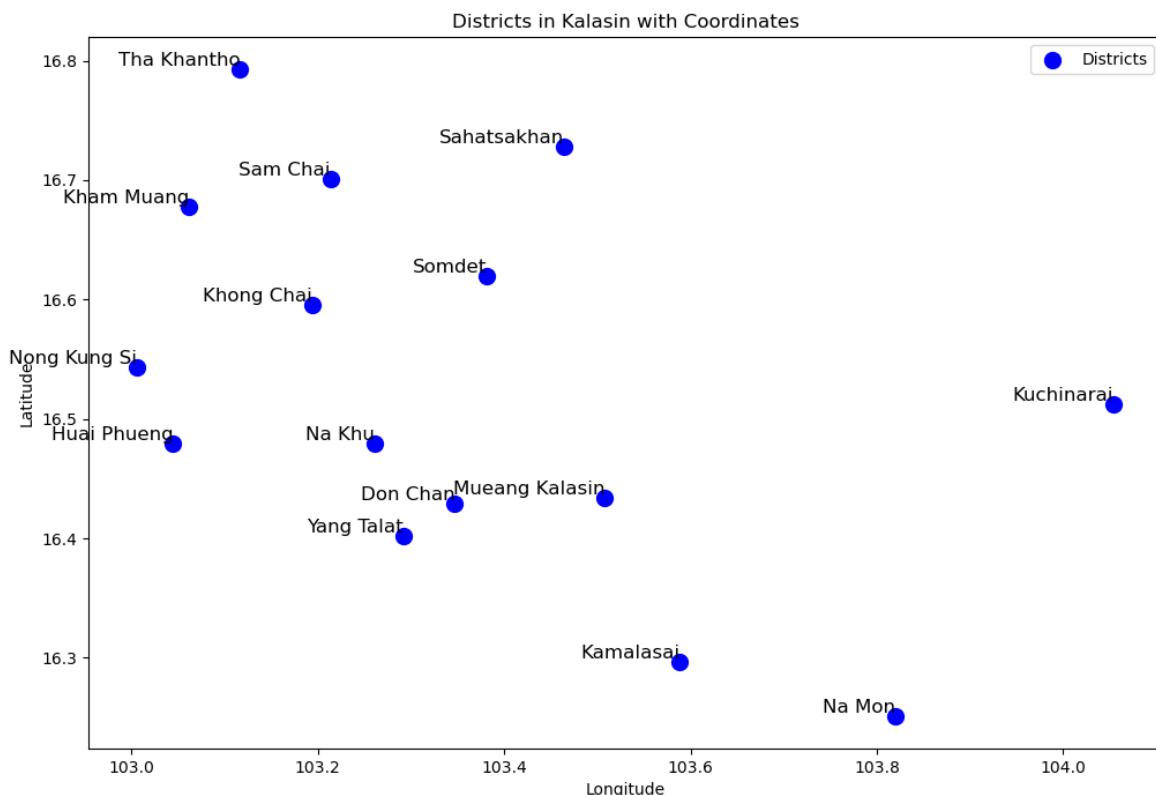
```

```

plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Districts in Kalasin with Coordinates')
# plt.grid()
plt.legend()
plt.show()

```

ผลลัพธ์



7.2.2 การคำนวณระยะทางระหว่างจุด

เมื่อได้ข้อมูลพิกัดของแต่ละจุดแล้ว สามารถคำนวณระยะทางระหว่างจุดสองจุดด้วยสูตร Haversine Formula ซึ่งใช้สำหรับคำนวณระยะทางบนพื้นผิวของทรงกลม (ในกรณีนี้คือโลก) โดยใช้ค่า latitude และ longitude จุด Haversine ดังนี้

$$d = 2r \cdot \arcsin \left(\sqrt{\sin^2 \left(\frac{\Delta\phi}{2} \right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2 \left(\frac{\Delta\lambda}{2} \right)} \right)$$

โดยที่ r คือรัศมีของโลก ($\approx 6371\text{km}$)

ϕ_1, ϕ_2 คือค่า Latitude ของจุดเริ่มต้นและจุดปลาย

λ_1, λ_2 คือค่า Longitude ของจุดเริ่มต้นและจุดปลาย

$$\Delta\phi = \phi_2 - \phi_1$$

$$\Delta\lambda = \lambda_2 - \lambda_1$$

โค้ด

```
import math

def haversine(lat1, lon1, lat2, lon2):
    R = 6371 # รัศมีของโลก (กิโลเมตร)
    phi1 = math.radians(lat1)
    phi2 = math.radians(lat2)
    delta_phi = math.radians(lat2 - lat1)
    delta_lambda = math.radians(lon2 - lon1)

    a = math.sin(delta_phi/2)**2 + math.cos(phi1) * math.cos(phi2) *
math.sin(delta_lambda/2)**2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))

    return R * c

# ตัวอย่างการคำนวณระยะทางระหว่าง Mueang Kalasin และ Na Mon
distance = haversine(16.4322, 103.5067, 16.2206, 103.5622)
print(f"Distance: {distance:.2f} km")
```

ผลลัพธ์

Distance: 24.26 km

สร้างตารางระยะทาง

```
import numpy as np

def create_distance_matrix(data):
    num_locations = len(data)
    distance_matrix = np.zeros((num_locations, num_locations))
    for i in range(num_locations):
        for j in range(num_locations):
            if i != j: # ระยะทางจากตัวเองถึงตัวเองคือ 0
                distance_matrix[i][j] = haversine(
                    data.iloc[i]['Latitude'], data.iloc[i]['Longitude'],
                    data.iloc[j]['Latitude'], data.iloc[j]['Longitude']
                )
    return distance_matrix

# สร้างเมทริกซ์ระยะทางเป็น NxN
distance_matrix = create_distance_matrix(data)
# แสดงตารางระยะทาง
print("Distance Matrix (km):")
print(distance_matrix)
```

ผลลัพธ์

```
Distance Matrix (km):
[[ 0.          38.94301329  58.85987433  17.50370157  23.26471586
  24.68663775  49.65132199  33.10143373  54.71239117  57.72288323
  54.83926452  43.18062896  26.77129182  17.13655604  37.9834496 ]
 [ 38.94301329  0.          38.35244341  25.13922081  58.69358303
  62.13847318  86.4659529   65.2034566   93.64701028  96.115005
  92.60883711  81.65849359  64.71072219  54.11550444  76.89862432]
[...]
[ 37.9834496   76.89862432  92.09004936  53.61198468  23.91297197
  20.14011587  20.43513455  32.34655055  16.77298188  23.46023133
  20.79376256  11.97056562  14.76453828  24.67880513  0.        ]]
]]
```

7.2.3 การหาลำดับเส้นทางที่สั้นที่สุด

หลังจากคำนวณระยะทางระหว่างทุกจุดแล้ว จะต้องสร้างกราฟที่มีระยะทางเป็นน้ำหนัก (Weight) ระหว่างจุด จากนั้นสามารถใช้วิธีการเชิงอิวาริสติก (Heuristic) เช่น Nearest Neighbor หรือ Dynamic Programming เช่น Held-Karp Algorithm เพื่อแก้ปัญหา TSP และหาลำดับการเดินทางที่เหมาะสมที่สุด

จากเมทริกซ์ระยะทางที่ได้ในหัวข้อ 7.2.2 สามารถใช้วิธี Nearest Neighbor เพื่อคำนวณลำดับเส้นทางและระยะทางรวมที่สั้นที่สุดได้ดังนี้

```
def nearest_neighbor(distance_matrix, start_index=0):
    """
    หาลำดับเส้นทางที่สั้นที่สุดด้วยวิธี Nearest Neighbor
    :param distance_matrix: เมทริกซ์ระยะทางระหว่างจุด
    :param start_index: จุดเริ่มต้น(index)
    :return: เส้นทางที่ได้และระยะทางรวม
    """

    num_locations = len(distance_matrix)
    visited = [False] * num_locations # ติดตามจุดที่เยี่ยมชมแล้ว
    path = [start_index] # เริ่มต้นจากจุดเริ่มต้น
    visited[start_index] = True
    total_distance = 0

    current_index = start_index
    for _ in range(num_locations - 1):
        # คำนวณระยะทางไปยังจุดที่ยังไม่ได้เยี่ยมชม
        min_distance = float('inf')
        next_index = -1
        for j in range(num_locations):
            if not visited[j] and distance_matrix[current_index][j] <
min_distance:
                min_distance = distance_matrix[current_index][j]
                next_index = j

        # อัปเดตเส้นทางและระยะทางรวม
        path.append(next_index)
        total_distance += min_distance
        visited[next_index] = True
        current_index = next_index
```

```

# กลับไปปั้งจุดเริ่มต้น
total_distance += distance_matrix[current_index][start_index]
path.append(start_index)

return path, total_distance

# เรียกใช้ฟังก์ชัน Nearest Neighbor
path, total_distance = nearest_neighbor(distance_matrix, start_index=0)

print("Shortest Path (Nearest Neighbor):", path)
print(f"Total Distance: {total_distance:.2f} km")

```

ผลลัพธ์

```

Shortest Path (Nearest Neighbor): [0, 13, 4, 12, 14, 11, 9, 8, 10, 6,
5, 7, 3, 1, 2, 0]
Total Distance: 338.72 km

```

7.2.4 การแสดงผลเส้นทาง

เพื่อทำให้การเดินทางเข้าใจง่ายขึ้น สามารถวาดกราฟแสดงเส้นทางที่ได้บนแผนที่โดยใช้ matplotlib

```

import matplotlib.pyplot as plt

def plot_path(data, path):
    """
    แสดงเส้นทางที่สั้นที่สุดบนแผนที่
    :param data: ข้อมูลคำແໜ່ງ (DataFrame)
    :param path: ลำดับเส้นทาง
    """
    plt.figure(figsize=(12, 8))
    # วัดจุดของแต่ละคำແໜ່ງ
    plt.scatter(data['Longitude'], data['Latitude'], c='blue', s=100,
label='Districts')

    # วัดเส้นทางตามลำดับใน path
    for i in range(len(path) - 1):
        start = path[i]
        end = path[i + 1]
        plt.plot(
            [data.iloc[start]['Longitude'], data.iloc[end]['Longitude']],
            [data.iloc[start]['Latitude'], data.iloc[end]['Latitude']],
            'r-', linewidth=2
        )
    # แสดงชื่อของแต่ละจุด
    for i, row in data.iterrows():

```

```

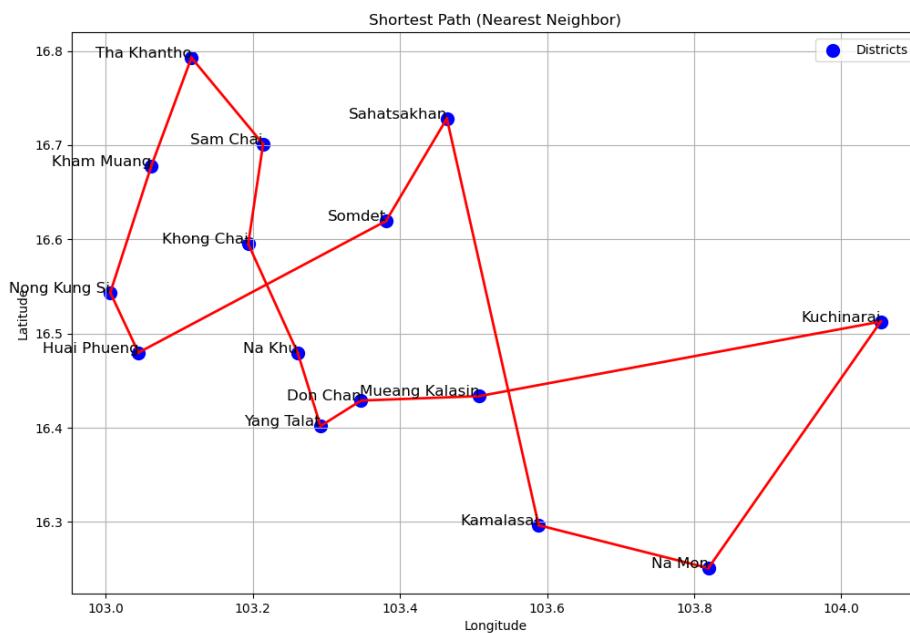
plt.text(row['Longitude'], row['Latitude'], row['District'],
fontsize=12, ha='right')

plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Shortest Path (Nearest Neighbor)')
plt.legend()
plt.grid()
plt.show()

# แสดงผลลัพธ์ทาง
plot_path(data, path)

```

ผลเส้นทาง



7.2.4 การแก้ปัญหา TSP ด้วยไลบรารี pymoo

นอกจากการใช้วิธี Nearest Neighbor ที่เป็น Heuristic แบบง่ายแล้ว ยังมีเครื่องมือและไลบรารีขึ้นสูง เช่น pymoo ที่สามารถใช้แก้ปัญหา TSP ด้วยอัลกอริทึมเชิงวิวัฒนาการ (Evolutionary Algorithms) เช่น Genetic Algorithm (GA) ซึ่งเหมาะสมสำหรับการหาคำตอบที่ใกล้เคียงกับค่าที่ดีที่สุดในปัญหาเชิงคอมพิวเตอร์ เรียล เช่น TSP โดย ขั้นตอนวิธีเชิงพันธุกรรม GA (Genetic Algorithm) เป็นอัลกอริทึมที่เลียนแบบหลักการของวิวัฒนาการในธรรมชาติ เช่น การคัดเลือก การข้ามพันธุ์ และการกลายพันธุ์ ช่วยให้สามารถค้นหาคำตอบในปัญหาที่มีความซับซ้อนสูงได้อย่างมีประสิทธิภาพ GA เหมาะสำหรับปัญหาที่มีความเป็นไปได้ของคำตอบจำนวนมาก เช่น TSP ที่มีชุดเส้นทางหลายล้านแบบ นอกจากนี้ GA ยังสามารถหลีกเลี่ยงการติดในกับดัก (Local Minima) ได้ดีกว่า Heuristic แบบง่าย เช่น Nearest Neighbor

pymoo ไลบรารี

pymoo เป็นไลบรารีในภาษา Python ที่ออกแบบมาเพื่อแก้ปัญหาการหาค่าที่เหมาะสมที่สุด (Optimization) ด้วยอัลกอริทึมเชิงวิวัฒนาการ (Evolutionary Algorithms) และแนวทางหรือวิธีการ

แก้ปัญหาที่เป็นการค้นหาเชิงอนุโลม (Metaheuristics) สามารถใช้ได้ทั้งปัญหาแบบหนึ่งวัตถุประสงค์ (Single-objective) และหลายวัตถุประสงค์ (Multi-objective) จุดเด่นคือใช้งานง่าย มีอัลกอริทึมให้เลือกหลากหลาย เช่น GA NSGA-II DE นอกจากนี้ยังสามารถปรับแต่งหรือสร้างอัลกอริทึมใหม่ ๆ ได้ตามความต้องการ

ติดตั้งไลบรารี

```
pip install pymoo
```

ขั้นตอนวิธีเชิงพันธุกรรม GA (Genetic Algorithm)

- เริ่มจากการสร้างประชากรเริ่มต้น (Initial Population) ซึ่งแต่ละตัวแทน (Chromosome) คือวิชีแก้ปัญหาแบบหนึ่ง เช่น ลำดับของเส้นทางใน TSP
- ประเมินความเหมาะสม (Fitness) ของแต่ละตัวแทน เช่น คำนวณระยะทางรวมของเส้นทาง
- คัดเลือกตัวแทนที่ดีที่สุดบางส่วน (Selection) เพื่อเข้าสู่กระบวนการข้ามพันธุ์
- ทำการข้ามพันธุ์ (Crossover) เพื่อผสมข้อมูลของตัวแทนสองตัวเข้าด้วยกัน
- เพิ่มความหลากหลายให้กับประชากรด้วยการกลายพันธุ์ (Mutation) เช่น สลับจุดในเส้นทางบางจุด
- สร้างประชากรใหม่จากขั้นตอนข้างต้น และวนกลับไปประเมิน Fitness ใหม่
- ทำขั้นตอนซ้ำหลายรอบ (Generation) เพื่อหาคำตอบที่ดีกว่าเดิม
- หยุดเมื่อครบจำนวนรอบหรือเมื่อ Fitness ดีที่สุดคงที่
- คำตอบที่ดีที่สุดคือเส้นทางที่มีระยะทางรวมสั้นที่สุด

โค้ดการใช้ GA

```
from pymoo.algorithms.soo.nonconvex.ga import GA
from pymoo.operators.sampling.rnd import PermutationRandomSampling
from pymoo.operators.crossover.ox import OrderCrossover
from pymoo.operators.mutation.inversion import InversionMutation
from pymoo.optimize import minimize
from pymoo.core.problem import Problem
import numpy as np
import pandas as pd
from math import radians, cos, sin, sqrt, atan2

def haversine(lat1, lon1, lat2, lon2):
    # Radius of Earth in kilometers
    R = 6371.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1
```

```

a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
c = 2 * atan2(sqrt(a), sqrt(1 - a))

distance = R * c
return distance

def create_distance_matrix(data):
    num_locations = len(data)
    distance_matrix = np.zeros((num_locations, num_locations))
    for i in range(num_locations):
        for j in range(num_locations):
            if i != j: # ระยะทางจากตัวเองถึงตัวเองคือ 0
                distance_matrix[i][j] = haversine(
                    data.iloc[i]['Latitude'], data.iloc[i]['Longitude'],
                    data.iloc[j]['Latitude'], data.iloc[j]['Longitude'])
    return distance_matrix

class TSPPProblem(Problem):
    def __init__(self, num_locations):
        super().__init__(n_var=num_locations, n_obj=1, n_constr=0, xl=0,
xu=num_locations-1, type_var=int)

    def _evaluate(self, X, out, *args, **kwargs):
        f = []
        for solution in X:
            dist = 0
            solution = list(solution) + [solution[0]] # กลับมาที่จุดเริ่ม
            for i in range(len(solution) - 1):
                dist += distance_matrix[solution[i]][solution[i + 1]]
            f.append(dist)
        out["F"] = np.array(f)

# โหลดข้อมูล CSV
file_path = 'kalasin_districts_english.csv'
data = pd.read_csv(file_path)

# สร้างเมทริกซ์ระยะทาง
distance_matrix = create_distance_matrix(data)

num_locations = len(data)
problem = TSPPProblem(num_locations)

algorithm = GA()

```

```

        pop_size=100,
        sampling=PermutationRandomSampling(),
        crossover=OrderCrossover(),
        mutation=InversionMutation(),
        eliminate_duplicates=True
    )

    res = minimize(
        problem,
        algorithm,
        termination=('n_gen', 200),
        seed=1,
        save_history=True,
        verbose=True
    )

    print("Best Path:", res.X)
    print(f"Total Distance: {res.F[0]:.2f} km")

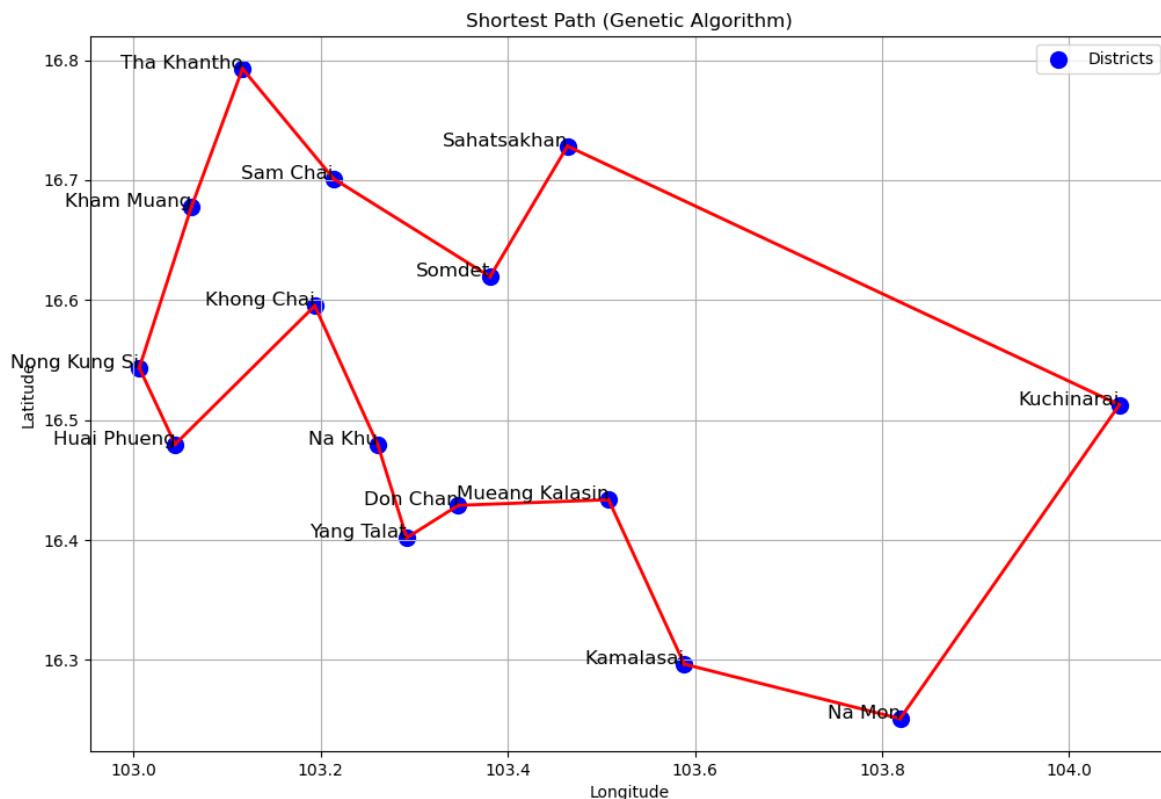
```

ผลลัพธ์

n_gen	n_eval	f_avg	f_min
1	100	6.552401E+02	5.050175E+02
2	200	5.953850E+02	4.857669E+02
3	300	5.650098E+02	4.697904E+02
4	400	5.404026E+02	4.680432E+02
5	500	5.237175E+02	4.680432E+02
6	600	5.092128E+02	4.611627E+02
7	700	4.987799E+02	4.504279E+02
8	800	4.901936E+02	3.948725E+02
9	900	4.802509E+02	3.948725E+02
10	1000	4.714015E+02	3.947083E+02
.			
.			
186	18600	3.104392E+02	3.041589E+02
187	18700	3.104392E+02	3.041589E+02
188	18800	3.104392E+02	3.041589E+02
189	18900	3.104392E+02	3.041589E+02
190	19000	3.103925E+02	3.041589E+02
191	19100	3.103925E+02	3.041589E+02
192	19200	3.103925E+02	3.041589E+02
193	19300	3.103925E+02	3.041589E+02
194	19400	3.103925E+02	3.041589E+02
195	19500	3.102861E+02	3.041589E+02

196	19600	3.102861E+02	3.041589E+02
197	19700	3.101799E+02	3.041589E+02
198	19800	3.101799E+02	3.041589E+02
199	19900	3.101799E+02	3.041589E+02
200	20000	3.101799E+02	3.041589E+02

Best Path: [5 7 2 1 3 0 13 4 12 14 6 10 8 9 11]
Total Distance: 304.16 km



ปัญหาการหาเส้นทางที่สั้นที่สุด (TSP) เพื่อเพิ่มประสิทธิภาพและคุณภาพของการแก้ปัญหาเส้นทางที่สั้นที่สุด (TSP) ให้ดียิ่งขึ้น บทนี้จึงได้นำเสนอการใช้งานไลบรารี pymoo ซึ่งเป็นเครื่องมือสำหรับการแก้ปัญหา Optimization ที่รองรับอัลกอริทึมเชิงวิวัฒนาการ (Evolutionary Algorithms) และ Metaheuristics หลายรูปแบบ หนึ่งในนั้นคือ Genetic Algorithm (GA) ซึ่งเป็นอัลกอริทึมที่ได้รับความนิยมสูงในการแก้ปัญหาที่มีความซับซ้อนสูงอย่างเช่น TSP

GA ทำงานโดยจำลองกระบวนการทางธรรมชาติ เช่น การคัดเลือก (Selection), การข้ามพันธุ์ (Crossover) และการกลายพันธุ์ (Mutation) ซึ่งช่วยให้สามารถค้นหาคำตอบที่ใกล้เคียงกับค่าที่ดีที่สุดได้อย่างมีประสิทธิภาพ GA มีข้อได้เปรียบเหนือวิธีการ Heuristic แบบง่าย เช่น Nearest Neighbor ตรงที่สามารถหลีกเลี่ยงการติดกับดักในจุดต่ำสุดเฉพาะที่ (Local Minima) ได้ดีกว่า และสามารถสำรวจทางเลือกของเส้นทางได้หลากหลายยิ่งขึ้น

บทนี้ได้เลือกใช้ pymoo ในการทำ Optimization กับข้อมูลจริงของอำเภอในจังหวัดกาฬสินธุ์ โดยใช้พิกัด (Latitude และ Longitude) เพื่อคำนวณระยะทางระหว่างจุดต่าง ๆ และหาลำดับเส้นทางที่มีระยะทาง

รวมสั้นที่สุด ผลลัพธ์จากการประมวลผลด้วย GA พบว่าระยะทางรวมที่ได้มักจะสั้นกว่าผลลัพธ์จากวิธี Nearest Neighbor อย่างมีนัยสำคัญ และมีแนวโน้มใกล้เคียงกับคำตอบที่ดีที่สุดในปัญหา TSP

เนื้อหาในส่วนนี้ยังได้แสดงตัวอย่างคัดสำหรับการสร้างคลาสปัญหา TSP (TSP Problem Class) และการเรียกใช้งานอัลกอริทึม GA ของ pymoo พร้อมกับการตั้งค่าตัวแปรสำคัญต่าง ๆ เช่น sampling, crossover และ mutation ที่เหมาะสมกับลักษณะปัญหาที่เป็นแบบ Permutation เช่น TSP

สิ่งที่น่าสนใจคือ นักศึกษาจะสามารถเห็นความแตกต่างระหว่างอัลกอริทึมแบบ Heuristic เป็นต้น (Nearest Neighbor) และอัลกอริทึมแบบ Metaheuristic (GA) ได้อย่างชัดเจน ผ่านการเปรียบเทียบผลลัพธ์ จากข้อมูลจริง นอกจากนี้ยังเป็นตัวอย่างที่ดีในการประยุกต์ใช้ความรู้ด้าน Python และ Data Science เข้ากับข้อมูลเชิงภูมิศาสตร์ของจังหวัดกาฬสินธุ์ ซึ่งสามารถนำไปต่อยอดเพื่อวางแผนเส้นทางขนส่งในโลกความจริงได้อย่างมีประสิทธิภาพ

7.3 การเขียนโปรแกรมระบบอินเตอร์เน็ตของสรรพสิ่ง (IoT)

ระบบอินเตอร์เน็ตของสรรพสิ่ง (Internet of Things - IoT) คือแนวคิดที่นำอุปกรณ์ต่าง ๆ มาสื่อสารและทำงานร่วมกันผ่านเครือข่ายอินเทอร์เน็ต โดยในระบบ IoT จะใช้อุปกรณ์ Microcontroller เช่น Arduino, ESP32 หรือ NodeMCU เพื่อควบคุมเซนเซอร์และอุปกรณ์ไฟฟ้า นอกจากนี้ยังมี Microprocessor อย่าง Raspberry Pi และ Jetson Nano ที่รองรับระบบปฏิบัติการและการประมวลผลข้อมูลที่ซับซ้อน การเขียนโปรแกรม IoT สามารถทำได้ด้วย MicroPython บน ESP32 เพื่อเชื่อมต่อ Wi-Fi อ่านค่าจากเซนเซอร์ และส่งข้อมูลไปยังคลาวด์ เช่น การอ่านอุณหภูมิด้วย DHT11 และแสดงผลบนแพลตฟอร์ม IoT ช่วยให้การพัฒนา IoT มีประสิทธิภาพและรวดเร็วมากยิ่งขึ้น

IoT ประกอบด้วยองค์ประกอบหลัก 3 ส่วนสำคัญ ได้แก่ Sensor Actuator และ Connectivity ซึ่งทำงานร่วมกันเพื่อสร้างระบบที่เชื่อมต่อและทำงานอัตโนมัติได้อย่างชาญฉลาด โดยรายละเอียดของแต่ละส่วน มีดังนี้

1. Sensor (ตัวเซนเซอร์) เซนเซอร์เป็นอุปกรณ์ที่ใช้ในการตรวจจับค่าต่าง ๆ ในสภาพแวดล้อมหรือระบบ เช่น อุณหภูมิ ความชื้น แสง ความดัน หรือการเคลื่อนไหว โดยเซนเซอร์จะแปลงค่าทางกายภาพเหล่านี้ให้อยู่ในรูปแบบข้อมูลดิจิทัลที่สามารถนำไปประมวลผลได้ ตัวอย่างเซนเซอร์ที่ใช้ในระบบ IoT ได้แก่

- DHT11/DHT22: เซนเซอร์วัดอุณหภูมิและความชื้น
- Ultrasonic Sensor (HC-SR04): เซนเซอร์วัดระยะทางโดยใช้คลื่นเสียง
- LDR (Light Dependent Resistor): เซนเซอร์วัดแสง
- MQ Series เซนเซอร์ตรวจจับแก๊สชนิดต่าง ๆ เช่น คาร์บอนไดออกไซด์ หรือก๊าซไวไฟ

2. Actuator (ตัวกระตุ้น) แอกซูเอเตอร์เป็นอุปกรณ์ที่ทำหน้าที่ตอบสนองต่อคำสั่งที่ได้รับจากระบบ IoT โดยจะแปลงข้อมูลดิจิทัลเป็นการกระทำในรูปแบบทางกายภาพ เช่น การเปิด-ปิด การเคลื่อนที่ หรือการปล่อยพลังงาน ตัวอย่างแอกซูเอเตอร์ที่ใช้ใน IoT ได้แก่

- Relay Module ใช้ควบคุมการเปิด-ปิดวงจรไฟฟ้า เช่น เปิด-ปิดหลอดไฟหรือเครื่องใช้ไฟฟ้า
- Servo Motor ใช้ในการควบคุมการหมุนหรือการเคลื่อนที่ในตำแหน่งที่กำหนด
- Stepper Motor ใช้ในระบบที่ต้องการการเคลื่อนไหวที่ละเอียด เช่น หุ่นยนต์หรือแขนกล
- Buzzer ใช้ส่งสัญญาณเสียง
- Solenoid Valve ใช้ในระบบควบคุมการไหลของของเหลวหรือแก๊ส

3. Connectivity (การเชื่อมต่อ) การเชื่อมต่อในระบบ IoT มีบทบาทสำคัญในการส่งข้อมูลระหว่างเซนเซอร์ แอกซูเอเตอร์ และแพลตฟอร์ม IoT หรือคลาวด์ โดยการเชื่อมต่อสามารถทำได้หลายรูปแบบ เช่น

- Wi-Fi การเชื่อมต่อผ่านเครือข่ายไร้สาย เช่น ESP32 ที่รองรับการเชื่อมต่อ Wi-Fi หมายสำหรับการส่งข้อมูลแบบเรียลไทม์
- Bluetooth ใช้ในระบบที่ต้องการการเชื่อมต่อระยะใกล้ เช่น การส่งข้อมูลจากเซนเซอร์ไปยังโทรศัพท์มือถือ
- LoRa การเชื่อมต่อที่ใช้พลังงานต่ำและครอบคลุมระยะไกล หมายสำหรับระบบ IoT ในพื้นที่ห่างไกล
- ZigBee ใช้ในการสร้างเครือข่าย IoT ภายในบ้าน เช่น การควบคุมอุปกรณ์智能家居 ไปยังเซิร์ฟเวอร์
- MQTT Protocol โปรโตคอลการสื่อสารที่นิยมใน IoT ใช้สำหรับส่งข้อมูลจากอุปกรณ์ไปยังเซิร์ฟเวอร์
- หรือคลาวด์อย่าง迫切 หมายด้วยการเปลี่ยนแปลง
- Ethernet ใช้ในระบบที่ต้องการความเสถียรสูง เช่น ระบบควบคุมอุตสาหกรรม

การทำงานร่วมกันของ Sensor Actuator และ Connectivity โดย Sensor จะตรวจจับข้อมูลจากสภาพแวดล้อม เช่น อุณหภูมิ ความชื้น หรือการเคลื่อนไหว Connectivity จะส่งข้อมูลที่ได้จากเซนเซอร์ไปยังตัวประมวลผล เช่น Microcontroller (ESP32/Arduino) หรือ Microprocessor (Raspberry Pi) ผ่านเครือข่าย Wi-Fi หรือ Bluetooth ระบบประมวลผลข้อมูลจากเซนเซอร์ และตัดสินใจส่งคำสั่งไปยัง Actuator เพื่อดำเนินการ เช่น เปิดไฟ ปิดวาล์ว หรือหมุนมอเตอร์

ตัวอย่างการใช้งานในระบบ IoT

- ระบบบ้านอัจฉริยะ (Smart Home) เซนเซอร์วัดอุณหภูมิส่งข้อมูลไปยังระบบประมวลผล และแอกซูเอเตอร์จะเปิดเครื่องปรับอากาศเมื่ออุณหภูมิสูงเกินกำหนด
- การเกษตรอัจฉริยะ (Smart Farming) เซนเซอร์ความชื้นในดินส่งข้อมูลผ่าน LoRa ไปยังระบบควบคุม และแอกซูเอเตอร์จะเปิดระบบน้ำหยดเมื่อดินแห้ง

- ระบบโปรแกรมอุตสาหกรรม (Industrial IoT) เช่นเซอร์ตรวจับความร้อนของเครื่องจักรส่งข้อมูลผ่าน MQTT ไปยังแพลตฟอร์มการวิเคราะห์ และแอคชันเตอร์จะปิดเครื่องจักรเมื่ออุณหภูมิสูงเกินค่าที่กำหนด

ระบบ IoT ได้รับความสนใจอย่างมากในยุคปัจจุบัน โดยช่วยให้สามารถเชื่อมต่อและควบคุมอุปกรณ์ต่าง ๆ ผ่านอินเทอร์เน็ตได้ ไม่ว่าจะเป็นการเก็บข้อมูล การควบคุมระบบไฟฟ้าในบ้าน หรือการวัดค่าต่าง ๆ ในงานอุตสาหกรรม การนำ MicroPython และ ESP32 มาใช้พัฒนา IoT ทำให้การพัฒนาระบบ IoT มีความง่ายขึ้น เนื่องจาก MicroPython เป็นภาษาสคริปต์ที่เหมาะสมกับการเขียนโปรแกรมสำหรับไมโครคอนโทรลเลอร์ และ ESP32 เป็นบอร์ดไมโครคอนโทรลเลอร์ที่ทรงพลังพร้อม Wi-Fi และ Bluetooth ในตัว

7.3.1 แนะนำ MicroPython และ ESP32

MicroPython แสดงตั้งรูปที่ 7.1 เป็นเวอร์ชันของภาษา Python ที่ถูกพัฒนาให้เหมาะสมกับไมโครคอนโทรลเลอร์ มีขนาดเล็กและประสิทธิภาพสูง รองรับการเขียนโปรแกรมสำหรับการควบคุมอุปกรณ์ IoT ต่าง ๆ อย่างง่ายดายคุณสมบัติเด่น ได้แก่ Syntax ที่เข้าใจง่ายเหมือน Python ใช้งานร่วมกับเซ็นเซอร์และอุปกรณ์ไฟฟ้าต่าง ๆ และรองรับบอร์ดไมโครคอนโทรลเลอร์ยอดนิยม เช่น ESP32 และ ESP8266



รูปที่ 7.1 แสดงไอคอนของ MicroPython

ที่มา <https://pyseek.com/2024/09/what-is-micropython/>

ตารางที่ 7.1 สรุปหน้าที่ของ GPIO ESP32 DEV KIT V1

หมายเลข ขา	ชื่อขา	หน้าที่ (Function)	หมายเลข ขา	ชื่อขา	หน้าที่ (Function)
1	3V3	แรงดันไฟฟ้า 3.3V สำหรับจ่ายไฟให้กับอุปกรณ์ภายนอก	6	IO0	GPIO0: ขา_out ประสีค์, ใช้สำหรับโหมดบูตโหลดเตอร์
2	GND	กราวด์ (Ground) สำหรับวงจร	17	IO2	GPIO2: ขา_out ประสีค์, รองรับ PWM และ ADC
3	VIN	แรงดันไฟฟ้าขาเข้า (5V) สำหรับจ่ายไฟให้กับบอร์ด	18	IO15	GPIO15: ขา_out ประสีค์, รองรับ PWM และ ADC
4	EN	ขา_eject สำหรับเปิด/ปิดการทำงานของ ESP32	19	IO13	GPIO13: ขา_out ประสีค์, รองรับ PWM และ ADC

5	IO23	GPIO23: ขาอ่อนกประสงค์ สำหรับอินพุต/เอาต์พุต	20	IO12	GPIO12: ขาอ่อนกประสงค์, รองรับ PWM และ ADC
6	IO22	GPIO22: ขาอ่อนกประสงค์, รองรับ I2C SCL	21	IO14	GPIO14: ขาอ่อนกประสงค์, รองรับ PWM และ ADC
7	IO1	GPIO1: ขาสำหรับ UART TX0	22	IO27	GPIO27: ขาอ่อนกประสงค์, รองรับ PWM และ ADC
8	IO3	GPIO3: ขาสำหรับ UART RX0	23	IO26	GPIO26: ขาอ่อนกประสงค์, รองรับ PWM และ ADC
9	IO21	GPIO21: ขาอ่อนกประสงค์, รองรับ I2C SDA	24	IO25	GPIO25: ขาอ่อนกประสงค์, รองรับ PWM และ ADC
10	IO19	GPIO19: ขาอ่อนกประสงค์, รองรับ SPI	25	IO33	GPIO33: ขาอ่อนกประสงค์, รองรับ PWM และ ADC
11	IO18	GPIO18: ขาอ่อนกประสงค์, รองรับ SPI	26	IO32	GPIO32: ขาอ่อนกประสงค์, รองรับ PWM และ ADC
12	IO5	GPIO5: ขาอ่อนกประสงค์, รองรับ SPI	27	IO35	GPIO35: ขาอ่อนกประสงค์, รองรับ ADC (อินพุตเท่านั้น)
13	IO17	GPIO17: ขาอ่อนกประสงค์, รองรับ UART	28	IO34	GPIO34: ขาอ่อนกประสงค์, รองรับ ADC (อินพุตเท่านั้น)
14	IO16	GPIO16: ขาอ่อนกประสงค์, รองรับ UART	29	IO39	GPIO39: ขาอ่อนกประสงค์, รองรับ ADC (อินพุตเท่านั้น)
15	IO4	GPIO4: ขาอ่อนกประสงค์, รองรับ PWM และ ADC	30	IO36	GPIO36: ขาอ่อนกประสงค์, รองรับ ADC (อินพุตเท่านั้น)

ESP32 เป็นไมโครคอนโทรลเลอร์ที่มาพร้อมกับโมดูล Wi-Fi และ Bluetooth รองรับการพัฒนา IoT อย่างสมบูรณ์แบบ โดยมีราคาไม่แพงและความสามารถหลากหลาย เช่น รองรับ Wi-Fi และ Bluetooth มี GPIO หลายพินสำหรับเชื่อมต่อกับเซ็นเซอร์ และใช้พลังงานต่ำ เหมาะสำหรับงานที่ใช้แบตเตอรี่

7.3.2 โปรแกรม IoT โดยใช้ MicroPython และ ESP32

การควบคุม LED ผ่าน Wi-Fi ด้วยการต่อ LED บน ESP32 ผ่านเครือข่าย Wi-Fi

```
import network
import socket
from machine import Pin

# การตั้งค่า Wi-Fi
ssid = 'Your_SSID'
password = 'Your_PASSWORD'

# การเชื่อมต่อ Wi-Fi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)
```

```

print("Connecting to Wi-Fi...")
while not wlan.isconnected():
    pass
print("Connected to Wi-Fi:", wlan.ifconfig())

# ตั้งค่า GPIO
led = Pin(2, Pin.OUT)

# สร้าง Web Server
def web_page():
    html = """<!DOCTYPE html>
<html>
<head><title>ESP32 LED Control</title></head>
<body>
<h1>ESP32 LED Control</h1>
<p>LED is currently: {}</p>
<a href="/?led=on"><button>Turn ON</button></a>
<a href="/?led=off"><button>Turn OFF</button></a>
</body>
</html>""".format("ON" if led.value() else "OFF")
    return html

# เปิดเซิร์ฟเวอร์
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
server = socket.socket()
server.bind(addr)
server.listen(1)

print("Web server running at:", addr)

while True:
    conn, addr = server.accept()
    print("Connection from:", addr)
    request = conn.recv(1024)
    request = str(request)

    if '/?led=on' in request:
        led.value(1)
    elif '/?led=off' in request:
        led.value(0)

    response = web_page()
    conn.send('HTTP/1.1 200 OK\nContent-Type: text/html\n\n')
    conn.send(response)
    conn.close()

```

การอ่านค่าจากเซ็นเซอร์ DHT11 ตัวอย่างนี้แสดงการอ่านค่าความชื้นและอุณหภูมิจากเซ็นเซอร์ DHT11 และส่งข้อมูลไปยัง Web Server

```
import dht
from machine import Pin
import time

# ตั้งค่า DHT11
sensor = dht.DHT11(Pin(4))

while True:
    try:
        sensor.measure()
        temp = sensor.temperature()
        hum = sensor.humidity()
        print(f"Temperature: {temp}°C, Humidity: {hum}%")
        time.sleep(2)
    except OSError as e:
        print("Error reading sensor:", e)
```

7.3.3 โปรแกรม IoT โดยใช้ MicroPython และ ESP32

ESP32 สามารถส่งข้อมูลไปยัง MQTT Broker เพื่อใช้ในระบบ IoT ได้อย่างง่ายดาย ตัวอย่างการใช้งาน MQTT

```
from umqtt.simple import MQTTClient
import network
from machine import Pin
import time

# ตั้งค่า Wi-Fi
ssid = 'Your_SSID'
password = 'Your_PASSWORD'

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)

while not wlan.isconnected():
    pass
print("Connected to Wi-Fi:", wlan.ifconfig())

# ตั้งค่า MQTT
broker = 'test.mosquitto.org'
client = MQTTClient("ESP32", broker)

client.connect()
print("Connected to MQTT Broker")
```

```
# ส่งข้อมูล
while True:
    temperature = 25 # สมมติค่าที่อ่านได้
    client.publish(b"home/temperature", str(temperature))
    print(f"Published temperature: {temperature}°C")
    time.sleep(5)
```

7.3.4 การควบคุมอุปกรณ์ IoT ผ่านแอปพลิเคชันบนสมาร์ทโฟน

การควบคุมอุปกรณ์ IoT ผ่านสมาร์ทโฟนเป็นหนึ่งในแนวทางสำคัญของการพัฒนา Smart Home หรือ Smart Factory ในยุคปัจจุบัน โดยช่วยให้ผู้ใช้งานสามารถสั่งการและตรวจสอบสถานะของอุปกรณ์ได้จากทุกที่ผ่านเครือข่ายอินเทอร์เน็ต ซึ่งในหัวข้อนี้จะเน้นการเชื่อมต่อ ESP32 กับแอปพลิเคชันยอดนิยมบนสมาร์ทโฟน เช่น Blynk หรือ Telegram Bot

โดยระบบจะใช้ ESP32 เป็นตัวควบคุมหลัก ซึ่งเชื่อมต่อกับอินเทอร์เน็ตผ่าน Wi-Fi และรอรับคำสั่งจากแอปพลิเคชันบนมือถือ เช่น สั่งเปิด-ปิดอุปกรณ์หรืออ่านค่าจากเซ็นเซอร์ โดยข้อมูลจะถูกส่งผ่าน Cloud Service ของแอปพลิเคชัน เช่น เซิร์ฟเวอร์ของ Blynk หรือ Telegram API ที่เชื่อมต่อกับ ESP32 ผ่าน HTTP หรือ MQTT Protocol

หนึ่งในฟังก์ชันสำคัญของระบบ IoT คือการควบคุมอุปกรณ์และตรวจสอบข้อมูลจากระยะไกลผ่านแอปพลิเคชันบนสมาร์ทโฟน ช่วยให้ผู้ใช้งานสามารถสั่งการอุปกรณ์และตรวจสอบข้อมูลได้ตลอดเวลา ไม่ว่าจะอยู่ที่ใดก็ตาม ตัวอย่างการใช้งานที่นิยม ได้แก่ การเปิด-ปิดอุปกรณ์ไฟฟ้า เช่น หลอดไฟหรือรีเลย์ ที่เชื่อมต่อกับ ESP32 ผ่านแอป Blynk หรือแอปควบคุมอื่น ๆ ทำให้สามารถสั่งเปิดไฟบ้านก่อนลีบบ้านได้อย่างสะดวก

นอกจากนี้ ระบบยังสามารถอ่านค่าจากเซ็นเซอร์ เช่น อุณหภูมิและความชื้นจาก DHT11 หรือ DHT22 และส่งข้อมูลไปแสดงผลบนแอปในรูปแบบ Dashboard ช่วยให้ผู้ใช้งานสามารถติดตามสภาพแวดล้อมหรือสถานะของอุปกรณ์ได้แบบเรียลไทม์

สิ่งที่ต้องเตรียมก่อนเริ่มต้นใช้งาน

ก่อนที่จะสามารถควบคุม ESP32 ผ่านแอปพลิเคชัน Blynk ได้นั้น ผู้ใช้งานจะต้องเตรียมความพร้อมในหลายส่วน ดังนี้

ขั้นแรกให้ติดตั้ง Blynk Library สำหรับ MicroPython ซึ่งเป็นไลบรารีที่ช่วยให้ ESP32 สามารถเชื่อมต่อและสื่อสารกับเซิร์ฟเวอร์ของ Blynk ได้อย่างมีประสิทธิภาพ จากนั้นดาวน์โหลดและติดตั้งแอปพลิเคชัน Blynk ลงบนสมาร์ทโฟน ซึ่งรองรับทั้งระบบ iOS และ Android

เมื่อเข้าสู่แอป Blynk แล้ว ให้สร้าง Project ใหม่ เพื่อเริ่มต้นการเชื่อมต่อกับ ESP32 ในโปรเจกต์นี้ ผู้ใช้จะต้องเพิ่ม Button Widget ที่เชื่อมกับ Virtual Pin V0 ซึ่งจะใช้สำหรับควบคุมการเปิด-ปิดของหลอดไฟ หรือ LED ที่เชื่อมต่อกับ ESP32

จากนั้นให้เพิ่ม Label Widget อีกสองตัวเพื่อแสดงค่าต่าง ๆ จากเซ็นเซอร์ โดยเชื่อม Label แรกกับ Virtual Pin V1 สำหรับแสดงค่าอุณหภูมิ และเชื่อม Label ที่สองกับ Virtual Pin V2 สำหรับแสดงค่าความชื้นที่อ่านได้จากเซ็นเซอร์ DHT11 หรือ DHT22

สุดท้าย คัดลอก Auth Token ที่ระบบของ Blynk สร้างขึ้นสำหรับโปรเจกต์นี้ แล้วนำไปใส่ในตัวแปร BLYNK_AUTH ภายในโค้ดของ ESP32 เพื่อให้การเชื่อมต่อ กับเซิร์ฟเวอร์ Blynk สามารถทำงานได้อย่างสมบูรณ์

โค้ดตัวอย่างงาน

```
import network
import BlynkLib
import time
from machine import Pin
import dht

# ใส่ Token ของโปรเจกต์ Blynk ของคุณ
BLYNK_AUTH = 'Your_Blynk_Auth_Token'

# ตั้งค่า Wi-Fi
ssid = 'Your_SSID'
password = 'Your_PASSWORD'

# เชื่อมต่อ Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(ssid, password)
while not wifi.isconnected():
    pass
print('Wi-Fi connected:', wifi.ifconfig())

# เริ่มต้นใช้งาน Blynk
blynk = BlynkLib.Blynk(BLYNK_AUTH)

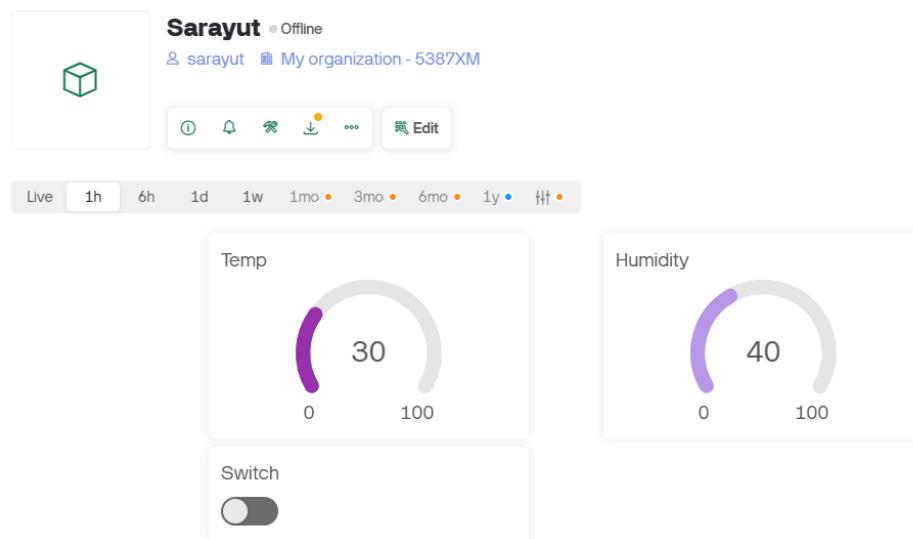
# ตั้งค่า GPIO
led = Pin(2, Pin.OUT) # เปลี่ยนเป็น핀ของรีเลย์หรือ LED จริง
sensor = dht.DHT11(Pin(4)) # DHT11 ต่อ กับ GPIO4

# เมื่อคุณใน Blynk (Virtual Pin V0)
@blynk.on("V0")
def v0_write_handler(value):
    if int(value[0]) == 1:
        led.value(1)
        print("LED ON")
    else:
        led.value(0)
        print("LED OFF")
```

```
# Loop อ่านค่าจาก DHT11 และส่งไปยังแอป Blynk (Virtual Pin V1 และ V2)
def read_dht_and_send():
    try:
        sensor.measure()
        temp = sensor.temperature()
        hum = sensor.humidity()
        blynk.virtual_write(1, temp) # ส่งไป V1
        blynk.virtual_write(2, hum) # ส่งไป V2
        print(f"Temp: {temp}°C, Hum: {hum}%")
    except Exception as e:
        print("Error reading sensor:", e)

# วนลูป Blynk + อ่านเซ็นเซอร์ทุก 5 วินาที
while True:
    blynk.run()
    read_dht_and_send()
    time.sleep(5)
```

แสดงตัวอย่างหน้าจอบน Blynk



ผู้แต่งเห็นว่า IoT เป็น "สภาพแวดล้อม" ระหว่างโลกดิจิทัลกับโลกจริง การเข้าใจองค์ประกอบของระบบ IoT ทั้งเซ็นเซอร์ ตัวขับเคลื่อน และการเชื่อมต่อ จะช่วยให้สามารถออกแบบระบบที่เชื่อมโยงอุปกรณ์ต่างๆ ได้อย่างมีประสิทธิภาพ การได้เรียนรู้ MicroPython และ ESP32 ซึ่งเป็นเครื่องมือยอดนิยมในงาน IoT ยิ่งช่วยให้สามารถพัฒนาโครงการที่ควบคุมอุปกรณ์จริง เช่น ระบบบ้านอัจฉริยะ (Smart Home) หรือการเกษตรอัจฉริยะ (Smart Farming) ได้อย่างรวดเร็วและง่ายดาย ถือเป็นทักษะที่ตลาดแรงงานยุคใหม่ต้องการอย่างมาก

7.4 การเขียนโปรแกรมระบบการเรียนรู้ของเครื่อง (Machine Learning)

ในยุคดิจิทัลปัจจุบันการเรียนรู้ของเครื่อง (Machine Learning : ML) หรือระบบการเรียนรู้ของเครื่อง ได้กลายเป็นเทคโนโลยีที่มีบทบาทสำคัญในหลากหลายสาขา ตั้งแต่การแพทย์ การเงิน ไปจนถึงการพัฒนาซอฟต์แวร์ ที่มีความซับซ้อนและมีประสิทธิภาพสูง โดย ML เป็นกระบวนการที่ทำให้คอมพิวเตอร์สามารถเรียนรู้จากข้อมูลและนำความรู้นั้นมาใช้ในการทำงานหรือจำแนกข้อมูลใหม่โดยไม่ต้องเขียนโปรแกรมคำสั่งเฉพาะตัว ระบบจะพยายามค้นหารูปแบบหรือความสัมพันธ์ของข้อมูลที่ซับซ้อนเพื่อช่วยในการตัดสินใจที่แม่นยำยิ่งขึ้น

หนึ่งในไลบรารีที่ได้รับความนิยมสูงสุดในวงการพัฒนา ML ด้วยภาษา Python คือ scikit-learn (sklearn) เนื่องจากเป็นไลบรารีที่ใช้งานง่าย มีเครื่องมือสำหรับการเรียนรู้ของเครื่องครบถ้วน ไม่ว่าจะเป็น อัลกอริทึมสำหรับการคาดถอย (Regression), การจำแนกประเภท (Classification) หรือการจัดกลุ่ม (Clustering) รวมถึงฟังก์ชันในการเตรียมข้อมูล ประเมินโมเดล และการปรับจูโน่โมเดล (Hyperparameter Tuning) ทำให้ scikit-learn เป็นเครื่องมือที่เหมาะสมสำหรับนักพัฒนาและนักวิจัยทั้งมือใหม่และมืออาชีพ

บทนี้จะนำเสนอการเขียนโปรแกรม ML ด้วย Scikit-learn ผ่านสองตัวอย่างที่น่าสนใจ ได้แก่ การทำนายราคาบ้านโดยใช้ข้อมูล California Housing และการจำแนกภาพตัวเลขจากชุดข้อมูล MNIST ซึ่งเป็น ตัวอย่างที่พับได้บ่อยในงานจริง โดยจะครอบคลุมตั้งแต่ขั้นตอนการเตรียมข้อมูล การเลือกโมเดล การฝึกสอน โมเดล (Training) และการทดสอบประสิทธิภาพของโมเดล (Evaluation) เพื่อให้เข้าใจหลักการทำงานของ ML ในบริบทที่นำไปใช้งานได้จริง

7.4.1 การทำนายราคาบ้าน

การทำนายราคาบ้านเป็นตัวอย่างที่สำคัญของงาน Machine Learning ประเภท Regression ซึ่ง เป็นกระบวนการทำนายค่าตัวเลขต่อเนื่อง เช่น การทำนายราคาบ้านในพื้นที่ต่าง ๆ จากข้อมูลที่เกี่ยวข้อง โดย ชุดข้อมูลที่นิยมนำมาใช้ในกรณีศึกษานี้คือ California Housing Dataset ซึ่งเป็นชุดข้อมูลที่รวบรวมข้อมูลจริง เกี่ยวกับอสังหาริมทรัพย์ในรัฐแคลิฟอร์เนีย ประเทศสหรัฐอเมริกา

ชุดข้อมูลนี้ประกอบด้วยตัวแปรเชิงตัวเลข (Numeric Features) ที่มีความสัมพันธ์กับราคาบ้าน เช่น จำนวนห้องนอนเฉลี่ยต่อบ้าน (Average Bedrooms), รายได้เฉลี่ยของครัวเรือนในพื้นที่ (Median Income), จำนวนประชากร (Population), และข้อมูลเชิงภูมิศาสตร์อย่างค่าละติจูด (Latitude) และลองจิจูด (Longitude) ที่แสดงตำแหน่งของบ้านแต่ละหลัง การใช้ข้อมูลเหล่านี้จะช่วยให้โมเดลสามารถเรียนรู้รูปแบบ ความสัมพันธ์ระหว่างตัวแปรกับราคาบ้านในพื้นที่นั้น ๆ ได้อย่างมีประสิทธิภาพ

ในการสร้างโมเดลทำนายราคาบ้าน บทนี้จะใช้ scikit-learn ซึ่งเป็นไลบรารี Machine Learning ยอดนิยมที่มีเครื่องมือสำหรับสร้างและประเมินโมเดล Regression หลากหลายรูปแบบ ตัวอย่างโมเดลที่ง่าย ต่อการเริ่มต้นคือ Linear Regression ซึ่งเป็นโมเดลเชิงเส้นที่พยากรณ์ความสัมพันธ์เชิงเส้นระหว่างตัวแปร อิสระและราคาบ้าน (ตัวแปรตาม) นอกจากนี้ผู้เรียนยังสามารถต่อ�อดไปใช้ Random Forest Regressor ซึ่ง

เป็นอัลกอริทึมแบบ Ensemble Learning ที่มีความสามารถในการจัดการข้อมูลที่ไม่เป็นเชิงเส้น (Non-linear) และลดปัญหาการการเหรอจนเกิน (Overfitting) ได้ดีขึ้น

การทำนายราคาบ้านด้วย Machine Learning ถือเป็นตัวอย่างที่ดีในการแสดงให้เห็นถึงขั้นตอนพื้นฐานของงาน Regression ตั้งแต่การเตรียมข้อมูล (Data Preprocessing), การสร้างและฝึกสอนโมเดล (Training), ไปจนถึงการประเมินผล (Evaluation) และการปรับปรุงโมเดลให้มีความแม่นยำสูงขึ้นตามความต้องการ

โค้ดตัวอย่างการทำนายราคาบ้าน

```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# โหลดข้อมูล California Housing
data = fetch_california_housing()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

# แบ่งข้อมูลเป็นชุดฝึกและชุดทดสอบ
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# สร้างโมเดล Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)

# ทำนายราคาบ้าน
y_pred = model.predict(X_test)

# คำนวณ Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# สรุมเลือกข้อมูล 100 ตัวอย่างจากชุดทดสอบเพื่อแสดงผล
num_samples = 100
indices = np.random.choice(range(len(y_test)), num_samples, replace=False)

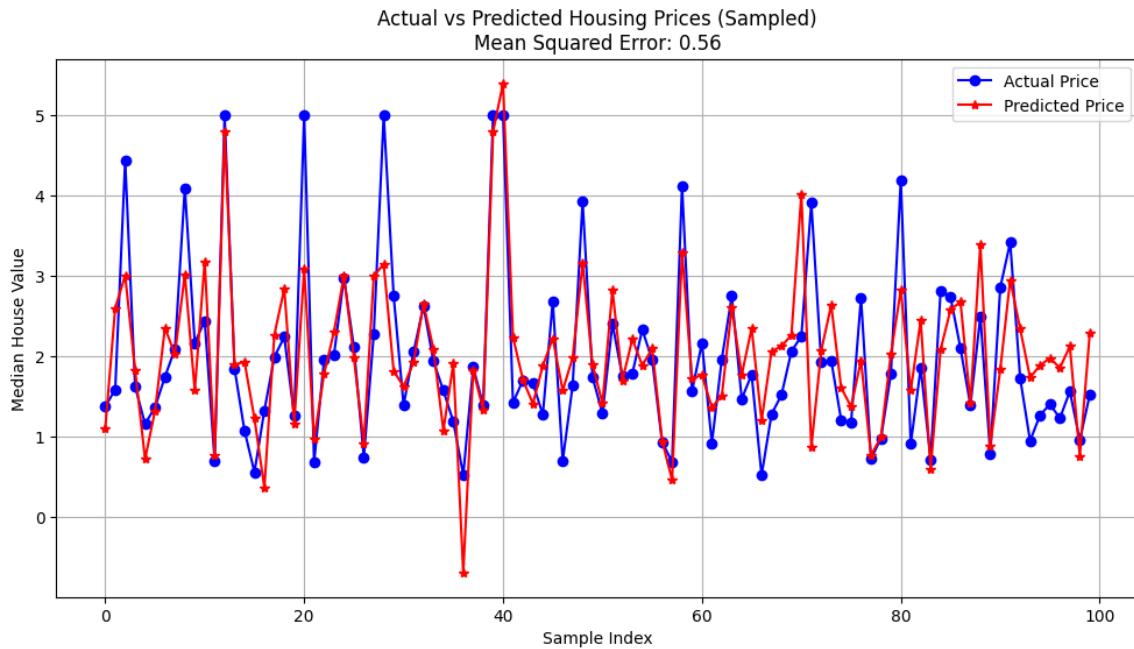
# สร้างกราฟเปรียบเทียบ ค่าจริง VS ค่าทำนาย
plt.figure(figsize=(12, 6))
plt.plot(range(num_samples), y_test.iloc[indices], 'bo-', label='Actual Price')
plt.plot(range(num_samples), y_pred[indices], 'r*-', label='Predicted Price')
plt.xlabel('Sample Index')
```

```

plt.ylabel('Median House Value')
plt.title(f'Actual vs Predicted Housing Prices (Sampled)\nMean Squared Error: {mse:.2f}')
plt.legend()
plt.grid(True)
plt.show()

```

แสดงผลลัพธ์



กราฟนี้แสดงการเปรียบเทียบระหว่างราคากำบังจริงและราคาก่อตัวที่โมเดล Linear Regression ทำนายได้จากชุดข้อมูล California Housing โดยเส้นสีน้ำเงินแสดงราคากำบังจริง ส่วนเส้นสีแดงแสดงค่าที่ทำนาย โมเดลสามารถจับแผลโน้มของข้อมูลได้ในระดับหนึ่ง แต่ยังมีความคลาดเคลื่อนในบางช่วง ค่า MSE ที่ 0.56 บ่งชี้ว่า โมเดลยังมีข้อผิดพลาดและอาจปรับปรุงได้ด้วยโมเดลที่ซับซ้อนขึ้น

7.4.2 การจำแนกกรุปภาพตัวเลข

การจำแนกภาพ (Image Classification) คือกระบวนการที่คอมพิวเตอร์หรือระบบปัญญาประดิษฐ์เรียนรู้ที่จะจำแนกหรือจัดประเภทของรูปภาพให้อยู่ในกลุ่มหรือหมวดหมู่ที่ถูกต้อง เช่น การจำแนกภาพสัตว์ เป็น "สุนัข" หรือ "แมว" หรือในกรณีนี้คือการจำแนกภาพตัวเลขเขียนด้วยมือให้อยู่ในหมวดหมู่ตัวเลขตั้งแต่ 0 ถึง 9 การจำแนกภาพจัดเป็นหนึ่งในปัญหาพื้นฐานของงาน Machine Learning ประเภท Classification ที่มีความท้าทายเนื่องจากข้อมูลภาพมักมีความหลากหลายสูง ภาพแต่ละภาพอาจมีลักษณะตัวอักษรที่แตกต่างกัน เช่น ขนาด รูปร่าง หรือความเอียง ซึ่งอาจเกิดจากความไม่สม่ำเสมอในการเขียนของมนุษย์

หนึ่งในชุดข้อมูลที่ได้รับความนิยมอย่างสูงในการศึกษาปัญหาการจำแนกภาพคือ MNIST (Modified National Institute of Standards and Technology Database) ซึ่งเป็นฐานข้อมูลภาพตัวเลขเขียนด้วยมือ ที่มีขนาด 28x28 พิกเซล รวมทั้งสิ้นกว่า 70,000 ภาพ แบ่งเป็นข้อมูลสำหรับฝึกสอน (Training) และข้อมูล

สำหรับทดสอบ (Testing) โดยข้อมูลแต่ละชุดจะประกอบด้วยภาพตัวเลขที่หลากหลายสไตล์ เช่น ตัวเลขที่เขียนตัวหนา ตัวอิง ตัวบาง หรือมีตำแหน่งในภาพที่ต่างกันเล็กน้อย ชุดข้อมูล MNIST จึงถือเป็นมาตรฐานเบื้องต้นสำหรับการทดสอบโมเดล Classification ทั้งในงานวิจัยและในอุตสาหกรรม นอกจากนี้ ยังเป็นชุดข้อมูลที่ง่ายต่อการเริ่มต้นสำหรับนักศึกษาหรือผู้ที่เพิ่งเริ่มต้นศึกษา Machine Learning

ในตัวอย่างนี้จะใช้โมเดล K-Nearest Neighbors (KNN) ซึ่งเป็นอัลกอริทึม Machine Learning แบบง่ายที่ไม่ต้องผ่านขั้นตอนการฝึกสอนโมเดล (Training Phase) แบบหนักเมื่อนอัลกอริทึมอื่น ๆ หลักการของ KNN คือ เมื่อต้องการจำแนกข้อมูลใหม่ ระบบจะเปรียบเทียบข้อมูลใหม่นั้นกับข้อมูลในชุดฝึกสอน และเลือก "k ตัวอย่างที่ใกล้เคียงที่สุด" (k Nearest Neighbors) มาเป็นตัวแทนในการตัดสินใจ หากในจำนวน k ตัวอย่างนี้ ตัวอย่างส่วนใหญ่เป็นตัวเลขใด โมเดลก็จะทำนายว่าข้อมูลใหม่เป็นตัวเลขนั้น เช่น หากข้อมูลใหม่มีเพื่อนบ้านใกล้สุด 5 ตัว ($k=5$) และมีตัวเลข "7" ปรากฏมากที่สุดใน 5 ตัวอย่างนั้น โมเดลก็จะจำแนกข้อมูลใหม่ว่า เป็น "7"

อัลกอริทึม KNN จึงเหมาะสมสำหรับผู้เริ่มต้นที่ต้องการทำความเข้าใจพื้นฐานของการจำแนกภาพและแนวคิดการวัดระยะทางระหว่างข้อมูล โดยเฉพาะเมื่อใช้กับชุดข้อมูล MNIST ที่มีความสมดุลและเข้าใจง่าย

โค้ดตัวอย่างการจำแนกรูปภาพ

```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# โหลดชุดข้อมูล MNIST
mnist = fetch_openml('mnist_784', version=1)
X = mnist.data
y = mnist.target.astype('int')

# แบ่งข้อมูลเป็นชุดฝึกและชุดทดสอบ
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# ปรับขนาดข้อมูล (Normalization)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# สร้างโมเดล KNN
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# ทำนายและประเมินผล
y_pred = knn.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

ผลลัพธ์

Accuracy: 94.58%

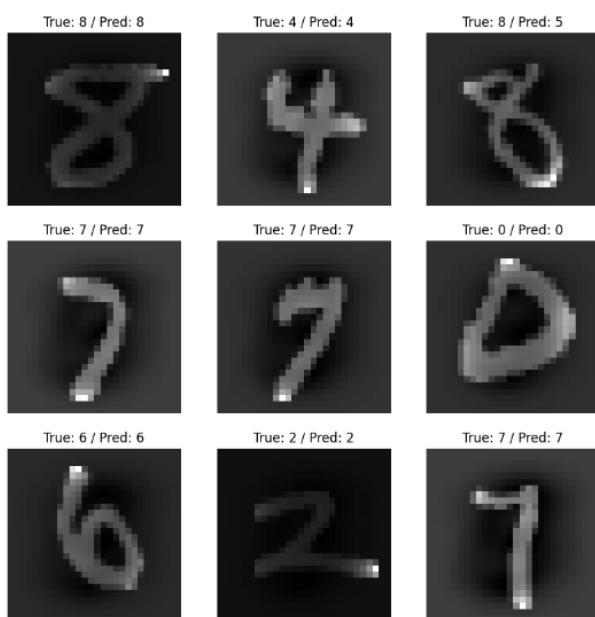
ໂດຍແສດງປະສິບອີກາພກຮ່ານຍໍດ້ວຍ Confusion Matrix

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

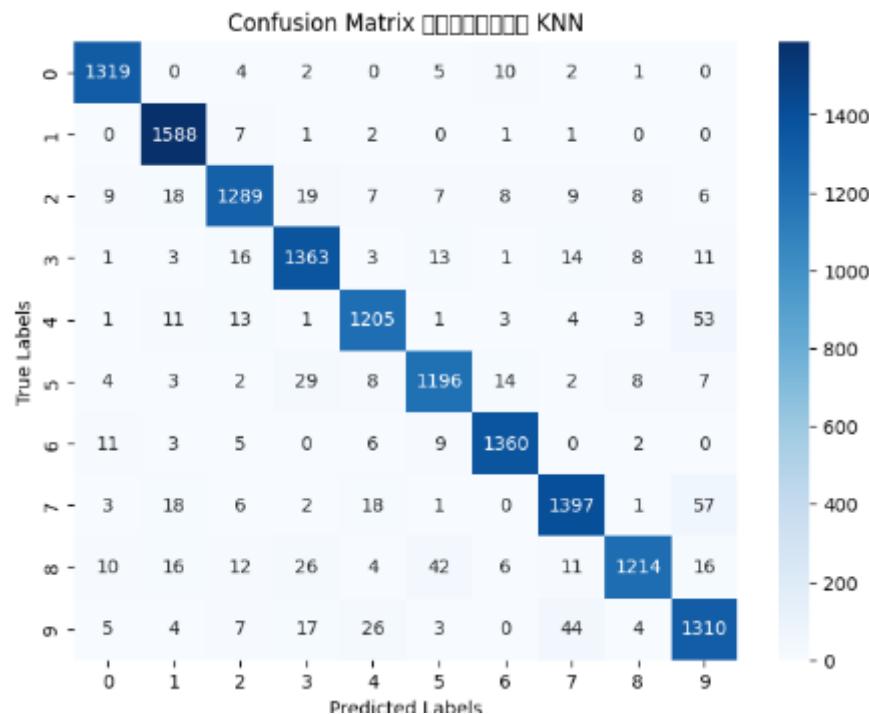
# ແສດງກາພຕ້ວອ່າງຈາກຊຸດທຄສອບ ພ້ອມຄໍາທີ່ກໍານາຍໄດ້
plt.figure(figsize=(10, 10))
for i in range(9):
    plt.subplot(3, 3, i + 1)
    plt.imshow(X_test[i].reshape(28, 28), cmap='gray')
    plt.title(f"True: {y_test.iloc[i]} / Pred: {y_pred[i]}")
    plt.axis('off')
plt.suptitle('ຕ້ວອ່າງກາພ MNIST ຈາກຊຸດທຄສອບ')
plt.show()

# ແສດງConfusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', xticklabels=range(10),
            yticklabels=range(10))
plt.xlabel('Predicted Labels')
```

ผลลัพธ์



ภาพนี้แสดงตัวอย่าง 9 ภาพจากชุดทดสอบ MNIST ซึ่งเป็นภาพตัวเลขเขียนด้วยมือที่ระบบนำมาทำนายด้วยโมเดล K-Nearest Neighbors (KNN) โดยในแต่ละภาพจะแสดงผลลัพธ์ของ ค่าจริง (True) และค่าที่โมเดลทำนายได้ (Pred) ตัวอย่างเช่น ภาพแรกแสดงเลข "8" ที่โมเดลทำนายถูกต้องว่าเป็น "8" ในขณะที่ภาพที่สามแสดงเลข "8" แต่โมเดลทำนายผิดเป็น "5" สะท้อนให้เห็นว่าโมเดล KNN สามารถจำแนกตัวเลขได้แม่นยำในหลายกรณี แต่ยังมีข้อผิดพลาดกับตัวเลขที่มีลักษณะใกล้เคียงกัน เช่น "8" และ "5"



ภาพนี้แสดง Confusion Matrix ของโมเดล KNN สำหรับการจำแนกตัวเลข MNIST โดยแบ่งแนวทแยงหลักแสดงจำนวนครั้งที่โมเดลทำนายถูกต้อง เช่น ตัวเลข "1" ถูกทำนายถูกถึง 1,588 ครั้ง ขณะที่ค่าที่อยู่นอกแนวทแยงแสดงจำนวนครั้งที่โมเดลทำนายผิด เช่น ตัวเลข "9" ถูกทำนายผิดเป็น "4" จำนวน 44 ครั้ง โมเดลมีความแม่นยำสูงในตัวเลขบางชุด เช่น "0", "1", "6" แต่ยังมีข้อผิดพลาดกับตัวเลขที่มีรูปทรงคล้ายกัน เช่น "5" กับ "8" หรือ "9" กับ "4" ภาพนี้ช่วยให้เห็นภาพรวมของความแม่นยำและข้อผิดพลาดของโมเดลในแต่ละคลาสได้อย่างชัดเจน

การเรียนรู้ Machine Learning ในบทนี้เปรียบเสมือนการเปิดประตูสู่โลกแห่งข้อมูล ที่เปลี่ยน "ข้อมูล ดิบ" ให้เป็น "องค์ความรู้" ที่ใช้แก้ปัญหาได้จริง เช่น การทำนายราคาบ้านหรือจำแนกภาพตัวเลข ด้วยโมเดล KNN และ Linear Regression ที่เป็นพื้นฐานของระบบ AI การนำ Scikit-learn มาใช้ในบทนี้ ช่วยให้เข้าใจขั้นตอนของการบวนการ Machine Learning ตั้งแต่การเตรียมข้อมูล การเลือกโมเดล ไปจนถึงการประเมินประสิทธิภาพ ซึ่งเป็นโครงสร้างสำคัญที่ใช้ในสายงาน Data Science หรือ AI Engineer

นอกจากนี้ยังเป็นทางให้ผู้เรียนเห็นศักยภาพของ Machine Learning ที่สามารถต่อยอดไปยังโมเดลขั้นสูง เช่น Deep Learning หรือ CNN เพื่อสร้างระบบที่มีความสามารถในการวิเคราะห์ข้อมูลเชิงลึก เช่น การรู้จำใบหน้า หรือการวิเคราะห์ภาพถ่ายทางการแพทย์ ML ไม่ใช่แค่เครื่องมือสำหรับนักวิทยาศาสตร์

ข้อมูล แต่ยังเป็นทักษะที่วิศวกรยุคใหม่ควรมี เพื่อเชื่อมโยงความรู้เชิงวิศวกรรมกับการประมวลผลข้อมูล ช่วยเพิ่มประสิทธิภาพและนวัตกรรมในงานวิศวกรรมและอุตสาหกรรม 4.0 ได้อย่างแท้จริง

7.5 งานวิจัยเพิ่มเติม

ในยุคปัจจุบัน การพัฒนาเทคโนโลยีและวิทยาศาสตร์ได้ก้าวหน้าไปมาก โดยเฉพาะในด้าน การเรียนรู้ของเครื่อง (Machine Learning) ซึ่งเป็นหนึ่งในเครื่องมือสำคัญที่มีบทบาทในหลากหลายสาขา การพัฒนาระบบหรือโปรแกรมที่ใช้ในการวิเคราะห์ข้อมูลต่าง ๆ อย่างมีประสิทธิภาพสามารถนำมาใช้ในการแก้ปัญหาหลายประการ ตั้งแต่การจำลองกระบวนการผลิต การวิเคราะห์ข้อมูลทางสถิติ ไปจนถึงการใช้เครื่องมือที่เกี่ยวข้องกับอินเทอร์เน็ตของสรรพสิ่ง (IoT) การประยุกต์ใช้งานของ Machine Learning จึงเป็นหัวข้อสำคัญในงานวิจัยต่าง ๆ ในบทความนี้จะพูดถึงงานวิจัยที่สำคัญในหลาย ๆ ด้านที่เกี่ยวข้องกับ Machine Learning ซึ่งช่วยพัฒนาและปรับปรุงกระบวนการต่าง ๆ ในอุตสาหกรรมและชีวิตประจำวันของเรา ความคิดและการประยุกต์ใช้งาน Machine Learning ในงานวิจัยต่าง ๆ

7.5.1. โปรแกรมจำลองกระบวนการผลิตคราฟต์เบียร์แบบโอลบราวน์

ศุภกร รักใหม่ (2019) ได้พัฒนาโปรแกรม BrewSimoPy โดยใช้ Python เพื่อจำลองกระบวนการผลิตคราฟต์เบียร์แบบโอลบราวน์ มีการใช้ Pygame สร้างอินเทอร์เฟซผู้ใช้ และ Matplotlib สำหรับแสดงกราฟข้อมูลเบียร์ในกระบวนการ เช่น ความเข้มข้นของมอลต์และค่าการคำนวนสีเบียร์ โปรแกรมนี้ยังรองรับการประมวลผลทางฟิสิกส์และเคมีของกระบวนการหมักเบียร์แบบกึ่งเรียลไทม์ ผ่านการคำนวนด้วย SciPy ช่วยให้ผู้ใช้งานเข้าใจกระบวนการผลิตเบียร์ทั้งเชิงทฤษฎีและปฏิบัติ โปรแกรมสามารถ export ผลการคำนวนออกเป็นไฟล์ PDF ได้ เหมาะสำหรับการใช้เป็นเครื่องมือการเรียนการสอนในรายวิชาเชิงวิศวกรรมอาหารหรือกระบวนการผลิตเครื่องดื่ม

7.5.2. การวิเคราะห์โครงข้อมูลแบบ 2 มิติจากข้อมูลกราฟิกแบบ DXF ในโปรแกรมฟรีแคน

จักรี ติยะวงศ์สุวรรณ และคณะ (2021) ได้นำเสนอการใช้ภาษา Python และไลบรารี ezdxf ร่วมกับ FreeCAD เพื่อวิเคราะห์แรงในโครงข้อมูลของมิติ โดยดึงข้อมูลจากไฟล์ DXF ที่สร้างในโปรแกรม CAD งานนี้พัฒนาโมดูล Python สำหรับวิเคราะห์โครงสร้าง โดยอาศัย NumPy ในการคำนวนเมทริกซ์ โมดูลนี้สามารถแสดงผลลัพธ์กราฟิกใน FreeCAD และส่งออกข้อมูลเป็น CSV เพื่อนำไปใช้ต่อในโปรแกรมสเปรดชีต ข้อดีคือช่วยลดขั้นตอนการนำเข้าข้อมูลและเพิ่มความสะดวกในการออกแบบโครงสร้าง งานนี้ทดสอบเบรียบเทียบผลลัพธ์กับโปรแกรม SUTStructor และ CCT-Truss พบร่วมผลลัพธ์มีความแม่นยำและตรงกันอย่างดี

7.5.3. โปรแกรม PyCovid สำหรับการวิเคราะห์ข้อมูล COVID-19

ศุภกร รักใหม่. (2021) ได้นำเสนอโปรแกรม PyCovid ถูกพัฒนาขึ้นโดยใช้ Python และมี GUI ที่สร้างด้วย Tkinter พร้อมแสดงผลกราฟข้อมูลด้วย Matplotlib และ Plotly โปรแกรมนี้ใช้เพื่อวิเคราะห์ข้อมูล COVID-19 แบบเรียลไทม์ สามารถดึงข้อมูลจากฐานข้อมูลสาธารณะและนำเสนอในรูปแบบกราฟและแผนที่ โลก ตัวอย่างกราฟที่แสดง เช่น จำนวนผู้ติดเชื้อรายวันและอัตราการฟื้นตัว โปรแกรมมีระบบเลือกประเทศ

และช่วงเวลาเพื่อวิเคราะห์ข้อมูลในระดับเบรียบเทียบ หน้าสำหรับงานด้าน Data Science และงานวิจัยเชิงระบบวิทยา โดยมีการสาธิตการใช้โปรแกรมเพื่อแสดงผลบน Web Browser ด้วย Flask

7.5.4. การศึกษาเปรียบเทียบอัลกอริทึม Nearest Neighbor และอัลกอริทึม Genetic ใน การแก้ปัญหา TSP

งานวิจัยนี้เปรียบเทียบประสิทธิภาพของ Nearest Neighbor Algorithm (NN) และ Genetic Algorithm (GA) ในการแก้ปัญหา TSP โดยใช้ข้อมูลตำแหน่งของเมือง 50 เมืองเพื่อทดสอบอัลกอริทึมทั้งสอง พบว่า GA สามารถหาคำตอบที่ดีกว่า NN ในกรณีที่มีจำนวนเมืองมากกว่า 20 เมือง เนื่องจาก NN มีแนวโน้ม จะติดกับดัก local optimum ขณะที่ GA สามารถค้นหาคำตอบใกล้เคียงกับคำตอบที่ดีที่สุดได้ดีกว่า งานวิจัย ยังได้แสดงการเปรียบเทียบเส้นทางที่ได้จากการแก้ปัญหา TSP ที่มีความซับซ้อนสูง ในอนาคต ผู้วิจัยตั้งใจจะทดลองกับ Particle Swarm Optimization (PSO) เพื่อเปรียบเทียบกับ GA เพิ่มเติม

7.5.5. การปรับปรุงเส้นทางการเดินรถโดยสารโดยใช้ Genetic Algorithm ร่วมกับเทคนิค Crossover กับแตกต่างกัน: กรณีศึกษาที่เมือง Konya ประเทศตุรกี

งานวิจัยนี้นำเสนอการใช้ Genetic Algorithm (GA) เพื่อหาทางเลือกการเดินทางที่เหมาะสมที่สุดใน Konya ประเทศตุรกี โดยเลือกใช้ crossover techniques ที่หลากหลาย เช่น PBX, OX1 และ OX2 เพื่อเปรียบเทียบประสิทธิภาพ ผลการทดลองพบว่า PBX ให้ผลลัพธ์ที่ดีที่สุด นอกจากนี้ยังมีการใช้ GA ร่วมกับเทคนิคอื่น ๆ เช่น ABC, SA, และ PSO เพื่อเปรียบเทียบกับ GA แบบ PBX+OX1 ซึ่งได้ผลลัพธ์ที่สุดในกรณีศึกษานี้ งานวิจัยนี้แสดงให้เห็นถึงความสำคัญของการเลือก crossover techniques ที่เหมาะสมในการแก้ปัญหา TSP แบบ real-world

7.5.6. ปัญหา TSP: การทบทวนอย่างครอบคลุมและการประยุกต์ใช้งานในเชิงคำนวณด้วยภาษา Python

งานวิจัยนี้ทบทวนแนวทางการแก้ปัญหา TSP ตั้งแต่อัลกอริทึมปัจจุบัน รวมถึงการนำเสนอการเขียนโปรแกรมแก้ปัญหาด้วย Brute Force Algorithm ในภาษา Python งานวิจัยได้กล่าวถึงข้อจำกัดของวิธี Brute Force ที่มีความซับซ้อนเชิงเวลาแบบแฟกทอเรียล พร้อมทั้งสาธิตการเขียนโปรแกรมสำหรับหัววิถีที่สั้นที่สุดของเมือง 10 แห่งในประเทศไทย ผลการทดลองแสดงภาพเส้นทางที่ดีที่สุดผ่านไลบรารี Matplotlib เพื่อความเข้าใจในเชิงกราฟิก

7.5.7. การพัฒนาต้นแบบชุดอุปกรณ์ฟาร์มเลี้ยงไก่แบบสมาร์ทด้วยอินเทอร์เน็ตของสรรพสิ่งร่วมกับแอปพลิเคชัน

วิจัยนี้พัฒนาชุดอุปกรณ์สำหรับฟาร์มเลี้ยงไก่ที่เชื่อมต่อ IoT และแอปบนมือถือ โดยใช้ NodeMCU ESP8266 ต่อกับเซ็นเซอร์ DHT11, มอเตอร์, รีเลย์, เซอร์โว และจอ LCD ภายในกล่องควบคุม ระบบใช้แอป Blynk เป็นตัวกลางให้ผู้ใช้งานเปิด/ปิดอุปกรณ์และให้อาหารไก่ผ่านสมาร์ทโฟน พร้อมบันทึกอุณหภูมิลงฐานข้อมูล ผลการทดลองพบว่าต้นแบบทำงานได้ดี ผู้ใช้สามารถควบคุมอุปกรณ์และให้อาหารไก่ผ่านแอปได้จริง และผลการประเมินโดยผู้เชี่ยวชาญ 5 คนอยู่ในระดับดีมาก (ค่าเฉลี่ยคะแนน 4.64, S.D. 0.39)

7.5.8. การพัฒนาต้นแบบเครื่องให้อาหารปลาอัตโนมัติโดยใช้แบนวิดอินเตอร์เน็ตในทุกสิ่ง

งานวิจัยนี้สร้างต้นแบบเครื่องให้อาหารปลาที่ควบคุมผ่าน IoT โดยใช้ NodeMCU (ESP8266) ควบคุมกลไกการให้อาหารและเซ็นเซอร์ระดับอาหาร พร้อมทั้งพัฒนาแอปพลิเคชันบนสมาร์ทโฟนเชื่อมต่อผ่าน Blynk server เป็นตัวกลางส่งข้อมูลระหว่างเครื่องให้อาหารกับแอป ผลการทดลองพบว่าการให้อาหารปลา 3 มื้อ มีความเที่ยงตรง ~90.6% และผู้ใช้มีความพึงพอใจต่อระบบในระดับมาก (เฉลี่ยคะแนน 4.06/5) แสดงถึงศักยภาพในการประยุกต์สูญญภัยต์เชิงพาณิชย์ต่อไป

7.5.9. ระบบไอโอทีสำหรับการตรวจสอบความชื้นและอุณหภูมิ เพื่อส่งเสริมการเพาะเลี้ยงเห็ดในโรงเรือนให้มีผลผลิตที่สมบูรณ์

งานวิจัยนี้ออกแบบระบบ IoT สำหรับควบคุมความชื้นและอุณหภูมิอัตโนมัติในโรงเรือนเพาะเห็ด เพื่อเพิ่มผลผลิตเห็ดให้สมบูรณ์ ระบบประกอบด้วยชุดควบคุมการให้น้ำ กระจายความร้อน ดูดอากาศ และสร้างความร้อนในโรงเรือน โดยเชื่อมต่อเซ็นเซอร์และตัวกระตุนต่าง ๆ ผ่านเครือข่าย IoT ผลการทดลองพบว่าการควบคุมสภาพโรงเรือนด้วย IoT ช่วยลดการปนเปื้อน (เชื้อราดำลดลง) และทำให้ได้ผลผลิตที่มีน้ำหนักและคุณภาพดีขึ้น อีกทั้งชุมชนผู้เพาะเห็ดมีความพึงพอใจต่อระบบในระดับสูงสุด

7.5.10. การวิเคราะห์ความเสี่ยงผิดนัดชำระบะหนี้บัตรเครดิตด้วย ML

งานวิจัยนี้ใช้ชุดข้อมูลธุรกรรมบัตรเครดิต 307,511 รายการ (ข้อมูลไม่สมดุลสูง) เพื่อสร้างโมเดลที่นำทางไปสู่การตัดสินใจ โดยเปรียบเทียบอัลกอริทึมหลายแบบ เช่น Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, XGBoost, SVM เป็นต้น ทั้งนี้ได้ใช้เทคนิค Oversampling/SMOTE เพื่อแก้ปัญหาข้อมูลชนกลุ่มน้อย ผลการทดลองชี้ว่าโมเดล XGBoost ให้ค่า Recall สูงสุดถึง 0.97 สำหรับการทำนายกลุ่มผิดนัดฯ แต่มี Accuracy เพียง ~0.37 และ F1 ~0.51 เนื่องจากเน้นจับกลุ่มเสี่ยงได้ครบถ้วน ขณะที่ KNN ให้ Recall ต่ำสุด 0.57 โดย Accuracy ~0.55 และ F1 ~0.46 ซึ่งต่ำที่สุดในบรรดาโมเดลที่ทดสอบทั้งหมด ถึงความจำเป็นของการเลือก metric ประเมินที่เหมาะสม (เช่น Recall) ในกรณีข้อมูลไม่สมดุล.

7.5.11. การตรวจจับสิ่งปฏิกูลของโครงสร้างเหล็กด้วยอัลกอริทึม KNN

งานวิจัยนี้ประยุกต์ Machine Learning เพื่อตรวจสอบชิ้นส่วนโครงเหล็กของโกดังสินค้าว่ามีสนิมหรือไม่ จากภาพถ่าย 133 รูปที่เก็บด้วยสมาร์ทโฟน แปลงข้อมูลภาพเป็นปริมาณพิกเซลสีแดง เขียว น้ำเงิน จากนั้นสร้างโมเดลจำแนกด้วย k-Nearest Neighbors เพื่อทำนายการมีสนิม ผลการทดลองพบว่าโมเดล KNN ที่พัฒนาขึ้นมีความถูกต้องประมาณ 70% ในการจำแนกภาพที่มีสนิมและไม่มีสนิมสามารถใช้เป็นเครื่องมือประเมินเบื้องต้นเพื่อวางแผนการซ่อมบำรุงโครงสร้างเหล็กได้ต่อไป

7.5.12. การเรียนรู้เครื่องจักรของเครื่อง

บทความนี้อธิบายถึงความสำคัญของ ชุดข้อมูล (Dataset) ในการเรียนรู้ของเครื่อง (Machine Learning) โดยยกตัวอย่างชุดข้อมูลมาตรฐานที่ใช้ปอยในการฝึกสอนโมเดล เช่น ชุดข้อมูล Iris ซึ่งใช้ในการจำแนกสายพันธุ์ดอกไอริสจากคุณสมบัติทางกายภาพ (เช่น ความยาวและความกว้างของกลีบเลี้ยง) และ ชุดข้อมูล MNIST ที่ใช้ในการจำแนกตัวเลขเขียนด้วยมือจากภาพดิจิทัล ข้อมูลสำคัญในชุดข้อมูลประกอบด้วย

Class Label หรือ Target ซึ่งเป็นค่าที่ไม่เดลทำนาย และ Attribute (Feature) ซึ่งเป็นคุณสมบัติต่าง ๆ ของข้อมูลที่ใช้ในการฝึกโมเดล

การนำ Machine Learning มาประยุกต์ใช้ในหลาย ๆ ด้านทำให้เห็นว่าการพัฒนาเทคโนโลยีในปัจจุบันสามารถนำไปใช้แก้ปัญหาต่าง ๆ ได้อย่างมีประสิทธิภาพ ตั้งแต่กระบวนการผลิตอาหาร การวิเคราะห์ข้อมูลในสาขาต่าง ๆ จนถึงการพัฒนาแอปพลิเคชันที่ช่วยให้ผู้ใช้งานสามารถควบคุมและติดตามผลได้ในเวลาจริง. นอกจากนี้ยังพบว่าเทคโนโลยีต่าง ๆ อย่าง Genetic Algorithm และ SMOTE มีบทบาทสำคัญในการช่วยปรับปรุงประสิทธิภาพของโมเดลในการทำนายหรือวิเคราะห์ข้อมูลที่มีความซับซ้อนหรือมีข้อมูลที่ไม่สมดุล

โดยรวมแล้ว งานวิจัยที่กล่าวถึงไม่เพียงแต่ช่วยพัฒนาความเข้าใจเกี่ยวกับการใช้งาน Machine Learning และ IoT แต่ยังสะท้อนให้เห็นถึงศักยภาพในการใช้เทคโนโลยีเหล่านี้ในการปรับปรุงคุณภาพชีวิตและการทำงานในหลาย ๆ สาขา. การเรียนรู้จากข้อมูลที่มีอยู่ในมือและการพัฒนาเครื่องมือใหม่ ๆ ที่สามารถแก้ปัญหาในโลกจริง เป็นสิ่งที่สามารถทำให้เกิดความก้าวหน้าทางเทคโนโลยีอย่างยั่งยืนในอนาคต

บทสรุป

บทที่ 7 นำเสนอการประยุกต์ใช้ความรู้ด้านการเขียนโปรแกรม Python เพื่อแก้ไขปัญหาหลากหลายรูปแบบในโลกแห่งความเป็นจริง โดยเฉพาะอย่างยิ่งในงานด้านวิศวกรรมและเทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง (IoT) ซึ่งเป็นแนวโน้มที่กำลังได้รับความนิยมอย่างมากในปัจจุบัน

เนื้อหาเริ่มต้นด้วยการเขียนโปรแกรมเพื่อคำนวณและจำลองสถานการณ์ในงานวิศวกรรม เช่น การสร้างภาพเคลื่อนไหวจำลองแรงโน้มถ่วงและการชนกันของลูกบอล โดยใช้ไลบรารี Pygame และการวิเคราะห์ข้อมูลด้วย Matplotlib ซึ่งช่วยให้เข้าใจพฤติกรรมทางฟิสิกส์ได้ดียิ่งขึ้น จากนั้นจึงขยายไปยังการแก้ปัญหาด้านโลจิสติกส์ผ่านการคำนวณหาเส้นทางที่สั้นที่สุด (TSP) ซึ่งเป็นประโยชน์อย่างมากในงานขนส่งและการจัดการเส้นทางเชิงพาณิชย์

ด้วยการเขียนโปรแกรมควบคุมระบบ IoT ที่ประกอบด้วยเซ็นเซอร์ (Sensor), ตัวขับเคลื่อน (Actuator) และการเชื่อมต่อ (Connectivity) พร้อมตัวอย่างการใช้งาน MicroPython กับ ESP32 เพื่อควบคุมอุปกรณ์ผ่าน Wi-Fi ซึ่งแสดงให้เห็นถึงการผสานโลกดิจิทัลเข้ากับอุปกรณ์ในโลกจริงอย่างเต็มรูปแบบ

การประยุกต์ Machine Learning ในงานวิศวกรรม เช่น การใช้ Scikit-learn เพื่อสร้างโมเดลทำนายราคาบ้านหรือจำแนกภาพตัวเลข MNIST ด้วย K-Nearest Neighbors (KNN) เป็นตัวอย่างการทำ AI มาช่วยในการวิเคราะห์ข้อมูลและสร้างระบบที่มีความแม่นยำในการทำนาย

การนำ Machine Learning และ IoT มาใช้งานในหลาย ๆ ด้านไม่เพียงแต่เพิ่มประสิทธิภาพการทำงานในสาขาต่าง ๆ แต่ยังช่วยให้เกิดการพัฒนาวัตกรรมที่สามารถแก้ปัญหาที่เกิดขึ้นในโลกแห่งความเป็นจริง

- การเขียนโปรแกรมทางวิศวกรรมจะเน้น การคำนวณและวิเคราะห์ปัญหาทางวิศวกรรม พร้อมการแสดงผลกราฟ

- Pygame ในการจำลองสถานการณ์สองมิติ
- Matplotlib ในการวิเคราะห์และแสดงผลกราฟ
- ปัญหาเส้นทางที่สั้นที่สุด (TSP) คือ การวิเคราะห์เส้นทางที่ครอบคลุมทุกจุดโดยใช้ระยะทางน้อยที่สุด
- การเตรียมข้อมูลของโปรแกรมอาจจะต้องอินพุตจากไฟล์ เช่น TSP ใช้พิกัดทางภูมิศาสตร์ (Latitude, Longitude) จากไฟล์ CSV
- การคำนวณระยะทางระหว่างจุด ใช้สูตร Haversine ในการคำนวณระยะทางบนพื้นโลก
- การสร้างเมทริกซ์ระยะทาง คือ จัดเก็บระยะทางระหว่างทุกจุดในรูปแบบเมทริกซ์
- วิธีการหาเส้นทางที่เหมาะสมที่สุด เช่น Nearest Neighbor และ Heuristic Algorithm เพื่อแก้ปัญหา TSP
- IoT (Internet of Things) การสื่อสารระหว่างอุปกรณ์ผ่านอินเทอร์เน็ตเพื่อทำงานร่วมกันอย่างอัตโนมัติ
- องค์ประกอบของ IoT ได้แก่ Sensor Actuator และ Connectivity คือองค์ประกอบหลักของระบบ IoT
- Sensor ในระบบ IoT อุปกรณ์สำหรับตรวจจับค่าทางกายภาพ เช่น อุณหภูมิ ความชื้น แสง และแก๊ส
- Actuator ในระบบ IoT อุปกรณ์ที่แปลงคำสั่งดิจิทัลเป็นการกระทำ เช่น การเปิดปิดไฟ หรือการหมุนมอเตอร์
- Connectivity ใน IoT การเชื่อมต่อระหว่างอุปกรณ์ เช่น Wi-Fi, Bluetooth, หรือ LoRa
- MicroPython และ ESP32 การเขียนโปรแกรม IoT ด้วย MicroPython บน ESP32 เพื่อการพัฒนาที่ง่ายและมีประสิทธิภาพ
- คุณสมบัติของ ESP32 มี Wi-Fi, Bluetooth, และ GPIO หลายpin เหมาะสำหรับการเชื่อมต่อกับเซ็นเซอร์
- การประยุกต์ใช้ IoT เช่น ระบบบ้านอัจฉริยะ การเกษตรอัจฉริยะ และระบบอุตสาหกรรม
- การสร้าง Web Server บน ESP32 ใช้ MicroPython เพื่อควบคุมอุปกรณ์ผ่านเครือข่าย Wi-Fi
- ความสามารถของ IoT ในปัจจุบัน IoT ช่วยเพิ่มประสิทธิภาพในการเชื่อมต่อและควบคุมอุปกรณ์ในงานวิศวกรรมและชีวิตประจำวัน

คำถามก้ายบก

1. ระบบ IoT (Internet of Things) มีองค์ประกอบหลักอะไรบ้าง?
2. เซ็นเซอร์ DHT11 มีหน้าที่และการใช้งานในระบบ IoT อย่างไร?
3. อธิบายขั้นตอนการตั้งค่า Wi-Fi บน ESP32 ด้วย MicroPython
4. แอคชูเอเตอร์ (Actuator) มีบทบาทสำคัญอย่างไรในระบบ IoT?

5. อธิบายการใช้ Pygame ในการจำลองการเคลื่อนที่ของลูกบอล
6. Matplotlib ใช้แสดงผลข้อมูลในรูปแบบกราฟอย่างไร
7. สูตร Haversine ใช้สำหรับคำนวณอะไร? และใช้ในสถานการณ์ใด?
8. การแก้ปัญหาเส้นทางที่สั้นที่สุด (TSP) มีความสำคัญในงานวิศวกรรมด้านใดบ้าง?
9. ESP32 มีคุณสมบัติใดที่เหมาะสมสำหรับการพัฒนา IoT?
10. Connectivity ในระบบ IoT ช่วยให้การส่งข้อมูลระหว่างเซ็นเซอร์และ Actuator มีความสะดวกอย่างไร?
11. อธิบายการทำงานของระบบบ้านอัจฉริยะโดยใช้เซ็นเซอร์และแอกชูเอเตอร์
12. วัดกราฟเส้นทางจากข้อมูล TSP โดยใช้ Matplotlib ต้องเริ่มต้นด้วยคำสั่งใด?
13. เขียนตัวอย่างโค้ดเพื่อเปิดไฟ LED โดยใช้ Web Server บน ESP32.
14. เซ็นเซอร์ LDR ใช้ตรวจสอบอะไร และมีการนำไปใช้ในระบบ IoT อย่างไร?
15. MicroPython แตกต่างจาก Python อย่างไร และทำไมจึงเหมาะสมกับการพัฒนา IoT?
16. อธิบายขั้นตอนการใช้ Matplotlib เพื่อแสดงผลข้อมูลการเคลื่อนที่ของลูกบอลใน Pygame
17. ระบบ IoT ในโรงงานอุตสาหกรรมสามารถเพิ่มประสิทธิภาพการผลิตได้อย่างไร?
18. การใช้ Pygame จำลองการชนของลูกบอลต้องกำหนดค่าคงที่ใดบ้าง?
19. MQTT Protocol มีประโยชน์อย่างไรในระบบ IoT?
20. การประยุกต์เขียนโปรแกรมในงานวิศวกรรมช่วยเพิ่มประสิทธิภาพในงานด้านต่าง ๆ ได้อย่างไร?

โจทย์การเขียนโปรแกรม

1. เขียนโปรแกรมโดยใช้ Pygame เพื่อจำลองการเคลื่อนที่ของลูกบอลในกรอบหน้าจอ ขณะที่ลูกบอลถูกแรงโน้มถ่วงดึงลงและเด้งกลับเมื่อชนกับขอบจอ แสดงเส้นทางการเคลื่อนที่ของลูกบอลในกราฟด้วย Matplotlib โดยมีเงื่อนไขดังนี้
 - ลูกบอลเริ่มต้นที่ตำแหน่งกลางหน้าจอ ($x=400, y=100$) และเคลื่อนที่ลง
 - ใช้แรงโน้มถ่วง (gravity) = 0.1
 - เมื่อลูกบอลชนขอบ ให้ลดพลังงานลง 10%
 - จำลองการเคลื่อนที่ 600 เฟรม

ผลลัพธ์ หน้าต่างแสดงการเคลื่อนที่ของลูกบอลแบบเรียลไทม์ และกราฟแสดงเส้นทางการเคลื่อนที่ (Trajectory) ของลูกบอล
2. เขียนโปรแกรมคำนวณระยะทางระหว่าง 3 อำเภอในจังหวัดกาฬสินธุ์ โดยใช้สูตร Haversine จากพิกัด Latitude และ Longitude
 - Mueang Kalasin: Latitude 16.4336, Longitude 103.5078
 - Na Mon: Latitude 16.2511, Longitude 103.8193

- Kuchinarai: Latitude 16.5127, Longitude 104.0536

ผลลัพธ์

ระยะทางระหว่าง Mueang Kalasin และ Na Mon: 38.94 กม.

ระยะทางระหว่าง Na Mon และ Kuchinarai: 38.35 กม.

ระยะทางระหว่าง Mueang Kalasin และ Kuchinarai: 58.86 กม.

3. เขียนโปรแกรม MicroPython สำหรับ ESP32 เพื่อสร้าง Web Server ที่ควบคุมการเปิด-ปิด LED บน GPIO 2 ผ่านเครือข่าย Wi-Fi โดย

- ให้ผู้ใช้กดปุ่มในหน้าเว็บเพื่อเปิดหรือปิด LED
- แสดงสถานะปัจจุบันของ LED บนหน้าเว็บ

ESP32 LED Control

LED is currently: OFF

[Turn ON] [Turn OFF]

4. เขียนโปรแกรมเพื่ocommunicate เส้นทางที่สั้นที่สุดสำหรับอำเภอในจังหวัดกาฬสินธุ์ โดยใช้วิธี Nearest Neighbor พร้อมแสดงเส้นทางและระยะทางรวม ข้อมูล: ใช้ตารางระยะทางระหว่างอำเภอที่กำหนดไว้ในหัวข้อ 7.2.2

- เริ่มต้นที่อำเภอ Mueang Kalasin
- เลือกเส้นทางที่ใกล้ที่สุดไปยังจุดถัดไป
- วนกลับมาที่จุดเริ่มต้นเมื่อยืนครบทุกจุด

เส้นทางที่สั้นที่สุด: Mueang Kalasin -> Na Mon -> Kamalasai -> Yang Talat -> ... -> Mueang Kalasin

ระยะทางรวม: 338.72 กม.

5. เขียนโปรแกรม MicroPython เพื่ออ่านค่าอุณหภูมิและความชื้นจากเซ็นเซอร์ DHT11 และแสดงผลในรูปแบบเรียลไทม์ทุก ๆ 2 วินาที

Temperature: 25°C, Humidity: 60%

Temperature: 24°C, Humidity: 58%

Temperature: 26°C, Humidity: 61%

6. เขียนโปรแกรมใช้ Matplotlib เพื่อวัดวงจรอิเล็กทรอนิกส์ที่ประกอบด้วยแบตเตอรี่ ตัวต้านทาน (Resistor) และหลอดไฟ (LED)

- แสดงแบตเตอรี่เป็นสัญลักษณ์เส้นแนวนอน
- ตัวต้านทานเป็นสัญลักษณ์ซิกแซก
- หลอดไฟเป็นวงกลม

- ใส่ข้อความกำกับแต่ละส่วน

7. เขียนโปรแกรมบน ESP32 เพื่ออ่านค่าความชื้นจากเซ็นเซอร์วัดความชื้นในดิน (Soil Moisture Sensor) และส่งข้อมูลไปยัง Web Server แสดงค่าความชื้นบน Web Server และ หากค่าความชื้นต่ำกว่า 30% ให้แสดงข้อความ "ดินแห้ง"

ความชื้นในดิน: 28%

สถานะ: ดินแห้ง

8. เขียนโปรแกรม ESP32 เพื่อกีบค่าจากเซ็นเซอร์ DHT11 และส่งข้อมูลไปยังคอมพิวเตอร์ จากนั้นใช้ Matplotlib บนคอมพิวเตอร์แสดงกราฟอุณหภูมิและความชื้นแบบเรียลไทม์

กราฟแสดงข้อมูลอุณหภูมิและความชื้นที่เปลี่ยนแปลงเมื่อเวลาผ่านไป

9. เขียนโปรแกรม MicroPython เพื่อสร้างนาฬิกาดิจิทัลที่แสดงเวลาและอุณหภูมิบนหน้าจอ OLED

เวลา: 12:00

อุณหภูมิ: 25°C

10. เขียนโปรแกรม ESP32 เพื่อส่งค่าความสว่างจาก LDR Sensor ไปยัง MQTT Broker และแสดงผลใน Web Dashboard

Published brightness: 80%

บรรณานุกรม

- เครือวัลย์ เนตรพนา, & ศิริสรรพ เหล่าแหหเกียรติ. (2566). การวิเคราะห์ความเสี่ยงในการผิดนัดชำระของลูกหนี้บัตรเครดิตโดยการใช้อัลกอริธึมการเรียนรู้ของเครื่อง. การประชุมวิชาการด้านวิทยาศาสตร์ ข้อมูล ครั้งที่ 3, 50-60. <https://www.kaggle.com/datasets/mishra5001/credit-card?resource=download>
- จักรี ติยะวงศ์สุวรรณ, จิรัชติ บรรจงศิริ, & สมภพ ลิ้มประไฟฟงษ์. (2021). การวิเคราะห์โครงข้อมูลน แบบ 2 มิติจากข้อมูลกราฟิกแบบ DXF ในโปรแกรมฟรีแอด. SAU Journal of Science & Technology, 7(2), 43-55. <https://saujournalst.sau.ac.th/>
- ณิศรา ศรีแก้ว, กอปร ศรีนิวน, & วุฒิพงษ์ กุศลคุ้ม. (2566). การประยุกต์ใช้การเรียนรู้ของเครื่องเพื่อตรวจจับภาพขึ้นส่วนที่มีสินิของโครงสร้างเหล็กอาคารโกดังสินค้า. การประชุมวิชาการวิศวกรรมโยธาแห่งชาติ ครั้งที่ 28, 24–26 พฤษภาคม 2566, ภูเก็ต. <https://www.kku.ac.th>
- พัชพล จันทาทอง, สุวิมล บรรคกิจุลย์ชัย, ไพศาล สิมาเลาเต่า, & อุบลรัตน์ ศิริสุขโภคฯ. (2566). การพัฒนาต้นแบบชุดอุปกรณ์ฟาร์มเลี้ยงไก่แบบสามารถตัวอยู่อินเทอร์เน็ตของสறฐ์ร่วมกับแอปพลิเคชัน. การประชุมวิชาการระดับชาติ ครั้งที่ 15 มหาวิทยาลัยราชภัฏนครปฐม. <https://www.npru.ac.th>
- ยุติธรรม ประมะ, & วงศ์ โชค. (2564). การพัฒนาต้นแบบเครื่องให้อาหารปลาอัตโนมัติโดยใช้แนวคิดอินเทอร์เน็ตในทุกสิ่ง. วารสารวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏกำแพงเพชร, 13(18), 75–76. <https://www.kpru.ac.th>
- ธรัช อารีราชภูร, & วรปภา อารีราชภูร. (2563). ระบบไอโอทีสำหรับการตรวจสอบความชื้นและอุณหภูมิ เพื่อส่งเสริมการเพาะเลี้ยงเห็ดในโรงเรือนให้มีผลผลิตที่สมบูรณ์. วารสารการประยุกต์ใช้เทคโนโลยีสารสนเทศ, 6(1), 8–10. <https://www.rmu.ac.th>
- ศุภกร รักใหม่. (2019). ไฟرون โครงสร้างไมโลกุล และคราฟท์เบียร์. Thai Journal of Physics, 36(3), 79-86. <https://ph01.tci-thaijo.org/index.php/SciTech/article/view/216335>
- ศุภกร รักใหม่. (2021). PyCovid: โปรแกรมสำหรับแสดงผลและวิเคราะห์ข้อมูลการแพร่ระบาดของเชื้อ COVID-19 . Thai Journal of Physics, 38 (1) , 13 - 21 . <https://ph01.tci-thaijo.org/index.php/tjp/article/view/238402>
- Adafruit. (n.d.). ESP32 - S2 Feather Development Board. Retrieved from <https://www.adafruit.com/product/4769>
- Bhattarai, S. (2024). Traveling Salesman Problem: A Comprehensive Review and Computational Implementation in Python. Scientific Researches in Academia, 2(2), 87-94. <https://doi.org/10.3126/sra.v2i2.74286>

- DataFlair. "Python Applications: Amazing Real-World Uses of Python Programming." [Online]. Available: <https://data-flair.training/blogs/python-applications/>
- Desuvit. "Innovative Technologies That Have Taken Over Software Development." [Online]. Available: <https://www.desuvit.com/innovative-technologies-that-have-taken-over-software-development/>
- Dogan, S., & Celik, E. (2023). Trip Route Optimization based on Bus Transit Using Genetic Algorithm with Different Crossover Techniques: A Case Study in Konya/Türkiye. International Journal of Engineering and Geosciences, 8 (3) , 1 5 6 - 1 6 3 . <https://doi.org/10.26833/ijeg.123456>
- Espressif Systems. (n.d.). ESP32 Overview. Retrieved from <https://www.espressif.com/en/products/socs/esp32/overview>
- Google LLC. (n.d.). Welcome to Google Colab. Retrieved January 12, 2025, from <https://colab.research.google.com>.
- IEEE. (2015). Internet of Things (IoT): Overview and Applications. IEEE Internet of Things Journal, 2(1), 1-16.
- KongRuksiam Studio. (2020, January 11). สรุป Machine Learning (EP.1) – การเรียนรู้ของเครื่อง Medium. URL: <https://kongruksiam.medium.com/สรุป-machine-learning-ep-1-71d2a7f3167b829406>
- Kumar, R., & Bansal, S. (2017). A comparative study of Nearest Neighbor Algorithm and Genetic Algorithm in solving Travelling Salesman Problem. International Journal of Advanced Research in Computer Science, 8 (5) , 1 2 3 - 1 2 9 . <https://doi.org/10.26833/ijeg.654321>
- LoRa Alliance. (n.d.). What is LoRaWAN?. Retrieved from <https://lora-alliance.org/>
- MicroPython. (n.d.). Getting Started with MicroPython on the ESP32. Retrieved from <https://micropython.org/>
- MQTT.org. (n.d.). MQTT: The Standard for IoT Messaging. Retrieved from <https://mqtt.org/>
- PyCom. (n.d.). Pycom IoT Development Boards. Retrieved from <https://pycom.io/>
- Python Software Foundation. (n.d.). Welcome to Python.org. Retrieved January 12, 2025, from <https://www.python.org>.

- Raspberry Pi Foundation. (n.d.). Getting Started with Raspberry Pi and IoT. Retrieved from <https://www.raspberrypi.org/>
- Sande, W., & Sande, C. (2020). Hello World! Computer Programming for Kids and Other Beginners. Third Edition. Manning Publications Co.
- Sofworld. "History of Computers." [Online]. Available: <http://www.sofworld.org/>
- SparkFun. (n.d.). IoT Starter Kit for ESP32. Retrieved from <https://www.sparkfun.com/>
- Sweigart, A. (2015). Automate the Boring Stuff with Python: Practical Programming for Total Beginners. No Starch Press.
- Sweigart, A. (2019). Beyond the Basic Stuff with Python: Best Practices for Writing Clean Code. No Starch Press.
- ThingSpeak. (n.d.). IoT Analytics Platform. Retrieved from <https://thingspeak.com/>
- TIOBE. "TIOBE Programming Community Index." [Online]. Available: <https://www.tiobe.com/tiobe-index/>
- Vaughan, L. (2019). Impractical Python Projects: Playful Programming Activities to Make You Smarter. No Starch Press.