**สรุปเนื้อหาเรื่อง Array**

อาเรย์(array) คือการเก็บข้อมูลเป็นชุดของข้อมูลที่มีลำดับของข้อมูล โดยจะประกอบด้วยข้อมูลหลายๆข้อมูลและมีชนิดของข้อมูลแบบเดียวกัน เช่น อาเรย์ของเลขจำนวนเต็ม(int), อาเรย์ของเลขทศนิยม(float), อาเรย์ของข้อความ(string) ซึ่งโดยปกติจะมีความแตกต่างจากชุดข้อมูลชนิดลิสต์ที่สามารถเก็บข้อมูลได้หลายชนิดภายในลิสต์เดียวกัน ใน python นั้นโดยปกติแล้วจะไม่มีชุดข้อมูลชนิดอาเรย์(array)มาให้ แต่หากต้องการใช้งานในรูปแบบอาเรย์ก็จะใช้การเก็บชุดข้อมูลชนิดลิสต์(list) แต่จะทำการเก็บข้อมูลเพียงชนิดเดียวภายในลิสต์ดังกล่าวแทน เพื่อนำไปใช้งานในรูปแบบของอาเรย์(array) ในกรณีที่ต้องใช้งานรูปแบบการเก็บข้อมูลชนิดอาเรย์โดยเฉพาะ ก็จำเป็นต้องใช้ไลบรารี่(library)เพิ่มเติม เช่น ชุดข้อมูลชนิดอาเรย์จากไลบรารี่ numpy แทน ซึ่งหากต้องการดำเนินการทางคณิตศาสตร์เฉพาะเจาะจงกับชุดข้อมูลชนิดอาเรย์นั้น การเลือกใช้งานอาเรย์จากไลบรารี่ numpy แทนการใช้การเก็บข้อมูลในรูปแบบลิสต์จะทำให้มีประสิทธิภาพมากกว่า

การเลือกใช้ชุดข้อมูลชนิดลิสต์(List) มาแทนเพื่อใช้งานในรูปแบบอาเรย์(Array)โดยตรงนั้นจะมีข้อแตกต่างจากการใช้อาเรย์(Array) ปกติ เพราะชุดข้อมูลชนิดลิสต์(List) สามารถจะเก็บข้อมูลต่างชนิดกันได้ โดยเราสามารถจะเก็บได้ทุกข้อมูลเช่น เลขจำนวนเต็ม(integer), เลขทศนิยม(float) และ ข้อความ(string) อยู่ภายในลิสต์เดียวกัน ดังนั้นรูปแบบของการเก็บข้อมูลนั้นจะต้องซับซ้อนขึ้นเพื่อรองรับการทำงานในลักษณะดังกล่าว ทำให้ประสิทธิภาพในการประมวลผลในบางกรณีอาจจะลดลงมากจนสังเกตเห็นได้ชัดเจน

1.**การสร้างชุดข้อมูลอาเรย์(array)**  
เราสามารถสร้างชุดข้อมูลอาเรย์โดยใช้รูปแบบเดียวกันกับการสร้างชุดข้อมูลชนิดลิสต์ โดยนำข้อมูลชนิดเดียวกันมาแยกแต่ละข้อมูลด้วยเครื่องหมาย “,” และข้อมูลทั้งหมดอยู่ภายในขอบเขตเครื่องหมาย “[]”

**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

# [80, 85, 75, 81, 90]

2.**การเข้าถึงข้อมูลแต่ละตัวในอาเรย์(array)**  
เราสามารถจะเข้าถึงข้อมูลแต่ละตัวในชุดข้อมูลอาเรย์โดยอ้างถึงลำดับของข้อมูล แบบเดียวกันกับชุดข้อมูลชนิดลิสต์ ซึ่งทำเราจะทำการระบุลำดับของข้อมูลที่ต้องการจะเก็บไว้ภายในเครื่องหมาย “[]”

**การเข้าถึงข้อมูลแต่ละตัวในอาเรย์(array)ด้วยลำดับข้อมูลซึ่งเป็นจำนวนเต็มบวก**  
ลำดับข้อมูลเป็นตำแหน่งที่ข้อมูลถูกจัดให้อยู่ในชุดข้อมูลของอาเรย์ โดยใน python ลำดับข้อมูลนั้นจะเริ่มต้นที่ลำดับที่ 0  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน จากลำดับของข้อมูลที่ระบุ

print( scores[0] )

# โปรแกรมทำการปรินต์ค่า

# 80

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน จากลำดับของข้อมูลที่ระบุ

print( scores[3] )

# โปรแกรมทำการปรินต์ค่า

# 81

**การเข้าถึงข้อมูลแต่ละตัวในอาเรย์(array)ด้วยลำดับข้อมูล ซึ่งเป็นจำนวนเต็มลบ**

**ลำดับของข้อมูลชนิดอาเรย์เหมือนกับข้อมูลชนิดลิสต์ เราพูดถึงลำดับข้อมูลจำนวนเต็มลบคือ ถ้าเราพูดถึงข้อมูลที่ลำดับ -1 หมายถึงข้อมูลตัวหลังสุดของข้อมูลในอาเรย์**

**ตัวอย่างการใช้งาน**

**# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์**

**# เพื่อทำการเก็บคะแนนสอบของนักเรียน**

**scores = [80, 85, 75, 81, 90]**

**# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน จากลำดับของข้อมูลที่ระบุ**

**print( scores[-1] )**

**# โปรแกรมทำการปรินต์ค่า**

**# 90**

**# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน จากลำดับของข้อมูลที่ระบุ**

**print( scores[-2] )**

**# โปรแกรมทำการปรินต์ค่า**

**# 81**

3.**การตรวจสอบจำนวนข้อมูลในอาเรย์(array)**

เราสามารถทำการตรวจสอบจำนวนของข้อมูลที่อยู่ในชุดข้อมูลอาเรย์ ซึ่งรูปแบบการหาจำนวนข้อมูลจะเหมือนกันการหาในข้อมูลชนิดลิสต์ โดยการใช้งานฟังก์ชัน **len()**  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการปรินต์ค่าข้อมูล จำนวนของข้อมูลคะแนนสอบของนักเรียนที่ถูกเก็บในชุดข้อมูลอาเรย์

print( len(scores) )

# โปรแกรมทำการปรินต์ค่า

# 5

4.**การเพิ่มข้อมูลลงไปในชุดข้อมูลอาเรย์(array)**  
เราจะใช้ฟังก์ชัน **append()** ในการเพิ่มข้อมูลใหม่ลงไปในชุดข้อมูลอาเรย์ ซึ่งเมื่อเราเพิ่มจำนวนเข้าไป ข้อมูลตัวนั้นจะไปอยู่ตัวสุดท้ายของข้อมูล  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการเพิ่มข้อมูลลงไปในชุดข้อมูลอาเรย์

scores.append(65)

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

# [80, 85, 75, 81, 90, 65]

5.**การลบข้อมูลออกจากชุดข้อมูลอาเรย์(array)**

เราสามารถลบข้อมูลออกจากชุดข้อมูลอาเรย์โดยใช้คีย์เวิร์ด del หรืออาจจะลบข้อมูลโดยใช้ฟังก์ชัน remove() และ pop() ในการลบข้อมูลออก ซึ่งการใช้งานจะเหมือนกับข้อมูลชนิดลิสต์

**การลบข้อมูลออกจากชุดข้อมูลอาเรย์(array) โดยใช้คีย์เวิร์ด del**  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการลบข้อมูลคะแนนสอบของนักเรียนในลำดับที่ระบุ ด้วยคีย์เวิร์ด del

del scores[4]

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

# [80, 85, 75, 81]

6.**การเปลี่ยนแปลงข้อมูลภายในชุดข้อมูลอาเรย์(array)**  
เราสามารถทำการเปลี่ยนแปลงข้อมูลภายในชุดข้อมูลอาเรย์ โดยลำดับของข้อมูลที่เราต้องการเปลี่ยนแปลง นั้นทำการหนดค่าใหม่ให้ข้อมูลที่ตำแหน่งดังกล่าว โดยใช้เครื่องหมาย “=” เหมือนกับชุดข้อมูลชนิดลิสต์  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการแก้ไขข้อมูลคะแนนสอบของนักเรียนในชุดข้อมูลอาเรย์ในตำแหน่งที่ระบุ

# ด้วยค่าของข้อมูลที่อยู่ด้านขวาของเครื่องหมาย '='

scores[2] = 78

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

# [80, 85, 78, 81, 90]

7.**การใช้เครื่องหมายดำเนินการเพื่อดำเนินการกับข้อมูลภายในชุดข้อมูลอาเรย์(array)**  
เราสามารถใช้เครื่องหมายดำเนินการเช่น “+”, “\*” เพื่อทำการดำเนินการกับข้อมูลในชุดข้อมูลอาเรย์ เหมือนกับการใช้เครื่องหมายดำเนินการกับชุดข้อมูลชนิดลิสต์ ดังนี้

**การใช้เครื่องหมายดำเนินการ ‘+’ เพื่อดำเนินการกับข้อมูลภายในชุดข้อมูลอาเรย์(array)**  
เครื่องหมาย “+” จะใช้ในการรวมข้อมูลจากอาเรย์ 2 ตัวให้เป็นอาเรย์ใหม่  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียนห้อง A

class\_a\_scores = [80, 85, 75, 81, 90]

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียนห้อง B

class\_b\_scores = [75, 80, 79, 85, 79]

# ทำการรวบรวมคะแนนสอบของนักเรียนห้อง A และห้อง B ซึ่งเก็บข้อมูลในชุดข้อมูลอาเรย์

# ด้วยเครื่องหมายดำเนินการ '+'

scores = class\_a\_scores + class\_b\_scores

# ทำการปรินต์ค่าข้อมูลคะแนนรวมจากการกลางภาคและปลายภาคของนักเรียนแต่ละคน

print( scores )

# โปรแกรมทำการปรินต์ค่า

# [80, 85, 75, 81, 90, 75, 80, 79, 85, 79]

**การใช้เครื่องหมายดำเนินการ ‘\*’ เพื่อดำเนินการกับข้อมูลภายในชุดข้อมูลอาเรย์(array)**  
เครื่องหมาย “\*” เพื่อจะคัดลอกข้อมูลเดิมซ้ำๆ หลายครั้ง

**ตัวอย่างการใช้งาน**

**# กำหนดตัวแปรสำหรับเก็บข้อมูลข้อความในรูปแบบชุดข้อมูลอาเรย์**

**messages = ["hello"]**

**# ทำการคัดลอกข้อมูลดังกล่าวซ้ำเป็นจำนวน 3 ข้อมูลด้วยเครื่องหมายดำเนินการ '\*'**

**messages = messages \* 3**

**# ทำการปรินต์ค่าชุดข้อมูลข้อความที่ได้ทำการคัดลอกเรียบร้อยแล้ว**

**print( messages )**

**# โปรแกรมทำการปรินต์ค่า**

**# ["hello", "hello", "hello"]**

**8.การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์(array) โดยระบุช่วงของข้อมูล**  
เราสามารถเข้าถึงข้อมูลในชุดข้อมูลอาเรย์โดยระบุเป็นช่วงของข้อมูลได้ โดยทำการระบุจุดเริ่มต้นและจุดสิ้นสุดของช่วงข้อมูล โดยใช้รูปแบบ [จุดเริ่มต้น:จุดสิ้นสุด] โดยจะดึงข้อมูลตั้งแต่ข้อมูลตัวที่ระบุในตำแหน่งเริ่ม ไปจนถึงข้อมูลก่อนตำแหน่งตำแหน่งสิ้นสุด(ไม่รวมข้อมูลในตำแหน่งสิ้นสุดที่ระบุ)เช่นเดียวกับการเข้าถึงข้อมูลโดยระบุช่วงจากชุดข้อมูลชนิดลิสต์

**การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์(array) โดยระบุช่วงของข้อมูล ทำการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุด**  
การเข้าถึงข้อมูลโดยการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุดนี้ จะดึงข้อมูลตั้งแต่ข้อมูลตัวแรกที่เราระบุเป็นจุดเริ่มต้น ไปจนถึงข้อมูลก่อนตำแหน่งที่ระบุในจุดสิ้นสุด(ไม่รวมข้อมูลในตำแหน่งที่ระบุในจุดสิ้นสุด)  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการปรินต์ค่าชุดข้อมูลคะแนนสอบของนักเรียนโดยทำการระบุช่วงของข้อมูล

# โดยทำการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุดของข้อมูล และใช้ลำดับอ้างอิงเป็นจำนวนเต็มบวก

print( scores[ 2:4 ] )

# โปรแกรมทำการปรินต์ค่า

# [75, 81]

# ทำการปรินต์ค่าชุดข้อมูลคะแนนสอบของนักเรียนโดยทำการระบุช่วงของข้อมูล

# โดยทำการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุดของข้อมูล และใช้ลำดับอ้างอิงเป็นจำนวนเต็มลบ

print( scores[ -3:-1 ] )

# โปรแกรมทำการปรินต์ค่า

# [75, 81]

**การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์(array)โดยระบุช่วงของข้อมูล ระบุเพียงจุดเริ่มต้น**  
การเข้าถึงข้อมูลโดยการระบุเพียงจุดเริ่มต้นนี้ จะดึงข้อมูลตั้งแต่ข้อมูลตัวที่ระบุในจุดเริ่มต้น ไปจนถึงข้อมูลตัวสุดท้ายของอาเรย์  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการปรินต์ค่าชุดข้อมูลคะแนนสอบของนักเรียนโดยทำการระบุช่วงของข้อมูล

# โดยทำการระบุเพียงจุดเริ่มต้นของข้อมูล

print( scores[ -2: ] )

# โปรแกรมทำการปรินต์ค่า

# [81, 90]

**การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์(array)โดยระบุช่วงของข้อมูล ระบุเพียงจุดสิ้นสุด**  
การเข้าถึงข้อมูลโดยการระบุเพียงจุดสิ้นสุด จะดึงข้อมูลตั้งแต่ข้อมูลตัวแรกของอาเรย์ไปจนถึงข้อมูลก่อนตำแหน่งที่ระบุในจุดสิ้นสุด(ไม่รวมข้อมูลในตำแหน่งที่ระบุในจุดสิ้นสุด)  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการปรินต์ค่าชุดข้อมูลคะแนนสอบของนักเรียนโดยทำการระบุช่วงของข้อมูล

# โดยทำการระบุเพียงจุดสิ้นสุดของข้อมูล

print( scores[ :2 ] )

# โปรแกรมทำการปรินต์ค่า

# [80, 85]

**การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์โดยระบุช่วงของข้อมูล ไม่ทำการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุด**  
การเข้าถึงข้อมูลโดยไม่ระบุทั้งจุดเริ่มต้นและจุดสิ้นสุด โดยจะดึงข้อมูลตั้งแต่ข้อมูลตัวแรกของอาเรย์ไปจนถึงข้อมูลตัวสุดท้ายของอาเรย์  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

scores = [80, 85, 75, 81, 90]

# ทำการปรินต์ค่าชุดข้อมูลคะแนนสอบของนักเรียนโดยทำการระบุช่วงของข้อมูล

# โดยไม่ทำการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุดของข้อมูล

print( scores[:] )

# โปรแกรมทำการปรินต์ค่า

# [80, 85, 75, 81, 90]

9.**ฟังก์ชันที่สามารถใช้งานของอาเรย์(เช่นเดียวกับชุดข้อมูลชนิดลิสต์)**

| **ฟังก์ชัน** | **รายละเอียด** |
| --- | --- |
| append() | เพิ่มข้อมูลเข้าไปต่อจากตำแหน่งสุดท้ายของชุดข้อมูล |
| extend() | ขยาย(extend)ข้อมูลทุกตัวในอาเรย์ไปไว้ยังอาเรย์อื่น |
| insert() | เพิ่มข้อมูลเข้าไปตรงตำแหน่งที่ระบุของชุดข้อมูล |
| remove() | ลบข้อมูลที่ระบุออกจากอาเรย์ |
| pop() | ลบข้อมูลและคืนค่าข้อมูลดังกล่าวจากตำแหน่งของข้อมูลที่ระบุในอาเรย์ |
| clear() | ลบข้อมูลทุกตัวออกจากอาเรย์ |
| index() | คืนค่าของตำแหน่งแรกที่พบข้อมูลที่ระบุ |
| count() | นับจำนวนของข้อมูลทั้งหมดจากตัวแปรที่ส่งเข้าไป |
| sort() | เรียงลำดับข้อมูลในชุดข้อมูล(ค่าตั้งต้นคือเรียงจากน้อยไปหามาก) |
| reverse() | สลับลำดับของการเรียงข้อมูลจากหน้าไปหลัง |
| copy() | คืนค่าลิสต์ใหม่ซึ่งคัดลอกมาจากข้อมูลในอาเรย์เดิม |

**10.การสร้างอาเรย์หลายมิติ(multidimensional arrays)**  
อาเรย์หลายมิติก็คือ การมีข้อมูลอาเรย์ซ้อนอยู่ภายในข้อมูลอาเรย์อีกชั้นหนึ่ง ซึ่งอาจจะมีอาเรย์ย่อยๆ เป็นส่วนประกอบภายใน

**การสร้างอาเรย์หลายมิติ**  
**ตัวอย่างการใช้งาน**

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์หลายมิติด้วยรูปแบบข้อมูลจาก

# ชุดข้อมูลชนิดลิสต์ซ้อนในชุดข้อมูลชนิดลิสต์อีกชั้นหนึ่ง เพื่อทำการเก็บข้อมูลพิกัดของตำแหน่ง

coordinates = [[110, 152], [80, 60], [111, 17] ]

# ทำการเข้าถึงข้อมูลในอาเรย์หลายมิติเพื่อเข้าถึงข้อมูลลำดับที่ 0 จากชุดข้อมูลอาเรย์

position1 = coordinates[0]

# ทำการปรินต์ค่าข้อมูลพิกัดของตำแหน่ง

print("Position1: {}".format(position1))

# โปรแกรมทำการปรินต์ค่า

# Position1: [110, 152]

# ทำการเข้าถึงข้อมูลในอาเรย์หลายมิติเพื่อเข้าถึงข้อมูลลำดับที่ 1 จากชุดข้อมูลอาเรย์

position2 = coordinates[1]

# ทำการปรินต์ค่าข้อมูลพิกัดของตำแหน่ง

print("Position2: {}".format(position2) )

# โปรแกรมทำการปรินต์ค่า

# Position2: [80, 60]

# ทำการเข้าถึงข้อมูลในอาเรย์หลายมิติเพื่อเข้าถึงข้อมูลลำดับที่ 2 จากชุดข้อมูลอาเรย์

position3 = coordinates[2]

# ทำการปรินต์ค่าข้อมูลพิกัดของตำแหน่ง

print("Position3: {}".format(position3))

# โปรแกรมทำการปรินต์ค่า

# Position3: [111, 17]

# ทำการเข้าถึงข้อมูลในอาเรย์หลายมิติเพื่อเข้าถึงข้อมูลลำดับย่อย จากชุดข้อมูลอาเรย์

position3\_x = coordinates[2][0]

position3\_y = coordinates[2][1]

# ทำการปรินต์ค่าข้อมูลพิกัดของตำแหน่ง

print("Position3: x={}, y={}".format (position3\_x, position3\_y) )

# โปรแกรมทำการปรินต์ค่า

# Position3: x=111, y=1

**สรุปเนื้อหาเรื่อง Function**

ฟังก์ชัน (Function) คือส่วนของโค้ดหรือโปรแกรมที่ทำงานเพื่อวัตถุประสงค์บางอย่าง ในภาษา Python เราสามารถสร้างฟังก์ชันของเราเองเพื่อให้ทำงานอย่างที่เราต้องการ ในการเขียนโปรแกรมนั้นเรามักจะแยกโค้ดที่มีการทำงานเหมือนๆ กันเป็นฟังก์ชันเอาไว้ และเรียกใช้ฟังก์ชันนั้นซ้ำๆ ซึ่งเป็นแนวคิดของการ reuse โค้ด นี่เป็นรูปแบบของการประกาศฟังก์ชันในภาษา Python

def function\_name(args...):

# statements

def function\_name(args...):

# statements

return value

ในรูปแบบของการประกาศฟังก์ชันในภาษา Python นั้นจะใช้คำสั่ง def และหลังจากนั้น function\_name เป็นชื่อของฟังก์ชัน และในวงเล็บ () เป็นการกำหนดพารามิเตอร์ของฟังก์ชัน พารามิเตอร์ของฟังก์ชันนั้นสามารถมีจำนวนเท่าไหร่ก็ได้หรือไม่มีก็ได้ และเช่นเดียวกับภาษาอื่นๆ ฟังก์ชันอาจจะมีหรือไม่มีการส่งค่ากลับ สำหรับฟังก์ชันที่ไม่มีการ return ค่ากลับนั้น เรามักจะเรียกว่า โพรซีเยอร์ (Procedure) ต่อไปมาดูตัวอย่างการประกาศและใช้งานฟังก์ชันในภาษา Python

def hello(name):

print('Hello %s' % name)

def count\_vowel(str):

vowel = 0

for c in str:

if c in ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u'):

vowel = vowel + 1

return vowel

def area(width, height):

c = width \* height

return c

ในตัวอย่าง เราได้สร้างฟังก์ชันจำนวน 3 ฟังก์ชัน ฟังก์ชันแรกมีชื่อว่า hello() เป็นฟังก์ชันสำหรับแสดงข้อความทักทายจากที่ชื่อส่งเข้ามา ฟังก์ชันนี้มีหนึ่งพารามิเตอร์คือ name สำหรับรับชื่อที่ส่งเข้ามาในฟังก์ชัน

ต่อมาฟังก์ชัน count\_vowel() เป็นฟังก์ชันสำหรับนับจำนวนสระใน String ฟังก์ชันนี้มีหนึ่ง String พารามิเตอร์ ในการทำงานของฟังก์ชันนั้นเราใช้คำสั่ง For loop ในการวนอ่านค่าทีละตัวอักษรเพื่อตรวจสอบว่าเป็นสระหรือไม่ด้วยคำสั่ง in และตัวแปร vowel นั้นใช้สำหรับนับจำนวนสระที่พบใน String ในตอนท้ายเราได้ส่งค่าของจำนวนสระที่นับได้กลับไปด้วยคำสั่ง return

และฟังกชันสุดท้ายคือฟังก์ชัน area() เป็นฟังก์ชันสำหรับหาพื้นที่ของรูปสี่เหลี่ยมด้านขนาน และฟังก์ชันมีพารามิเตอร์สองตัวสำหรับความกว้างและความยาวของสี่เหลี่ยม และฟังก์ชันทำการ return ผลลัพธ์ที่เป็นพื้นที่กลับไปด้วยคำสั่ง return

## การเรียกใช้งานฟังก์ชันในภาษา Python

หลังจากเราได้สร้างฟังก์ชันในตัวอย่างก่อนหน้าแล้ว ต่อไปเราจะมาเรียกใช้งานฟังก์ชันเหล่านั้น ในการเรียกใช้ฟังก์ชันนั้นเราจะใช้ชื่ของฟังก์ชันและส่งอาร์กิวเมนต์ให้สอดคล้องกับพารามิเตอร์ที่กำหนดไว้ในฟังก์ชัน ดังนั้นอาร์กิวเมนต์คือค่าที่ส่งเข้าไปในฟังก์ชันตอนใช้งาน ส่วนพารามิเตอร์นั้นคือตัวแปรทีกำหนดไว้ในฟังก์ชันเพื่อรับค่าจากอาร์กิวเมนต์ มาดูตัวอย่างการเรียกใช้งานฟังก์ชันในภาษา Python

def hello(name):

print('Hello %s' % name)

def count\_vowel(str):

vowel = 0

for c in str:

if c in ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u'):

vowel = vowel + 1

return vowel

def area(width, height):

c = width \* height

return c

# calling functions

hello('Danny')

hello('Mateo')

print('Vowel in string = %d' % count\_vowel('marcuscode.com'))

print('Vowel in string = %d' % count\_vowel('Python'))

print('Area = %d' % area(8, 4))

ในตัวอย่าง เป็นการเรียกใช้งานฟังก์ชันที่เราสร้างขึ้น เราได้เรียกใช้ฟังก์ชัน hello() และส่งอาร์กิวเมนต์ที่เป็น String เข้าไปยังฟังก์ชัน เราเรียกใช้ฟังก์ชันนี้สองครั้ง ซึ่งนี่เองเป็นการ reuse โค้ดในการเขียนโปรแกรม

หลังจากนั้นเราเรียกใช้ฟังก์ชัน count\_vowel() และฟังก์ชัน area() และส่งพารามิเตอร์ที่ถูกต้องไปยังฟังก์ชัน และเพราะว่าฟังก์ชันเหล่านี้มีการ return ค่ากลับ เราสามารถนำค่าเหล่านี้ไปใช้งานได้ต่อไป เราได้นำไปใช้กับฟังก์ชัน print() เพื่อจัดรูปแบบการแสดงผล

## Default Argument Values

ในภาษา Python เราสามารถสร้างฟังก์ชันโดยการกำหนด Default Argument ให้กับฟังก์ชันพารามิเตอร์ได้ Default Argument เป็นการการกำหนดค่าเริ่มต้นให้กับอาร์กิวเมนต์ที่ส่งเข้ามายังฟังก์ชัน นั่นทำให้เราสามารถเรียกใช้งานฟังก์ชันโดยการส่งอาร์กิวเมนต์น้อยกว่าจำนวนที่กำหนดไว้ในฟังก์ชันได้ ซึ่งอำนวยความสะดวกในการใช้งานมากขึ้น มาดูตัวอย่างการสร้างและใช้งานฟังก์ชันกับ Default Argument

def show\_info(name, salary = 84360, lang = "Python"):

print('Name: %s' % name)

print('Salary: %d' % salary)

print('Language: %s' % lang)

print()

ในตัวอย่าง เราได้สร้างฟังก์ชัน show\_info() สำหรับแสดงข้อมูลของโปรแกรมเมอร์ ข้อมูลที่จำเป็นต้องการจะแสดงนั้นมีชื่อ เงินเดือน และภาษาที่เขียน ในฟังก์ชันของเรานั้นมี 3 พารามิเตอร์ พารามิเตอร์แรก name นั้นเป็นพารามิเตอร์แบบปกติ และสองพารามิเตอร์นั้นเป็น Default Argument ซึ่งเรากำหนดค่าเริ่มต้นให้กับพารามิเตอร์โดยใช้เครื่องหมาย = ในการกำหนดพารามิเตอร์นั้น Default Argument ต้องอยู่หลังพารามิเตอร์แบบปกติเสมอ

# calling function

show\_info('Mateo')

show\_info('Mateo', 105000)

show\_info('Danny', 120000, 'Java')

ในการเรียกใช้งานฟังก์ชันนั้น เราต้องทำการส่งค่าอาร์กิวเมนต์สำหรับพารามิเตอร์แบบปกติเสมอ ส่วนพารามิเตอร์แบบ Default Argument นั้นเป็นทางเลือก ในตัวอย่าง คำสั่งเราเรียกใช้ฟังก์ชันโดยอาร์กิวเมนต์เพียงหนึ่งตัวเข้าไป ทำให้สองอาร์กิวเมนต์ที่เหลือที่เป็น Default Argument ใช้ค่าเริ่มต้นของมันแทน คือ 84360 สำหรับเงินเดือน และ "Python" สำหรับภาษาเขียนโปรแกรม

ต่อมาเราเรียกใช้ฟังก์ชันโดยการส่งสองอาร์กิวเมนต์เข้าไป ทำให้มีเพียงพารามิเตอร์สุดท้ายเท่านั้นที่ใช้ค่าเริ่มต้น และในคำสั่งสุดท้ายเป็นการส่งค่าครบจำนวนให้กับทุกอาร์กิวเมนต์

Name: Mateo

Salary: 84360

Language: Python

Name: Mateo

Salary: 105000

Language: Python

Name: Danny

Salary: 120000

Language: Java

นี่เป็นผลลัพธ์การทำงานของโปรแกรม ในการเรียกใช้งานฟังก์ชันกับ Default Argument

## Keyword Arguments

ในภาษา Python เราสามารถเรียกใช้งานฟังก์ชันในรูปแบบของ Keyword Argument โดยการใช้ชื่อของพารามิเตอร์สำหรับส่งอาร์กิวเมนต์ ในการใช้งานนั้น พารามิเตอร์ต้องมีการกำหนดในรูปแบบของ Default Argument ก่อน มาดูตัวอย่างการใช้งาน Keyword Arguments ในภาษา Python

def create\_button(id, color = '#ffffff', text = 'Button', size = 16):

print('Button ID: %d' % id)

print('Attributes:')

print('Color: %s' % color)

print('Text: %s' % text)

print('Size: %d px' % size)

print()

create\_button(10)

create\_button(11, color = '#4286f4', text = 'Sign up')

create\_button(id = 12, color = '#323f54', size = 24)

create\_button(color = '#1cb22b', text = 'Log in', size = 32, id = 13)

ในตัวอย่าง เราได้สร้างฟังก์ชันสำหรับการสร้างปุ่ม ในการเรียกใช้งานฟังก์ชันนั้น เราสามารถเรียกโดยวิธีการส่งแบบ Keyword Argument ได้ในรูปแบบของ argument = value และสามารถสลับตำแหน่งของอาร์กิวเมนต์ได้ และในฟังก์ชันนั้นเรามีอาร์กิวเมนต์ id แบบซึ่งเป็นอาร์กิวเมนต์แบบปกติ ในการส่งค่านั้นต้องส่งเป็นลำดับแรกเสมอ เหมือนในคำสั่งการเรียกใช้งานสองอันแรก หรือสามารถส่งแบบ Keyword Argument ก็ได้เช่นกันเหมือนในคำสั่งที่สามและสี่

Button ID: 10

Attributes:

Color: #ffffff

Text: Button

Size: 16 px

Button ID: 11

Attributes:

Color: #4286f4

Text: Sign up

Size: 16 px

Button ID: 12

Attributes:

Color: #323f54

Text: Button

Size: 24 px

Button ID: 13

Attributes:

Color: #1cb22b

Text: Log in

Size: 32 px

นี่เป็นผลลัพธ์การทำงานของโปรแกรม เราได้เรียกใช้งานฟังก์ชันเพื่อสร้างปุ่ม 4 ปุ่มในรูปแบบต่างๆ ของการใช้ Keyword Argument

## Lambda Expressions

Lambda Expressions คือ anonymous function ที่เป็นฟังก์ชันที่มีการทำงานขนาดเล็กอยู่ภายในที่สามารถมีได้เพียง Expression เดียวเท่านั้น เราสามารถสร้างโดยใช้คำสั่ง lambda เราสามารถใช้ Lambda Expressions สร้างออบเจ็คของฟังก์ชันได้ และค่า return จะเป็นค่าที่ได้จากผลลัพธ์ของ Expression ของฟังก์ชัน มาดูตัวอย่างการใช้งาน

f = lambda x: x + 1

print(f(2))

print(f(8))

g = lambda a, b: (a + b) / 2

print(g(3, 5))

print(g(10, 33))

def make\_incrementor(n):

return lambda x: x + n

f = make\_incrementor(13)

print(f(0))

print(f(1))

print(f(5))

ในตัวอย่าง เราได้สร้าง Lambda Expressions เป็นจำนวนสามฟังก์ชัน ฟังก์ชันแรกเป็นฟังก์ชันสำหรับเพิ่มตัวเลขขึ้น 1 และฟังก์ชันที่สองเป็นฟังก์ชันสำหรับหาค่าเฉลี่ยของตัวเลขสองจำนวน คุณจะสังเกตุได้ว่าฟังก์ชันแรกนั้นมี 1 อาร์กิวเมนต์และฟังก์ชันที่สองนั้นมี 2 อาร์กิวเมนต์ และฟังก์ชันสุดท้ายนั้นเป็นการ return ฟังก์ชันกลับภายในฟังก์ชันอีกที และเป็นฟังก์ชันสำหรับเพิ่มตัวเลขขึ้นจำนวน n จากอาร์กิวเมนต์ที่ใส่เข้าไป

3

9

4.0

21.5

13

14

18

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

นอกจากนี้ Lambda Expressions ยังมีประโยชน์เพื่อใช้งานกับ built-in function เช่น ฟังก์ชัน filter() และฟังก์ชัน map() ในภาษา Python มันใช้เป็นอาร์กิวเมนต์ส่งเข้าไปในฟังก์ชัน เพื่อสร้าง Expression ให้กับฟังก์ชัน มาดูตัวอย่างการใช้งาน

numbers = [2, 15, 5, 7, 10, 3, 28, 30]

print(list(filter(lambda x: x % 5 == 0, numbers)))

print(list(map(lambda x: x \* 2, numbers)))

ในตัวอย่าง เรามีลิสต์ของตัวเลข Integer และเราได้ใช้ฟังก์ชัน filter() และฟังก์ชัน map() ซึ่งเป็นฟังก์ชันที่มีอาร์กิวเมนต์ตัวแรกเป็นฟังก์ชัน และตัวที่สองเป็นลิสต์ ในการทำงานของฟังก์ชัน filter() เราได้ใช้ฟังก์ชันกรองเอาตัวเลขที่ตรงกันกับ Lambda Expressions ซึ่งก็คือตัวเลขในลิสต์ที่หารด้วย 5 ลงตัว และในการใช้ฟังก์ชัน map() เป็นการเชื่อมโยงค่าในลิสต์ให้ตรงกับ Lambda Expressions คือการเพิ่มตัวเลขให้เป็นสองเท่า ซึ่งทั้งสองฟังก์ชันนี้ส่งค่ากลับเป็นออบเจ็ค และเราใช้ฟังก์ชัน list() เพื่อแปลงออบเจ็คให้เป็นสิสต์

[15, 5, 10, 30]

[4, 30, 10, 14, 20, 6, 56, 60]

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

#### **สรุป**

ฟังก์ชันเป็นส่วนหนึ่งที่มีความสำคัญในการพัฒนาโปรแกรม ซึ่งในเนื้อหาตอนนี้ได้แนะนำวิธีการสร้างฟังก์ชันขึ้นมาใช้งานแบบต่างๆ เช่น การสร้างฟังก์ชั่นที่ไม่มีการส่งค่า การสร้างฟังก์ชั่นที่ไม่มีการคืนค่า การสร้างฟังก์ชั่นที่มีการรรับค่าแต่ไม่มีการส่งค่ากลับ เป็นต้น นอกจากนี้ยังได้รู้จักกับวิธีการส่งค่าอาร์กิวเมนต์ให้กับค่าพารามิเตอร์ และยังได้รู้จักวิธีการสร้างค่าพารามิเตอร์ไว้รอรับค่าอาร์กิวเมนต์ รวมไปถึงสร้างฟังก์ชันที่ไม่ต้องกำหนดชื่อ ด้วย lambda ทำให้ลดเวลาในการเขียนคำสั่งโปรแกรม ซึ่งที่กล่าวมาทั้งหมดนี้ ไม่ต้องใช้คำสั่ง import เลย