**ฟังก์ชัน(Functions)**

ฟังก์ชัน หรือ โปรแกรมย่อย มีความสำคัญเนื่องจากเราเขียนโปรแกรมได้สักระยะ จนเกิดเทคนิคหรือทราบวิธีการที่ง่ายขึ้นของการเขียนโปรแกรม เห็นว่าโปรแกรมของเราจะมีจำนวนบรรทัดมากถ้าเป็นการคำนวณสูตร ใช้หลายครั้งก็อาจจะเกิดข้อผิดพลาดเวลาพิมพ์โปรแกรมผิด ดังนั้นเราจึงสร้างโปรแกรมย่อยหรือฟังก์ชันเพื่อเก็บโปรแกรมอีกชนิดหนึ่ง เพื่อง่ายและมีการเกิดข้อผิดพลาดน้อยที่สุด

**โดยฟังก์ชันแบ่งออกเป็น 2 ประเภท**

1. ฟังก์ชันที่ผู้เขียนโปรแกรมสร้างขึ้นเอง (User Defined Functions)

2.ฟังก์ชันที่สร้างจากผู้อื่นถูกเก็บไว้ในไลบรารี (Library) มีทั้งนำมาใช้ได้ฟรี (Open Source) และมีค่าใช้จ่าย (Commercial)

**หลักการสร้างฟังก์ชันขึ้นเองและหลักการทำงานแบ่งออกเป็น 4 ประเภท**

1) การสร้างฟังก์ชันที่ไม่มีการส่งค่าและไม่มีการรับค่า

**def** area\_Triangle() :  
 area = 0.5\*5\*6  
 print(**"Area ="**,area,)  
area\_Triangle ()

- ไม่มีการส่งค่ากลับ คือ ไม่มีการ **return** ตัวแปร

- ไม่มีการรับค่า คือ **def** area\_Triangle() : ไม่มีตัวแปรอยู่ในวงเล็บหลังชื่อฟังก์ชัน

2) การสร้างฟังก์ชันที่ไม่มีการส่งค่าแต่มีการรับค่า

**def** area\_Triangle(High,Base) :  
 area = 0.5\*High\*Base  
 print(**"Area ="**,area,)  
area\_Triangle (5,6)

- ไม่มีการส่งค่ากลับ คือ ไม่มีการ **return** ตัวแปร

- มีการรับค่า คือ มีตัวแปร 2 ตัวแปร อยู่ในวงเว็บหลังฟังก์ชัน ดังนี้ **def** area\_Triangle(High,Base) :

โดยจะรับค่าจากฟังก์ชัน print(**"Area ="**, area\_Triangle(5,6) )

3) การสร้างฟังก์ชันที่มีการส่งค่าแต่ไม่มีการรับค่า

**def** area\_Triangle() :  
 area = 0.5\*5\*6  
 **return** area   
print(**"Area ="**, area\_Triangle())

- มีการส่งค่า คือ ข้อมูลที่มีการคำนวณแล้วจะเก็บในตัวแปล **area** แล้วส่งค่ากลับ (**return** area) ไปยังฟังก์ชัน area\_Triangle()

- ไม่มีการรับค่า คือ ไม่มีการใส่ตัวแปรในวงเล็บหลังชื่อฟังก์ชัน area\_Triangle()

4) การสร้างฟังก์ชันที่มีการส่งค่ากลับและมีการรับค่า

**def** area\_Triangle(High,Base) :  
 area =0.5\*High\*Base  
 **return** area  
print(**"Area ="**, area\_Triangle(5,6) )

- มีการส่งค่า คือ ข้อมูลที่มีการคำนวณแล้วจะเก็บในตัวแปล **area** แล้วส่งค่ากลับ (**return** area) ไปยังฟังก์ชัน area\_Triangle()

- มีการรับค่า คือ มีตัวแปร 2 ตัวแปร อยู่ในวงเว็บหลังฟังก์ชัน ดังนี้ **def** area\_Triangle(High,Base) :

โดยจะรับค่าจากฟังก์ชัน print(**"Area ="**, area\_Triangle(5,6) )

***อาร์กิวเมนต์ (Argument)*** *คือ ค่าคงที่ ที่ใช้สำหรับส่งค่าไปยังฟังก์ชั่นที่เรียกใช้งาน เพื่อนำไปประมวลผลต่อ ค่าอาร์กิวเมนต์จะถูกแนบส่งไปพร้อมกับชื่อฟังก์ชั่นอยู่ในเครื่องหมายวงเล็บ หากมีอาร์กิวเมนต์หลายตัวจะคั่นด้วยเครื่องหมาย comma (,)* ***ดังนี้*** print(**"Area ="**, area\_Triangle(5,6) )

***พารามิเตอร์ (Parameter)*** *คือ ตัวแปรที่ประกาศไว้รอรับค่าอาร์กิวเมนต์ อยู่ในเครื่องหมายวงเล็บหลังชื่อฟังก์ชั่น หากมีพารามิเตอร์หลายตัวจะคั่นด้วยเครื่องหมาย comma (,)* ***ดังนี้* def** area\_Triangle(High,Base) :

**การส่งค่าอาร์กิวเมนต์มี 2 วิธี คือ**

**1.การส่งค่าข้อมูล (Call by Value)**

ค่าอาร์กิวเมนต์จะถูกทำการ copy แล้วส่งให้กับพารามิเตอร์ของฟังก์ชัน จำนวนพารามิเตอร์ต้องเท่ากับจำนวนอาร์กิวเมนต์ที่ส่งมาให้ วิธีนี้สิ้นเปลืองหน่วยความจำ

High = float(input(**"ความสูง ="**))   
Base = float(input(**"ความยาวฐาน ="**))  
**def** area\_Triangle() :  
 area = 0.5\*High\*Base   
 **return** area   
print(**"Area ="**, area\_Triangle() )

**2. การส่งค่าแบบอ้างอิง (Call by Reference)**

ใช้กับข้อมูลชนิดออปเจ็ค โดยส่งค่าตำแหน่งอาร์กิวเมนต์ในหน่วยความจำ ไปให้กับพารามิเตอร์ เมื่อมีการเปลี่ยนแปลงข้อมูลของค่าพารามิเตอร์จะทำให้อาร์กิวเมนต์ที่ส่งไปเปลี่ยนตามไปด้วย วิธีนี้จะช่วยประหยัดหน่วยความจำ

**1 def** msg(f\_lst):  
2 print(**"ค่าพารามิเตอร์ที่รับมาจาก arg\_lst ="**, f\_lst)  
3 f\_lst.append(**"easy to learn."**)  
4 print(**"เปลี่ยนค่าพารามิเตอร์ = "**, f\_lst)  
  
5 arg\_lst = [**"Python is"**]  
6 print(**"ค่าอาร์กิวเมนต์ที่ส่งให้ค่าพารามิเตอร์ ="**, arg\_lst)  
7 msg(arg\_lst)  
8 print(**"ค่าอาร์กิวเมนต์หลังจากเปลี่ยนค่าพารามิเตอร์ ="**, arg\_lst)

หลักการทำงานจะเป็นไปตามนี้ คือ รับข้อมูลบรรทัดที่5 เข้าบรรทัดที่ 1 เพิ่มข้อมูลบรรทัดที่3 ต่อท้ายข้อมูลบรรทัดที่5

แสดงข้อมูลแบบสมบูรณ์บรรทัดที่ 8

**รูปแบบการส่งค่าอาร์กิวเมนต์ให้กับค่าพารามิเตอร์**

**การส่งค่าอาร์กิวเมนต์แบบ Required arguments**

เป็นการส่งอาร์กิวเมนต์ให้พารามิเตอร์ ตามชนิดข้อมูล และลำดับ โดยต้องมีจำนวนเท่ากันด้วย

**def req\_arg(numlst, str\_, num):**

**print ("แสดงค่าข้อมูลในลิสต์ =", numlst)**

**print ("แสดงค่าข้อมูลในตัวแปร num =", num)**

**print ("แสดค่าข้อมูลในตัวแปร str\_ =", str\_)**

**lst = [1,2,3]; msg = "Python"**

**req\_arg(lst, msg, 50)**

**แสดงผล**

แสดงค่าข้อมูลในลิสต์ = [1, 2, 3]

แสดงค่าข้อมูลในตัวแปร num = 50

แสดค่าข้อมูลในตัวแปร str\_ = Python

**การส่งค่าอาร์กิวเมนต์แบบ Keyword arguments**

เป็นการส่งอาร์กิวเมนต์ให้พารามิเตอร์ได้โดยไม่ต้องเรียงลำดับ แต่ต้องมีชื่อเดียวกันและต้องมีจำนวนเท่ากัน

**def key\_arg(lst, msg, num):**

**print("แสดงค่าข้อมูลในลิสต์ =", lst)**

**print("แสดงค่าข้อมูลในตัวแปร num =", num)**

**print("แสดงค่าข้อมูลในตัวแปร msg =", msg)**

**แสดงผล**

แสดงค่าข้อมูลในลิสต์ = [1, 2, 3]

แสดงค่าข้อมูลในตัวแปร num = 10

แสดงค่าข้อมูลในตัวแปร msg = Python is easy

**key\_arg(lst=[1, 2, 3], num=10, msg="Python is easy")การส่งค่าอาร์กิวเมนต์แบบ Default arguments**

มีค่าพารามิเตอร์บางตัวถูกกำหนดค่าไว้ล่วงหน้า ซึ่งเมื่อเรียกใช้ก็จะส่งค่าอาร์กิวเมนต์เฉพาะจำนวนที่เหลือให้พารามิเตอร์เท่านั้น

**def def\_arg(r, pi=3.14):**

**result = 2\* pi \* r**

**print("เส้นรอบวงกลม =", result)**

**def\_arg(7) แสดงผล**

เส้นรอบวงกลม = 43.96

**การส่งค่าอาร์กิวเมนต์แบบ Variable-length arguments**

เป็นการสร้างพารามิเตอร์ไว้รับค่าอาร์กิวเมนต์แบบไม่จำกัดจำนวน โดยชื่อพารามิเตอร์ต้องมีเครื่องหมาย (\*)นำหน้า

**def varleng\_arg(\*num):**

**print("จำนวนข้อมูลในตัวแปร num :")**

**for var in num:**

**print(var, end=" ")**

**print(" ")**

**result = num[2] \* 2**

**print("ผลคูณพารามิเตอร์ num[2] \* 2 =", result)**

**varleng\_arg(10, 20, 30, 40)**

**แสดงผล**

จำนวนข้อมูลในตัวแปร num :

10 20 30 40

ผลคูณพารามิเตอร์ num[2] \* 2 = 60

**Lambda Expressions**

คือ anonymous function ที่เป็นฟังก์ชันที่มีการทำงานขนาดเล็กอยู่ภายใน เราสามารถสร้างโดยใช้คำสั่ง lambda เราสามารถใช้ Lambda Expressions สร้างออบเจ็กต์ของฟังก์ชันได้ และค่า return จะเป็นค่าที่ได้จากผลลัพธ์ของ Expression ของฟังก์ชัน

|  |  |
| --- | --- |
| โค้ดโปรแกรม | การแสดงผล |
| f = lambda x: x + 1  print(f(2))  print(f(8)) | 3  9 |
| g = lambda a, b: (a + b) / 2  print(g(3, 5))  print(g(10, 33)) | 4.0  21.5 |
| def make\_incrementor(n):  return lambda x: x + n  f = make\_incrementor(13)  print(f(0))  print(f(1))  print(f(5)) | 13  14  18 |

นอกจากนี้ Lambda Expressions ยังมีประโยชน์เพื่อใช้งานกับ built-in function เช่น ฟังก์ชัน filter() และฟังก์ชัน map() ในภาษา Python

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

print(list(filter(lambda x: x % 5 == 0, numbers))) [15, 5, 10, 30]

print(list(map(lambda x: x \* 2, numbers))) [4, 30, 10, 14, 20, 6, 56, 60]

## Aray (อาเรย์)

## numpy เป็นมอดูลที่ทำให้เราสามารถใช้ออบเจ็กต์ชนิดที่เรียกว่า ndarray ซึ่งหมายถึงอาเรย์หลายมิติ

## ก่อนที่จะเริ่มใช้งานสิ่งที่ต้องทำเป็นอย่างแรกก็คือทำการ import เรียกใช้ขึ้นมาก่อน

## import numpy as np

## ตัวย่อ np นี้จะใช้แบบนี้ไปตลอดในบทความทุกบท เพราะค่อนข้างเป็นสากล แม้แต่เวลาที่เรียกชื่อฟังก์ชันต่างๆใน numpy ก็จะเรียกโดยขึ้นต้นด้วย np.

## อาเรย์มีแอตทริบิวต์ติดตัวที่สามารถให้ข้อมูลของตัวอาเรย์นั้น เช่น

## shape รูปร่างของอาเรย์

## size จำนวนสมาชิกในอาเรย์

## ndim จำนวนมิติของอาเรย์

**การสร้างชุดข้อมูลอาเรย์(array)**

เราสามารถสร้างชุดข้อมูลอาเรย์โดยใช้รูปแบบเดียวกันกับการสร้างชุดข้อมูลชนิดลิสต์ โดยนำข้อมูลชนิดเดียวกันมาแยกแต่ละข้อมูลด้วยเครื่องหมาย “,” และข้อมูลทั้งหมดอยู่ภายในขอบเขตเครื่องหมาย “[]”

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

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

**การเข้าถึงข้อมูลแต่ละตัวในอาเรย์(array)**

เราสามารถจะเข้าถึงข้อมูลแต่ละตัวในชุดข้อมูลอาเรย์โดยอ้างถึงลำดับของข้อมูล แบบเดียวกันกับชุดข้อมูลชนิดลิสต์ ซึ่งทำการระบุลำดับของข้อมูลที่ต้องการเข้าถึงไว้ภายในเครื่องหมาย “[]”

การเข้าถึงข้อมูลแต่ละตัวในอาเรย์(array)ด้วยลำดับข้อมูลซึ่งเป็นจำนวนเต็มบวก

ลำดับข้อมูลเป็นตำแหน่งที่ข้อมูลถูกบรรจุอยู่ในชุดข้อมูลอาเรย์ โดยใน python ลำดับข้อมูลนั้นจะเริ่มต้นที่ลำดับที่ 0

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# 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(negative indexing)**

เราสามารถจะอ้างถึงลำดับของชุดข้อมูลในอาเรย์โดยใช้เลขจำนวนเต็มลบในรูปแบบเดียวกับที่ใช้ในชุดข้อมูลชนิดลิสต์ ซึ่งเป็นคุณลักษณะที่ไม่ค่อยมีในการใช้งานอาเรย์โดยทั่วไป หากเราอ้างถึงข้อมูลลำดับที่ -1 ก็หมายถึงเราอ้างถึงข้อมูลตัวสุดท้ายของชุดข้อมูลอาเรย์ หากอ้างถึงข้อมูลลำดับที่ -2 ก็หมายถึงเราอ้างถึงข้อมูลก่อนตัวสุดท้าย ตามลำดับ

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน จากลำดับของข้อมูลที่ระบุ

print( scores[-1] )

# โปรแกรมทำการปรินต์ค่า

# 90

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน จากลำดับของข้อมูลที่ระบุ

print( scores[-2] )

# โปรแกรมทำการปรินต์ค่า

# 81

**การตรวจสอบจำนวนข้อมูลในอาเรย์(array)**

เราสามารถทำการตรวจสอบจำนวนของข้อมูลที่มีอยู่ภายในชุดข้อมูลอาเรย์ ซึ่งใช้รูปแบบเดียวกับการหาจำนวนข้อมูลที่มีอยู่ในชุดข้อมูลชนิดลิสต์ โดยการใช้งานฟังก์ชัน len()

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการปรินต์ค่าข้อมูล จำนวนของข้อมูลคะแนนสอบของนักเรียนที่ถูกเก็บในชุดข้อมูลอาเรย์

print( len(scores) )

# โปรแกรมทำการปรินต์ค่า

# 5

**การเพิ่มข้อมูลลงไปในชุดข้อมูลอาเรย์(array)**

เราสามารถใช้ฟังก์ชัน append() ในการเพิ่มข้อมูลใหม่ลงไปในชุดข้อมูลอาเรย์ ซึ่งจะเพิ่มเข้าไปหลังข้อมูลลำดับสุดท้ายที่มีอยู่ในอาเรย์

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการเพิ่มข้อมูลลงไปในชุดข้อมูลอาเรย์

scores.append(65)

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

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

**การลบข้อมูลออกจากชุดข้อมูลอาเรย์(array)**

เราสามารถลบข้อมูลออกจากชุดข้อมูลอาเรย์โดยใช้คีย์เวิร์ด del หรืออาจจะลบข้อมูลโดยใช้ฟังก์ชัน remove() และ pop() ในการลบข้อมูลออก

**การลบข้อมูลออกจากชุดข้อมูลอาเรย์(array) โดยใช้คีย์เวิร์ด del**

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการลบข้อมูลคะแนนสอบของนักเรียนในลำดับที่ระบุ ด้วยคีย์เวิร์ด del

del scores[4]

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

# [80, 85, 75, 81]

จากตัวอย่างข้างต้น เราใช้คีย์เวิร์ด del เพื่อทำการลบข้อมูลออกจากอาเรย์จากตำแหน่งของข้อมูลลำดับที่ 4 ซึ่งก็คือตัวเลข 90 ออกจากอาเรย์ ทำให้ข้อมูลในอาเร์เหลือเพียง [80, 85, 75, 81]

**การลบข้อมูลออกจากชุดข้อมูลอาเรย์(array) โดยใช้ฟังก์ชัน remove()**

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการลบข้อมูลคะแนนสอบของนักเรียนที่มีค่าตามข้อมูลที่ระบุให้ฟังก์ชัน remove()

scores.remove(85)

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

# [80, 75, 81, 90]

จากตัวอย่างข้างต้น เราใช้ฟังก์ชัน remove() เพื่อทำการลบข้อมูลออกจากอาเรย์โดยลบเฉพาะข้อมูล 85 ออก ทำให้ข้อมูลในอาเรย์เหลือเพียง [80, 75, 81]

**การลบข้อมูลออกจากชุดข้อมูลอาเรย์(array) โดยใช้ฟังก์ชัน pop()**

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการลบข้อมูลคะแนนสอบของนักเรียนตามตำแหน่งที่ระบุ และคืนค่าข้อมูลที่ตำแหน่งดังกล่าว

# โดยใช้ฟังก์ชัน pop()

pop\_value = scores.pop(2)

# ทำการปรินต์ค่าข้อมูลคะแนนสอบที่ถูกลบออกจากอาเรย์

print(pop\_value)

# โปรแกรมทำการปรินต์ค่า

# 75

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

# [80, 85, 81, 90]

จากตัวอย่างข้างต้น เราใช้ฟังก์ชัน pop() เพื่อทำการลบข้อมูลออกจากอาเรย์โดยลบเฉพาะข้อมูลลำดับที่ 2 ออกและยังคืนค่าข้อมูลที่ทำการลบออกมาด้วย ทำให้ข้อมูลในอาเรย์เหลือเพียง [80, 85, 81, 90]

**การเปลี่ยนแปลงข้อมูลภายในชุดข้อมูลอาเรย์(array)**

เราสามารถทำการเปลี่ยนแปลงข้อมูลภายในชุดข้อมูลอาเรย์ โดยอ้างถึงลำดับของข้อมูลที่ต้องการเปลี่ยนแปลง จากนั้นทำการหนดค่าใหม่ให้ข้อมูลที่ตำแหน่งดังกล่าว โดยใช้เครื่องหมาย “=” เช่นเดียวกับชุดข้อมูลชนิดลิสต์

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการแก้ไขข้อมูลคะแนนสอบของนักเรียนในชุดข้อมูลอาเรย์ในตำแหน่งที่ระบุ

# ด้วยค่าของข้อมูลที่อยู่ด้านขวาของเครื่องหมาย '='

scores[2] = 78

# ทำการปรินต์ค่าข้อมูลคะแนนสอบของนักเรียน

print( scores )

# โปรแกรมทำการปรินต์ค่า

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

จากตัวอย่างจะเห็นว่า ต้องการเปลี่ยนแปลงข้อมูลของคะแนนในลำดับที่ 2 คือ 75 โดยจะถูกแทนที่ด้วยข้อมูลใหม่คือ 78 ลงในชุดข้อมูลอาเรย์

**การใช้เครื่องหมายดำเนินการเพื่อดำเนินการกับข้อมูลภายในชุดข้อมูลอาเรย์(array)**

เราสามารถใช้เครื่องหมายดำเนินการเช่น “+”, “\*” เพื่อทำการดำเนินการกับข้อมูลในชุดข้อมูลอาเรย์ เช่นเดียวกับการใช้เครื่องหมายดำเนินการกับชุดข้อมูลชนิดลิสต์ ดังนี้

**การใช้เครื่องหมายดำเนินการ ‘+’ เพื่อดำเนินการกับข้อมูลภายในชุดข้อมูลอาเรย์(array)**

เครื่องหมาย “+” จะใช้ในการรวมข้อมูลจากอาเรย์ 2 ตัวให้เป็นอาเรย์ใหม่

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# 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, 7

**การใช้เครื่องหมายดำเนินการ ‘\*’ เพื่อดำเนินการกับข้อมูลภายในชุดข้อมูลอาเรย์(array)**

เครื่องหมาย “\*” เพื่อจะคัดลอกข้อมูลเดิมซ้ำๆ หลายครั้ง

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรสำหรับเก็บข้อมูลข้อความในรูปแบบชุดข้อมูลอาเรย์

messages = ["hello"]

# ทำการคัดลอกข้อมูลดังกล่าวซ้ำเป็นจำนวน 3 ข้อมูลด้วยเครื่องหมายดำเนินการ '\*'

messages = messages \* 3

# ทำการปรินต์ค่าชุดข้อมูลข้อความที่ได้ทำการคัดลอกเรียบร้อยแล้ว

print( messages )

# โปรแกรมทำการปรินต์ค่า

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

**การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์(array) โดยระบุช่วงของข้อมูล**

เราสามารถเข้าถึงข้อมูลในชุดข้อมูลอาเรย์โดยระบุเป็นช่วงของข้อมูลได้ โดยทำการระบุจุดเริ่มต้นและจุดสิ้นสุดของช่วงข้อมูล โดยใช้รูปแบบ [จุดเริ่มต้น:จุดสิ้นสุด] โดยจะดึงข้อมูลตั้งแต่ข้อมูลตัวที่ระบุในตำแหน่งเริ่ม ไปจนถึงข้อมูลก่อนตำแหน่งตำแหน่งสิ้นสุด(ไม่รวมข้อมูลในตำแหน่งสิ้นสุดที่ระบุ)เช่นเดียวกับการเข้าถึงข้อมูลโดยระบุช่วงจากชุดข้อมูลชนิดลิสต์

**การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์(array) โดยระบุช่วงของข้อมูล ทำการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุด**

การเข้าถึงข้อมูลโดยการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุดนี้ จะดึงข้อมูลตั้งแต่ข้อมูลตัวที่ระบุในตำแหน่งที่ระบุในจุดเริ่มต้น ไปจนถึงข้อมูลก่อนตำแหน่งที่ระบุในจุดสิ้นสุด(ไม่รวมข้อมูลในตำแหน่งที่ระบุในจุดสิ้นสุด)

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# 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]

จากตัวอย่าง

scores[ 2:4 ] เราได้ทำการระบุขอบเขตของข้อมูลตำแหน่งของจุดเริ่มต้นและจุดสิ้นสุดเป็น 2 และ 4 ดังนั้นข้อมูลสำหรับช่วงที่กำหนดก็จะเริ่มพิจารณาตั้งแต่ข้อมูลตำแหน่งที่ 2(รวมตำแหน่งที่สองด้วย) ไปจนถึงตำแหน่งที่ 4(โดยไม่รวมข้อมูลตำแหน่งนี้เข้าไป) ดังนั้นข้อมูลที่ได้คือข้อมูลตำแหน่งที่ 2 และ 3

scores[ -3:-1 ] เราได้ทำการระบุขอบเขตของข้อมูลตำแหน่งของจุดเริ่มต้นและจุดสิ้นสุดเป็น -3 และ 1 โดยเริ่มพิจารณาตั้งแต่ข้อมูลตำแหน่งที่ -3(ซึ่งก็คือตัวที่สามนับจากตัวสุดท้าย) ไปจนถึงตำแหน่งที่ -1(คือข้อมูลตัวสุดท้าย) โดยไม่รวมข้อมูลตัวสุดท้ายนี้เข้าไปด้วย ดังนั้นข้อมูลที่ได้คือ ข้อมูลตำแหน่งที่ 2 และ 3

**การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์(array)โดยระบุช่วงของข้อมูล ระบุเพียงจุดเริ่มต้น**

การเข้าถึงข้อมูลโดยการระบุเพียงจุดเริ่มต้นนี้ จะดึงข้อมูลตั้งแต่ข้อมูลตัวที่ระบุในจุดเริ่มต้น ไปจนถึงข้อมูลตัวสุดท้ายของอาเรย์

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการปรินต์ค่าชุดข้อมูลคะแนนสอบของนักเรียนโดยทำการระบุช่วงของข้อมูล

# โดยทำการระบุเพียงจุดเริ่มต้นของข้อมูล

print( scores[ -2: ] )

# โปรแกรมทำการปรินต์ค่า

# [81, 90]

จากตัวอย่าง เราทำการระบุเฉพาะตำแหน่งเริ่มต้นของขอบเขตข้อมูลเท่านั้น ซึ่งจะทำการพิจารณา ข้อมูลตั้งแต่ตำแหน่งที่ระบุในจุดเริ่มต้น(รวมข้อมูลตำแหน่งเริ่มต้นด้วย) ไปจนถึงข้อมูลตัวสุดท้ายของอาเรย์ ดังนั้นข้อมูลที่ได้คือข้อมูลตำแหน่งที 3 และ 4

**การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์(array)โดยระบุช่วงของข้อมูล ระบุเพียงจุดสิ้นสุด**

การเข้าถึงข้อมูลโดยการระบุเพียงจุดสิ้นสุดนี้ จะดึงข้อมูลตั้งแต่ข้อมูลตัวแรกของอาเรย์ไปจนถึงข้อมูลก่อนตำแหน่งที่ระบุในจุดสิ้นสุด(ไม่รวมข้อมูลในตำแหน่งที่ระบุในจุดสิ้นสุด)

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการปรินต์ค่าชุดข้อมูลคะแนนสอบของนักเรียนโดยทำการระบุช่วงของข้อมูล

# โดยทำการระบุเพียงจุดสิ้นสุดของข้อมูล

print( scores[ :2 ] )

# โปรแกรมทำการปรินต์ค่า

# [80, 85]

จากตัวอย่าง

scores[ :2 ] เราทำการระบุเฉพาะตำแหน่งการสิ้นสุดของขอบเขตข้อมูลเท่านั้น ซึ่งจะทำการพิจารณา ข้อมูลตั้งแต่ตำแหน่งแรกจนถึงตำแหน่งสิ้นสุดที่ระบุ(โดยไม่รวมข้อมูลที่ตำแหน่งสิ้นสุด) ดังนั้นข้อมูลที่ได้คือข้อมูลตำแหน่งที่ 0, 1

**การเข้าถึงข้อมูลในชุดข้อมูลอาเรย์โดยระบุช่วงของข้อมูล ไม่ทำการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุด**

การเข้าถึงข้อมูลโดยไม่ระบุทั้งจุดเริ่มต้นและจุดสิ้นสุดนี้ จะดึงข้อมูลตั้งแต่ข้อมูลตัวแรกของอาเรย์ไปจนถึงข้อมูลตัวสุดท้ายของอาเรย์

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# กำหนดตัวแปรเพื่อเก็บข้อมูลชนิดอาเรย์ด้วยรูปแบบข้อมูลจากชุดข้อมูลชนิดลิสต์

# เพื่อทำการเก็บคะแนนสอบของนักเรียน

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

# ทำการปรินต์ค่าชุดข้อมูลคะแนนสอบของนักเรียนโดยทำการระบุช่วงของข้อมูล

# โดยไม่ทำการระบุทั้งจุดเริ่มต้นและจุดสิ้นสุดของข้อมูล

print( scores[:] )

# โปรแกรมทำการปรินต์ค่า

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

จากตัวอย่าง เราไม่ได้ทำการระบุตำแหน่งเริ่มต้นและสิ้นสุดของข้อมูล ซึ่งจะทำการดึงข้อมูลตั้งแต่ข้อมูลตัวแรกไปจนถึงตัวสุดท้ายจากชุดข้อมูลอาเรย์

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

**การสร้างอาเรย์หลายมิติ(multidimensional arrays)**

อาเรย์หลายมิติก็คือ การมีข้อมูลอาเรย์ซ้อนอยู่ภายในข้อมูลอาเรย์อีกชั้นหนึ่ง ซึ่งอาจจะมีอาเรย์ย่อยๆ เป็นส่วนประกอบภายใน

การสร้างอาเรย์หลายมิติ

ตัวอย่างการใช้งาน

# -\*- coding: utf-8 -\*-

# 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=17