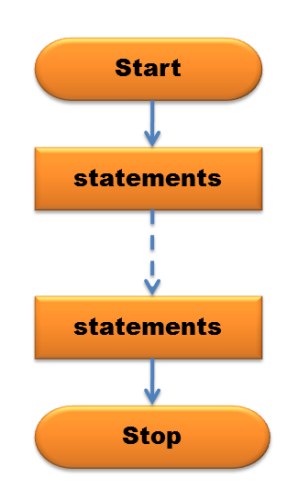
บทที่ 5 คำสั่งควบคุมทิศทางการทำงานของโปรแกรม (Control Statement)

ในการเขียนโปรแกรมโดยทั่วไป ผู้อ่านสามารถกำหนดการทำงานของโปรแกรมได้โดยใช้คำสั่งควบคุมทิศทางการทำงานของโปรแกรมซึ่งมี 3 รูปแบบ คือ

1. คำสั่งควบคุมแบบตามลำดับ (Sequence Control Statement)
2. คำสั่งควบคุมแบบมีทางเลือก (Selection Control Statement)
3. คำสั่งควบคุมแบบทำซ้ำ (Iteration Control Statement)

# คำสั่งควบคุมแบบตามลำดับ (Sequence Control Statement)

**คำสั่งควบคุมแบบตามลำดับ (Sequence Control Statement)** เป็นรูปแบบการทำงานที่ทุกคำสั่งจะทำงานตามลำดับ โดยไม่ข้ามคำสั่งใดๆ ไป แต่ละคำสั่งจะถูกเรียกใช้งานเพียงครั้งเดียว คือ การทำงานจะเริ่มจากคำสั่งที่ 1 แล้วทำคำสั่งที่ 2 จากนั้นทำคำสั่งถัดไปเรื่อยๆ จนครบทุกคำสั่ง ซึ่งสามารถแสดงรูปแบบการทำงานของคำสั่งแบบตามลำดับได้ดังรูป

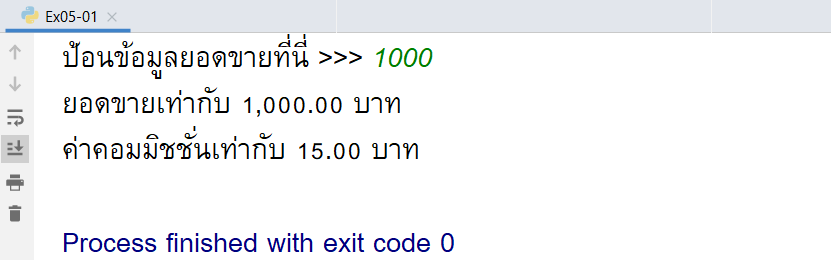
[[python\_05\_01.bmp]]

[[การทำงานของคำสั่งควบคุมแบบตามลำดับ (Sequence Control Statement)]]

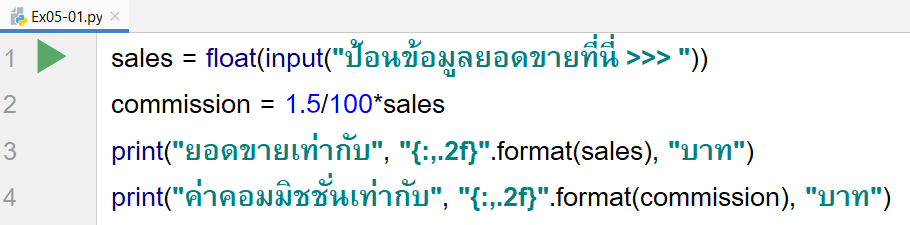
**ตัวอย่างโปรแกรมที่ 5-1** โปรแกรมการใช้คำสั่งควบคุมแบบตามลำดับ เพื่อคำนวณค่าคอมมิชชั่น คิดจาก 1.5% ของยอดขาย

ตัวอย่างนี้เป็นโปรแกรมที่สามารถประยุกต์ใช้กับงานที่มีการคำนวณค่าคอมมิชชั่นจากยอดขายของพนักงานแต่ละคนสามารถขายสินค้าได้

การทำงานของโปรแกรมจึงเริ่มต้นจากการรับทราบข้อมูลยอดขายจึงต้องมีการป้อนข้อมูลยอดขาย จากนั้นโปรแกรมจะนำค่าข้อมูลยอดขายมาคำนวณค่าคอมมิชชั่น และแสดงผลลัพธ์ของโปรแกรม ดังนี้

[[python\_05\_03.bmp]]

ซึ่งจากผลลัพธ์ของโปรแกรมข้างต้น เราสามารถเขียนโค้ดได้ดังนี้

[[python\_05\_02.bmp]]

จากโปรแกรมที่ 5-1 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 รับข้อมูลยอดขายด้วยฟังก์ชัน input() และแปลงค่าข้อมูลเป็นชนิด float แล้วเก็บค่าใน

ตัวแปร sales

บรรทัดที่ 2 คำนวณค่าคอมมิชชั่น โดยคิดจาก 1.5% ของยอดขาย และเก็บค่าผลลัพธ์ในตัวแปร commission

บรรทัดที่ 3-4 แสดงผลยอดขายและค่าคอมมิชชั่นด้วยฟังก์ชัน print() โดยใช้ฟังก์ชัน format() จัดรูปแบบการแสดงผล

# คำสั่งควบคุมแบบมีทางเลือก (Selection Control Statement)

**คำสั่งควบคุมแบบมีทางเลือก (Selection Control Statement)** เป็นรูปแบบการทำงานที่มีการตรวจสอบเงื่อนไขของคำสั่งก่อนทำงาน เพื่อตัดสินใจเลือกทิศทางการทำงานของโปรแกรม โดยจะข้ามการทำงานของบางชุดคำสั่งไป ซึ่งมี 3 คำสั่งคือ คำสั่ง if, คำสั่ง if…else และคำสั่ง nested if

## คำสั่ง if : คำสั่งควบคุมให้โปรแกรมทำงานหรือไม่ทำงานในชุดคำสั่งที่กำหนด

คำสั่ง if เป็นคำสั่งที่ใช้ควบคุมให้โปรแกรมตัดสินใจทำงานหรือไม่ทำงานในชุดคำสั่งที่กำหนด โดยตรวจสอบจากนิพจน์เงื่อนไขทางตรรกศาสตร์ที่กำหนดว่าเป็นจริงหรือเท็จ ถ้านิพจน์ที่กำหนดให้เป็นจริง (true) โปรแกรมจะทำงานที่ชุดคำสั่งที่อยู่หลังคำสั่ง if แต่ถ้านิพจน์ที่กำหนดไว้เป็นเท็จ (false) โปรแกรมจะข้ามไปทำงานที่คำสั่งต่อไปทันที ซึ่งมีรูปแบบการทำงานดังนี้

if boolean\_expression:

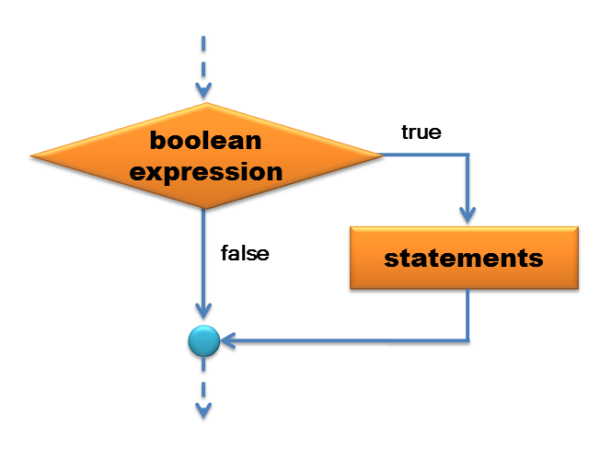
statements

โดยที่ **boolean\_expression** เป็นนิพจน์เงื่อนไขทางตรรกศาสตร์ ซึ่งมีผลการตรวจสอบเงื่อนไขเป็น

จริง (true) หรือเท็จ (false) เท่านั้น

**statements** เป็นชุดของคำสั่งที่จะทำงานเมื่อเงื่อนไขที่กำหนดให้เป็นจริง (true)

จากลักษณะการทำงานของคำสั่งสามารถอธิบายด้วย Flowchart ได้ดังนี้

[[python\_05\_04.bmp]]

ข้อควรระวังในภาษา Python **statements** จะเป็นชุดคำสั่งที่เป็นกลุ่มคำสั่ง (Code Block) หลังเงื่อนไขของ **boolean\_expression** ที่ให้ค่าเป็นจริง จะใช้การ**ย่อหน้า**เป็นการกำหนด block เนื่องจากในรูปแบบไวยากรณ์จะไม่มีการใช้เครื่องหมายวงเล็บปีกกา {…} ครอบกลุ่มคำสั่งดังกล่าว

ตัวอย่าง เช่น

score1 = 40

if score1 >= 50:

print(“Pass”)

print(“OK”)

print(“Bye”)

#ผลที่ได้คือ

Bye

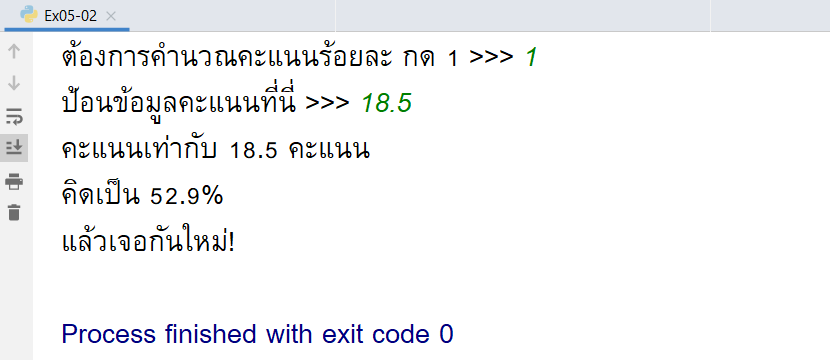
จากตัวอย่างข้างต้น อธิบายได้ว่ากลุ่มคำสั่งที่อยู่หลังคำสั่ง if นั้น ผู้อ่านจะต้องเขียนให้อยู่ในย่อหน้าถัดเข้าไป ให้อยู่ในตำแหน่งย่อหน้าที่ตรงกัน ในตัวอย่างนี้กลุ่มคำสั่งที่อยู่หลังคำสั่ง if ประกอบด้วย 2 คำสั่ง คือ print(“Pass”) และ print(“OK”) ส่วนคำสั่ง print(“Bye”) เป็นคำสั่งที่ทำงานต่อจากคำสั่ง if จึงต้องเขียนให้อยู่ในตำแหน่งย่อหน้าที่ตรงกับคำสั่ง if นั่นหมายความว่า ไม่ว่านิพจน์เงื่อนไขของ score1 >= 50 จะให้ค่าเป็นจริงหรือเป็นเท็จ ก็ไม่มีผลกับการทำงานของคำสั่ง print(“Bye”) ดังนั้นเพื่อเพิ่มความเข้าใจ ขอให้ผู้อ่านได้ศึกษาในตัวอย่างต่อไปนี้

**ตัวอย่างโปรแกรมที่ 5-2** โปรแกรมการใช้คำสั่ง if เพื่อคำนวณคะแนนเปอร์เซ็นต์จากคะแนนเต็ม 35 คะแนน

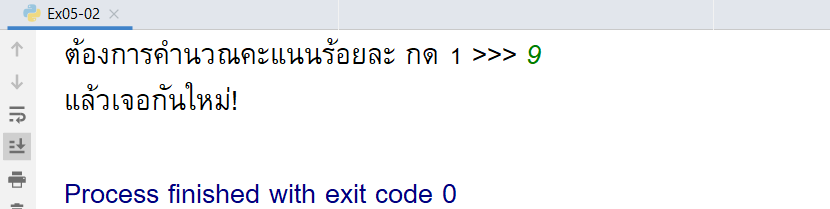
เมื่อกดเลข 1 บนแป้นพิมพ์

ตัวอย่างนี้เป็นโปรแกรมที่สามารถประยุกต์ใช้กับงานที่มีการตัดสินใจ ซึ่งผู้เขียนจะใช้ค่าตัวเลขเป็นตัวตรวจสอบโดยใช้คำสั่ง if ในตัวอย่างนี้ใช้เลข 1 สำหรับกรณีที่ต้องการให้เกิดการทำงานในส่วนของการคำนวณคะแนนเป็นร้อยละหรือเปอร์เซ็นต์ต่อไป หากค่าตัวเลขไม่ใช่เลข 1 ก็จะข้ามการทำงานในส่วนนี้ไป

การทำงานของโปรแกรมจึงเริ่มต้นจากการป้อนข้อมูลตัวเลข เพื่อให้โปรแกรมนำค่าตัวเลขไปตรวจสอบแล้วทำงานตามเงื่อนไขที่กำหนด และแสดงผลลัพธ์ของโปรแกรม ดังนี้

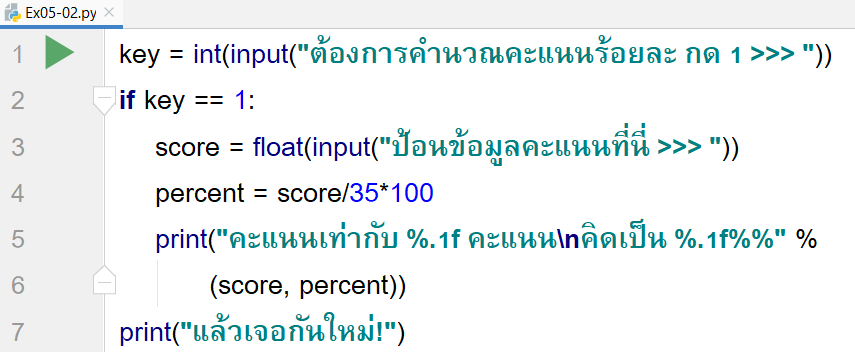
[[python\_05\_06.bmp]]

[[ผลการทำงานของโปรแกรม กรณีกดเลข 1]]

[[python\_05\_07.bmp]]

[[ผลการทำงานของโปรแกรม กรณีไม่ได้กดเลข 1]]

ซึ่งจากผลลัพธ์ของโปรแกรมข้างต้น เราสามารถเขียนโค้ดได้ดังนี้

[[python\_05\_05.bmp]]

จากโปรแกรมที่ 5-2 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 รับข้อมูลตัวเลขและแปลงค่าข้อมูลเป็นชนิด int แล้วเก็บค่าในตัวแปร key

บรรทัดที่ 2 ตรวจสอบค่าข้อมูลในตัวแปร key ว่ามีค่าเท่ากับ 1 หรือไม่

* + - ถ้าผลการตรวจสอบเป็นจริง โปรแกรมจะทำงานที่ชุดคำสั่งในบรรทัดที่ 3-5
    - ถ้าผลการตรวจสอบเป็นเท็จ โปรแกรมจะข้ามการทำงานของชุดคำสั่งในบรรทัดที่ 3-5 ไปทำงานที่คำสั่งในบรรทัด 6

บรรทัดที่ 3 รับข้อมูลคะแนนและแปลงค่าข้อมูลเป็นชนิด float แล้วเก็บค่าในตัวแปร score

บรรทัดที่ 4 คำนวณคะแนนเปอร์เซ็นต์และเก็บผลลัพธ์ในตัวแปร percent

บรรทัดที่ 5 แสดงผลข้อมูลคะแนนและเปอร์เซ็นต์ แต่เนื่องจากข้อมูลเปอร์เซ็นต์ที่ได้จากการคำนวณจะให้ผลเป็นเลขทศนิยมที่มีตำแหน่งทศนิยมไม่ทราบจำนวน จึงใช้รหัสการแสดงผลและส่วนขยายร่วมกับฟังก์ชัน print() เพื่อจัดรูปแบบการแสดงผลให้มีทศนิยม 1 ตำแหน่ง

บรรทัดที่ 6 แสดงผลข้อความ แล้วเจอกันใหม่!

## คำสั่ง if…else : คำสั่งควบคุมให้โปรแกรมเลือกทำงานในทางเลือกใดทางเลือกหนึ่งจาก 2 ทางเลือก

**คำสั่ง if…else** เป็นคำสั่งควบคุมให้โปรแกรมเลือกทำงานในทางเลือกใดทางเลือกหนึ่งจาก 2 ทางเลือก โดยตรวจสอบนิพจน์เงื่อนไขทางตรรกศาสตร์ที่กำหนดว่าเป็นจริงหรือเท็จ ถ้านิพจน์ที่กำหนดให้เป็นจริง (true) โปรแกรมจะทำงานที่ชุดคำสั่งในทางเลือกหลังคำสั่ง if แต่ถ้านิพจน์ที่กำหนดไว้เป็นเท็จ (false) โปรแกรมจะทำงานที่ชุดคำสั่งในทางเลือกหลังคำสั่ง else ซึ่งมีรูปแบบการทำงานดังนี้

if boolean\_expression:

statement\_1

else:

statement\_2

โดยที่ **boolean\_expression** เป็นนิพจน์เงื่อนไขทางตรรกศาสตร์ ซึ่งมีผลการตรวจสอบ

เงื่อนไขเป็นจริง (true) หรือเท็จ (false) เท่านั้น

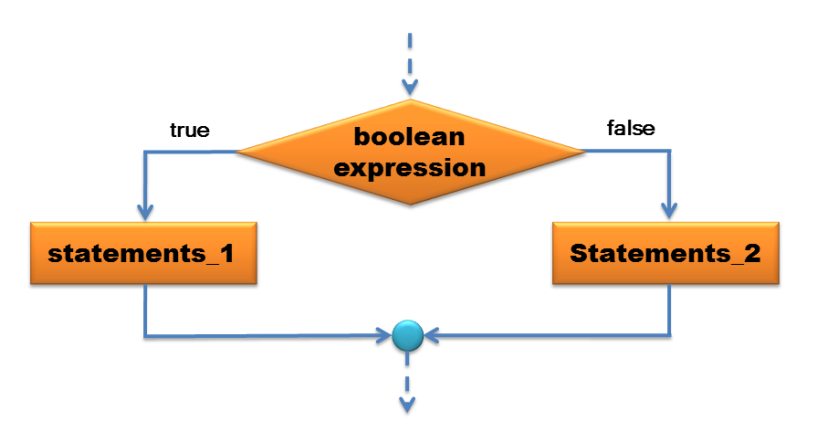
**statements\_1** เป็นชุดของคำสั่งที่จะทำงานเมื่อเงื่อนไขที่กำหนดให้เป็นจริง

(true)

**statements\_2** เป็นชุดของคำสั่งที่จะทำงานเมื่อเงื่อนไขที่กำหนดให้เป็นเท็จ

(false)

จากลักษณะการทำงานของคำสั่งสามารถอธิบายด้วย Flowchart ได้ดังนี้

[[python\_05\_08.bmp]]

ตัวอย่าง เช่น

age = 59

if age > 60:

print("พักผ่อนได้")

else:

print("ทำงานก่อน")

#ผลที่ได้คือ

ทำงานก่อน

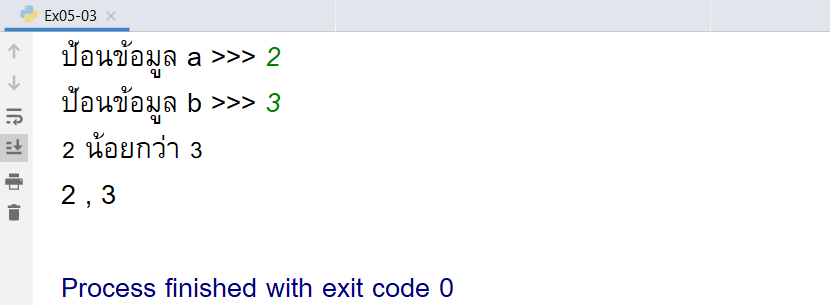
จากตัวอย่างข้างต้น มีการตรวจสอบอายุเกษียณราชการ โดยตรวจสอบค่าข้อมูลของตัวแปร age มากกว่า 60 หรือไม่

* ถ้าผลการตรวจสอบเป็นจริง จะแสดงข้อความว่า “พักผ่อนได้”
* ถ้าผลการตรวจสอบเป็นเท็จ จะแสดงข้อความว่า “ทำงานก่อน”

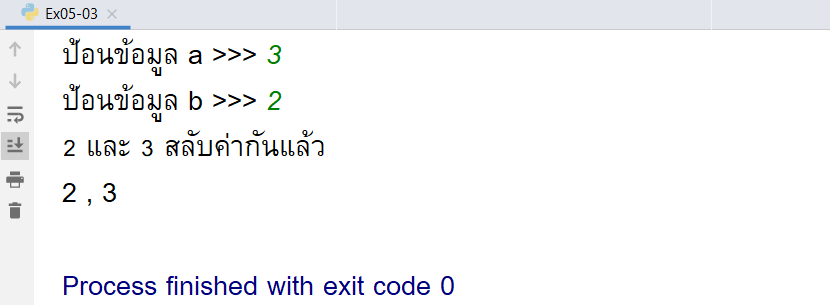
**ตัวอย่างโปรแกรมที่ 5-3** โปรแกรมการใช้คำสั่ง if...else เพื่อตรวจสอบและเรียงลำดับเลข 2 จำนวนจากน้อยไปมาก

ตัวอย่างนี้เป็นโปรแกรมที่สามารถประยุกต์ใช้กับงานที่มีการตัดสินใจเหมือนกับตัวอย่างโปรแกรมที่ 5-2 แต่จะเป็นการใช้คำสั่ง if...else ควบคุมให้เกิดการทำงานใน 2 กรณี โดยเป็นการใช้กรณีของการเรียงลำดับตัวเลข 2 จำนวน

การทำงานของโปรแกรมจึงเริ่มต้นจากการป้อนข้อมูลตัวเลข 2 จำนวน เพื่อให้โปรแกรมนำค่าตัวเลขไปตรวจสอบแล้วทำงานตามเงื่อนไขที่กำหนด และแสดงผลลัพธ์ของโปรแกรม ดังนี้

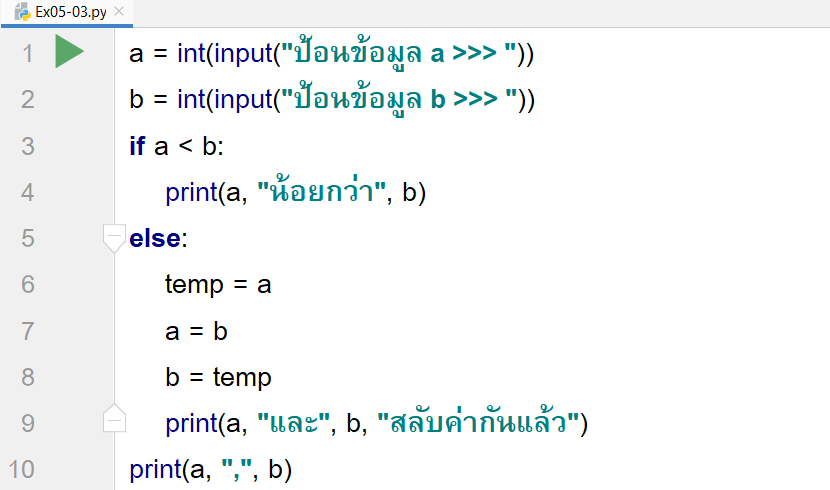
[[python\_05\_10.bmp]]

[[ผลการทำงานของโปรแกรม กรณีค่าในตัวแปร a น้อยกว่า ค่าในตัวแปร b]]

[[python\_05\_11.bmp]]

[[ผลการทำงานของโปรแกรม กรณีค่าในตัวแปร a ไม่น้อยกว่าค่าในตัวแปร b]]

ซึ่งจากผลลัพธ์ของโปรแกรมข้างต้น เราสามารถเขียนโค้ดได้ดังนี้

[[python\_05\_09.bmp]]

จากโปรแกรมที่ 5-3 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1-2 รับข้อมูลตัวเลข 2 ค่า และแปลงค่าข้อมูลเป็นชนิด int แล้วเก็บค่าในตัวแปร a และ b ตามลำดับ

บรรทัดที่ 3 ตรวจสอบค่าข้อมูลในตัวแปร a ว่ามีค่าน้อยกว่าข้อมูลในตัวแปร b หรือไม่

* + - ถ้าผลการตรวจสอบเป็นจริง โปรแกรมจะทำงานที่ชุดคำสั่งในบรรทัดที่ 4
    - ถ้าผลการตรวจสอบเป็นเท็จ โปรแกรมจะข้ามการทำงานของชุดคำสั่งในบรรทัดที่ 4 ไปทำงานที่คำสั่งในบรรทัด 6-9

บรรทัดที่ 4 แสดงผลข้อความ ค่าในตัวแปร a น้อยกว่า ค่าในตัวแปร b

บรรทัดที่ 6 กำหนดค่าตัวแปร temp ให้เท่ากับค่าในตัวแปร a

บรรทัดที่ 7 กำหนดค่าตัวแปร a ให้เท่ากับค่าในตัวแปร b

บรรทัดที่ 8 กำหนดค่าตัวแปร b ให้เท่ากับค่าในตัวแปร a

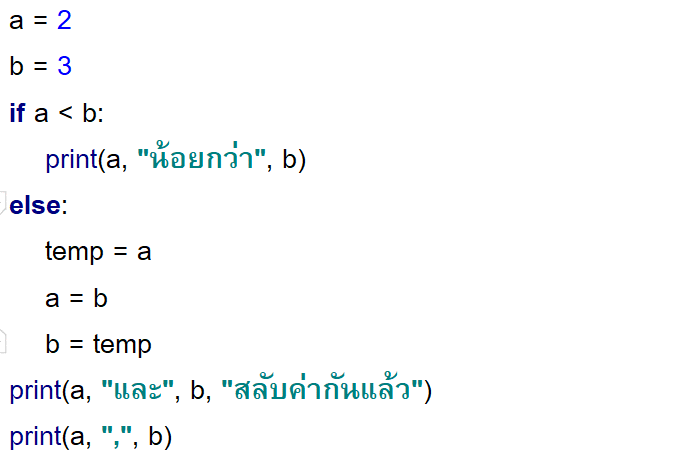
บรรทัดที่ 9 แสดงผลข้อความ ค่าในตัวแปร a และ ค่าในตัวแปร b สลับค่ากันแล้ว

บรรทัดที่ 10 แสดงผลข้อความ ค่าในตัวแปร a เครื่องหมาย , ค่าในตัวแปร b

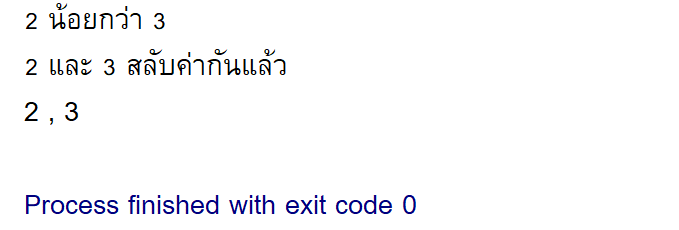
**Warn**

ในกรณีที่มีกลุ่มคำสั่งหลังคำสั่ง if หรือหลังคำสั่ง else มากกว่า 1 คำสั่ง และผู้อ่านไม่ได้จัดวางคำสั่งให้อยู่ในตำแหน่งย่อหน้าที่ตรงกัน จะทำให้โปรแกรมทำงานเฉพาะคำสั่งที่อยู่ในตำแหน่งย่อหน้าที่ตรงกัน เท่านั้น ซึ่งทำให้ผลการทำงานของโปรแกรมผิดพลาดได้

ตัวอย่าง เช่น

[[python\_05\_12.bmp]]

ผลการทำงานของโปรแกรมแสดงได้ดังนี้

[[python\_05\_13.bmp]]

จากตัวอย่างข้างต้น เกิดข้อผิดพลาดตรงคำสั่ง print(a, "และ", b, "สลับค่ากันแล้ว") ซึ่งไม่ได้จัดวางให้อยู่ในตำแหน่งย่อหน้าที่ตรงกับกลุ่มคำสั่งหลังคำสั่ง else ในการแสดงผลการทำงานจึงมีการแสดงผลบรรทัดนี้ออกมาด้วย ดังนั้น **ผู้เขียนแนะนำให้ผู้อ่านพึงระวังในเรื่องของการจัดวางคำสั่งให้อยู่ในตำแหน่งย่อหน้าที่ตรงกัน**

## คำสั่ง if...elif…else : คำสั่งควบคุมให้โปรแกรมเลือกทำงานในทางเลือกใดทางเลือกหนึ่งจากหลายทางเลือก

**คำสั่ง if…elif…else** เป็นคำสั่งควบคุมให้โปรแกรมเลือกทำงานในทางเลือกใดทางเลือกหนึ่งจากหลายทางเลือก และแต่ละทางเลือกจะมีการกำหนดนิพจน์เงื่อนไขทางตรรกศาสตร์ของแต่ละทางเลือกไว้ด้วย โดยโปรแกรมจะตรวจสอบนิพจน์ของแต่ละทางเลือก หากพบว่าทางเลือกไหนมีนิพจน์เป็นจริง (true) ก็จะทำงานที่ชุดคำสั่งภายในทางเลือกนั้น โดยไม่พิจารณาทางเลือกอื่นที่เหลืออีก ซึ่งมีรูปแบบการทำงานดังนี้

if boolean\_expression\_1:

statement\_1

elif boolean\_expression\_2:

statement\_2

.

.

.

else:

statement\_n

โดยที่ **boolean\_expression\_1** เป็นนิพจน์เงื่อนไขทางตรรกศาสตร์ ซึ่งมีผลการตรวจสอบ

เงื่อนไขเป็นจริง (true) หรือเท็จ (false) เท่านั้น

**boolean\_expression\_2** เป็นนิพจน์เงื่อนไขทางตรรกศาสตร์ ซึ่งมีผลการตรวจสอบ

เงื่อนไขเป็นจริง (true) หรือเท็จ (false) เท่านั้น

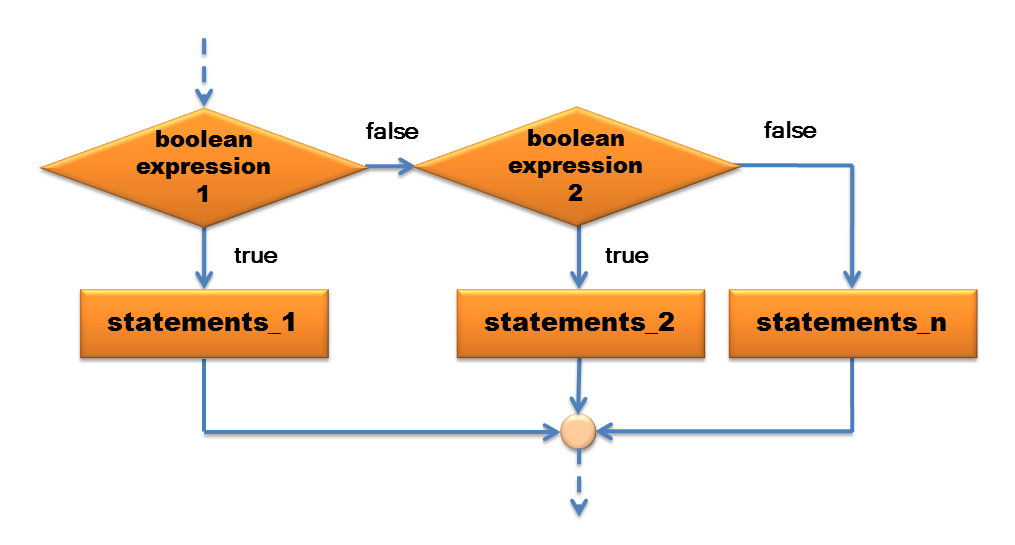
**statements\_1** เป็นชุดของคำสั่งที่จะทำงานเมื่อเงื่อนไข boolean\_expression\_1 ที่กำหนดเป็นจริง (true)

**statements\_2** เป็นชุดของคำสั่งที่จะทำงานเมื่อเงื่อนไข boolean\_expression\_2 ที่กำหนดเป็นจริง (true)

**statements\_n** เป็นชุดของคำสั่งที่จะทำงานเมื่อเงื่อนไขที่กำหนดให้ทั้งหมด

เป็นเท็จ (false)

จากลักษณะการทำงานของคำสั่งสามารถอธิบายด้วย Flowchart ได้ดังนี้

[[python\_05\_14.bmp]]

ตัวอย่าง เช่น

score = 78

if score >= 80:

grade = 'A'

elif score >= 70:

grade = 'B'

elif score >= 60:

grade = 'C'

elif score >= 50:

grade = 'D'

else:

grade = 'F'

print(grade)

#ผลที่ได้คือ

B

จากตัวอย่างข้างต้น มีการตรวจสอบค่าข้อมูลของตัวแปร score ตามเงื่อนไข ดังนี้

* ตรวจสอบว่า ตัวแปร score มีค่ามากกว่าหรือเท่ากับ 80 หรือไม่

ถ้าผลการตรวจสอบเป็นจริง จะได้เกรด A

* ถ้าผลการตรวจสอบเป็นเท็จ จะตรวจสอบว่า ตัวแปร score มีค่ามากกว่าหรือเท่ากับ 70 หรือไม่

ถ้าผลการตรวจสอบเป็นจริง จะได้เกรด B

* ถ้าผลการตรวจสอบเป็นเท็จ จะตรวจสอบว่า ตัวแปร score มีค่ามากกว่าหรือเท่ากับ 60 หรือไม่

ถ้าผลการตรวจสอบเป็นจริง จะได้เกรด C

* ถ้าผลการตรวจสอบเป็นเท็จ จะตรวจสอบว่า ตัวแปร score มีค่ามากกว่าหรือเท่ากับ 50 หรือไม่

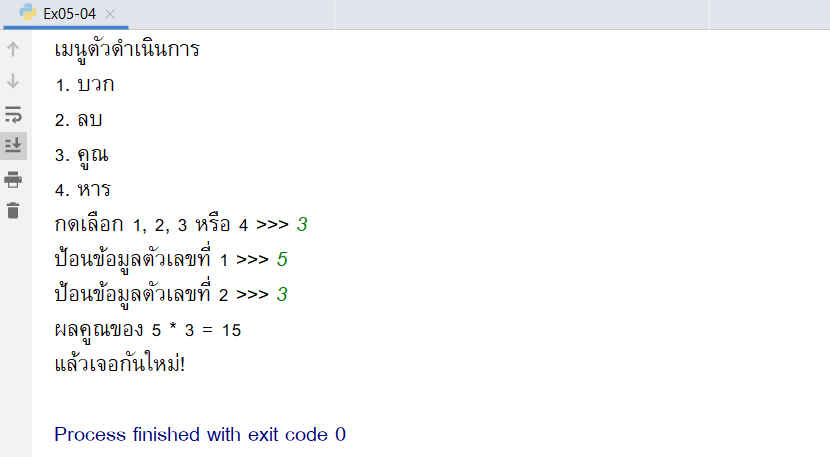
ถ้าผลการตรวจสอบเป็นจริง จะได้เกรด D

* ถ้าผลการตรวจสอบทั้งหมดเป็นเท็จ จะได้เกรด F

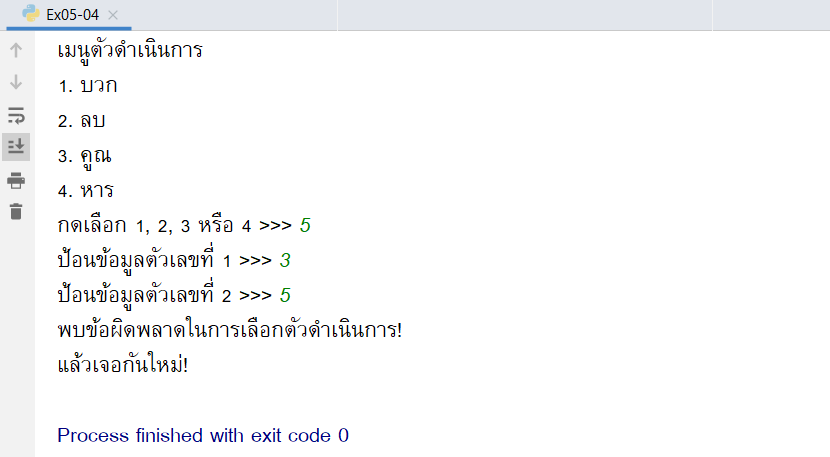
**ตัวอย่างโปรแกรมที่ 5-4** โปรแกรมการใช้คำสั่ง if...elif…else เพื่อตรวจสอบการดำเนินการกับตัวเลข 2 จำนวน ให้ตรงกับตัวดำเนินการที่เลือก

ตัวอย่างนี้เป็นโปรแกรมที่สามารถประยุกต์ใช้กับงานที่มีการตัดสินใจในหลายทางเลือก โดยการใช้คำสั่ง if...elif…else ตรวจสอบการเลือกตัวดำเนินการที่ต้องการให้เกิดการการดำเนินการระหว่างตัวเลข 2 จำนวน

การทำงานของโปรแกรมจึงเริ่มต้นจากการสร้างรายการตัวดำเนินการเป็นเมนูตัวเลขให้เลือก และป้อนข้อมูลตัวเลข 2 จำนวน เพื่อให้โปรแกรมนำค่าตัวเลขไปตรวจสอบ เพื่อเลือกตัวดำเนินการ และทำงานตามตัวดำเนินการดังกล่าวกับตัวเลข 2 จำนวนนั้น และแสดงผลลัพธ์ของโปรแกรม ดังนี้

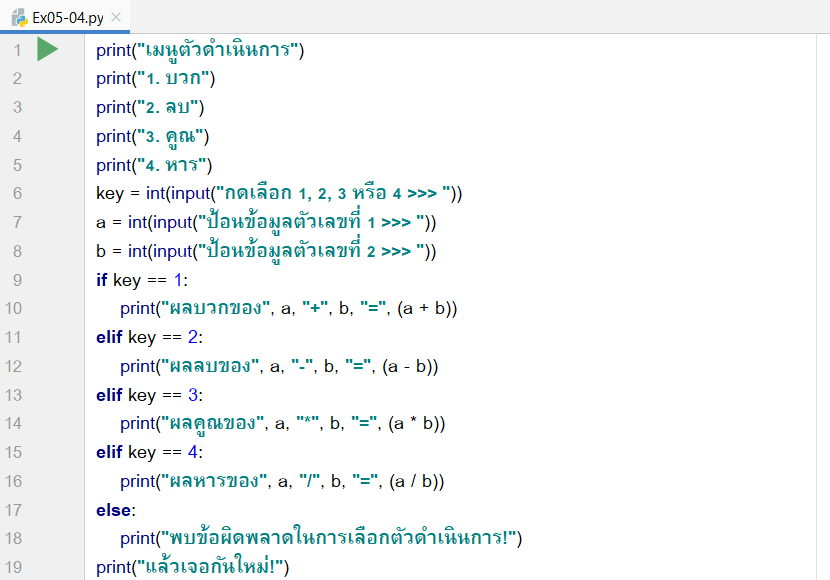
[[python\_05\_16.bmp]]

[[ผลการทำงานของโปรแกรม กรณีค่าในตัวแปร key = 3, a = 5 และ b = 3]]

[[python\_05\_17.bmp]]

[[ผลการทำงานของโปรแกรม กรณีค่าในตัวแปร key = 5, a = 3 และ b = 5]]

ซึ่งจากผลลัพธ์ของโปรแกรมข้างต้น เราสามารถเขียนโค้ดได้ดังนี้

[[python\_05\_15.bmp]]

จากโปรแกรมที่ 5-4 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1-5 แสดงข้อความตัวดำเนินการสำหรับการบวก ลบ คูณ และหาร ตามลำดับ

บรรทัดที่ 6 รับข้อมูลตัวเลขเพื่อเลือกตัวดำเนินการ และแปลงค่าข้อมูลเป็นชนิด int แล้วเก็บค่าใน

ตัวแปร key

บรรทัดที่ 7-8 รับข้อมูลตัวเลข 2 ค่า และแปลงค่าข้อมูลเป็นชนิด int แล้วเก็บค่าในตัวแปร a และ b ตามลำดับ

บรรทัดที่ 9-18 ตรวจสอบค่าข้อมูลในตัวแปร key ว่าเท่ากับตัวเลข 1, 2, 3 หรือ 4 เพื่อทำงานตามชุดคำสั่งที่มีเงื่อนไขค่าตัวเลขเท่ากับตัวแปร key ในกรณีที่ค่าข้อมูลในตัวแปร key ไม่ตรงกับเลขใดเลย โปรแกรมจะทำงานตามชุดคำสั่งหลังคำสั่ง else คือแสดงข้อความว่า “พบข้อผิดพลาดในการเลือกตัวดำเนินการ!”

บรรทัดที่ 19 แสดงข้อความว่า “แล้วเจอกันใหม่!”

# คำสั่งควบคุมแบบทำซ้ำ (Iteration Control Statement)

**คำสั่งควบคุมแบบทำซ้ำ (Iteration Control Statement)** เป็นรูปแบบการทำงานแบบวนลูปที่มีการทำงานซ้ำชุดคำสั่งชุดเดิมไปเรื่อยๆ จนกว่าเงื่อนไขที่กำหนดจะเป็นเท็จ จึงจะออกจากลูปคำสั่ง ซึ่งมี 2 คำสั่ง คือ คำสั่ง while และ คำสั่ง for

## คำสั่ง while : คำสั่งควบคุมแบบทำซ้ำด้วยจำนวนรอบที่ไม่แน่นอน

**คำสั่ง while** เป็นคำสั่งควบคุมการทำงานของโปรแกรมแบบทำซ้ำที่มีการตรวจสอบเงื่อนไขที่เป็นนิพจน์ทางตรรกศาสตร์ก่อนจะทำงานที่ชุดคำสั่งภายในลูป โดยโปรแกรมจะทำงานที่ชุดคำสั่งภายในลูปก็ต่อเมื่อนิพจน์ที่กำหนดให้เป็นจริง และเมื่อโปรแกรมทำงานที่ชุดคำสั่งภายในลูปเสร็จสิ้น โปรแกรมจะตรวจสอบนิพจน์ใหม่อีกครั้ง และจะหยุดการทำซ้ำเมื่อนิพจน์ที่กำหนดให้เป็นเท็จ โดยจำนวนรอบของการทำซ้ำจะไม่แน่นอนขึ้นอยู่กับเงื่อนไขที่ใช้ในการตรวจสอบ และโปรแกรมจะออกจากลูปไปทำงานที่คำสั่งถัดไปทันที ซึ่งมีรูปแบบการทำงานดังนี้

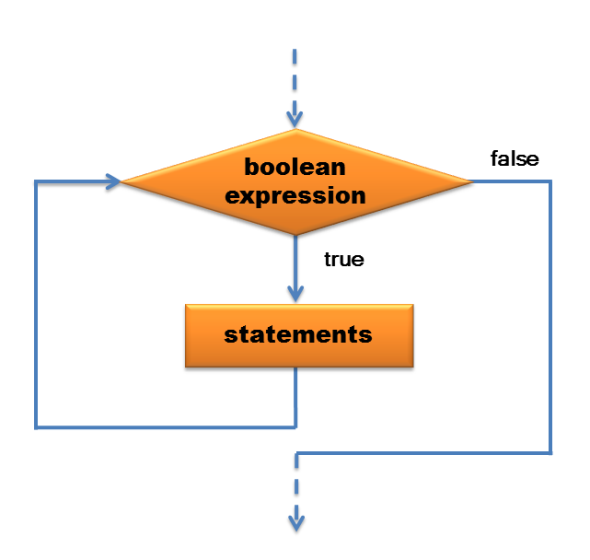
|  |
| --- |
| while bolean\_expression:  statements |

โดยที่ **boolean\_expression** เป็นนิพจน์ที่เป็นเงื่อนไขทางตรรกศาสตร์ ไว้ตรวจสอบ

เงื่อนไขว่าเป็นจริงหรือเท็จก่อนการทำงานเสมอ

**statements** เป็นชุดคำสั่งที่ต้องการให้มีการทำซ้ำ

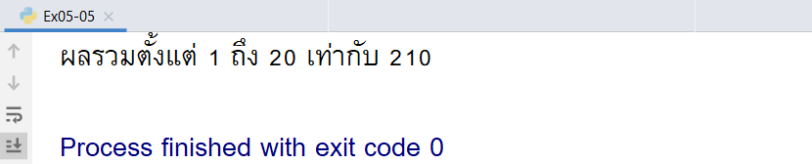
จากลักษณะการทำงานของคำสั่งสามารถอธิบายด้วย Flowchart ได้ดังนี้

[[python\_05\_18.bmp]]

**ตัวอย่างโปรแกรมที่ 5-5** โปรแกรมการใช้คำสั่ง while เพื่อคำนวณผลรวมตั้งแต่ 1 ถึง 20

ตัวอย่างนี้เป็นโปรแกรมที่สามารถประยุกต์ใช้กับงานที่มีการทำซ้ำ ๆ โดยการใช้คำสั่ง while ตรวจสอบเงื่อนไขของค่าตัวเลขที่ใช้การคำนวณผลรวมซ้ำ ๆ กัน ตั้งแต่ 1 ถึง 20

การทำงานของโปรแกรมจึงเริ่มต้นจากการการกำหนดค่าเริ่มต้นของการทำซ้ำ คำนวณผลรวมซ้ำไปเรื่อย ๆ จนถึงค่าสิ้นสุด และแสดงผลลัพธ์ของโปรแกรม ดังนี้

[[python\_05\_20.bmp]]

ซึ่งจากผลลัพธ์ของโปรแกรมข้างต้น เราสามารถเขียนโค้ดได้ดังนี้

|  |
| --- |
| [[python\_05\_19.bmp]] |

จากโปรแกรมที่ 5-5 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1-2 ประกาศตัวแปร count เป็นข้อมูลชนิด int มีค่าเท่ากับ 1 (ใช้เป็นตัวนับ) และตัวแปร total เป็นข้อมูลชนิด int มีค่าเท่ากับ 0 (ใช้เก็บค่าผลรวม)

บรรทัดที่ 3-5 วนลูปด้วยคำสั่ง while โดยกำหนดนิพจน์เงื่อนไขให้ตรวจสอบว่าค่าตัวแปร count มีค่าน้อยกว่าหรือเท่ากับ 20 หรือไม่

* ถ้าผลการตรวจสอบเป็นจริง ให้ทำงานที่ชุดคำสั่งทำบรรทัดที่ 4-5
* ถ้าผลการตรวจสอบเป็นเท็จ ให้ทำงานที่คำสั่งทำบรรทัดที่ 6

บรรทัดที่ 4 นำค่าของ count บวกด้วยค่าของ total และเก็บผลรวมในตัวแปร total

บรรทัดที่ 5 เพิ่มค่าตัวแปร count ขึ้น 1 ค่า

บรรทัดที่ 6 แสดงค่าผลรวมที่เก็บในตัวแปร total

จากโค้ดโปรแกรมพบว่า ลูปคำสั่ง while จะทำงานจนกว่าค่าตัวแปร count จะมีค่ามากกว่า 20 จึงจะหยุดการทำซ้ำและออกจากลูปคำสั่ง while

## คำสั่ง while in range : คำสั่งควบคุมแบบทำซ้ำด้วยจำนวนรอบที่แน่นอน

นอกจากการวนลูปด้วยการเพิ่มค่าให้ตัวแปร count จนครบจำนวนลูปที่ต้องการแล้ว ผู้อ่านไม่จำเป็นต้องทราบจำนวนลูปที่ต้องการทำซ้ำ แต่ในกรณีที่ผู้อ่านทราบจำนวนลูปที่ต้องการทำซ้ำ ในภาษา Python มีฟังก์ชัน range() ที่สามารถใช้ร่วมกับคำสั่ง while เพื่อกำหนดช่วงข้อมูลที่เป็นตัวเลขสำหรับการวนลูป ซึ่งมีรูปแบบการทำงานดังนี้

|  |
| --- |
| while count in range(value):  statements |

โดยที่ **count** ตัวแปรที่ใช้ในการวนลูป

**value** ช่วงข้อมูลที่เป็นตัวเลขสำหรับการทำซ้ำที่ต้องการ ประกอบด้วยข้อมูล

ตัวเลข 2 ค่าคือ ค่าเริ่มต้น และ ค่าสุดท้ายของการทำซ้ำ + 1

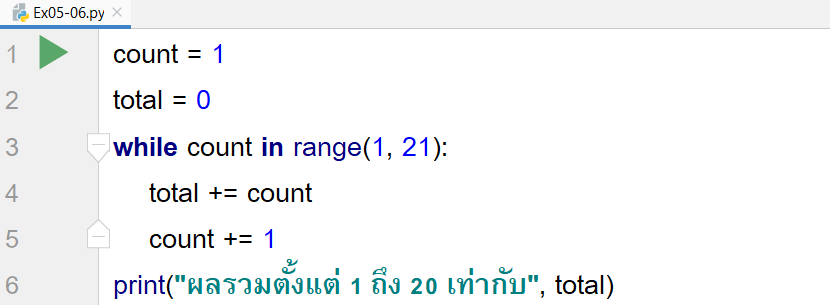
**statements** เป็นชุดคำสั่งที่ต้องการให้มีการทำซ้ำ

ขั้นตอนการตรวจสอบคือ จะทำการตรวจสอบว่า ค่า count อยู่ในช่วงข้อมูลของ format ในฟังก์ชัน range หรือไม่ ถ้ามีให้ทำงานตาม statements ถ้าไม่มีก็จะไม่ทำงานตาม statements และออกจากคำสั่งลูป

**ตัวอย่างโปรแกรมที่ 5-6** โปรแกรมการใช้คำสั่ง while in range เพื่อคำนวณผลรวมตั้งแต่ 1 ถึง 20

ตัวอย่างนี้เป็นโปรแกรมที่ทำงานเหมือนตัวอย่างโปรแกรมที่ 5-5 แต่จะเป็นการใช้คำสั่ง while in range แทนการตรวจสอบเงื่อนไขการทำซ้ำ

การทำงานของโปรแกรมจึงเริ่มต้นจากการการกำหนดค่าเริ่มต้นของการทำซ้ำ คำนวณผลรวมซ้ำไปเรื่อย ๆ จนถึงค่าสิ้นสุด และแสดงผลลัพธ์ของโปรแกรม ดังนี้

[[python\_05\_21.bmp]]

จากโปรแกรมที่ 5-6 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1-2 กำหนดตัวแปร count ใช้เป็นตัวนับ ให้มีค่าเท่ากับ 1 และตัวแปร total ใช้เก็บค่าผลรวม ให้มีค่าเท่ากับ 0

บรรทัดที่ 3-5 ใช้คำสั่ง while in range วนลูป โดยตรวจสอบค่าตัวแปร count อยู่ในช่วง 1 ถึง 20 หรือไม่

* ถ้าผลการตรวจสอบเป็นจริง ให้ทำงานตามคำสั่งบรรทัดที่ 4-5 เมื่อทำงานเสร็จ

ให้ตรวจสอบใหม่อีกครั้ง

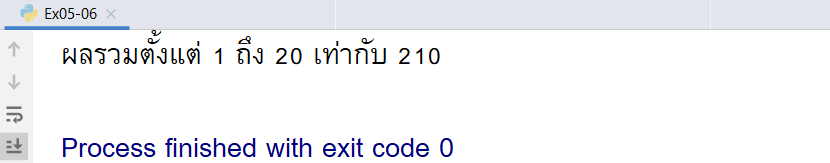
* ถ้าผลการตรวจสอบเป็นเท็จ ให้ข้ามไปทำงานตามคำสั่งบรรทัดที่ 6

บรรทัดที่ 4 นำค่าของตัวแปร count บวกด้วยค่าของตัวแปร total และเก็บผลในตัวแปร total

บรรทัดที่ 5 เพิ่มค่าตัวแปร count ขึ้น 1 ค่า

บรรทัดที่ 6 แสดงผลค่าผลรวมที่เก็บในตัวแปร total

ผลการทำงานของโปรแกรมแสดงได้ดังนี้

[[python\_05\_22.bmp]]

## คำสั่ง while True : คำสั่งควบคุมแบบทำซ้ำโดยทำอย่างน้อย 1 รอบ

โดยทั่วไปการทำงานของคำสั่ง while จะตรวจสอบเงื่อนไขก่อน เมื่อเงื่อนไขเป็นจริงโปรแกรมก็จะ ทำงานตามชุดคำสั่งภายในลูป แต่ในบางครั้งอาจต้องการให้โปรแกรมทำงานตามชุดคำสั่งภายในลูปก่อนการตรวจสอบเงื่อนไข เช่น การรับข้อมูลซ้ำๆ จนกว่าข้อมูลจะถูกต้อง นั่นหมายความว่า ผู้อ่านจะต้องรับข้อมูลก่อน แล้วนำข้อมูลนั้นไปตรวจสอบว่าถูกต้องหรือไม่ หากไม่ถูกต้องก็จะต้องรับข้อมูลใหม่ไปเรื่อยๆ ในกรณึเช่นนี้ผู้อ่านสามารถใช้คำสั่ง while ในการจัดการ ซึ่งมีรูปแบบการทำงานดังนี้

|  |
| --- |
| while True:  data = input(“Enter data >>> ”)  if data == checkvalue:  break |

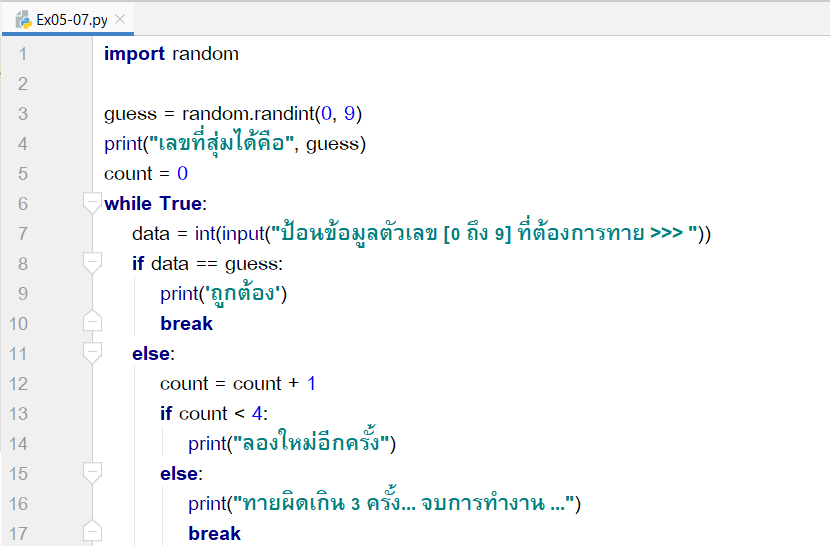
โดยที่ **data** ข้อมูลที่รับเข้ามาทางแป้นพิมพ์

**checkvalue** ข้อมูลที่ต้องการตรวจสอบ

**statements** เป็นชุดคำสั่งที่ต้องการให้มีการทำซ้ำ

ขั้นตอนการตรวจสอบคือ จะตรวจสอบว่ามีค่า data เท่ากับค่า checkvalue หรือไม่ ถ้าไม่ใช่จะวนลูปเพื่อรับข้อมูล data ใหม่ ถ้าใช่ก็จะ break การทำงาน และออกจากคำสั่งลูป

**ตัวอย่างโปรแกรมที่ 5-7** โปรแกรมการใช้คำสั่ง while True เพื่อทายตัวเลขที่สุ่มจากฟังก์ชัน random() โดยมีโค้ดดังนี้

[[python\_05\_23.bmp]]

จากโปรแกรมที่ 5-7 อธิบายการทำงานของโปรแกรมได้ดังนี้

บรรทัดที่ 1 import random เพื่อให้สามารถใช้ฟังก์ชัน random() ได้

บรรทัดที่ 3 ใช้ฟังก์ชัน random() สุ่มตัวเลขจำนวนเต็ม 1 ค่า ที่อยู่ระหว่าง 0 ถึง 9 แล้วเก็บค่าในตัวแปร guess

บรรทัดที่ 4 แสดงผลค่าตัวเลขที่สุ่มได้

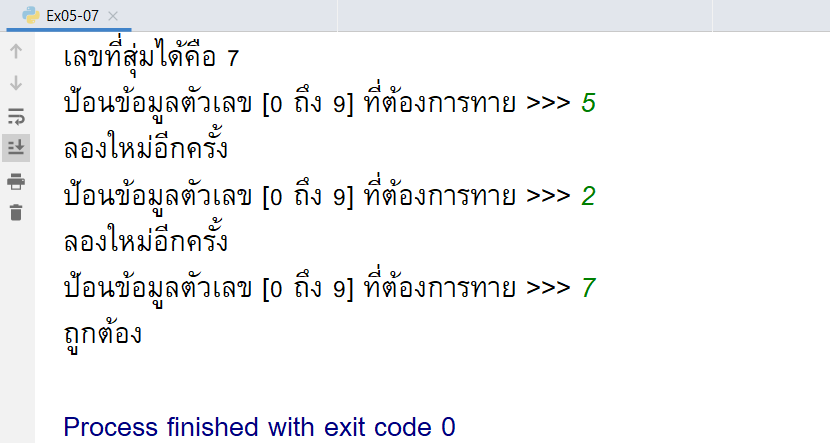
บรรทัดที่ 5 กำหนดค่าให้ตัวแปร count เท่ากับ 0

บรรทัดที่ 6 วนลูปด้วยคำสั่ง while True เพื่อให้โปรแกรมทำงานตามชุดคำสั่งภายในลูปบรรทัด 7-17บรรทัดที่ 7 รับข้อมูลตัวเลขที่ต้องการทายและแปลงค่าข้อมูลเป็นชนิด int แล้วเก็บค่าในตัวแปร data

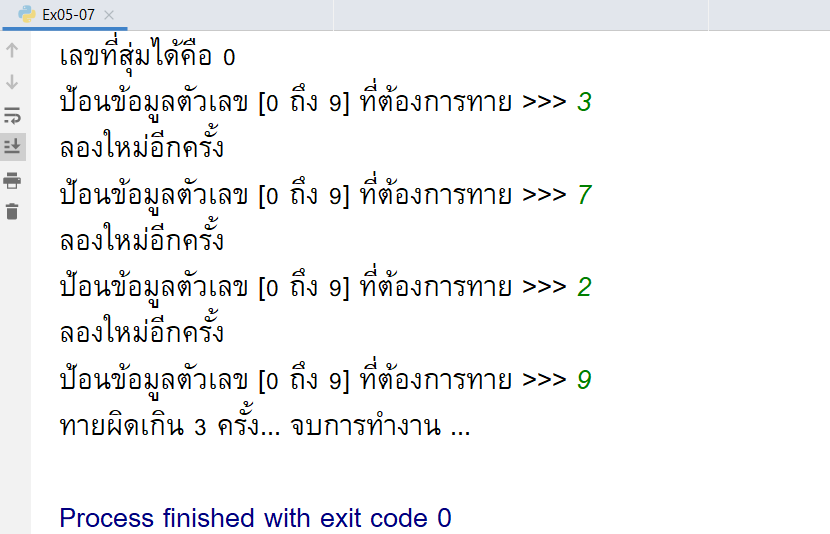
บรรทัดที่ 8-10 ถ้า data มีค่าเท่ากับ guess โปรแกรมก็จะจะแสดงผลข้อความ “ถูกต้อง” แล้ว break การทำงาน และออกจากคำสั่งลูป

บรรทัดที่ 11-17 ถ้า data มีค่าไม่เท่ากับ guess จะบวกค่า 1 ให้กับตัวแปร count จำนวนครั้งของการทายผิด ถ้า count มีค่าน้อยกว่า 4 จะแสดงผลข้อความ ลองใหม่อีกครั้ง เพื่อวนลูปรับข้อมูล data ใหม่ แต่ถ้า count มีค่าตั้งแต่ 4 คือทายเกิน 3 ครั้ง จะแสดงผลข้อความ ทายผิดเกิน 3 ครั้ง... จบการทำงาน ...

ผลการทำงานของโปรแกรมแสดงได้ดังนี้

[[python\_05\_24.bmp]]

[[ผลการทำงานของโปรแกรม กรณีทายถูก]]

[[python\_05\_25.bmp]]

[[ผลการทำงานของโปรแกรม กรณีทายผิด]]

## คำสั่ง while else : คำสั่งควบคุมส่วนที่อยู่นอกเหนือการทำซ้ำ

นอกจากการใช้งานคำสั่ง while ตามที่ได้กล่าวมาแล้วนั้น ภาษา Python ยังอนุญาตให้ผู้อ่านใช้ else ร่วมกับคำสั่ง while เพื่อให้ดำเนินการกับชุดคำสั่งที่ไม่ตรงตามเงื่อนไขของ while ได้ ซึ่งมีรูปแบบการทำงานดังนี้

|  |
| --- |
| while boolean\_expression:  statement\_1  else:  statement\_2 |

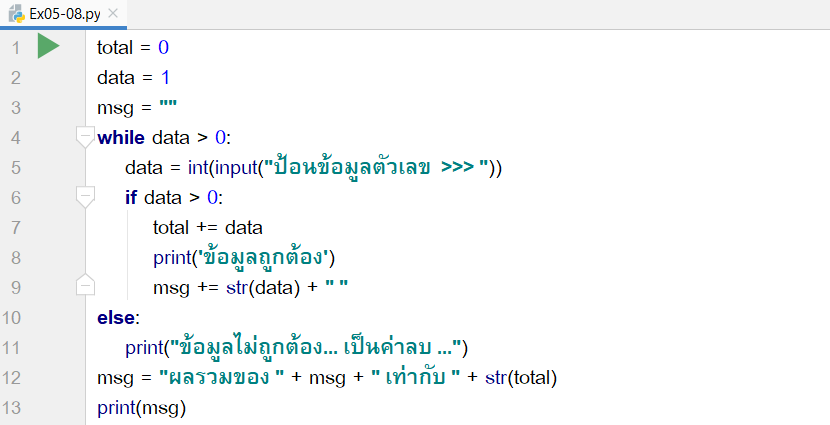
โดยที่ **boolean\_expression** เป็นนิพจน์ที่เป็นเงื่อนไขทางตรรกศาสตร์ โดยตรวจสอบ

เงื่อนไขว่าเป็นจริงหรือเท็จก่อนการทำงานเสมอ

**statement\_1** เป็นชุดคำสั่งที่ต้องการให้มีการทำซ้ำ

**statement\_2** เป็นชุดคำสั่งที่ต้องการทำ ในกรณีที่เงื่อนไขเป็นเท็จ

**ตัวอย่างโปรแกรมที่ 5-8** โปรแกรมการใช้คำสั่ง while else เพื่อคำนวณผลรวมของตัวเลขที่รับเข้ามาทางแป้นพิมพ์ โดยเป็นตัวเลขที่ไม่ใช่ค่าลบ ซึ่งมีโค้ดดังนี้

[[python\_05\_26.bmp]]

จากโปรแกรมที่ 5-8 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 import random เพื่อให้สามารถใช้ฟังก์ชัน random()

บรรทัดที่ 3 ใช้ฟังก์ชัน random() สุ่มตัวเลขจำนวนเต็ม 1 ค่า ที่อยู่ระหว่าง 0 ถึง 9 แล้วเก็บค่าในตัวแปร guess

บรรทัดที่ 4 แสดงผลค่าตัวเลขที่สุ่มได้

บรรทัดที่ 5 กำหนดค่าให้ตัวแปร count เท่ากับ 0

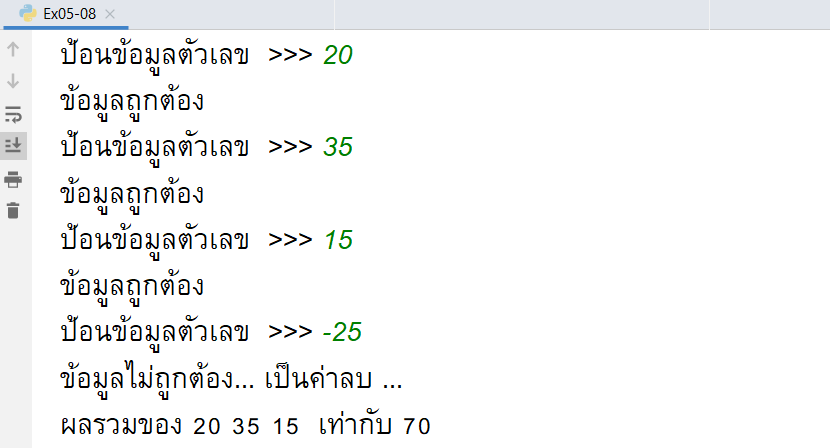
บรรทัดที่ 6 วนลูปด้วยคำสั่ง while True เพื่อให้โปรแกรมทำงานตามชุดคำสั่งภายในลูปบรรทัด 7-17บรรทัดที่ 7 รับข้อมูลตัวเลขที่ต้องการทายและแปลงค่าข้อมูลเป็นชนิด int แล้วเก็บค่าในตัวแปร data

บรรทัดที่ 8-10 ถ้า data มีค่าเท่ากับ guess โปรแกรมก็จะแสดงผลข้อความ “ถูกต้อง” แล้ว break การ

ทำงาน และออกจากคำสั่งลูป

บรรทัดที่ 11-17 ถ้า data มีค่าไม่เท่ากับ guess จะบวกค่า 1 ให้กับตัวแปร count จำนวนครั้งของการทายผิด ถ้า count มีค่าน้อยกว่า 4 จะแสดงผลข้อความ “ลองใหม่อีกครั้ง” เพื่อวนลูปรับข้อมูล data ใหม่ แต่ถ้า count มีค่าตั้งแต่ 4 คือทายเกิน 3 ครั้ง จะแสดงผลข้อความ “ทายผิดเกิน 3 ครั้ง... จบการทำงาน ...”

ผลการทำงานของโปรแกรมแสดงได้ดังนี้

[[python\_05\_27.bmp]]

## คำสั่ง for : คำสั่งควบคุมแบบทำซ้ำด้วยจำนวนรอบที่แน่นอน

**คำสั่ง for** เป็นคำสั่งควบคุมการทำงานของโปรแกรมแบบทำซ้ำที่มีการตรวจสอบเงื่อนไขที่เป็นนิพจน์ทางตรรกศาสตร์ก่อนการทำงานที่ชุดคำสั่งภายในลูปทุกครั้ง ซึ่งเป็นการตรวจสอบกับช่วงข้อมูลที่ได้กำหนดไว้ โดยโปรแกรมจะทำงานที่ชุดคำสั่งภายในลูปเมื่อเงื่อนไขเป็นจริง ซึ่งจะเพิ่มหรือลดค่าตัวแปรควบคุมตามคำสั่งที่กำหนดไว้เมื่อทำงานที่ชุดคำสั่งภายในลูปเสร็จสิ้น แล้วกลับมาตรวจสอบนิพจน์เงื่อนไขอีกครั้ง และโปรแกรมจะหยุดทำซ้ำที่ชุดคำสั่งภายในลูปเมื่อนิพจน์เงื่อนไขเป็นเท็จ ซึ่งมีรูปแบบการทำงานดังนี้

|  |
| --- |
| for count in range(value):  statements |

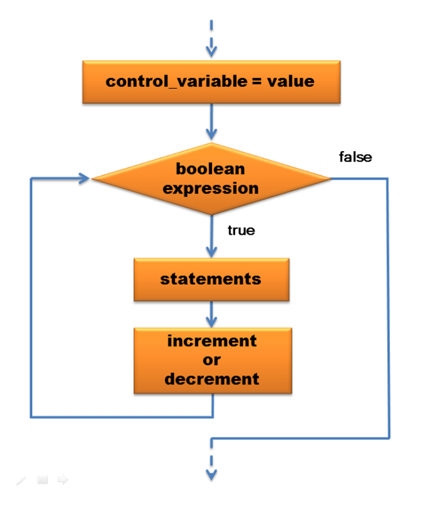
โดยที่ **count** ตัวแปรที่ใช้ในการวนลูป

**value** ช่วงข้อมูลที่เป็นตัวเลขสำหรับการทำซ้ำที่ต้องการ ประกอบด้วยข้อมูล

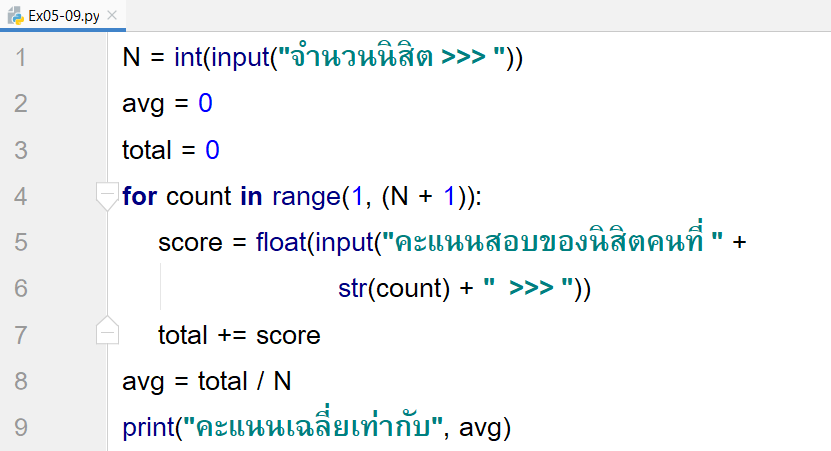
ตัวเลข 2 ค่า คือ ค่าเริ่มต้น และค่าสุดท้ายของการทำซ้ำ + 1

**statements** เป็นชุดคำสั่งที่ต้องการให้มีการทำซ้ำ

จากลักษณะการทำงานของคำสั่งสามารถอธิบายด้วย Flowchart ได้ดังนี้

[[python\_05\_28.bmp]]

**ตัวอย่างโปรแกรมที่ 5-9** โปรแกรมการใช้คำสั่ง for เพื่อคำนวณคะแนนเฉลี่ยของนักศึกษา จากการรับข้อมูลคะแนนนักศึกษาจำนวน N คน มีโค้ดดังนี้

[[python\_05\_29.bmp]]

จากโปรแกรมที่ 5-9 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 รับข้อมูลจำนวนนิสิตด้วยฟังก์ชัน input() และแปลงค่าข้อมูลเป็นชนิด int แล้วเก็บค่าใน

ตัวแปร N

บรรทัดที่ 2 กำหนดตัวแปร avg ให้มีค่าเท่ากับ 0 ใช้เก็บค่าคะแนนเฉลี่ย

บรรทัดที่ 3 กำหนดตัวแปร total ให้มีค่าเท่ากับ 0 ใช้เก็บค่าผลรวมของคะแนนนิสิตทุกคน

บรรทัดที่ 4-6 ใช้คำสั่ง for วนลูป โดยตรวจสอบค่าตัวแปร count อยู่ในช่วง 1 ถึง N หรือไม่

* ถ้าผลการตรวจสอบเป็นจริง ให้ทำงานตามคำสั่งบรรทัดที่ 5-6 เมื่อทำงานเสร็จ

ให้ตรวจสอบใหม่อีกครั้ง

* ถ้าผลการตรวจสอบเป็นเท็จ ให้ข้ามไปทำงานตามคำสั่งบรรทัดที่ 7

บรรทัดที่ 5 รับข้อมูลคะแนนสอบของนิสิตด้วยฟังก์ชัน input() และแปลงค่าข้อมูลเป็นชนิด float แล้วเก็บค่าในตัวแปร score

บรรทัดที่ 6 นำค่าของตัวแปร score บวกกับค่าของตัวแปร total และเก็บค่าในตัวแปร total

บรรทัดที่ 7 นำผลรวมของคะแนนนักศึกษาทุกคนมาคำนวณหาคะแนนเฉลี่ยเก็บผลในตัวแปร avg

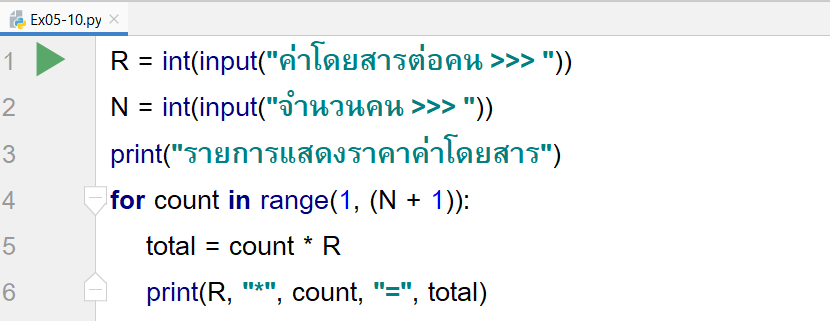
บรรทัดที่ 8 แสดงผลค่าคะแนนเฉลี่ยที่เก็บในตัวแปร avg

ผลการทำงานของโปรแกรมแสดงได้ดังนี้

[[python\_05\_30.bmp]]

จากโค้ดโปรแกรมพบว่า โปรแกรมจะทำงานที่ชุดคำสั่งในบรรทัดที่ 5-6 ซ้ำไปเรื่อยๆ จนกว่าค่าตัวแปร count จะมีค่ามากกว่า N จึงจะหยุดการทำซ้ำและออกจากคำสั่ง for

**ตัวอย่างโปรแกรมที่ 5-10** โปรแกรมการใช้คำสั่ง for เพื่อแสดงค่าโดยสาร จากการรับข้อมูลค่าโดยสารต่อคนและจำนวนคน มีโค้ดดังนี้

 [[python\_05\_31.bmp]]

จากโปรแกรมที่ 5-10 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 รับข้อมูลค่าโดยสารต่อคนด้วยฟังก์ชัน input() และแปลงค่าข้อมูลเป็นชนิด int แล้วเก็บค่าในตัวแปร R

บรรทัดที่ 2 รับข้อมูลจำนวนคนด้วยฟังก์ชัน input() และแปลงค่าข้อมูลเป็นชนิด int แล้วเก็บค่าใน

ตัวแปร N

บรรทัดที่ 3 แสดงผลข้อความ รายการแสดงราคาค่าโดยสาร

บรรทัดที่ 4-6 ใช้คำสั่ง for วนลูป โดยตรวจสอบค่าตัวแปร count อยู่ในช่วง 1 ถึง N หรือไม่

* ถ้าผลการตรวจสอบเป็นจริง ให้ทำงานตามคำสั่งบรรทัดที่ 5-6 เมื่อทำงานเสร็จ

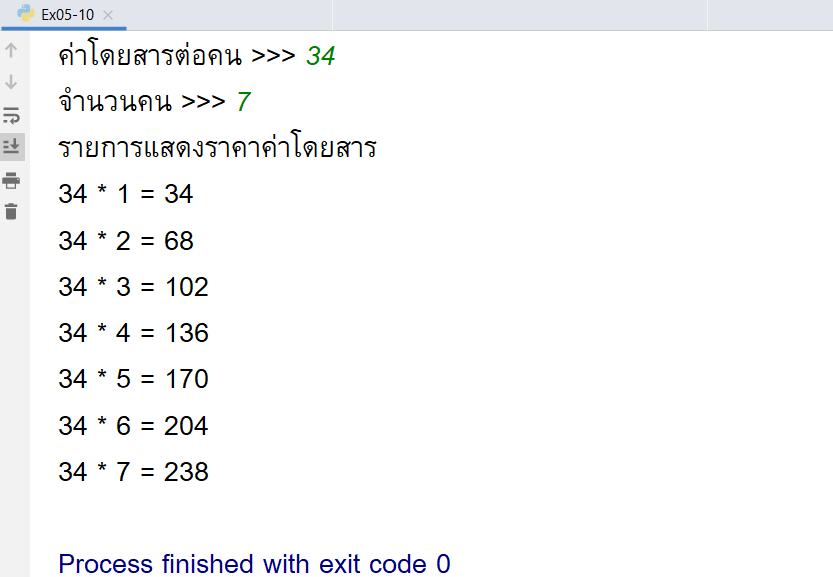
ให้ตรวจสอบใหม่อีกครั้ง

* ถ้าผลการตรวจสอบเป็นเท็จ ให้จบการทำงาน

บรรทัดที่ 5 นำค่าของตัวแปร count คูณกับค่าของตัวแปร R และเก็บผลในตัวแปร total

บรรทัดที่ 6 แสดงผลค่าโดยสารต่อคน จำนวนคน และค่าโดยสารรวม ที่เก็บในตัวแปร R, count และ total ตามลำดับ

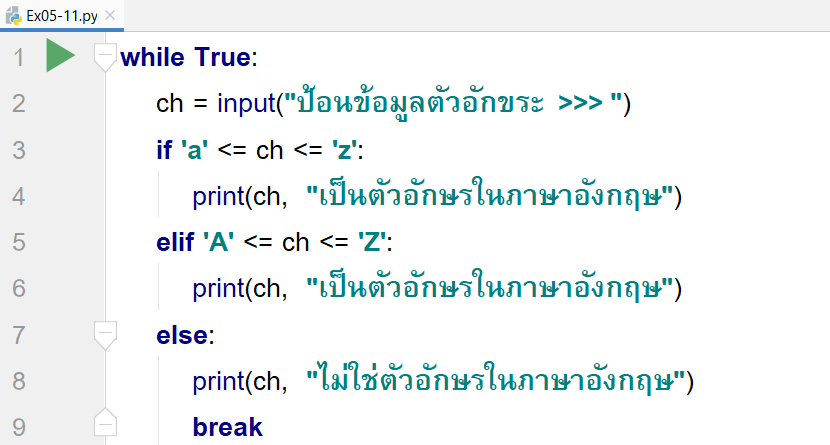
ผลการทำงานของโปรแกรมแสดงได้ดังนี้

[[python\_05\_32.bmp]]

# คำสั่ง break : คำสั่งออกจากการทำงานในลูปทันที

**คำสั่ง break** เป็นคำสั่งที่ใช้ร่วมกับคำสั่งการทำซ้ำ เพื่อหยุดการทำงานก่อนครบตามจำนวนรอบหรือตามเงื่อนไขที่กำหนด โดยไม่สนใจคำสั่งอื่นๆ ที่เหลืออยู่ภายในบล็อกของการทำซ้ำ เพื่อความเข้าใจให้ผู้อ่านศึกษาตัวอย่างต่อไปนี้

**ตัวอย่างโปรแกรมที่ 5-11** โปรแกรมการใช้คำสั่ง break เพื่อรับข้อมูลและแสดงผลตัวอักษรภาษาอังกฤษ มีโค้ดดังนี้

[[python\_05\_33.bmp]]

จากโปรแกรมที่ 5-11 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 วนลูปด้วยคำสั่ง while True เพื่อให้โปรแกรมทำงานตามชุดคำสั่งภายในลูปบรรทัด 2-9

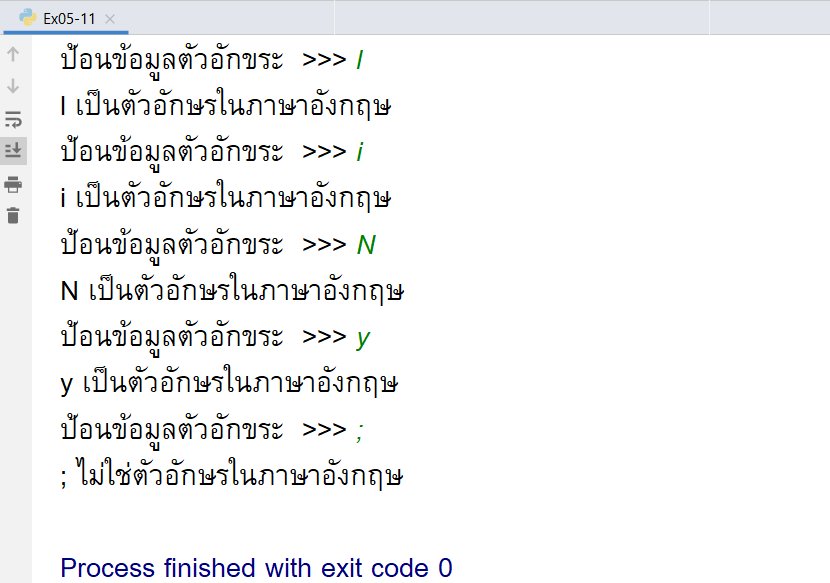
บรรทัดที่ 2 รับข้อมูลตัวอักขระด้วยฟังก์ชัน input() แล้วเก็บค่าในตัวแปร ch

บรรทัดที่ 3-9 ถ้า ch มีค่าเท่ากับตัวอักษรในภาษาอังกฤษทั้งตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ โปรแกรมก็

จะแสดงผลข้อความ ch เป็นตัวอักษรในภาษาอังกฤษ และรับข้อมูล ch ใหม่ แต่ถ้าไม่ใช่

โปรแกรมก็จะแสดงผลข้อความ ch ไม่ใช่ตัวอักษรในภาษาอังกฤษ แล้ว break การทำงาน และออกจากคำสั่งลูป

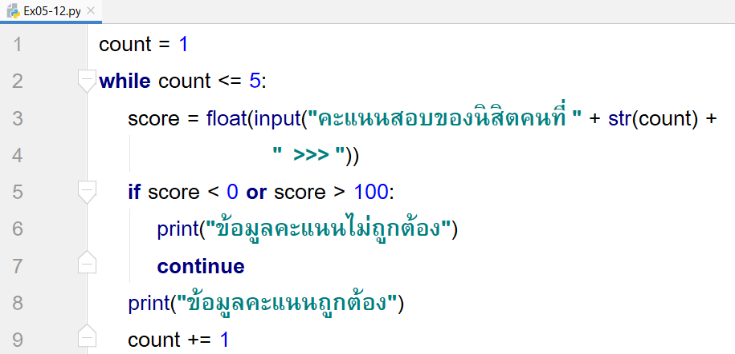
ผลการทำงานของโปรแกรมแสดงได้ดังนี้

[[python\_05\_34.bmp]]

# คำสั่ง continue : คำสั่งบังคับให้ข้ามไปทำงานรอบต่อไปทันที

**คำสั่ง continue** เป็นคำสั่งที่ทำงานตรงข้ามกับคำสั่ง break โดย คำสั่ง continue จะเป็นคำสั่งที่บังคับให้โปรแกรมข้ามไปทำงานในรอบต่อไปทันที โดยไม่สนใจคำสั่งที่เหลืออยู่ในลูป เพื่อความเข้าใจให้ผู้อ่านศึกษาตัวอย่างต่อไปนี้

**ตัวอย่างโปรแกรมที่ 5-12** โปรแกรมการใช้คำสั่ง continue เพื่อรับข้อมูลคะแนนของนักศึกษาจำนวน 5 คน ถ้าคะแนนน้อยกว่า 0 หรือ มากกว่า 100 ให้รับข้อมูลคะแนนต่อไปจนกว่าจะครบ โดยมีโค้ดดังนี้

[[python\_05\_35.bmp]]

จากโปรแกรมที่ 5-12 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 กำหนดตัวแปร count มีค่าเท่ากับ 1 ใช้เป็นตัวแปรควบคุมจำนวนครั้งของการทำซ้ำ

บรรทัดที่ 2-8 เป็นคำสั่งทำซ้ำด้วยคำสั่ง while โดยทำซ้ำจนกว่าค่าตัวแปร count จะมีค่ามากกว่า 5 จึงจะออกจากลูปคำสั่ง

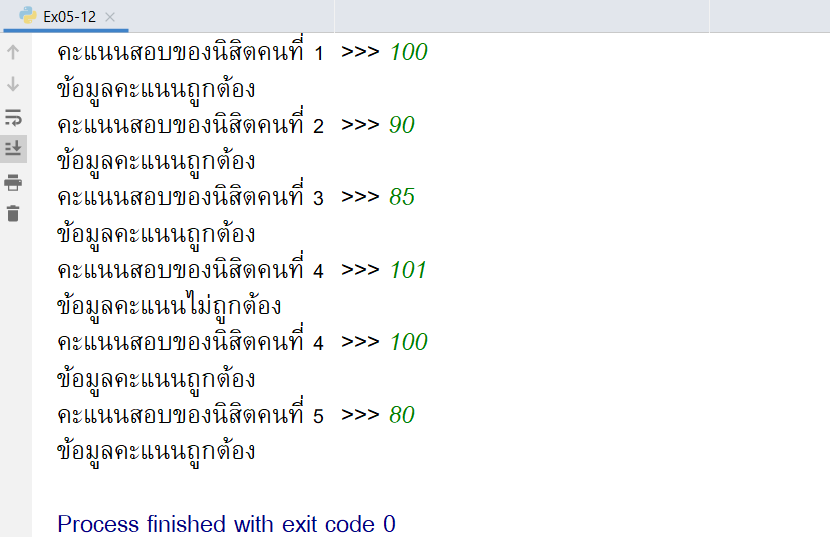
บรรทัดที่ 3 เป็นการตรวจสอบเงื่อนไขของคำสั่ง ถ้าค่าของตัวแปร score ที่รับเข้ามา มีค่าน้อยกว่า 0 หรือมากกว่า 100 จริง ให้ทำงานที่คำสั่งในบรรทัดที่ 5-6

บรรทัดที่ 5-6 ในกรณีที่นิพจน์เงื่อนไขคำสั่งในบรรทัดที่ 4 เป็นจริง โปรแกรมจะข้ามคำสั่งที่เหลือทั้งหมดไปทำงานในรอบถัดไป ด้วยคำสั่ง continue ทันที โดยไม่สนใจว่าคำสั่งที่เหลือเป็นคำสั่งอะไร

ในกรณีที่นิพจน์เงื่อนไขคำสั่งในบรรทัดที่ 4 เป็นเท็จ โปรแกรมจะข้ามไปทำคำสั่งบรรทัดที่ 7-8

บรรทัดที่ 7-8 แสดงข้อความว่า ข้อมูลคะแนนถูกต้อง และบวกค่า count เพิ่มขึ้น 1 ค่า แล้วกลับไปตรวจสอบเงื่อนไขที่บรรทัดที่ 2 จนกว่าค่าตัวแปร count จะมีค่ามากกว่า 5 ก็จะจบการทำงาน

ผลการทำงานของโปรแกรมแสดงได้ดังนี้

[[python\_05\_36.bmp]]

**Tip:**

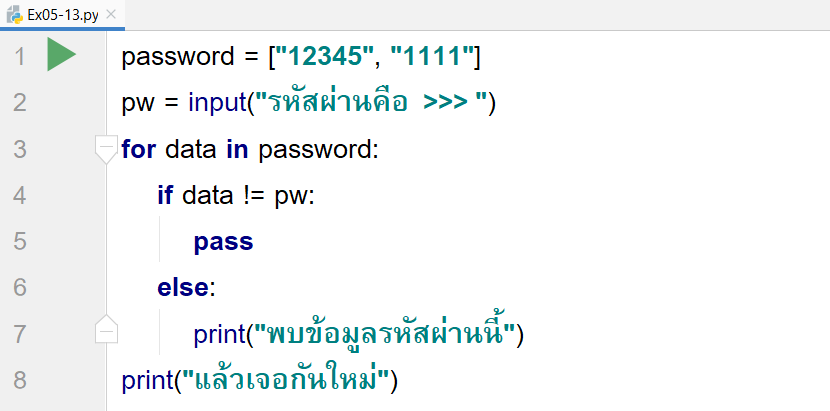
คำสั่ง break และคำสั่ง continue เป็นคำสั่งที่ใช้ร่วมกับคำสั่งการทำซ้ำเหมือนกัน แต่มีความต่างกันคือ

* คำสั่ง break จะหยุดการทำงานและออกจากการทำงานในลูปทันที
* คำสั่ง continue จะบังคับให้โปรแกรมข้ามไปทำงานที่รอบต่อไปทันที

# คำสั่ง pass: คำสั่งบังคับให้ผ่านไปทำงานในชุดคำสั่งถัดไป

**คำสั่ง pass** เป็นคำสั่งที่ใช้ในกรณีที่ไม่ต้องการให้มีการดำเนินการใดๆ ของชุดคำสั่ง และผ่านไปทำงานในชุดคำสั่งที่อยู่บรรทัดถัดไป เพื่อความเข้าใจให้ผู้อ่านศึกษาตัวอย่างต่อไปนี้

**ตัวอย่างโปรแกรมที่ 5-13** โปรแกรมการใช้คำสั่ง pass เพื่อตรวจสอบข้อมูลรหัสผ่าน มีโค้ดดังนี้

[[python\_05\_37.bmp]]

จากโปรแกรมที่ 5-13 อธิบายการทำงานของโปรแกรมได้ ดังนี้

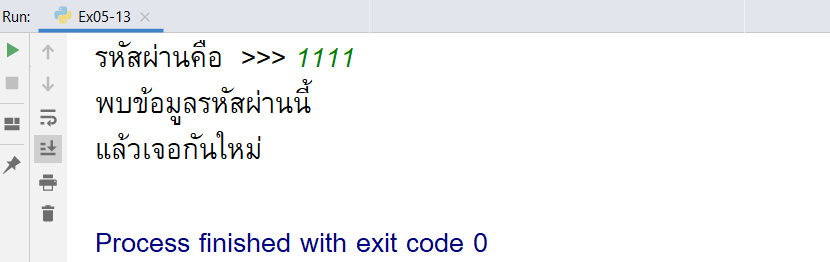
บรรทัดที่ 1 กำหนดตัวแปร password เป็นข้อมูล list ที่มีรหัสผ่าน 2 ค่า คือ “12345” และ “1111”

บรรทัดที่ 2 รับข้อมูลรหัสผ่านด้วยฟังก์ชัน input() และเก็บค่าในตัวแปร pw

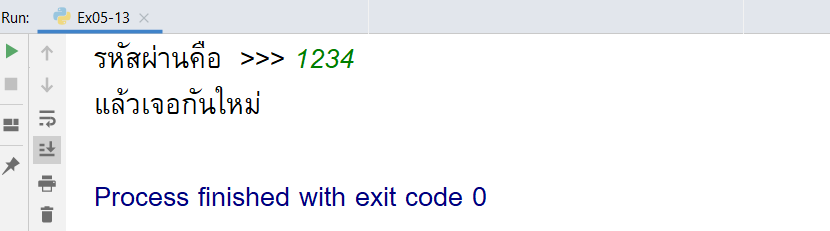
บรรทัดที่ 3-7 ใช้คำสั่ง for วนลูปเพื่อนำข้อมูล password แต่ละตัวมาเก็บไว้ในตัวแปร data เพื่อนำมาเปรียบเทียบกับตัวแปร pw ถ้าข้อมูลไม่ตรงกัน โปรแกรมก็จะผ่านการทำงานด้วยคำสั่ง pass ในบรรทัดที่ 5 ถ้าตรงกันก็จะแสดงข้อความว่า พบข้อมูลรหัสผ่านนี้ ตามคำสั่งในบรรทัด ที่ 7

บรรทัดที่ 8 แสดงข้อความว่า แล้วเจอกันใหม่

ผลการทำงานของโปรแกรมแสดงได้ดังนี้

[[python\_05\_38.bmp]]

[[ผลการทำงานของโปรแกรม กรณีพบรหัสผ่าน]]

[[python\_05\_39.bmp]]

[[ผลการทำงานของโปรแกรม กรณีไม่พบรหัสผ่าน]]

### สรุปท้ายบท

บทนี้ผู้อ่านได้ศึกษาคำสั่งควบคุมทิศทางการทำงานของโปรแกรม ซึ่งเป็นส่วนสำคัญที่ต้องเข้าใจ เพื่อให้สามารถเขียนโปรแกรมไปในทิศทางที่ต้องการได้ ประกอบด้วยคำสั่งแบบตามลำดับ (Sequence Control Statement), คำสั่งแบบมีทางเลือก (Selection Control Statement) และคำสั่งแบบทำซ้ำ (Iteration Control Statement) และใช้คำสั่ง break, continue, pass ในการจัดการการทำงานของโปรแกรมให้ได้วัตถุประสงค์ตามที่ต้องการ

# แบบฝึกหัด

1. จงเขียนโปรแกรมคำนวณการแลกคูปอง โดยรับข้อมูลจำนวนราคาสินค้า โดยจำนวนเงินที่ใช้แลกคูปองคิดจาก 10% ของราคาสินค้า และกำหนดให้แลกเป็นคูปอง 500 บาท, 100 บาท และ 50 บาท
2. จงเขียนโปรแกรมคำนวณค่าใช้จ่ายทั้งหมดสำหรับการท่องเที่ยวของสมาชิกกลุ่มหนึ่ง โดยมีเงื่อนไขคือ ถ้ามีจำนวนสมาชิกเกิน 10 คน จะได้รับส่วนลด 20% ของค่าใช้จ่ายรวม กำหนดให้รับข้อมูลอัตราค่าใช้จ่ายปกติ, จำนวนสมาชิก และแสดงผลค่าใช้จ่ายทั้งหมด
3. จงเขียนโปรแกรมคำนวณเงินโบนัสให้กับพนักงานขายของบริษัทแห่งหนึ่ง โดยมีเงื่อนไขคือ เงินโบนัสที่ได้จะขึ้นกับยอดขายและระยะเวลาที่ทำงานของพนักงานแต่ละคน ซึ่งพนักงานที่มีอายุงานตั้งแต่ 1 ปีขึ้นไป จึงจะมีสิทธิได้รับเงินโบนัส, พนักงานที่มีอายุงานน้อยกว่า 5 ปี จะได้รับเงินโบนัส 3% ของยอดขาย, ส่วนพนักงานที่มีอายุงานตั้งแต่ 5 ปี ขึ้นไปจะได้รับโบนัส 5% ของยอดขาย และแสดงผลว่า มีพนักงานกี่คนที่มีสิทธิ์ได้รับเงินโบนัส พนักงานแต่ละคนได้รับเงินโบนัสเท่าไร และจำนวนเงินโบนัสทั้งหมดที่บริษัทต้องจ่ายให้กับพนักงานเป็นจำนวนเงินเท่าไร กำหนดให้รับข้อมูลจำนวนพนักงาน จำนวนยอดขาย และระยะเวลาการทำงานของแต่ละคน
4. จงเขียนโปรแกรมคำนวณหาคะแนนรวม 100% และเกรดของนักศึกษาจำนวนหนึ่ง โดยมีเกณฑ์การคำนวณเกรดดังนี้

* คะแนนรวมมากกว่า 75 คะแนน จะได้เกรด H
* คะแนนรวม 50 ถึง 75 คะแนน ได้เกรด S
* คะแนนรวมน้อยกว่า 50 คะแนน ได้เกรด U

และกำหนดให้

* ข้อมูลคะแนนงาน (30 คะแนน) และคะแนนสอบ (35 คะแนน) โดยมีคะแนนเต็มรวมเท่ากับ 65 คะแนน \*\*ในกรณีที่คะแนนที่ไม่ถูกต้อง จะต้องมีการแจ้งเตือนรับข้อมูลใหม่ \*\*
* เมธอดคำนวณเกรด (calGrade) คิดจากคะแนนรวมตามเกณฑ์ที่กำหนดข้างต้น คืนค่าเป็นข้อมูลชนิด char

ให้แสดงผล

* ชื่อนักศึกษา คะแนนรวม และเกรดของนักศึกษาแต่ละคน
* จำนวนนักศึกษาที่ได้เกรด H, เกรด S และเกรด U ตามลำดับ