

บทที่ 3

ตัวแปรและชนิดข้อมูลในภาษาโปรแกรมไพธอน

การเขียนโปรแกรมภาษาโปรแกรมไพธอน สิ่งแรกที่คุณเริ่มพัฒนาโปรแกรมจำเป็นต้องรู้คือ ตัวแปรและชนิดข้อมูล ซึ่งภาษาโปรแกรมไพธอนได้จัดเตรียมชนิดข้อมูลไว้ให้ใช้งานอย่างมาก ในบทนี้ผู้อ่านจะได้รู้จักกับวิธีการประกาศสร้างตัวแปร การกำหนดค่าให้กับตัวแปร และชนิดข้อมูลพื้นฐานต่าง ๆ ได้แก่ ชนิดข้อมูลจำนวนเต็มที่มีทั้งตัวเลขจำนวนเต็มบวก จำนวนเต็มลบ ชนิดข้อมูลจำนวนทศนิยม ชนิดข้อมูลค่าความจริงที่ให้ผลลัพธ์เป็นค่าจริง (True) หรือเป็นค่าเท็จ (False) ชนิดข้อมูลสายอักขระหรือสตริง อีกทั้งยังจะได้รู้จักกับชนิดข้อมูลจำนวนเชิงซ้อน

3.1 การตั้งชื่อตัวแปรในภาษาโปรแกรมไพธอน

ตัวแปร (Variable) คือ ชื่อที่ตั้งขึ้นมาใช้สำหรับเก็บค่าหรือชนิดข้อมูลต่าง ๆ ที่ต้องการ แล้วนำตัวแปรที่ตั้งขึ้นมาไปเขียนเป็นคำสั่งโปรแกรมสำหรับประมวลผล ตัวแปรที่ตั้งขึ้นมาใช้งานนั้นสามารถเปลี่ยนค่าข้อมูลที่เก็บอยู่ในขณะที่โปรแกรมทำการประมวลผลได้ กฎการตั้งชื่อตัวแปรของภาษาโปรแกรมไพธอนมีดังนี้

1. การตั้งชื่อตัวแปรด้วยตัวอักษรพิมพ์เล็กและตัวอักษรพิมพ์ใหญ่ ภาษาโปรแกรมไพธอนจะถือว่าเป็นคนละชื่อตัวแปร
2. ชื่อตัวแปรต้องขึ้นต้นด้วยตัวอักษรภาษาอังกฤษ (A-Z, a-z) เท่านั้น แล้วตามด้วยตัวเลข เช่น Animal, animal_01, big10, book10group เป็นต้น

3. ชื่อตัวแปรต้องไม่มีช่องว่าง จุด และสัญลักษณ์พิเศษ ยกเว้นเครื่องหมาย underscore(_) เช่น msg_1 หรือ _msg_1
4. ควรตั้งชื่อตัวแปรให้สื่อความหมายกับค่าข้อมูลที่จะเก็บ เพราะทำให้อ่านและเข้าใจได้ง่าย เช่น Tax ใช้เก็บค่าภาษี, studentName หรือ st_name ใช้เก็บชื่อนักเรียน, passwd ใช้เก็บรหัสผ่าน เป็นต้น
5. สัญลักษณ์ต่อไปนี้ห้ามนำมาใช้ตั้งชื่อตัวแปร

$$\%^&()_+=-*/\}\{\! \$\#$$
6. เมื่อต้องการผสมคำตั้งชื่อตัวแปรควรใช้เครื่องหมาย underscore เชื่อม หรือกำหนดให้คำแรกใช้ตัวอักษรภาษาอังกฤษพิมพ์เล็กขึ้นต้น แล้วคำที่สองให้ขึ้นต้นด้วยตัวใหญ่ เช่น Table_name หรือ tableName เป็นต้น
7. การตั้งชื่อตัวแปรต้องไม่ซ้ำกับคำสงวน (Reserved keywords) ซึ่งมีทั้งหมด 33 คำ ผู้อ่านสามารถตรวจสอบคำสงวนได้โดยการเพิ่มคำสั่ง

```
import keyword
```

ลงในส่วนการประกาศแล้วใช้คำสั่ง `print(keyword.kwlist)` มีดังต่อไปนี้

and	as	assert	break	class	continue
def	del	elif	else	except	False
finally	for	from	global	if	import
in	is	lambda	None	nonlocal	not
or	pass	raise	return	True	try
while	with	yield			

ในการตั้งชื่อตัวแปรขึ้นมาใช้งานควรตั้งชื่อให้สื่อความหมายกับชนิดข้อมูลที่จัดเก็บ เมื่อพัฒนาโปรแกรมไปสักระยะหนึ่งโปรแกรมมีขนาดใหญ่ขึ้น ตัวแปรมีจำนวนมากขึ้นอาจจะทำให้สับสนหากตั้งชื่อตัวแปรไม่สอดคล้อง ทำให้ต้องเสียเวลามากในการแก้ไขโปรแกรม

3.2 การประกาศตัวแปร

การประกาศสร้างตัวแปร (Declaration) คือ การสร้างตัวแปรสำหรับเก็บค่าข้อมูล เช่น ตัวเลข ตัวอักษร ข้อความ เป็นต้น ก่อนนำไปประมวลผลอีกครั้ง ภาษาโปรแกรมไพธอนไม่จำเป็นต้องกำหนดชนิดข้อมูลให้กับตัวแปร ตัวแปรที่ประกาศสร้างขึ้นมาใช้งานจะกำหนดค่าข้อมูลไว้หรืออาจสร้างตัวแปรขึ้นมาเป็นค่าว่างไว้ก่อนก็ได้ แล้วจึงกำหนดค่าข้อมูลให้กับตัวแปรทีหลัง ตัวอย่างการสร้างตัวแปรจะเป็นดังต่อไปนี้

ตัวอย่าง 3.1

การประกาศตัวแปร (Declaration)

```
1 msg = 'Python Programming' # ประกาศตัวแปร msg เป็นสตริง (String)
2 num = 2415 # ประกาศตัวแปร num เป็นชนิดข้อมูลจำนวนเต็ม (Integer)
3 score = 78.54 # ประกาศตัวแปร score เป็นชนิดข้อมูลทศนิยม (float)
4 grad = ' ' # ประกาศตัวแปร grad เก็บสตริงช่องว่าง
```

ชื่อตัวแปรจะอยู่ทางด้านซ้ายมือของเครื่องหมายเท่ากับ (=) ส่วนทางด้านขวามือของเครื่องหมายเท่ากับคือ ชนิดข้อมูลที่กำหนดให้กับตัวแปร ถ้าเป็นชนิดข้อมูลสายอักขระหรือสตริง หรือเรียกสั้น ๆ ว่า 'สตริง' จะอยู่ในเครื่องหมาย '...' หรือ '...' ถ้ายังไม่กำหนดค่าให้กับตัวแปรหรือต้องการให้ตัวแปรเก็บค่าว่างให้ตั้งเป็น `var = ' '` หรือถ้ายังไม่ได้กำหนดชนิดข้อมูลให้กับตัวแปรให้ใช้ `None` เช่น `var = None` ถ้าเพียงแต่ตั้งชื่อตัวแปรแล้วสั่งให้โปรแกรมทำงานจะทำให้เกิดการแจ้งเตือนข้อผิดพลาดขึ้น

ในภาษาโปรแกรมไพธอนเวอร์ชัน 3 สามารถสร้างตัวแปรเป็นภาษาไทยได้ แต่ถึงอย่างไรก็ตามเพื่อความเป็นมาตรฐานในการเขียนคำสั่งโปรแกรมและให้ง่ายต่อผู้ที่ต้องการนำโปรแกรมของเราไปพัฒนาต่อในอนาคต ดังนั้นจึงควรสร้างตัวแปรเป็นภาษาอังกฤษและตั้งชื่อให้สื่อความหมาย

การประกาศสร้างตัวแปรเป็นชื่อภาษาไทย แสดงดังตัวอย่างต่อไปนี้

ตัวอย่าง 3.2

การเขียนคำสั่งโปรแกรมตั้งชื่อตัวแปรเป็นภาษาไทย

```

1 ปากกา = 'สีแดง'      # ประกาศตัวแปรชื่อ 'ปากกา' เป็นชนิดข้อมูลสตริง
2 โทรศัพท์ = 'Samsung'  # ประกาศตัวแปรชื่อ 'โทรศัพท์' เป็นชนิดข้อมูลสตริง
3 print('ปากกาสี =', ปากกา)    # แสดงผลข้อมูลจากค่าตัวแปร 'ปากกา'
4 print('โทรศัพท์ยี่ห้อ =', โทรศัพท์)  # แสดงผลข้อมูลจากค่าตัวแปร 'โทรศัพท์'

```

ปากกาสี = สีแดง
โทรศัพท์ยี่ห้อ = Samsung

**3.3 ตัวแปรชนิดค่าคงที่**

ค่าคงที่ (Constant) คือ ตัวแปรที่สร้างขึ้นมาเก็บค่าข้อมูลที่จะไม่มีการเปลี่ยนแปลงค่าในระหว่างการประมวลผล เช่น ค่า π มีค่าเท่ากับ 3.14 เราสามารถตั้งชื่อตัวแปรเป็นค่าคงที่ได้เท่ากับ `pi = 3.14` หรือค่าภาษีมูลค่าเพิ่มค่าเท่ากับ 7% เราสามารถตั้งชื่อตัวแปรเป็นค่าคงที่ได้เท่ากับ `vat = 0.07` เป็นต้น ดังแสดงตัวอย่างต่อไปนี้

ตัวอย่าง 3.3การประกาศตัวแปรชนิดค่าคงที่หาพื้นที่วงกลม ด้วยกำหนดค่า `pi = 3.14`

```

1 pi = 3.14          # สร้างตัวแปร pi เป็นค่าคงที่
2 area = pi * 5 * 5  # คูณค่าตัวแปร pi เพื่อหาพื้นที่วงกลม
3 circumference = 2 * pi * 10 # คูณค่าตัวแปร pi เพื่อหาเส้นรอบวงกลม
4 print('พื้นที่วงกลม =', area)    # แสดงผลลัพธ์ตัวแปร area
5 print('เส้นรอบวงกลม =', circumference)  # แสดงผลลัพธ์ตัวแปร
  → circumference

```

พื้นที่วงกลม = 78.5
เส้นรอบวงกลม = 62.800000000000004



3.4 ชนิดข้อมูลจำนวนเต็ม

ชนิดข้อมูลจำนวนเต็ม (Integers) ประกอบไปด้วยจำนวนเต็มบวก (Positive Integers) เช่น **10**, **15**, **255** จำนวนเต็มลบ (Negative Integers) เช่น **-14**, **-2** และ ศูนย์ (Zero) จำนวนเต็มเหล่านี้สามารถเขียนในรูปเลขฐานต่าง ๆ ได้แก่ เลขฐานสอง (Binary) เลขฐานแปด (Octal) และเลขฐานสิบ (Hexadecimal) จำนวนเต็มเลขฐานสิบประกอบด้วยตัวเลข **0-9** เลขฐานสองประกอบด้วยตัวเลขสองตัว ได้แก่ **0** และ **1** เลขฐานแปดประกอบด้วยตัวเลขแปดตัว ได้แก่ **0-7** และเลขฐานสิบหกประกอบด้วยตัวเลข สิบหกตัว ได้แก่ **0-9** และตัวอักษรอีกหก ตัว ได้แก่ **A-F** เพื่อความสะดวกในการแทนค่า **10-15** ซึ่งเป็นเลขสองหลัก

ในภาษาโปรแกรมไพธอนไม่ได้กำหนดความยาวของชนิดข้อมูลจำนวนเต็ม นั่นหมายความว่าผู้เขียนโปรแกรมสามารถประกอบตัวแปรเก็บชนิดข้อมูลจำนวนเต็มให้มีความยาวเท่าไรก็ได้ แต่ก็มีผลกระทบต่อการใช้งานหน่วยความจำไม่เพียงพอ

ตัวอย่าง 3.4

การดำเนินการกับชนิดข้อมูลจำนวนเต็มที่มีความยาวไม่จำกัด

```
1 num1 = 546135478945123456214 # สร้างตัวแปร num1 เป็นชนิดข้อมูลจำนวนเต็ม
2 num2 = 14567489412457845679 # สร้างตัวแปร num2 เป็นชนิดข้อมูลจำนวนเต็ม
3 print('ผลบวกของ num1 กับ num2 = ', num1 + num2) # แสดงผลลัพธ์ num1 +
  ↳ num2
4 print('ผลลบของ num1 กับ num2 = ', num1 - num2) # แสดงผลลัพธ์ num1 -
  ↳ num2
```

ผลบวกของ num1 กับ num2 = 560702968357581301893

ผลลบของ num1 กับ num2 = 531567989532665610535



การดำเนินการกับชนิดข้อมูลจำนวนเต็มเลขฐานสอง ฐานแปด และฐานสิบหก มีวิธีการดำเนินการดังต่อไปนี้

- ถ้าต้องการกำหนดเลขจำนวนเต็มเป็นเลขฐานสองให้ใช้สัญลักษณ์ **0b** นำหน้า เช่น **0b1001011 + 0b1101101**

- ถ้าต้องการกำหนดเลขจำนวนเต็มเป็นเลขฐานแปดให้ใช้สัญลักษณ์ **0o** นำหน้า เช่น **0o12451 + 0o54125**
- ถ้าต้องการกำหนดเลขจำนวนเต็มเป็นเลขฐานสิบหกให้ใช้สัญลักษณ์ **0x** นำหน้า เช่น **0x76ade + 0x97c7a**

นอกจากนี้สามารถเรียกใช้ฟังก์ชันแปลงเลขฐานให้เป็นเลขฐานอื่น ๆ ได้ โดยรูปแบบการใช้งานคือ ชื่อฟังก์ชัน(ตัวแปร)

- ฟังก์ชัน **int()** เป็นฟังก์ชันแปลงเลขฐานที่ต้องการให้เป็นเลขฐานสิบ
- ฟังก์ชัน **bin()** เป็นฟังก์ชันแปลงเลขฐานที่ต้องการให้เป็นเลขฐานสอง
- ฟังก์ชัน **oct()** เป็นฟังก์ชันแปลงเลขฐานที่ต้องการให้เป็นเลขฐานแปด
- ฟังก์ชัน **hex()** เป็นฟังก์ชันแปลงเลขฐานที่ต้องการให้เป็นเลขฐานสิบหก

ตัวอย่าง 3.5

การแปลงเลขฐานสิบเป็นเลขฐานสอง ฐานแปด และฐานสิบหก

```
1 x = 6749 # สร้างตัวแปร x เก็บชนิดข้อมูลจำนวนเต็มเลขฐานสิบ
2 print('แปลงเลขฐานสิบเป็นฐานสอง =', bin(x)) # ฟังก์ชันแปลงเลขฐานสิบเป็นฐานสอง
3 print('แปลงเลขฐานสิบเป็นฐานแปด =', oct(x)) # ฟังก์ชันแปลงเลขฐานสิบเป็นฐานแปด
4 print('แปลงเลขฐานสิบเป็นฐานสิบหก =', hex(x)) # ฟังก์ชันแปลงเลขฐานสิบเป็นฐานสิบ
   ↪   หก
```

แปลงเลขฐานสิบเป็นฐานสอง = 0b1101001011101

แปลงเลขฐานสิบเป็นฐานแปด = 0o15135

แปลงเลขฐานสิบเป็นฐานสิบหก = 0x1a5d



ตัวอย่าง 3.6

การแปลงเลขฐานสอง ฐานแปด และฐานสิบหก เป็นเลขฐานสิบ

```
1 a = 0b100111011001 # สร้างตัวแปร a เก็บชนิดข้อมูลจำนวนเต็มเลขฐานสอง
2 b = 0o6453 # สร้างตัวแปร b เก็บชนิดข้อมูลจำนวนเต็มเลขฐานแปด
3 c = 0xAC41 # สร้างตัวแปร c เก็บชนิดข้อมูลจำนวนเต็มเลขฐานสิบหก
4 print('แปลงเลขฐานสองเป็นฐานสิบ =', int(a)) # แสดงผลลัพธ์การแปลงเลขฐาน
5 print('แปลงเลขฐานแปดเป็นฐานสิบ =', int(b))
6 print('แปลงเลขฐานสิบหกเป็นฐานสิบ =', int(c))
```

แปลงเลขฐานสองเป็นฐานสิบ = 2521
 แปลงเลขฐานแปดเป็นฐานสิบ = 3371
 แปลงเลขฐานสิบหกเป็นฐานสิบ = 44097



ตัวอย่าง 3.7

การบวกเลขฐานสองและการแปลงผลลัพธ์จากเลขฐานสิบ เป็นเลขฐานสอง ฐานแปด และฐานสิบหก

```
1 a = 0b1001011; b = 0b1101101 # สร้างตัวแปร a และ b
2 c = a + b # บวกค่าตัวแปร a และ b
3 print('ผลบวกของ a กับ b =', c) # แสดงผลลัพธ์ตัวแปร C
4 print('แปลงผลลัพธ์จากผลบวก a กับ b เป็นเลขฐานสอง = ', bin(c)) # แสดงผลลัพธ์
5 print('แปลงผลลัพธ์จากผลบวก a กับ b เป็นเลขฐานแปด = ', oct(c))
6 print('แปลงผลลัพธ์จากผลบวก a กับ b เป็นเลขฐานสิบหก = ', hex(c))
```

ผลบวกของ a กับ b = 184
 แปลงผลลัพธ์จากผลบวก a กับ b เป็นเลขฐานสอง = 0b10111000
 แปลงผลลัพธ์จากผลบวก a กับ b เป็นเลขฐานแปด = 0o270
 แปลงผลลัพธ์จากผลบวก a กับ b เป็นเลขฐานสิบหก = 0xb8



3.5 ชนิดข้อมูลจำนวนทศนิยม

ชนิดข้อมูลประเภทจำนวนทศนิยม (Float) ประกอบด้วยตัวเลข 2 ส่วน โดยมีเครื่องหมายจุด (.) คั่นระหว่างตัวเลขที่อยู่ด้านหน้าและด้านหลัง ตัวเลขด้านหน้าเป็นตัวเลขจำนวนเต็ม และตัวเลขที่อยู่ด้านหลังจุดเรียกว่า ทศนิยม การเขียนเลขทศนิยมสามารถเขียนได้สองแบบคือ แบบแรก เช่น 15.5, 20.451, 4.513400000 เป็นต้น และแบบที่สอง เช่น

$$4.517458\text{E} - 2 = 4.517458 \times 10^{-2} = 0.04517458$$

หรือ

$$1.46547\text{e}4 = 1.465473 \times 10^4 = 14654.73$$

เป็นต้น ในแบบที่สองจะสังเกตเห็นว่ามีตัวอักษร E และ e ซึ่งเป็นการเขียนในรูปแบบของเลขยกกำลังสิบ (Exponential Form)

ตัวอย่าง 3.8

การแสดงผลการดำเนินการบวก ลบ กับชนิดข้อมูลจำนวนทศนิยม

```

1 # สร้างตัวแปร float
2 a = 341.451E-3
3 b = 251.147e-2
4 x = 10.5
5 y = 17.5
6 d = a - b # ลบแล้วเก็บผลลัพธ์ไว้ที่ตัวแปร d
7 z = x + y # ลบแล้วเก็บผลลัพธ์ไว้ที่ตัวแปร z
8 print('ค่าตัวเลขจำนวนทศนิยมของ b = ', b) # แสดงผลลัพธ์ตัวแปร b
9 print('ผลลัพธ์จากการลบ a กับ b = ', d) # แสดงผลลัพธ์ตัวแปร d
10 print('ผลลัพธ์จากการบวก x กับ y = ', z) # แสดงผลลัพธ์ตัวแปร z

```

ค่าตัวเลขจำนวนทศนิยมของ b = 2.51147
 ผลลัพธ์จากการลบ a กับ b = -2.170019
 ผลลัพธ์จากการบวก x กับ y = 28.0



ในกรณีที่ต้องการแปลงชนิดข้อมูลจำนวนเต็ม ให้เป็นชนิดข้อมูลจำนวนทศนิยม ให้ใช้ฟังก์ชัน `float()` มีวิธีการเรียกใช้งานแสดงดังตัวอย่างที่ 3.8

ตัวอย่าง 3.9

การแปลงชนิดข้อมูลอักขระหรือสตริง เป็นชนิดข้อมูลจำนวนทศนิยม

```
1 a = 56; b = 55 # สร้างตัวแปร a และ b เก็บชนิดข้อมูลจำนวนเต็ม
2 print('ผลลัพธ์จากการแปลงค่าจากตัวแปร a =', float(a)) # แปลงค่า a เป็น
  ↳ float
3 print('ผลลัพธ์จากการแปลงค่าจากตัวแปร b =', float(b)) # แปลงค่า b เป็น
  ↳ float
4 print('ผลลัพธ์จากการบวกค่าของ a + b =', float(a) + float(b)) # นำ a +
  ↳ b
5 print('ผลลัพธ์จากการคูณค่าของ a * b =', float(a) * float(b)) # นำ a *
```

ผลลัพธ์จากการแปลงค่าจากตัวแปร a = 56.8
 ผลลัพธ์จากการแปลงค่าจากตัวแปร b = 55.8
 ผลลัพธ์จากการบวกค่าของ a + b = 111.0
 ผลลัพธ์จากการคูณค่าของ a * b = 3080.0



3.6 ชนิดข้อมูลจำนวนเชิงซ้อน

จำนวนเชิงซ้อน (Complex) ได้ถูกนำมาใช้งานในสาขาต่าง ๆ เช่น Electrical Engineering, Fluid Dynamics, Quantum Mechanics, Computer Graphics, Dynamic Systems และสาขาอื่น ๆ อีกโดยเฉพาะอย่างยิ่งในด้านวิทยาศาสตร์และวิศวกรรม จำนวนเชิงซ้อนจะเขียนอยู่ในรูปของคู่ $x + yi$ เช่น $5 + 2i$, $30 - 6i$ เป็นต้น โดยเรียก x เป็นจำนวนจริง (Real Part) และ y เป็นส่วนจินตภาพ (Imaginary Part) ภาษาโปรแกรมไพธอนจะใช้ตัวอักษร j แทนตัวอักษร i

การใช้ตัวดำเนินการกับชนิดข้อมูลจำนวนเชิงซ้อนทำได้ปกติเหมือนกับชนิดข้อมูลจำนวนเต็มและชนิดข้อมูลทศนิยม แสดงดังตัวอย่างต่อไปนี้

ตัวอย่าง 3.10

การใช้ตัวดำเนินการบวก ลบ กับชนิดข้อมูลเชิงซ้อน

```

1 a = 20 + 5j; b = 15 + 6j; c = 6 - 7j # สร้างตัวแปรเป็นชนิดข้อมูลเชิงซ้อน
2 x = a + b # บวกค่าตัวแปร a กับ b
3 y = b * c # คูณค่าตัวแปร b กับ c
4 print('ผลลัพธ์จากการบวกระหว่าง a + b = ', x) # แสดงผลลัพธ์ตัวแปร x
5 print('ผลลัพธ์จากการคูณระหว่าง b * c = ', y) # แสดงผลลัพธ์ตัวแปร y

```

ผลลัพธ์จากการบวกระหว่าง $a + b = (35+11j)$ ผลลัพธ์จากการคูณระหว่าง $b * c = (132-69j)$ **3.7 ชนิดข้อมูลตรรกะ**

ชนิดข้อมูลตรรกะหรือชนิดข้อมูลค่าความจริง (Boolean) ให้ผลลัพธ์เพียงสองค่า คือ ค่าจริง (True) หรือค่าเท็จ (False) อย่างไรก็ตามหนึ่ง เราสามารถกำหนดค่า True หรือ False ให้กับตัวแปรเพื่อนำไปเปรียบเทียบได้เลย

ตัวอย่าง 3.11

การเขียนคำสั่งโปรแกรมใช้งานชนิดข้อมูลตรรกะ

```
1 x = True; y = False; z = True # สร้างตัวแปรเป็นชนิดข้อมูลตรรกะ
2 print('ผลการเปรียบเทียบ x == y หรือไม่ = ', x == y) # แสดงผลการเปรียบเทียบ
  ↳ x, y
3 print('ผลการเปรียบเทียบ y == z หรือไม่ = ', y == z) # แสดงผลการเปรียบเทียบ
  ↳ y, z
4 print('ผลการเปรียบเทียบ z == True หรือไม่ = ', z == True) # แสดงผลการ
  ↳ เปรียบเทียบ z
```

ผลการเปรียบเทียบ x == y หรือไม่ = False
ผลการเปรียบเทียบ y == z หรือไม่ = False
ผลการเปรียบเทียบ z == True หรือไม่ = True



สามารถเรียกใช้งานฟังก์ชัน `bool()` ตรวจสอบค่าข้อมูลในตัวแปรที่สร้างขึ้นมาใช้งานว่า ได้เก็บค่าข้อมูลไว้หรือไม่ ถ้าตัวแปรใดมีการกำหนดค่าข้อมูลไว้จะได้ผลลัพธ์เป็น `True` แต่ถ้าตัวแปรใดไม่มีการกำหนดค่าหรือกำหนดค่าเป็น `0`, `None` หรือ `' '` จะให้ผลลัพธ์เป็น `False` ถ้าตัวแปรประกาศเป็นช่องว่าง `' '` ผลลัพธ์ที่ได้เป็น `True` ให้ผู้อ่านพิจารณาการใช้งานฟังก์ชัน `bool()` จากตัวอย่าง 3.11

ตัวอย่าง 3.12การเรียกใช้งานฟังก์ชัน `bool()` ตรวจสอบค่าข้อมูลในตัวแปร

```

1 b = ''; c = ' '; n = None; num = 245
2 print('มีข้อมูลอยู่ในตัวแปร b หรือไม่ = ', bool(b)) # แสดงผลตรวจสอบค่าตัวแปร b
3 print('มีข้อมูลอยู่ในตัวแปร c หรือไม่ = ', bool(c)) # แสดงผลตรวจสอบค่าตัวแปร c
4 print('มีข้อมูลอยู่ในตัวแปร n หรือไม่ = ', bool(n)) # แสดงผลตรวจสอบค่าตัวแปร n
5 print('มีข้อมูลอยู่ในตัวแปร num หรือไม่ = ', bool(num)) # แสดงผลตรวจสอบค่าตัว
  ↳ แปร num

```

มีข้อมูลอยู่ในตัวแปร b หรือไม่ = False
 มีข้อมูลอยู่ในตัวแปร c หรือไม่ = True
 มีข้อมูลอยู่ในตัวแปร n หรือไม่ = False
 มีข้อมูลอยู่ในตัวแปร num หรือไม่ = True



3.8 ชนิดข้อมูลสายอักขระหรือสตริง

ชนิดข้อมูลสายอักขระหรือสตริง (String) หรือเรียกสั้น ๆ ว่า ชนิดข้อมูลสตริง เป็นการนำตัวอักขระหลาย ๆ ตัวมาเรียงต่อกันเป็นข้อความหรือประโยค การประกาศชนิดข้อมูลนี้ขึ้นมาใช้งานจะอยู่ในเครื่องหมาย Single Quote ('...') หรือ Double Quote ("...") ให้เลือกใช้รูปแบบใดรูปแบบหนึ่ง แสดงดังตัวอย่างต่อไปนี้

ตัวอย่าง 3.13

การประกาศสร้างตัวแปรชนิดข้อมูลสตริงขึ้นมาใช้งาน

```
1 msg = 'Python Programming' # สร้างตัวแปร msg เก็บชนิดข้อมูลสตริงใช้เครื่องหมาย  
2 msg1 = 'Language' # สร้างตัวแปร msg1 เก็บชนิดข้อมูลสตริงใช้เครื่องหมาย ' '  
3 print(msg) # แสดงผลลัพธ์ตัวแปร msg  
4 print(msg1) # แสดงผลลัพธ์ตัวแปร msg1  
5 print(msg, msg1) # แสดงผลลัพธ์ตัวแปร msg และ msg1
```

```
Python Programming  
Language  
Python Programming Language
```



3.9 การแปลงชนิดข้อมูล

ในการพัฒนาโปรแกรมบางครั้งเราจำเป็นต้องแปลงชนิดข้อมูล (Data Type Conversion) ก่อนนำไปดำเนินการ หากไม่มีการแปลงชนิดข้อมูลก่อนจะทำให้เกิดข้อผิดพลาดขึ้นมาได้ ยกตัวอย่างการป้อนข้อมูลผ่านทางแป้นพิมพ์ด้วยฟังก์ชัน `input()` ถึงแม้จะป้อนข้อมูลเป็นตัวเลขจำนวนเต็มหรือจำนวนทศนิยม เมื่อนำไปดำเนินการกับชนิดข้อมูลจำนวนเต็มหรือชนิดข้อมูลทศนิยมจะทำให้เกิดการแจ้งเตือนข้อผิดพลาดขึ้น เนื่องจากตัวเลขที่ป้อนเข้ามานั้นเป็นชนิดข้อมูลสตริง แสดงดังตัวอย่างต่อไปนี้

ตัวอย่าง 3.14

การแจ้งเตือนข้อผิดพลาดเมื่อนำชนิดข้อมูลสตริงดำเนินการกับชนิดข้อมูลจำนวนเต็ม

```
1 num = input('Enter number = ') # กำหนดให้ป้อนข้อมูลผ่านคีย์บอร์ด
2 print(50 + num) # แสดงผลลัพธ์การบวก 50 กับค่าตัวแปร num
```

```
Enter number = 60
TypeError Traceback (most recent call last)
<ipython-input-1-43111086d092> in <module>
1 num = input('Enter number = ') # กำหนดให้ป้อนข้อมูลผ่านคีย์บอร์ด -----
2 print(50 + num) # แสดงผลลัพธ์การบวก 50 กับค่าตัวแปร num
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

จากตัวอย่างเห็นได้ว่าการแจ้งเตือนข้อผิดพลาดขึ้น เนื่องจากตัวแปร `num` เป็นชนิดข้อมูลสตริง ซึ่งไม่สามารถนำไปดำเนินการบวกกับ `50` ในบรรทัดที่ 2 ได้ เพราะเป็นชนิดข้อมูลจำนวนเต็ม ดังนั้นจึงต้องแปลงชนิดข้อมูลก่อนโดยการเรียกใช้งานฟังก์ชันแปลงค่าชนิดข้อมูลก่อนนำไปดำเนินการต่อ

- ถ้าต้องการแปลงเป็นชนิดข้อมูลจำนวนเต็มให้เรียกใช้งานใช้ฟังก์ชัน `int()`
- ถ้าต้องการแปลงเป็นชนิดข้อมูลจำนวนทศนิยมให้เรียกใช้งานใช้ฟังก์ชัน `float()`
- ถ้าต้องการแปลงเป็นชนิดข้อมูลจำนวนเชิงซ้อนให้เรียกใช้งานใช้ฟังก์ชัน `complex()`

ตัวอย่าง 3.15

การแปลงชนิดข้อมูลสตริงเป็นชนิดข้อมูลจำนวนเต็ม

```
1 num = int(input('กรุณาป้อนตัวเลขของคุณ : ')) # รับข้อมูลผ่านคีย์บอร์ดพร้อมแปลง
   ↳ ข้อมูล
2 x = '60' # สร้างตัวแปร x เป็นชนิดข้อมูลสตริง
3 y = int(x) + num # แปลงค่าตัวแปร x เป็นชนิดข้อมูลจำนวนเต็มแล้วบวกกับค่าตัวแปร
   ↳ num
4 print('ผลลัพธ์จากการบวก x + num =', y) # แสดงผลลัพธ์จากค่าตัวแปร y
```

กรุณาป้อนตัวเลขของคุณ : 50

ผลลัพธ์จากการบวก x + num = 110

**ตัวอย่าง 3.16**

การแปลงชนิดข้อมูลสตริงเป็นชนิดข้อมูลจำนวนทศนิยม

```
1 flo = float(input('กรุณาป้อนตัวเลขของคุณ : ')) # รับข้อมูลผ่านคีย์บอร์ดพร้อมแปลง
   ↳ ข้อมูล
2 x = '152.124' # สร้างตัวแปร x เป็นชนิดข้อมูลสตริง
3 y = float(x) + flo # แปลงค่าตัวแปร x เป็นชนิดข้อมูลจำนวนทศนิยมบวกกับค่าตัวแปร
   ↳ flo
4 print('ผลลัพธ์จากการบวก x + flo =', y) # แสดงผลลัพธ์จากค่าตัวแปร y
```

กรุณาป้อนตัวเลขของคุณ : 345.21

ผลลัพธ์จากการบวก x + flo = 497.33399999999995



ตัวอย่าง 3.17

การแปลงชนิดข้อมูลสตริงเป็นชนิดข้อมูลจำนวนเชิงซ้อน

```

1 a = '20+5j'; b = '15+6j'; # สร้างตัวแปร a และ b เป็นชนิดข้อมูลสตริง
2 x = complex(a) # แปลงค่าตัวแปร a เป็นชนิดข้อมูลเชิงซ้อน
3 y = complex(b) # แปลงค่าตัวแปร b เป็นชนิดข้อมูลเชิงซ้อน
4 z = complex(7,5) # แปลงค่าจำนวนจริงเป็นชนิดข้อมูลเชิงซ้อน
5 print('ผลลัพธ์การแปลงชนิดข้อมูลจำนวนเชิงซ้อนจากตัวแปร a =', x) # แสดงผลลัพธ์ x
6 print('ผลลัพธ์การแปลงชนิดข้อมูลจำนวนเชิงซ้อนจากตัวแปร b =', y) # แสดงผลลัพธ์ y
7 print('ค่าจำนวนจริงของ c =', x.real, 'และค่าจินตภาพของ c =', x.imag)
8     # แสดงผลลัพธ์การใช้ฟังก์ชัน real() และ imag()
9 print('ผลลัพธ์การแปลงเครื่องหมายตัวดำเนินการของตัวแปร y =', y.conjugate())
10     # แสดงผลลัพธ์การใช้ฟังก์ชัน conjugate ()
11 print('ผลลัพธ์การสร้างคู่อันดับจากตัวแปร z =', z) # แสดงผลลัพธ์ z

```

ผลลัพธ์การแปลงชนิดข้อมูลจำนวนเชิงซ้อนจากตัวแปร a = (20+5j)
 ผลลัพธ์การแปลงชนิดข้อมูลจำนวนเชิงซ้อนจากตัวแปร b = (15+6j)
 ค่าจำนวนจริงของ c = 20.0 และค่าจินตภาพของ c = 5.0
 ผลลัพธ์การแปลงเครื่องหมายตัวดำเนินการของตัวแปร y = (15-6j)
 ผลลัพธ์การสร้างคู่อันดับจากตัวแปร z = (7+5j)



ตัวอย่าง 3.18

การแปลงชนิดข้อมูลสตริงเป็นชนิดข้อมูลเชิงซ้อน กรณีเกิดการแจ้งเตือนข้อผิดพลาด

```

1 c = '25 + 9j'; # สร้างตัวแปรเก็บชนิดข้อมูลสตริงเก็บไว้ในตัวแปร c ที่มีช่องว่าง
2 print(complex(c)) # แสดงผลลัพธ์ตัวแปร c

```

ValueError Traceback (most recent call last)

<ipython-input-2-e8d7c7a0a2cf> in <module>

```

1 c = '25+9j ; # สร้างตัวแปรเก็บชนิดข้อมูลสตริงเก็บไว้ในตัวแปร c ที่มีช่องว่าง ----->
2 print(complex(c)) # แสดงผลลัพธ์ตัวแปร c

```

ValueError: complex() arg is a malformed string



3.10 การกำหนดค่าให้กับตัวแปร

ตัวแปรที่ประกาศสร้างขึ้นมาใช้งานจะมีการกำหนดค่าให้ ในภาษาโปรแกรมไพธอนมีรูปแบบการกำหนดค่าให้กับตัวแปรหลากหลายวิธี ผู้อ่านสามารถศึกษาได้จากตัวอย่างดังต่อไปนี้

ตัวอย่าง 3.19

การสร้างตัวแปรครั้งละหลาย ๆ ตัว และกำหนดเป็นชนิดข้อมูลเดียวกัน

```
1 num1 = num2 = num3 = 50 # สร้างตัวแปร num1, num2 และ num3 เป็นชนิดข้อมูล
   ↳ จำนวนเต็ม
2 print('ค่าข้อมูลในตัวแปร num1 = ', num1) # แสดงผลลัพธ์ตัวแปร num1
3 print('ค่าข้อมูลในตัวแปร num2 = ', num2) # แสดงผลลัพธ์ตัวแปร num2
4 print('ค่าข้อมูลในตัวแปร num3 = ', num3) # แสดงผลลัพธ์ตัวแปร num3
```

ค่าข้อมูลในตัวแปร num1 = 50
ค่าข้อมูลในตัวแปร num2 = 50
ค่าข้อมูลในตัวแปร num3 = 50



จากตัวอย่างในบรรทัดที่ 1 ได้ประกาศตัวแปร num1, num2 และ num3 พร้อมทั้งกำหนดค่าให้กับตัวแปรทั้ง 3 ตัว มีค่าเท่ากับ 50 และแสดงผลค่าข้อมูลที่เก็บอยู่ในตัวแปรทั้ง 3 ด้วยฟังก์ชัน `print()` ในบรรทัดที่ 2-4 ตามลำดับ

ตัวอย่าง 3.20

การสร้างตัวแปรครั้งละหลาย ๆ ตัว และกำหนดชนิดข้อมูลจำนวนเต็มและชนิดข้อมูลสตริง

```
1 num1, num2, msg = 50, 100, 'The world is beautiful.'
2 print('ค่าข้อมูลในตัวแปร num1 =', num1)
3 print('ค่าข้อมูลในตัวแปร num2 =', num2)
4 print('ค่าข้อมูลในตัวแปร msg =', msg)
```

ค่าข้อมูลในตัวแปร num1 = 50
ค่าข้อมูลในตัวแปร num2 = 100
ค่าข้อมูลในตัวแปร msg = The world is beautiful.



จากตัวอย่างได้ประกาศสร้างตัวแปร num1, num2 และ msg แต่ละตัวแปรจะถูกคั่นด้วยเครื่องหมาย comma (,) และกำหนดเป็นชนิดข้อมูลจำนวนเต็มและชนิดข้อมูลสตริงมีค่าเท่ากับ 50, 100 และ 'The world is beautiful.' ตามลำดับ การประกาศตัวแปรเช่นนี้จะมีการเรียงลำดับตัวแปรในการเก็บค่าข้อมูลที่ได้กำหนดไว้ สังเกตได้จากเมื่อใช้ฟังก์ชัน print() แสดงผลค่าข้อมูลที่เก็บอยู่ในแต่ละตัวแปร

ตัวอย่าง 3.21

การสร้างตัวแปรครั้งละหลาย ๆ ตัว แต่แยกเก็บชนิดข้อมูลอยู่ในบรรทัดเดียวกัน

```
1 x = 200; y = 300; msg = 'Python programming language'
2 print('ค่าข้อมูลในตัวแปร x =', x)
3 print('ค่าข้อมูลในตัวแปร y =', y)
4 print('ค่าข้อมูลในตัวแปร msg =', msg)
```

ค่าข้อมูลในตัวแปร x = 200
ค่าข้อมูลในตัวแปร y = 300
ค่าข้อมูลในตัวแปร msg = Python programming language



จากตัวอย่างการเขียนคำสั่งโปรแกรมได้ประกาศสร้างตัวแปร `x` และ `y` พร้อมทั้งกำหนดค่าข้อมูลเท่ากับ **200** และ **300** ตามลำดับ เป็นชนิดข้อมูลจำนวนเต็ม อีกทั้งยังได้ประกาศสร้างตัวแปร `msg` กำหนดค่าข้อมูลเป็นชนิดข้อมูลสตริง ผู้อ่านจะสังเกตเห็นว่าเมื่อสร้างตัวแปรให้อยู่ในบรรทัดเดียวกันจะมีเครื่องหมาย semicolon (`:`) คั่นระหว่างตัวแปร

3.11 การตรวจสอบชนิดข้อมูล

การพัฒนาโปรแกรมที่มีขนาดใหญ่และมีการประกาศสร้างตัวแปรไว้เป็นจำนวนมาก อาจทำให้ผู้พัฒนาโปรแกรมหลงลืมว่าตัวแปรเก็บชนิดข้อมูลประเภทใดไว้ เพื่อป้องกันการนำตัวแปรมาประมวลผลหรือดำเนินการผิดพลาด เราอาจจะต้องตรวจสอบชนิดข้อมูลของตัวแปรนั้นก่อน โดยการเรียกใช้งานฟังก์ชัน `type()` ซึ่งเป็นฟังก์ชัน Built-in แสดงตัวอย่างการเรียกใช้งานดังต่อไปนี้

ตัวอย่าง 3.22

การตรวจสอบชนิดข้อมูลด้วยฟังก์ชัน `type()`

```
1 msg = 'Python Programming language'
2 num = 254
3 float_ = 354.213
4 print(type(msg))
5 print(type(num))
6 print(type(float_))
```

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



จากตัวอย่างการเขียนคำสั่งโปรแกรม ได้ประกาศสร้างตัวแปร `msg`, `num` และ `float_` หากต้องการทราบว่าทั้ง 3 ตัวแปรเก็บชนิดข้อมูลประเภทใดไว้ ให้เรียกใช้งานฟังก์ชัน `type()` ดังแสดงในบรรทัดที่ 4-6

3.12 นิพจน์

นิพจน์ (Expressions) ในภาษาโปรแกรมไพธอน คือ ข้อความที่แสดงการดำเนินการเพื่อคำนวณหรือทำการเปรียบเทียบหาค่าต่าง ๆ โดยที่ในการดำเนินการจะประกอบด้วยค่าคงที่หรือตัวแปร สิ่งเหล่านี้เรียกว่า ตัวถูกดำเนินการ (Operand) และตัวดำเนินการ (Operator)

พิจารณานิพจน์ต่อไปนี้

$$z = a / b * (c + d)$$

นิพจน์นี้ประกอบด้วยตัวถูกดำเนินการ 5 ตัวคือ a, b, c, d และ z สำหรับตัวดำเนินการได้แก่ เครื่องหมาย +, /, * และ =

ตัวดำเนินการในภาษาโปรแกรมไพธอนมีหลายแบบให้เลือกใช้งาน และแต่ละแบบทำหน้าที่แตกต่างกันออกไป เช่น +, -, *, /, <, >, = เป็นต้น ผู้อ่านสามารถที่จะนำตัวดำเนินการมาผสมเป็นนิพจน์ให้ทำงานร่วมกันได้ โดยแบ่งตัวดำเนินการออกเป็นกลุ่มได้ดังหัวข้อต่อไป

3.13 ตัวดำเนินการทางคณิตศาสตร์

ภาษาไพธอนมีตัวดำเนินการทางคณิตศาสตร์ ได้แก่ +, -, *, /, %, //, ** ซึ่งตาราง 3.1 ได้แสดงสัญลักษณ์ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators) ความหมายของสัญลักษณ์ ตัวอย่างการใช้งาน และผลลัพธ์ที่ได้จากการคำนวณ ซึ่งผลลัพธ์ที่ได้จากการคำนวณจะแสดงผลลัพธ์ของชนิดข้อมูลที่ไม่เหมือนกัน ขึ้นอยู่กับการกำหนดชนิดข้อมูลให้กับตัวแปร จากตารางกำหนดค่าตัวแปร a และ b เป็นชนิดข้อมูลจำนวนเต็ม (Integer) ผลลัพธ์ที่ได้จากการจะแสดงผลออกมาได้เป็นชนิดข้อมูลจำนวนทศนิยม (float) ส่วนผลลัพธ์ที่เหลือจะแสดงผลเป็นชนิดข้อมูลจำนวนเต็ม (Integer)

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้	ผลลัพธ์ (a=5, b=3)
+	บวก (Addition)	c = a + b	c = 8
-	ลบ (Subtraction)	c = a - b	c = 2
*	คูณ (Multiplication)	c = a * b	c = 15
/	หาร (Division)	c = a / b	c = 1.6666667
%	หารเอาเศษ (Modulo)	c = a % b	c = 2
//	หารปัดเศษ (Floor Division)	c = a // b	c = 1
**	ยกกำลัง (Exponent)	c = a ** b	c = 125

ตาราง 3.1: สัญลักษณ์ตัวดำเนินการคำนวณทางคณิตศาสตร์

ตัวอย่าง 3.23

การนำตัวดำเนินการทางคณิตศาสตร์ต่าง ๆ มาใช้คำนวณหาผลลัพธ์


```
1 a = 5; b = 3; # สร้างตัวแปร a และ b เป็นชนิดข้อมูลจำนวนเต็ม
2 c = a + b # บวกค่าตัวแปร a กับ b
3 d = a / b # หารค่าตัวแปร a กับ b
4 x = a ** b # ยกกำลังค่าตัวแปร a กับ b
5 z = a // b # หารเอาเศษค่าตัวแปร a กับ b
6 print('ผลลัพธ์ของ a + b คือ ', c) # แสดงผลลัพธ์ที่ค่าตัวแปร c
7 print('ผลลัพธ์ของ a / b คือ ', d) # แสดงผลลัพธ์ค่าตัวแปร d
8 print('ผลลัพธ์ของ a ** b คือ ', x) # แสดงผลลัพธ์ค่าตัวแปร x
9 print('ผลลัพธ์ของ a // b คือ ', z) # แสดงผลลัพธ์ค่าตัวแปร z
```

ผลลัพธ์ของ a + b คือ 8

ผลลัพธ์ของ a / b คือ 1.6666666666666667

ผลลัพธ์ของ a ** b คือ 125

ผลลัพธ์ของ a // b คือ 1



3.14 ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการเปรียบเทียบ (Comparison Operator) จะทำการเปรียบเทียบค่าข้อมูลที่ละ 2 ตัว ด้วยเครื่องหมายการเปรียบเทียบ ดังแสดงในตาราง 3.2 ผลลัพธ์ที่ได้จะมีเพียง 2 ค่าคือ จริง (True) และเท็จ (False) ซึ่งเป็นรูปแบบของตรรกะ

จากตัวอย่างที่ 3.23 แสดงการเขียนคำสั่งโปรแกรมโดยใช้ตัวดำเนินการเปรียบเทียบแบบต่าง ๆ ผู้อ่านจะสังเกตเห็นว่าในบรรทัดที่ 9 สามารถประยุกต์ใช้ตัวดำเนินการเปรียบเทียบกับตัวถูกดำเนินการได้มากกว่า 2 ตัว โดยการทำงานจะเปรียบเทียบตัวถูกดำเนินการคู่ทางด้านซ้ายมือก่อน จากนั้นจึงเปรียบเทียบกับตัวถูกดำเนินการทางด้านขวามือ เช่น `b == d` หรือไม่ ผลลัพธ์คือ จริง และ `d < a` หรือไม่ คำตอบคือ จริง ทำให้ผลลัพธ์การเปรียบเทียบเป็นจริง

ตัวอย่าง 3.24

การใช้งานตัวดำเนินการเปรียบเทียบและผลลัพธ์

```
1 a = 5; b = 3; c = 15; d = 3 # สร้างตัวแปร a, b, c และ d เป็นชนิดข้อมูล
   ↳ จำนวนเต็ม
2 w = (a == c) # เปรียบเทียบค่าตัวแปร a เท่ากับ c หรือไม่
3 x = (d <= a) # เปรียบเทียบค่าตัวแปร d น้อยกว่าหรือเท่ากับ a หรือไม่
4 y = (b >= c) # เปรียบเทียบค่าตัวแปร b มากกว่าหรือเท่ากับ c หรือไม่
5 z = (b == d < a) # เปรียบเทียบค่าตัวแปร b เท่ากับ d และน้อยกว่า a หรือไม่
6 print('a เท่ากับ c หรือไม่ = ', w) # แสดงผลลัพธ์ที่เก็บอยู่ในค่าตัวแปร w
7 print('d น้อยกว่าหรือเท่ากับ a หรือไม่ = ', x) # แสดงผลลัพธ์ค่าตัวแปร x
8 print('b มากกว่าหรือเท่ากับ c หรือไม่ = ', y) # แสดงผลลัพธ์ค่าตัวแปร y
9 print('b เท่ากับ d และน้อยกว่า a หรือไม่ = ', z) # แสดงผลลัพธ์ค่าตัวแปร z
```

a เท่ากับ c หรือไม่ = False
d น้อยกว่าหรือเท่ากับ a หรือไม่ = True
b มากกว่าหรือเท่ากับ c หรือไม่ = False
b เท่ากับ d และน้อยกว่า a หรือไม่ = True



สัญลักษณ์	ความหมาย	ตัวอย่างการใช้	ผลลัพธ์ (a=5, b=3)
<code>==</code>	เท่ากับ (Equal)	<code>a == b</code>	False
<code>!=</code>	ไม่เท่ากับ (Not Equal)	<code>a != b</code>	True
<code><</code>	น้อยกว่า (Less than)	<code>a < b</code>	False
<code><=</code>	น้อยกว่าหรือเท่ากับ (Less than or Equal)	<code>a <= b</code>	False
<code>></code>	มากกว่าหรือเท่ากับ (Greater than or Equal)	<code>a >= b</code>	True

ตาราง 3.2: สัญลักษณ์ตัวดำเนินการเปรียบเทียบ

3.15 ตัวดำเนินการกำหนดค่า

ตัวดำเนินการกำหนดค่า (Assignment Operator) เป็นตัวดำเนินการที่ทำหน้าที่กำหนดค่าข้อมูล หรือข้อมูลที่เก็บไว้ในตัวแปรที่อยู่ทางด้านขวามือให้กับตัวแปรที่อยู่ทางด้านซ้ายมือ มีสัญลักษณ์ของตัวดำเนินการประเภนี้แสดงในตาราง 3.3

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้	ผลลัพธ์ (a=5, b=3)
=	กำหนดค่า	a = b นำค่าตัวแปร b มาใส่ ในตัวแปร a	a = 3
+=	บวกก่อนแล้ว กำหนดค่า	a += b คำนวณ a + b แล้ว เอาผลมาเก็บไว้ที่ a	a = 8
-=	ลบก่อนแล้ว กำหนดค่า	a -= b คำนวณ a - b แล้ว เอาผลมาเก็บไว้ที่ a	a = 2
*=	คูณก่อนแล้ว กำหนดค่า	a *= b คำนวณ a * b แล้ว เอาผลมาเก็บไว้ที่ a	a = 15
/=	หารก่อนแล้ว กำหนดค่า	a /= b คำนวณ a / b แล้ว เอาผลมาเก็บไว้ที่ a	a = 1.667
%=	หารเอาเศษแล้ว กำหนดค่า	a %= b คำนวณ a % b แล้ว เอาผลมาเก็บไว้ที่ a	a = 2
**=	ยกกำลังแล้ว กำหนดค่า	a **= b คำนวณ a ** b แล้ว เอาผลมาเก็บไว้ที่ a	a = 125
//=	หารเอาส่วนแล้ว กำหนดค่า	a //= b คำนวณ a // b แล้ว เอาผลมาเก็บไว้ที่ a	a = 1

ตาราง 3.3: สัญลักษณ์ตัวดำเนินการกำหนดค่า

ตัวอย่าง 3.25

การใช้งานตัวดำเนินการกำหนดค่าและผลลัพธ์

```

1 a = 9; b = 5; c = 4; d = 6 # สร้างตัวแปร a, b, c และ d เป็นชนิดข้อมูล
  ↳ จำนวนเต็ม
2 a = b # กำหนดค่าตัวแปร a เท่ากับ b
3 c += b # บวกแล้วเก็บผลลัพธ์ไว้ที่ตัวแปร c
4 d -= b # ลบแล้วเก็บผลลัพธ์ไว้ที่ตัวแปร b
5 print('ผลลัพธ์ a = b :', a) # แสดงผลลัพธ์ค่าตัวแปร a
6 print('ผลลัพธ์ c += b :', c) # แสดงผลลัพธ์ค่าตัวแปร c
7 print('ผลลัพธ์ d -= b :', d) # แสดงผลลัพธ์ค่าตัวแปร d

```

```

ผลลัพธ์ a = b : 5
ผลลัพธ์ c += b : 9
ผลลัพธ์ d -= b : 1

```



3.16 ตัวดำเนินการตรรกศาสตร์

ตัวดำเนินการตรรกศาสตร์ (Logical Operator) เป็นตัวดำเนินการที่ใช้เปรียบเทียบระหว่างตัวถูกดำเนินการ 2 ตัว เพื่อตัดสินใจว่าเป็นจริง (**True**) หรือเป็นเท็จ (**False**) มีตัวดำเนินการประเภทนี้ในภาษาโปรแกรมไพธอนให้ใช้งานอยู่ 3 ชนิด ได้แก่ **and**, **or** และ **not** สามารถนำตารางความจริง (Truth table) มาประยุกต์เปรียบเทียบระหว่างตัวถูกดำเนินการได้

ในตาราง 3.5 เราได้แสดงค่าความจริงเมื่อนำตัวดำเนินการตรรกศาสตร์มาดำเนินการกับตัวถูกดำเนินการ p และ q เมื่อใช้ตัวดำเนินการ **and** จะได้ผลลัพธ์ **True** เมื่อ p และ q เป็น **True** เท่านั้น แต่หากใช้ตัวดำเนินการ **or** จะได้ผลลัพธ์ **False** เมื่อ p และ q เป็น **False** ทั้งสอง สำหรับการใช้ตัวดำเนินการ **not** เป็นการเปลี่ยนค่าตรงข้าม **True** เป็น **False** และ **False** เป็น **True**

ตัวดำเนินการ	ความหมาย	อธิบายการใช้งาน	ผลลัพธ์ (a=True, b=False)
and	และ	a and b	False
or	หรือ	a or b	True
not	นิเสธ (ไม่)	not a	False
		not b	True

ตาราง 3.4: ตัวดำเนินการตรรกศาสตร์ในภาษาโปรแกรมไพธอน

p	q	p and q	p or q	not p	not q
True	True	True	True	False	False
True	False	False	True	False	True
False	True	False	True	True	False
False	False	False	False	True	True

ตาราง 3.5: ตารางความจริง (Truth Table)

ตัวอย่าง 3.26

การใช้งานตัวดำเนินการตรรกศาสตร์และการประยุกต์ใช้เขียนคำสั่งโปรแกรม

```
1 a = 5; b = 3; c = 15 # สร้างตัวแปร a, b และ c เป็นชนิดข้อมูลจำนวนเต็ม
2 # ใช้ตัวดำเนินการเปรียบเทียบและตรรกศาสตร์เปรียบเทียบค่าตัวแปร
3 x = a == b and c < b
4 y = b > a or a > c
5 z = not (b > a or c < a)
6
7 print('ผลลัพธ์ของ a == b and c < b คือ ', x) # แสดงผลลัพธ์ตัวแปร x
8 print('ผลลัพธ์ของ b > a or a > c คือ ', y)   # แสดงผลลัพธ์ตัวแปร y
9 print('ผลลัพธ์ของ not (b > a or c < a) คือ ', z) # แสดงผลลัพธ์ตัวแปร z
```

ผลลัพธ์ของ a ==b and c < b คือ False
ผลลัพธ์ของ b > a or a > c คือ False
ผลลัพธ์ของ not (b > a or c < a) คือ True



3.17 ตัวดำเนินการระดับบิต

ตัวดำเนินการระดับบิต (Bitwise Operator) เป็นตัวดำเนินการเปรียบเทียบข้อมูลในระดับบิตที่มีค่า **0** และ **1** โดยไม่คำนึงถึงว่าชนิดข้อมูลเป็นอะไรภาษาโปรแกรมไพธอนจะแปลงข้อมูลให้อยู่ในรูปแบบเลขฐานสอง คือ **0** และ **1** ก่อน แล้วจึงนำไปดำเนินการหาผลลัพธ์การดำเนินการระหว่างตัวดำเนินการและตัวถูกดำเนินการจะให้ค่าเป็นเท็จ (**0**) หรือจริง (**1**) ผู้อ่านสามารถเปรียบเทียบการคำนวณได้กับตาราง 3.4 สำหรับสัญลักษณ์ต่าง ๆ ที่ใช้ดำเนินการระดับบิตแสดงในตารางที่ 3.6

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้	ผลลัพธ์ (a=5, b=3)
&	และ (and)	a & b	a = 0101, b = 0011 (a & b = 1)
	หรือ (or)	a b	a = 0101, b = 0011 (a b = 7)
^	xor	a ^ b	a = 0101, b = 0011 (a ^ b = 6)
~	คอมพลีเมนต์	~a	a = 0101, b = 0011 (~a = -6, ~b = 4)
<<	เลื่อนบิตไปทางซ้าย	a << b	a=0101, b = 0011 (a << 1 = 10, b << 1 = 6)
>>	เลื่อนบิตไปทางขวา	a >> b	a = 0101, b = 0011 (a >> 1 = 2, b >> 1 = 1)

ตาราง 3.6: สัญลักษณ์ตัวดำเนินการระดับบิตที่ใช้ในภาษาโปรแกรมไพธอน

ตัวอย่าง 3.27

การใช้งานตัวดำเนินการระดับบิต

```

1  a = 5
2  b = 3
3  c = 5
4  x = a & b
5  print('ผลลัพธ์ของ a & b คือ ', x)
6  y = a ^ c
7  print('ผลลัพธ์ของ a ^ c คือ', y)
8  z = a } b
9  print('ผลลัพธ์ของ a } b คือ ', z)'

```

ผลลัพธ์ของ a & b คือ 1
 ผลลัพธ์ของ a ^ b คือ 0
 ผลลัพธ์ของ a } b คือ 7



3.18 ตัวดำเนินการตรวจสอบสมาชิก

ตัวดำเนินการตรวจสอบสมาชิก (Membership Operator) ใช้สำหรับดำเนินการเปรียบเทียบ ค้นหาที่กำหนดในตัวแปรที่สนใจ เป็นสมาชิกในตัวแปรที่จะทำการเปรียบเทียบหรือไม่ ใช้กับ ชนิดข้อมูลสตริง (String), ลิสต์ (List) และ ทูเพิล (Tuple) ซึ่งเราจะกล่าวถึงชนิดข้อมูลนี้ใน บทที่ 4 และ 5 โดยละเอียด

ในตาราง 3.7 เราได้แสดงตัวดำเนินการตรวจสอบสมาชิก และตัวอย่างการใช้งาน เมื่อ กำหนดให้

```
cars = ['Honda', 'Toyota', 'BMW', 'Mercedes-Benz' ,
↪ 'Nissan' , 'Mazda' ]
```

และ

```
x = 'Ford'
```

สัญลักษณ์	ความหมาย	การใช้งาน	อธิบายผลลัพธ์
in	เป็นสมาชิก	x in cars	ให้ค่าเป็น False เพราะ 'Ford' ไม่มีอยู่ในลิสต์ของ cars
not in	ไม่เป็นสมาชิก	x not in cars	ให้ค่าเป็น True เพราะ 'Ford' ไม่มีอยู่ในลิสต์ของ cars

ตาราง 3.7: ตัวดำเนินการตรวจสอบสมาชิก

ตัวอย่าง 3.28

การใช้งานตัวดำเนินการตรวจสอบสมาชิก

```

1 cars = ['Honda', 'Toyota', 'BMW', 'Mercedes-Benz', 'Nissan',
  → 'Mazda' ]
2 my_car = 'Ford'
3 x = my_car in cars
4 y = my_car not in cars
5 print('ผลลัพธ์ของ my_car in cars =', x)
6 print('ผลลัพธ์ของ my_car not in cars =', y)

```

ผลลัพธ์ของ my_car in cars = False
 ผลลัพธ์ของ my_car not in cars = True



3.19 ตัวดำเนินการเอกลักษณ์

ตัวดำเนินการเอกลักษณ์ (Identity Operator) เป็นตัวดำเนินการที่นำมาใช้เพื่อเปรียบเทียบข้อมูลระหว่างตัวแปร 2 ตัว มีให้เลือกใช้งานอยู่ 2 ตัวคือ **is** และ **is not** เพื่อบอกว่ามีความเท่ากันหรือไม่เท่ากัน ตัวดำเนินการ **is** จะให้คำตอบเหมือนกับการใช้ตัวดำเนินการ **==** ส่วน **is not** ให้คำตอบเหมือนกับการใช้ตัวดำเนินการ **!=**

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้	ผลลัพธ์ (a=5, b=3)
<code>is</code>	เป็น, อยู่	<code>a is b</code>	ให้ค่าเป็น False เพราะ a ไม่เท่ากับ b
<code>is not</code>	ไม่เป็น, ไม่อยู่	<code>a is not b</code>	ให้ค่าเป็น True เพราะ a ไม่เท่ากับ b

ตาราง 3.8: ตัวดำเนินการเอกลักษณ์ที่ใช้ในภาษาโปรแกรมไพธอน

ตัวอย่าง 3.29

การใช้งานตัวดำเนินการเอกลักษณ์เปรียบเทียบค่าตัวแปร

```

1 a = 5; b = 3
2 x = a is b
3 y = a is not b
4 print('ผลลัพธ์ของ a is b คือ ', x)
5 print('ผลลัพธ์ของ a is not b คือ ', y)

```

ผลลัพธ์ของ `a is b` คือ False
 ผลลัพธ์ของ `a is not b` คือ True



3.20 ลำดับความสำคัญตัวดำเนินการ

ในหนึ่งนิพจน์อาจจะมีมากกว่าหนึ่งตัวดำเนินการแต่ละตัวดำเนินการมีลำดับความสำคัญในการทำงานก่อนหลัง (Operator precedence) แตกต่างกัน บางตัวดำเนินการมีความสำคัญในการทำงานที่เท่ากัน เราจึงมีความจำเป็นที่จะต้องทราบถึงลำดับการทำงานของตัวดำเนินการเพื่อนำไปเขียนเป็นคำสั่งโปรแกรมให้ทำงานได้ถูกต้อง และป้องกันข้อผิดพลาดจากการทำงานของโปรแกรม

การหาคำตอบจากนิพจน์เริ่มจากทางด้านซ้ายไปทางด้านขวาเสมอ ในทางคอมพิวเตอร์ก็ทำแบบนี้เช่นเดียวกัน แต่มีลำดับความสำคัญของเครื่องหมายเข้ามาเกี่ยวข้อง เพื่อควบคุมการ

ลำดับ	ตัวดำเนินการ	คำอธิบาย
1	(...)	ใช้เพื่อแบ่งนิพจน์และลำดับการทำงาน
2	**	สัญลักษณ์ยกกำลัง
3	~, +, -	คอมพลีเมนต์, unary plus, unary minus
4	*, /, %, //	การคูณ, การหาร, การหารเอาเศษ, การหารเอาส่วน
5	+, -	การบวกและการลบ
6	>>, <<	การเลื่อนบิตทางขวา, การเลื่อนบิตทางซ้าย
7	&	AND ในระดับบิต
8	^,	xor, or ในระดับบิต
9	<=, <, >, >=	ตัวดำเนินการเปรียบเทียบ
10	==, !=	เท่ากับและไม่เท่ากับ
11	=, %=, /=, //=, -=, +=, *=, **=	ตัวดำเนินการกำหนดค่า
12	is, is not	อยู่ ไม่อยู่ ตัวดำเนินการเอกลักษณ์
13	in, not in	อยู่ใน ไม่อยู่ใน ตัวดำเนินการความเป็นสมาชิก
14	not, or, and	ตัวดำเนินการตรรกศาสตร์

ตาราง 3.9: ลำดับความสำคัญของตัวดำเนินการ

หาผลลัพธ์จำเป็นต้องใช้เครื่องหมายวงเล็บ (...) ครอบนิพจน์ที่เราต้องการให้ดำเนินการก่อน เพราะเครื่องหมายวงเล็บมีลำดับความสำคัญที่สุดให้ผู้อ่านพิจารณาจากตัวอย่างต่อไปนี้

ตัวอย่าง 3.30

การเขียนคำสั่งโปรแกรมเปรียบเทียบการจัดลำดับความสำคัญของตัวดำเนินการ

```
1 x = 10 + 4 / 2 * 9
2 y = ((10 + 4) / 2) * 9
3 print('ผลลัพธ์ของ x = 10 + 4 / 2 * 9 คือ ', x)
4 print('ผลลัพธ์ของ ((10 + 4) / 2) * 9 คือ ', y)
```

ผลลัพธ์ของ $x = 10 + 4 / 1 * 9$ คือ 28.0

ผลลัพธ์ของ $((10+4) / 2) * 9$ คือ 63.0



จากตัวอย่างการเขียนคำสั่งโปรแกรม แสดงให้เห็นถึงลำดับการประมวลผลของตัวดำเนินการในบรรทัด 1 จะเริ่มจากดำเนินการหารก่อน คือ $4 / 2$ ได้เท่ากับ 2 จากนั้นคูณกับ 9 ได้เท่ากับ 18 แล้วบวกกับ 10 ผลลัพธ์ที่ได้จึงเท่ากับ 28.0

ในบรรทัดที่ 2 มีการจัดลำดับการประมวลผลโดยใช้เครื่องหมายวงเล็บ โดยเริ่มจากวงเล็บด้านในสุดคือ $10+4$ ได้เท่ากับ 14 แล้วจึงหารด้วย 2 ได้เท่ากับ 7 จากนั้นคูณกับ 7 จากนั้นคูณกับ 9 ผลลัพธ์ที่ได้จึงเท่ากับ 63.0

สรุปก่อนจบบท

ในบทนี้ผู้อ่านได้รู้จักกับชนิดข้อมูลต่าง ๆ การสร้างตัวแปรและกำหนดค่าข้อมูลก่อนนำไปใช้งาน การแปลงชนิดข้อมูลหนึ่งให้เป็นอีกชนิดข้อมูลหนึ่ง นอกจากนี้ยังได้รู้จักฟังก์ชันที่ใช้สำหรับแปลงเลขฐานสอง ฐานแปด ฐานสิบ และฐานสิบหก และยังได้รู้จักกับฟังก์ชันที่ใช้ตรวจสอบชนิดข้อมูลกับนิพจน์และตัวดำเนินการประเภทต่าง ๆ เช่น ตัวดำเนินการทางคณิตศาสตร์ ตัวดำเนินการเปรียบเทียบ เป็นต้น ซึ่งช่วยให้ผู้อ่านนำไปประยุกต์สำหรับการพัฒนาโปรแกรม

แบบฝึกหัด

1. จงบอกชนิดข้อมูลจากการประกาศตัวแปรดังต่อไปนี้

```

1 a = 'This is a book.'
2 b = 9 + 8 + 7 + 6 + 5
3 c = (b ** 2) / 6
4 d = b <= 6 * c
5 e = 2.71828 * 10000
6 f = 9.0
7 g = 'Bye ' * b
8 h = 2 * 3.06456E-4

```

2. จงเขียนโปรแกรมคำนวณภาษีที่ต้องจ่าย โดยมีอัตราภาษีเท่ากับ 7% กำหนดให้ตัวแปรอัตราภาษีเป็นค่าคงที่ และให้โปรแกรมรับยอดเงินรายได้สุทธิผ่านทางแป้นพิมพ์
3. จงเขียนโปรแกรมคำนวณอายุปัจจุบัน โดยให้โปรแกรมรับข้อมูลวันเกิดผ่านทางแป้นพิมพ์
4. จงเขียนผลลัพธ์ที่ได้จากโปรแกรมต่อไปนี้

```

1 x = 345
2 print(bin(x))
3 print(oct(x))
4 print(hex(x))

```

5. จงเขียนผลลัพธ์ที่ได้จากโปรแกรมต่อไปนี้

```

1 x = 125 + 315 / 3 + 5 * 2 + 15 * 4
2 y = 451 + 315 / 3 + 5 * 2 * 3 + 15 * 4 * 12
3 print(x)
4 print(y)

```