Mahidol University

Wisdom of the Land

Mahidol University

Lecture 15: Numpy

Lecturers:

Aj. Jidapa Kraisangka

Aj. Akara Supratak

Aj. Tipajin Thaipisutikul

NumPy


```
import numpy as np
```

- เป็นหนึ่งใน Python library ที่ใช้สำหรับการประมวลผลข้อมูลอาเรย์หลายมิติ (n-D array)
- อาเรย์ (array) นิยมใช้สำหรับเก็บข้อมูลประเภทเดียวกันแบบเป็นลำดับได้
- การคำนวนทางคณิตศาสตร์จะทำได้เร็วมาก ควรใช้ของ NumPy ในการคำนวน ไม่ควรเขียนเอง

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

การสร้าง numpy array


```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

การสร้าง numpy array มิติต่างๆ

0-D Array

```
import numpy as np
arr = np.array(42)   # 0-D array
print(arr)
```

```
42
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

การสร้าง numpy array มิติต่างๆ

1-D Array

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])  # 1-D array
print(type(a))
print(a.shape)
print(a[0], a[1], a[2])  # index
```

จะให้ค่าจำนวน element ในแต่ละมิติของ array มาในรูปแบบของ tuple

```
[1 2 3 4 5]
1 2 3
<class 'numpy.ndarray'>
(5,)
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

2-D Array

```
import numpy as np
arr = np.array([[1,2,3], [4,5,6]])
print(arr)
print(arr[0,0], arr[0,1], arr[1,0], arr[-1,-1])  # index
print(arr.shape)
```

```
[[1 2 3]
 [4 5 6]]
1 2 4 6
(2, 3)
```

สร้าง numpy array

สามารถใช้คำสั่ง `np.arange([start,] stop, [step])` ในการสร้าง numpy array ได้

```
import numpy as np
a = np.arange(5)          # ระบุเฉพาะ stop จะเริ่มจาก 0 เสมอ
b = np.arange(5,10)       # ระบุ start และ stop
c = np.arange(5,10,2)     # ระบุ start, stop และ step
print(a)
print(b)
print(c)
```

```
[0 1 2 3 4]
[5 6 7 8 9]
[5 7 9]
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

Array Slicing


```
import numpy as np
a = np.array([
  [1,2,3,4],
  [5,6,7,8],
  [9,10,11,12]
])
print(a)
# ดึง 2-D array จาก a ที่แถวที่ 0 และ 1 หลักที่ 1 และ 2
b = a[:2, 1:3]
print(b)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[2 3]
 [6 7]]
```

Array Slicing


```
import numpy as np
a = np.array([
  [1,2,3,4],
  [5,6,7,8],
  [9,10,11,12]
])
print(a)
# ดึง 2-D array จาก a ที่แถวที่ 0 และ 1 หลักที่ 1 และ 2
b = a[:2, 1:3]
print(b)

b[0,0] = 77     # b[0,0] จะอ้างอิงถึงข้อมูลเดียวกันกับ a[0, 1]

print(a)
print(b)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[2 3]
 [6 7]]
[[ 1 77  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[77  3]
 [ 6  7]]
```

Array Slicing

การใช้ slicing จะไม่ทำให้มิติ (dimension) ของ numpy array ลดลง

```
import numpy as np
# 2D array of shape (3,4)
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print(a)

row_r1 = a[1, :]     # มิติลดลงถ้าเราใช้ตัวเลข integer
row_r2 = a[1:2, :]   # มิติเท่าเดิม
print(row_r1, row_r1.shape)
print(row_r2, row_r2.shape)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[5 6 7 8] (4,)
[[5 6 7 8]] (1, 4)
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

การใช้ slicing จะไม่ทำให้มิติ (dimension) ของ numpy array ลดลง

```
import numpy as np
# 2D array of shape (3,4)
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print(a)

col_r1 = a[:,1]      # มิติลดลงถ้าเราใช้ตัวเลข integer
col_r2 = a[:,1:2]    # มิติเท่าเดิม
print(col_r1, col_r1.shape)
print(col_r2, col_r2.shape)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[ 2  6 10] (3,)
[[ 2]
 [ 6]
 [10]] (3, 1)
```

Integer array indexing

เราสามารถเข้าถึง element โดยใช้ list ของ index ในแต่ละมิติได้

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])

# ใน 2D-array เราสามารถระบุตำแหน่งของ element
# โดนใช้ list ของ row index และ column index ได้
out = a[[0,1,2], [0,1,0]]
print(out)
print(out.shape)

# ตัวอย่างด้านบนจะให้ค่าเท่ากับ code ด้านล่างนี้
print(a[0,0],a[1,1],a[2,0])
```

```
[1 4 5]
(3,)
1 4 5
```

เราสามารถแก้ไขค่าใน column ต่าง ๆ ได้

```
import numpy as np
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
print(a)

# ระบุ array ของ column index ที่จะแก้ไข
b = np.array([0, 2, 0, 1])

# แสดงผล
print(a[np.arange(4), b])

# เพิ่มค่าให้กับ column เหล่านั้นไป 10
a[np.arange(4), b] += 10
print(a)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[ 1  6  7 11]
[[11  2  3]
 [ 4  5 16]
 [17  8  9]
 [10 21 12]]
```

Boolean array indexing

เราสามารถเข้าถึง element ตามเงื่อนไข (condition) ต่าง ๆ ได้

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
bool_idx = (a > 2)   # จะได้ boolean array ที่มีจำนวนมิติเท่ากัน
                     # โดยแต่ละ element เป็น True หรือ False
                     # ที่แสดงสถานะว่า element นั้นเป็นไปตาม
                     # เงื่อนไขที่กำหนดหรือไม่
print(bool_idx)

# เราสามารถเข้าถึง element ต่างโดยใช้ boolean array ได้
# โดยจะดึงมาเฉพาะ element ที่เป็น True เท่านั้น
print(a[bool_idx])
```

```
[[False False]
 [ True  True]
 [ True  True]]
[3 4 5 6]
```

Boolean array indexing

เราสามารถเข้าถึง element ตามเงื่อนไข (condition) ต่าง ๆ ได้

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
bool_idx = (a > 2)   # จะได้ boolean array ที่มีจำนวนมิติเท่ากัน
                     # โดยแต่ละ element เป็น True หรือ False
                     # ที่แสดงสถานะว่า element นั้นเป็นไปตาม
                     # เงื่อนไขที่กำหนดหรือไม่
print(bool_idx)

# เราสามารถเข้าถึง element ต่างโดยใช้ boolean array ได้
# โดยจะดึงมาเฉพาะ element ที่เป็น True เท่านั้น
print(a[bool_idx])
# เราสามารถทำตามด้านบนแบบย่อได้ดังนี้
print(a[a > 2])
```

```
[[False False]
 [ True  True]
 [ True  True]]
[3 4 5 6]
[3 4 5 6]
```

Data Types in NumPy

ประเภทของข้อมูลใน numpy array

- i - integer (`int, np.int32, np.int64`)
- b - boolean
- u - unsigned integer
- f - float (`float, np.float32, np.float64`)
- m - timedelta
- M - datetime
- S - string
- U - unicode string
- ...

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

Data Types in NumPy


```
import numpy as np

arr = np.array([1, 2, 3, 4])
print(arr.dtype)

arr = np.array(['apple', 'banana', 'cherry'])
print(arr.dtype)
#'U64' is a 64 character unicode. That's a normal string in Py3 numpy.
```

```
int64
<U6
```

เปลี่ยน dtype ใน numpy

เราสามารถใช้คำสั่ง `astype` ในการเปลี่ยนประเภทของข้อมูลบใน numpy array ได้

```
import numpy as np
arr = np.array([1.1, 2.1, 3.1])
print(arr)
print(arr.dtype)
newarr = arr.astype(int)
print(newarr)
print(newarr.dtype)
```

```
[1.1 2.1 3.1]
float64
[1 2 3]
int64
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

เปลี่ยน dtype ใน numpy

เราสามารถใช้คำสั่ง `astype` ในการเปลี่ยนประเภทของข้อมูลใน numpy array ได้

```
import numpy as np
arr = np.array([1.1, 2.1, 3.1])
print(arr)
print(arr.dtype)
newarr = arr.astype(int)
print(newarr)
print(newarr.dtype)
newarr = newarr.astype(float)
print(newarr)
print(newarr.dtype)
```

```
[1.1 2.1 3.1]
float64
[1 2 3]
int64
[1. 2. 3.]
float64
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

Math functions in NumPy


```
import numpy as np
arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])

out = np.add(arr1, arr2)
# arr1 + arr2 (element-wise)
print(out)
out = np.subtract(arr1, arr2)
# arr1 - arr2 (element-wise)
print(out)
out = np.multiply(arr1, arr2)
print(out)
out = np.divide(arr1, arr2)
print(out)
```

```
[30 32 34 36 38 40]
[-10 -10 -10 -10 -10 -10]
[200 231 264 299 336 375]
[0.5        0.52380952 0.54545455 0.56521739
0.58333333 0.6        ]
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

Math functions in NumPy


```
import numpy as np
arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])
arr3 = np.array([[1,2,3], [4,5,6]])

out = np.sum(arr1)
print(out)
out = np.sum(arr2)
print(out)
out = np.sum(arr1 - arr2)
print(out)
out = np.sum(arr3)
print(out)
```

```
75
135
-60
21
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

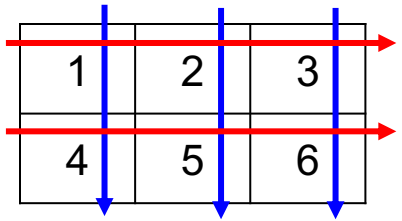
Math functions in NumPy


```
import numpy as np
arr = np.array([[1,2,3], [4,5,6]])

# หาผลรวมตามมิติแรก
out = np.sum(arr, axis=0)
print(out)

# หาผลรวมตามมิติที่สอง
out = np.sum(arr, axis=1)
print(out)
```

```
[5 7 9]
[ 6 15]
```



สามารถทำได้เหมือนกันกับ

`mean()` → **หาค่าเฉลี่ย**

`std()` → **หาค่าเบี่ยงเบนมาตรฐาน**

ดูเพิ่มเติมที่

https://www.w3schools.com/python/numpy/numpy_ufunc.asp

รับค่ามาใส่ 2D array โดยใช้ `input()`

```
import numpy as np

# Number of rows and columns
n_rows, n_cols = input().split(' ')
n_rows, n_cols = int(n_rows), int(n_cols)
rows = []
for r in range(n_rows):
    values = input().strip().split(' ')
    rows.append(np.array(values))
input_matrix = np.array(rows, dtype=np.int8)
print(input_matrix, type(input_matrix), input_matrix.dtype)
```

```
2 3
1 2 3
4 5 6
[[1 2 3]
 [4 5 6]] <class 'numpy.ndarray'> int8
```

ตัวอย่าง

อ่านค่าจากไฟล์ sample_matrix.txt มาใส่ 2D array

```
import numpy as np
with open('sample_matrix.txt', 'r') as f:
    lines = f.readlines()
    n_rows, n_cols = lines[0].split(' ')
    n_rows, n_cols = int(n_rows), int(n_cols)
    rows = []
    for l in lines[1:]:
        values = l.split(' ')
        rows.append(values)
input_matrix = np.array(rows, dtype=np.int8)
print(n_rows, n_cols, input_matrix.shape)
print(input_matrix)
print(type(input_matrix), input_matrix.dtype)
```

sample_matrix.txt

```
4 5
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
-5 -4 -3 -2 -1
```

```
4 5 (4, 5)
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [-5 -4 -3 -2 -1]]
<class 'numpy.ndarray'> int8
```

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu

ตัวอย่าง

แสดงค่าใน 2D array ที่มีค่ามากกว่า 10 และคำนวนหาผมรวมของค่าเหล่านั้น

```
import numpy as np
x = np.array([[0, 10, 20], [20, 30, 40]])
print(x)

# แสดงผล
print(x[x>10])

# หาผลรวม
print(np.sum(x[x>10]))
```

```
[[ 0 10 20]
 [20 30 40]]
[20 20 30 40]
110
```

นับจำนวนตัวเลขใน array

```
import numpy as np

x = np.array([1,1,1,2,2,2,5,25,1,1])
unique, counts = np.unique(x, return_counts=True)

print np.asarray((unique, counts)).T
```

```
[[ 1  2  5 25]
 [ 5  3  1  1]]
[[ 1  5]
 [ 2  3]
 [ 5  1]
 [25  1]]
```

นับจำนวนตัวเลขใน array

อ่านเพิ่มเติมได้จากลิ้งนี้
https://numpy.org/doc/stable/reference/generated/numpy.unique.html?highlight=unique#numpy.unique

```
import numpy as np

x = np.array([1,1,1,2,2,2,5,25,1,1])
unique, counts = np.unique(x, return_counts=True)
for v, c in zip(unique, counts):
  print(f'{v}: {c}')
```

```
1: 5
2: 3
5: 1
25: 1
```

อ่านเพิ่มเติมเกี่ยวกับ NumPy

Reference: https://numpy.org/doc/stable/reference/

Exercise: https://www.w3resource.com/python-exercises/numpy/

Contact: [jidapa.kra, dolvara.gun, akara.sup, tipajin.tha]@mahidol.edu