แผนบริหารการสอนประจำบทที่ 6

ทัพเพิล เซตและดิกชันนารี

หัวข้อประจำบท

1. ทัพเพิล
2. เซต
3. ดิกชันนารี

วัตถุประสงค์เชิงพฤติกรรม

1. เพื่อให้ผู้เรียนสามารถอธิบายเกี่ยวกับทัพเพิลได้
2. เพื่อให้ผู้เรียนสามารถอธิบายเกี่ยวกับเซตได้
3. เพื่อให้ผู้เรียนสามารถอธิบายเกี่ยวกับดิกชันนารีได้
4. เพื่อให้ผู้สามารถบอกความแตกต่าง ทัพเพิล เซต และดิกชันนารี และเลือกนำไปใช้งานได้อย่างเหมาะสม
5. เพื่อให้ผู้เรียนสามารถนำความรู้เรื่อง ทัพเพิล เซต และดิกชันนารี ประยุกต์ใช้งานในการแก้โจทย์ปัญหาได้อย่างถูกต้อง

วิธีการสอนและกิจกรรม

1. ผู้สอนบรรยายในชั้นเรียน และโปรแกรมนำเสนอ ตามหัวข้อเนื้อหาในเอกสารประกอบการสอน
2. ผู้เรียนทำแบบฝึกหัดท้ายบท และฝึกเขียนโปรแกรมด้วยคอมพิวเตอร์

สื่อการเรียนการสอน

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

การวัดผลและการประเมินผล

1. สังเกตจากการอภิปราย การตอบคำถาม และซักถามระหว่างเรียน
2. การปฏิบัติบนเครื่องคอมพิวเตอร์
3. การทำแบบฝึกหัดท้ายบท

บทที่ 6
ทัพเพิล เซตและดิกชันนารี

การเก็บข้อมูลแบบแถวลำดับภาษาไพธอน นอกเหนือจากการเก็บข้อมูลในรูปแบบของลีสต์ ยังมีเครื่องมือที่มีลักษณะของการเก็บข้อมูลแบบแถวลำดับที่มีคุณสมบัติและจุดประสงค์การนำไปใช้แตกต่างจากลีสต์ ประกอบไปด้วย ทัพเพิล เซตและดิกชันนารี การเก็บข้อมูลแบบแถวลำดับมีประโยชน์มากในการนำไปใช้งานของโปรแกรม ข้อดีและข้อเสียแตกต่างกันขึ้นอยู่กับลักษณะงานโปรแกรมที่นำไปใช้ โครงสร้างและลักษณะการเก็บข้อมูลแบบแถวลำดับดังกล่าว มีรายละเอียดดังต่อไปนี้

ทัพเพิล

ทัพเพิล (Tuple) หมายถึง การใช้ตัวแปรเดียวเก็บข้อมูลต่างประเภทกันได้ สามารถเก็บข้อมูลได้ปริมาณมาก ทัพเพิลจะแตกต่างจากลีสต์ที่เก็บข้อมูลประเภทค่าคงที่ ไม่สามารถเปลี่ยนแปลงได้ การเพิ่มหรือลบสมาชิกภายในทัพเพิลโดยตรงไม่ได้ สามารถเข้าถึงข้อมูลได้เร็วผ่านตัวชี้หรือดัชนีเหมือนลีสต์ (สุชาติ คุ้มมณี, 2558) การจัดเรียงแถวลำดับของข้อมูลในทัพเพิล มีการทำงานคล้ายกับตัวแปรลีสต์ แต่แตกต่างกันตรงที่ลีสต์เป็นโครงสร้างข้อมูลที่สามารถเปลี่ยนแปลงได้ ส่วนทัพเพิลนั้นเป็นโครงสร้างข้อมูลที่ไม่สามารถเปลี่ยนแปลงได้ (Elkner, 2014) ทัพเพิลอยู่ในกลุ่มของลีสต์ แต่การใช้งานบ่อยครั้งจะมีความแตกต่าง ขึ้นอยู่กับสถานะการณ์และจุดประสงค์ของการนำไปใช้ ทัพเพิลลักษณะข้อมูลจะไม่เปลี่ยนรูป โดยทั่วไปจะบรรจุส่วนประกอบที่อยู่ภายในแตกต่างกันเพื่อการใช้งานโดยไม่สามารถเพิ่มหรือจัดเรียงข้อมูลได้ ส่วนลีสต์สามารถเปลี่ยนรูปข้อมูล ส่วนประกอบภายใตลีสต์จะมีลักษณะเหมือนกัน สามารถเข้าไปจัดการส่วนประกอบที่อยู่ภายในในลีสต์ได้ (Rossum, 2013)

ตัวแปรชนิดทัพเพิลที่อยู่ในกลุ่มของตัวแปรแบบแถวลำดับ ค่าที่จัดเก็บจะจัดเรียงตามลำดับการใส่ค่าข้อมูลไว้ในตัวแปรและจะกำหนดหมายเลขลำดับ ให้กับค่าในโครงสร้างของตัวแปร แต่เนื่องจากตัวแปรชนิด ทัพเพิล จัดเป็นโครงสร้างแบบ Immutable คือ โครงสร้างที่มีการกำหนดค่าข้อมูลลงในตัวแปรที่มีโครงสร้าบแบบนี้เรียบร้อย การนำตัวแปรที่อยู่ในโครงสร้างแบบนี้ไปใช้ จะไม่สามารถเพิ่มหรือลบค่าข้อมูลที่จัดเก็บในโครงสร้างนี้ได้ ยกเว้นการแก้ไขค่าข้อมูลที่มีอยู่แล้วในตัวแปร (โชติพันธุ์ หล่อเลิศสุนทร และ ฐิตะพันธุ์ หล่อเลิศสุนทร, 2559)

จากลักษณะและคุณสมบัติของโครงสร้างและการเก็บข้อมูลทัพเพิล ขั้นตอนการจัดการและการใช้งานทัพเพิลมีหลากหลายวิธีการ มีรายละเอียดดังต่อไปนี้

1. การสร้างตัวแปรและการเข้าถึงสมาชิกชนิดทัพเพิล

ทัพเพิลประกอบด้วยสมาชิกของข้อมูลที่แตกต่างกัน โดยใช้เครื่องหมาย คอมมาหรือจุลภาค (,) (ราชบัณฑิตยสถาน, 2564) คั่นระหว่างข้อมูลและใช้สัญลักษณ์นี้เป็นหลักของการสร้างตัวแปรแบบทัพเพิล (Rossum, 2013) โดยทั่วไปเพื่อความเป็นระเบียบในการจัดลำดับข้อมูล จะใช้สัญลักษณ์วงเล็บ หรือ “( )” ในการจัดรูปแบบของตัวแปรประเภททัพเพิล มีรายละเอียดและลักษณะการสร้างตัวแปร ดังโปรแกรมตัวอย่างที่ 6.1

ตัวอย่างที่ 6.1 การสร้างตัวแปรทัพเพิล

บรรทัดที่	คำสั่ง
1	tpl1 = 1,3,5,7,9
2	tpl2 =(4,5,6,7,8)
3	tpl3 = 10,
4	tpl4, tpl5, tpl6 = (1,2,3),5,6
5	print(tpl1,type(tpl1))
6	print(tpl2,type(tpl2))
7	print(tpl3,type(tpl3))
8	print(tpl4,type(tpl4))
9	print(tpl5,type(tpl5))
10	print(tpl6,type(tpl6))
ผลลัพธ์	
(1, 3, 5, 7, 9) <class 'tuple'> (4, 5, 6, 7, 8) <class 'tuple'> (10,) <class 'tuple'> (1, 2, 3) <class 'tuple'> 5 <class 'int'> 6 <class 'int'>	

การเก็บข้อมูลของทัพเพิล มีลักษณะการเก็บข้อมูลแบบลำดับเช่นเดียวกันกับลีสต์ มีรูปแบบการประกาศตัวแปรได้หลายประเภท โดยใช้เครื่องหมายคอมมาในการแยกสมาชิก สมาชิกสามารถมีชนิดของข้อมูลที่แตกต่างกันได้และสามารถประกาศตัวแปร กำหนดค่าได้หลายตัวพร้อมกัน ตำแหน่งในสมาชิกของทัพเพิล สามารถอ้างอิงด้วยตัวชี้ลำดับของตำแหน่ง ดังโปรแกรมตัวอย่างที่ 6.1

ตัวอย่างที่ 6.2 การเก็บข้อมูลภายในลิสต์ลักษณะของการซ้อนกันของข้อมูล

บรรทัดที่	คำสั่ง
1	tpl = (0.5,-44),(5,2,3),88,-9
2	print(tpl[0])
3	print(tpl[1])
4	print(tpl[2])
ผลลัพธ์	
(0.5, -44) (5, 2, 3) 88	

การเก็บข้อมูลภายในลิสต์สามารถเก็บข้อมูลในลักษณะของการซ้อนกันของข้อมูลแบบหลายมิติได้ มีลักษณะการเก็บข้อมูลเช่นเดียวกันกับลิสต์ การอ้างอิงตำแหน่งของข้อมูล จะใช้เลขลำดับอ้างอิง ดังโปรแกรมตัวอย่างที่ 6.2

2. ความแตกต่างระหว่างทัพเพิลกับลิสต์

ทัพเพิลและลิสต์มีการเก็บข้อมูลแบบแถวลำดับเช่นเดียวกัน แต่จะมีส่วนที่แตกต่างกันในการใช้งาน มีข้อกำหนดรายละเอียดดังตารางที่ 6.1

ตารางที่ 6.1 ความแตกต่างระหว่างทัพเพิลกับลิสต์

ลักษณะ	ลิสต์	ทัพเพิล
การเปลี่ยนแปลงข้อมูลและโครงสร้าง	เปลี่ยนแปลง	ไม่เปลี่ยนแปลง
การสร้าง	lst = [i, j]	tpl = (i, j)
การเข้าถึง	a = lst[i]	a = tpl[i]
การแก้ไข	lst[i] = a	ทำไม่ได้
การเพิ่ม	lst += [a]	ทำไม่ได้
การลบ	del lst[i]	ทำไม่ได้
การเลื่อน	lst[i:j:k]	tpl[i:j:k]
การเลื่อนและกำหนดค่า	lst[i:j] = [ ]	ทำไม่ได้
คำสั่งวนซ้ำ	for elem in lst:.	for elem in tpl:. . .

จากตารางที่ 6.1 เปรียบเทียบความแตกต่างการทำงานระหว่างลีสต์กับทัพเพิล โดยการเข้าถึง การเลื่อนและการใช้คำสั่งวนซ้ำ มีลักษณะการทำงานคล้ายกัน แต่ทัพเพิลไม่สามารถ แก้ไข เพิ่ม ลบ เลื่อนและกำหนดค่าของทัพเพิลได้ ดังโปรแกรมตัวอย่างที่ 6.3

ตัวอย่างที่ 6.3 การไม่อนุญาตให้จัดการข้อมูลภายในทัพเพิล

บรรทัดที่	คำสั่ง
1	lst = (5, 8)
2	del lst[0]
3	print (lst)
ผลลัพธ์	
Traceback (most recent call last): File "C:\Python36\test.py", line 2, in <module> del lst[0] TypeError: 'tuple' object doesn't support item deletion	

โปรแกรมตัวอย่างที่ 6.3 พบว่าการกระทำที่จะให้เกิดการเปลี่ยนแปลงภายในทัพเพิล จะไม่สามารถทำได้ จะสามารถทำได้ในบางกรณีเท่านั้น ดังรายละเอียดของตารางที่ 6.1

3. การใช้คำสั่งวนซ้ำเข้าถึงสมาชิกในทัพเพิล

การเข้าถึงสมาชิกทัพเพิล สามารถใช้คำสั่งวนซ้ำ เพื่ออ่านข้อมูลของสมาชิกได้โดยคำสั่งวนซ้ำประกอบด้วยคำสั่ง for และ for...in range โดยคำสั่ง for มีรูปแบบของการเข้าถึงสมาชิกภายในทัพเพิล มีรายละเอียดดังต่อไปนี้

ตัวอย่างที่ 6.4 การใช้คำสั่ง for เข้าถึงสมาชิกทัพเพิล

บรรทัดที่	คำสั่ง
1	tpl = ('I','have','money',33000,'bath.')
2	for i in tpl:
3	print (i)
ผลลัพธ์	
I have	

money 33000 bath.

นอกเหนือจากคำสั่ง for การใช้ถึงสมาชิกภายในทัพเพิล สามารถใช้คำสั่งวนซ้ำ for...in rang ในการเข้าถึงสมาชิกภายในทัพเพิลได้ มีรายละเอียดดังโปรแกรมตัวอย่างที่ 6.5

ตัวอย่างที่ 6.5 การใช้คำสั่ง for...in range เข้าถึงสมาชิกในทัพเพิล

บรรทัดที่	คำสั่ง
1	tpl1 = ('I','have','money',33000,'bath.')
2	for i in range(len(tpl1)):
3	print ('ตำแหน่ง : ',i,' = ',tpl1[i])
4	print('-'*15)
5	for i in range(2):
6	print ('ตำแหน่ง : ',i,' = ',tpl1[i])
7	print('-'*15)
8	for i in range(len(tpl1)):
9	if tpl1[i]=='bath.':
10	print ('ตำแหน่ง : ',i,' = ',tpl1[i])

ผลลัพธ์

```
ตำแหน่ง :  0  =  I
ตำแหน่ง :  1  =  have
ตำแหน่ง :  2  =  money
ตำแหน่ง :  3  =  33000
ตำแหน่ง :  4  =  bath.
---------------
ตำแหน่ง :  0  =  I
ตำแหน่ง :  1  =  have
---------------
ตำแหน่ง :  4  =  bath.
```

การใช้คำสั่งควบคุมทิศทางการทำงานของโปรแกรมแบบวนซ้ำ นอกเหนือจากการเข้าถึงสมาชิกทั่วไปทั้งหมด ยังสามารถกำหนดจำนวนขนาดของการเข้าถึงสมาชิกได้ดังตัวอย่างบรรทัดที่ 5 มีการเข้าถึงสมาชิกเพียง 2 ตำแหน่งเท่านั้น และยังมีการใช้งานมากสำหรับการท่องไปภายในทัพเพิล เพื่อตรวจสอบหรือดึงข้อมูลภายของสมาชิกภายในทัพเพิลออกมาใช้งาน ตามเงื่อนไขที่กำหนด โดยตัวอย่างบรรทัดที่ 9 ให้นำข้อมูลสมาชิกที่มีค่าเท่ากับ bath เท่านั้นขึ้นมาแสดง ดังผลลัพธ์ของโปรแกรมตัวอย่างที่ 6.5 จะเห็นได้ว่าโครงสร้างการทำงานของทัพเพิล สามารถใช้คำสั่งวนซ้ำในการทำงานแบบซ้อนได้

4. การปฏิบัติการของทัพเพิล

การปฏิบัติการทัพเพิล คือ การดำเนินการข้อมูลของสมาชิกภายในทัพเพิล โดยใช้สัญลักษณ์ทางคณิตศาสตร์ สัญลักษณ์ที่นำมาใช้คือ “+” และ “*” มีรายละเอียดของการใช้ปฏิบัติการสัญลักษณ์แต่ละชนิด ดังโปรแกรมตัวอย่างที่ 6.6

ตัวอย่างที่ 6.6 การปฏิบัติการของทัพเพิล

บรรทัดที่	คำสั่ง
1	tpl1 = ('I','have','money',33000,'bath.')
2	tpl2 = ('Computer','Programming')
3	tpl3=tpl1+tpl2
4	tpl4=tpl2*3
5	print(tpl3)
6	print(tpl4)
7	print(tpl3+tpl4)
ผลลัพธ์	
('I', 'have', 'money', 33000, 'bath.', 'Computer', 'Programming') ('Computer', 'Programming', 'Computer', 'Programming', 'Computer', 'Programming') ('I', 'have', 'money', 3 3 0 0 0 , 'bath. ', 'Computer', 'Programming', 'Computer', 'Programming', 'Computer', 'Programming', 'Computer', 'Programming')	

การปฏิบัติของทัพเพิลที่ใช้สัญลักษณ์ “+” หมายถึงการรวมกันของทัพเพิล ระหว่างสองทัพเพิล และการใช้สัญลักษณ์ “*” คือ การเพิ่มข้อมูลสมาชิกภายในทัพเพิล โดยการคัดลอกข้อมูลตามจำนวนที่กำหนดและเพิ่มลงในทัพเพิล ดังผลลัพธ์โปรแกรมตัวอย่างที่ 6.6

5. การอ่านข้อมูลภายในทัพเพิล

การเก็บข้อมูลภายในทัพเพิล มีลักษณะของการใช้ตัวเลขระบุตำแหน่ง โดยมีโครงสร้างตำแหน่งของการเก็บข้อมูลสมาชิก ดังตัวอย่างการเก็บข้อมูลตารางที่ 6.2

ตารางที่ 6.2 การอ่านข้อมูลภายในทัพเพิล

ตำแหน่งบวก	[0]	[1]	[2]	[3]	[4]	[5]	[6]
lst	I	have	money	33000	bath.	Computer	Programming
ตำแหน่งลบ	[-7]	[-6]	[-5]	[-4]	[-3]	[-2]	[-1]

การอ่านข้อมูลภายในทัพเพิล สามารถกำหนดรูปแบบของการอ่านข้อมูล โดยใช้สัญลักษณ์ “:” ซึ่งเป็นส่วนที่ระบุเงื่อนไขตำแหน่งในการอ่านข้อมูล โดยจากข้อมูลในตารางที่ 6.2 ผลจากการใช้คำสั่งอ่านข้อมูล มีการใช้รูปแบบคำสั่งที่แตกต่างกัน ดังโปรแกรมตัวอย่างที่ 6.7

ตัวอย่างที่ 6.7 รูปแบบการอ่านข้อมูลภายในทัพเพิล

คำสั่ง	ผลลัพธ์	ความหมาย
tpl = ('I','have','money',33000,'bath.','Computer','Programming')		
print (tpl[:])	('I', 'have', 'money', 33000, 'bath.', 'Computer', 'Programming')	อ่านข้อมูลทั้งหมดภายในทัพเพิล
print (tpl[:4])	('I', 'have', 'money', 33000)	อ่านข้อมูลใน 4 ตำแหน่งแรกของทัพเพิล
print (tpl[4:])	('bath.', 'Computer', 'Programming')	อ่านข้อมูลหลัง 4 ตำแหน่งแรกของทัพเพิล
print (tpl[2:4])	('money', 33000)	อ่านข้อมูลหลังตำแหน่งที่ 2 ไปถึงตำแหน่งที่ 4
print (tpl[1:6:2])	('have', 33000, 'Computer')	อ่านข้อมูลตำแหน่งที่ 1 ไปถึงตำแหน่งที่ 6 โดยข้ามตำแหน่งครั้งละ 2
print (tpl[-1:-5:-2])	('Programming', 'bath.')	อ่านตำแหน่งที่ -1 ถอยหลังมายังตำแหน่งก่อน -5 โดยข้ามตำแหน่งครั้งละ 2

คำสั่ง	ผลลัพธ์	ความหมาย
print (tpl[:-2])	('I', 'have', 'money', 33000, 'bath.')	อ่านตำแหน่งแรกสุด จนถึงตำแหน่งก่อน -2
print (tpl[-2:])	('Computer', 'Programming')	อ่านตำแหน่ง -2 ไปยังตำแหน่งสุดท้าย

6. การแปลงทัพเพิลเป็นลีสต์และแปลงลีสต์เป็นทัพเพิล

การใช้งานทัพเพิลไม่สามารถเพิ่มข้อมูลสมาชิกภายในได้ ดังนั้นจึงมีวิธีการ โดยการแปลงทัพเพิลให้เป็นลีสต์ เพื่อดำเนินการเปลี่ยนแปลงข้อมูลสมาชิกภายในและสามารถแปลงจากลีสต์กลับมาเป็นทัพเพิลได้อีกครั้ง มีรายละเอียดดังโปรแกรมตัวอย่างที่ 6.8

ตัวอย่างที่ 6.8 การแปลงทัพเพิลเป็นลีสต์และแปลงลีสต์เป็นทัพเพิล

บรรทัดที่	คำสั่ง
1	tpl = ('I','have','money',33000,'bath.','Computer','Programming')
2	print(tpl)
3	lst= list(tpl)
4	lst.append("Language")
5	print (lst)
6	tpl= tuple(lst)
7	print (tpl)
ผลลัพธ์	
('I', 'have', 'money', 33000, 'bath.', 'Computer', 'Programming') ['I', 'have', 'money', 33000, 'bath.', 'Computer', 'Programming', 'Language'] ('I', 'have', 'money', 33000, 'bath.', 'Computer', 'Programming', 'Language')	

การแปลงทัพเพิลเป็นลีสต์ จะใช้คำสั่ง list() ในการแปลงตัวแปรและสามารถแปลงตัวแปรกลับจากลีสต์มาเป็นทัพเพิลได้ โดยใช้คำสั่ง tuple() การใช้เทคนิควิธีการแปลงระหว่างทัพเพิลกับลีสต์สามารถนำไปประยุกต์ใช้งานได้ตามสถานะการณ์ การนำวิธีการนี้มาใช้ เกิดจากมีข้อจำกัดของการใช้งานทัพเพิล

เซต

เซต (Set) หมายถึง การใช้หลักการทางคณิตศาสตร์ บ่งบอกถึงสิ่งอยู่ในกลุ่มหรือเซต ลักษณะของสิ่งที่อยู่ในเซตเรียกว่า “สมาชิก” และในเซตจะมีสมาชิกไม่เหมือนกัน สมาชิกในกลุ่มไม่ได้เรียงลำดับ การดำเนินการภายในเซตประกอบไปด้วย การกำจัดสมาชิกที่ซ้ำกันในเซต การทดสอบสมาชิก เช่น union intersection difference เป็นต้น (สุชาติ คุ้มมณี, 2558) ภาษาไพธอนมีเครื่องมือที่ใช้จัดการโครงสร้างของข้อมูล โดยใช้หลักของเซตทางคณิตศาสตร์ ความแตกต่างระหว่างลีสต์ คือ เซตไม่มีการจัดเรียงข้อมูลและสมาชิกในเซตจะไม่มีการซ้ำกัน (Halterman, 2016) ตัวแปรเซตมีรูปแบบของการใช้งานดังต่อไปนี้

1. การใช้งานตัวแปรเซต

การสร้างตัวแปรเซตจะใช้เครื่องหมาย วงเล็บปีกา หรือ “{ }” (ราชบัณฑิตยสถาน, 2554) เป็นสัญลักษณ์ในการสร้างตัวแปรเซต ดังโปรแกรมตัวอย่างที่ 6.9

ตัวอย่างที่ 6.9 การสร้างตัวแปรแบบเซต

บรรทัดที่	คำสั่ง
1	st = {'a','e','i','o','u',10.5,'u','i'}
2	print (st)
3	print (type(st))
ผลลัพธ์ครั้งที่ 1	
{'i', 10.5, 'e', 'u', 'a', 'o'} <class 'set'>	
ผลลัพธ์ครั้งที่ 2	
{10.5, 'o', 'a', 'i', 'e', 'u'} <class 'set'>	

การสร้างตัวแปรเซตและการดำเนินการภายในเซต เซตจะจัดรูปแบบของข้อมูลภายในเซตโดยการตัดสมาชิกที่เหมือนกันออกไปและทำการจัดเรียงใหม่ ผลลัพธ์ที่ได้มีสมาชิกไม่ซ้ำกันและการแสดงผลลัพธ์ของข้อมูลแต่ละครั้ง จะมีการเรียงลำดับข้อมูลที่ไม่เหมือนกัน ดังนั้น สมาชิกของเซตไม่มีตัวเลขอ้างอิงตำแหน่ง ดังผลลัพธ์ที่ได้โปรแกรมตัวอย่างที่ 6.9

ตัวอย่างที่ 6.10 การอ้างอิงตำแหน่งข้อมูลของเซต

บรรทัดที่	คำสั่ง
1	st = {'a','e','i','o','u',10.5,'u','i'}
2	print (st[1])
ผลลัพธ์	
Traceback (most recent call last): File "C:\Python36\test.py", line 2, in <module> print (st[1]) TypeError: 'set' object does not support indexing	

โปรแกรมตัวอย่างที่ 6.10 เมื่อทำการระบุตำแหน่งเพื่อแสดงข้อมูลของสมาชิกภายในเซต ไม่สามารถใช้คำสั่งอ้างอิงไปยังข้อมูลนี้ได้ เนื่องจากสมาชิกในเซตนั้นไม่มีตำแหน่งในการจัดเรียงข้อมูล

การแปลงตัวแปรลีสต์และทัพเพิลเป็นตัวแปรเซต คือ ความสามารถในการนำข้อมูลในตัวแปรของลีสต์และทัพเพิลมาใช้งานในรูปแบบตัวแปรเซต การดำเนินการจะคัดลอกสมาชิกในลีสต์หรือทัพเพิลมายังตัวแปรเซต โดยใช้คำสั่ง set() ดังโปรแกรมตัวอย่างที่ 6.11

ตัวอย่างที่ 6.11 การแปลงตัวแปรลีสต์และทัพเพิลเป็นตัวแปรเซต

บรรทัดที่	คำสั่ง
1	lst = ['a','e','i','o','u',10.5,'u','i']
2	print (lst,type(lst))
3	tpl = ('dog','cat','bird','dog',200)
4	print (tpl,type(tpl))
5	st1 = set(lst)
6	st2 = set(tpl)
7	print(st1,type(st1))
8	print(st2,type(st2))
ผลลัพธ์	
['a', 'e', 'i', 'o', 'u', 10.5, 'u', 'i'] <class 'list'> ('dog', 'cat', 'bird', 'dog', 200) <class 'tuple'> {10.5, 'i', 'e', 'a', 'u', 'o'} <class 'set'> {200, 'dog', 'bird', 'cat'} <class 'set'>	

การตรวจสอบการเป็นสมาชิกภายในเซต คือ การตรวจสอบข้อมูลที่ของตัวแปร เปรียบเทียบข้อมูลที่อยู่ในเซตมีหรือไม่ การตรวจสอบจะใช้คำสั่ง in ผลลัพธ์ที่ได้จะมีค่าความเป็นจริงและเท็จ รูปแบบการใช้คำสั่ง ดังโปรแกรมตัวอย่างที่ 6.12

ตัวอย่างที่ 6.12 การตรวจสอบการเป็นสมาชิกของเซต

บรรทัดที่	คำสั่ง
1	st1=set('python 3.6')
2	st2={'dog','cat','bird','dog',200}
3	print (st1)
4	print (st2)
5	x='t' in st1
6	y='3.6' in st1
7	z='dog' in st2
8	print(x)
9	print(y)
10	print(z)
ผลลัพธ์	
{'p', '3', ' ', 't', '6', 'h', 'y', '.', 'o', 'n'} {'bird', 'cat', 'dog', 200} True False True	

โปรแกรมตัวอย่างที่ 6.12 มีการแปลงจากลีสต์เป็นตัวแปรเซต ข้อมูลที่อยู่ในลีสต์เป็นข้อความ แต่เมื่อได้ทำการแปลงข้อมูลเป็นเซต สมาชิกของข้อมูลจะถูกแยกออกเป็นแต่ละตัวอักษร รูปแบบของเซต เมื่อทำการตรวจสอบข้อมูล ‘t’ จากเซตด้วยตัวแปร x จึงได้ผลลัพธ์เป็นจริง เนื่องจากเมื่อเมื่อแยกข้อมูลเป็นเซต จึงมีข้อมูลอยู่ในเซต ส่วนตัวแปร y ทำการตรวจสอบจึงได้ผลลัพธ์เป็นเท็จและตรวจสอบข้อมูลจากตัวแปร y ได้ผลลัพธ์เป็นจริง เนื่องจากข้อมูลเป็นสมาชิกของเซต ดังนั้นกระบวนการของการประกาศตัวแปร การตรวจสอบการเป็นสมาชิกของเซต ถือว่าเป็นรูปแบบการใช้งานพื้นฐานของเซต ก่อนที่จะนำไปใช้งานต่อไป

2. การดำเนินการเซตทางคณิตศาสตร์

การดำเนินการเซตทางคณิตศาสตร์ของภาษาไพธอน ได้เตรียมคำสั่งที่ใช้ในการดำเนินการประกอบด้วยการใช้สัญลักษณ์ ดังตารางที่ 6.3

ตารางที่ 6.3 การดำเนินการเซตทางด้านคณิตศาสตร์

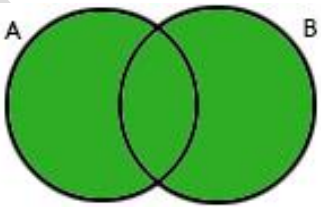
ลำดับที่	การดำเนินการ	สัญลักษณ์
1	ยูเนียน (Union)	|
2	อินเตอร์เซคชัน (Intersection)	&
3	ผลต่าง (Difference)	-
4	ผลต่างสมการ (Symmetric Difference)	^

ที่มา: สุชาติ คุ้มมณี (2558)

จากตารางที่ 6.3 การดำเนินการทางด้านเซตและสัญลักษณ์ที่ใช้ มีรายละเอียดของการทำงานต่อไปนี้

2.1 ยูเนียน

ยูเนียน (Union) หมายถึง ลักษณะของเซตที่ประกอบด้วยสมาชิกของเซต A และเซต B ใช้สัญลักษณ์ “|” ยูเนียนของเซต ดังภาพที่ 6.1



ภาพที่ 6.1 การยูเนียนของเซต

ที่มา : Stein et al. (2011)

จากความหมายของยูเนียนและสัญลักษณ์ที่ใช้ สามารถนำไปเขียนโปรแกรม โดยมีรายละเอียดและผลลัพธ์ที่ได้ โปรแกรมตัวอย่างที่ 6.13

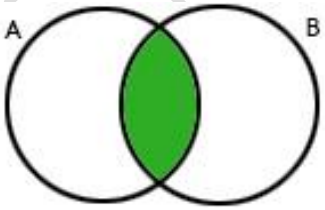
ตัวอย่างที่ 6.13 การยูเนียนของเซต

บรรทัดที่	คำสั่ง
1	st1={'a','b','c','d','e','f','g','h','i'}
2	st2={'a','e','i','o','u'}
3	st3 = st1 | st2
4	print (st3)
ผลลัพธ์	
{'i', 'a', 'e', 'b', 'o', 'u', 'd', 'c', 'f', 'h', 'g'}	

การยูเนียน จะทำการนำสมาชิกที่อยู่ในเซต st1 และเซต st2 มารวมกันและทำการตรวจสอบสมาชิกที่ซ้ำ โดยการตัดสมาชิกที่ซ้ำออกไปให้เหลือเพียงสมาชิกเดียว สมาชิกที่ได้ประกอบด้วยสมาชิกที่ไม่ได้จัดเรียง i a e b o u d c f h และ g ดังผลลัพธ์ของโปรแกรมตัวอย่างที่ 6.13

2.2 อินเตอร์เซคชัน

อินเตอร์เซคชัน (Intersection) หมายถึง ลักษณะของเซตที่ประกอบด้วยสมาชิกของเซต A หรือเซต B ใช้สัญลักษณ์ “&” อินเตอร์เซคชันของเซต ดังภาพที่ 6.2



ภาพที่ 6.2 การอินเตอร์เซคชันของเซต

ที่มา : Stein et al. (2011)

จากความหมายของอินเตอร์เซคชันและสัญลักษณ์ที่ใช้ สามารถนำไปใช้ในโปรแกรม โดยมีรายละเอียดและผลลัพธ์ ดังโปรแกรมตัวอย่างที่ 6.14

ตัวอย่างที่ 6.14 การอินเตอร์เซคชันของเซต

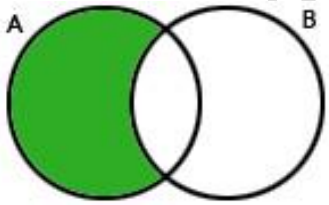
บรรทัดที่	คำสั่ง
1	st1={'a','b','c','d','e','f','g','h','i'}
2	st2={'a','e','i','o','u'}

3	st3 = st1 & st2
4	print (st3)
ผลลัพธ์	
{'i', 'a', 'e'}	

การอินเตอร์เซคชันของเซต จะทำการนำสมาชิกที่อยู่ในเซต st1 และเซต st2 มาทำการตรวจสอบ โดยเลือกเฉพาะสมาชิกที่มีอยู่ทั้งในเซต st2 และเซต st1 สมาชิกที่อยู่ทั้งสองเซตประกอบด้วย i a และ e ดังผลลัพธ์ของโปรแกรมตัวอย่างที่ 6.14

2.3 ผลต่าง

ผลต่าง (Difference) หมายถึง ลักษณะของเซตที่ประกอบด้วยสมาชิกของเซต A แต่ไม่เป็นสมาชิกของเซต B ใช้สัญลักษณ์ “-” ผลต่างของเซต ดังภาพที่ 6.3



ภาพที่ 6.3 ผลต่างของเซต

ที่มา : Stein et al. (2011)

จากความหมายของผลต่างและสัญลักษณ์ที่ใช้ สามารถนำไปเขียนโปรแกรม โดยมีรายละเอียดและผลลัพธ์ที่ได้ ดังโปรแกรมตัวอย่างที่ 6.15

ตัวอย่างที่ 6.15 การหาผลต่างของเซต

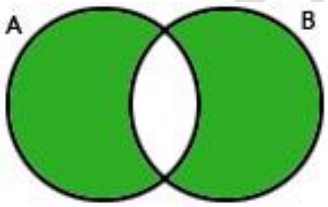
บรรทัดที่	คำสั่ง
1	st1={'a','b','c','d','e','f','g','h','i'}
2	st2={'a','e','i','o','u'}
3	st3 = st1 - st2
4	st4 = st2 - st1
5	print (st3)
6	print (st4)
7	print (st4-st3)

ผลลัพธ์
{'g', 'f', 'b', 'c', 'd', 'h'}
{'o', 'u'}
{'o', 'u'}

ผลต่างของเซต จะทำการตรวจสอบสมาชิกที่อยู่ในเซตที่หนึ่งและเซตที่สอง โดยตัดสมาชิกที่อยู่ในเซตที่หนึ่งที่มีสมาชิกซ้ำกับเซตที่สองออกและแสดงเฉพาะสมาชิกที่ไม่ซ้ำ ดังผลลัพธ์ที่ได้โปรแกรมตัวอย่างที่ 6.15

2.4 ผลต่างสมการ

ผลต่างสมการ (Symmetric Difference) หมายถึง ลักษณะของเซตที่ประกอบด้วยสมาชิกของเซต A หรือเซต B แต่ไม่ใช่สมาชิกที่อยู่ทั้งในเซต A และเซต B ใช้สัญลักษณ์ “^” ผลต่างสมการของเซต ดังภาพที่ 6.4



ภาพที่ 6.4 ผลต่างสมการ

ที่มา : Stein et al. (2011)

จากความหมายของผลต่างสมการและสัญลักษณ์ที่ใช้ สามารถนำไปเขียนโปรแกรม โดยมีรายละเอียดและผลลัพธ์ที่ได้ ดังโปรแกรมตัวอย่างที่ 6.16

ตัวอย่างที่ 6.16 ผลต่างสมการของเซต

บรรทัดที่	คำสั่ง
1	st1={'a','b','c','d','e','f','g','h','i'}
2	st2={'a','e','i','o','u'}
3	st3=set("programming language")
4	st4 = st1 ^ st2
5	st5 = st2 ^ st1
6	st6 = st2 ^ st3

7	print (st4)
8	print (st5)
9	print (st6)
ผลลัพธ์	
{'f', 'h', 'o', 'g', 'u', 'c', 'd', 'b'}	
{'h', 'c', 'o', 'u', 'g', 'd', 'f', 'b'}	
{'n', 'm', 'p', 'l', ' ', 'r', 'g'}	

ผลต่างสมการของเซต จะทำการตรวจสอบสมาชิกที่อยู่ในเซตที่ 1 และเซตที่ 2 ตรวจสอบสมาชิกภายในเซต สมาชิกที่มีเหมือนกันจะถูกตัดออก นำสมาชิกที่เหลือทั้งสองเซตที่ไม่ถูกตัด ทำการจัดเรียงข้อมูลใหม่ ดังผลลัพธ์ที่ได้ของโปรแกรมตัวอย่างที่ 6.16

3. ตัวดำเนินการที่ใช้ในการเปรียบเทียบเซต

ตัวดำเนินการทางคณิตศาสตร์ สามารถนำสัญลักษณ์หรือคำสั่ง ใช้ดำเนินการเปรียบเทียบระหว่างเซตได้ ผลลัพธ์ที่ได้จากการเปรียบเทียบอยู่รูปของความเป็นจริงและเท็จ โดยมีรูปแบบดังต่อไปนี้

3.1 สัญลักษณ์ดำเนินการเปรียบเทียบ

สัญลักษณ์ที่ใช้ดำเนินการในการเปรียบเทียบระหว่างเซต ใช้สัญลักษณ์พื้นฐาน โดยมีรายละเอียดการใช้สัญลักษณ์ดังโปรแกรมตัวอย่างที่ 6.17

ตัวอย่างที่ 6.17 การใช้สัญลักษณ์ในการเปรียบเทียบเซต

บรรทัดที่	คำสั่ง
1	st1={'a','b','c','d','e','f','g','h','i'}
2	st2={'a','e','i','o','u'}
3	st3={'1','2','3','4','5'}
4	print (len(st3)==len(st2))
5	print (len(st1)>len(st2))
6	print (len(st2)>=len(st3))
7	print (len(st1)<len(st2))
8	print (len(st3)<=len(st2))
9	print (len(st3)!=len(st2))

ผลลัพธ์
True
True
True
False
True
False

โปรแกรมตัวอย่างที่ 6.17 นับจำนวนของสมาชิกที่อยู่เซต st1 st2 และ st3 โดยคำสั่ง len() ทำการเปรียบเทียบจำนวนโดยใช้สัญลักษณ์เปรียบเทียบทางคณิตศาสตร์ ผลลัพธ์ที่ได้จากการเปรียบเทียบแต่ละเซตจะมีค่าความเป็นจริงหรือเท็จเท่านั้น

3.2 คำสั่งดำเนินการเปรียบเทียบ

การดำเนินการเปรียบเทียบ นอกเหนือจากการใช้สัญลักษณ์ สามารถใช้คำสั่งในการเปรียบเทียบได้ คำสั่งที่ใช้ในการเปรียบเทียบ คือ คำสั่ง in และคำสั่ง not in มีรูปแบบการใช้คำสั่งดังโปรแกรมตัวอย่างที่ 6.18

ตัวอย่างที่ 6.18 การใช้คำสั่งในการเปรียบเทียบเซต

บรรทัดที่	คำสั่ง
1	st1={'a','b','c','d','e','f','g','h','i'}
2	st2={'a','e','i','o','u'}
3	st3={'1','2','3','4','5'}
4	print ('d' in st1)
5	print ('g' in st2)
6	print ('3' not in st2)
7	print ('a' not in st3)
ผลลัพธ์	
True	
False	
True	
True	

คำสั่งที่ใช้ในการเปรียบเทียบเซต คำสั่ง in หมายถึง ข้อมูลนั้นเป็นสมาชิกที่อยู่ในเซต และคำสั่ง not in หมายถึง สมาชิกนั้นไม่ได้เป็นสมาชิกที่อยู่ในเซต ดังนั้นผลการเปรียบเทียบที่ได้ดังผลลัพธ์โปรแกรมตัวอย่างที่ 6.18

3.3 คำสั่งดำเนินการเปรียบเทียบมากกว่า 2 เงื่อนไข

การเปรียบเทียบเซตโดยใช้สัญลักษณ์และคำสั่ง คือ ลักษณะของการเปรียบเทียบแบบหนึ่งเงื่อนไข การเปรียบเทียบเพื่อหาผลลัพธ์อาจมีมากกว่าหนึ่งเงื่อนไข สามารถใช้คำสั่งอีกประเภทที่ใช้สำหรับการตรวจสอบเงื่อนไขมากกว่าหนึ่งเงื่อนไขขึ้นไป คำสั่งที่ใช้ คือ คำสั่ง and และ or มีรูปแบบการใช้คำสั่งดังต่อไปนี้

ตัวอย่างที่ 6.19 การใช้คำสั่งในการเปรียบเทียบมากกว่าหนึ่งเงื่อนไข

บรรทัดที่	คำสั่ง
1	st1={'a','b','c','d','e','f','g','h','i'}
2	st2={'a','e','i','o','u'}
3	st3={'1','2','3','4','5'}
4	print ('d' in st1 and 'i' not in st2)
5	print ('a' in st2 and 'a' in st3)
6	print ('e' not in st3 or i not in st1)
7	print ('u' not in st1 or 'f' in st2)
8	print (('1' not in st2 or '5' in st3)and '4' not in st1)
ผลลัพธ์	
False False True True True	

การใช้คำสั่งในการเปรียบเทียบมากกว่าหนึ่งเงื่อนไข จะใช้คำสั่ง and ซึ่งจะให้ผลลัพธ์เป็นจริงก็ต่อเมื่อ เงื่อนไขทุกเงื่อนไขในการเปรียบเทียบเป็นจริงทุกเงื่อนไข นอกเหนือจากนั้นจะได้ผลลัพธ์เป็นเท็จ และคำสั่ง or จะให้ผลลัพธ์ที่เป็นจริงก็ต่อเมื่อ มีหนึ่งเงื่อนไขที่เป็นจริงอย่างน้อยหนึ่งเงื่อนไข กรณีทุกเงื่อนไขเป็นเท็จ จะได้ผลลัพธ์เป็นเท็จ ดังผลลัพธ์ที่ได้ของโปรแกรม 6.19

ดิกชันนารี

ดิกชันนารี (Dictionary) หมายถึง ตัวแปรที่มีการเก็บข้อมูลในรูปแบบเดียวกันหรือต่างกันได้ ข้อมูลที่อยู่ภายใน สามารถเปลี่ยนแปลงได้ตลอดเวลาเช่นเดียวกับลีสต์ โครงสร้างของดิกชันนารี จะมีการใช้คีย์ (key) อ้างอิงกับข้อมูล เพื่อเป็นตัวชี้ตำแหน่งในการเรียกใช้ข้อมูลนั้น คีย์ที่อยู่ภายในดิกชันนารี จะต้องไม่ซ้ำกัน (สุชาติ คุ้มมณี, 2558) การเก็บข้อมูลประกอบไปด้วยรหัสและค่าข้อมูลรวมกัน การนำข้อมูลไปใช้ จะต้องระบุรหัสพร้อมกับค่าข้อมูล แทนการใช้หมายเลขลำดับเช่นเดียวกับลีสต์และทัพเพิล โครงสร้างข้อมูลเป็นลักษณะที่สามารถเปลี่ยนแปลงได้ เช่น เพิ่ม ลบ และแก้ไขสมาชิกที่อยู่ในตัวแปรดิกชันนารี (โชติพันธุ์ หล่อเลิศสุนทร และ ฐิตะพันธุ์ หล่อเลิศสุนทร, 2559)

1. การประกาศตัวแปรดิกชันนารี

การสร้างตัวแปรดิกชันนารี จะใช้สัญลักษณ์ “{ }” หรือวงเล็บปีกา การกำหนดรายละเอียดภายในดิกชันนารี ใช้สัญลักษณ์ “:” อ้างอิงระหว่างรหัสและข้อมูล ดังโปรแกรมตัวอย่างที่ 6.20

ตัวอย่างที่ 6.20 การประกาศตัวแปรดิกชันนารี

บรรทัดที่	คำสั่ง
1	dict1 = {'Name':'Ekarch', 'Surname': 'Thammasa', 'Tel': '0817603302'}
2	dict2 = { 'Code': 456 }
3	dict3 = { 'Id': 5692301, 1: 'Python Programming' }
4	print(dict1,type(dict1))
5	print(dict2,type(dict2))
6	print(dict3,type(dict3))
ผลลัพธ์	
{'Name': 'Ekarch', 'Surname': 'Thammasa', 'Tel': '0817603302'} <class 'dict'> {'Code': 456} <class 'dict'> {'Id': 123, 1: 'Python Programming'} <class 'dict'>	

การทำงานของตัวแปรดิกชันนารี จะมีสองส่วนที่เกี่ยวข้องกัน โดยส่วนหน้าของสัญลักษณ์ “:” หมายถึง รหัสที่ใช้อ้างไปยังข้อมูลสมาชิกในดิกชันนารี ตั้งชื่อรหัสให้สอดคล้องกับข้อมูลกับข้อมูลที่อ้างถึงและส่วนที่สองหลังสัญลักษณ์ “:” คือ ค่าของข้อมูลที่อ้างอิงของแต่ละรหัสในดิกชันนารี การประกาศตัวแปรดิกชันนารี มีรายละเอียดดังโปรแกรมตัวอย่างที่ 6.20

2. การเข้าถึงข้อมูลดิกชันนารี

การเข้าถึงข้อมูลที่อยู่ในดิกชันนารี คือ การอ้างอิงชื่อของดิกชันนารีเพื่ออ่านข้อมูลตามรหัสที่ได้อ้างอิง ผลลัพธ์ที่ได้ดังโปรแกรมตัวอย่างที่ 6.21

ตัวอย่างที่ 6.21 การเข้าถึงข้อมูลในดิกชันนารี

บรรทัดที่	คำสั่ง
1	dict1 = {'Name':'Ekarch', 'Surname': 'Thammasa', 'Tel': '0817603302'}
2	dict2 = { 'Code': 456 }
3	dict3 = { 'Id': 123, 1: 'Python Programming' }
4	print(dict1['Name'])
5	print(dict2['Code'])
ผลลัพธ์	
Ekarch 456	

3. ฟังก์ชันเกี่ยวกับดิกชันนารี

การปฏิบัติการในตัวแปรแบบดิกชันนารี มีฟังก์ชันที่ใช้ในการจัดการ เปลี่ยนแปลงรูปแบบของข้อมูลที่อยู่ในดิกชันนารี มีรายละเอียดและการทำงานของฟังก์ชัน ดังตารางที่ 6.4

ตารางที่ 6.4 ฟังก์ชันที่ใช้งานกับดิกชันนารี

ฟังก์ชัน	รูปแบบ	ความหมาย
values()	ดิกชันนารี.values()	การอ่านค่าของข้อมูลทั้งหมดที่อยู่ในตัวแปร
clear()	ดิกชันนารี.clear()	การลบข้อมูลทั้งหมดออกจากตัวแปร
copy()	ดิกชันนารี2=ดิกชันนารีที่1.copy()	การคัดลอกข้อมูลจากดิกชันนารีที่1 ไปยังดิกชันนารีที่2
update()	ดิกชันนารี.update()	การแก้ไขข้อมูลภายในดิกชันนารี
fromkeys()	ดิกชันนารี.fromkeys(ลิสต์)	การคัดลอกข้อมูลลิสต์มาสร้างเป็นคีย์ในดิกชันนารี
get()	ดิกชันนารี.get(คีย์)	การอ่านข้อมูลในดิกชันนารีโดยการอ้างอิงผ่านคีย์

ตารางที่ 6.4 ฟังก์ชันที่ใช้งานกับดิกชันนารี (ต่อ)

ฟังก์ชัน	รูปแบบ	ความหมาย
items()	ดิกชันนารี.items()	การอ่านค่าในดิกชันนารีและคืนค่าออกมาในรูปแบบทัพเพิลที่อยู่ภายใต้ลีสต์
keys()	ดิกชันนารี.keys()	แสดงคีย์ที่ใช้ในการอ้างอิงของข้อมูลของตัวแปรดิกชันนารี
setdefault()	ดิกชันนารี.setdefault()	การกำหนดค่าเริ่มต้นให้กับคีย์ที่เพิ่มเข้าไปในตัวแปรดิกชันนารี
pop()	ดิกชันนารี.pop(คีย์)	การดึงข้อมูลออกจากดิกชันนารี โดยการอ้างอิงผ่านคีย์
popitem()	ดิกชันนารี.pop()	การดึงข้อมูลออกจากดิกชันนารีทีละสมาชิก

ที่มา : Tagliaferri (2016)

การเข้าถึงข้อมูลภายในตัวแปรดิกชันนารี มีฟังก์ชันให้สามารถจัดการข้อมูลได้หลายรูปแบบ ซึ่งจากความหมายการทำงานของแต่ละฟังก์ชัน สามารถนำไปเขียนโปรแกรม โดยมีรายละเอียดการทำงานดังโปรแกรมตัวอย่างที่ 6.22

ตัวอย่างที่ 6.22 การใช้ฟังก์ชันเข้าถึงข้อมูลของตัวแปรดิกชันนารี

บรรทัดที่	คำสั่ง
1	print('----------ผลลัพธ์การใช้ฟังก์ชัน----------')
2	lst=("Python","Programming","Language")
3	course={'course_id':5692301,'name':'หลักการเขียนโปรแกรม'}
4	print("ฟังก์ชัน values=",course.values())
5	new_course=course.copy()
6	print("ฟังก์ชัน copy=",new_course.values())
7	course.update({'unit':'3 หน่วยกิต'})
8	print("ฟังก์ชัน update=",course.values())
9	print("ฟังก์ชัน get=",course.get('name'))
10	print("ฟังก์ชัน items=",new_course.items())
11	print("ฟังก์ชัน keys=",course.keys())

```
12  new_course.pop('name')
13  print("ฟังก์ชัน pop=",new_course)
14  new_course.setdefault('major', None)
15  print ("ฟังก์ชัน setdefault=",new_course)
16  print("ฟังก์ชัน popitem=",new_course.popitem())
17  dict = dict.fromkeys(lst)
18  print("ฟังก์ชัน fromkeys=",dict)
19  new_course.clear()
20  print("ฟังก์ชัน clear=",new_course)
```

ผลลัพธ์

```
----------ผลลัพธ์การใช้ฟังก์ชัน----------
ฟังก์ชัน values= dict_values([5692301, 'หลักการเขียนโปรแกรม'])
ฟังก์ชัน copy= dict_values([5692301, 'หลักการเขียนโปรแกรม'])
ฟังก์ชัน update= dict_values([5692301, 'หลักการเขียนโปรแกรม', '3 หน่วยกิต'])
ฟังก์ชัน get= หลักการเขียนโปรแกรม
ฟังก์ชัน items= dict_items([('course_id', 5692301), ('name', 'หลักการเขียนโปรแกรม')])
ฟังก์ชัน keys= dict_keys(['course_id', 'name', 'unit'])
ฟังก์ชัน pop= {'course_id': 5692301}
ฟังก์ชัน setdefault= {'course_id': 5692301, 'major': None}
ฟังก์ชัน popitem= ('major', None)
ฟังก์ชัน fromkeys= {'Python': None, 'Programming': None, 'Language': None}
ฟังก์ชัน clear= {}
```

การเก็บข้อมูลแบบแถวลำดับดิกชันนารี มีฟังก์ชันที่ใช้ในการจัดการสมาชิกหรือข้อมูล เพื่อให้การนำไปใช้งานในการดำเนินการ มีความสะดวกรวดเร็วและมีประสิทธิภาพเพิ่มมากขึ้น คุณสมบัติของฟังก์ชันที่เกี่ยวข้องกับดิกชันนารี มีผลลัพธ์ที่ได้ดังโปรแกรมตัวอย่างที่ 6.22

สรุป

ทัพเพิล เซตและดิกชันนารี มีโครงสร้างการเก็บข้อมูลแบบแถวลำดับเช่นเดียวกันกับลีสต์ แต่มีรูปแบบของการเก็บข้อมูลและจัดการข้อมูลภายในที่แตกต่างกัน ตัวแปรทัพเพิล โครงสร้างข้อมูลภายในไม่สามารถเปลี่ยนรูปได้ เช่น เพิ่ม ลบ แก้ไข หรือจัดเรียงข้อมูล แต่ลีสต์สามารถจัดการข้อมูล

ดังกล่าวได้ ตัวแปรเซต มีโครงสร้างการเก็บข้อมูลที่ไม่มีตำแหน่งของข้อมูลหรือไม่มีเลขลำดับของข้อมูลเหมือนกับลิสต์ แต่มีหลักการเมื่อนำไปใช้ จะมีการตรวจสอบของสมาชิกที่จะต้องไม่ซ้ำกัน โดยตัดข้อมูลที่ซ้ำออก การจัดเรียงแต่ละครั้ง ข้อมูลภายในเซตจะถูกจัดเรียงไม่เหมือนกัน การดำเนินการกับสมาชิกที่อยู่ภายในเซตใช้หลักการทางคณิตศาสตร์ เช่น ยูเนียน อินเตอร์เซคชัน ผลต่างและผลต่างสมการ ตัวแปรดิกชันนารี มีรูปแบบโครงสร้างการเก็บข้อมูลสองส่วนประกอบด้วย คีย์ในการอ้างอิงและข้อมูล ดังนั้น เมื่อต้องการเรียกข้อมูลใด จะต้องเรียกผ่านคีย์ที่กำหนดอ้างอิงกับข้อมูล โดยข้อมูลภายในดิกชันนารี สามารถเปลี่ยนแปลงได้ การดำเนินการสามารถใช้ฟังก์ชันในการจัดการข้อมูลภายในดิกชันนารีได้ เพื่อความสะดวกและรวดเร็วในการทำงาน จากลักษณะโครงสร้างการเก็บข้อมูลทั้ง 3 ประเภท ที่มีรูปแบบ ข้อดีและข้อเสียแตกต่างกัน ดังนั้น การนำไปใช้ในการเขียนโปรแกรม ควรจะต้องพิจารณาถึงคุณสมบัติเหล่านี้ไปใช้งานในโปรแกรม เพื่อให้โปรแกรมมีประสิทธิภาพในการทำงาน เนื่องจากโครงสร้างการเก็บข้อมูลมีความสำคัญต่อการทำงานของโปรแกรมคอมพิวเตอร์

แบบฝึกหัด

1. ให้ผู้เรียนอธิบายหลักการทำงานและรูปแบบการเก็บข้อมูลของตัวแปรทัพเพิล
2. ให้ผู้เรียนอธิบายหลักการทำงานและรูปแบบการเก็บข้อมูลของตัวแปรเซต
3. ให้ผู้เรียนอธิบายหลักการทำงานและรูปแบบการเก็บข้อมูลของตัวแปรดิกชันนารี
4. ให้ผู้เรียนอธิบายความแตกต่างและยกตัวอย่างประกอบ ระหว่างตัวแปรทัพเพิลกับลีสต์
5. ให้ผู้เรียนอธิบายความแตกต่างระหว่างตัวแปรเซตกับลีสต์ พร้อมทั้งยกตัวอย่างประกอบ
6. ให้ผู้เรียนอธิบายความแตกต่าง ยกตัวอย่างประกอบ ระหว่างตัวแปรดิกชันนารีกับลีสต์
7. ให้ผู้เรียนอธิบายฟังก์ชันที่ใช้ในการจัดการข้อมูลในตัวแปรดิกชันนารีมีอะไรบ้าง พร้อมเขียนโปรแกรมตัวอย่างอธิบายการทำงานของแต่ละฟังก์ชัน
8. กำหนดให้ตัวแปรลีสต์ lst1=("python language") และ lst2=("programming") ให้ผู้เรียนเขียนโปรแกรมให้มีข้อมูลในตัวแปรเซตดังต่อไปนี้

   {'p', 'o', 'n', 'g', 'a'}

   {'h', 'r', 'i', 'e', 'm', 'y', ' ', 'u', 't', 'l'}

   {'h', 'e', ' ', 'y', 'u', 't', 'l'}

   {'h', 'p', 'r', 'o', 'i', 'e', 'n', 'g', 'a', 'm', 'y', ' ', 'u', 't', 'l'}
9. กำหนดให้ตัวแปรดิกชันนารี sport={'sport_id':'ฟุตบอลชาย','time':90} ให้ผู้เรียนเขียนโปรแกรม โดยใช้ฟังก์ชันให้ได้ผลลัพธ์ดังต่อไปนี้

   {'sport_id': 'ฟุตบอลชาย', 'time': 90}

   {'sport_id': 'ฟุตบอลชาย', 'time': 90, 'number': '11'}

   {'sport_id': 'ฟุตบอลชาย', 'number': '11'}

   dict_keys(['sport_id', 'number'])
10. ให้ผู้เรียนเขียนโปรแกรมประยุกต์ใช้ตัวแปรดิกชันนารี โดยให้รับข้อมูลน้ำหนักและส่วนสูงเก็บไว้ในตัวแปรเซต นำข้อมูลที่ได้หาค่าดัชนีมวลกาย

เอกสารอ้างอิง

โชติพันธุ์ หล่อเลิศสุนทร และ ฐิตะพันธุ์ หล่อเลิศสุนทร. (2559). **คู่มือเรียนเขียนโปรแกรม python (ภาคปฏิบัติ)**. กรุงเทพฯ: คอร์ฟังก์ชั่น.

ราชบัณฑิตยสถาน. (2564). **พจนานุกรมฉบับราชบัณฑิตยสถาน พ.ศ. 2554.** พิมพ์ครั้งที่ 2. กรุงเทพฯ: ราชบัณฑิตยสถาน.

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

Elkner, Jeffrey. (2014). **Practical Programming (in Python)**. 3rded. London: Pearson.

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

Stein, Cliff L., Drysdale, Robert and Bogart, Kenneth. (2011). **Discrete Mathematics for Computer Scientists**. London: Pearson.

Tagliaferri, Lisa. (2016). **Understanding Dictionaries in Python 3**. (online) (cited 20 August 2016). Available from: https://www.digitalocean.com/community/tutorials/understanding-dictionaries-in-python-3