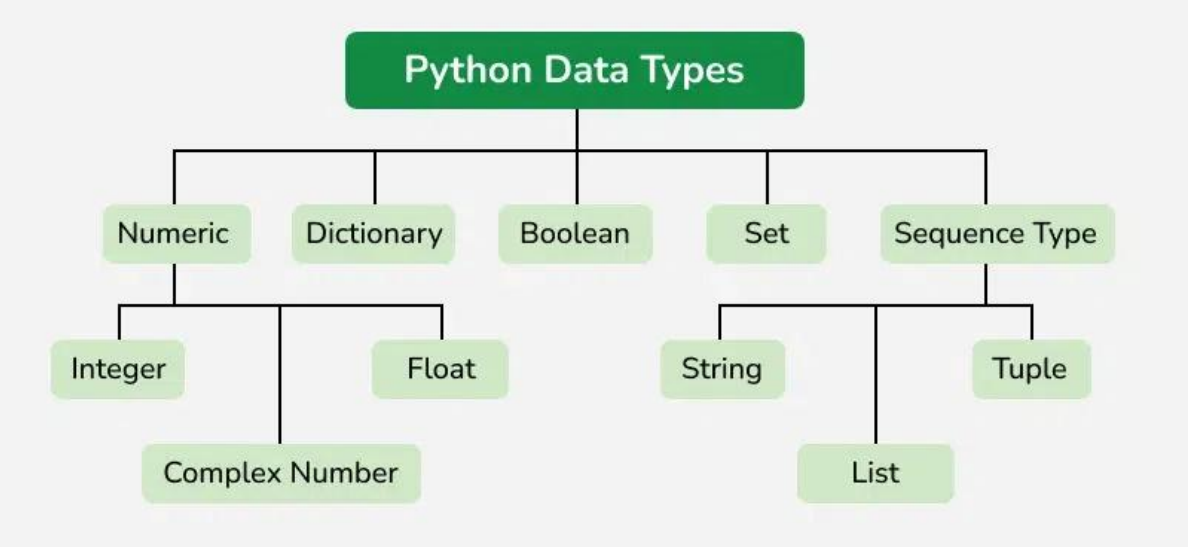
หน่วยที่ 3 ข้อมูลพื้นฐานภาษาไพทอน

ข้อมูลพื้นฐานภาษาไพทอน

ก่อนที่จะเริ่มต้นการเขียนโปรแกรมไม่ว่าภาษาใดก็ตาม จะต้องศึกษาถึงข้อมูลพื้นฐานซึ่งถือว่าเป็นหน่วยที่ย่อยที่สุดของภาษาแต่ละภาษา ภาษาไพทอนก็เช่นเดียวกัน เราควรจะต้องรู้จักข้อมูลพื้นฐานต่าง ๆ ที่ต้องรู้ก่อนการเขียนโปรแกรมดังนี้

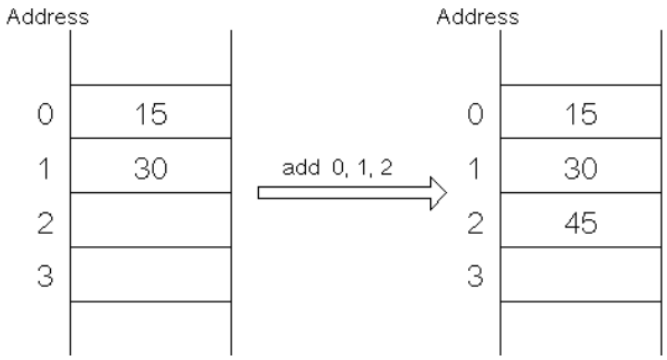
1. ตัวแปร
2. กฎการตั้งชื่อของภาษาไพทอน
3. การกำหนดค่าให้กับตัวแปร
4. ชนิดของข้อมูล
5. การตรวจสอบชนิดของข้อมูล
6. การแปลงชนิดของข้อมูล



1. ตัวแปร (Variable)

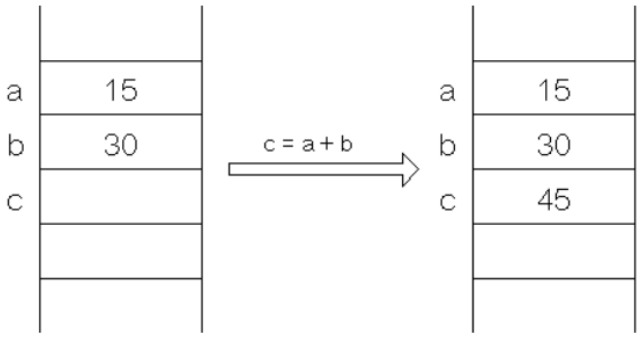
ในการเขียนโปรแกรมคอมพิวเตอร์กระบวนการสำคัญที่เกิดขึ้น คือ การรับข้อมูล การประมวลผลข้อมูล และการแสดงผลข้อมูล จะเห็นวาสิ่งที่เป็นส่วนสำคัญที่สุด คือ ข้อมูล การทำงานของโปรแกรมขณะใดขณะหนึ่งจะตองมีการเก็บข้อมูลไวในคอมพิวเตอร์ โดยรับข้อมูลจากอุปกรณ์รับข้อมูลไปเก็บไวในสวนที่เรียกวา หน่วยความจำ และส่งข้อมูลจากหน่วยความจำไปประมวลผลในหน่วยประมวลผลกลาง โดยผ่านคำสั่งต่าง ๆ เมื่อประมวลผลเสร็จแลวก็นำผลลัพธ์ที่ได้กลับมาเก็บไวที่หน่วยความจำอีก เมื่อตองการใหแสดงผลก็จะใชคำสั่งใหไปอ่านข้อมูลจากหน่วยความจำส่งข้อมูลนั้นไปยังอุปกรณ์แสดงผล

พิจารณาจากตัวอย่างการบวกเลข แสดงดังรูปที่ 3.1 โดยให้รับค่าจำนวนเต็ม 2 ค่าจากผู้ใช้ แล้วนำไปเก็บอยู่ในหน่วยความจำตำแหน่งที่ 0 มีค่า 15 ตำแหน่งที่ 1 มีค่า 30 หากต้องการบวกจะนำค่าทั้ง 2 มาบวกกันและเก็บไว้ที่ตำแหน่งที่ 2 จะต้องใช้คำสั่งในการบวก สมมติดังตัวอย่าง Add 0 , 1 , 2 คือ การเอาค่า ณ ตำแหน่งที่ 0 และ 1 มาบวกัน และนำไปเก็บไว้ที่ตำแหน่งที่ 2 จะได้ว่า มีการอ่านค่า 15 จากตำแหน่งที่ 0 ไปบวกกับค่า 30 จากตำแหน่งที่ 1 ได้ผลลัพธ์ คือ 45 นำไปเก็บไว้ในตำแหน่งที่ 2



รูปที่ 3.1 แสดงแบบจำลองการบวกเลขในคอมพิวเตอร์

ในความเป็นจริงตำแหน่งต่าง ๆ ไม่ได้อ้างด้วยเลข 0 1 หรือ 2 แต่อ้างด้วยระบบการอ้างที่อยู่ ซึ่งเป็นสิ่งซับซ้อนกว่านั้น การเขียนในลักษณะที่เห็นในตัวอย่างเป็นการเขียนในภาษาคอมพิวเตอร์ยุคแรก เช่น ภาษาแอสแซมบลี (Assembly) เพื่อความสะดวกในการอ้างถึงข้อมูล ณ ตำแหน่งต่าง ๆ ในหน่วยความจำ จึงได้มีการพัฒนากันต่อมา จนกระทั่งเป็นการอ้างในระบบของชื่อตัวแปร (Variable) ที่ใช้ในปัจจุบัน พิจารณาจากตัวอย่างที่ใช้ระบบชื่อตัวแปร แสดงดังรูปที่ 3.2 ผู้เขียนโปรแกรมสร้างหรือกำหนดตัวแปรขึ้นมา จะเกิดการจองพื้นที่ในหน่วยความจำให้โดยอัตโนมัติ ผู้ใช้ไม่จำเป็นต้องรู้ว่า ตัวแปรนั้นอยู่ที่ตำแหน่งใด การอ้างถึงชื่อตัวแปร คือ การอ้างถึงข้อมูลในหน่วยความจำ เพราะฉะนั้นหากผู้ใช้กำหนดตัวแปร a และ b เพื่อเก็บข้อมูล 15 และ 30 ตามลำดับ เมื่อต้องการหาผลบวกและนำค่าไปเก็บไว้ในที่หนึ่ง ก็สร้างหรือกำหนดตัวแปรขึ้นมาเพื่อเก็บผลลัพธ์ สมมติชื่อว่า c การสั่งให้ทำงานก็สามารถทำได้โดยใช้คำสั่ง c = a + b ซึ่งอ่านเข้าใจได้ง่ายกว่าคำสั่ง add 0 , 1 , 2



รูปที่ 3.2 แสดงแบบจำลองการบวกเลขโดยอ้างชื่อตัวแปร

ตัวแปร (Variable) หมายถึง ชื่อที่กำหนดขึ้นเพื่อใช้อ้างอิงค่าข้อมูลในตำแหน่งบนหน่วยความจำ ซึ่งค่าข้อมูลนี้สามารถนำไปเรียกใช้ในโปรแกรมได้ หรือใช้สำหรับเก็บค่าข้อมูลระหว่างการประมวลผล ซึ่งอาจเป็นค่าข้อมูลนำเข้า ค่าข้อมูลที่เกิดจากการดำเนินการ หรือเก็บค่าข้อมูลผลลัพธ์ที่ต้องการแสดงผล

2. กฎการตั้งชื่อของภาษาไพทอน

การเขียนโปรแกรมในบางครั้งอาจต้องมีการตั้งชื่อต่าง ๆ ให้กับตัวแปร ฟังก์ชัน คลาส หรือเมธอด เป็นต้น การตั้งชื่อใด ๆ ก็ตาม จะต้องเป็นไปตามกฎการตั้งชื่อของภาษาไพทอน ซึ่งภาษาไพทอนกำหนดกฎในการตั้งชื่อไว้ มีดังนี้

1. ต้องขึ้นต้นด้วยตัวอักษร หรือเครื่องหมาย Underscore (_) เท่านั้น แต่ห้ามขึ้นต้นด้วยตัวเลข
2. ชื่อที่ตั้งขึ้นจะต้องประกอบด้วยตัวอักษร เครื่องหมาย Underscore (_) หรือตัวเลข 0 – 9 เท่านั้น
3. ภายในชื่อห้ามเว้นช่องว่าง หรือห้ามใช้สัญลักษณ์พิเศษ ตัวอย่างสัญลักษณ์พิเศษ เช่น + – * / % : . : ; , ‘ “ ^ = < > ( ) [ ] { } ! @ # $ % & ? ฿ | \ ~ เป็นต้น
4. ชื่อที่ตั้งตัวอักษรตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ ถือว่าเป็นคนละตัวกัน (Case-sensitive) เช่น CAT, CaT, Cat หรือ cat ทั้งหมดนี้ถือว่าเป็นชื่อที่แตกต่างกันหรือเป็นคนละชื่อ
5. ชื่อที่ตั้งจะมีความยาวเท่าใดก็ได้ ไม่จำกัดความยาวของตัวอักษร
6. ชื่อที่ตั้งควรสื่อความหมาย เช่น กำหนดชื่อตัวแปรว่า pi จะรู้ได้ทันทีว่าตัวแปรนี้เก็บค่าไพน์ที่มีค่าเท่ากับ 3.14 หรือกำหนดชื่อตัวแปรว่า price จะรู้เลยว่าตัวแปรนี้เก็บค่าราคา เป็นต้น
7. ห้ามตั้งชื่อซ้ำกับคำสงวน (Reserved Word)

คำสงวน (Reserved Words) หมายถึง คำเฉพาะที่ถูกสงวนไว้ ห้ามนำไปใช้ในการตั้งชื่อใด ๆ โดยคำสงวนในภาษาไพทอน มีดังนี้

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

ตัวอย่าง การตั้งชื่อตัวแปร

ชื่อตัวแปรที่ถูกต้อง	
h	Grade
Num1	LEB8
_SUM	SUM_Nagative
MaxNum	Math_2
average	ABC_3

ชื่อตัวแปรที่ไม่ถูกต้อง	
A+B (มีเครื่องหมาย +)	not (เป็นคำสงวน)
My var (มีช่องว่าง)	LEB.First (มีสัญลักษณ์พิเศษ .)
&cost (มีสัญลักษณ์พิเศษ &)	_Data-2 (มีสัญลักษณ์พิเศษ -)
9SUM (ขึ้นต้นด้วยตัวเลข)	Hello! (มีสัญลักษณ์พิเศษ !)
SUM@A (มีสัญลักษณ์พิเศษ @)	in (เป็นคำสงวน)

3. การกำหนดค่าให้กับตัวแปร (Assignment Operators)

ตัวดำเนินการกำหนดค่าเป็นตัวดำเนินการพื้นฐานที่ใช้สำหรับสั่งให้คอมพิวเตอร์คำนวณ หรืออาจใช้ในการกำหนดค่าเริ่มต้นให้กับตัวแปร หรือย้ายค่าตัวแปรจากที่หนึ่งไปยังอีกที่หนึ่งก็ได้ โดยใช้เครื่องหมาย **เท่ากับ (=)**

ความหมายของตัวดำเนินการกำหนดค่า คือ การกำหนดค่าที่อยู่ทางขวามือของคำสั่งให้กับตัวแปรที่อยู่ทางซ้ายมือ หรือ การกำหนดค่าที่อยู่ทางขวามือของเครื่องหมายเท่ากับ (=) ให้กับตัวแปรที่อยู่ทางซ้ายมือของเครื่องหมายเท่ากับ (=) ดังนี้

รูปแบบ การกำหนดค่าให้กับตัวแปร

ชื่อตัวแปร = ค่าที่ต้องการกำหนดให้กับตัวแปร

มีรูปแบบต่าง ๆ ดังนี้

ตัวแปร = ตัวแปร | **ตัวแปร = ค่าคงที่** | **ตัวแปร = นิพจน์**

ความหมาย

เครื่องหมาย = (เท่ากับ) ในภาษาไททอน หมายถึง การกำหนดค่าที่อยู่ด้านขวา ให้กับตัวแปรด้านซ้าย

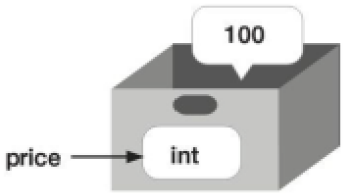
ตัวแปร คือ การนำค่าตัวแปรฝั่งขวาไปเก็บในตัวแปรฝั่งซ้าย

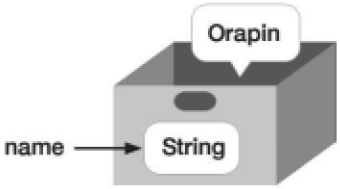
ค่าคงที่ คือ การนำค่าคงที่ไปเก็บในตัวแปรฝั่งซ้าย

นิพจน์ คือ การนำค่าผลลัพธ์การประมวลผลจากนิพจน์ ไปเก็บในตัวแปรฝั่งซ้าย

ตัวอย่าง เช่น

ตัวแปร = ตัวแปร	ตัวแปร = ค่าคงที่	ตัวแปร = นิพจน์
a = b	price = 100	area = 1 / 2 * b * h
i = j	age = 13	speed = distance / time
x = y	gpa = 3.98	total = 60 * hours + minutes
age = z	j = 120.00	pi = 22 / 7
P = Q	name = “Orapin”	average = (a + b + c) / 3





นอกจากนี้ยังมีรูปแบบเพิ่มเติม หากต้องการกำหนดค่าให้กับตัวแปรหลายตัวด้วยค่าเดียวกัน มีรูปแบบดังนี้

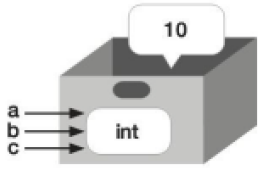
รูปแบบ

ตัวแปร1 = ตัวแปร2 = ... = ค่าคงที่
ตัวแปร1 = ตัวแปร2 = ... = ตัวแปร
ตัวแปร1 = ตัวแปร2 = ... = นิพจน์

ตัวอย่าง เช่น

a = b = c = 10 แปลความหมายได้ว่า

c = 10
b = c
a = b



หรือ

a = b = c = d แปลความหมายได้ว่า

c = d
b = c
a = b

หรือ

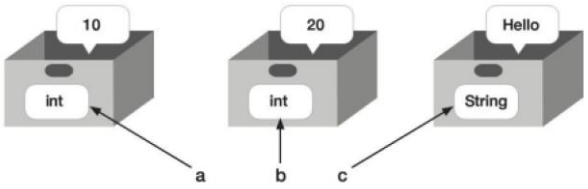
i = j = k = 2 + 3 แปลความหมายได้ว่า

k = 5
j = k
i = j

หรืออีกอย่าง เช่น

a , b , c = 10 , 20 , "Hello" แปลความหมายได้ว่า

a = 10
b = 20
c = "Hello"



หรืออีกอย่าง เช่น

a , b = 5 , 10
temp , a , b = a , b , temp แปลความหมายได้ว่า

a = 5
b = 10
temp = a
a = b
b = temp

ผลรัน
a = ?
b = ?

4. ชนิดของข้อมูล (Data Type)

ชนิดของข้อมูล คือ สิ่งที่ใช้กำหนดลักษณะและขอบเขตของข้อมูลนั้น ๆ โดยข้อมูลที่มีชนิดของข้อมูลแตกต่างกันก็จะเก็บข้อมูลได้ในลักษณะที่แตกต่างกัน และขอบเขตของข้อมูลที่เก็บได้ก็จะไม่เท่ากันด้วย ซึ่งชนิดข้อมูลพื้นฐานในภาษาไพทอน แบ่งออกเป็น 5 ชนิด ดังนี้

1. ข้อมูลชนิดตัวเลข (Numeric) แบ่งออกได้เป็น 3 ประเภท ดังนี้

1.1 ข้อมูลเลขจำนวนเต็ม (Integer Number) คือ ข้อมูลที่ประกอบด้วยเลขจำนวนเต็มที่เป็นทั้งค่าบวก ค่าลบ ไม่มีทศนิยม นำไปคำนวณได้ และยังสามารถใช้กับเลขฐานได้ ดังนี้

- เลขฐานสิบ เขียนเป็นเลขปกติที่ใช้ในชีวิตประจำวัน เช่น 150 , 200 , 0 , -50 , -100
- เลขฐานสอง ใช้ตัวอักษร **0b** นำหน้า เช่น **0b**1010
- เลขฐานแปด ใช้ตัวอักษร **0o** นำหน้า เช่น **0o**12
- เลขฐานสิบหก ใช้ตัวอักษร **0x** นำหน้า เช่น **0x**A

ตัวอย่าง เช่น

เลขฐานสิบ	เลขฐานสอง	เลขฐานแปด	เลขฐานสิบหก
10	**0b**1010	**0o**12	**0x**A
-11	**–0b**1011	**–0o**13	**–0x**B
120	**0b**1111000	**0o**170	**0x**78
16	**0b**10000	**0o**20	**0x**10

1.2 ข้อมูลเลขจำนวนทศนิยม (Floating Point Number) หรือ เรียกว่า**เลขจำนวนจริง**ก็ได้

คือ ข้อมูลที่ประกอบด้วยเลขจำนวนที่เป็นทศนิยม มีเครื่องหมายมหัพภาค (**.**) หรือ จุด คั่นระหว่างตัวเลข นำไปคำนวณได้ และสามารถนำไปเขียนในรูปแบบของเลขยกกำลังสิบ (Exponential Form) ได้อีกด้วย ตัวอย่าง เช่น

10.00 , 150.25 , 0.00 , –120.50 , –10.50

2.34e2 มีค่าเท่ากับ 2.34×10^{2} หรือ มีค่าเท่ากับ 234.0

4.567e–2 มีค่าเท่ากับ 4.567×10^{-2} หรือ มีค่าเท่ากับ 0.04567

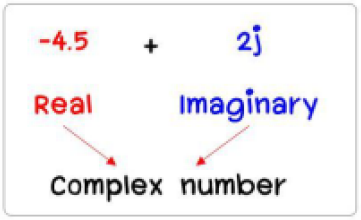
1.3 ข้อมูลเลขจำนวนเชิงซ้อน (Complex Number) คือ ข้อมูลที่ประกอบด้วยเลขจำนวนเชิงซ้อน โดยเลขจำนวนเชิงซ้อนเกิดจากการนำจำนวนจริง (Real) จำนวนจินตภาพ (Imaginary) มารวมกัน ซึ่งจะถูกใช้งานในทางวิทยาศาสตร์และวิศวกรรม

โดยรูปแบบการเขียนจำนวนเชิงซ้อน คือ **a + bi** โดยเรียก **a** ว่า **ส่วนของจำนวนจริง** และเรียก **b** ว่า **ส่วนของจำนวนจินตภาพ** ในไพทอนจะนำตัวอักษร j หรือ J มาแทน i ซึ่งก็คือ การนำมากำกับต่อท้ายค่าที่เป็นจำนวนจินตภาพ

ตัวอย่าง เช่น **–4.5 + 2j**

เรียก –4.5 ว่าเป็นส่วนของจำนวนจริง

เรียก 2j ว่าเป็นส่วนของจำนวนจินตภาพ



** Python จะรับรู้ชนิดข้อมูลแต่ละประเภทจากการที่ผู้เขียนโปรแกรมกำหนดค่าลงไปในชุดคำสั่ง ดังนั้น ผู้เขียนโปรแกรมจะต้องพึงระวังในการกำหนดค่าข้อมูลกับตัวแปร ให้ตรงตามชนิดข้อมูลที่ต้องการใช้งาน

ตัวอย่างข้อมูลชนิดตัวเลข เช่น

```
A = 10          #ตัวแปร A มีค่าเท่ากับ 10  มีชนิดเป็นจำนวนเต็ม
B = 0b1010      #ตัวแปร B เก็บข้อมูลเลขฐานสอง        มีค่าเท่ากับ 10  มีชนิดเป็นจำนวนเต็ม
C = 0o12        #ตัวแปร C เก็บข้อมูลเลขฐานแปด        มีค่าเท่ากับ 10  มีชนิดเป็นจำนวนเต็ม
D = 0xA         #ตัวแปร D เก็บข้อมูลเลขฐานสิบหก      มีค่าเท่ากับ 10  มีชนิดเป็นจำนวนเต็ม
E = 10.0        #ตัวแปร E มีค่าเท่ากับ 10.0       มีชนิดเป็นจำนวนทศนิยม
F = 10 + 2j     #ตัวแปร F มีค่าเท่ากับ 10 + 2j    มีชนิดเป็นจำนวนเชิงซ้อน
G = 15e–2       #ตัวแปร G มีค่าเท่ากับ 0.15       มีชนิดเป็นจำนวนทศนิยม
H = 35e+2       #ตัวแปร H มีค่าเท่ากับ 3500.0     มีชนิดเป็นจำนวนทศนิยม
```

2. ข้อมูลชนิดตรรกะ (Boolean) เป็นข้อมูลที่มีค่าความจริงเป็น**จริง**หรือเป็น**เท็จ**เท่านั้น ซึ่งจะแทนค่าจริงด้วย True และแทนค่าเท็จด้วย False ตัวอย่าง เช่น

```
A = 9 > 5           #ตัวแปร A มีค่าเท่ากับ True
B = 9 – 5 > 9 + 5   #ตัวแปร B มีค่าเท่ากับ False
C = True            #ตัวแปร C มีค่าเท่ากับ True
D = False           #ตัวแปร D มีค่าเท่ากับ False
E = 2 == 5          #ตัวแปร E มีค่าเท่ากับ False
```

กำหนดให้ A = 'B' , B = 'a' , C = –5.5 , D = 2

```
a = ( not ( A != 'A' ) and ( ( C // 2 ) > D ) )     #ตัวแปร a มีค่าเท่ากับ False
b = ( B <= 'T' ) or ( ( D – C ) > 0 )               #ตัวแปร b มีค่าเท่ากับ True
c = ( A <= 'a' ) and ( ( C + D ) == 3 )             #ตัวแปร c มีค่าเท่ากับ False
d = ( C ** D ) > ( C * D )                          #ตัวแปร d มีค่าเท่ากับ True
```

3. ข้อมูลแบบเรียงลำดับ (Sequence) จะประกอบด้วยข้อมูลที่จัดเก็บเรียงลำดับต่อกัน ได้แก่ ข้อความ (String), ลิสต์ (List) และทูเพิล (Tuple) โดยมีรายละเอียดดังนี้

3.1 ข้อมูลชนิดสายอักขระ (String) หมายถึง ข้อความหรือกลุ่มของตัวอักษร ตัวเลขที่ไม่สามารถนำไปคำนวณได้ จำนวนตั้งแต่ 1 ตัวขึ้นไป มาเรียงต่อกันภายในเครื่องหมาย single quote ('...') หรือ double quote ("...") เช่น 'Hello' , "Hello" , 'xyz' , "xyz" เป็นต้น โดยตัวอักษรแต่ละตัวในสตริงจะมีตำแหน่ง Index ที่แตกต่างกันออกไป

โดยการกำหนดค่าตัวแปรนิดนี้ ต้องเขียนให้อยู่ในบรรทัดเดียวกัน แต่สามารถใช้เครื่องหมาย backslash (\) แทรก หากต้องการแทรกเป็บบรรทัดใหม่

ข้อมูลในกลุ่มนี้ได้แก่ ตัวอักษร (A ถึง Z, a ถึง z) ตัวเลข (0 – 9) และสัญลักษณ์พิเศษต่าง ๆ เช่น + – * / % : . : ; , ‘ “ ^ = < > ( ) [ ] { } ! @ # $ % & ? ฿ | \ ~ เป็นต้น

ตัวอย่างข้อมูลชนิดตัวเลข เช่น

```
strA = '35788'          #ตัวแปร strA มีค่าเท่ากับ 35788  เป็นข้อมูลสายอักขระ
strB = "35788"          #ตัวแปร strB มีค่าเท่ากับ 35788  เป็นข้อมูลสายอักขระ
strC = "Monday"         #ตัวแปร strC มีค่าเท่ากับ Monday  เป็นข้อมูลสายอักขระ
strD = "Jan"            #ตัวแปร strD มีค่าเท่ากับ Jan  เป็นข้อมูลสายอักขระ
strE = "A & " \
       "B"              #ตัวแปร strE มีค่าเท่ากับ A & B  เป็นข้อมูลสายอักขระ
srtF = "A"              #ตัวแปร strF มีค่าเท่ากับ A  เป็นข้อมูลสายอักขระ
```

การหาความยาวของสายอักขระหรือสตริง

จะใช้ฟังก์ชัน len( ) เช่น

```
len("Hello")        #ผลลัพธ์ คือ 5
len('xyz')          #ผลลัพธ์ คือ 3
```

การค้นหาตัวอักษรหรือกลุ่มตัวอักษรในสตริง

การค้นหาตัวอักษร ทำได้โดยระบุ Index ของสตริง เช่น

```
"Hello"[0]          #ผลลัพธ์ คือ H
"Hello"[4]          #ผลลัพธ์ คือ o
"Hello"[-5]         #ผลลัพธ์ คือ H
"Hello"[-1]         #ผลลัพธ์ คือ o
```

การค้นหากลุ่มตัวอักษร ทำได้โดยระบุช่วงของ Index ของกลุ่มตัวอักษรที่ต้องการค้นหา เช่น

```
"Hello"[0:5]    #ค้นหากลุ่มตัวอักษรจากตำแหน่งที่ 0 ถึงตำแหน่งที่ 4      ผลลัพธ์ คือ Hello
"Hello"[:5]     #ค้นหากลุ่มตัวอักษรจากตำแหน่งแรกสุด ถึงตำแหน่งที่ 4    ผลลัพธ์ คือ Hello
"Hello"[0:]     #ค้นหากลุ่มตัวอักษรจากตำแหน่งที่ 0 ถึงตำแหน่งสุดท้าย    ผลลัพธ์ คือ Hello
"Hello"[-2:]    #ค้นหากลุ่มตัวอักษรจากตำแหน่งที่ –2 ถึงตำแหน่งสุดท้าย   ผลลัพธ์ คือ lo
```

3.2 ข้อมูลชนิดลิสต์ (List) เป็นชนิดข้อมูลที่สามารถเก็บข้อมูลประเภทเดียวกันหรือต่างประเภทกันก็ได้ และข้อมูลในลิสต์สามารถซ้ำกันได้ หากภายหลังข้อมูลมีการเปลี่ยนแปลงก็สามารถแก้ไขข้อมูลในลิสต์ได้ ข้อมูลในลิสต์จะมีลำดับที่แน่นอนว่า ข้อมูลใดอยู่ในตำแหน่งไหน เนื่องจากมีตำแหน่ง Index กำกับข้อมูลอยู่ โดยชนิดของข้อมูลประเภทนี้จะถูกครอบไว้ด้วยเครื่องหมายวงเล็บเหลี่ยม หรือ วงเล็บก้ามปู หรือ square brackets [...] และคั่นข้อมูลด้วยเครื่องหมายคอมม่า (,) ข้อมูลในเครื่องหมาย [...] สามารถเป็นค่าว่างได้ ซึ่งจะเรียกว่า ลิสต์ว่าง (Empty List) เช่น

```
a = [1 , 2 , 3 , 4 , 5]                                  # List of integers
b = ['apple' , 'banana' , 'cherry']                      # List of strings
c = [1 , 'hello' , 3.14 , True]                          # Mixed data types
d = [550 , 1350.5 , 820.8 , 200]                         # List of integers and float
e = ["apple" , "orange" , "kiwi" , "grape", "banana"] # List of strings
```

หรือจะกำหนดเป็นลิสต์ซ้อนลิสต์ก็ได้ เช่น

```
student = [ ["001" , "Apply" , 20] ,
            ["002" , "Cherry" , 30] ,
            ["003" , "Kiwi" , 15] ]
```

การหาความยาวของลิสต์

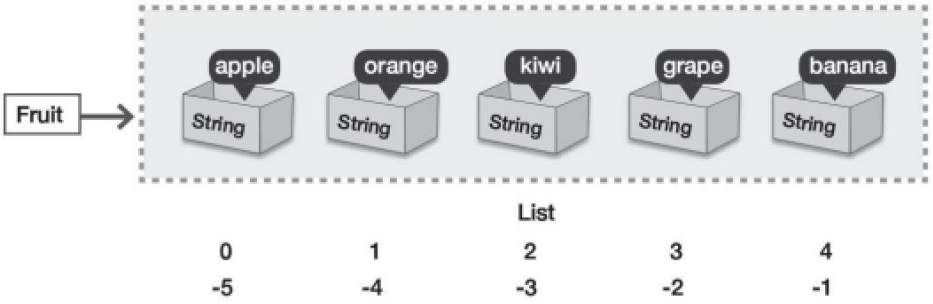
จะใช้ฟังก์ชัน len( ) เช่น

```
len(["apple" , "orange" , "kiwi" , "grape", "banana"])    #ผลลัพธ์ คือ 5
len(e)                                                    #ผลลัพธ์ คือ 5
len(c)                                                    #ผลลัพธ์ คือ 4
```

การค้นหาตัวอักษรหรือกลุ่มตัวอักษรในลิสต์

การค้นหาตัวอักษร ทำได้โดยระบุ Index ของลิสต์ เช่น

```
fruit = ["apple" , "orange" , "kiwi" , "grape", "banana"]
```




```
fruit[0]          #ผลลัพธ์ คือ apple
fruit[–1]         #ผลลัพธ์ คือ banana
```

3.3 ข้อมูลชนิดทูเพิล (Tuple) เป็นชนิดข้อมูลที่สามารถเก็บข้อมูลประเภทเดียวกันหรือต่างประเภทกันก็ได้ และข้อมูลในทูเพิลสามารถซ้ำกันได้เช่นเดียวกับลิสต์ทุกประการ แต่เมื่อกำหนดข้อมูลลงในทูเพิลแล้วจะ**ไม่สามารถเปลี่ยนแปลงค่าได้** ชนิดของข้อมูลประเภทนี้จึงเหมาะที่จะนำมากำหนดให้กับข้อมูลที่ไม่ต้องการให้เปลี่ยนแปลงค่าได้ในภายหลัง

ข้อมูลในทูเพิลจะมีลำดับที่แน่นอนเนื่องจากมีตำแหน่ง Index กำกับข้อมูลอยู่ โดยชนิดข้อมูลประเภทนี้จะถูกครอบไว้ด้วยเครื่องหมายวงเล็บ ( ) และคั่นข้อมูลด้วยเครื่องหมายคอมม่า (,) เช่น

```
price = ("$6.57" , "$10" , "$20.25")
color = ("red" , "green" , "blue" , "pink")
num = (111 , 222 , 333 , 111)
```

การหาค่าความยาว การค้นหาข้อมูลและการค้นหากลุ่มข้อมูลในทูเพิล หลักการจะเหมือนกับลิสต์ทุกประการ ส่วนการแก้ไขข้อมูล การเพิ่มข้อมูล และการลบข้อมูลในทูเพิลจะไม่สามารถทำได้ เพราะทูเพิลไม่สามารถเปลี่ยนแปลงค่าได้

4. ข้อมูลชนิดเซต (Set) เป็นชนิดข้อมูลที่ไม่มีตำแหน่ง Index กำกับข้อมูลอยู่ ดังนั้นจึงไม่มีลำดับที่แน่นอนว่าข้อมูลใดอยู่ในตำแหน่งไหน เมื่อสร้างเซตแล้วจะไม่สามารถแก้ไขข้อมูลในเซตได้ แต่สามารถเพิ่มค่าข้อมูลใหม่ลงไปในเซตได้ โดยชนิดข้อมูลประเภทนี้จะถูกครอบไว้ด้วยเครื่องหมายวงเล็บปีกกา {...} เช่น

```
day_set = {"Monday" , "Tuesday" , "Wednesday" , "Thursday" , "Friday" , "Saturday" , "Sunday"}
set1 = {1 , 2 , 3 , 4}
```

การหาความยาวของเซต

จะใช้ฟังก์ชัน len( ) เช่น

```
len(day_set)        #ผลลัพธ์ คือ 7
len(set1)           #ผลลัพธ์ คือ 4
```

การค้นหาข้อมูลในเซต

การค้นหาว่ามีข้อมูลใด ๆ อยู่ในเซตหรือไม่ ทำได้โดยเรียกใช้ฟังก์ชัน in( ) ดังรูปแบบต่อไปนี้

ข้อมูลที่ต้องการค้นหา in ชื่อตัวแปรเซต

ตัวอย่าง เช่น

```
"Friday" in day_set        #ผลลัพธ์ คือ True
"monday" in day_set        #ผลลัพธ์ คือ False
```

5. ข้อมูลชนิดดิกชันนารี (Dictionary) เป็นชนิดข้อมูลที่เก็บข้อมูลในรูปแบบ key และ value โดย key กับ value จะสัมพันธ์กัน กล่าวคือ จะใช้ key เป็น Index ในการเข้าถึงข้อมูล (value) แต่ละตัว ซึ่งค่า key ต้องไม่ซ้ำกัน

ข้อมูลชนิดนี้เมื่อสร้างขึ้นแล้วสามารถเปลี่ยนแปลงแก้ไขได้ โดยชนิดข้อมูลประเภทนี้จะถูกครอบไว้ด้วยเครื่องหมายวงเล็บปีกกา {...} ดังรูปแบบต่อไปนี้

```
ชื่อตัวแปรดิกชันนารี = {
        key ตัวที่ 1   :   value ตัวที่ 1
        key ตัวที่ 2   :   value ตัวที่ 2
                .....
        key ตัวสุดท้าย   :   value ตัวสุดท้าย
}
```

ตัวอย่าง เช่น

```
student = {1 : "Manee" , 2 : "Mana" , 3 : "Chujai" , 4 : "Piti" , 5 : "Sommai" , 6 : "Somchai"}
evaluation = {
              "A" : "very good" ,
              "B" : "good" ,
              "C" : "fair" ,
              "D" : "poor"
             }
```

การหาความยาวของดิกชันนารี

จะใช้ฟังก์ชัน len( ) คือ len(ชื่อตัวแปรดิกชันนารี) เช่น

```
len(student)          #ผลลัพธ์ คือ 6
len(evaluation)       #ผลลัพธ์ คือ 4
```

การค้นหาข้อมูลในดิกชันนารี

การค้นหาว่ามีข้อมูลใด ๆ ในดิกชันนารี ทำได้หลายรูปแบบ ดังรูปแบบต่อไปนี้

```
ชื่อตัวแปรดิกชันนารี[key]
ชื่อตัวแปรดิกชันนารี.get(key)
```

ตัวอย่าง เช่น

```
student[3]            #ผลลัพธ์ คือ Chujai
student.get(6)        #ผลลัพธ์ คือ Somchai
Evaluation["A"]       #ผลลัพธ์ คือ very good
evaluation.get("C")   #ผลลัพธ์ คือ fair
```

5. การตรวจสอบชนิดของข้อมูล (Data Type Checking)

เนื่องจากในภาษาไพทอนไม่ต้องมีการประกาศชนิดของข้อมูลของตัวแปรไพทอน นักเรียนสามารถรับรู้ชนิดข้อมูลจากค่าที่กำหนดในชุดคำสั่ง แต่ในบางครั้งก็มีความจำเป็นต้องทราบข้อมูลของตัวแปรก่อนที่จะประมวลผลอย่างใดอย่างหนึ่ง ดังนั้น นักเรียนสามารถใช้ฟังก์ชัน type() ตรวจสอบชนิดของข้อมูลได้ ซึ่งมีรูปแบบการทำงานดังนี้

```
type(variable)
```

โดยที่ variable คือ ชื่อตัวแปรที่ต้องการตรวจสอบชนิดของข้อมูล

ตัวอย่าง เช่น

```
a = 10
type(a)                 #ชนิดข้อมูลของตัวแปร a คือ <class 'int'>
b = 10.0
type(b)                 #ชนิดข้อมูลของตัวแปร b คือ <class 'float'>
c = 15e–2
type(c)                 #ชนิดข้อมูลของตัวแปร c คือ <class 'float'>
d = 10 + 2j
type(d)                 #ชนิดข้อมูลของตัวแปร d คือ <class 'complex'>
e = 9 > 5
type(e)                 #ชนิดข้อมูลของตัวแปร e คือ <class 'bool'>
f = "Nangrong"
type(f)                 #ชนิดข้อมูลของตัวแปร f คือ <class 'str'>
g = [1 , 'hello' , 3.14 , True]
type(g)                                     #ชนิดข้อมูลของตัวแปร g คือ <class 'list'>
h = ("red" , "green" , "blue" , "pink")
type(h)                                     #ชนิดข้อมูลของตัวแปร h คือ <class 'tuple'>
i = {1 , 2 , 3 , 4}
type(i)                                     #ชนิดข้อมูลของตัวแปร i คือ <class 'set'>
j = {1 : "apple" , 2 : "orange" , 3 : "grape"}
type(j)                                     #ชนิดข้อมูลของตัวแปร j คือ <class 'dict'>
```

6. การแปลงชนิดของข้อมูล (Data Type Conversion)

การแปลงชนิดข้อมูลในภาษาไพทอน นักเรียนสามารถใช้ Built-in Function ที่มีอยู่แล้วในไพทอนได้เลย โดยฟังก์ชันเหล่านั้นจะมีชื่อเหมือนกับชนิดข้อมูลนั้น ๆ ซึ่งก็คือ ชื่อของคลาสนั่นเอง ตัวอย่าง เช่น

- ฟังก์ชัน **int()** ใช้แปลงข้อมูลประเภทใด ๆ ให้เป็นชนิดตัวเลขจำนวนเต็ม
- ฟังก์ชัน **str()** ใช้แปลงข้อมูลประเภทใด ๆ ให้เป็นชนิดตัวอักษร

ตารางแสดงตัวอย่างของฟังก์ชันสำหรับการแปลงข้อมูลในภาษาไฟทอน

ฟังก์ชัน	ความหมาย
int(x [,base])	แปลงข้อมูล x จากฐานที่กำหนด base ให้เป็นจำนวนเต็ม
float(x)	แปลงข้อมูล x ให้เป็นเลขจำนวนทศนิยม
Complex(real [,im])	สร้างตัวเลขจำวนเชิงซ้อนจากค่า real และค่า imagine
str(x)	แปลงข้อมูล x ให้เป็นตัวอักษร
tuple(s)	แปลงข้อมูลแบบเรียงลำดับ s ให้เป็นข้อมูลทูเพิล
list(s)	แปลงข้อมูลแบบเรียงลำดับ s ให้เป็นข้อมูลลิสต์
set(s)	แปลงข้อมูลแบบเรียงลำดับ s ให้เป็นข้อมูลทูเซต
dict(d)	แปลงข้อมูล d ให้เป็นข้อมูลดิกชันนารี
chr(x)	แปลงข้อมูลเลขจำนวนเต็ม x .ให้เป็นตัวอักษร
ord(x)	แปลงข้อมูลตัวอักษร x ให้เป็นค่าตัวเลขจำนวนเต็ม
hex(x)	แปลงข้อมูลตัวเลข x ให้เป็นเลขฐานสิบหก ชนิดตัวอักษร
oct(x)	แปลงข้อมูลตัวเลข x ให้เป็นเลขฐานแปด ชนิดตัวอักษร
bin(x)	แปลงข้อมูลตัวเลข x ให้เป็นเลขฐานสอง ชนิดตัวอักษร

ใบงานหน่วยที่ 3

1. ชนิดของข้อมูลพื้นฐานในภาษาไพทอน มีกี่ชนิด อะไรบ้าง

.....

.....

2. พิจารณาชื่อตัวแปรต่อไปนี้ว่าถูกต้องตามหลักการตั้งชื่อของภาษาไพทอนหรือไม่ โดยกาเครื่องหมาย ✓ ลงในช่องที่ตรงกับคำตอบนั้น

ข้อที่	ชื่อตัวแปร	ถูกต้อง	ไม่ถูกต้อง
2.1	Number		
2.2	N AME		
2.3	SALARY$		
2.4	7ELEVEN		
2.5	_Point		

ข้อที่	ชื่อตัวแปร	ถูกต้อง	ไม่ถูกต้อง
2.6	not		
2.7	ToTal_Score		
2.8	X+Y		
2.9	MARK1		
2.10	D.1		

3. ให้พิจารณาชื่อที่กำหนดให้เป็นคำสงวน (Reserved Word) โดยกาเครื่องหมาย ✓ ลงในช่องที่ตรงกับคำตอบนั้น

ข้อที่	ชื่อตัวแปร	คำสงวน	ไม่ใช่คำสงวน
3.1	cat		
3.2	if		
3.3	Test		
3.4	is		
3.5	print		

ข้อที่	ชื่อตัวแปร	คำสงวน	ไม่ใช่คำสงวน
3.6	break		
3.7	and		
3.8	grade		
3.9	book		
3.10	and		

4. พิจารณาข้อมูลที่กำหนดให้มีชนิดของข้อมูลเป็นแบบใด โดยกาเครื่องหมาย ✓ ลงในช่องที่ตรงกับคำตอบนั้น

ข้อที่	ข้อมูล	int	float	complex	bool	str
4.1	"NangrongSchool"					
4.2	–120					
4.3	False					
4.4	125e-3					
4.5	15 + 2j					

5. พิจารณาว่าผลลัพธ์ของข้อมูลที่กำหนดให้มีชนิดข้อมูลเป็นแบบใด โดยกาเครื่องหมาย ✓ ลงในช่องที่ตรงกับคำตอบนั้น

ข้อที่	ข้อมูล	int	float	complex	bool	str
5.1	a = 7 / 4					
5.2	a = 100 + 50000					
5.3	a = 15 > 9					
5.4	a = 2.0 + 5j					
5.5	a = "Buriram"					

6. จงเขียนคำสั่งประกาศตัวแปร และกำหนดค่าตามเงื่อนไขต่อไปนี้

6.1 ประกาศตัวแปร pi เป็นตัวแปรเก็บข้อมูลเลขทศนิยม มีค่าเท่ากับ 3.142857

.....

.....

6.2 ประกาศตัวแปร str เป็นตัวแปรเก็บข้อมูลสายอักขระ มีค่าเท่ากับ "Nangrong Buriram"

.....

.....

6.3 ประกาศตัวแปร max เป็นตัวแปรเก็บข้อมูลเลขจำนวนเต็ม มีค่าเท่ากับ 1000

.....

.....

6.4 ประกาศตัวแปร x เป็นจำนวนเชิงซ้อน โดยมีค่า 15 เป็นส่วนของจำนวนจริง และ 3 เป็นส่วนของจำนวนจินตภาพ

.....

.....

6.5 ประกาศตัวแปร status เป็นตัวแปรเก็บข้อมูลเลขชนิดตรรกะ มีค่าความจริงเป็นจริง (True)

.....

.....

7. จงเขียนคำสั่งประกาศตัวแปร และกำหนดค่าตามเงื่อนไขต่อไปนี้

ข้อมูล A มีสมาชิกดังนี้

Apple , Banana , Cherry , Mango , Pineapple , Grapes , Watermelon , Strawberry

ข้อมูล B มีสมาชิกดังนี้

Football , Basketball , Badminton , Swimming , Tennis , Volleyball

7.1 ประกาศตัวแปรชื่อ fruit เป็นข้อมูลชนิดลิสต์ (List) ซึ่งมีสมาชิกของข้อมูล A ทั้งหมด

.....

7.2 ประกาศตัวแปรชื่อ sport เป็นข้อมูลชนิดลิสต์ (List) ซึ่งมีสมาชิกของข้อมูล B ทั้งหมด

.....

7.3 ประกาศตัวแปรชื่อ fruit เป็นข้อมูลชนิดทูเพิล (Tuple) ซึ่งมีสมาชิกของข้อมูล A ทั้งหมด

.....

7.4 ประกาศตัวแปรชื่อ sport เป็นข้อมูลชนิดทูเพิล (Tuple) ซึ่งมีสมาชิกของข้อมูล B ทั้งหมด

.....

7.5 ประกาศตัวแปรชื่อ fruit เป็นข้อมูลชนิดเซต (Set) ซึ่งมีสมาชิกของข้อมูล A ทั้งหมด

.....

7.6 ประกาศตัวแปรชื่อ sport เป็นข้อมูลชนิดเซต (Set) ซึ่งมีสมาชิกของข้อมูล B ทั้งหมด

.....

ผลลัพธ์

กรุณากรอกอายุ : 18

คุณอายุ 18 ปี

ชนิดข้อมูลของไพธอน

ชนิดข้อมูล คือ สิ่งที่ใช้กำหนดลักษณะและขอบเขตของข้อมูลนั้น ๆ โดยข้อมูลที่มีชนิดของข้อมูลแตกต่างกัน ก็จะเก็บข้อมูลได้ในลักษณะที่ต่างกัน และขอบเขตของข้อมูลที่เก็บได้ก็จะไม่เท่ากันด้วย โดยชนิดข้อมูลพื้นฐานในภาษาไพธอน แบ่งออกได้เป็น 7 ประเภท ดังนี้

Numeric

เป็นชนิดข้อมูลประเภทตัวเลข แบ่งได้ 3 ประเภท คือ

Integers

เป็นเลขจำนวนเต็มบวกหรือลบ จะเป็นเลขฐานสิบ, เลขฐานสอง (นำหน้าตัวเลขด้วย 0b), เลขฐานแปด (นำหน้าตัวเลขด้วย 0o) หรือเลขฐานสิบหก (นำหน้าด้วย 0x) ก็ได้ ตัวอย่างเช่น

เลขฐานสิบ	เลขฐานสอง	เลขฐานแปด	เลขฐานสิบหก
10	0b1010	0o12	0xA
-11	-0b1011	-0o13	-0xB

Floating point numbers

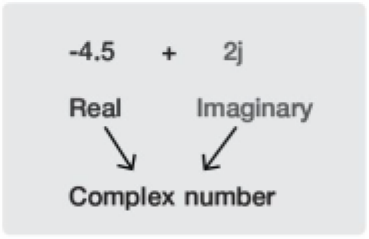
เป็นเลขทศนิยม (บางทีเรียกว่า จำนวนจริง) เช่น 10.0, 15.333, 2.99e2 (มีค่าเท่ากับ 2.99×10^2 หรือ 299.0), 9.109e-2 (มีค่าเท่ากับ 9.109×10^{-2}) หรือ 0.09109 เป็นต้น

Complex numbers

เป็นชนิดข้อมูลแบบเลขจำนวนเชิงซ้อน โดยเลขจำนวนเชิงซ้อนเกิดจากการนำจำนวนจริง (Real) และจำนวนจินตภาพ (Imaginary) มารวมกัน

รูปแบบของจำนวนเชิงซ้อน คือ a+bi โดย a คือ ส่วนของจำนวนจริง และ b คือ ส่วนของจำนวนจินตภาพ ในไพธอนจะนำตัวอักษร j หรือ J มากำกับต่อท้ายค่าที่เป็นจำนวนจินตภาพ

ตัวอย่างเช่น -4.5+2j จะเรียก -4.5 ว่าเป็นส่วนของเลขจำนวนจริง และ 2j เป็นส่วนของจำนวนจินตภาพ



ตัวอย่างที่ 3.3

```
a = 10
b = 10.0
c = 5.5+2j
print(type(a))
print(type(b))
print(type(c))
```

ผลลัพธ์

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

Boolean

เป็นชนิดข้อมูลตรรกะที่มีได้เพียง 2 ค่า คือ จริง (True) และ เท็จ (False)

ตัวอย่างที่ 3.4

```
a = 3 > 2
b = 5 < 1
print(a , type(a))
print(b , type(b))
```

ผลลัพธ์

```
True <class 'bool'>
False <class 'bool'>
```

String

สตริง (String) เป็นชนิดข้อมูลประเภทข้อความหรือกลุ่มของตัวอักษร โดยสตริงจะถูกครอบไว้ด้วยเครื่องหมาย " " หรือ ' ' เช่น "Hello", 'Hello', 'xyz', "xyz" เป็นต้น โดยตัวอักษรแต่ละตัวในสตริงจะมีตำแหน่ง index ที่แตกต่างกันออกไป ดังรูป

การหาความยาวสตริง

จะใช้ฟังก์ชั่น len() เช่น len("Hello") ผลลัพธ์ คือ 5

การค้นหาตัวอักษรและกลุ่มตัวอักษรในสตริง

การค้นหาตัวอักษร ทำได้โดยระบุ index ของสตริง เช่น

"Hello"[0] ผลลัพธ์ คือ H

"Hello"[4] ผลลัพธ์ คือ o

"Hello"[-5] ผลลัพธ์ คือ H

"Hello"[-1] ผลลัพธ์ คือ o

การค้นหากลุ่มตัวอักษร ทำได้โดยระบุช่วงของ index ของกลุ่มตัวอักษรที่ต้องการค้นหา เช่น

```
"Hello"[0:5]
```

ค้นหากลุ่มตัวอักษรจากตำแหน่งที่ 0 ถึงตำแหน่งที่ 4 ผลลัพธ์ คือ Hello

```
"Hello"[:5]
```

ค้นหากลุ่มตัวอักษรจากตำแหน่งแรกสุดถึงตำแหน่งที่ 4 ผลลัพธ์ คือ Hello

```
"Hello"[0:]
```

ค้นหากลุ่มตัวอักษรจากตำแหน่งที่ 0 ไปจนถึงตำแหน่งสุดท้าย ผลลัพธ์ คือ Hello

```
"Hello"[-2:]
```

ค้นหากลุ่มตัวอักษรจากตำแหน่ง -2 ไปจนถึงตำแหน่งสุดท้าย ผลลัพธ์ คือ lo

ตัวอย่างที่ 3.5

```
print("Hello"[0])
print("Hello"[4])
print("Hello"[-5])
print("Hello"[-1])
print("Hello"[0:5])
print("Hello"[:5])
print("Hello"[0:])
print("Hello"[-2:])
```

ผลลัพธ์

```
H
o
H
o
Hello
Hello
Hello
lo
```

List

ลิสต์ (List) เป็นชนิดข้อมูลที่สามารถเก็บข้อมูลประเภทเดียวกันหรือต่างประเภทกันก็ได้ และข้อมูลในลิสต์สามารถซ้ำกันได้ หากภายหลังข้อมูลมีการเปลี่ยนแปลงก็สามารถแก้ไขข้อมูลในลิสต์ได้ ข้อมูลในลิสต์มีจะลำดับที่แน่นอนว่าข้อมูลใดอยู่ในตำแหน่งไหนเนื่องจากมีตำแหน่ง index กำกับข้อมูลอยู่ โดยชนิดข้อมูลประเภทนี้จะถูกครอบไว้ด้วยเครื่องหมาย [ ] เช่น

```
Price = [550,1350.75,820.5,200]
fruit = ["apple","orange","kiwi","grape","banana"]
number = [2,4,6,8]
amount = [250,250,350,300]
```

หรือจะกำหนดเป็นลิสต์ซ้อนลิสต์ก็ได้ เช่น

```
student = [["001","Apple",20],
           ["002","Cherry",30],
           ["003","Kiwi",15]]
```

การหาความยาวลิสต์

จะใช้ฟังก์ชั่น len() เช่น

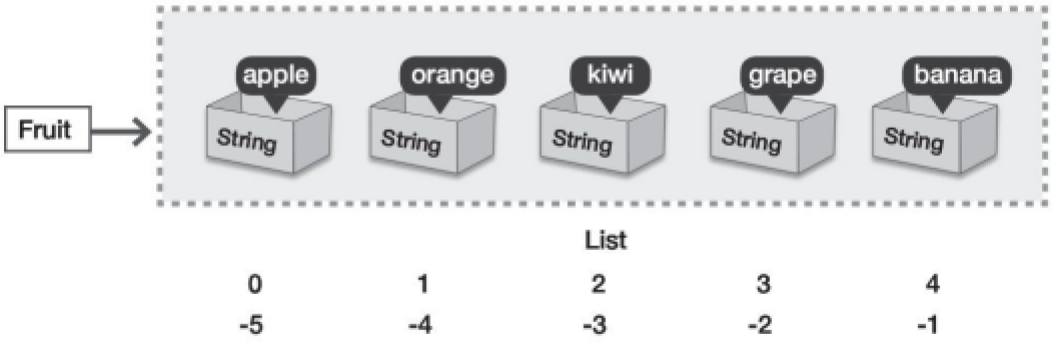
```
len(["apple","orange","kiwi","grape","banana"])
```

ผลลัพธ์ คือ 5

การค้นหาตัวอักษรและกลุ่มตัวอักษรในลิสต์

การค้นหาตัวอักษร ทำได้โดยระบุ index ของลิสต์ เช่น

```
fruit = ["apple","orange","kiwi","grape","banana"]
```



fruit[0] ผลลัพธ์ คือ apple

fruit[-1] ผลลัพธ์ คือ banana

การค้นหากลุ่มตัวอักษร ทำได้โดยระบุช่วงของ index ของกลุ่มตัวอักษรที่ต้องการค้นหา เช่น

```
fruit[2:4]
```

ค้นหากลุ่มตัวอักษรจากตำแหน่งที่ 2 ถึงตำแหน่งที่ 3 ผลลัพธ์ คือ ["kiwi" , "grape"]

```
fruit[-5:-3]
```

ค้นหากลุ่มตัวอักษรจากตำแหน่งที่ -5 ถึงตำแหน่งที่ -4 ผลลัพธ์ คือ ["apple" , "orange"]

```
fruit[:5]
```

ค้นหากลุ่มตัวอักษรจากตำแหน่งแรกสุดถึงตำแหน่งที่ 4 ผลลัพธ์ คือ ["apple" , "orange" , "kiwi" , "grape" , "banana"]

```
fruit[-4:]
```

ค้นหากลุ่มตัวอักษรจากตำแหน่งที่ -4 ถึงตำแหน่งสุดท้าย ผลลัพธ์ คือ ["orange" , "kiwi" , "grape" , "banana"]

ตัวอย่างที่ 3.6

```
fruit = ["apple","orange","kiwi","grape","banana"]
print(fruit[0])
print(fruit[-1])
print(fruit[2:4])
print(fruit[-5:-3])
print(fruit[:5])
```

ผลลัพธ์

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

การแก้ไขข้อมูลในลิสต์

การแก้ไขข้อมูลในลิสต์ ทำได้โดยระบุตำแหน่ง index ที่ต้องการแก้ไขข้อมูล และกำหนดค่าข้อมูลใหม่ลงไป

ตัวอย่างที่ 3.7

```
fruit = ["apple","orange","kiwi","grape","banana"]
print(fruit)
fruit[0] = "cherry"
fruit[4] = "apple"
print(fruit)
```

ผลลัพธ์

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

การเพิ่มข้อมูลในลิสต์

การเพิ่มข้อมูลในลิสต์สามารถใช้ฟังก์ชั่น `append()` หรือ `insert()` ก็ได้ ดังรูปแบบต่อไปนี้

ตัวแปร `list.append`(ข้อมูลที่ต้องการเพิ่มเข้าลิสต์)

ตัวแปร `list.insert`(index,ข้อมูลที่ต้องการเพิ่มเข้าลิสต์)

ตัวอย่างที่ 3.8

```
fruit = ["apple","orange","kiwi","grape","banana"]
print(fruit)
fruit.append("cherry")
print(fruit)
fruit.insert(0,"cherry")
fruit.insert(2,"papaya")
print(fruit)
```

ผลลัพธ์

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

การลบข้อมูลในลิสต์

การลบข้อมูลในลิสต์สามารถใช้ฟังก์ชั่น remove() หรือ del() ก็ได้ ดังรูปแบบต่อไปนี้

```
ตัวแปร list.remove(ข้อมูลในลิสต์ที่ต้องการลบ)
del ตัวแปร list[index]
```

ตัวอย่างที่ 3.9

```
fruit = ["apple","orange","kiwi","grape","banana"]
fruit.remove("orange")
print(fruit)
color = ["red","green","blue"]
del color[1]
print(color)
```

ผลลัพธ์

```
['apple', 'kiwi', 'grape', 'banana']
['red', 'blue']
```

Tuple

ทูเพิล (Tuple) เป็นชนิดข้อมูลที่สามารถเก็บข้อมูลประเภทเดียวกันหรือต่างประเภทกันก็ได้ และข้อมูลในทูเพิลสามารถซ้ำกันได้เช่นเดียวกับลิสต์ทุกประการ แต่เมื่อกำหนดข้อมูลลงในทูเพิลแล้วจะไม่สามารถเปลี่ยนแปลงค่าได้ ชนิดข้อมูลประเภทนี้จึงเหมาะที่จะนำมากำหนดให้กับข้อมูลที่ไม่ต้องการให้เปลี่ยนแปลงค่าได้ในภายหลัง

ข้อมูลในทูเพิลจะมีลำดับที่แน่นอนเนื่องจากมีตำแหน่ง index กำกับข้อมูลอยู่ โดยชนิดข้อมูลประเภทนี้จะถูกครอบไว้ด้วยเครื่องหมาย () เช่น

```
price_tag = ("$6.57","$10","$20.25")
color = ("red","green","blue","pink")
num = (111,222,333,111)
```

การหาค่าความยาว การค้นหาข้อมูล และการค้นหากลุ่มข้อมูลในทูเพิล หลักการจะเหมือนกับลิสต์ทุกประการ ส่วนการแก้ไขข้อมูล การเพิ่มข้อมูล และการลบข้อมูลในทูเพิลจะไม่สามารถทำได้ เพราะทูเพิลไม่สามารถเปลี่ยนแปลงค่าได้

Set

เซต (Set) เป็นชนิดข้อมูลที่ไม่มีตำแหน่ง index กำกับข้อมูลอยู่ ดังนั้นจึงไม่มีจะลำดับที่แน่นอนว่าข้อมูลใดอยู่ในตำแหน่งไหน เมื่อสร้างเซตแล้วจะไม่สามารถแก้ไขข้อมูลในเซตได้ แต่สามารถเพิ่มค่าข้อมูลใหม่ลงไปในเซตได้ โดยชนิดข้อมูลประเภทนี้จะถูกครอบไว้ด้วยเครื่องหมาย { } เช่น

```
day_set = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"}
```

การหาความยาวเซต

จะใช้ฟังก์ชั่น len() เช่น

```
day_set = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"}
len(day_set) ผลลัพธ์ คือ 7
```

การค้นหาข้อมูลในเซต

การค้นหาว่ามีข้อมูลใด ๆ ในเซตหรือไม่ ทำได้โดยเรียกใช้ฟังก์ชั่น in() ดังรูปแบบต่อไปนี้

```
ข้อมูลที่ต้องการ in ตัวแปรเซต
```

ตัวอย่างที่ 3.10

```
day_set = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"}
print("Friday" in day_set)
print("monday" in day_set)
```

ผลลัพธ์

```
True
False
```

การเพิ่มข้อมูลในเซต

การเพิ่มข้อมูลในเซตสามารถใช้ฟังก์ชั่น add() หรือ update() ก็ได้ ดังรูปแบบต่อไปนี้

ตัวแปร set.add(ข้อมูลที่ต้องการเพิ่มเข้าเซต)

ตัวแปร set.update(ข้อมูลตัวที่ 1, ข้อมูลตัวที่ 2, ... , ข้อมูลตัวสุดท้ายที่ต้องการเพิ่ม)

ตัวอย่างที่ 3.11

```
flavor_set = {"Chocolate", "Vanilla", "Strawberry"}
flavor_set.add("Coffee")
print(flavor_set)
flavor_set.update(["Chocolate Chip","Cookies and Cream"])
print(flavor_set)
```

ผลลัพธ์

```
{'Coffee', 'Chocolate', 'Vanilla', 'Strawberry'}
{'Chocolate', 'Vanilla', 'Strawberry', 'Cookies and Cream', 'Chocolate Chip',
'Coffee'}
```

การลบข้อมูลในเซต

การลบข้อมูลในเซตสามารถใช้ฟังก์ชั่น remove() หรือ discard() ก็ได้ ดังรูปแบบต่อไปนี้

ตัวแปร set.remove(ข้อมูลในเซตที่ต้องการลบ)

ตัวแปร set.discard(ข้อมูลในเซตที่ต้องการลบ)

ตัวอย่างที่ 3.12

```
flavor_set = {"Chocolate", "Vanilla", "Strawberry"}
flavor_set.remove("Coffee")
print(flavor_set)
```

ผลลัพธ์

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-17-31e4bafbc0d4> in <module>
      1 flavor_set = {"Chocolate", "Vanilla", "Strawberry"}
----> 2 flavor_set.remove("Coffee")
      3 print(flavor_set)

KeyError: 'Coffee'
```

จะพบว่าใช้ฟังก์ชั่น `remove()` สั่งลบข้อมูล "Coffee" ซึ่งไม่มีอยู่ในเซต จึงมีข้อผิดพลาดแจ้งให้ทราบ หากลองเปลี่ยนโค้ดโปรแกรมไปใช้ฟังก์ชั่น `discard()` ดังนี้

```
flavor_set = {"Chocolate", "Vanilla", "Strawberry"}

flavor_set.discard("Coffee")

print(flavor_set)
```

จะพบว่าโปรแกรมไม่แจ้งข้อผิดพลาดให้ทราบ และยังคงพิมพ์ค่าข้อมูลของเซตออกมาได้ตามปกติ ดังนี้

```
{'Chocolate', 'Vanilla', 'Strawberry'}
```

Dictionary

ดิกชันนารี (Dictionaries) เป็นชนิดข้อมูลที่เก็บข้อมูลในรูปแบบ key และ value โดย key กับ value จะสัมพันธ์กัน กล่าวคือ จะใช้ key เป็น index ในการเข้าถึงข้อมูล (value) แต่ละตัว ซึ่งค่า key ต้องไม่ซ้ำกัน

ข้อมูลชนิดนี้เมื่อสร้างขึ้นแล้วสามารถเปลี่ยนแปลงแก้ไขได้ โดยชนิดข้อมูลประเภทนี้จะถูกครอบไว้ด้วยเครื่องหมาย `{ }` ดังรูปแบบต่อไปนี้

```
ชื่อตัวแปรดิกชันนารี {
                key ตัวที่ 1 : value ตัวที่ 1
                key ตัวที่ 2 : value ตัวที่ 2
                …
                key ตัวสุดท้าย : value ตัวสุดท้าย
}
```

การหาความยาวดิกชันนารี

จะใช้ฟังก์ชั่น len() คือ len(ชื่อตัวแปรดิกชันนารี)

```
evaluation_dict = {
  "A": "very good",
  "B": "good",
  "C": "fair",
  "D" : "poor"
}
len(evaluation_dict) ผลลัพธ์ คือ 4
```

การค้นหาข้อมูลในดิกชันนารี

การค้นหาข้อมูลใด ๆ ในดิกชันนารี ทำได้หลายรูปแบบ ดังต่อไปนี้

```
ตัวแปรดิกชันนารี[key]
ตัวแปรดิกชันนารี.get(key)
```

ตัวอย่างที่ 3.13

```
evaluation_dict = {
  "A": "very good",
  "B": "good",
  "C": "fair",
  "D" : "poor"
}
print(evaluation_dict["A"])
print(evaluation_dict.get("C"))
```

ผลลัพธ์

```
very good
fair
```

การแก้ไขข้อมูลในดิกชันนารี

การแก้ไขข้อมูลในดิกชันนารี ทำได้โดยระบุตำแหน่ง key ที่ต้องการแก้ไขข้อมูล และกำหนดค่าข้อมูลใหม่ลงไป ดังรูปแบบต่อไปนี้

```
ตัวแปรดิกชันนารี[key] = ค่าข้อมูลใหม่
```

การเพิ่มข้อมูลในดิกชันนารี

การเพิ่มข้อมูลในดิกชันนารี ทำได้ดังรูปแบบต่อไปนี้

```
ตัวแปรดิกชันนารี[key] = ค่าข้อมูลที่ต้องการเพิ่ม
```

การลบข้อมูลในดิกชันนารี

การลบข้อมูลในดิกชันนารี ทำได้ดังรูปแบบต่อไปนี้

```
ตัวแปรดิกชันนารี.pop(key)
del ดิกชันนารีที่ต้องการลบข้อมูล[key]
```

ตัวอย่างที่ 3.14

```
evaluation_dict = {
  "A": "very good",
  "B": "good",
  "C": "fair",
  "D" : "poor"
}
```

```
evaluation_dict["A"] = "excellent"
evaluation_dict["F"] = "fail"
print(evaluation_dict)
evaluation_dict.pop("F")
del evaluation_dict["D"]
print(evaluation_dict)
```

ผลลัพธ์

```
{'A': 'excellent', 'B': 'good', 'C': 'fair', 'D': 'poor', 'F': 'fail'}
{'A': 'excellent', 'B': 'good', 'C': 'fair'}
```

****

- ห้ามใช้ชื่อซ้ำกับ**คำสงวน (Reserved Word)** ในภาษา Python ดังนี้

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

หมายเหตุ การใช้ชื่อตัวแปรควรให้สื่อความหมาย จะทำให้ผู้อ่านเข้าใจและสามารถปรับปรุงแก้ไขได้ง่าย

การใช้งานตัวแปรในภาษา Python นั้น ไม่ต้องมีการประกาศหรือกำหนดตัวแปรขึ้นมาก่อน อีกทั้งไม่ต้องกำหนดชนิดของข้อมูล ผู้อ่านสามารถกำหนดค่าข้อมูลหรือการคำนวณใดๆ ที่ต้องการแล้วเก็บค่าผลลัพธ์ในตัวแปรได้เลย Python จะรับรู้ชนิดข้อมูลจากบริบทของคำสั่ง ซึ่งหมายความว่าชนิดข้อมูลจะเป็นไปตามค่าของข้อมูลที่จัดเก็บนั่นเอง

ตัวอย่างเช่น

```
name = "I am Python"   #ตัวแปร name มีค่าเท่ากับ I am Python มีชนิดเป็นข้อความ
score = 86             #ตัวแปร score มีค่าเท่ากับ 86 มีชนิดเป็นจำนวนเต็ม
grade = 4.0            #ตัวแปร grade มีค่าเท่ากับ 4.0 มีชนิดเป็นจำนวนทศนิยม
```

ชนิดของข้อมูล (Data Type)

ในการเขียนโปรแกรมจะมีการใช้งานข้อมูลอยู่ตลอดเวลา ซึ่งข้อมูลแต่ละชนิดก็จะมีรูปแบบและขอบเขตที่แตกต่างกัน ดังนั้น ก่อนผู้อ่านจะเริ่มเขียนโปรแกรม ควรรู้จักกับข้อมูลชนิดต่างๆ ก่อน

ข้อมูลชนิดตัวเลข (Number)

ข้อมูลชนิดตัวเลข (Number) จะประกอบด้วยเลขจำนวนเต็ม, เลขจำนวนทศนิยม และเลขจำนวนเชิงซ้อน โดยมีรายละเอียดดังนี้

1. **ข้อมูลเลขจำนวนเต็ม (Integer Number)** ประกอบด้วยเลขจำนวนเต็มที่เป็นทั้งค่าบวก ค่าลบ และยังสามารถใช้กับเลขฐานได้ดังนี้
   - เลขฐานสอง ใช้ตัวอักษร 0b นำหน้า
   - เลขฐานแปด ใช้ตัวอักษร 0o นำหน้า
   - เลขฐานสิบหก ใช้ตัวอักษร 0x นำหน้า

2. **ข้อมูลเลขจำนวนทศนิยม (Floating Point Number)** ประกอบด้วยเลขจำนวนที่เป็นทศนิยม มีเครื่องหมาย . คั่นระหว่างตัวเลข และสามารถเขียนในรูปแบบของเลขยกกำลังสิบ (Exponential Form) ได้อีกด้วย
3. **ข้อมูลเลขจำนวนเชิงซ้อน (Complex Number)** ประกอบด้วยเลขจำนวนเชิงซ้อน ซึ่งจะถูกใช้ในงานทางวิทยาศาสตร์และวิศวกรรม โดยเขียนอยู่ในรูปแบบของคู่ x+yi โดยเรียก x ว่า จำนวนจริง และเรียก y ว่า ส่วนจินตภาพ และจะใช้ j แทน i

Python จะรับรู้ชนิดข้อมูลแต่ละประเภทจากการที่ผู้อ่านกำหนดค่าลงในชุดคำสั่ง ดังนั้น ผู้อ่านจะต้องพึงระวังในการกำหนดค่าข้อมูลกับตัวแปร ให้ตรงตามชนิดข้อมูลที่ต้องการใช้งาน

ตัวอย่างเช่น

```
A = 10      #ตัวแปร A มีค่าเท่ากับ 10 มีชนิดเป็นจำนวนเต็ม
B = 0b1010  #ตัวแปร B เก็บข้อมูลเลขฐานสอง มีค่าเท่ากับ 10 มีชนิดเป็นจำนวนเต็ม
C = 0o12    #ตัวแปร C เก็บข้อมูลเลขฐานแปด มีค่าเท่ากับ 10 มีชนิดเป็นจำนวนเต็ม
D = 0xA     #ตัวแปร D เก็บข้อมูลเลขฐานสิบหก มีค่าเท่ากับ 10 มีชนิดเป็นจำนวนเต็ม
E = 10.0    #ตัวแปร E มีค่าเท่ากับ 10.0 มีชนิดเป็นจำนวนทศนิยม
F = 10x+2j  #ตัวแปร F มีค่าเท่ากับ 10x+2j มีชนิดเป็นจำนวนเชิงซ้อน
G = 15e-2   #ตัวแปร G มีค่าเท่ากับ 0.15 มีชนิดเป็นจำนวนทศนิยม
H = 35e+2   #ตัวแปร H มีค่าเท่ากับ 3500.0 มีชนิดเป็นจำนวนทศนิยม
```

ในส่วนของการดำเนินการกับข้อมูลชนิดตัวเลขในแต่ละประเภท ผู้อ่านจะได้ศึกษาและเรียนรู้ในหัวข้อตัวดำเนินการต่อไป

ข้อมูลชนิดค่าความจริง (Boolean)

ข้อมูลชนิดค่าความจริง (Boolean) เป็นข้อมูลที่มีค่าความจริงเป็นจริงหรือเป็นเท็จเท่านั้น ซึ่งจะแทนค่าความจริงที่เป็นจริงด้วย True และจะแทนค่าความจริงที่เป็นเท็จด้วย False

ตัวอย่างเช่น

```
A = 9>5     #ตัวแปร A มีค่าเท่ากับ True
B = 9-5>9+5 #ตัวแปร B มีค่าเท่ากับ False
C = True    #ตัวแปร C มีค่าเท่ากับ True
D = False   #ตัวแปร D มีค่าเท่ากับ False
```

ข้อมูลชนิด None

ข้อมูลชนิด None เป็นข้อมูลที่ไม่มีค่าข้อมูล หรือยังไม่ได้รับการกำหนดค่า ซึ่งไม่ใช่ข้อมูลที่เก็บค่าศูนย์ หรือข้อมูลที่เป็นค่าว่างแต่อย่างใด

ผู้อ่านสามารถนำ None มาใช้ตรวจสอบสถานะของการดำเนินการใดๆ ที่ไม่สามารถให้ค่าผลลัพธ์ได้ โดยใช้รูปแบบเหมือนกับการตรวจสอบข้อมูลชนิด Boolean

ตัวอย่างเช่น

```
ans = None          #ตัวแปร ans มีค่าเท่ากับ None
  X = 3             #ตัวแปร X มีค่าเท่ากับ 3
  Y = 0             #ตัวแปร Y มีค่าเท่ากับ 0
If Y != 0
ans = X/Y

#ผลที่ได้คือ ตัวแปร ans มีค่าเท่ากับ None
```

ข้อมูลแบบเรียงลำดับ (Sequence)

ข้อมูลแบบเรียงลำดับ (Sequence) จะประกอบด้วยข้อมูลที่จัดเก็บเรียงลำดับต่อกัน ได้แก่ ข้อความ (String), ลิสต์ (List) และทูเปิล (Tuple) โดยมีรายละเอียดดังนี้

1. **ข้อมูลชนิดข้อความ (String)** เป็นข้อมูลที่ประกอบด้วยข้อความ ตัวอักษร ตัวเลขที่ไม่สามารถคำนวณได้ จำนวนตั้งแต่ 1 ตัว มาเรียงต่อกันภายในเครื่องหมาย double quote ("...") หรือเครื่องหมาย single quote ('...')

   โดยการกำหนดค่าตัวแปรชนิดนี้ ผู้อ่านจะต้องเขียนให้อยู่ในบรรทัดเดียวกัน แต่สามารถใช้เครื่องหมาย backslash (\) แทรก หากต้องการแยกเป็นบรรทัดใหม่

   ข้อมูลในกลุ่มนี้ ได้แก่ ตัวอักษร (A ถึง Z, a ถึง z) ตัวเลข (0 ถึง 9) และสัญลักษณ์พิเศษต่างๆ เช่น + - = $ * เป็นต้น

   ในกรณีที่มีข้อมูลเพียง 1 ตัว Python จะรับรู้ชนิดของข้อมูลนี้ว่าเป็น**ชนิดตัวอักขระ (Character)** ซึ่งจะมีตัวเลขแทนแต่ละตัวอักขระไว้สำหรับใช้ในการดำเนินการ เรียกตัวเลขเหล่านั้นว่า **รหัสแอสกี้ (Ascii Code)**

ตัวอย่างเช่น

```
strA = "3020158"        #ตัวแปร strA มีค่าเท่ากับ 3020158
strB = '3020158'        #ตัวแปร strB มีค่าเท่ากับ 3020158
strC = '"Monday"'       #ตัวแปร strC มีค่าเท่ากับ "Monday"
strD = "'Jan'"          #ตัวแปร strD มีค่าเท่ากับ 'Jan'
strE = "A & " \
  "B"                   #ตัวแปร strE มีค่าเท่ากับ A & B
strF = "A"              #ตัวแปร strF มีค่าเท่ากับ A
noF = ord(strF)         #ord เป็นฟังก์ชันแปลง "A" เป็นค่าตัวเลข มีค่าเท่ากับ 65
```

2. **ข้อมูลชนิดลิสต์ (List)** เป็นข้อมูลที่ประกอบด้วยรายการข้อมูลชนิดใดๆ เรียงต่อกันภายในเครื่องหมายวงเล็บก้ามปู [...] และคั่นด้วยเครื่องหมายคอมม่า (,) ข้อมูลในเครื่องหมายวงเล็บก้ามปู [...] สามารถเป็นค่าว่างได้ ซึ่งจะเรียกว่า **ลิสต์ว่าง (Empty List)**

ตัวอย่างเช่น

```
listA = ["Anna", 20, "Bangkok", "Aree", 22, "Chiangmai"]
#ตัวแปร listA มีค่าเท่ากับ ["Anna", 20, "Bangkok", "Aree", 22, "Chiangmai"]

listA.append("Adam")
#ตัวแปร listA มีค่าเท่ากับ ["Anna", 20, "Bangkok", "Aree", 22, "Chiangmai", "Adam"]
#ตัวแปร listA[0] มีค่าเท่ากับ Anna
#ตัวแปร listA[1] มีค่าเท่ากับ 20
#ตัวแปร listA[2] มีค่าเท่ากับ Bangkok
#ตัวแปร listA[3] มีค่าเท่ากับ Aree
#ตัวแปร listA[4] มีค่าเท่ากับ 22
#ตัวแปร listA[5] มีค่าเท่ากับ Chiangmai
```

3. **ข้อมูลชนิดทูเปิล (Tuple)** เป็นข้อมูลที่ประกอบด้วยรายการข้อมูลชนิดใดๆ เรียงต่อกัน คั่นด้วยเครื่องหมายคอมม่า (,) และสามารถเป็นค่าว่างได้เหมือนกับข้อมูลชนิดลิสต์ แต่จะเรียงต่อกันภายในเครื่องหมายวงเล็บ (...) หรือไม่ต้องมีเครื่องหมายวงเล็บ (...) ครอบก็ได้ และไม่สามารถเพิ่ม เปลี่ยนแปลง หรือลบข้อมูลในทูเปิลได้

ตัวอย่างเช่น

```
tupleA = ("Anna", 20, "Bangkok", "Aree", 22, "Chiangmai")
#ตัวแปร strA มีค่าเท่ากับ ("Anna", 20, "Bangkok", "Aree", 22, "Chiangmai")

tupleB = "Anna", 20, "Bangkok", "Aree", 22, "Chiangmai"
#ตัวแปร strB มีค่าเท่ากับ ("Anna", 20, "Bangkok", "Aree", 22, "Chiangmai")

#ตัวแปร tupleA[0] มีค่าเท่ากับ Anna
#ตัวแปร tupleA[1] มีค่าเท่ากับ 20
#ตัวแปร tupleA[2] มีค่าเท่ากับ Bangkok
#ตัวแปร tupleB[3] มีค่าเท่ากับ Aree
#ตัวแปร tupleB[4] มีค่าเท่ากับ 22
#ตัวแปร tupleB[5] มีค่าเท่ากับ Chiangmai
**ตัวแปร tupleA และ tupleB ไม่สามารถใช้ฟังก์ชัน append() เนื่องจากเป็นข้อมูลชนิด tuple**
```

ข้อมูลชนิดเซต (Set)

ข้อมูลชนิดเซต (Set) จะประกอบด้วยรายการข้อมูลชนิดใดๆ เรียงต่อกัน คั่นด้วยเครื่องหมายคอมม่า (,) สามารถเป็นค่าว่างได้ และสามารถเพิ่ม เปลี่ยนแปลง หรือลบข้อมูลในเซตได้เหมือนกับข้อมูลชนิดลิสต์ แต่มีข้อแตกต่างกันคือ ข้อมูลชนิดเซตจะเรียงต่อกันภายในเครื่องหมายวงเล็บปีกกา {...} การจัดเก็บข้อมูลจะไม่มีลำดับ กล่าวคือ ไม่สามารถเข้าถึงข้อมูลด้วยเลขตำแหน่งข้อมูล และจะไม่เก็บข้อมูลที่ซ้ำกัน

ตัวอย่างเช่น

```
setA = {"Anna", 20, "Bangkok", "Aree", 22, "Chiangmai", 22}
#ตัวแปร setA มีค่าเท่ากับ {"Chiangmai", "Aree", "Anna", 20, 22, "Bangkok"}

setA.add("Aree")
#ตัวแปร setA มีค่าเท่ากับ {20, 22, "Chiangmai", "Aree", "Anna", "Bangkok"}
**เมื่อแสดงผลข้อมูลใน setA ข้อมูลที่ได้จะไม่เรียงลำดับ และข้อมูลที่ซ้ำกันจะแสดงออกมาเพียงค่าเดียวเท่านั้น**
```

ข้อมูลชนิดดิกชันนารี (Dictionary)

ข้อมูลชนิดดิกชันนารี (Dictionary) เป็นข้อมูลที่ประกอบด้วยรายการข้อมูลชนิดใดๆ เรียงต่อกันและคั่นด้วยเครื่องหมายคอมม่า (,) เหมือนกับข้อมูลชนิดลิสต์ แต่จะเรียงต่อกันภายในเครื่องหมายวงเล็บปีกกา {...} และอยู่ในรูปแบบที่ประกอบด้วย 2 ส่วนคือ key: value โดยที่ key เป็นข้อมูลที่เป็นตัวชี้ และ value เป็นค่าข้อมูลที่สามารถอ้างอิงมาจาก key ได้

ตัวอย่างเช่น

```
scores = {'Anna': 20.6, 'Meena': 22.1, 'Wanna': 23.0}
scores['Weena'] = 21.7

#ตัวแปร scores['Weena'] มีค่าเท่ากับ 21.7
#ตัวแปร scores['Meena'] มีค่าเท่ากับ 22.1
#ตัวแปร scores['Wanna'] มีค่าเท่ากับ 23.0
#ตัวแปร scores['Anna'] มีค่าเท่ากับ 20.6
```

ส่วนของการใช้งานในรายละเอียด และฟังก์ชันสำหรับการดำเนินการกับข้อมูลแบบเรียงลำดับ ข้อมูลชนิดเซต และข้อมูลชนิดดิกชันนารี ผู้อ่านจะได้ศึกษาและเรียนรู้ในบทที่ 6 ต่อไป

การตรวจสอบชนิดของข้อมูล (Data Type Checking)

เนื่องจากในภาษา Python ผู้อ่านไม่ต้องประกาศชนิดข้อมูลของตัวแปร Python เราสามารถรับรู้ชนิดข้อมูลจากค่าที่กำหนดในชุดคำสั่ง แต่ในบางครั้งก็มีความจำเป็นต้องทราบชนิดข้อมูลของตัวแปรก่อนที่จะประมวลผลอย่างใดอย่างหนึ่ง ดังนั้น ผู้อ่านสามารถใช้ฟังก์ชัน **type()** ตรวจสอบชนิดของข้อมูลได้ ซึ่งมีรูปแบบการทำงานดังนี้

```
type(variable)
```

โดยที่ **variable** เป็นตัวแปรที่ต้องการตรวจสอบชนิดข้อมูล

ตัวอย่างเช่น

```
a = 5
print(type(a))
#ชนิดข้อมูลของตัวแปร a คือ <class 'int'>
b = 5.0
print(type(b))
#ชนิดข้อมูลของตัวแปร b คือ <class 'float'>
c = [1, 2, 3]
print(type(c))
#ชนิดข้อมูลของตัวแปร c คือ <class 'list'>
d = (1, 2, 3)
print(type(d))
#ชนิดข้อมูลของตัวแปร d คือ <class 'tuple'>
e = {1, 2, 3}
print(type(e))
#ชนิดข้อมูลของตัวแปร e คือ <class 'set'>
f = {1: "one", 2: "two", 3: "three"}
print(type(f))
#ชนิดข้อมูลของตัวแปร f คือ <class 'dict'>
g = float
print(type(g))
#ชนิดข้อมูลของตัวแปร g คือ <class 'type'>
h = len
print(type(h))
#ชนิดข้อมูลของตัวแปร h คือ <class 'builtin_function_or_method'>
```

จากตัวอย่างข้างต้น นอกจากชนิดข้อมูลที่ใช้แล้ว ภาษา Python ยังสามารถตรวจสอบชื่อที่ใช้เป็นชนิดข้อมูลหรือชื่อที่ใช้เป็นฟังก์ชันในภาษา Python ได้ เช่น

- float ตรวจสอบได้เป็น <class 'type'> ซึ่งหมายถึง ชนิดข้อมูล
- len ตรวจสอบได้เป็น <class 'builtin_function_or_method'> ซึ่งหมายถึง ฟังก์ชัน

เมื่อผู้อ่านทราบชนิดข้อมูลโดยการใช้ฟังก์ชัน type() แล้ว หากต้องการจะแปลงชนิดข้อมูลก่อนจะประมวลผลใดๆ ก็สามารถทำได้ ซึ่งผู้อ่านจะได้ศึกษาและเรียนรู้ในหัวข้อถัดไป

การแปลงชนิดของข้อมูล (Data Type Conversion)

การแปลงชนิดข้อมูลในภาษา Python ผู้อ่านสามารถใช้ Built-in Function ที่มีอยู่แล้วใน Python ได้เลย โดยฟังก์ชันเหล่านั้นจะมีชื่อเหมือนกับชนิดข้อมูลนั้นๆ ซึ่งก็คือ ชื่อของคลาสนั่นเอง ตัวอย่างเช่น

- ฟังก์ชัน int() ใช้แปลงข้อมูลประเภทใดๆ ให้เป็นชนิดตัวเลขจำนวนเต็ม
- ฟังก์ชัน str() ใช้แปลงข้อมูลประเภทใดๆ ให้เป็นชนิดตัวอักษร

ตารางแสดงตัวอย่างของฟังก์ชันสำหรับการแปลงข้อมูลในภาษา Python

ชื่อฟังก์ชัน	คำอธิบาย
int(x [,base])	แปลงข้อมูล x จากฐานที่กำหนด base ให้เป็นเลขจำนวนเต็ม
float(x)	แปลงข้อมูล x ให้เป็นเลขจำนวนทศนิยม
complex(real [,im])	สร้างตัวเลขจำนวนเชิงซ้อนจากค่า real และค่า imagine
str(x)	แปลงข้อมูล x ให้เป็นตัวอักษร
tuple(s)	แปลงข้อมูลแบบเรียงลำดับ s ให้เป็นข้อมูลทูเปิล
list(s)	แปลงข้อมูลแบบเรียงลำดับ s ให้เป็นข้อมูลลิสต์
set(s)	แปลงข้อมูลแบบเรียงลำดับ s ให้เป็นข้อมูลเซต
dict(d)	แปลงข้อมูล d ให้เป็นข้อมูลดิกชันนารี
chr(x)	แปลงข้อมูลเลขจำนวนเต็ม x ให้เป็นตัวอักขระ
ord(x)	แปลงข้อมูลตัวอักขระ x ให้เป็นค่าเลขจำนวนเต็ม
hex(x)	แปลงข้อมูลตัวเลข x ให้เป็นเลขฐานสิบหก ชนิดตัวอักษร
oct(x)	แปลงข้อมูลตัวเลข x ให้เป็นเลขฐานแปด ชนิดตัวอักษร
bin(x)	แปลงข้อมูลตัวเลข x ให้เป็นเลขฐานสอง ชนิดตัวอักษร

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ASCII ranges for uppercase, lowercase characters and digits are as follows:

'A' – 'Z' : 65 - 90

'a' – 'z' : 97 - 122

'0' – '9' : 48 – 57