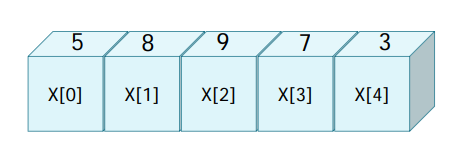
**สรุปเนื้อหา Array**

นายธีรพัฒร์ รุจิโภชน์ รหัสนักศึกษา 362516232022 วิศวกรรมไฟฟ้า (สมทบ)

อาร์เรย์ (Array) เป็นตัวแปรชุดให้สำหรับเก็บข้อมูลที่มีความสัมพันธ์ กันโดยจะเก็บไว้ในชื่อเดียวกัน

สมาชิดแต่ละตัวของ Array จะเรียกว่า Element หรือ Cell ตัวเลขที่ใช้ระบุตำแหน่งสมาชิกของ Array เรียกว่า Index หรือ Subscript

****

ตัวอย่าง Array X ที่มี 5 Element ซึ่งจะเริ่มตันตั้งแต่ Index 0 ถึง 4

ตัวแปรเปรียบเหมือนกล่อง

Int score = 30;

อาร์เรย์ คือแถวของตัวแปรชนิดเดียวกัน เปรียบได้กับการนำ กล่องหลายๆใบมาวางเรียงกันเป็นแถว

int[] scores = { 30, 50, 85, 10, 45 };

**ประเภทของอาร์เรย์**

1.อาร์เรย์ของชนิดข้อมูลพื้นฐาน

**1.อาร์เรย์ของชนิดข้อมูลพื้นฐาน**

ตัวแปรอาร์เรย์ (Array) หรือตัวแปรชุด เป็นโครงสร้างข้อมูล ( Data Structure) ทำหน้าที่ในการจองเนื้อที่ในหน่วยความจำ โดยแบ่งเนื้อที่ขนาดหน่วยความจำตามที่ระบุขนาดหน่วยความจำในตัวแปรของอาร์เรย์ เพื่อเก็บข้อมูลแต่ละช่องของขนาดหน่วยความจำ เมื่อต้องการเรียกใช้หรือจัดเก็บเพียงระบุตำแหน่งของข้อมูลในอาร์เรย์ (Array) เพื่อเป็นตัวบ่งชี้ตำแหน่งของข้อมูล เช่น salary[0], salary[1],…, salary[4] เป็นต้น ซึ่งประเภทตัวแปรอาร์เรย์ มีดังนี้

## 1.ตัวแปรอาร์เรย์แบบ 1 มิติ

ตัวแปรแบบอาร์เรย์ ที่ใช้ในการเก็บข้อมูลโดยมีการอ้างอิงที่ใช้เลขดัชนีเพียงหนึ่งค่าจะเรียกว่า ตัวแปรแบบอาร์เรย์ 1 มิติ เช่น การเก็บข้อมูลคะแนน ที่อ้างอิงด้วยรหัสนักศึกษา

ตัวอย่าง

int age[10];

เป็นการประกาศตัวแปรแบบอาร์เรย์เพื่อเก็บข้อมูลอายุ เมื่อต้องการเก็บข้อมูลอายุของนักศึกษาคนที่ 1 สามารถทำได้ดังนี้

age[1] = 18;

**การประกาศ ตัวแปรอาร์เรย์**

การประกาศตัวแปรแบบอาร์เรย์สามารถทำได้คล้ายกับการประกาศตัวแปรอื่นๆ ทั่วไป โดยการกำหนดซื่อของตัวแปร ชนิดของข้อมูล และขนาดของข้อมูล ตัวอย่างเช่น

int grades[5]; /\* Array ขนาด 5 ของ int \*/

เป็นการประกาศตัวแปรแบบอาร์เรย์ 1 มิติชื่อ grades ให้เป็นข้อมูลแบบอักษรโดยมีขนาดเท่ากับ 5 อีลีเมนต์ ซึ่งการประกาศตัวแปรนี้จะจองหน่วยความจำเท่ากับ 2 byte \* 5 = 10 byte

**1.1การให้ค่าเริ่มต้น (Array Initialization)**

การกำหนดค่าเริ่มต้นของตัวแปรแบบอาร์เรย์ 1 มิติ สามารถกำหนดได้สำหรับทุกชนิดข้อมูล ดังตัวอย่างต่อไปนี้

int grades[5] = {98,87,92,79,85};

char codes[6] = {‘s’, ‘a’, ‘m’, ‘p’, ‘l’, ‘e’};

double width[7] = {10.96, 6.43, 2.58, 0.86, 5.89, 7.56,8.22};

int temp[4] = {10,20,30,40};

float temp[4] = {98.6, 97.2, 99.0 , 101.5};

การเรียงข้อมูลคือค่าแรกจะถูกกำหนดให้กับอีลีเมนต์ตัวที่ 1 (ดัชนีที่ 0) และ ค่าที่สองจะถูกกำหนดให้อีลีเมนต์ตัวที่ 2 (ดัชนีที่ 1) และ ถัด ๆ ไป

เนื่องจากตัวแปลภาษาซี (C Complier) จะไม่สนใจช่องว่าง (space bar) ในโปรแกรม ดังนั้นเราสามารถกำหนดค่าเริ่มต้นให้สามารถขึ้นบรรทัดใหม่ได้ดังนี้

int gallons[20] = {19, 16, 14, 19, 20,

18, 12, 10, 22, 15,

18, 17, 16, 24, 23,

19, 15, 18, 21, 5 };

จะเห็นว่าใช้ 4 บรรทัดในการกำหนดค่าเริ่มต้นให้กับตัวแปร gallons และถ้ามีการกำหนดค่าเริ่มต้นให้กับตัวแปรอาร์เรย์ไม่ครบ ก็จะทำการให้ค่าเริ่มต้นเฉพาะค่าที่มีการกำหนดส่วนที่เหลือจะกำหนดให้เป็นศูนย์ เช่น

float length[7] = {7.8, 6.4, 4.9, 11.2};

จะได้ว่าสี่ค่าแรกคือ length[0] ถึง length[3] จะมีค่าเริ่มต้น ส่วน length[4] ถึง length[6] จะถูกกำหนดให้มีค่าเป็นศูนย์ ในการกำหนดค่าเริ่มต้นจะไม่สามารถทำการกำหนดค่าเริ่มต้นของ อีลีเมนต์ตัวหลังได้ ในขณะที่ยังไม่มีการกำหนดค่าให้กับอีลีเมนต์ตัวแรก ๆ และถ้าไม่มีการกำหนดค่าเริ่มต้น

ในการประกาศตัวแปรแบบอาร์เรย์พร้อมทั้งกำหนดค่าเริ่มต้น สามารถทำได้โดยไม่จำเป็นต้องกำหนดขนาดของอาร์เรย์ ก็ได้เช่น

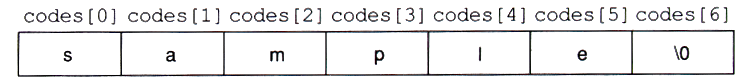
char codes[6] = {‘s’, ‘a’, ‘m’, ‘p’, ‘l’, ‘e’};

char codes[] = {‘s’, ‘a’, ‘m’, ‘p’, ‘l’, ‘e’};

ทั้งสองบรรทัดทำงานเหมือนกัน โดยตัวแปลภาษา จะทราบด้วยตัวเองว่าขนาดของ อาร์เรย์จะมีขนาดเป็น 6 ในกรณีจำเป็นจะต้องมีการประกาศค่าเริ่มต้น ในกรณีให้ค่าเริ่มต้นแก่ตัวแปรอักษรสามารถ กระทำได้ดังนี้ด้วย คือ

char code[] = “sample”;

ซึ่งทำให้เข้าใจได้ว่า สายตัวอักษร หรือเรียกว่า สตริง (String) คือลำดับของตัวอักษรที่เรียงต่อกันนั้นเอง แต่ในกรณีที่เป็นการให้ค่าแบบใช้เครื่องหมายคำพูด หรือแบบสายอักษรนี้ จะทำให้ขนาดของอาร์เรย์มีขนาดเพิ่มจากเดิมมาหนึ่งค่า (ดังรูปที่ 6.4) เพื่อใช้ในการเก็บ “\0” หรือเรียกว่า “*null character”* ซึ่งจะถูกเพิ่มเข้าไปโดยอัตโนมัติในส่วนท้ายของสายอักษรซึ่งมีค่าเท่ากับ 0 ทำหน้าที่ในการบอกการสิ้นสุดของสายอักษร



**1.2การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน**

การส่งผ่านค่าอาร์เรย์ไปยังฟังก์ชันสามารถทำได้ สำหรับกรณีการส่งผ่านเฉพาะบางอีลีเมนต์จะสามารถส่งได้โดยค่าของแต่ละอีลีเมนต์ได้โดยตรง เช่น

find\_min(grades[2], grades[6]);

จะเป็นการส่งผ่านค่า grades[2] และ grades[6] ไปยังฟังก์ชัน *find\_min*()

กรณีที่ต้องการส่งผ่านค่าของอาร์เรย์ทั้งหมด สามารถกระทำได้ง่ายกว่าเพียงแต่ส่งเฉพาะชื่อของอาร์เรย์มาเท่านั้นเช่น การส่งผ่านอาร์เรย์ grades ไปยังฟังก์ชัน find\_max() ได้ดังนี้ find\_max(grades); จะทำให้ค่าของ grades สามารถเรียกใช้งานได้ภายใน find\_max()

ในการส่งผ่านค่าในรูปแบบของตัวแปรทั่วไปจะเห็นว่าจะทำการสำเนาข้อมูลไปให้กับฟังก์ชันนั้น ซึ่งทำให้เสียเวลา แต่ในกรณีของอาร์เรย์ในการส่งผ่านจะทำการอ้างถึงตัวแปรอาร์เรย์โดยตรงนั้นคือ อาร์เรย์ภายนอกกับภายในฟังก์ชันจะเป็นตัวเดียวกัน เช่น

int nums[5];

char keys[256];

double units[500], prices[500];

เราสามารถเรียกใช้ฟังก์ชัน โดยการส่งผ่านค่าอาร์เรย์เหล่านั้นไปได้ดังนี้

find\_max(nums);

find\_ch(keys);

calc\_tot(nums, units, prices);

**2.ตัวแปรอาร์เรย์แบบ 2 มิติ**

ตัวแปรอาร์เรย์แบบ 2 มิติจะเป็นตัวแปรที่มีการอ้างอิงถึงค่าข้อมูลโดยใช้ค่าเลขดัชนี 2 ค่าซึ่งประกอบไปด้วยค่าดัชนีที่ใช้ในการอ้างอิงในแนวแถว (row) และค่าดัชนีที่ใช้อ้างอิงในแนวคอลัมน์(column) ตัวอย่างเช่น

8 16 9 52

3 15 27 6

14 25 2 10

ซึ่งเรียกอาร์เรย์แบบนี้ว่าอาร์เรย์ 2 มิติของจำนวนเต็ม ซึ่งประกอบด้วย 3 แถว 4 คอลัมน์ การประกาศตัวแปรอาร์เรย์ 2 มิติจะถ้ากับ 1 มิติแต่เพิ่มเติมการกำหนดขนาด ซึ่งจะต้องระบุทั้งในแนวแถวและคอลัมน์ เช่น

int val[3][4];

เป็นการประกาศตัวแปรชื่อ val เป็นตัวแปรอาร์เรย์ 2 มิติเพื่อเก็บข้อมูลจำนวนเต็ม ซึ่งจะจองหน่วยความจำเท่ากับ 2 ไบต์ \* 3 \* 4 = 24 ไบต์

double prices[10][5];

เป็นการประกาศตัวแปร prices ซึ่งจะจองหน่วยความจำเท่ากับ 8 byte \* 10 \* 50 = 4000 ไบต์

char code[6][10];

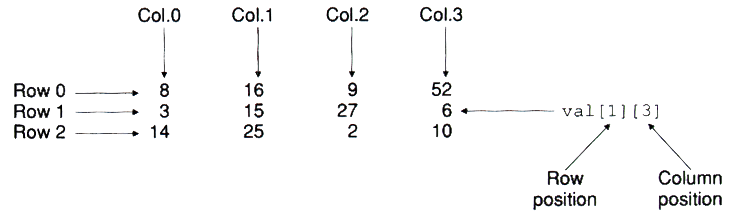
เป็นการประกาศตัวแปร code ซึ่งจะจองหน่วยความจำเท่ากับ 1 ไบต์ \* 6 \* 10 = 60 ไบต์ ในการอ้างอิงถึงข้อมูลสำหรับอีลีเมนต์ใดในอาร์เรย์ 2 มิติ ทำได้โดยการระบุค่าเลขดัชนีทั้งในแนวแถวและแนวคอลัมน์ จากรูปที่ 6.6 ตำแหน่งของ val[1][3] จะสามารถบอกตำแหน่งกำหนดค่าดัชนีในแนวแถวเป็น 1 และในแนวคอลัมน์ 3 เช่นเดียวกับตัวแปรอาร์เรย์ 1 มิติ และตัวแปรทั่วไป ที่สามารถใช้งานได้ที่ตำแหน่งต่าง ๆ ภายในโปรแกรม เช่น

num = val[2][3];

val[0][0] = 62;

new\_nu = 4 \* ( val[1][0] – 5);

sum\_row0 = val[0][0] + val[0][1 + val[0][2 + val[0][3];



ตัวแปรอาร์เรย์ 2 มิติจะมีลักษณะเช่นเดียวกับตัวแปรอาร์เรย์ 1 มิติที่จะสามารถประกาศใช้งานได้ทั้งในและนอกฟังก์ชัน ซึ่งอาร์เรย์ที่ประกาศไว้ในฟังก์ชันจะเป็นชนิดแบบ *“Local”* แต่ถ้าประกาศไว้ข้างนอกจะเรียกว่า *“Global*” ตัวอย่าง เช่น

int bingo[2][3];

main()

{

static double lotto[104][6];

double pick\_six[52][6];

:

}

การกำหนดค่าเริ่มต้นในตัวแปรอาร์เรย์ 2 มิติสามารถกระทำได้เช่นเดียวกันกับตัวแปรอาร์เรย์ 1 มิติโดยจะเพิ่มส่วนการแบ่งแยกในแต่ละแถวด้วยเครื่องหมาย “{ }” และ “,” เช่น

int val[3][4] = {{8,16, 9,52},

{3,15,27, 6},

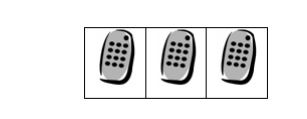
{14,25,2,10}};

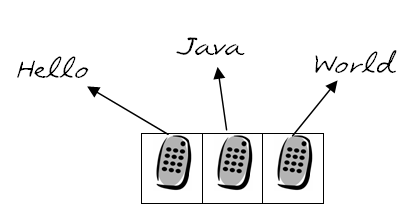
จากคำสั่งเป็นการประกาศตัวแปร val เป็นแบบอาร์เรย์ 2 มิติที่มีชนิดของข้อมูลเป็นจำนวนเต็ม โดยมีขนาดเป็น 3 แถว 4 คอลัมน์และมีการประกาศค่าเริ่มต้นด้วย ในการกำหนดค่าเริ่มต้นค่าในเครื่องหมายปีกกาชุดแรกจะเป็นการกำหนดค่าให้กับอาร์เรย์ในแถวที่ 0 และเครื่องหมายปีกกาชุดที่สองจะเป็นการกำหนดค่าให้กับอาร์เรย์ในแถวที่ 1 และเครื่องหมายปีกกาชุดที่สามจะเป็นจะเป็นการกำหนดค่าให้กับอาร์เรย์ในแถวที่ 2

**3.ตัวแปรอาร์เรย์ชนิด 3 มิติ**

   อาร์เรย์หลายมิติสามารถมีได้ตั้งแต่ 3 มิติ 4 มิติ หรือหลายมิติ ซึ่งในรูปทื่ 8-18 แสดงอาร์เรย์ที่มีมิติ 3 มิติ   
ถึงแม้ว่าอาร์เรย์ 3 มิติ ที่แท้จริงจะเป็นดังในรูปที่ 8-18 แต่ในภาษา Cจะมองเป็นแบบอื่น คือ การสร้างอาร์เรย์ 3 มิติ จะสร้างจากอาร์เรย์ 2 มิติหลายอัน และอาร์เรย์ 2 มิติ ก็จะสร้างมาจากอาร์เรย์ 1 มิติ ฉะนั้นจะเรียกอาร์เรย์ 3 มิติ ว่าเป็นอาร์เรย์ของอาร์เรย์นั้นเอง ลักษณะที่ภาษา C มองกับอาร์เรย์ 3 มิติ  
**การประกาศและการกำหนดลักษณะของอาร์เรย์ 3 มิติ**   
  อาร์เรย์หลายมิติจะเหมือนกับอาร์เรย์ 1 มิติ ในการประกาศและกำหนดลักษณะก่อนการใช้ ซึ่งการประกาศและกำหนดลักษณะนั้นจะเป็นตัวบอกให้คอมไพเลอร์รู้ว่า อาร์เรย์ที่สร้างมีชื่ออะไร มีชนิดข้อมูลของแต่ละ Element เป็นอะไร และมีขนาดของแต่ละมิติเป็นเท่าไร ซึ่งตัวอย่างที่แสดงในรูปที่ 8-19 สามารถทำการประกาศและกำหนดลักษณะได้ดังนี้  
**int table [3] [5][4];**

**2.อาร์เรย์ของเรฟเฟอร์เรนซ์**

 String[] test = new String[3];



test[0] = "Hello"; ¬

test[1] = "Java"; ¬

test[2] = "World";

**การสร้างอาร์เรย์ของเรฟเฟอร์พร้อมกับกำหนดให้อ้างถึงวัตถุ**

String[] test = { "Hello", "Java", "World"};

StringBuffer[] sb = { new StringBuffer("Hello"),

new StringBuffer("Java"),

new StringBuffer("World") };

อาร์เรย์ของออบเจ็กต์

อาร์เรย์สามารถเก็บ reference ของ Object ได้โดยกําหนดให้อาเรย์เป็น Class นั้น ๆ ในตอน ประกาศอาร์เรย์มีรูปแบบดังนี้ className [] arrayName = new className[size];

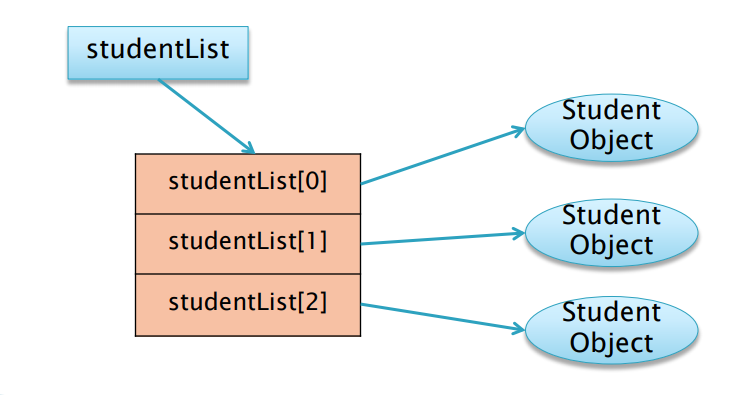
เช่น

Student [] studentList = new Student[10];

Student [] studentList = new Student[3];

studentList[0] = new Student();

studentList[1] = new Student();

studentList[2] แ= new Student();

**Class Arraylist**

ในภาษา Java มีกลุ่มคลาสที่เรียกว่า ตอลเล็กชั่น (Collection) ซึ่งออบเจ็กตืจากคลาสนี้สามารถใช้สะสมออบเจ็กต์ต่างๆไว้ได้ เช่น Class Arraylist ดีกว่า Array ที่สามารถเพิ่มสมาชิกได้ โดยไม่ต้องประกาศไว้ล่วงหน้า การใช้ Class ArrayList จะต้อง import java.util.ArrayList ตัวอย่างการสร้างออบเจ็กต์จาก Class ArrayList

ArrayList nameList = new ArrayList();

**Method ที่สำคัญของ ArrayList**

**add(int index,Object obj)**

ใชใส้ ออบเจ็กต์ลงในอาร์เรย์ที่ตําแหน่ง index

**remove(int index)**

นําออบเจ็กต์ตําแหน่ง index ออกจากอาร์เรย์

**get(int index)**

คืนค่าออบเจ็กต์ในตําแหน่ง index

**indexof(Object obj)**

คืนค่า index ของออบเจ็กต์ที่ระบุ

**Size**

สําหรับหาขนาดของ ArrayList

**2. การสร้างอาเรย์ขึ้นจากลิสต์**

มีอยู่หลายวิธีในการสร้างอาเรย์ แต่วิธีที่พื้นฐานที่สุดคือสร้างขึ้นมาจากลิสต์, ทูเพิล หรือเรนจ์ โดยใช้ np.array (ลิสต์)

aray = np.array(range(3,7))

araya = np.array([[1,2],[3,4]])

print(aray)

print(araya)

เท่านี้ก็จะได้อาเรย์ขึ้นมาตามที่ต้องการ ถ้าดูเผินๆจะเห็นว่าไม่ต่างอะไรจากลิสต์นัก แต่ความจริงแล้วต่างไปพอสมควร ที่อาจจะเห็นได้เป็นอย่างแรกก็คือเวลาที่สั่ง print จะออกมาเป็นแถวเป็นระเบียบร้อย

**3.ชนิดของข้อมูลในอาเรย์**

สามารถตรวจสอบชนิดของข้อมูลในอาเรย์ได้โดยดูที่แอตทริบิวต์ชื่อ dtype เช่นลองใช้อาเรย์ที่สร้างในตัวอย่างก่อน

araye = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])

arayu = np.array([[1,2],[3,4,5]])

print(araye.dtype) # ได้ int64

print(arayu.dtype) # ได้ object

จะเห็นว่า arayu ซึ่งสร้างขึ้นมาจากลิสต์ที่มีจำนวนสมาชิกไม่เท่ากันนั้นได้ข้อมูลประเภท object แทนที่จะเป็น int อย่างที่ควรเป็นส่วน araye นั้นกลายเป็นชนิด int ตามที่ควรจะเป็น โดยเลข 64 ใน int64 นี้บอกถึงขนาดของหน่วยความจำเป็นบิตที่ใช้ในการเก็บตัวเลข ปกติแล้วในภาษาไพธอนจำนวนเต็มจะไม่ได้แบ่งชนิดย่อย แต่ในบางภาษาเช่นภาษาซีจำนวนเต็มจะถูกแบ่งเป็นชนิดตามขนาดของหน่วยความจำที่ ใช้ปกติแล้วถ้าไม่ได้ระบุอะไรข้อมูลจำนวนเต็มจะถูกตั้งให้เป็น int64 ซึ่งเป็นขนาดใหญ่สุด นอกจาก int64 แล้วก็ยังมี int8 int16 int32 ซึ่งจะกินพื้นที่น้อยกว่า และ float เองก็มีแบ่งเช่นกันชนิดของข้อมูลสามารถกำหนดได้ตอนที่สร้างอาเรย์ขึ้นมา โดยเพิ่มคีย์เวิร์ด dtype เข้าไป

ari = np.array([1,2,3,4],dtype='int16')

ariy = np.array([1,2,3,4],dtype='float32')

print(ariy) # ได้ [ 1. 2. 3. 4.]

**4. การอ้างอิงถึงข้อมูลในอาเรย์**

เช่น เดียวกับลิสต์ อาเรย์ก็ใช้การเติมวงเล็บเหลี่ยม [ ] เพื่ออ้างอิงข้อมูล โดยเลขในกรณีสองมิตินั้น ตัวแรกคือดัชนีของแนวตั้ง (เลขแถว) และตัวหลังคือแนวนอน (เลขหลัก)

ariyu = np.array([[1,2,3],[4,5,6]])

print(ariyu[0][1]) # ได้ 2

print(ariyu[1][2]) # ได้ 6

แต่นอกจากนั้นแล้วยังสามารถเขียนโดยใช้เป็นคู่อันดับโดยมีจุลภาค , คั่นในวงเล็บเหลี่ยมตัวเดียวแทนการใส่วงเล็บเหลี่ยมสองอันได้ด้วย ซึ่งแบบนี้ลิสต์ไม่สามารถทำได้

print(ariyu[0,2]) # ได้ 3

print(ariyu[1,1]) # ได้ 5

การเข้าถึงสมาชิกทีละหลายตัวก็ทำได้ด้วยการใช้โคลอน : เช่นเดียวกับลิสต์ แต่สำหรันอาเรย์สองมิติแทนที่จะใช้วงเล็บเหลี่ยมวางต่อกันสองอันสามารถใช้จุลภาคแทนได้เช่นกัน

print(ariyu[1][:]) # ได้ [4 5 6]

print(ariyu[1,:]) # ได้ [4 5 6]

**5. การเขียนทับข้อมูลในอาเรย์**

เช่นเดียวกับลิสต์ อาเรย์ก็สามารถเขียนทับแก้ข้อมูลข้างในเมื่อไหร่ก็ได้ด้วยการใส่ค่าทับลงไป

aruy = np.array([[2,2,2],[2,2,2],[2,2,2]])

print(aruy)

aruy[1,1] = 3

print(aruy)

แต่ที่สะดวกกว่านั้นก็คือหากอยากแก้ค่าของหลายตัวในแถวหรือหลักเดียวกันทีเดียวก็สามารถทำได้

aruy[1,:] = 4

print(aruy)

aruy[:,2] = 5

print(aruy)

aruy[:,:] = 6

print(aruy)

จะเห็นว่าถ้าเขียน [:,:] จะเป็นการอ้างถึงข้อมูลทั้งหมด และทุกตัวจะถูกเปลี่ยนให้มีค่าเท่ากันหมด ลักษณะนี้เรียกว่าเป็นการกระจายค่า (broadcast) คือสามารถใช้เลขตัวเดียวมาแทนค่าเพื่อยัดใส่แทนสมาชิกหลายตัวในอาเรย์ได้ในทีเดียวที่น่าสนใจกว่านั้นหน่อยคือการกระจายค่าไม่ได้จำกัดอยู่แค่ตัวเลขเดี่ยว แต่ยังใช้กับอาเรย์ที่มีจำนวนมิติน้อยกว่าได้ด้วย เช่น

aruy[:,:] = np.array([7,8,9]) # หรือจะใช้ในรูปลิสต์ aruy[:,:] = [7,8,9] ก็ได้

print(aruy)

**6. การสร้างอาเรย์ด้วยฟังก์ชัน**  
 นอกจากจะสร้างจากลิสต์แล้วอาเรย์ก็ยังสร้างจากฟังก์ชันต่างๆที่มีอยู่ในมอดูล numpy เองได้ ในที่นี้ขอแนะนำฟังก์ชันที่ใช้บ่อย ได้แก่

|  |  |
| --- | --- |
| np.arange | สร้างอาเรย์หนึ่งมิติที่มีเลขเรียงกัน |
| np.linspace | สร้างอาเรย์หนึ่งมิติตามจำนวนที่กำหนดโดยเว้นช่วงเท่าๆกัน |
| np.ones | สร้างอาเรย์ที่มีแต่เลข 1 ตามขนาดที่กำหนด |
| np.zeros | สร้างอาเรย์ที่มีแต่เลข 0 ตามขนาดที่กำหนด |
| np.full | สร้างอาเรย์ที่มีแต่เลขอะไรก็ได้เลขเดียวตามที่กำหนด โดยมีขนาดตามที่กำหนด |
| np.empty | สร้างอาเรย์เปล่าๆ ตามขนาดที่กำหนด |
| np.identity | สร้างอาเรย์ที่เป็นเมทริกซ์เอกลักษณ์ (เมทริกซ์สองมิติที่มีค่าเป็น 1 เมื่อพิกัดแนวตั้งและนอนเท่ากัน นอกนั้นเป็น 0) |
| np.eye | เหมือนกับ np.identity แต่ขนาดแนวตั้งและนอนไม่ต้องเท่ากันก็ได้ |

ในการสร้างอาเรย์ที่เป็นตัวเลขเรียงกันใช้ np.arange หรือ np.linspace

print(np.arange(10))

print(np.arange(1.5,8,2)) # อาร์กิวเมนต์ที่ใช้เหมือนกับ range แต่ไม่จำเป็นต้องเป็นจำนวนจริง

print(np.linspace(4, 7, 6)) # (จุดเริ่ม, จุดปลาย, จำนวน)

np.linspace ถ้าใส่คีย์เวิร์ด endpoint=0 ไปจะหมายความว่าไม่รวมเลขจุดปลายด้วย และผลที่ได้จะต่างไปจากเดิมเลย

print(np.linspace(1.,4.,6)) # ได้ [ 1. 1.6 2.2 2.8 3.4 4. ]

print(np.linspace(1.,4.,6,endpoint=0)) # ได้ [ 1. 1.5 2. 2.5 3. 3.5]

**7. คุณสมบัติการถ่ายทอดของอาเรย์**

ขอทิ้งท้ายด้วยเรื่องสำคัญอีกอย่างหนึ่งเกี่ยวกับอาเรย์ที่ค่อนข้างสำคัญและจำเป็นต้องทำความเข้าใจไม่เช่นนั้นอาจก่อให้เกิดข้อผิดพลาดที่ไม่อาจค้นหาสาเหตุได้ นั่นคือการที่อาเรย์มีคุณสมบัติการถ่ายทอดอาเรย์ที่ถูกสร้างขึ้นมาถือเป็นออบเจ็กต์อันหนึ่ง ถ้าหากมีตัวแปรมารับด้วย = มันก็จะแทนออบเจ็กต์อาเรย์นั้น และหากมีตัวแปรมารับต่อตัวแปรนั้นด้วย = อีก จะกลายเป็นว่าตัวแปรใหม่นั้นแทนออบเจ็กต์อาเรย์ตัวเดียวกัน หากมีการเปลี่ยนแปลงค่าภายในก็จะมีการเปลี่ยนแปลงไปตามกัน

ตัวอย่าง

aime = np.array([1,2,3,4,5])

aiment = aime

print(aiment) # ได้ [1 2 3 4 5]

print(aime) # ได้ [1 2 3 4 5]

aiment[1] = 6

print(aiment) # ได้ [1 6 3 4 5]

print(aime) # ได้ [1 6 3 4 5]

จะเห็นว่า aiment = aime ทำให้ aiment กลายเป็นตัวแปรที่ใช้แทน aime พอมีการเปลี่ยนแปลงภายใน aiment (เช่นในที่นี้คือแก้ค่าสมาชิก [1] เป็น 6) aime ก็จะเปลี่ยนแปลงตาม คุณสมบัติข้อนี้สามารถใช้กับลิสต์ได้ผลในทำนองเดียวกัน จึงดูไม่ใช่เรื่องอะไรแปลกใหม่แต่นอกจากนี้แล้วอาเรย์ยังสามารถถ่ายทอดด้วยชิ้นส่วนที่ถูกตัดมาด้วย

aime = np.array([1,2,3,4,5])

aimez = aime[2:]

print(aimez) # ได้ [3 4 5]

print(aime) # ได้ [1 2 3 4 5]

aimez[1] = 8

print(aimez) # ได้ [3 8 5]

print(aime) # ได้ [1 2 3 8 5]

จะเห็นว่า aimez = aime[2:] ทำให้ aimez กลายเป็นตัวแทนของส่วนตั้งแต่ตำแหน่ง [2] ของ aime จากนั้นพอมีการแก้ค่าภายใน aimez ค่าภายใน aime ก็จะถูกเปลี่ยนแปลงไปด้วย คุณสมบัตินี้ไม่สามารถใช้กับลิสต์ได้

**8. สรุปข้อแตกต่างระหว่างอาเรย์กับลิสต์**

•อาเรย์จะต้องประกอบด้วยข้อมูลเพียงชนิดเดียวเท่านั้น

•อาเรย์ต้องมีจำนวนข้อมูลในแต่ละแถวเท่ากันหมด

•อาเรย์สามารถคำนวณทางคณิตศาสตร์ได้โดยตรง

•อาเรย์คำนวณได้เร็วกว่า

•อาเรย์มีวิธีการเข้าถึงข้อมูลภายในได้ยืดหยุ่นกว่า

•อาเรย์มีคุณสมบัติการถ่ายทอดภายในชิ้นส่วนประกอบ