แผนบริหารการสอนประจำบทที่ 3

คำสั่งควบคุมทิศทางการทำงานของโปรแกรม

หัวข้อประจำบท

1. ทิศทางการทำงานแบบเรียงลำดับ
2. คำสั่งควบคุมทิศทางแบบทางเลือก
3. คำสั่งควบคุมทิศทางแบบวนซ้ำ

วัตถุประสงค์เชิงพฤติกรรม

1. เพื่อให้ผู้เรียนสามารถอธิบาย โครงสร้างการทำงานของโปรแกรม
2. เพื่อให้ผู้เรียนสามารถอธิบายเกี่ยวกับการทำงานของโปรแกรมแบบเรียงลำดับ แบบทางเลือก และแบบวนซ้ำ
3. เพื่อให้ผู้เรียนสามารถอธิบายคำสั่งควบคุมทิศทางแบบทางเลือก และคำสั่งควบคุมทิศทางแบบวนซ้ำ
4. เพื่อให้ผู้เรียนสามารถอธิบายใช้คำสั่งควบคุมทิศทางแบบทางเลือก และคำสั่งควบคุมทิศทางแบบวนซ้ำได้อย่างถูกต้อง
5. เพื่อให้ผู้เรียนสามารถเขียนโปรแกรมโดยเลือกใช้คำสั่งควบคุมทิศทางแบบทางเลือก และคำสั่งควบคุมทิศทางแบบวนซ้ำ แก้โจทย์ปัญหาได้อย่างถูกต้อง

วิธีการสอนและกิจกรรม

1. ผู้สอนบรรยายในชั้นเรียน และโปรแกรมนำเสนอ ตามหัวข้อเนื้อหาในเอกสารประกอบการสอน
2. ผู้เรียนทำแบบฝึกหัดท้ายบท และฝึกเขียนโปรแกรมด้วยคอมพิวเตอร์

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน วิชา หลักการเขียนโปรแกรมคอมพิวเตอร์
2. โปรแกรม Python รุ่น 3.8.10
3. โปรแกรม PowerPoint

การวัดผลและการประเมินผล

1. สังเกตจากการอภิปราย การตอบคำถาม และซักถามระหว่างเรียน
2. การปฏิบัติบนเครื่องคอมพิวเตอร์
3. การทำแบบฝึกหัดท้ายบท

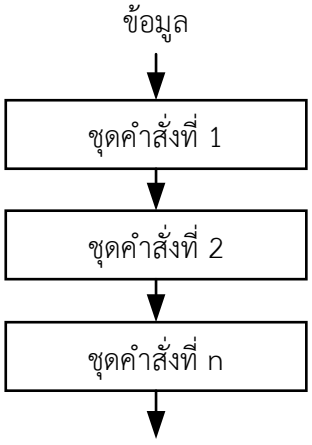
บทที่ 3
คำสั่งควบคุมทิศทางการทำงานของโปรแกรม

การทำงานของโปรแกรมคอมพิวเตอร์ จะมีทิศทางการทำงานของโปรแกรมขึ้นอยู่กับการออกแบบกระบวนการทำงานของโปรแกรม เมื่อดำเนินการเขียนโปรแกรม จะใช้คำสั่งควบคุมการทำงานให้โปรแกรมสามารถทำงานได้ตามทิศทางที่ต้องการ (Willis and Newsome, 2011) คำสั่งควบคุมทิศทางการทำงานเป็นมาตรฐานของทุกภาษาคอมพิวเตอร์ โดยแต่ละภาษาคอมพิวเตอร์มีคำสั่งควบคุมทิศทางการทำงานโดยใช้หลักการเดียวกัน แตกต่างที่โครงสร้างและรูปแบบของคำสั่งที่นำมาใช้ ขึ้นอยู่กับแต่ละภาษาคอมพิวเตอร์ที่ได้กำหนดกฎเกณฑ์ในการใช้คำสั่ง

คำสั่งควบคุมทิศทาง (Control Flow Statements) หมายถึง คำสั่งหรือกฎเกณฑ์ ที่ใช้สำหรับควบคุมทิศทางการทำงาน เพื่อให้บรรลุเป้าหมาย (สุชาติ คุ้มมณี, 2558) และเป็นการสั่งงานโดยชุดคำสั่งที่จะถูกประมวลผลด้วยโปรแกรม (Dierbach, 2013) ทิศทางการทำงานของโปรแกรมประกอบไปด้วยทิศทางการทำงาน 3 ประเภท ได้แก่ ทิศทางแบบเรียงลำดับ ทิศทางแบบทางเลือก และทิศทางแบบทำซ้ำ โดยมีรายละเอียดดังต่อไปนี้

ทิศทางการทำงานแบบเรียงลำดับ

ทิศทางการทำงานแบบเรียงลำดับ (Sequence Statements) คือ ลักษณะของการทำงานของโปรแกรมที่มีการทำงานทำงานทุกขั้นตอน โดยทิศทางการทำงาน เรียงลำดับการทำงานตั้งแต่ต้นจนจบของการทำงาน ไม่มีการข้ามกระโดดหรือแบ่งเส้นทางการทำงานของโปรแกรม (Halterman, 2016) มีรายละเอียด ดังภาพที่ 3.1



ภาพที่ 3.1 โครงสร้างการทำงานแบบเรียงลำดับ

ทิศทางการทำงานแบบเรียงลำดับนี้ เป็นโครงสร้างที่ทำงานทุกขั้นตอน ดังนั้นจะไม่มีคำสั่งที่จะควบคุมทิศทางการทำงานยกตัวอย่างโปรแกรมดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 3.1 ทิศทางการทำงานแบบเรียงลำดับ

บรรทัดที่	คำสั่ง
1	salary = float(input("เงินเดือน : "))
2	saving_money = salary*5/100
3	money = salary-saving_money
4	print("เงินที่ได้รับจริง : ",money)
ผลลัพธ์	
เงินเดือน : 30000 (รับข้อมูลจากแผงแป้นอักขระ) เงินที่ได้รับจริง : 28500.0	

โปรแกรมตัวอย่างที่ 3.1 คำนวณเงินรับที่ได้ซึ่งได้จากเงินเดือนหักประกันสังคม 5 เปอร์เซ็นต์ รวมกับเงินล่วงเวลา ซึ่งการทำงานเริ่มจากข้อมูลนำเข้าเงินเดือน คำนวณเงินหักประกันสังคม 5 เปอร์เซ็นต์ คำนวณเงินที่ได้รับจากเงินเดือนลบเงินประกันสังคมและนำข้อมูลออกแสดงผลลัพธ์ของเงินที่ได้รับ ซึ่งทุกครั้งที่มีการรับเงินเดือนจะต้องผ่านกระบวนการต่อมาเหมือนกันหมดทุกครั้ง การทำงานลักษณะนี้เป็นโปรแกรมที่มีทิศทางการทำงานแบบเรียงลำดับ

คำสั่งควบคุมทิศทางแบบทางเลือก

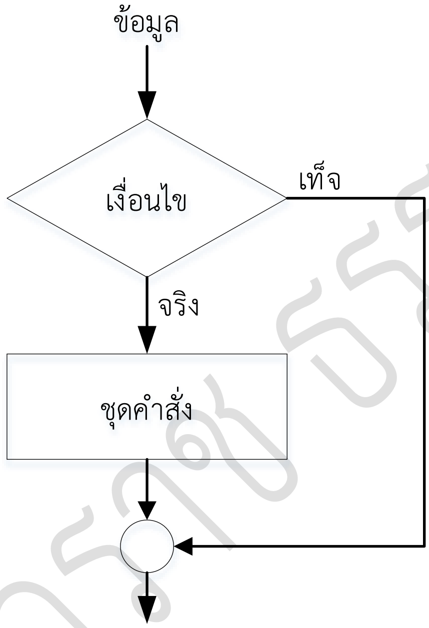
คำสั่งควบคุมทิศทางแบบทางเลือก (Conditional Statement) หมายถึง โครงสร้างของโปรแกรมที่มีการตรวจสอบข้อมูลกับเงื่อนไขที่กำหนด ซึ่งเงื่อนไขนั้นมีตั้งแต่ 1 เงื่อนไขหรือมากกว่า 1 เงื่อนไขขึ้นไป ซึ่งเมื่อมีการตรวจสอบ ผลลัพธ์ที่ได้ตรงกับเงื่อนไขใด โปรแกรมก็จะทำงานไปในทิศทางของทางเลือกเงื่อนไขนั้น (Fior, 2016) คำสั่งควบคุมทิศทางแบบทางเลือกในภาษาไพธอนนั้น มีคำสั่งทิศทางการทำงาน 3 คำสั่ง ประกอบด้วยคำสั่ง if if...else และ if...elif...else ซึ่งแต่ละคำสั่งมีรายละเอียดดังนี้

1. คำสั่ง if

คำสั่ง if คือ คำสั่งโครงสร้างทางเลือกแบบ 1 ทางเลือก การทำงานของโปรแกรมจะประมวลผลเพียงชุดคำสั่งเดียว โครงสร้างการใช้คำสั่งมีรูปแบบดังต่อไปนี้

```
If (เงื่อนไข) :
        ชุดคำสั่ง
```

จากโครงสร้างรูปแบบการใช้คำสั่ง if มีลักษณะการทำงาน รายละเอียดดังผังโปรแกรมต่อไปนี้



ภาพที่ 3.2 การทำงานของคำสั่ง if

ภาพที่ 3.2 โครงสร้างคำสั่งแบบ if มีหลักการทำงานคือ เมื่อมีการรับข้อมูลเข้ามา ข้อมูลนั้นจะนำเข้าไปทำการตรวจสอบในเงื่อนไข ถ้าข้อมูลเข้าเงื่อนไขที่เป็นจริง จะทำการประมวลชุดคำสั่ง แต่ถ้าไม่เข้าเงื่อนไขหรือเงื่อนไขเป็นเท็จ จะไม่มีการทำคำสั่งเกิดขึ้น คำสั่ง if มีรูปแบบและลักษณะการทำงานดังโปรแกรมตัวอย่างที่ 3.2

ตัวอย่างที่ 3.2 การใช้คำสั่ง if

บรรทัดที่	คำสั่ง
1	point = float(input("คะแนนเก็บ = "))
2	final = float(input("คะแนนปลายภาค = "))
3	total=point+final

4	print ("คุณได้คะแนน ",total)
5	if total>=50 :
6	print ("สอบผ่านวิชาหลักการเขียนโปรแกรม")

ผลลัพธ์ที่ 1	
คะแนนเก็บ = 33	(รับข้อมูลจากแผงแป้นอักขระ)
คะแนนปลายภาค = 20	(รับข้อมูลจากแผงแป้นอักขระ)
คุณได้คะแนน 53.0	
สอบผ่านวิชาหลักการเขียนโปรแกรม	
ผลลัพธ์ที่ 2	
คะแนนเก็บ = 35	(รับข้อมูลจากแผงแป้นอักขระ)
คะแนนปลายภาค = 13	(รับข้อมูลจากแผงแป้นอักขระ)
คุณได้คะแนน 48.0	

โปรแกรมตัวอย่างที่ 3.2 โปรแกรมผลการเรียนของนักศึกษา โดยให้มีการรับนำข้อมูลเข้ามาจากแผงแป้นอักขระและเก็บไว้ในตัวแปร point และ final จากนั้นนำข้อมูลในตัวแปรทั้งสองคำนวณหาผลรวมคะแนนที่ได้เก็บไว้ที่ตัวแปร total ทำการตรวจสอบกับเงื่อนไข ถ้าข้อมูลคะแนนรวมมากกว่าหรือเท่ากับ 50 ให้แสดงข้อความ “สอบผ่านวิชาหลักการเขียนโปรแกรม” ดังผลลัพธ์ที่ 1 และถ้าข้อมูลคะแนนรวมน้อยกว่า 50 โปรแกรมจะไม่แสดง ดังผลลัพธ์ที่ 2

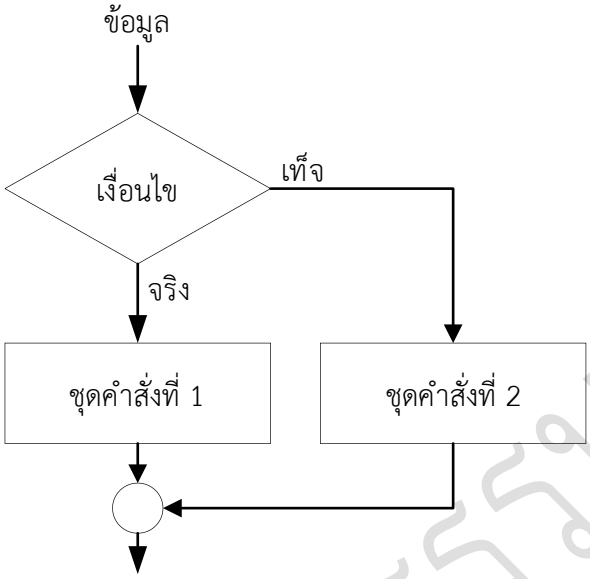
ดังนั้นสามารถสรุปได้ว่า การใช้คำสั่ง if จะมีการประมวลหรือกระทำต่อชุดคำสั่งที่อยู่ในเงื่อนไขเพียงทางเลือกเดียวเท่านั้น โดยจะไม่มีการประมวลต่อทิศทางที่ไม่ตรงกับเงื่อนไขเป็นลักษณะของทางเลือกแบบ 1 เงื่อนไข

2. คำสั่ง if...else

คำสั่ง if...else คือ คำสั่งที่มีการทำงานของโปรแกรมที่มีทิศทางการทำงานแบบ 2 ทางเลือก โครงสร้างการใช้คำสั่งมีรูปแบบดังต่อไปนี้

```
If (เงื่อนไข) :
        ชุดคำสั่งที่ 1
else :
        ชุดคำสั่งที่ 2
```

จากโครงสร้างรูปแบบการใช้คำสั่ง if...else มีลักษณะการทำงาน รายละเอียดดังผังโปรแกรมต่อไปนี้



ภาพที่ 3.3 การทำงานของคำสั่ง if...else

ภาพที่ 3.3 โครงสร้างคำสั่งแบบ if...else ซึ่งมีหลักการทำงานคือ เมื่อมีการรับข้อมูลเข้ามา ข้อมูลนั้นจะนำเข้าไปทำการตรวจสอบในเงื่อนไข ถ้าข้อมูลเข้าเงื่อนไขที่เป็นจริง จะทำการประมวลชุดคำสั่งที่ 1 แต่ถ้าไม่เข้าเงื่อนไขหรือเงื่อนไขเป็นเท็จ จะประมวลผลคำสั่งชุดที่ 2 คำสั่ง if มีรูปแบบและลักษณะการทำงานดังโปรแกรมตัวอย่างที่ 3.3

ตัวอย่างที่ 3.3 การใช้คำสั่ง if...else

บรรทัดที่	คำสั่ง
1	point = float(input("คะแนนเก็บ = "))
2	final = float(input("คะแนนปลายภาค = "))
3	total=point+final
4	print ("คุณได้คะแนน ",total)
5	if point>=50 :
6	print ("คุณสอบผ่านวิชาหลักการเขียนโปรแกรม")
7	else :
8	print ("คุณสอบไม่ผ่านวิชาหลักการเขียนโปรแกรม")
ผลลัพธ์ที่ 1	
คะแนนเก็บ = 32	(รับข้อมูลจากแผงแป้นอักขระ)

คะแนนปลายภาค = 20	(รับข้อมูลจากแผงแป้นอักขระ)
คุณได้คะแนน 52.0	
คุณสอบผ่านวิชาหลักการเขียนโปรแกรม	
ผลลัพธ์ที่ 2	
คะแนนเก็บ = 35	(รับข้อมูลจากแผงแป้นอักขระ)
คะแนนปลายภาค = 13	(รับข้อมูลจากแผงแป้นอักขระ)
คุณได้คะแนน 48.0	
คุณสอบไม่ผ่านวิชาหลักการเขียนโปรแกรม	

โปรแกรมตัวอย่างที่ 3.3 โปรแกรมตรวจสอบคะแนนของนักศึกษา โดยให้มีการรับข้อมูลเข้ามาจากแผงแป้นอักขระและเก็บไว้ในตัวแปร point และ final จากนั้นนำข้อมูลในตัวแปรไปทำการคำนวณหาผลรวมและเก็บไว้ในตัวแปร total ตรวจสอบกับเงื่อนไข ถ้าข้อมูลคะแนนรวมมากกว่าหรือเท่ากับ 50 ให้แสดงข้อความ “สอบผ่านวิชาหลักการเขียนโปรแกรม” ดังผลลัพธ์ที่ 1 และถ้าข้อมูลคะแนนรวมน้อยกว่า 50 จะแสดงข้อความ “สอบไม่ผ่านวิชาหลักการเขียนโปรแกรม” ดังผลลัพธ์ที่ 2

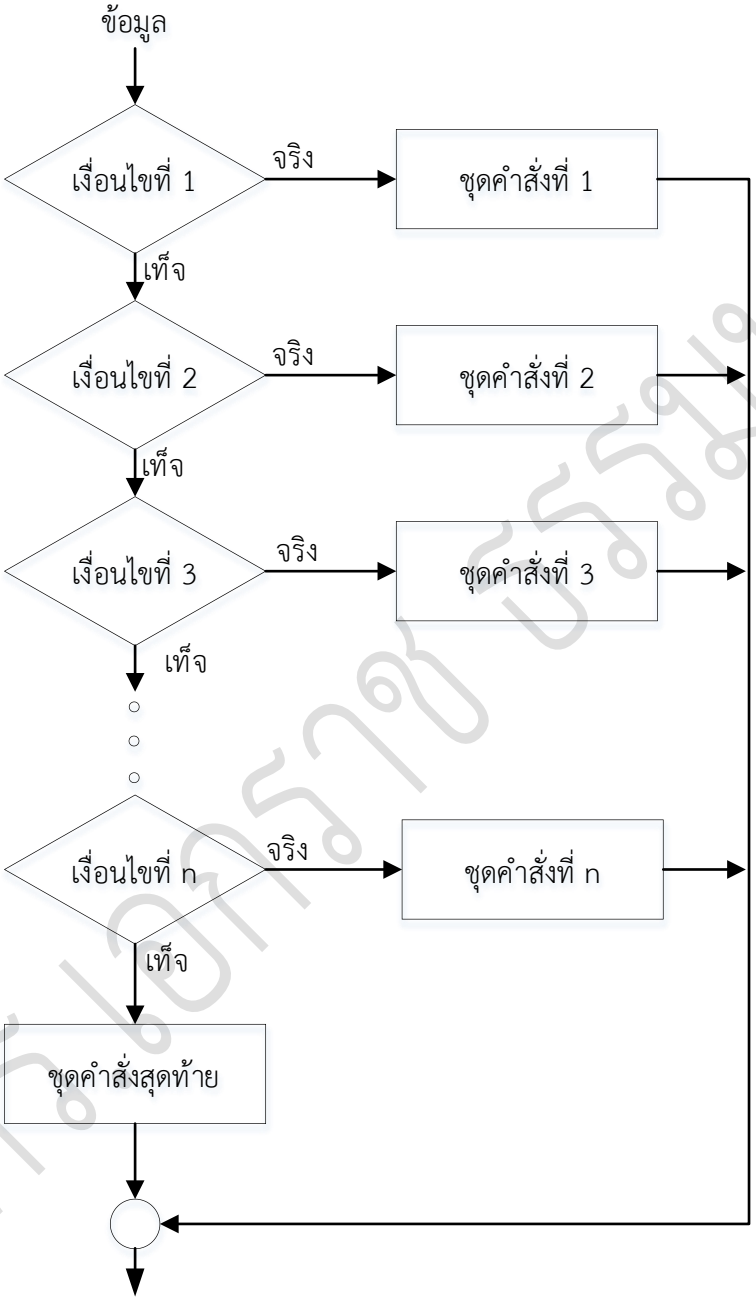
การใช้คำสั่ง if...else จะมีความแตกต่างจากคำสั่ง if คือ มีการประมวลผลชุดคำสั่งทั้งสองทิศทาง ซึ่งแต่ละชุดคำสั่งจะแยกทิศทางตามเงื่อนไขของโปรแกรม ทั้งในทิศทางที่เป็นจริงและเป็นเท็จ ดังผลลัพธ์ที่ได้ของโปรแกรม

3. คำสั่ง if...elif...else

คำสั่ง if...elif...else คือ คำสั่งการทำงานของโปรแกรม ที่มีเงื่อนไขจำนวนมาก สามารถใช้คำสั่งโครงสร้างทางเลือกแบบมากกว่า 2 ทางเลือก ในการตรวจสอบเงื่อนไข โครงสร้างการใช้คำสั่งมีรูปแบบดังต่อไปนี้

```
If (เงื่อนไขที่ 1) :
        ชุดคำสั่งที่ 1
elif (เงื่อนไขที่ 2) :
        ชุดคำสั่งที่ 2
                .
                .
                .
elif (เงื่อนไขที่ n) :
        ชุดคำสั่งที่ n
else :
        ชุดคำสั่งสุดท้าย
```

จากโครงสร้างรูปแบบการใช้คำสั่ง if...elif...else มีลักษณะการทำงาน รายละเอียดดังผังโปรแกรมต่อไปนี้



ภาพที่ 3.4 การทำงานของคำสั่ง if...elif...else

โครงสร้างการทำงานของคำสั่ง if...elif...else จะมีการตรวจสอบเงื่อนจำนวนมาก ซึ่งการตรวจสอบจะทำการตรวจสอบทีละเงื่อนที่อยู่ในลำดับแรก ถ้าไม่เข้าเงื่อนไขหรือเงื่อนไขเป็นจริงจะประมวลผลชุดคสั่งที่อยู่ในเงื่อนไขนั้น แต่ถ้าหากไม่เข้าเงื่อนไขจะตรวจสอบเงื่อนไขที่อยู่ในลำดับถัด

มาลงมาจนสิ้นสุด สามารถนำหลักการและรูปแบบการทำงานของคำสั่งใช้ในการเขียนโปรแกรม มีรายละเอียดการทำงานดังโปรแกรมตัวอย่างต่อไปนี้

ตัวอย่างที่ 3.4 การใช้คำสั่ง if...elif...else

บรรทัดที่	คำสั่ง
1	point = float(input("นักศึกษาได้คะแนน = "))
2	if point>=80 :
3	print ("คุณได้เกรด A")
4	elif(point>=70) :
5	print ("คุณได้เกรด B")
6	elif(point>=60) :
7	print ("คุณได้เกรด C")
8	elif(point>=50) :
9	print ("คุณได้เกรด D")
10	else :
11	print ("คุณได้เกรด E")
ผลลัพธ์	
นักศึกษาได้คะแนน = 66 (รับข้อมูลจากแผงแป้นอักขระ) คุณได้เกรด C	

โปรแกรมตัวอย่างที่ 3.4 แสดงเกรดที่ได้จากการประมวลผลข้อมูลคะแนนที่รับเข้ามา โดยมีรูปแบบการทำงานของเงื่อนไขที่มากกว่า 2 เงื่อนไขขึ้นไป เงื่อนไขแรกจะใช้คำสั่ง if ในการตรวจสอบเงื่อนไขลำดับแรก คือการตรวจสอบคะแนนที่มากกว่าหรือเท่ากับ 80 ถ้าเข้าเงื่อนไขหรือเงื่อนไขเป็นจริงจะแสดงข้อความ “คุณได้เกรด A” แต่ถ้าไม่เข้าเงื่อนไขหรือเงื่อนไขเป็นเท็จ เงื่อนไขที่อยู่ลำดับถัดมาจะถูกทำการตรวจสอบโดยใช้คำสั่ง elif ตามด้วยตามลำดับเงื่อนไขต่อมาจนสิ้นสุดเงื่อนไขในการตรวจสอบ จะใช้คำสั่ง else ในลำดับสุดท้ายเพื่อประมวลผลชุดคำสั่งที่ไม่เข้าเงื่อนไขที่อยู่ในลำดับบน ซึ่งเป็นส่วนสุดท้ายของโปรแกรม

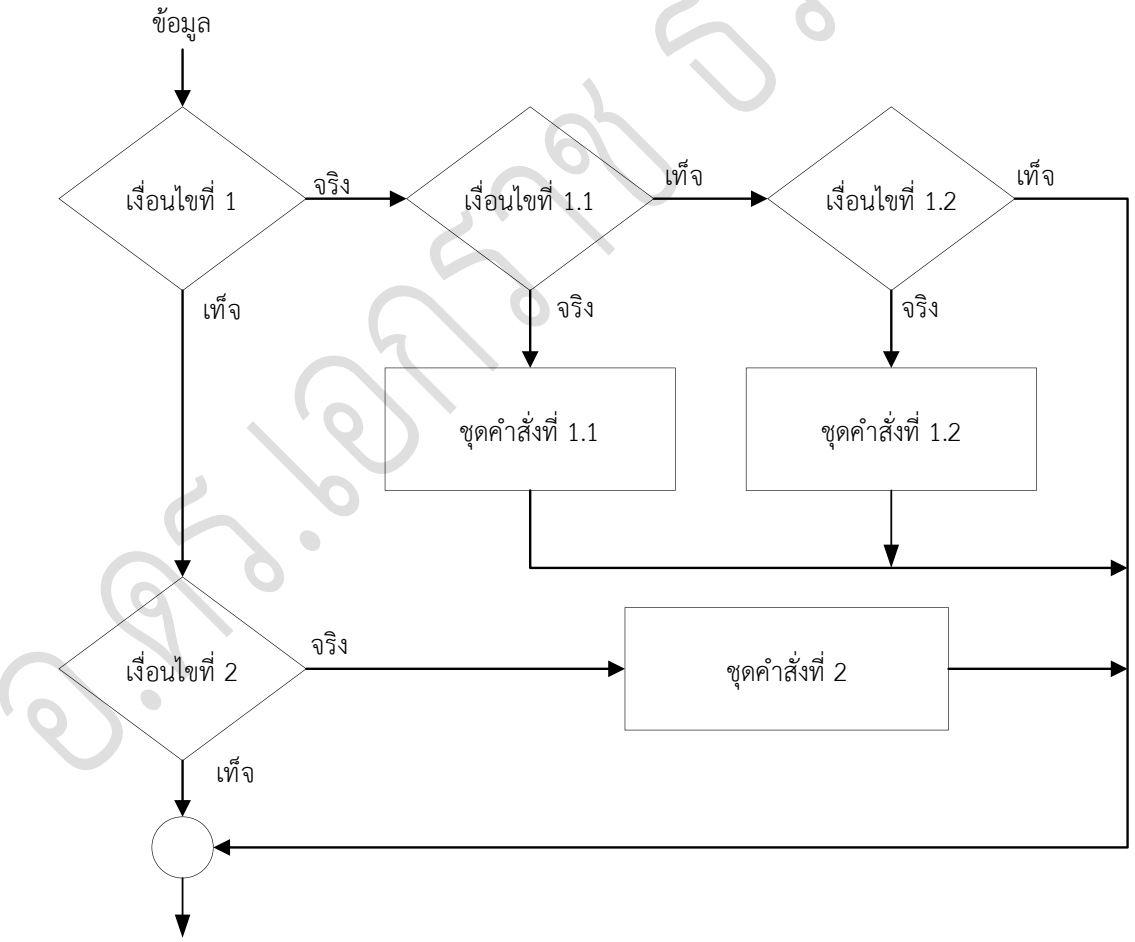
4. Nested if

โครงสร้างคำสั่งแบบ Nested if คือ การทำงานของโปรแกรมอาจมีการตรวจสอบเงื่อนไขหลายชั้น ลักษณะของการทำงานดังกล่าว จะมีการใช้คำสั่ง if ซ้อนกัน รูปแบบลักษณะของการคำสั่ง

if ซ้อนด้วยคำสั่ง ลักษณะการซ้อนกันของคำสั่งอาจมีมากกว่ากว่า 1 ชั้นได้ โครงสร้างการใช้คำสั่งมีรูปแบบดังต่อไปนี้

```
คำสั่ง if
        คำสั่ง if
                คำสั่ง if
                        ...
```

จากโครงสร้างรูปแบบการใช้คำสั่ง Nested if มีลักษณะการทำงาน รายละเอียดดังผังโปรแกรมต่อไปนี้



ภาพที่ 3.5 การทำงานของคำสั่ง Nested if

จากภาพที่ 3.5 การทำงานของโปรแกรมในการตรวจสอบเงื่อนไข ที่มีบางเงื่อนไขที่จะต้องมีการตรวจสอบซ้ำอีกครั้ง เช่น เงื่อนไขที่ 1 มีการตรวจสอบซ้อนข้างใน คือ เงื่อนไข 1.1 และ 1.2 ถ้าเข้าเงื่อนไขใด ก็จะประมวลผลชุดคำสั่งนั้น ซึ่งโครงสร้างการทำงานของคำสั่ง Nested if สามารถนำไปใช้ โดยมีรูปแบบและลักษณะการทำงานดังโปรแกรมตัวอย่างที่ 3.5

ตัวอย่างที่ 3.5 รูปแบบโครงสร้างแบบ Nested if

บรรทัดที่	คำสั่ง
1	member = input("คุณเป็นสมาชิกหรือไม่ (y/n) : ")
2	price = float(input("กรอกราคาสินค้า : "))
3	if (member=="y"):
4	if(price>= 1000) :
5	money=price-(price*10/100)
6	else :
7	money=price-(price*5/100)
8	else :
9	money=price
10	print("เงินจ่ายค่าสินค้า : ",money)
ผลลัพธ์ที่ 1	
คุณเป็นสมาชิกหรือไม่ (y/n) : y	(รับข้อมูลจากแผงแป้นอักขระ)
กรอกราคาสินค้า : 15000	(รับข้อมูลจากแผงแป้นอักขระ)
เงินจ่ายค่าสินค้า : 13500.0	
ผลลัพธ์ที่ 2	
คุณเป็นสมาชิกหรือไม่ (y/n) : n	(รับข้อมูลจากแผงแป้นอักขระ)
กรอกราคาสินค้า : 15000	(รับข้อมูลจากแผงแป้นอักขระ)
เงินจ่ายค่าสินค้า : 15000.0	

โปรแกรมตัวอย่างที่ 3.5 ลักษณะของการใช้คำสั่งซ้อนกันของคำสั่ง if...else... ซึ่งเงื่อนไขแรกของคำสั่งให้ตรวจสอบการเป็นสมาชิก ถ้ากรอกข้อมูล y จะถือว่าเป็นสมาชิกและได้รับส่วนลด 10% ถ้าซื้อสินค้าราคา 1000 บาท ขึ้นไป และจะได้รับส่วนลด 5% เมื่อซื้อสินค้าราคาต่ำกว่า 1000 บาท ดังผลลัพธ์ที่ 1 แต่ถ้ากรอกข้อมูลการเป็นสมาชิกไม่ใช่ y โปรแกรมจะถือว่าไม่ได้เป็นสมาชิก จะไม่มี

ส่วนลดให้ ดังผลลัพธ์ที่ 2 การซ้อนกันของคำสั่ง นอกเหนือจากตัวอย่างดังกล่าว ยังสามารถใช้คำสั่งอื่นมาทำงานซ้อนกันได้หลายคำสั่งและสามารถซ้อนกันได้หลายชั้น

ลักษณะการทำงานของโปรแกรมโครงสร้างทิศทางแบบทางเลือก มีทั้งแบบ 1 ทางเลือก จะใช้คำสั่ง if แบบ 2 ทางเลือก จะใช้คำสั่ง if...else และแบบมากกว่า 2 ทางเลือก จะใช้คำสั่ง if...elif...else นอกเหนือจากนี้มีลักษณะของการใช้คำสั่งทั้ง 3 มาทำงานในโปรแกรมร่วมกันลักษณะตรวจสอบเงื่อนไขซ้อนกันเรียกว่า Netted สามารถนำมาแก้ไขปัญหาการทำงานของโปรแกรมที่มีความซับซ้อนให้มีประสิทธิภาพเพิ่มมากขึ้น

คำสั่งควบคุมทิศทางแบบวนซ้ำ

โครงสร้างทิศทางการทำงานแบบทางเลือก จะมีการตรวจสอบเงื่อนไขและประมวลผลชุดคำสั่งที่อยู่ภายใต้เงื่อนไขนั้น แยกการทำงานออกจากกันและเมื่อประมวลผลตามเงื่อนไขก็จะสิ้นสุดการทำงานหรือไปทำงานต่อส่วนของโปรแกรมที่อยู่ถัดไป แต่จะมีการทำงานของโปรแกรมอีกประเภทที่มีการตรวจสอบเงื่อนไขและมีการวนกลับมาประมวลผลชุดคำสั่งอีกครั้ง

คำสั่งทิศทางแบบวนซ้ำ (Loop Statement) หมายถึง โครงสร้างการทำงานที่มีการตรวจสอบข้อมูลกับเงื่อนไข และมีผลสรุปอย่างใดอย่างหนึ่ง ที่จะมีการกลับไปทำงานซ้ำคำสั่งอีกครั้ง (Halterman, 2016) หรือการทำงานแบบซ้ำไปซ้ำมาของชุดคำสั่ง ตามเงื่อนไขที่ได้กำหนดไว้ โดยใช้ตัวดำเนินการแบบต่าง ๆ มาเป็นตัวช่วยในการกำหนดเงื่อนไขการทำงาน (กิตินันท์ พลสวัสดิ์, 2554)

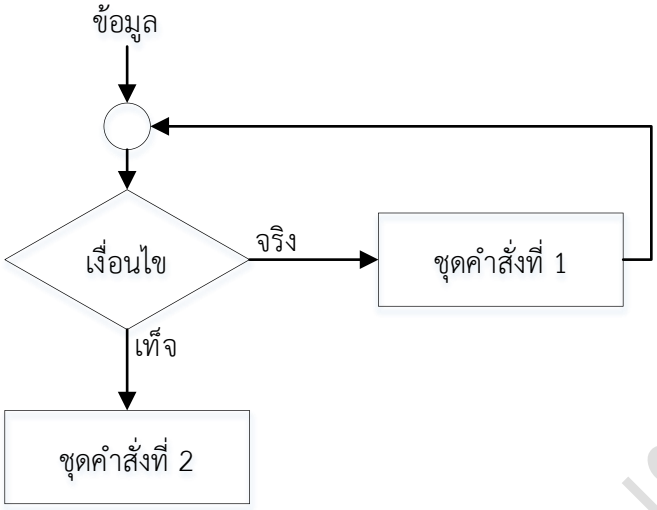
การทำงานของโครงสร้างทิศทางแบบวนซ้ำ มีคำสั่งที่ใช้ในภาษาไพธอนหลายประเภท ซึ่งแต่ละประเภทจะมีลักษณะ รูปแบบและการทำงานที่แตกต่างกัน มีรายละเอียดของการใช้คำสั่งดังต่อไปนี้

1. คำสั่ง while

คำสั่ง while คือ คำสั่งทีมีลักษณะของการตรวจสอบเงื่อนไขก่อน เมื่อเข้าเงื่อนไขหรือเงื่อนไขเป็นจริงจะประมวลผลชุดคำสั่งนั้นซ้ำ โครงสร้างการใช้คำสั่งมีรูปแบบดังต่อไปนี้

```
while (เงื่อนไข) :
    ชุดคำสั่งที่ 1
ชุดคำสั่งที่ 2
```

จากโครงสร้างรูปแบบการใช้คำสั่ง while มีลักษณะการทำงาน รายละเอียดดังผังโปรแกรมต่อไปนี้



ภาพที่ 3.6 โครงสร้างแบบวนซ้ำ

โครงสร้างของคำสั่ง while มีลักษณะของการตรวจสอบข้อมูลกับเงื่อนไข หากเข้าเงื่อนไขหรือเงื่อนไขเป็นจริง จะทำการประมวลผลชุดคำสั่งที่อยู่ภายใต้เงื่อนไขนั้น ข้อมูลที่ได้จากประมวลจะถูกส่งกลับไปประมวลผลที่เงื่อนไขอีกครั้ง จะกระทำเช่นนี้ไปจนกว่าการตรวจสอบเงื่อนไขและพบว่าไม่เข้าเงื่อนไขหรือเงื่อนไขเป็นเท็จ จึงหยุดการวนซ้ำและไปประมวลคำสั่งอื่นต่อไป โดยมีลักษณะของการทำงานดังโปรแกรมตัวอย่างที่ 3.6

ตัวอย่างที่ 3.6 การใช้คำสั่ง while

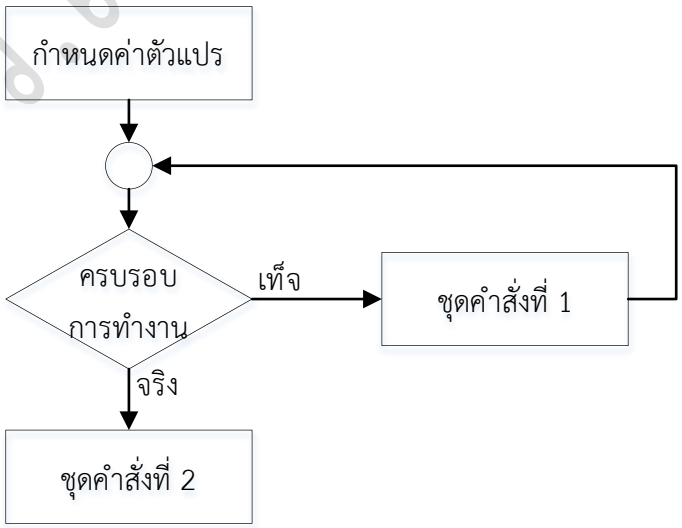
บรรทัดที่	คำสั่ง
1	i=1
2	total=0
3	num=int(input("จำนวนผู้เล่นกีฬา : "))
4	while (i<=num):
5	print("อายุคนที่",i)
6	age=int(input())
7	i=i+1
8	total = total+age
9	average = total/num
10	print ("อายุเฉลี่ยนักกีฬา จำนวน :",num,"มีอายุเฉลี่ย",average)
ผลลัพธ์	
จำนวนผู้เล่นกีฬา : 3 (รับข้อมูลจากแผงแป้นอักขระ)	

```
อายุคนที่ 1
19                                          (รับข้อมูลจากแผงแป้นอักขระ)
อายุคนที่ 2
27                                          (รับข้อมูลจากแผงแป้นอักขระ)
อายุคนที่ 3
22                                          (รับข้อมูลจากแผงแป้นอักขระ)
อายุเฉลี่ยนักกีฬา จำนวน : 3 มีอายุเฉลี่ย 22.666666666666668
```

โปรแกรมตัวอย่างที่ 3.6 เริ่มต้นด้วยการกำหนดตัวแปร i ทำหน้าที่ในการนับจำนวนผู้เล่นกีฬาแต่ละรอบมีค่าเท่ากับ 1 และตัวแปร total ทำหน้าที่ในการหาอายุรวมเท่ากับ 0 การทำงานของโปรแกรมจะรับข้อมูลจำนวนนักกีฬาทั้งหมดที่ต้องการหาอายุเฉลี่ยเก็บไว้ที่ตัวแปร num เริ่มการทำงานตัวแปร i จะถูกตรวจสอบกับตัวแปร num ผ่านคำสั่ง while โดยมีเงื่อนไข i น้อยกว่าหรือเท่ากับ num คือ ถ้าจำนวนผู้เล่นกีฬายังไม่เกินจำนวนนักกีฬาทั้งหมดของแต่ละรอบ ให้ทำซ้ำต่อกันไปเรื่อย ๆ ภายในคำสั่งน while จะรับข้อมูลอายุไว้ในตัวแปร age และตัวแปร total คำนวณอายุรวมในแต่ละรอบ ช่วงระหว่างแต่ละรอบการทำงาน ตัวแปร i จะเพิ่มค่าทีละ 1 เพื่อนับรอบการทำงานทำซ้ำไปจน i มีค่ามากกว่า num จึงหยุดการทำซ้ำและให้คำนวณหาค่าเฉลี่ยเก็บไว้ที่ตัวแปร average พร้อมทั้งแสดงอายุเฉลี่ย ดังผลลัพธ์ที่ได้ของโปรแกรม

2. คำสั่ง for

คำสั่ง for คือ คำสั่งที่มีลักษณะของการนับรอบการทำงาน ถ้าเกินจำนวนที่กำหนด จะหยุดการวนซ้ำ โครงสร้างการใช้คำสั่งมีรูปแบบดังต่อไปนี้



ภาพที่ 3.7 โครงสร้างของคำสั่ง for

การใช้คำสั่ง for มีรูปแบบของการใช้งานหลายประเภท แต่ละรูปแบบมีโครงสร้างและรายละเอียดการทำงานดังต่อไปนี้

2.1 คำสั่ง for...in range()

คำสั่ง for...in range() คือ คำสั่งที่มีลักษณะให้ทำการนับค่าตั้งแต่ค่าเริ่มต้นของตัวแปร ไปจนถึงค่าสุดท้ายที่กำหนด มีรูปแบบการใช้คำสั่งดังต่อไปนี้

```
for ตัวแปร in range (ค่าสุดท้าย) :
        ชุดคำสั่งที่ 1
ชุดคำสั่งที่ 2
```

คำสั่ง for...in range() สามารถนำไปเขียนโปรแกรม โดยมีรูปแบบและลักษณะการทำงานดังโปรแกรมตัวอย่างที่ 3.7

ตัวอย่างที่ 3.7 การใช้คำสั่ง for...in range()

บรรทัดที่	คำสั่ง
1 2	for k in range(5): print (k)
ผลลัพธ์	
0 1 2 3 4	

คำสั่ง for...in range() มีลักษณะรูปแบบการทำงานแบบวนซ้ำของโปรแกรม โดยแต่ละรอบของการวนซ้ำ จะมีการเพิ่มค่าทีละ 1 ดังโปรแกรมตัวอย่างที่ 3.7 แสดงค่าของตัวแปรในการวนซ้ำแต่ละรอบ ค่าเริ่มต้นของตัวแปร k ไม่ได้กำหนด ดังนั้นค่าเริ่มต้นลำดับแรกของค่า k จะมีค่าเท่ากับ 0 และเมื่อโปรแกรมทำงานโปรแกรมก็จะแสดงค่าเริ่มแรก 0 และเพิ่มค่าทีละ 1 ต่อเนื่องไปจนกระทั้งจนถึงค่าสุดท้ายลำดับที่ 5 มีค่าเท่ากับ 4 โปรแกรมจะสิ้นสุดการทำงาน ดังนั้น ค่าสุดท้ายจะมีค่าน้อยกว่าค่าที่กำหนดสุดท้าย 1 ค่าเสมอ

2.2 คำสั่ง for...in range() แบบกำหนดค่าเริ่มต้นและค่าสุดท้าย

คำสั่ง for...in range() แบบกำหนดค่าเริ่มต้นและค่าสุดท้าย คือ คำสั่งที่มีลักษณะการกำหนดค่าเริ่มต้นและค่าสุดท้าย โดยจะนับค่าจากการกำหนดค่าเริ่มต้นให้กับตัวแปร มีรูปแบบการใช้คำสั่งดังต่อไปนี้

```
for ตัวแปร in range (ค่าเริ่มต้น, ค่าสุดท้าย):
    ชุดคำสั่ง
```

คำสั่ง for...in range() แบบกำหนดค่าเริ่มต้นและค่าสุดท้าย สามารถนำไปเขียนโปรแกรมโดยมีรูปแบบและลักษณะการทำงานดังโปรแกรมตัวอย่างที่ 3.8

ตัวอย่างที่ 3.8 การใช้คำสั่ง for...in range() โดยกำหนดค่าเริ่มต้นและค่าสุดท้าย

บรรทัดที่	คำสั่ง
1 2	for k in range(1,5): print (k)
ผลลัพธ์	
1 2 3 4	

คำสั่ง for...in range() แบบกำหนดค่าเริ่มต้นและค่าสุดท้าย มีการกำหนดค่าเริ่มต้นและค่าสุดท้ายของตัวแปร จากโปรแกรมตัวอย่างที่ 3.8 ให้เริ่มต้นลำดับแรกมีค่าเท่ากับ 1 ทำซ้ำไปจนกระทั่งจนถึงค่าสุดท้าย

2.3 คำสั่ง for...in range() แบบกำหนดจำนวนเพิ่มค่า

คำสั่ง for...in range() แบบกำหนดจำนวนเพิ่มค่า มีการกำหนดตัวเลข 3 จำนวนประกอบด้วย ค่าเริ่มต้น ค่าสุดท้ายและค่าที่เพิ่ม โดยมีรายละเอียดของคำสั่งดังต่อไปนี้

```
for ตัวแปร in range (ค่าเริ่มต้น, ค่าสุดท้าย, ค่าที่เพิ่ม):
    ชุดคำสั่ง
```

คำสั่ง for...in range() แบบกำหนดจำนวนเพิ่มค่า สามารถนำไปเขียนโปรแกรม โดยมีรูปแบบและลักษณะการทำงานดังโปรแกรมตัวอย่างที่ 3.9

ตัวอย่างที่ 3.9 การใช้คำสั่ง for...in range() แบบกำหนดจำนวนเพิ่มค่า

บรรทัดที่	คำสั่ง
1 2	for k in range(1,14,2): print (k)
ผลลัพธ์	
1 3 5 7 9 11 13	

คำสั่ง for...in range() สำหรับกำหนดจำนวนการเพิ่ม มีการทำงานที่คล้ายกับ for แบบกำหนดค่าเริ่มต้น แต่ for...in range() แบบกำหนดค่าเริ่มต้น จะมีค่าที่เพิ่มเพียงค่าละ 1 เท่านั้น แต่ for แบบกำหนดจำนวนการเพิ่มค่า สามารถเพิ่มค่ามากกว่า 1 ดังโปรแกรมตัวอย่างที่ 3.9 จำนวนค่าที่เพิ่ม จะเพิ่มทีละ 2 ค่า

2.4 คำสั่ง for...in range() แบบกำหนดจำนวนลดค่า

คำสั่ง for...in range() แบบกำหนดจำนวนลดค่า จะมีการกำหนดค่า 3 ตำแหน่ง ได้แก่ ค่าเริ่มต้น ค่าสุดท้ายและค่าที่ลด โดยมีรายละเอียดของคำสั่งดังต่อไปนี้

```
for ตัวแปร in range (ค่าเริ่มต้น, ค่าสุดท้าย, ค่าที่ลด):
    ชุดคำสั่ง
```

คำสั่ง for...in range() แบบกำหนดจำนวนลดค่า สามารถนำไปเขียนโปรแกรม โดยมีรูปแบบและลักษณะการทำงานดังโปรแกรมตัวอย่างที่ 3.10

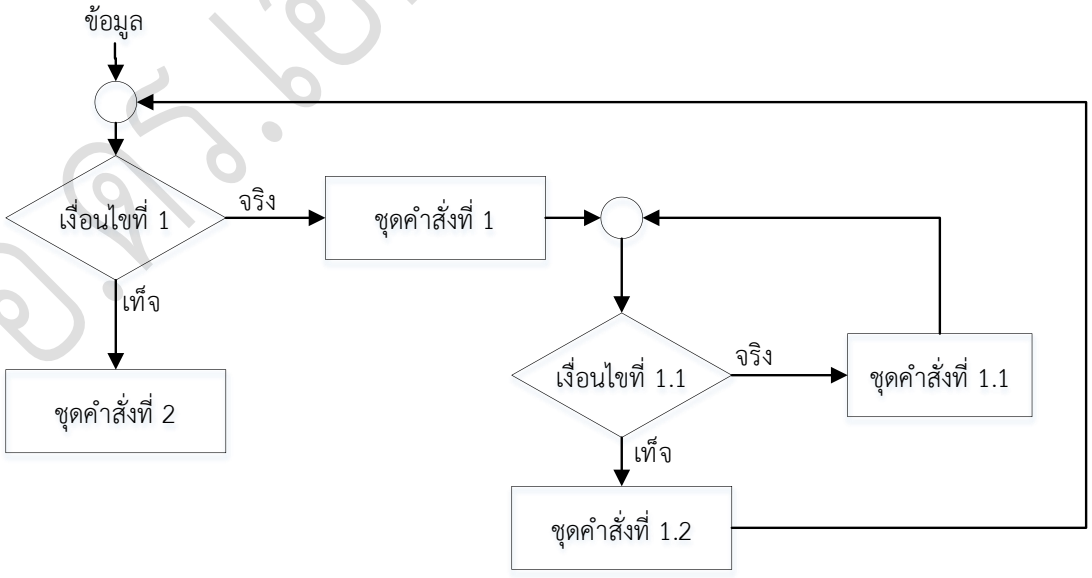
ตัวอย่างที่ 3.10 การใช้คำสั่ง for...in range() แบบกำหนดจำนวนลดค่า

บรรทัดที่	คำสั่ง
1 2	for k in range(10,0,-3): print (k)
ผลลัพธ์	
10 7 4 1	

คำสั่ง for...in range() แบบกำหนดจำนวนลดค่า มีรูปแบบคล้ายกับแบบเพิ่มค่า นั่นคือ จะทำการกำหนดค่าค่าเริ่มต้นมากกว่าค่าสุดท้าย และกำหนดจำนวนในการลดค่าในแต่ละรอบ โดยค่าที่ให้ลดอยู่ภายใต้เครื่องหมายลบ (-) ดังโปรแกรมตัวอย่างที่ 3.10

3. Nested Loop

Nested Loop คือ ลักษณะการทำงานของโปรแกรม ที่มีโครงสร้างของการทำงานของคำสั่งที่มีการทำซ้ำในคำสั่งวนซ้ำอีกรอบ ซึ่งอาจมีการซ้อนกันหลายชั้นของคำสั่ง มีโครงสร้างและลักษณะการทำงานดังต่อไปนี้



ภาพที่ 3.8 โครงสร้าง Nested Loop

โครงสร้างการทำงานของคำสั่ง Nested Loop สามารถนำไปเขียนโปรแกรม โดยมีรูปแบบและลักษณะการทำงานดังโปรแกรมตัวอย่างที่ 3.11

ตัวอย่างที่ 3.11 การทำงานแบบ Nested Loop

บรรทัดที่	คำสั่ง
1	i=1
2	while (i<=3):
3	print("สูตรคูณแม่ ",i)
4	for k in range(1,13):
5	print(i, " * ",k," = ",i*k)
6	print("----------")
7	i=i+1

ผลลัพธ์

```
สูตรคูณแม่  1
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10
1 * 11 = 11
1 * 12 = 12
----------
สูตรคูณแม่  2
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
```

```
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
2 * 11 = 22
2 * 12 = 24
---------------------------
สูตรคูณแม่ 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
3 * 11 = 33
3 * 12 = 36
---------------------------
```

การสร้างโปรแกรมสูตรคูณ โดยใช้หลักการโครงสร้างแบบ Nested Loop โครงการทำงานของโปรแกรม ประกอบด้วยการใช้คำสั่ง for ที่จะให้เลขลำดับ 1 ถึง 12 ซ้อนในอยู่ในคำสั่ง while ที่เป็นแม่สูตรคูณ 1 ถึง 3 ผลลัพธ์ที่ได้ดังโปรแกรมตัวอย่างที่ 3.11

โครงสร้างการทำงานของโปรแกรมแบบซ้อนหรือ Netted นอกเหนือจาการซ้อนกันของคำสั่งในโครงสร้างแบบเดียวกันดังที่กล่ามาข้างต้น การซ้อนกันของคำสั่งสามารถซ้อนกันต่างโครงสร้างได้ บางโจทย์ปัญหาอาจต้องใช้โครงสร้างแบบทางเลือกซ้อนอยู่ในคำสั่งวนซ้ำหรือคำสั่งวนซ้ำซ้อนอยู่ในคำสั่งแบบทางเลือก โดยมีรายละเอียดดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 3.12 การซ้อนกันของคำสั่งโครงแบบวนซ้ำและคำสั่งแบบทางเลือก

บรรทัดที่	คำสั่ง
1	print("----โปรแกรมคำนวณเงินจ่ายหลังหักส่วนลดสินค้าแต่ละรายการ-----")
2	i=1
3	total=0
4	money=0
5	num=int(input("จำนวนสินค้า : "))
6	while (i<=num):
7	print("ราคาสินค้าที่",i)
8	price=float(input())
9	if price>=1000:
10	discout=5
11	elif price>=500:
12	discout=3
13	else:
14	discout=0
15	money=price-(price*discout/100)
16	i=i+1
17	total = total+money
18	print ("รวมเป็นเงินทั้งสิ้น",total)

ผลลัพธ์

```
----โปรแกรมคำนวณเงินจ่ายหลังหักส่วนลดสินค้าแต่ละรายการ-----
จำนวนสินค้า : 3
ราคาสินค้าที่ 1
3000                                    (รับข้อมูลจากแผงแป้นอักขระ)
ราคาสินค้าที่ 2
800                                     (รับข้อมูลจากแผงแป้นอักขระ)
ราคาสินค้าที่ 3
400                                     (รับข้อมูลจากแผงแป้นอักขระ)
รวมเป็นเงินทั้งสิ้น 4026.0
```

โปรแกรมตัวอย่างที่ 3.12 โครงสร้างการทำงานของโปรแกรม ที่ให้คำนวณเงินจ่ายสินค้ารวมทั้งหมดหลังหักส่วนลดของสินค้าแต่ละรายการ สินค้าจะมีส่วนลดที่แตกต่างกันตามราคาสินค้า เริ่มต้นการทำงานของโปรแกรม โดยการรับตัวเลขจำนวนสินค้าทั้งหมด และนำข้อมูลเข้าแต่ละรายการผ่านคำสั่ง while การรับข้อมูลราคาสินค้าแต่ละรายการมีส่วนลดที่แตกต่างกัน โดยการคำนวณสินค้าหลังหักส่วนลดนั้นใช้คำสั่ง if ในการตรวจสอบเงื่อนไขคำนวณราคาสินค้าหลังหักส่วนลดแต่ละรายการ ซึ่งการทำงานดังกล่าว จะเห็นได้ว่ามีการซ้อนกันของคำสั่งต่างทิศทาง ทั้งนี้เพื่อแก้ปัญหาโจทย์ที่มีความซ้อน การใช้คำสั่งแบบ Nested จึงนำมาใช้เพื่อให้โปรแกรมมีประสิทธิภาพตรงกับวัตถุประสงค์การทำงานของโปรแกรม

สรุป

ทิศทางการทำงานของโปรแกรม จะมีลักษณะการทำงาน 3 ประเภท ประกอบด้วย ทิศทางการทำงานแบบเรียงลำดับ ทิศทางการทำงานแบบทางเลือกและทิศทางการทำงานแบบวนซ้ำ ลักษณะทิศทางการทำงานแบบเรียงลำดับ จะมีการประมวลผลทุกขั้นตอนการทำงานของโปรแกรม ดังนั้น โครงสร้างทิศทางแบบเรียงลำดับ จึงไม่มีคำสั่งที่ใช้ในการควบคุมทิศทางการทำงานของโปรแกรม แต่โครงสร้างทิศทางการทำงานแบบทางเลือกและแบบทำซ้ำ จะมีเงื่อนไขที่จะเลือกทิศทางการทำงานของโปรแกรม จึงมีคำสั่งตรวจสอบเงื่อนไข ซึ่งทิศทางการทำงานแบบทางเลือก จะมีลักษณะของการเลือกประมวลชุดคำสั่งที่ไม่เหมือนกันและหลังจากประมวลผลเสร็จสิ้นแต่ของคำสั่งตรงเงื่อนไข โปรแกรมสามารถจะจบการทำงานและไปดำเนินการในขั้นตอนต่อไป แต่ทิศทางการทำงานแบบวนซ้ำ จะมีการตรวจสอบเงื่อนไขและให้มีการทำซ้ำชุดคำสั่งเดิมอีกครั้ง โดยการทำซ้ำจะกระทำไปจนกว่า จะไม่เข้าเงื่อนไขที่กำหนด ถึงจะหยุดการทำซ้ำและสิ้นสุดการทำงาน คำสั่งที่ใช้ในการควบคุมทิศทางการทำงานแบบทางเลือก ประกอบด้วยคำสั่ง if คือ คำสั่งที่ใช้ในการตรวจสอบเงื่อนไขเพียงเงื่อนไขเดียว คำสั่ง if...else คือ คำสั่งที่ใช้ในการตรวจสอบเงื่อนไข 2 เงื่อนไข คำสั่ง if...elif...else คือ คำสั่งที่ใช้ในการตรวจสอบเงื่อนไขมากกว่า 2 เงื่อนไขขึ้นไป และการทำงานของคำสั่ง if ทั้ง 3 ประเภท ที่ทำงานซ้อนกัน เรียกว่า Nested if ส่วนคำสั่งควบคุมทิศทางการทำงานของโปรแกรมแบบวนซ้ำ ประกอบด้วยคำสั่ง while และ for และคำสั่งเหล่านี้สามารถทำงานร่วมกันในลักษณะของการซ้อนกันของคำสั่งได้ เรียกว่า Nested Loop นอกเหนือจากนี้ยังมีการใช้คำสั่งที่ซ้อนกันระหว่างโครงสร้างทิศทางแบบทางเลือกซ้อนกับแบบวนซ้ำและแบบวนซ้ำซ้อนกับแบบทางเลือก เพื่อให้แก้โจทย์ปัญหาหาที่มีความซับซ้อน ดังนั้น คำสั่งควบคุมทิศทางการทำงานของโปรแกรมถือว่ามีความสำคัญมากที่จะกำหนดทิศทางการทำงานของโปรแกรม ให้ได้ผลลัพธ์ตามเงื่อนที่กำหนดและตามความต้องการของการนำโปรแกรมไปใช้ต่อไป

แบบฝึกหัด

1. ให้ผู้เรียนอธิบายโครงสร้างการทำงานของโปรแกรมและความแตกต่างแต่ละประเภท
2. ให้ผู้เรียนอธิบายการทำงานของคำสั่งควบคุมทิศทางแบบทางเลือกแต่ละคำสั่ง
3. ให้ผู้เรียนอธิบายการทำงานของคำสั่งควบคุมทิศทางแบบวนซ้ำแต่ละคำสั่ง
4. ให้ผู้เรียนอธิบายความแตกต่างระหว่างคำสั่งควบคุมทิศทางการทำงานแบบทางเลือกและแบบวนซ้ำ
5. ให้ผู้เรียนเขียนโปรแกรมหาพื้นที่ของสามเหลี่ยมหน้าจั่ว
6. ให้ผู้เรียนเขียนโปรแกรมแปลงจากองศาฟาเรนไฮต์เป็นองศาเซลเซียส
7. ให้ผู้เรียนเขียนโปรแกรมโดยหาเกรดที่ได้ มีเงื่อนไขช่วงคะแนนดังนี้

คะแนนรวม	80-100	เกรด A	คะแนนรวม	75-79	เกรด B+
คะแนนรวม	70-74	เกรด B	คะแนนรวม	65-69	เกรด C+
คะแนนรวม	60-64	เกรด C	คะแนนรวม	55-59	เกรด D+
คะแนนรวม	50-54	เกรด D	คะแนนรวม	0-49	เกรด E

คะแนนรวมได้มาจาก คะแนนเก็บ คะแนนกลางภาค และคะแนนปลายภาค

8. ให้ผู้เรียนเขียนโปรแกรมหาเงินงวดต่อเดือนรถยนต์ โดยมีอัตราดอกเบี้ย โดย จะต้องมีเงินดาวน์ 15% ของราคารถยนต์ ซึ่งถ้าผ่อนต่ำกว่า 3 ปี คิดดอกเบี้ย 3% ถ้าผ่อน 4-5 ปี คิดดอกเบี้ย 4% และ 6 ปีคิดดอกเบี้ย 5% โดยมีสูตรการคำนวณดังนี้

เงินดาวน์ = ราคารถยนต์ * 15 / 100

ยอดจัดไฟแนนท์ = ราคารถยนต์ – เงินดาวน์

เงินดอกเบี้ย = (ยอดจัดไฟแนนท์ * ดอกเบี้ย / 100) * จำนวนปี

เงินงวดต่อเดือน = (ยอดจัดไฟแนนท์ + เงินดอกเบี้ย) / จำนวนเดือน

9. ให้ผู้เรียนเขียนโปรแกรมหาผลรวมของเลขคู่ ระหว่าง 20 ถึง 40
10. ให้ผู้เรียนเขียนโปรแกรมหาราคารวมจากสินค้าทั้งหมดในตะกร้า พร้อมกับเงินทอน โดยใช้คำสั่ง while และคำสั่ง for

เอกสารอ้างอิง

กิตินันท์ พลสวัสดิ์. (2554). **คู่มือเรียนและใช้งาน Visual C# 2010 ฉบับสมบูรณ์**. กรุงเทพฯ: ไอดีซี พรีเมียร์.

สุชาติ คุ้มมณี. (2558). **เชี่ยวชาญการเรียนด้วยภาษาไพธอน**. มหาสารคาม. คณะวิทยาการสารสนเทศ: มหาวิทยาลัยมหาสารคาม.

Dierbach, Charles. (2013). **Introduction to Computer Science Using Python: A Computational Problem-Solving Focus**. New Jersey: Wiley.

Fior, James M.(2016). **Computer Programming with Python and Multisim**. 3rd ed. London: Pearson.

Halterman, Richard L. (2016). **Fundamentals of Python Programming**. Tennessee: Southern Adventist University.

Willis, Thearon and Newsome, Bryan. (2011). **Beginning Microsoft Visual Basic 2010**. Indianapolis: Wiley.