ลิสต์แบบหลายมิติ และ การใช้ไลบรารี (Part1 v.2)

ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย
๒๕๕๘

Topics

- **ส่วนที่ 1 ลิสต์แบบหลายมิติ**
- ลิสต์หลายมิติใน Python
- ไลบรารี NumPy
  - การสร้างอาร์เรย์
  - การเข้าถึง
  - การใช้งานทางคณิตศาสตร์ (element-wise)
  - การใช้งานด้วยฟังก์ชั่นเฉพาะ เช่น sum, mean, max, min
  - การคูณเมทริกซ์ (dot operator)
  - enumerate vs. ndenumerate
  - การเขียนอ่าน text file
- Application1 kNN
  - โจทย์ Iris
  - นำผลลัพธ์มา plot Histogram

Topics (ต่อ)

- **ส่วนที่ 2 การใช้ไลบรารีอื่นนอกจาก NumPy**
- ไลบรารีอื่นๆ
  - ความแตกต่างระหว่าง from vs. import
  - ไลบรารี Matplotlib กับการวาดกราฟ
  - ไลบรารี Scipy กับการจัดการรูปภาพ

- Application2: Image Convolution
- Application3: Face Detection
- Application4: Linear Transformation

ส่วนที่ 1 ลิสต์แบบหลายมิติ

ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย
๒๕๕๘

ลิสต์ใน Python

```
x = [1,2,3,4,5,6,7,8,9,10]
y = [1]*5 + [2]*5
sum = [0]*len(x)
for i in range(len(x)):
    sum[i] = x[i] + y[i]
```

ลองทำ print(x+y)

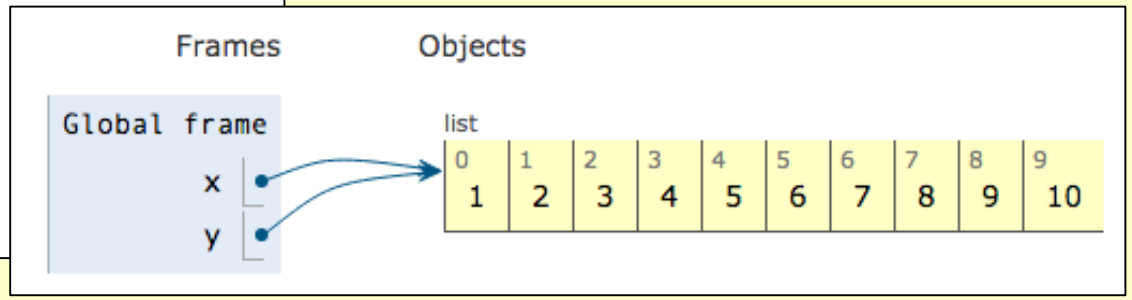
```
>>> print(sum)
[2, 3, 4, 5, 6, 8, 9, 10, 11, 12]
>>> print(sum[0])
2
>>> print(sum[-1])
12
>>> print(sum[8:10])
[11, 12]
```

คำสั่งน่ารู้

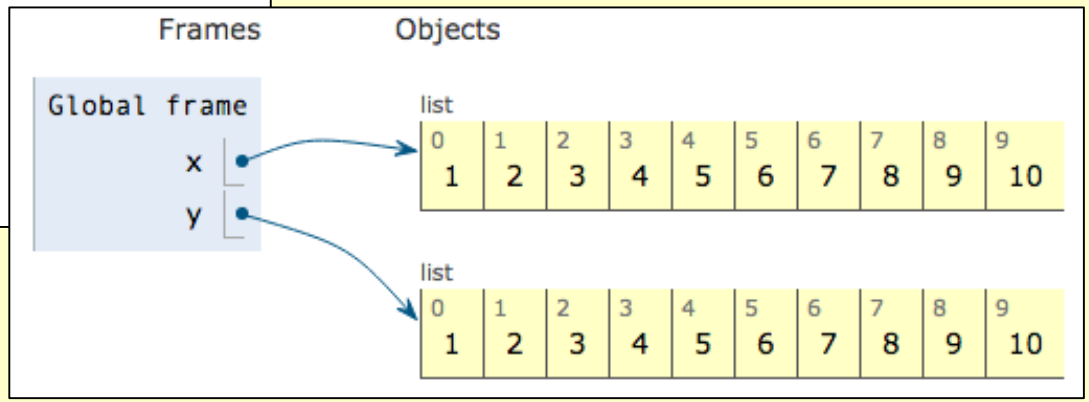
```
x.append(e)
x.insert(i, e)
x.pop(i)
x.remove(e)
x.count(e)
x.sort()
x.reverse()
```

ลิสต์ใน Python (cont.)

```
>>> x = [1,2,3,4,5,6,7,8,9,10]
>>> y = x
>>> print(x == y)
True
>>> print(x is y)
True
```

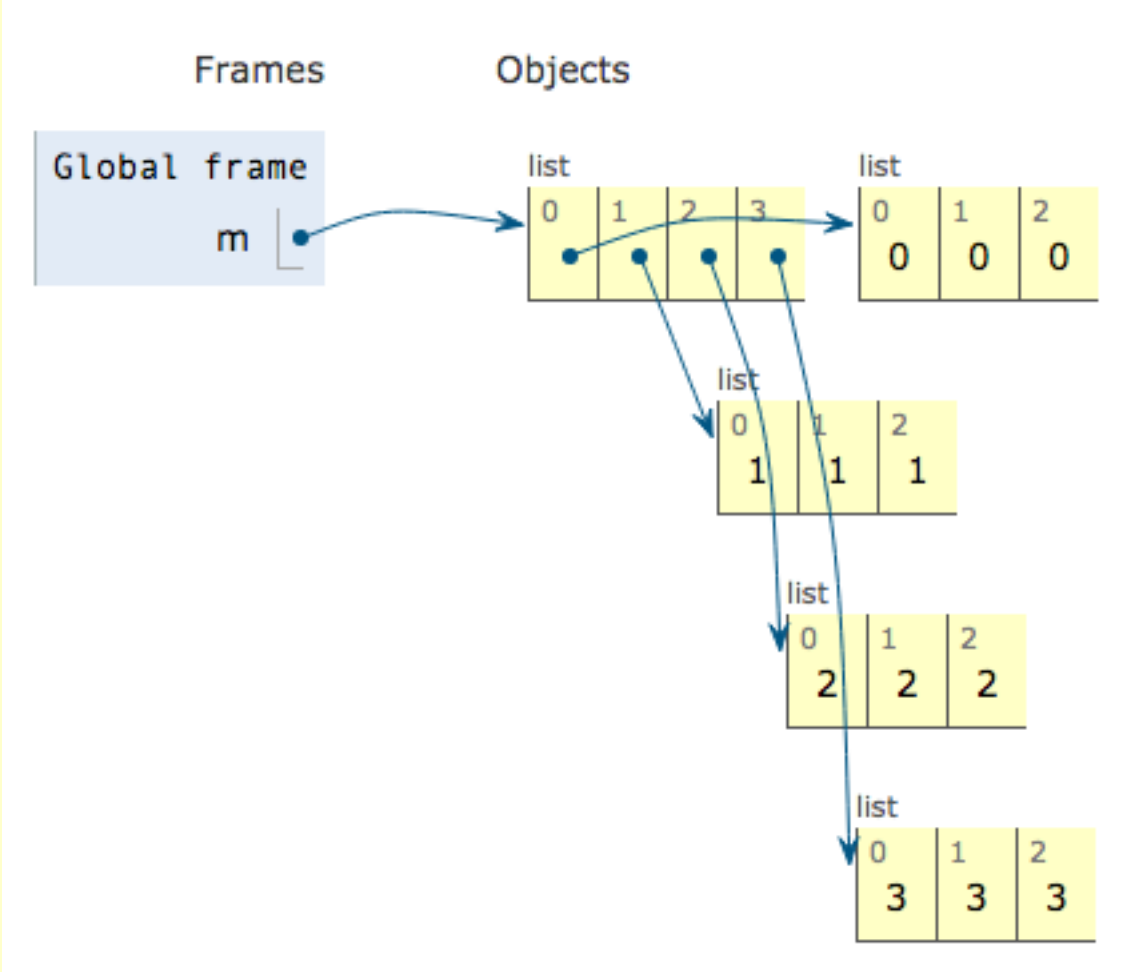



```
>>> x = [1,2,3,4,5,6,7,8,9,10]
>>> y = list(x)
>>> print(x == y)
True
>>> print(x is y)
False
```



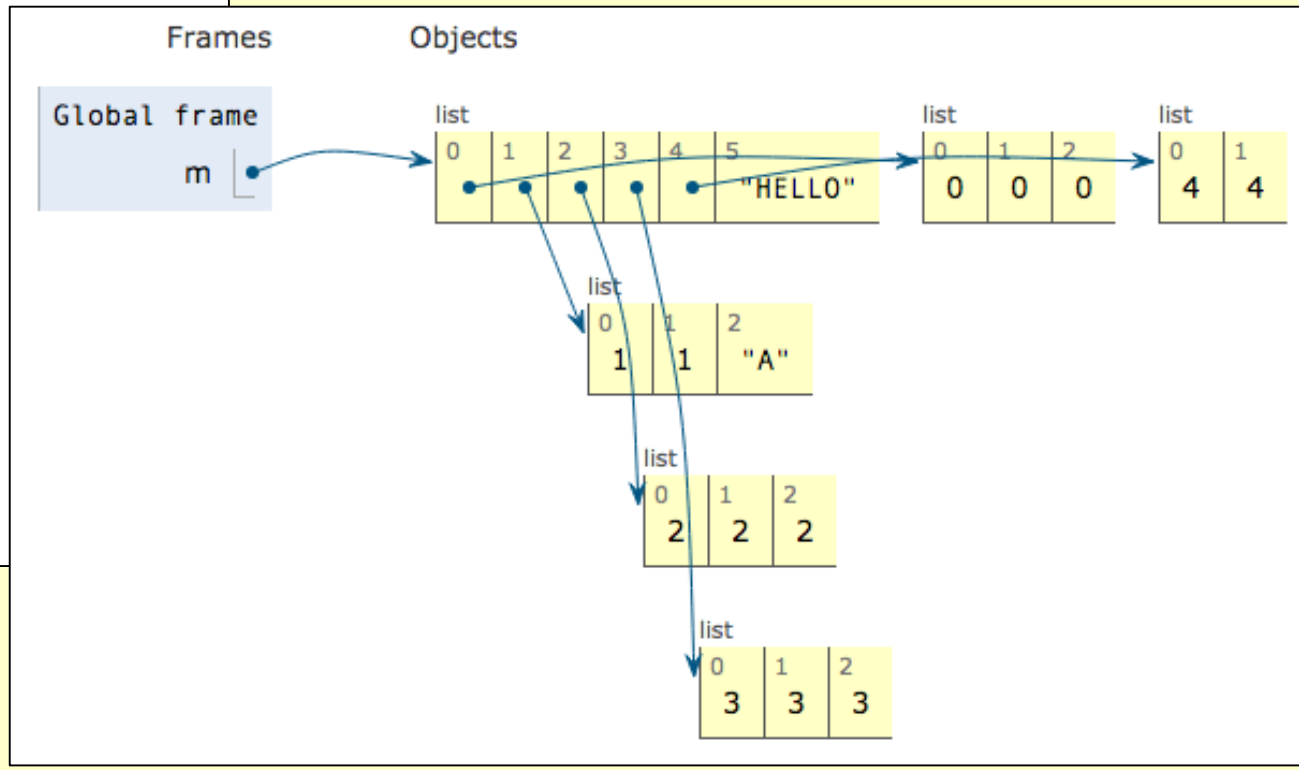
ลิสต์ของลิสต์ใน Python

```
m = [[0, 0, 0],
     [1, 1, 1],
     [2, 2, 2],
     [3, 3, 3]]
>>> len(m)
4
>>> m[1]
[1, 1, 1]
>>> m[1][:]
[1, 1, 1]
>>> m[1][0]
1
>>> len(m[1])
3
```



ลิสต์ของลิสต์ใน Python (cont.)

```
m = [[0, 0, 0],
     [1, 1, 1],
     [2, 2, 2],
     [3, 3, 3]]
m.append([4,4])
m.append("HELLO")
m[1][2] = 'A'
```




```
>>> m
[[0, 0, 0], [1, 1, 'A'], [2, 2, 2], [3, 3, 3], [4, 4], 'HELLO']
```

ลองเขียนดู : Matrix Transpose

เขียนโปรแกรมรับค่าเมตริกจากผู้ใช้ขนาด 3 * 3 โดยรับค่าเมตริกทีละแถว จากนั้นให้คำตอบเป็น Matrix Transpose

```
Input matrix row1: 1 2 3
Input matrix row2: 4 5 6
Input matrix row3: 7 8 9

Matrix: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Matrix.T: [[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```

Numerical Python (NumPy)

- อาร์เรย์หรือลิสต์ใน Python ไม่สามารถคำนวณ arithmetic operations (+,-,*,/)

```
>>> a = [1,3,5,7,9]
>>> b = [3,5,6,7,9]
>>> c = a + b
>>> print c
[1, 3, 5, 7, 9, 3, 5, 6, 7, 9]
```

ผลลัพธ์ผิด!

- ดังนั้นจึงต้องการไลบรารี Numerical Python (NumPy) เป็นลิสต์ที่มีความสามารถทางคณิตศาสตร์เพิ่มขึ้นมา (อีกมาก!!!)

```
import numpy
```

NumPy คืออะไร

- NumPy เป็นไลบรารีหนึ่งของไพธอน โดยเป็นไลบรารีหลักที่ใช้ในการคำนวณทางวิทยาศาสตร์
- NumPy มากับโครงสร้างข้อมูลที่เรียกว่าอาเรย์หลายมิติ (ndarray) และมีฟังก์ชั่นพื้นฐานที่ใช้ในการจัดการและวิเคราะห์ข้อมูลอาเรย์เหล่านี้
  - เช่น การคำนวณทางคณิตศาสตร์ การปรับมิติของอาเรย์ การทำ discrete Fourier transform การคำนวณ linear algebra การคำนวณทางสถิติพื้นฐาน เป็นต้น
- NumPy ถูกพัฒนาด้วยภาษา C ดังนั้นจึงสามารถประมวลผลได้เร็ว ซึ่งเป็นส่วนผสมของไลบรารี NumArray และ Numeric ในอดีต (ปัจจุบันใช้แต่ NumPy)

คุณสมบัติของอาเรย์ใน NumPy

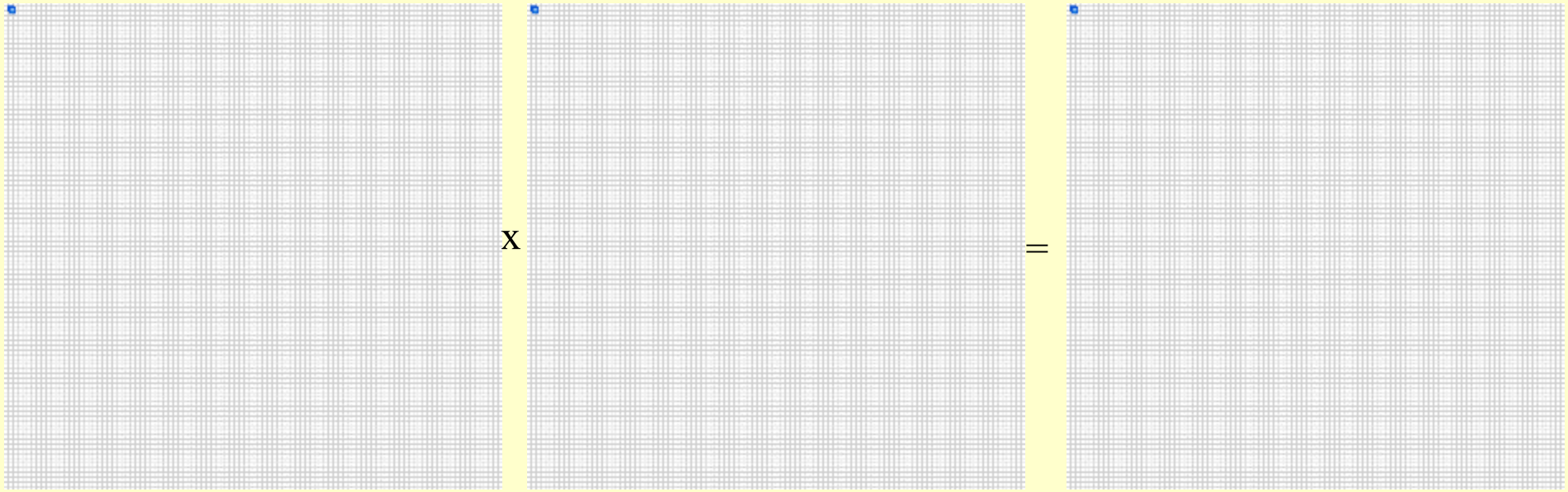
- สร้างแล้วมีขนาดตายตัว เปลี่ยนแปลงไม่ได้ ไม่เหมือน ลิสต์ที่มากับภาษาไพธอน
- ค่าในอาเรย์หนึ่งๆ ต้องเป็นประเภทเดียวกันทั้งหมด (ยกเว้นอาเรย์ของออบเจค ยังไม่ได้เรียนออบเจค)
- มีฟังก์ชั่นพื้นฐานที่ใช้ในการจัดการและวิเคราะห์ข้อมูลอาเรย์ ที่มีประสิทธิภาพสูง

เมทริกซ์ใน NumPy

- อาเรย์สองมิติใน NumPy เรียกว่าเมทริกซ์
- จำนวนของมิติของอาเรย์ใน NumPy เรียกว่าแร้งค์ (rank)
- รูปร่าง (shape) ของอาเรย์ถูกแสดงโดย tuple โดยบอกขนาดของอาเรย์ในแต่ละมิติ

ตัวอย่างประสิทธิภาพของ NumPy

มีอาเรย์ 2 มิติ ขนาดเท่ากัน 10,000 x 10,000



ถ้าต้องการอาเรย์ c โดยที่ $c_{ij} = a_{ij} * b_{ij}$ ต้องทำอย่างไร

หา $c_{ij} = a_{ij} * b_{ij}$ แบบที่ 1 ใช้ลิสต์ (ไม่ใช้ NumPy)

```
import time
test_size = 10000
t1 = time.time()
Matrix_a = [[2 for x in range(test_size)] for x in range(test_size)]
Matrix_b = [[3 for x in range(test_size)] for x in range(test_size)]
Matrix_c = [[0 for x in range(test_size)] for x in range(test_size)]
t2 = time.time()
print("time used for list initialization = ", t2-t1)

for i in range(test_size):
    for j in range(test_size):
        Matrix_c[i][j] = Matrix_a[i][j] * Matrix_b[i][j]
t3 = time.time()
print("time used for for loop = ", t3-t2)
```

```
time used for list initialization = 15.21 seconds
time used for for loop = 32.69 seconds
```

หา $c_{ij} = a_{ij} * b_{ij}$ แบบที่ 2 ใช้อาเรย์ใน NumPy

```
import numpy as np
import time
test_size = 10000
t1 = time.time()
Matrix_a = 2 * np.ones(shape=(10000,10000))
Matrix_b = 3 * np.ones(shape=(10000,10000))
t2 = time.time()
print("time used for Matrices initialization = ", t2-t1)

Matrix_c = Matrix_a * Matrix_b
t3 = time.time()
print("time used for Matrix calculation = ", t3-t2)
```

```
time used for Matrices initialization =  1.26
time used for Matrix calculation =  0.37
```

การสร้างอาร์เรย์ใน NumPy (1 มิติ)

```
# as vectors from lists
>>> import numpy as np
>>> a = np.array([1,3,5,7,9])
>>> b = np.array([3,5,6,7,9])
>>> c = a + b
>>> c
array([ 4,  8, 11, 14, 18])
>>> print(c)
[ 4 8 11 14 18]
>>> type(c)
(<type 'numpy.ndarray'>)
>>> c.shape
(5,)
```

การสร้างอาร์เรย์ใน NumPy (2 มิติ)

```
>>> M = np.array([[1, 2, 3], [3, 6, 9], [2, 4, 6]])
>>> print(M)
[[1 2 3]
 [3 6 9]
 [2 4 6]]
>>> M.shape
(3, 3)
>>> print(M.dtype)  # get type of an array
Int64
```

```
#only one type
>>> M[0,0] = 'A'
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    M[0,0] = 'A'
ValueError: invalid literal for int() with base 10: 'A'
```

การสร้างอาเรย์ใน NumPy ด้วยฟังก์ชัน

```
import numpy as np

x = np.zeros((2, 3))# สร้างอาเรย์ 2 มิติ 2 บรรทัด 3 คอลัมน์ ค่าเริ่มต้นทุกช่องเป็น 0

y = np.ones((2, 3)) สร้างอาเรย์ 2 มิติ 2 บรรทัด 3 คอลัมน์ ค่าเริ่มต้นทุกช่องเป็น 1

z1 = np.arange(10) # สร้างอาเรย์ 1 มิติที่มีค่าเริ่มจาก 0 และเพิ่มค่าที่ละ 1

z2 = np.arange(2,10,dtype=np.float)# สร้างอาเรย์ 1 มิติที่มีค่าเริ่มจาก 2.0 ถึง 9.0 เป็นเลขทศนิยม และเพิ่มค่าที่ละ 1

z3 = np.arange(2,3,0.1)
```

การสร้างอาเรย์ใน NumPy ด้วยฟังก์ชัน (ต่อ)

```
import numpy as np

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

y = np.zeros_like(x)

y = np.ones_like(x)

z = np.identity(4,dtype=float)
```

การอ้างอิงข้อมูลใน NumPy

```
>>> print(M)
[[1 2 3]
 [3 6 9]
 [2 4 6]]
>>> print(M[0])  # this is just like a list of lists
[1 2 3]
>>> print(M[1, 2])  # comma separated indices
9
>>> print(M[1, 1:3])  # and slices
[6 9]
```

```
>>> M[1, 2] = 7
>>> print(M)
[[1 2 3]
 [3 6 7]
 [2 4 6]]
>>> M[:, 0] = [0, 9, 8]
>>> print(M)
[[0 2 3]
 [9 6 7]
 [8 4 6]]
```

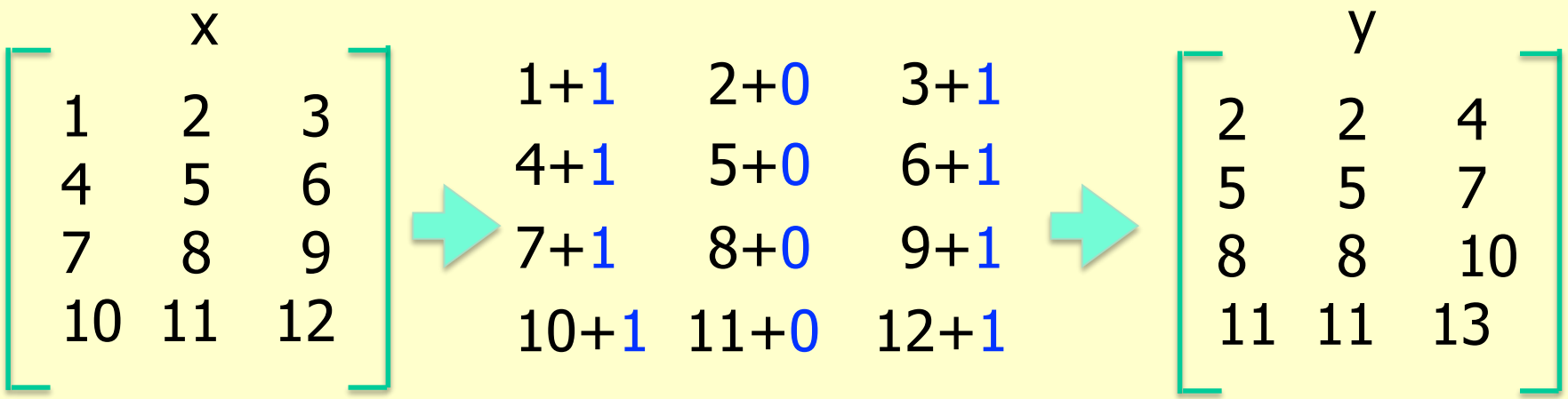
การบวก ลบ คูณ หาร อาเรย์ (ค่าต่อค่า)

```
import numpy as np
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

print(x+y)
print(np.add(x,y))

print(x-y)
print(np.subtract(x,y))

print(x*y)     # * element wise multiplication
print(np.multiply(x,y))

print(x/y)
print(np.divide(x,y))

print(np.sqrt(x))
```

บวก ลบ คูณ หาร ค่าต่อค่า อาเรย์สองตัวต้องมีรูปร่างเดียวกัน

ถ้าอาเรย์มีรูปร่างไม่เหมือนกัน จะทำบวก ลบ คูณ หาร อย่างไร??

เพิ่มค่าในอาเรย์โดยใช้ค่าจากเวคเตอร์ (แบบลูป)




```
import numpy as np
x = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
v = np.array([1,0,1])
y = np.empty_like(x) #สร้างเมทริกซ์ว่าง ๆที่มีรูปร่างแบบ x
for i in range(4):
    y[i,:] = x[i, :] + v
print(y)
```

บอร์ดคาสติ้ง (Broadcasting)

- บอร์ดคาสติ้งใช้ในการอธิบายวิธีการทำงานของโอเปอร์เรเตอร์ทางคณิตศาสตร์ระหว่างอะเรย์ที่มีจำนวนมิติ รูปร่าง ขนาด ต่างกันอย่างไร
- โดยทั่วไปอาเรย์ที่มีขนาดเล็กกว่าจะถูกบอร์ดคาสไปยังอาเรย์ที่มีขนาดใหญ่กว่า

```
x = np.array([1,2,3])
x = x + [1]
print(x)
```

```
x = np.array([[1,2,3],[4,5,6]])
x[0] = x[0] + [1]
print(x)
x[:,0] = x[:,0] + [1]
print(x)
```


http://cs231n.github.io/python-numpy-tutorial/#numpy

ตัวอย่างบอร์ดคาสติ้ง

```
# Good example
import numpy as np
x = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
v = np.array([1,0,1])
y = x + v
print(y)
```

```
# Error example
import numpy as np
x = np.array([1,2,3],float)
y = np.array([4,5], float)
z = x + y
print(z)
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: operands could not be broadcast together with shapes (3,) (2,)
```

ตย ฟังก์ชันทางคณิตศาสตร์ที่น่าสนใจอื่นๆใน NumPy

abs, sign, sqrt, log, log10, exp, sin, cos, tan, arcsin, arccos, arctan, sinh, cosh, tanh, arcsinh, arccosh, และ arctanh

ทำงานเป็นค่าต่อค่า (element-wise) เหมือน บวก ลบ คูณ หาร

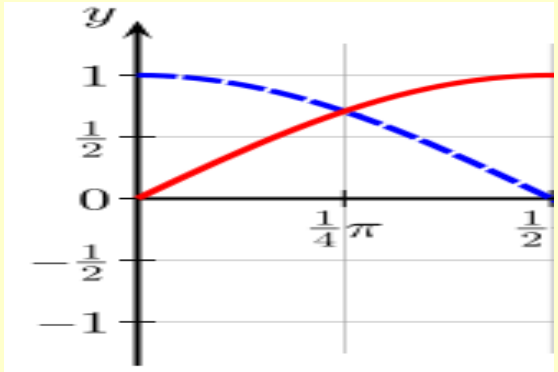
```
import numpy as np
a = np.array([1, 4, 9], float)
np.sqrt(a)
```

CH08_1

- การหาอนุพันธ์ (derivative) ทำได้ดังสมการด้านล่าง ซึ่งสามารถใช้ในการหาทิศทางการเปลี่ยนแปลงของฟังก์ชัน โดยหากผลอนุพันธ์เป็นบวก หมายถึงฟังก์ชันกำลังเพิ่มขึ้น และหากผลอนุพันธ์เป็นลบ หมายถึงฟังก์ชันกำลัง

$$f'(x_i) = \frac{f(x_i + \Delta x) - f(x_i - \Delta x)}{2\,\Delta x}$$

- จงหาอนุพันธ์และคำนวณผลรวมของทิศทางการเปลี่ยนแปลง (เครื่องหมาย) ของฟังก์ชัน sin และ ประมวลผลเช่นเดียวกันฟังก์ชัน cos
- กำหนดให้ $x_i = [0, \pi/2]$ และกำหนดให้ $\Delta x = 0.1$
- ไม่ต้องรวมทุก x_i เริ่มต้นและสิ้นสุด (Hint: x_i มี 15 ค่า)
- **ห้ามใช้ loop!**



CH08_2

- การถดถอยโลจิสติก (logistic regression) สามารถใช้ทำนาย (predict) ความน่าจะเป็น (probability) จากตัวแปรทำนาย (predictors)
- จงใช้สมการถดถอยด้านล่าง เพื่อ**ทำนาย**ความน่าจะเป็นในการสอบผ่านของนิสิต 5 คน เมื่อกำหนดให้มีตัวแปรทำนาย 2 ตัวได้แก่ คะแนนสอบกลางภาค (score) และ เกรด (GPA)
- นิสิตจะมีโอกาสผ่าน (True) เมื่อ p > 0.5
- **ห้ามใช้ loop! ให้ใช้ numpy อาร์เรย์ 2 มิติ ชื่อ "data" ในการเก็บข้อมูลนิสิตเท่านั้น!!!**

	Score	GPA
1	15	3.78
2	29	2.00
3	10	2.50
4	25	2.85
5	30	3.96

$$logit(p_i) = -3.98 + 0.2 * score_i + 0.5 * GPA_i$$

$$p(x_i) = \frac{1}{1 + e^{-logit(p_i)}}$$

การคูณเมทริกซ์ (dot operator)

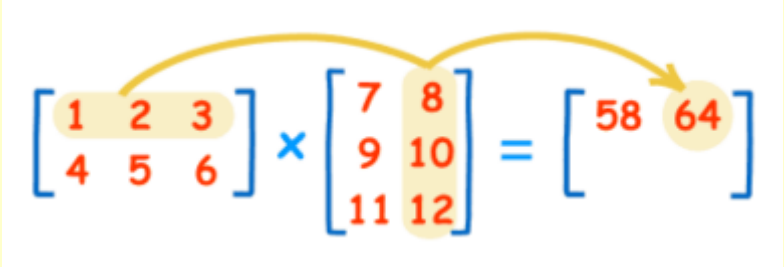


Figure source: https://www.mathsisfun.com/algebra/matrix-multiplying.html

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

การคูณเมทริกซ์ (dot operator) (CH08_3)

ร้านขายอาหารตามสั่งมีราคาอาหารคือ
ข้าวแกง 25 บาท
ข้าวผัด 30 บาท
สุกี้ทะเล 45 บาท
ในสัปดาห์ที่ผ่านมาที่ร้านขายอาหารได้ตามนี้

	จันทร์	อังคาร	พุธ	พฤหัส	ศุกร์
ข้าวแกง	75	120	70	90	80
ข้าวผัด	80	90	100	70	50
สุกี้ทะเล	50	45	70	65	50

ในสัปดาห์ที่ผ่านมาร้านอาหารมีรายได้เท่าไหร่

ฟังก์ชัน sum() ใน NumPy

	จันทร์	อังคาร	พุธ	พฤหัส	ศุกร์
ข้าวแกง	75	120	70	90	80
ข้าวผัด	80	90	100	70	50
สุกี้ทะเล	50	45	70	65	50

โคัดจากหน้าที่แล้ว

```
import numpy as np
x = np.array([25,30,45])
y =
np.array([[75,120,70,90,80],[80,90,100,70,50],[50,45
,70,65,50]])
print(np.sum(y))
print(np.sum(y, axis=0)) # บวกแนวตั้ง
print(np.sum(y, axis=1)) # บวกแนวนอน
```

จำนวนจานทั้งหมดที่ขายได้แต่ละวัน

อาหารแต่ละประเภทที่ขายได้ใน 1 สัปดาห์

ฟังก์ชัน mean(), std(), max(), argmax()ใน NumPy

	จันทร์	อังคาร	พุธ	พฤหัส	ศุกร์
ข้าวแกง	75	120	70	90	80
ข้าวผัด	80	90	100	70	50
สุกี้ทะเล	50	45	70	65	50

โคดจากหน้าที่แล้ว

```
import numpy as np
x = np.array([25,30,45])
y =
np.array([[75,120,70,90,80],[80,90,100,70,50],[50,45
,70,65,50]])
```

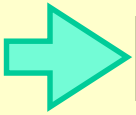
ประเภทของอาหารที่ขายได้โดยเฉลี่ยต่อวัน

```
print(np.mean(y, axis=1)) # ค่าเฉลี่ยแนวนอน
print(np.std(y, axis=1))
print(np.max(y, axis=1))# 120, 100, 70
print(np.argmax(y, axis=1)) # 1, 2, 2
```

เมทริกซ์ทรานสโพส (matrix transpose)

```
import numpy as np
y =
np.array([[75,120,70,90,80],[80,90,100,70,50],[50,45,70
,65,50]])
print(y)
print(y.T)
```

	จันทร์	อังคาร	พุธ	พฤหัส	ศุกร์
ข้าวแกง	75	120	70	90	80
ข้าวผัด	80	90	100	70	50
สุกี้ทะเล	50	45	70	65	50



	ข้าวแกง	ข้าวผัด	สุกี้ทะเล ทะเล
จันทร์	75	80	50
อังคาร	120	90	45
พุธ	70	100	70
พฤหัส	90	70	65
ศุกร์	80	50	50

ลิสต์อินูเมอเรต (enumerate)

```
>>years = ['Freshy','Sophomore','Junior','Senior']
>>list(enumerate(years))
[(0, 'Freshy'), (1, 'Sophomore'), (2, 'Junior'), (3, 'Senior')]
>>list(enumerate(years, start=1))
[(1, 'Freshy'), (2, 'Sophomore'), (3, 'Junior'), (4, 'Senior')]
```

```
for i, year in enumerate(years):
    print(i, year)
```

```
0 Freshy
1 Sophomore
2 Junior
3 Senior
```

ได้ผลออกมาเป็นลิสต์ของ tuple โดยมี index ของตำแหน่งข้อมูล และ ค่าของข้อมูล

เอ็นดีอินูเมอเรต (ndenumerate)

```
import numpy as np
a = np.array([[1,2],[3,4]])
for index,x in np.ndenumerate(a):
    print(index,x)
```

```
(0, 0) 1
(0, 1) 2
(1, 0) 3
(1, 1) 4
```

ได้ผลออกมาเป็น

- ลิสต์ของ tuple ลำดับข้อมูลในอาร์เรย์
- และ ค่าของข้อมูล

การอ่านข้อมูลจากไฟล์เข้าอาเรย์

ตัวอย่างไฟล์ข้อมูล ArrayData.csv

```
# this is a header
# this file is generated on Oct 7, 2015
# it is just an example file for NumPy I/O
1;1;1;1;1;1;1;1;1;1
1;2;1;2;1;2;1;2;1;2;1
2;2;2;2;2;2;2;2;2;2;2
3;3;3;3;3;3;3;3;3;3;3
4;5;6;6;7;8;9;5;1;3;2
```

การอ่านข้อมูลจากเท็กซ์ไฟล์เข้าอาเรย์ (ต่อ)

ใช้ **loadtxt** อ่านข้อมูลทั้งหมด **(ทุกคอลัมน์)**

```
import numpy as np
data = np.loadtxt("ArrayData.csv",skiprows=3, \
         delimiter=";")
print(data)
```

การอ่านไฟล์เลือกเอามาเฉพาะบางคอลัมน์

```
import numpy as np
data = np.loadtxt("ArrayData.csv",skiprows=3, \
         delimiter=";",usecols=(0,1,2))
print(data)
```

การเขียนข้อมูลอาเรย์ไปเท็กซ์ไฟล์

ใช้ **savetxt** เขียนข้อมูลอาร์เรยใน **NumPy** ใส่ในไฟล์

```
import numpy as np

Matrix_a = 2 * np.ones(shape=(10000,10000))
np.savetxt("Matrix_a.csv", Matrix_a, fmt='%.1e',\
    newline="\n", delimiter=";")
```

ลองเขียนดู (CH08_04)

อ่านไฟล์คะแนนเก็บของนิสิตและทำการคำนวณหาคะแนนรวมโดยใช้ loadtxt() และ savetxt()

```
# scores of students in IT521 class
# year: 2557/1
STUDENT_ID,hw1,hw2,quiz1,exam1,quiz2,exam2
5622770071,4,4,2,28,1,38
5622770790,5,3,1,30,2,28
5622771079,2,4,3,12,3,34
5622771152,3,5,4,13,4,23
```

ผลลัพธ์

```
77.00
69.00
58.00
52.00
```

ส่วนที่ 2 การใช้ไลบรารีอื่นนอกจาก NumPy

ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย
๒๕๕๘

ความแตกต่างระหว่าง from vs import

- import และ from เป็นการเรียกใช้งานไลบรารี (module) แต่ต่างกันตรงที่
  - import เมื่อเรียกใช้งาน ต้องมีชื่อ module นำหน้า
  - from เมื่อเรียกใช้งาน ไม่ต้องมีชื่อ module นำหน้า

```
import random
n = random.randint(0, 9)
```

```
from random import *
n = randint(0, 9)
```

- ถึงแม้ว่า from จะใช้งานได้สะดวกกว่า แต่มีข้อเสีย
  - ทำให้ไม่สามารถรู้ได้จากโค้ดว่าคำสั่งใดเรียกใช้ไลบรารีอื่น
  - เกิดความกำกวมในกรณีที่คำสั่งเดียวอยู่ได้มากกว่า 1 module
    - open() อยู่ในทั้ง os, urlib

```
import os
import urlib
urlib.open(foo)
```

```
from os import *
from urlib import *
open(foo)
```

ไลบรารีอื่นๆ

- **matplotlib**

เป็นไลบรารีหลักที่ใช้ในการวาดกราฟ 2 มิติ โดยให้คุณภาพภาพที่มีความละเอียดสูง มีลักษณะกราฟมากมายที่เหมาะกับการแสดงผลทางวิศวกรรม

- **PIL (Python Imaging Library)**

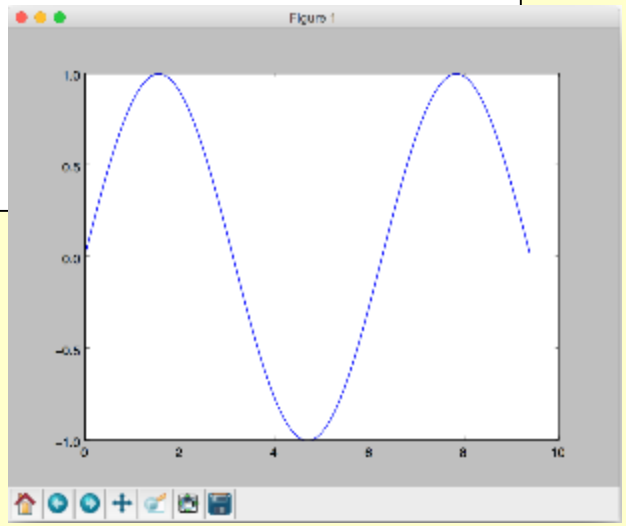
เป็นไลบรารีหลักที่ใช้ในการประมวลผลรูปภาพกราฟิกส์ (image processing) ในไพธอน โดยรองรับไฟล์ภาพที่มีรูปแบบ (format) ที่หลากหลาย

- **SciPy**

เป็นชุดของ open-source ซอฟต์แวร์ที่มีชุดของไลบรารีที่ช่วยในการคำนวณทางคณิตศาสตร์ วิทยาศาสตร์ และวิศวกรรมศาสตร์ NumPy และ matplotlib เป็นตัวอย่างของไลบรารีที่เป็นส่วนหนึ่งของ SciPy

ตย การใช้งาน: การวาดกราฟ (sine curve)

```
import numpy as np
import matplotlib.pyplot as plt

# คำนวณค่าจุด x,y ที่จะอยู่บน sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)

# พล็อตกราฟ
plt.plot(x, y)
plt.show()
```



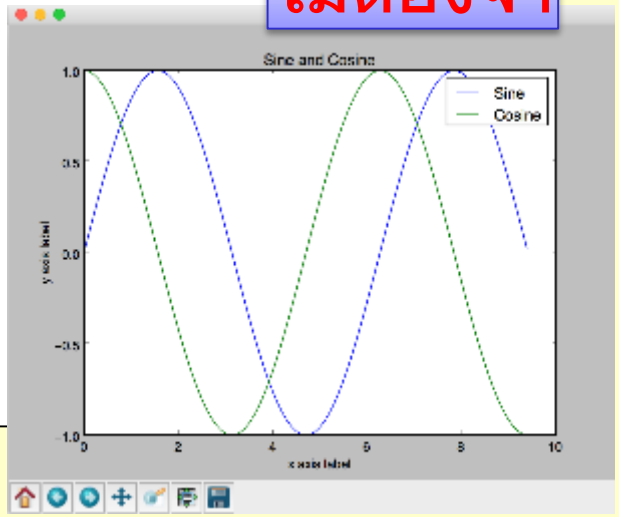
http://cs231n.github.io/python-numpy-tutorial/#numpy

ตย การใช้งาน: การวาดกราฟ (sine และ cos curves)

```
import numpy as np
import matplotlib.pyplot as plt

# คำนวณค่าจุด x,y ที่จะอยู่บน sine และ cos curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# พล็อตกราฟ โดยมีการระบุรายละเอียดของกราฟ
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```

เพิ่มจากหน้าที่แล้ว

ไม่ต้องจำ



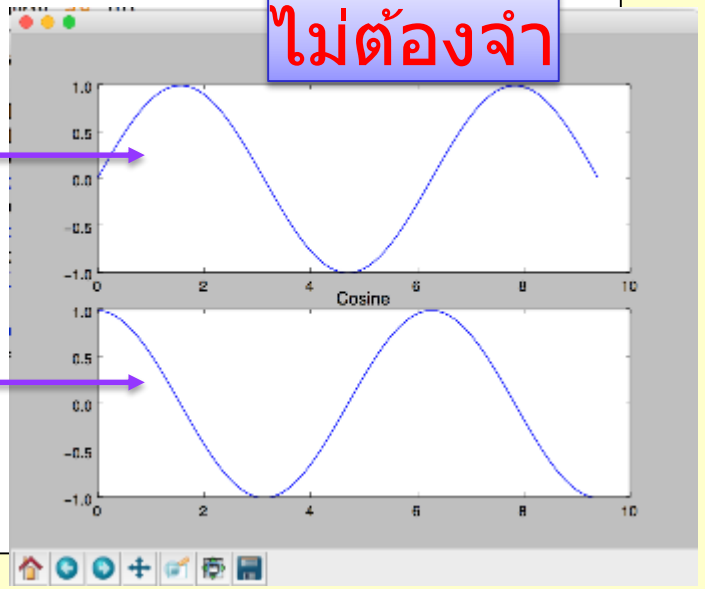
http://cs231n.github.io/python-numpy-tutorial/#numpy

ตย การใช้งาน: การวาดกราฟ (แยกเป็นสองกราฟ บน ล่าง)

```
import numpy as np
import matplotlib.pyplot as plt

# คำนวณค่าจุด x,y ที่จะอยู่บน sine และ cos curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

#ระบุรูปแบบการแสดงกราฟย่อย
plt.subplot(2, 1, 1)
plt.plot(x, y_sin)
plt.title('Sine')
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')
plt.show()
```

2 บรรทัด 1 คอลัมน์

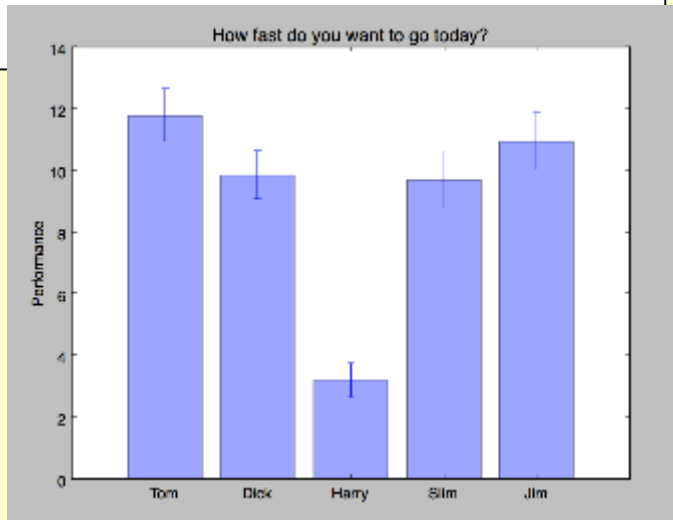
ไม่ต้องจำ



http://cs231n.github.io/python-numpy-tutorial/#numpy

ตย การใช้งาน: การวาดกราฟ (Bar Chart)

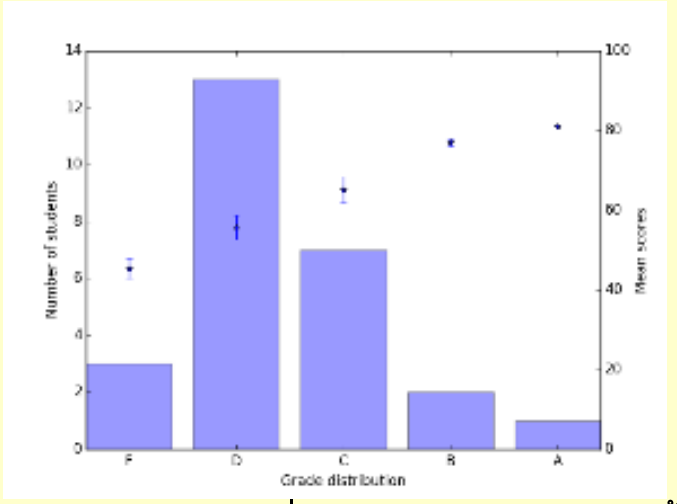
```
import numpy as np
import matplotlib.pyplot as plt

# Example data
people = ('Tom', 'Dick', 'Harry', 'Slim', 'Jim')
x_pos = np.arange(len(people))
performance = 3 + 10 * np.random.rand(len(people))
error = np.random.rand(len(people))

plt.bar(x_pos, performance, yerr=error, align='center',alpha=0.4)
plt.xticks(x_pos, people)
plt.ylabel('Performance')
plt.title('How fast do you want to go today?')
plt.show()
```



ลองเขียนดู CH08_5

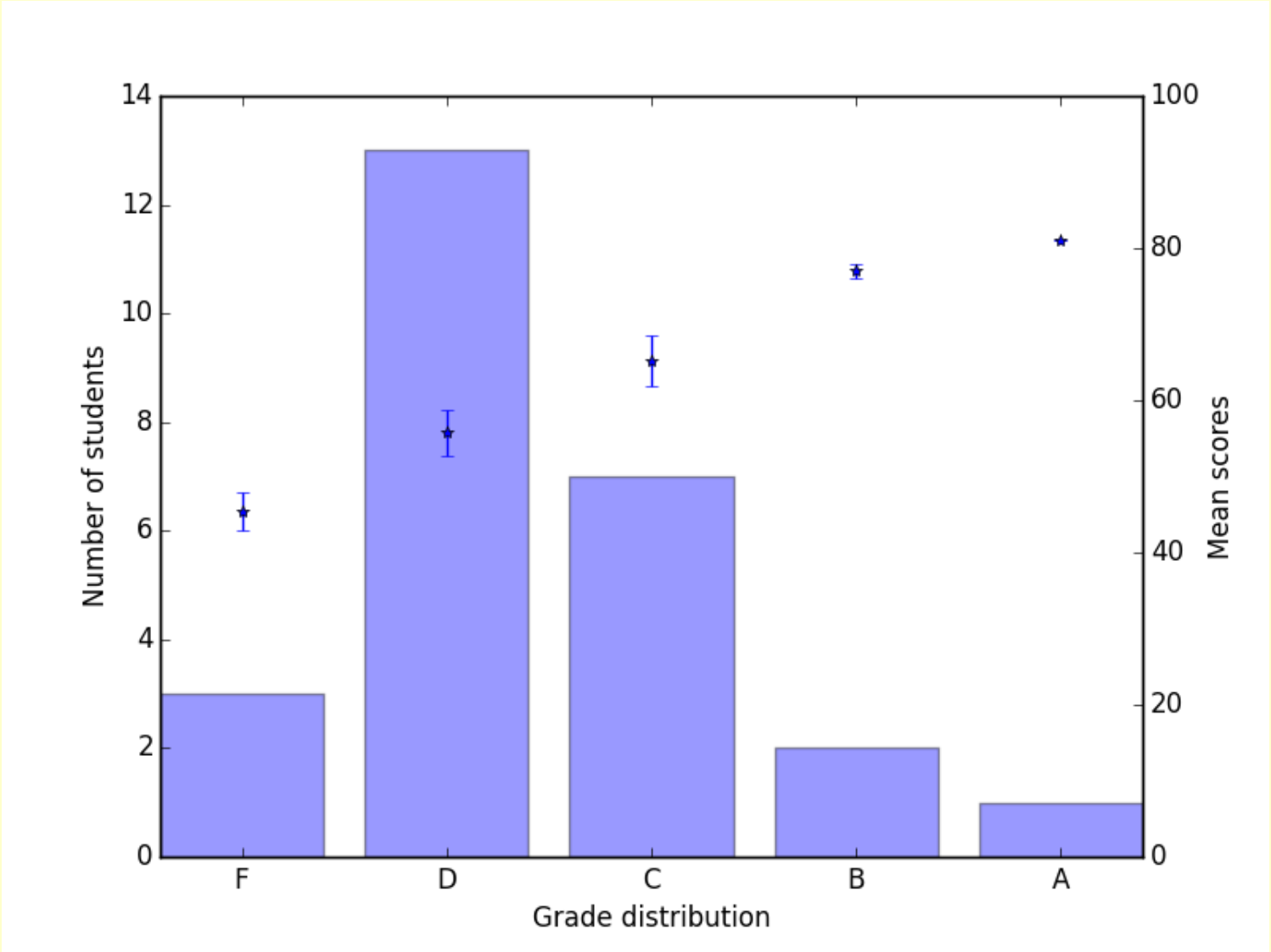
เขียนโคดเพิ่มเติมจากโปรแกรมในโจทย์ CH08_4 แทนที่จะเขียนคะแนนรวมของนิสิตแต่ละคนใส่ในไฟล์ ให้นำคะแนนรวมชุดนี้ไปหาความถี่ของนิสิตในแต่ละช่วงคะแนนต่อไปนี้

$x < 50$
$50 \leq x < 60$
$60 \leq x < 70$
$70 \leq x < 80$
$80 \leq x$



แล้วใช้ matplotlib เพื่อการวาดกราฟแสดงความถี่ออกมา นอกจากนี้
1) ทำการ label แต่ละช่วงเป็นเกรด F, D, C, B, A ในแกน x ตามลำดับ
2) แสดง mean และ standard deviation ของแต่ละช่วงคะแนน
3) มี label แกน x เป็น “Grade distribution”
4) มี label แกน y (1^{st} y-axis) เป็น “Number of students”
5) มี label แกน y’ (2^{nd} y-axis) เป็น Score

ลองเขียนดู CH08_5 (ต่อ)



CH08_5: Hints

จากคะแนนดิบที่อ่านเข้ามา
หา sum_scores (list ที่เก็บผลรวมคะแนนของนิสิตแต่ละคน)
[81. 69. 58. 52. 59. 53. 59. 59. 69. 52. 59. 54. 76. 55. 46.
60. 48. 78. 64. 68. 64. 56. 42. 62. 51. 58.]

หา grades (list ของ tuple (เกรด, คะแนน) ของนิสิตแต่ละคน
[('A', 81.0), ('C', 69.0), ('D', 58.0), ('D', 52.0), ('D', 59.0), ('D', 53.0), ('D', 59.0), ('D', 59.0),
('C', 69.0), ('D', 52.0), ('D', 59.0), ('D', 54.0), ('B', 76.0), ('D', 55.0), ('F', 46.0), ('C', 60.0),
('F', 48.0), ('B', 78.0), ('C', 64.0), ('C', 68.0), ('C', 64.0), ('D', 56.0), ('F', 42.0), ('C', 62.0),
('D', 51.0), ('D', 58.0)]

หา freq (list ห้าช่อง ที่เก็บจำนวนนิสิตที่ได้เกรดแต่ละเกรด ช่อง 0 คือจำนวน F, ...)
[3, 13, 7, 2, 1]

หา grouped_scores (list ห้าช่อง ที่เก็บ list ของคะแนนในแต่ละเกรด)
[[46.0, 48.0, 42.0], [58.0, 52.0, 59.0, 53.0, 59.0, 59.0, 52.0, 59.0, 54.0, 55.0, 56.0, 51.0,
58.0], [69.0, 69.0, 60.0, 64.0, 68.0, 64.0, 62.0], [76.0, 78.0], [81.0]]

หา std (list ห้าช่อง ที่เก็บ standard deviation ของคะแนนในแต่ละเกรด)
[2.4944382578492941, 2.9652823488302129, 3.3135467156409146, 1.0, 0.0]

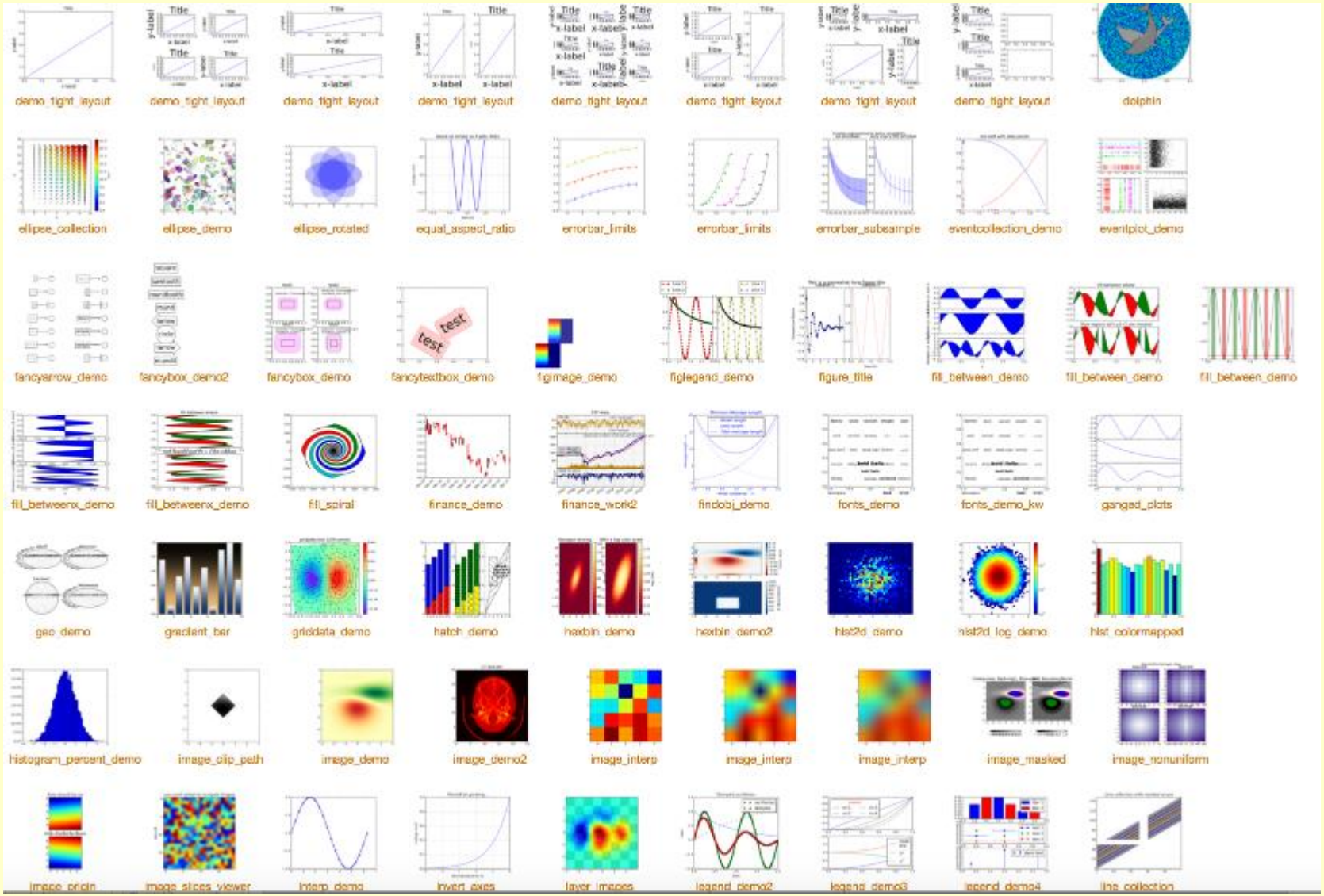
หา mean (list ห้าช่อง ที่เก็บ mean ของคะแนนในแต่ละเกรด)
[45.333333333333336, 55.769230769230766, 65.142857142857139, 77.0, 81.0]

นำ freq, std, และ mean ไปวาดกราฟ

CH08_5 : Hints

```
import numpy as np
import matplotlib.pyplot as plt

def read_sum_scores():
    ???

def get_grade_stat(sum_scores):
    ???

sum_scores = read_sum_scores() # อ่านแฟ้มเพื่อหาคะแนนรวมของนิสิตแต่ละคน
freq, std, mean = get_grade_stat(sum_scores) #หาสถิติที่ต้องการ
fig, ax = plt.subplots(1, 1)
ax2 = ax.twinx()
grades = ('F', 'D', 'C', 'B', 'A')
x_pos = np.arange(len(grades))
plt.xticks(x_pos, grades)
ax.set_xlabel("Grade distribution")
ax.set_ylabel("Number of students")
ax.bar(x_pos, np.array(freq),  align='center', alpha=0.4)
ax2.set_ylabel("Mean scores")
ax2.set_ylim(0, 100)
plt.errorbar(x_pos, np.array(mean), np.array(std), \
             linestyle='None', marker='*')
plt.show()
```

ตัวอย่างการแสดงผลของ matplotlib อื่นๆ



http://matplotlib.org/gallery.html

Application 1: Basic Image Processing

https://softwaredevelopmentperestroika.wordpress.com/2014/02/11/image-processing-with-python-numpy-scipy-image-convolution/

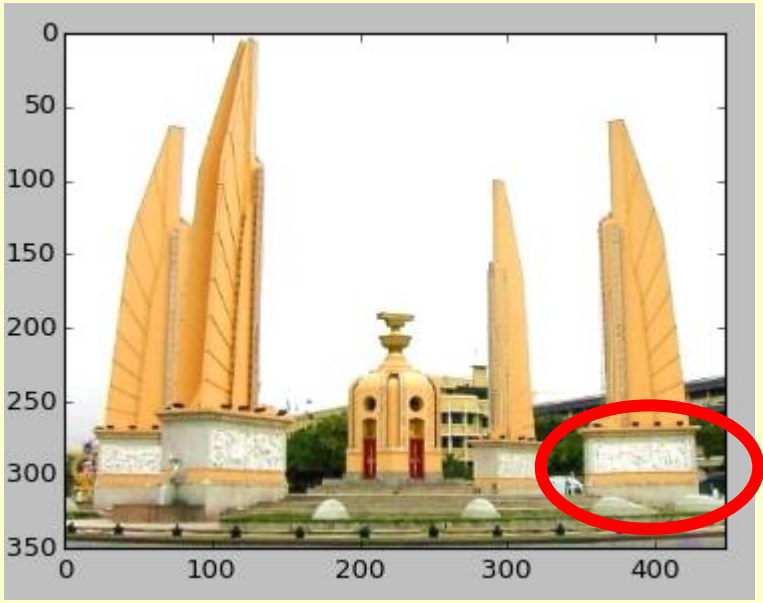
การแสดงรูปภาพใน python ด้วย matplotlib

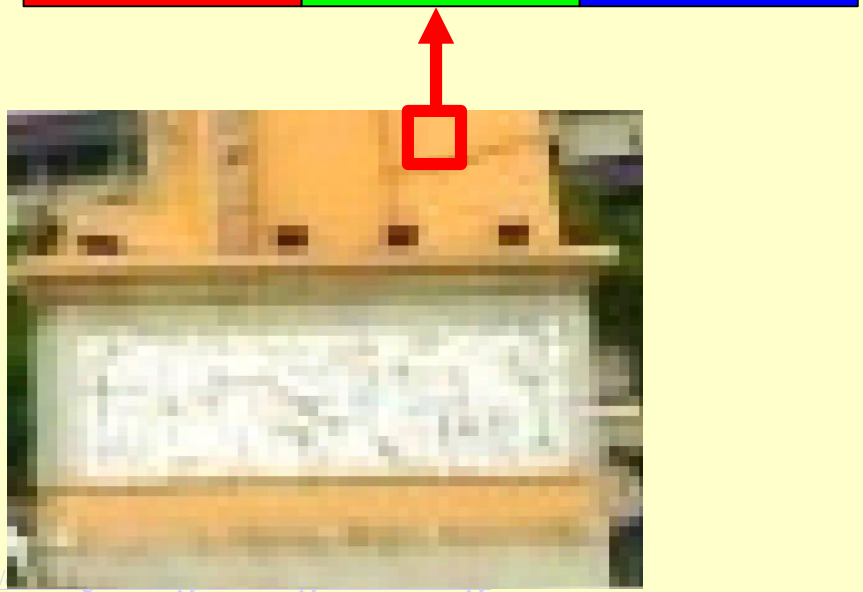
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

image = mpimg.imread('monument.png')
plt.imshow(image)

plt.show()
```

เพื่อความง่ายโปรแกรมที่จะเขียนจากนี้ไปใช้กับแฟ้ม png เท่านั้น

image เป็นอาเรย์ที่แต่ละช่องมีค่า 0 ถึง 1 แทนความเข้มของสี

1 pixel = 1 จุดสี มี 3 สีย่อย





1 ภาพ เป็นอาเรย์ 3 สามมิติ

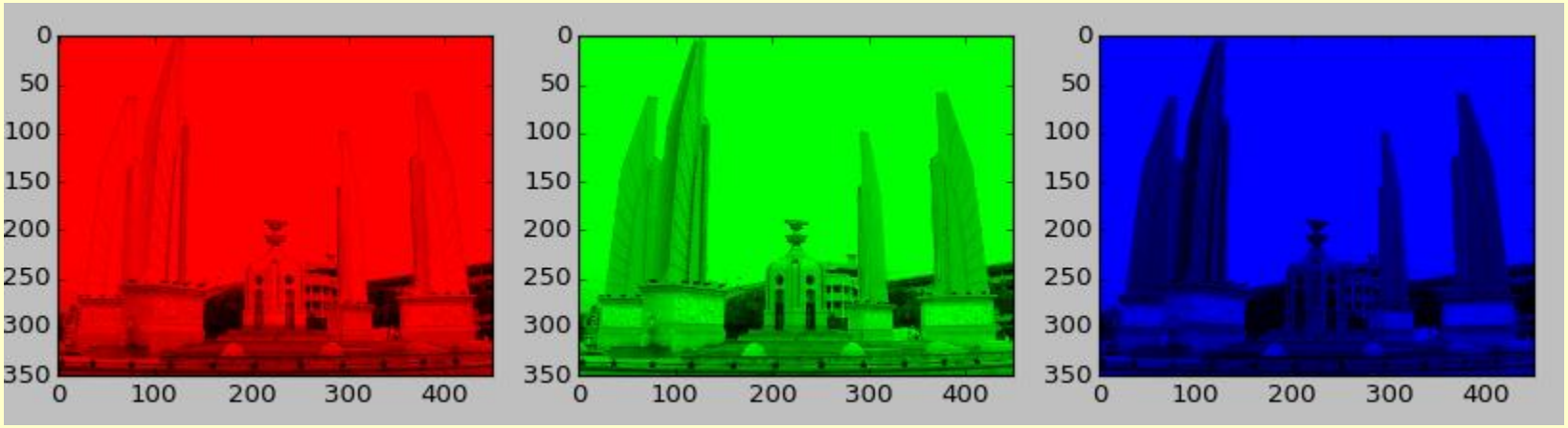
- monument.png เป็นภาพขนาด 350 x 449
- ภาพขนาด 350 x 449 มี 350 x 449 จุด
- 1 จุด เก็บค่าความเข้มของ 3 สีย่อย R G B
- ภาพนี้จึงเก็บข้อมูล 350 x 449 x 3 ค่า

```
img = mpimg.imread('monument.png')
```

img.shape → (350, 449, 3)

1 ภาพ เป็นอาเรย์ 3 สามมิติเก็บค่าความเข้มของ R G B

```
>>> image.shape
(350, 449, 3)
```



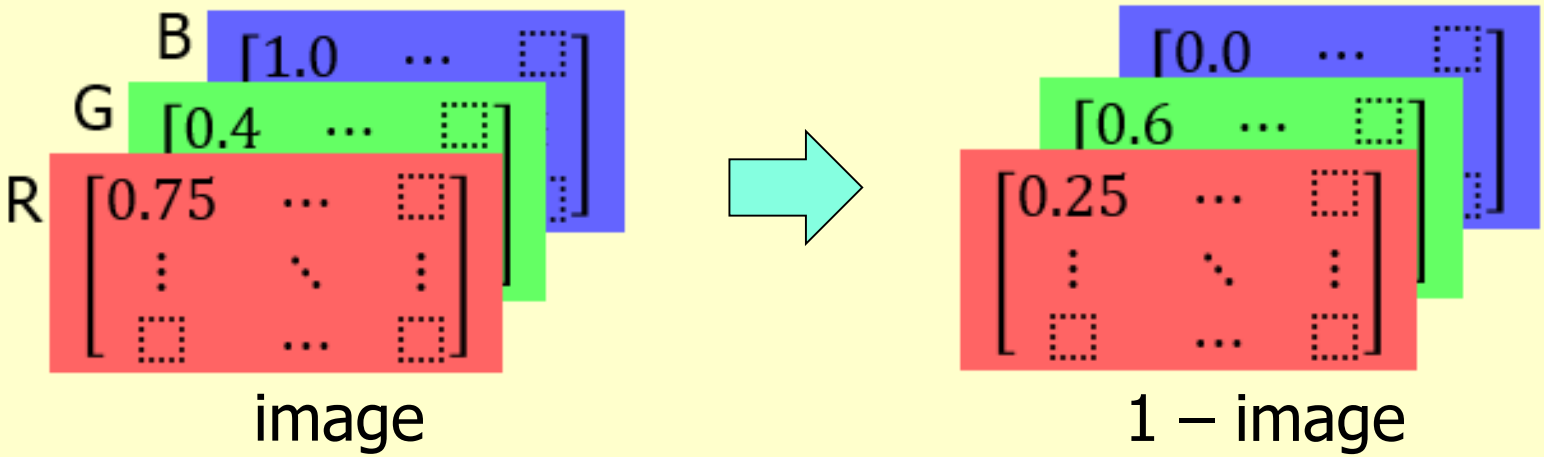
อาร์เรย์ 2 มิติสีแดง
image[:,:,0]

อาร์เรย์ 2 มิติสีเขียว
image[:,:,1]

อาร์เรย์ 2 มิติสีน้ำเงิน
image[:,:,2]

การประมวลผลภาพเบื้องต้น

- image = mpimg.imread('monument.png')
- image.shape → (350, 449, 3)
- image = 1 – image (broadcast & element-wise op.)

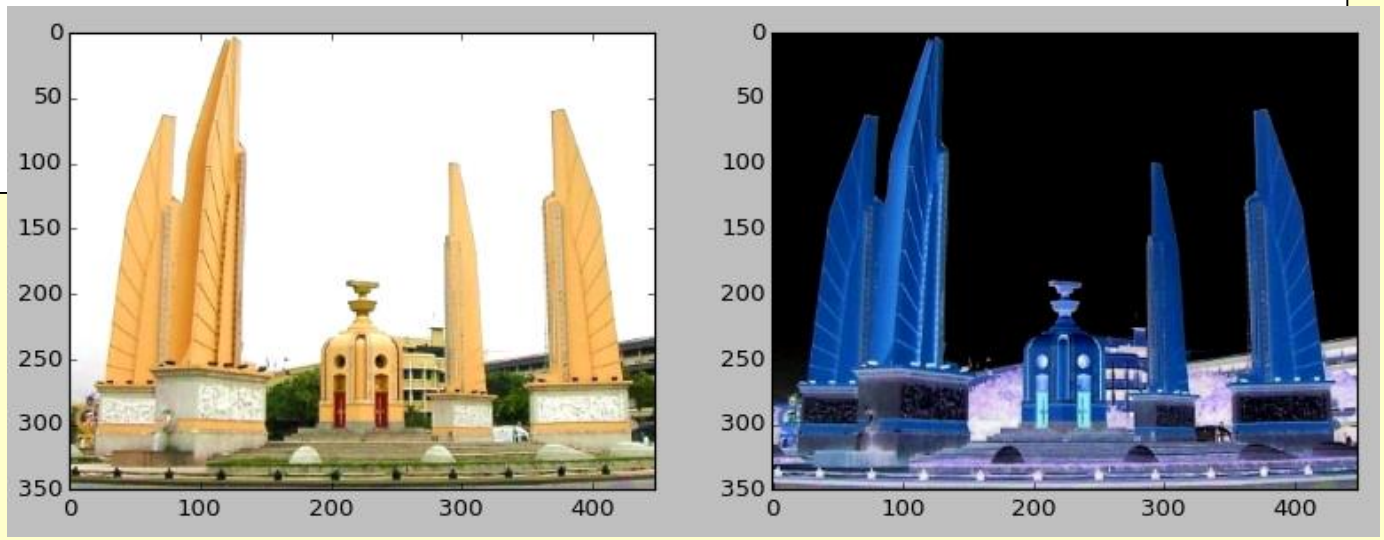


ทำแบบนี้แล้ว ภาพเปลี่ยนแปลงไปอย่างไร ?

การประมวลผลภาพเบื้องต้น : ภาพ Negative

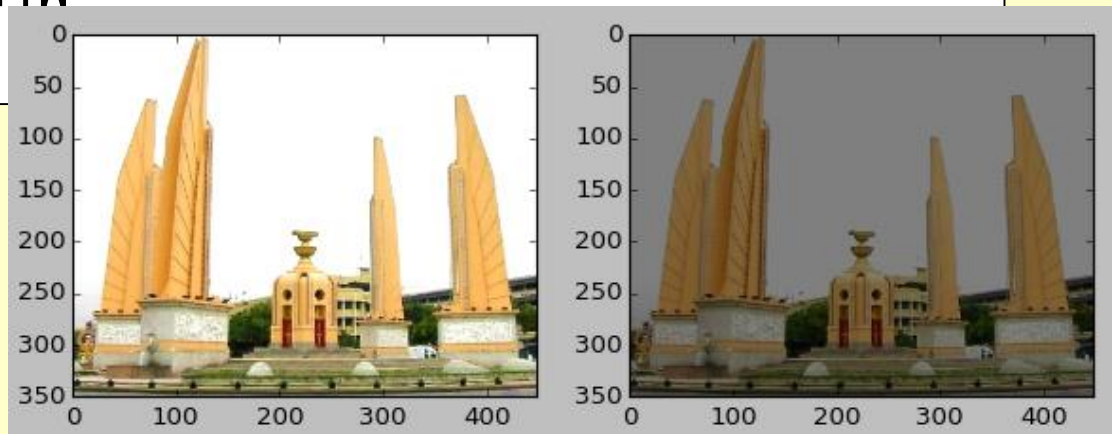
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

image = mpimg.imread('monument.png')
plt.subplot(1, 2, 1)
plt.imshow(image)

negative = 1 - image
plt.subplot(1, 2, 2)
plt.imshow(negative)

plt.show()
```

1 – image คือนำค่าทุกช่องไปลบออกจาก 1 ความเข้มสีเปลี่ยนเป็นตรงข้าม (ขาว → ดำ, เหลือง → น้ำเงิน, ...)
broadcast & element-wise subtract



ลองทำดู : การลดความสว่างภาพ

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread("monument.png")

img_dim = ____________________________________   # ???

plt.subplot(1, 2, 1)   # 1 แถว  2 คอลัมน์  คอลัมน์ที่ 1
plt.imshow(img)        # วาดลงหน่วยความจำก่อน

plt.subplot(1, 2, 2)   # 1 แถว  2 คอลัมน์  คอลัมน์ที่ 2
plt.imshow(img_dim)    # วาดลงหน่วยความจำก่อน
plt.show() #แสดงออกจอภาพ
```

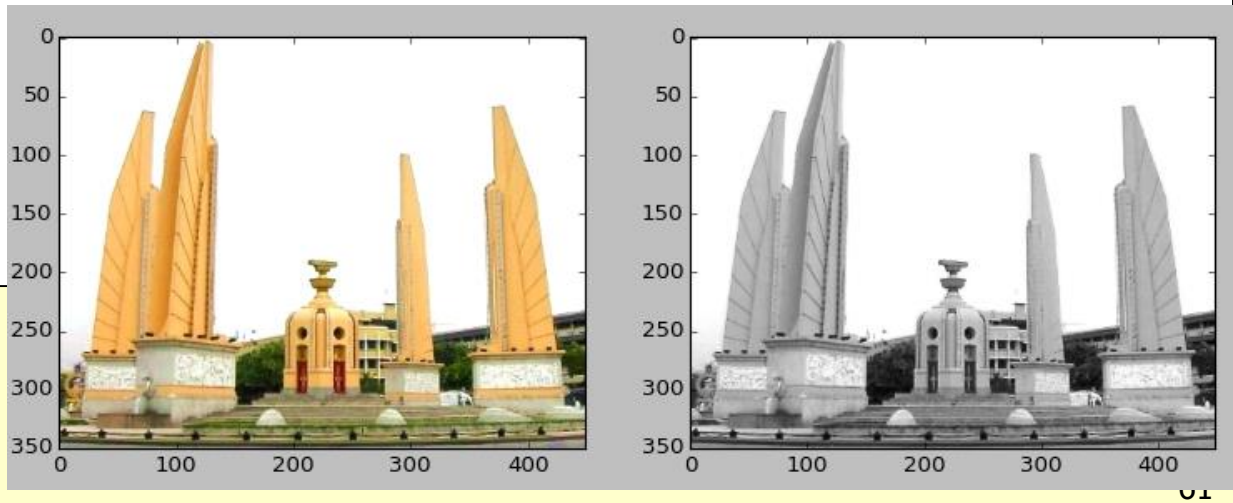


การประมวลผลภาพเบื้องต้น : ภาพสีเทา

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

image = mpimg.imread('monument.png')
plt.subplot(1, 2, 1)
plt.imshow(image)
gray = np.ndarray(image.shape)
gray[:,:,0] = \
gray[:,:,1] = \
gray[:,:,2] = (image[:,:,0]+image[:,:,1]+image[:,:,2]) / 3
plt.subplot(1, 2, 2)
plt.imshow(gray)

plt.show()
```

จุดสีที่ R G B มีค่าความเข้มเท่ากัน ได้จุดสีเทา

นำค่าความเข้มของ R G B มาหาค่าเฉลี่ย แล้วเปลี่ยน R G B ให้เป็นความเข้มระดับเดียวกัน



ภาพสีเทาแบบประหยัดเนื้อที่

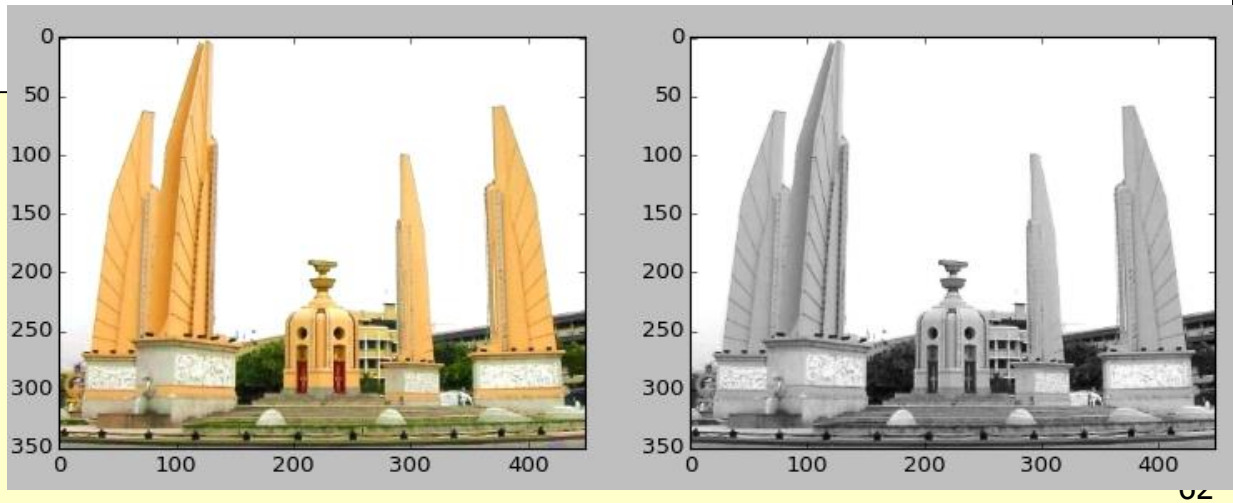
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

image = mpimg.imread('monument.png')
plt.subplot(1, 2, 1)
plt.imshow(image)
gray = np.ndarray(image.shape(0:2))
gray = (image[:,:,0]+image[:,:,1]+image[:,:,2]) / 3
plt.subplot(1, 2, 2)
plt.imshow(gray, cmap='gray')

plt.show()
```

ใช้อาเรย์ 2 มิติ เก็บค่าความเข้มสีเทา

image.shape → (350,449,3)
image.shape(0:2) → (350,449)

ภาพ 350x449 ใช้อาเรย์ขนาด 350x449

ถ้าไม่ใส่ → สีเพี้ยนๆ



เขียนฟังก์ชันทำภาพสีเทาแบบประหยัดเนื้อที่

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
#-----------------------------------
def grayscale(image):
    return (image[:,:,0]+image[:,:,1]+image[:,:,2]) / 3
#-----------------------------------
image = mpimg.imread('monument.png')
plt.subplot(1, 2, 1)
plt.imshow(image)

gray = grayscale(image)
plt.subplot(1, 2, 2)
plt.imshow(gray, cmap='gray')

plt.show()
```

element-wise addition

broadcast 3 & element-wise division

สีเทา เขาว่าสูตรนี้ดีกว่า 0.299*R + 0.587*G + 0.114*B

- CH08_6 จงเขียนโปรแกรมเพื่อแปลงรูปภาพจากรูปสีให้เป็นรูปสีเทา (gray scale) ด้วยสูตรข้างบนนี้เทียบกับสูตร (R + G + B)/3

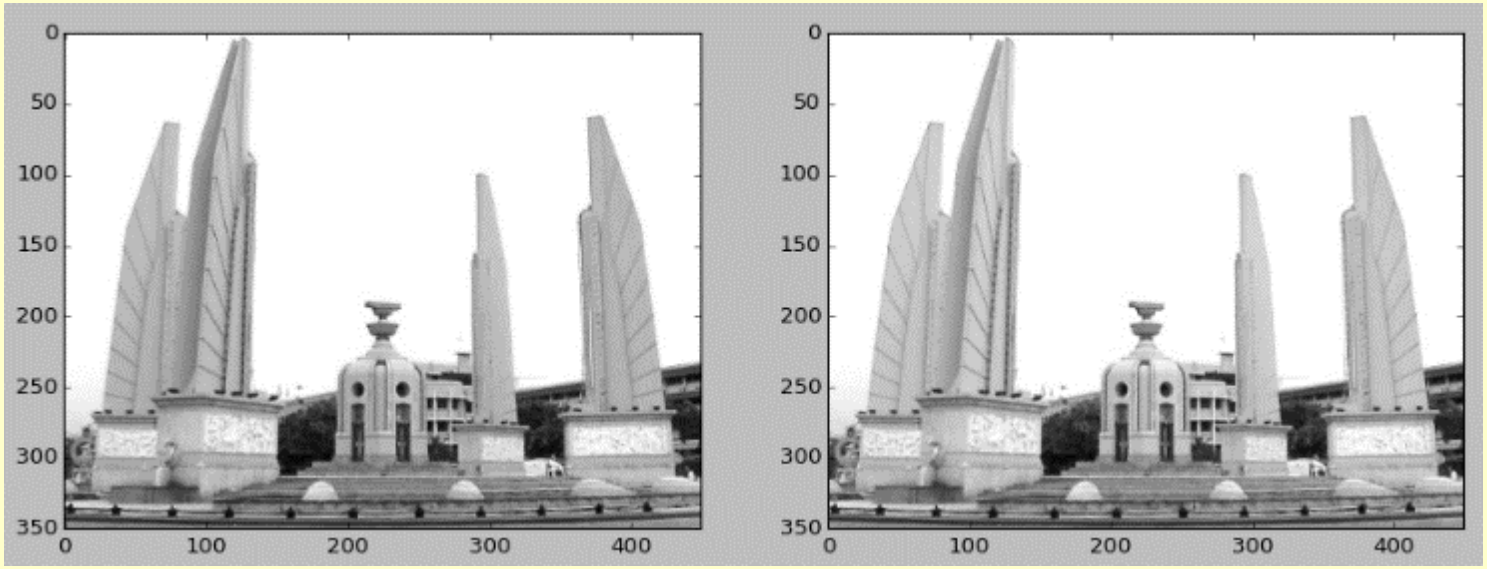
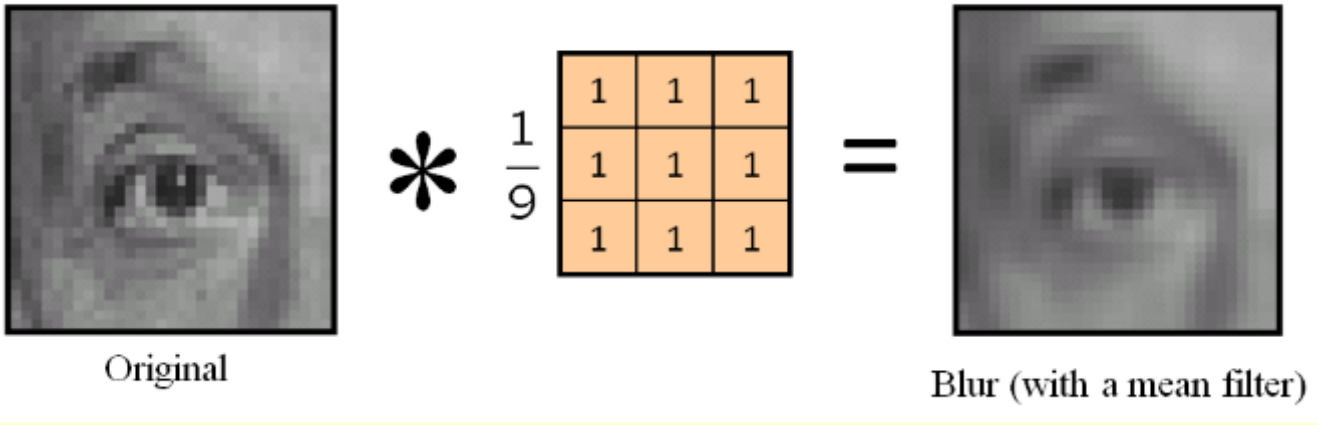
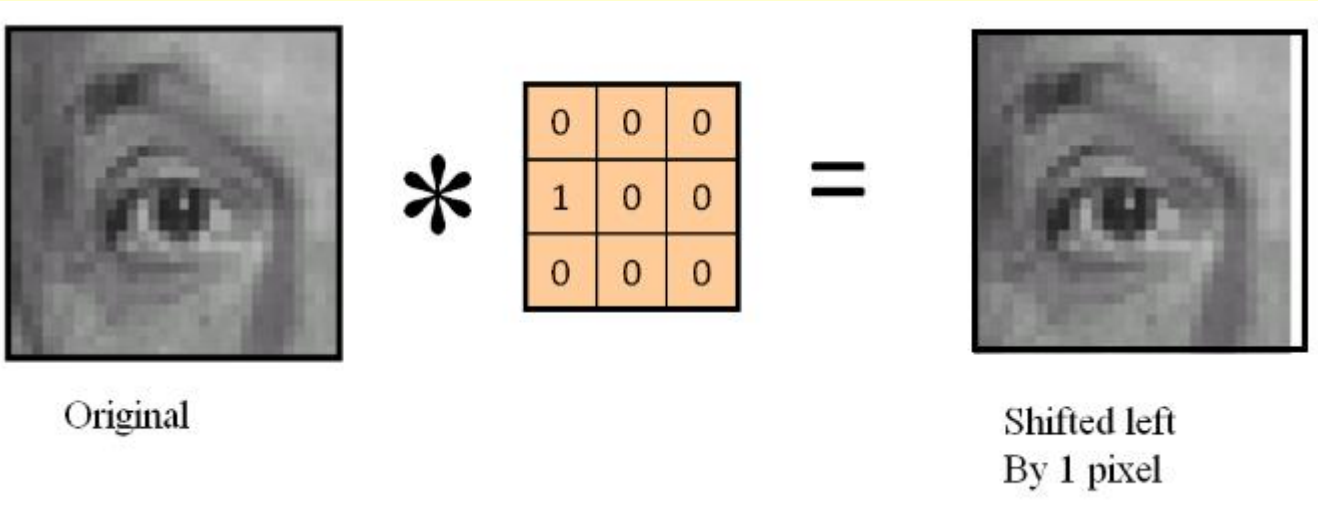


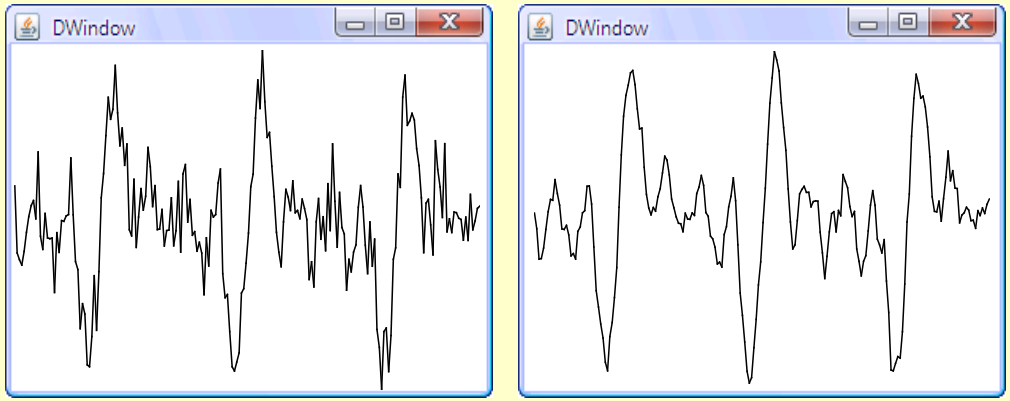
Image Convolution

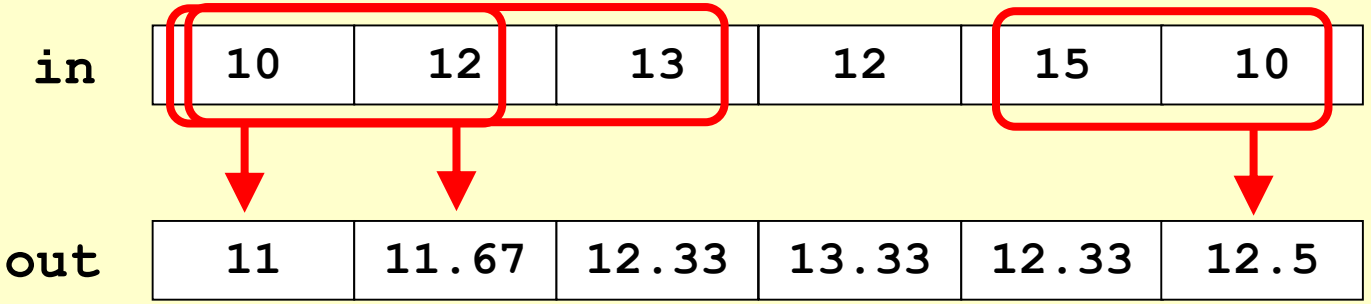
- Image Convolution คือการนำรูปภาพมาผ่านตัวกรอง (kernel) เพื่อให้ได้ผลลัพธ์ตามที่ต้องการเช่น blur, ขยับรูป, จับขอบรูป เป็นต้น



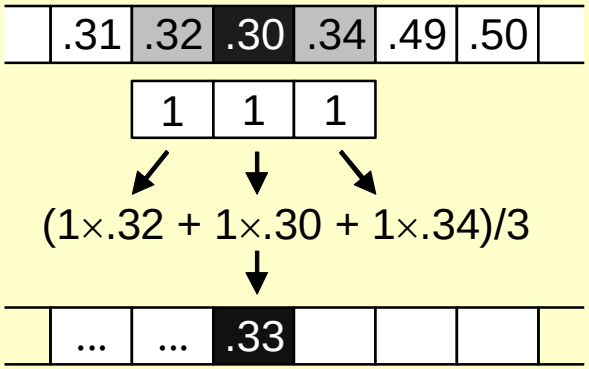


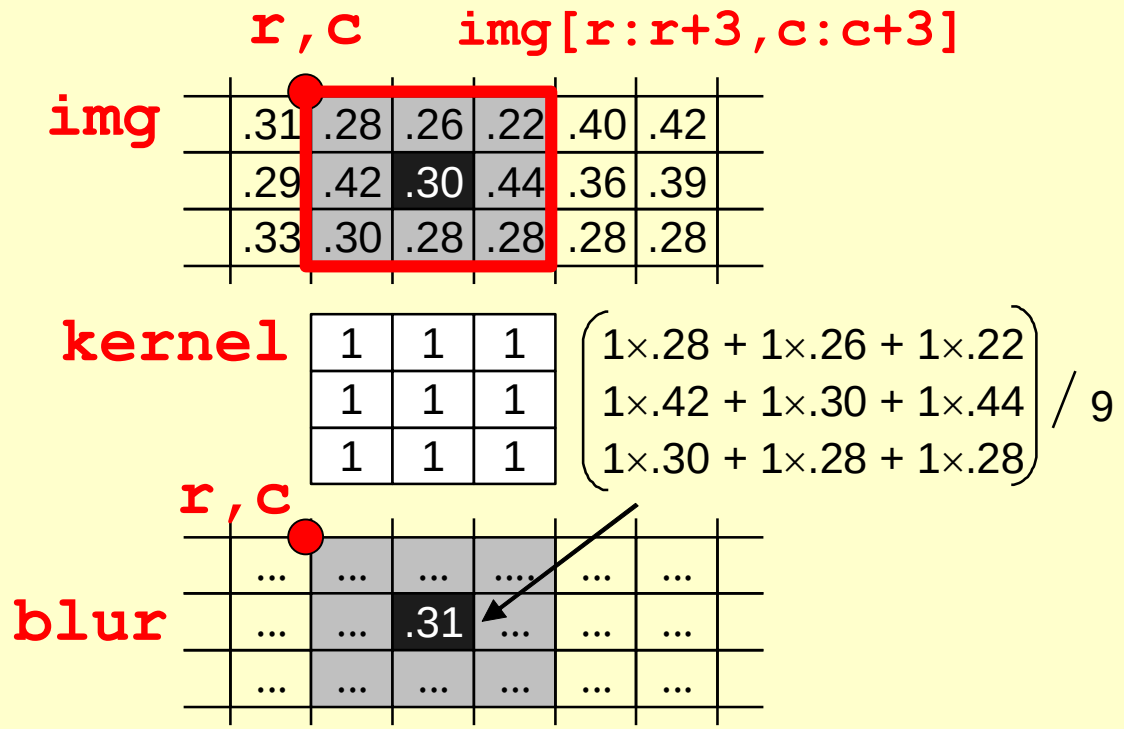
ทบทวน : Moving Average หนึ่งมิติ





Moving Average 2 มิติกับภาพ ได้ภาพเบลอ (blur)






```
kernel = np.array([[1/9, 1/9, 1/9],
                   [1/9, 1/9, 1/9],
                   [1/9, 1/9, 1/9]])
...
blur[r+1,c+1] = np.sum( img[r:r+3,c:c+3]*kernel )
```

sum all elements

element-wise multiplication

คำนวณตรงนี้ได้ค่าความเข้มสีเทาแค่จุดเดียว

ต้องคำนวณ moving average ของทุกจุดในภาพ

```
kernel = np.array([[1/9, 1/9, 1/9],
                   [1/9, 1/9, 1/9],
                   [1/9, 1/9, 1/9]])
blur = np.ndarray(img.shape) #จองอาเรย์ผลลัพธ์ก่อน
for (r,c),e in np.ndenumerate( img[1:-1, 1:-1] ):
    blur[r+1,c+1] = np.sum( img[r:r+3,c:c+3]*kernel )
```

ไม่คำนวณตรงขอบ

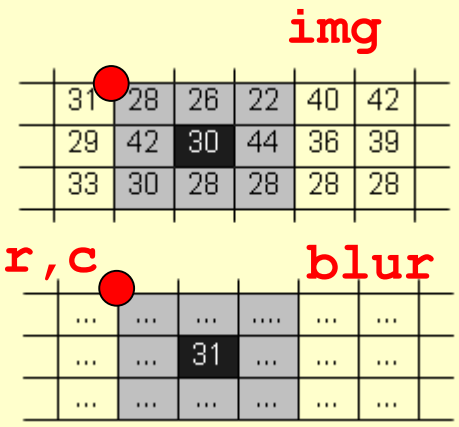
img.shape → (350,449)

img[1:-1,1:-1].shape → (348,447)

for (r,c),e in np.ndenumerate(img[1:-1,1:-1]) :

ได้ (r,c) → (0,0), (0,1), ..., (0,446),
(1,0), (1,1), ..., (1,446),
...
(347,0), (347,1), ..., (347,446)

ไม่คำนวณจุดที่ขอบ

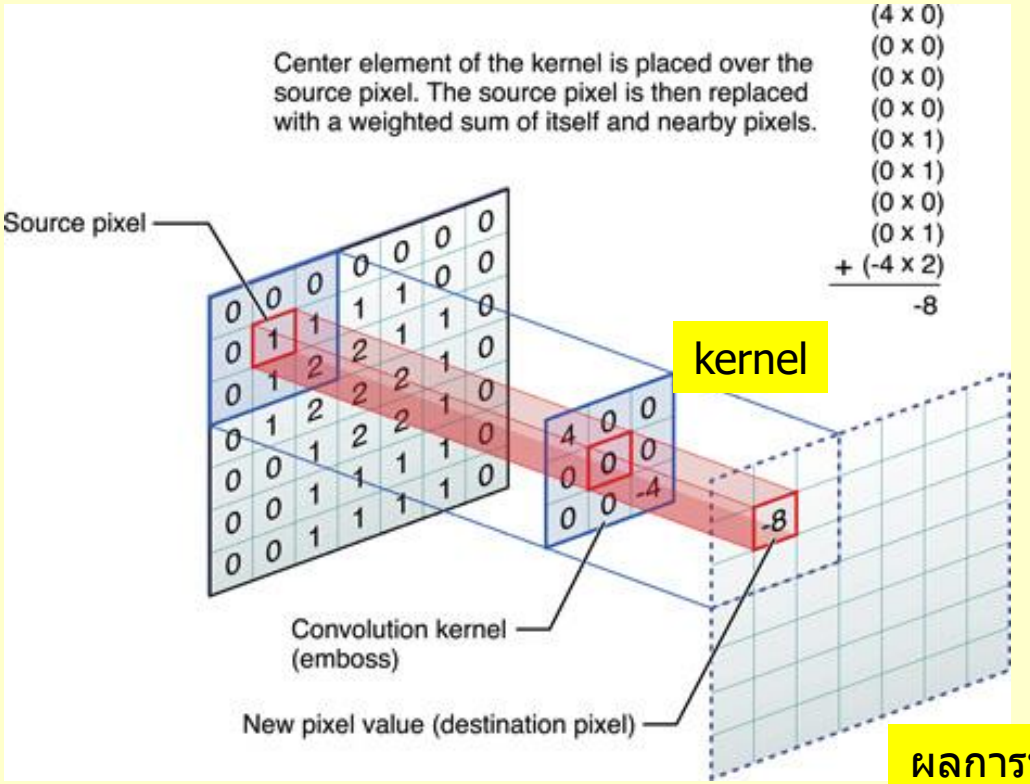


การประมวลผลภาพเบื้องต้น : ภาพเบลอ

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
def blur(img):
    kernel = np.array([[1/9, 1/9, 1/9],
                       [1/9, 1/9, 1/9],
                       [1/9, 1/9, 1/9]])
    blur = np.ndarray(img.shape)
    for (r, c), e in np.ndenumerate(img[1:-1,1:-1]):
        blur[r+1,c+1] = np.sum(img[r:r+3, c:c+3] * kernel)
    return blur

def grayscale(image):
    return (image[:,:,0] + image[:,:,1] + image[:,:,2]) / 3

image = mpimg.imread('monument.png')
imggray = grayscale(image)
plt.subplot(1, 2, 1)
plt.imshow(imggray, cmap='gray')
plt.subplot(1, 2, 2)
plt.imshow(blur(imggray), cmap='gray')
plt.show()
```

เรียกส่วนนี้ว่า convolution

Convolution



ผลการทำ convolute

เปลี่ยนค่าของเมทริกซ์ kernel จะได้การประมวลผลภาพแบบอื่น ๆ

เขียนฟังก์ชัน convolute : blur เรียกใช้ convolute

```
def convolute(img, kernel):  # assume kernel is a 3x3 matrix
    result = np.ndarray(img.shape)
    for (r, c), e in np.ndenumerate(img[1:-1,1:-1]):
        v = np.sum(img[r:r+3, c:c+3] * kernel)
        result[r+1,c+1] = min(1,max(0,v))
    return result

def blur(img):
    blur_matrix = np.array([[1/9, 1/9, 1/9],
                            [1/9, 1/9, 1/9],
                            [1/9, 1/9, 1/9]])
    img = convolute(img, blur_matrix)
    return img

def grayscale(image):
    return (image[:,:,0] + image[:,:,1] + image[:,:,2]) / 3
image = mpimg.imread('monument.png')
imggray = grayscale(image)
plt.subplot(1, 2, 1)
plt.imshow(imggray, cmap='gray')
plt.subplot(1, 2, 2)
plt.imshow(blur(imggray), cmap='gray')
plt.show()
```

ตัดให้อยู่ช่วง [0,1]

ต้องการ blur มากๆ เรียกซ้ำ ๆ

การประมวลผลภาพ : อีก

แบบนี้ทำอะไร ?

```
def ????????(img):
    kernel = np.array([[1,  1, 1],
                       [1, -8, 1],
                       [1,  1, 1]])
    img = convolute(img, kernel)
    return img
```

```
def ????????(img):
    kernel = np.array([[1,  1, 1],
                       [1, -8, 1],
                       [1,  1, 1]])
    img = convolute(img, kernel)
    return -img
```

ติดลบตรงนี้ได้อะไร ?

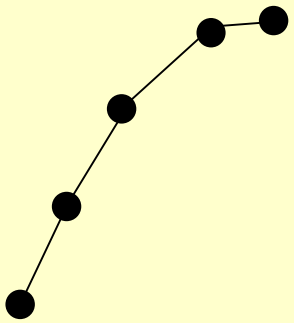
สรุป

- นิสิตต้อง
  - รู้จักการแสดงภาพและประมวลผลภาพเบื้องต้น
  - เข้าใจการใช้ broadcasting และ element-wise operation กับ numpy array
  - เขียนการประมวลผลเมทริกซ์แบบ convolution ได้
  - เขียนการประมวลผลภาพเบื้องต้นได้

Application 2: Linear Transformation

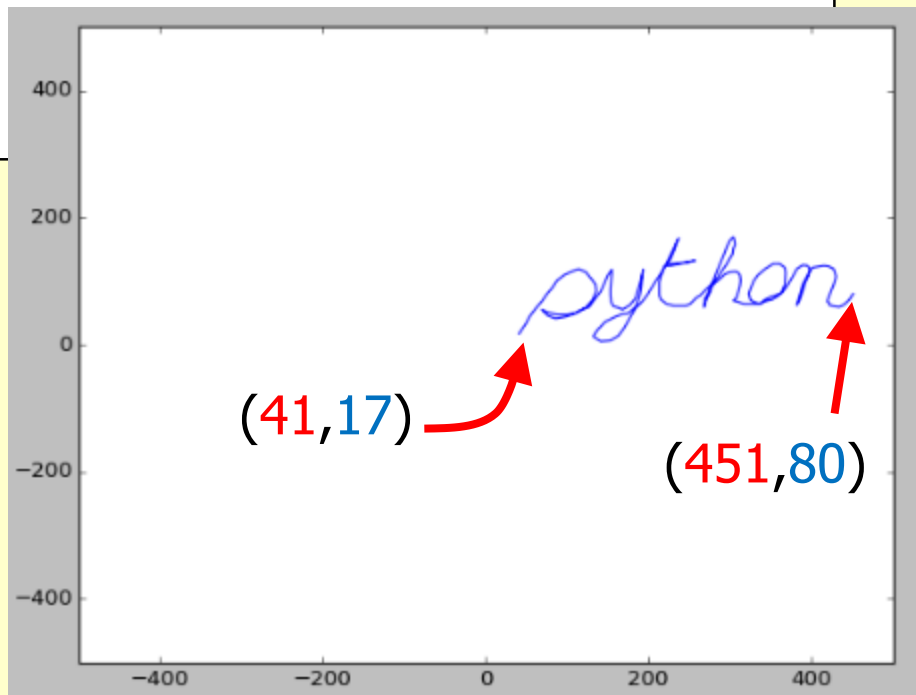
โปรแกรมลากเส้นจากรายการของจุด

```
import numpy as np
import matplotlib.pyplot as plt

xy = np.loadtxt("polyline.csv", delimiter=",")
# xy = np.array([[41,17],[49,32],[54,44],...,[451,80])
# xy[:,0] คือ np.array([41,49,54,...,451]) พิกัด x
# xy[:,1] คือ np.array([17,32,44,...,80])  พิกัด y
                xmin, xmax, ymin, ymax
plt.axis((-500,500,-500,500))
plt.plot( xy[:,0], xy[:,1] )
plt.show()  พิกัด x      พิกัด y
```



ลากเส้นระหว่างจุด

41, 17
49, 32
54, 44
61, 56
69, 68
...
446, 70
451, 80

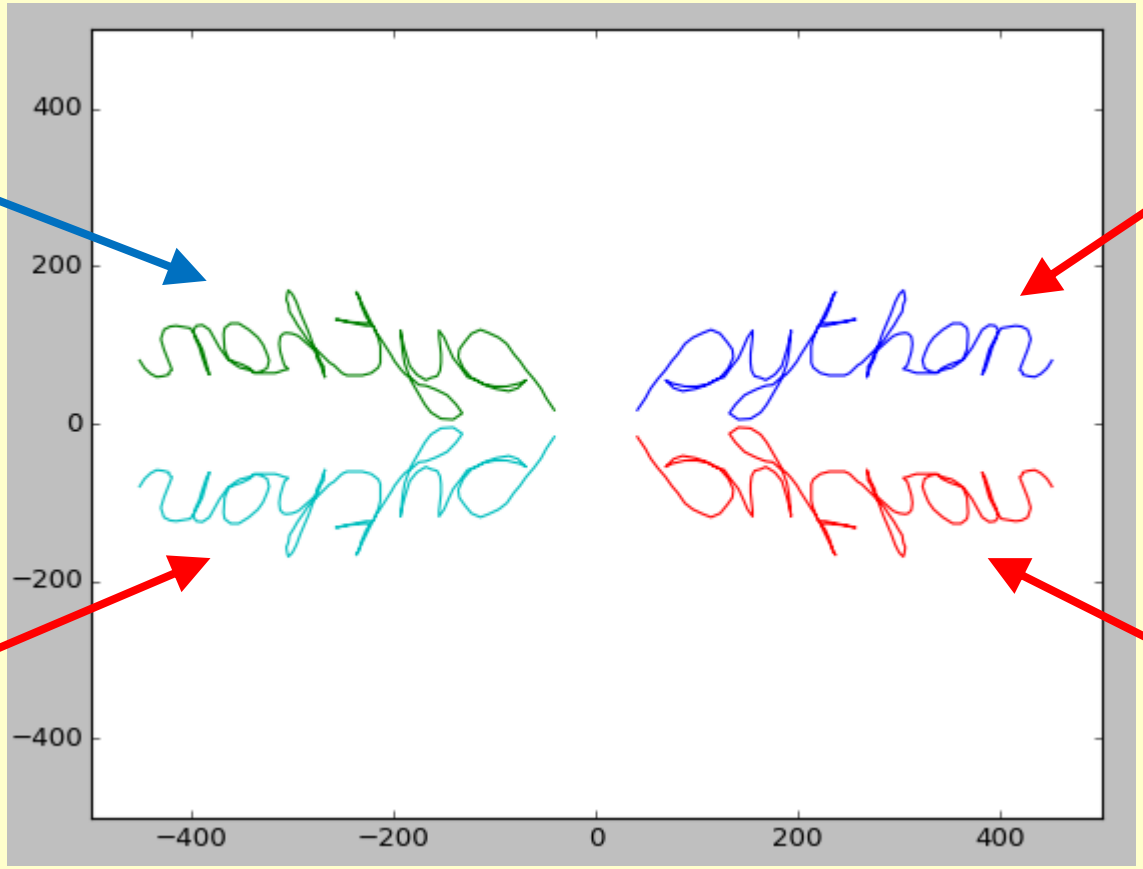


การแปลงพิกัดของจุดแบบง่าย (มีหลายแบบ)

-41, 17
-49, 32
-54, 44
-61, 56
-69, 68
...
-446, 70
-451, 80

x ใหม่คือ –x เดิม, y ใหม่คือ y เดิม

41, 17
49, 32
54, 44
61, 56
69, 68
...
446, 70
451, 80

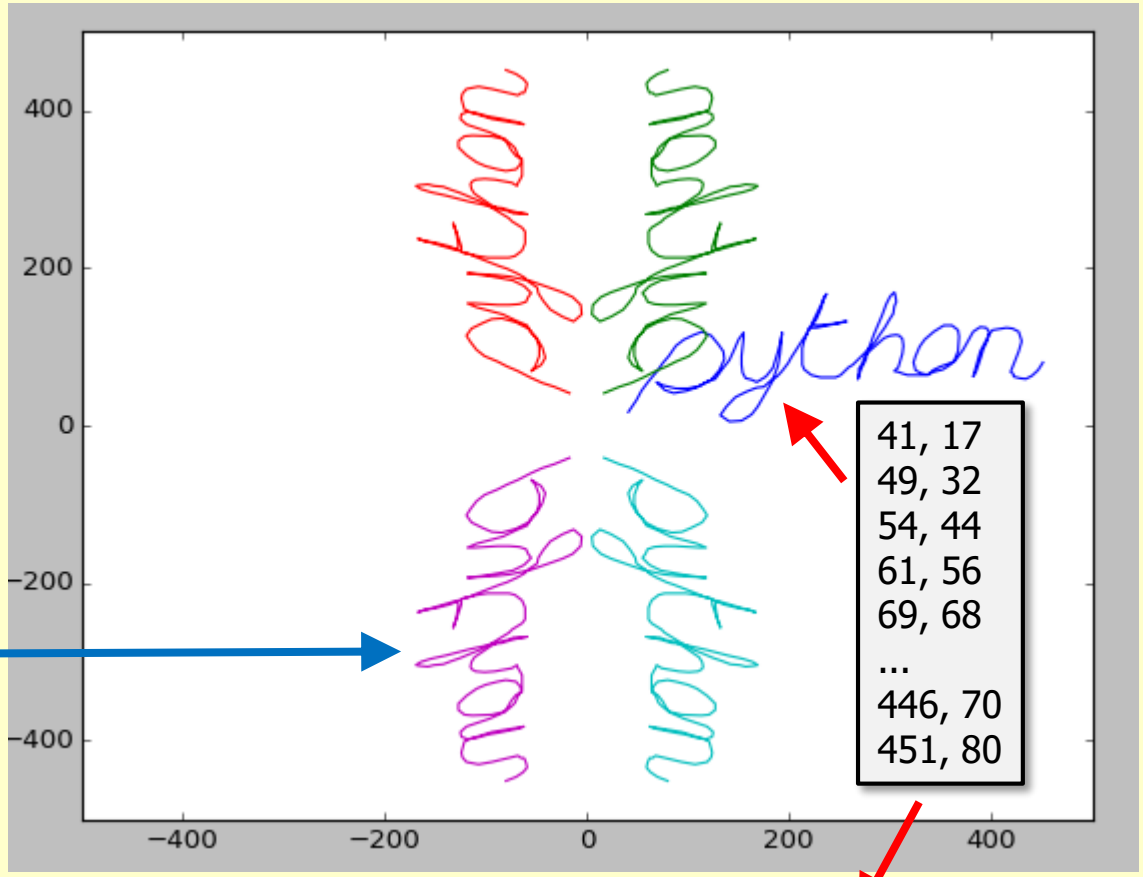


-41, -17
-49, -32
-54, -44
-61, -56
-69, -68
...
-446, -70
-451, -80

41, -17
49, -32
54, -44
61, -56
69, -68
...
446, -70
451, -80

การแปลงพิกัดของจุดแบบง่าย (มีหลายแบบ)

-17, 41
-32, 49
-44, 54
-56, 61
-68, 69
...
-70, 446
-80, 451



41, 17
49, 32
54, 44
61, 56
69, 68
...
446, 70
451, 80

17, 41
32, 49
44, 54
56, 61
68, 69
...
70, 446
80, 451

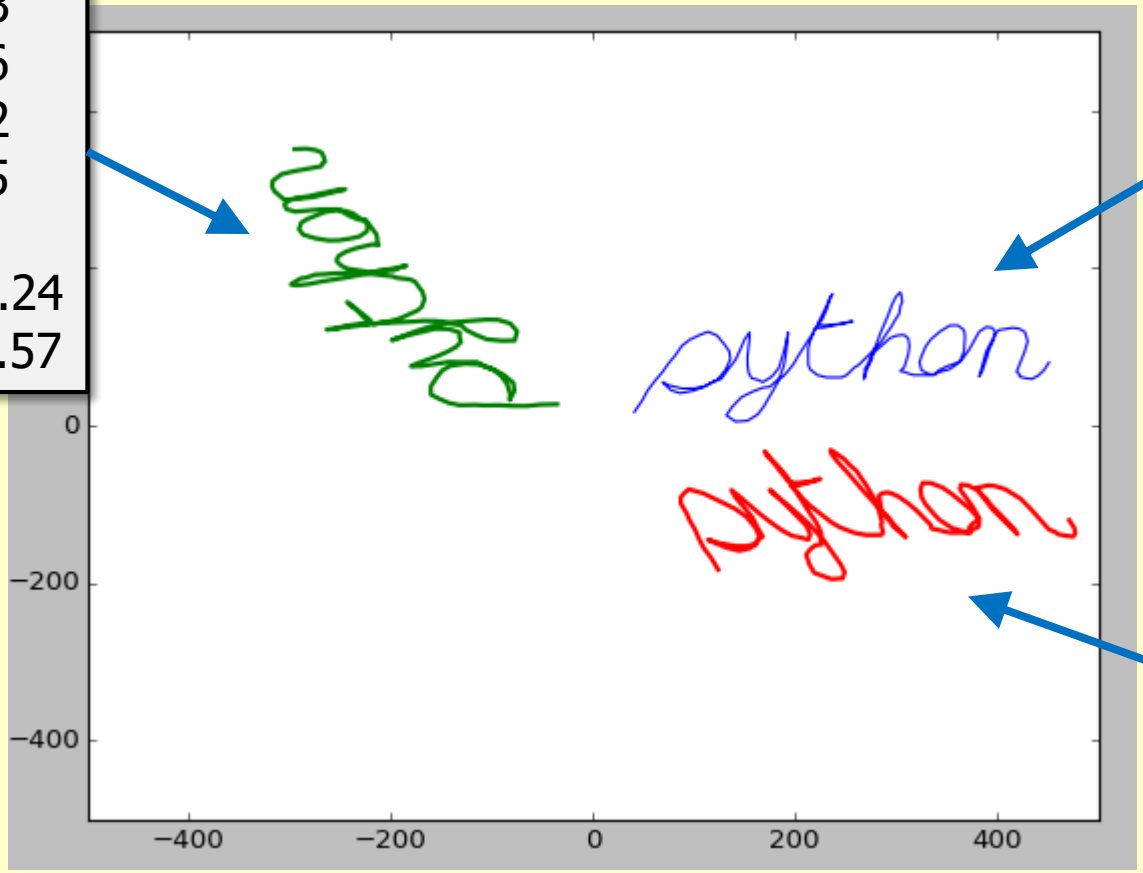
-17, -41
-32, -49
-44, -54
-56, -61
-68, -69
...
-70, -446
-80, -451

17, -41
32, -49
44, -54
56, -61
68, -69
...
70, -446
80, -451

x ใหม่คือ -y เดิม, y ใหม่คือ -x เดิม

การแปลงพิกัดของจุดแบบซับซ้อนขึ้น

-35.22, 27.00
-52.21, 26.43
-65.10, 24.76
-78.99, 24.82
-93.38, 25.75
...
-283.62, 351.24
-294.78, 350.57



41, 17
49, 32
54, 44
61, 56
69, 68
...
446, 70
451, 80

124, -183
117, -168
110, -156
105, -144
101, -132
...
476, -130
471, -120

วิธีแปลงพิกัด (x,y) ของจุดด้วยการคูณเมทริกซ์ 2x2

x, y
41, 17
49, 32
54, 44
61, 56
69, 68
...
446, 70
451, 80

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

x', y'
-41, 17
-49, 32
-54, 44
-61, 56
-69, 68
...
-446, 70
-451, 80

x, y
41, 17
49, 32
54, 44
61, 56
69, 68
...
446, 70
451, 80

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

x', y'
17, 41
32, 49
44, 54
56, 61
68, 69
...
70, 446
80, 451

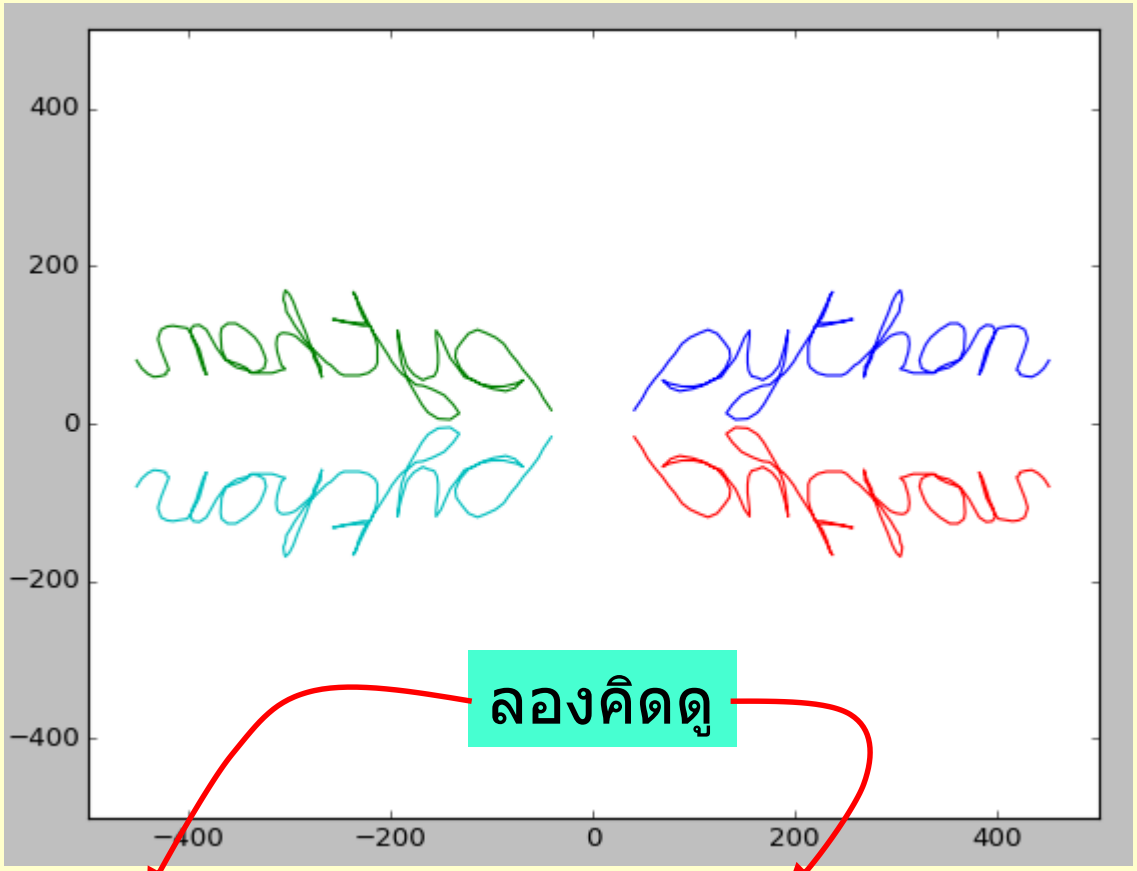
วิธีแปลงพิกัด (x,y) ของจุดด้วยการคูณเมทริกซ์ 2x2

-41, 17
-49, 32
-54, 44
-61, 56
-69, 68
...
-446, 70
-451, 80

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

41, 17
49, 32
54, 44
61, 56
69, 68
...
446, 70
451, 80



-41, -17
-49, -32
-54, -44
-61, -56
-69, -68
...
-446, -70
-451, -80

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x \\ -y \end{bmatrix}$$

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ -y \end{bmatrix}$$

41, -17
49, -32
54, -44
61, -56
69, -68
...
446, -70
451, -80

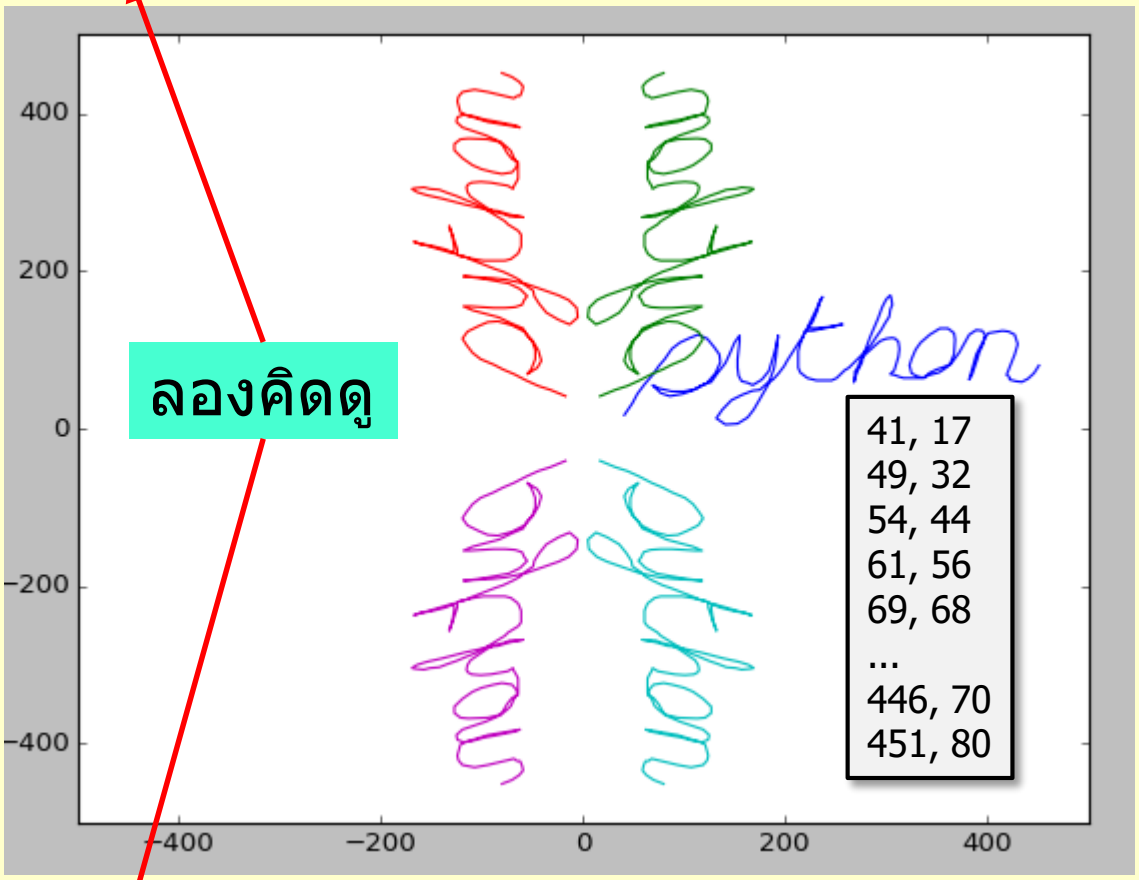
วิธีแปลงพิกัด (x,y) ของจุดด้วยการคูณเมทริกซ์ 2x2

-17, 41
-32, 49
-44, 54
-56, 61
-68, 69
...
-70, 446
-80, 451

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix}$$

17, 41
32, 49
44, 54
56, 61
68, 69
...
70, 446
80, 451



-17, -41
-32, -49
-44, -54
-56, -61
-68, -69
...
-70, -446
-80, -451

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix}$$

17, -41
32, -49
44, -54
56, -61
68, -69
...
70, -446
80, -451

โปรแกรมแปลงพิกัดของจุดต่าง ๆ ด้วยวิธีคูณเมทริกซ์

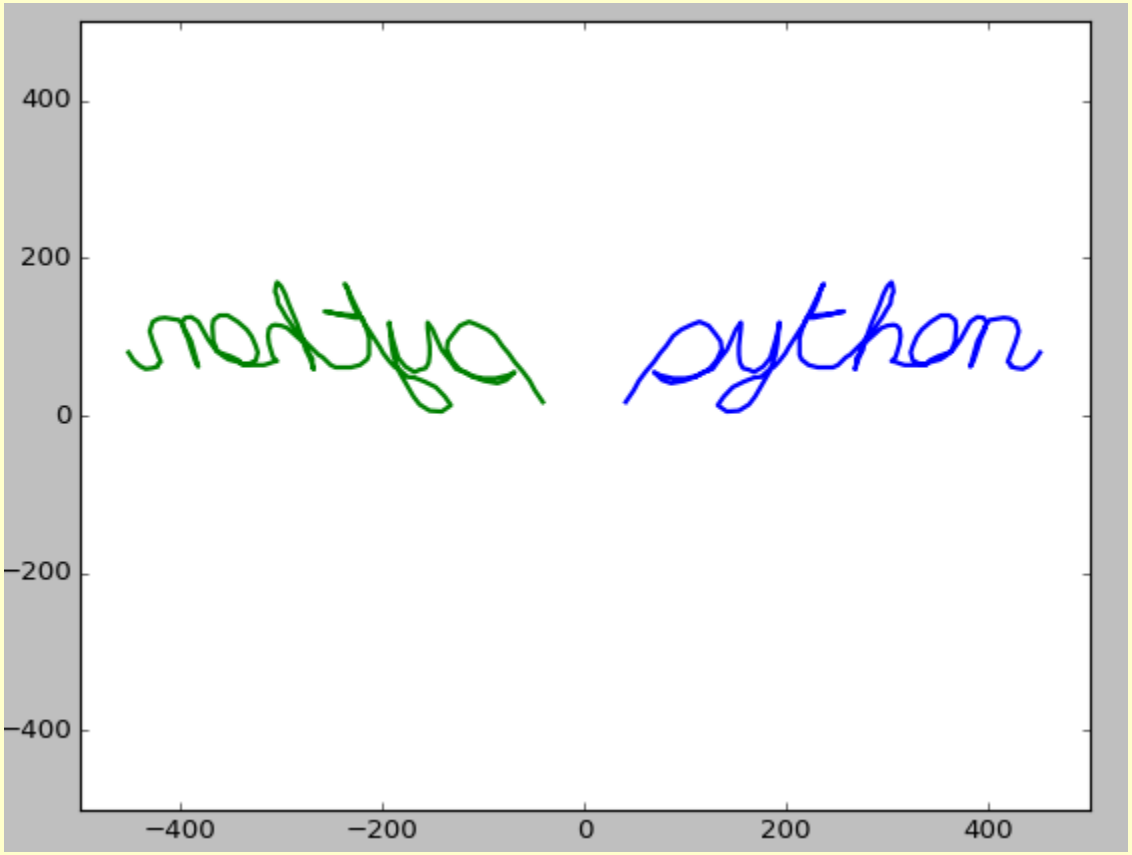
```
import numpy as np
import matplotlib.pyplot as plt

def transform(points, M):
    newp = []
    for (x,y) in points:
        newp.append( np.dot(M,(x,y)) )
    return np.array(newp)

xy = np.loadtxt("polyline.csv", delimiter=",")
# xy อาเรย์ 2 มิติ มิติแรกแทนแถว มิติที่สองแทนหลัก
M_flipHor = np.array([[-1,0],[0,1]])
newxy = transform(xy,M_flipHor)

plt.axis((-500,500,-500,500))
plt.plot( xy[:,0], xy[:,1] )          # วาดเส้นเดิม
plt.plot( newxy[:,0], newxy[:,1] ) # วาดเส้นใหม่
plt.show()
```

หยิบแต่ละจุดใน points มาคูณกับเมทริกซ์ M ได้พิกัดใหม่เก็บใส่ newp

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Horizontal Flip : [[-1,0],[0,1]]




```
M_hflip = np.array([[-1,0],[0,1]])
newxy = transform(xy,M)
```

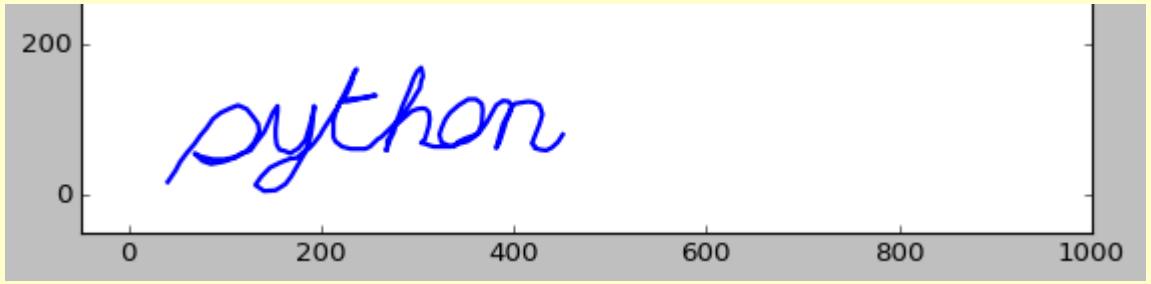
$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

เมทริกซ์การแปลงพิกัดแบบอื่น (scale)

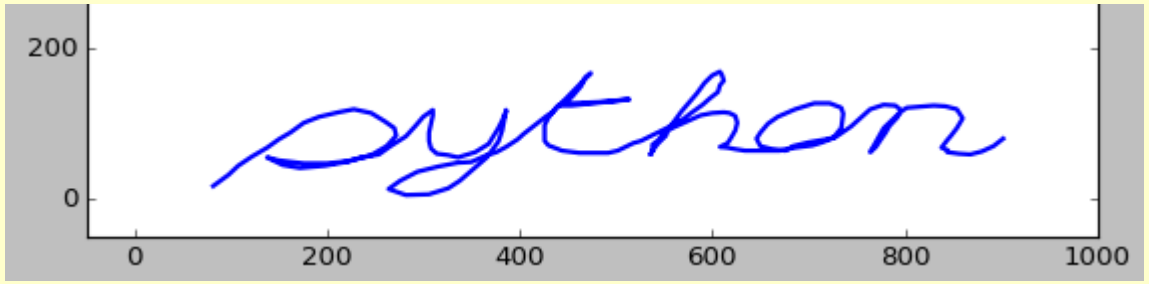
$$\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} S_x x \\ S_y y \end{bmatrix}$$

ปรับขนาดตามแนวนอนและแนวตั้ง

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$



$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2x \\ y \end{bmatrix}$$

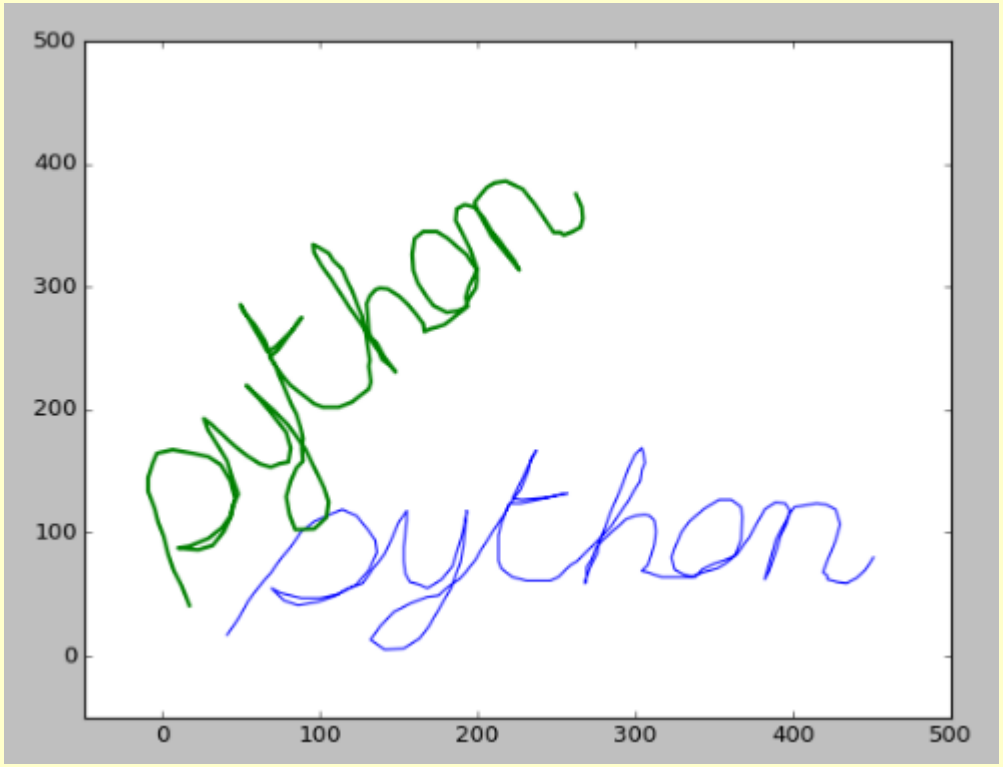


เมทริกซ์การแปลงพิกัดแบบอื่น (rotate)

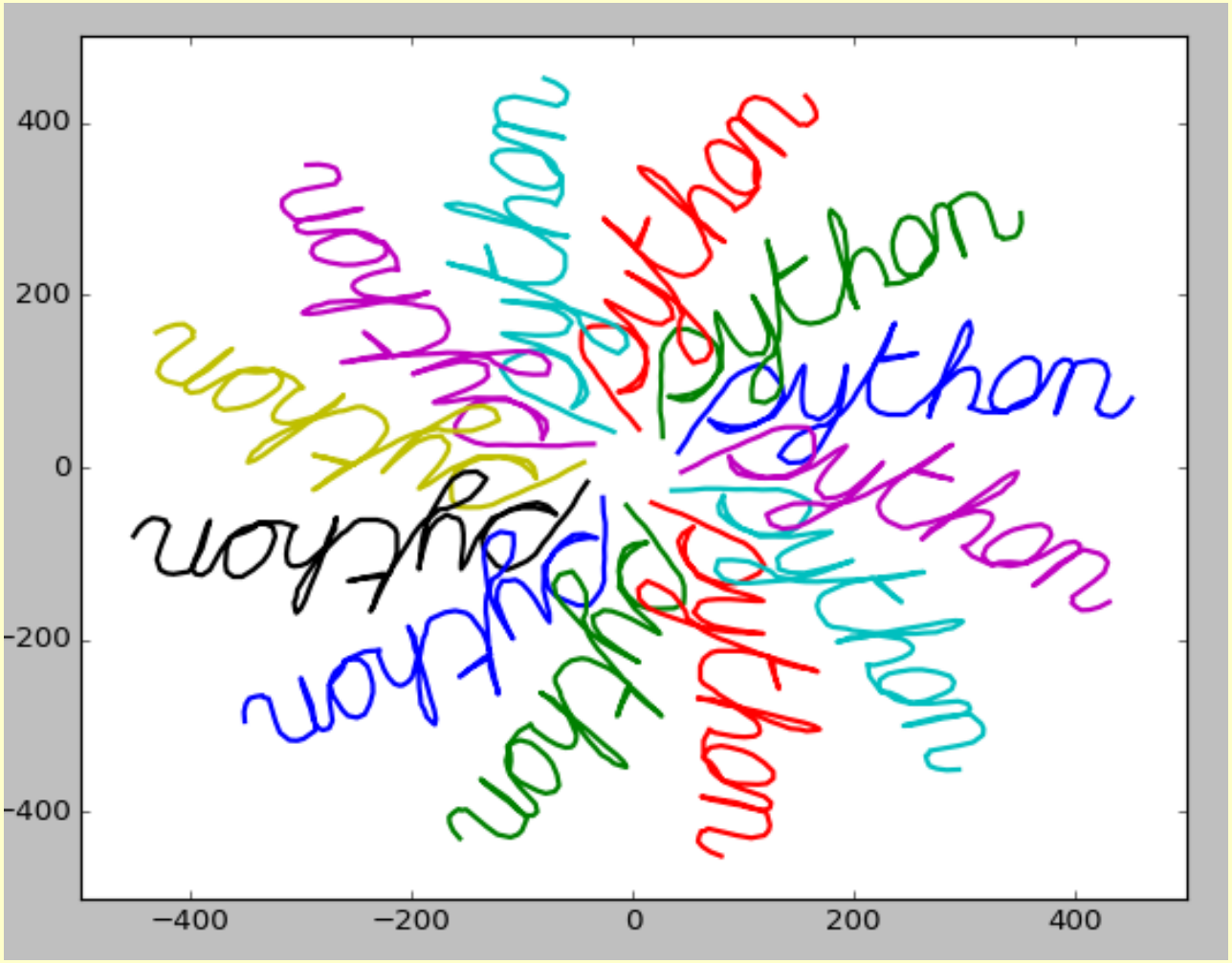
$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

$$\begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



ตัวอย่าง : วาดแล้วหมุนทีละ π/6 จำนวน 12 ครั้ง



โปรแกรมสาธิตการแปลงพิกัด

```
import numpy as np
import matplotlib.pyplot as plt
import math
def transform(points, M):
  newp = []
  for (x,y) in points:
     newp.append( np.dot(M,(x,y)) )
  return np.array(newp)

xy = np.loadtxt("polyline.csv", delimiter=",")
theta = math.pi/6
M = np.array([[math.cos(theta), -math.sin(theta)], \
              [math.sin(theta),  math.cos(theta)]])
plt.axis((-500,500,-500,500))

for i in range(12):
  plt.plot( xy[:,0], xy[:,1] )
  xy = transform(xy, M)

plt.show()
```

วาดแล้วหมุนไป $\pi/6$
จำนวน 12 ครั้ง

การแปลงพิกัดแบบไม่ต้องใช้วงวนแปลงทีละจุด

```
def transform(points, M):
    newp = []
    for (x,y) in points:
        newp.append( np.dot(M,(x,y)) )
    return np.array(newp)
```

```
def transform(points, M):
    return np.dot(points, M.T)
```

จริงหรือเนี่ย !!!

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} ax_k + by_k \\ cx_k + dy_k \end{bmatrix} \begin{matrix} \longleftarrow x_k' \\ \longleftarrow y_k' \end{matrix}$$

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix}^T = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix} = \begin{matrix} x' & y' \\ \begin{bmatrix} ax_1 + by_1 & cx_1 + dy_1 \\ ax_2 + by_2 & cx_2 + dy_2 \\ \vdots & \vdots \\ ax_n + by_n & cx_n + dy_n \end{bmatrix} \end{matrix}$$

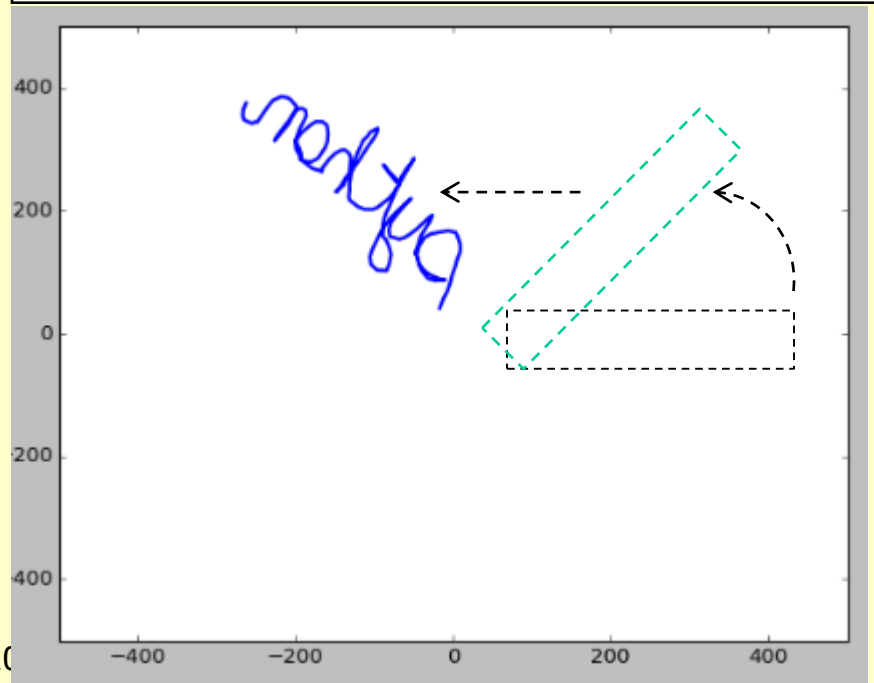
ตัวอย่างฟังก์ชันแปลงพิกัดแบบต่าง ๆ

```
def transform(points, M):
  return np.dot(points, M.T)

def flipHor(points):
  return transform(points,np.array([[-1,0],[0,1]]))

def flipVer(points):
  return transform(points,np.array([[1,0],[0,-1]]))

def scale(points, sx, sy):
  return transform(points,np.array([[sx,0],[0,sy]]))

def rotate(points, degree):
  rad = math.radians(degree)
  M = np.array([[math.cos(rad), -math.sin(rad)], \
                [math.sin(rad),  math.cos(rad)]])
  return transform(points,M)
```

ถ้าต้องแปลงพิกัดหลายขั้นตอนต่อเนื่องกัน

```
...
xy = np.loadtxt("polyline.csv", delimiter=",")
xy = rotate(xy, 45)
xy = flipHor(xy)
# xy = flipHor( rotate(xy,45) )
plt.axis((-500,500,-500,500))
plt.plot( xy[:,0], xy[:,1] )
plt.plot( newxy[:,0], newxy[:,1] )
plt.show()
```

เรียกฟังก์ชันซ้อนกันหลายครั้ง



$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \right)$$

ถ้าต้องแปลงพิกัดหลายขั้นตอนต่อเนื่องกัน

```
...
xy = np.loadtxt("polyline.csv", delimiter=",")
rad = math.pi/4
M = np.array([[math.cos(rad), -math.sin(rad)], \
              [math.sin(rad),  math.cos(rad)]])
M = np.dot([[-1,0],[0,1]], M) #rotate and flip hor.
xy = transform(xy, M)
plt.axis((-500,500,-500,500))
plt.plot( xy[:,0], xy[:,1] )
plt.show()
```

คูณเมทริกซ์การแปลง 2x2 ให้เสร็จก่อน

$$\left(\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\cos\frac{\pi}{4} & -\sin\frac{\pi}{4}\\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4}\end{bmatrix}\right)\begin{bmatrix}x\\ y\end{bmatrix}$$

เร็วกว่า

$$\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}\left(\begin{bmatrix}\cos\frac{\pi}{4} & -\sin\frac{\pi}{4}\\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4}\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}\right)$$

สรุป

- นิสิตต้อง
  - รู้จักการแปลงพิกัด (เชิงเส้น) ของจุดเบื้องต้น
  - เข้าใจการคูณเมทริกซ์ของ numpy ด้วยคำสั่ง np.dot(A,B)
  - เขียนวิธีการแปลงพิกัดของจุดเบื้องต้นด้วยคำสั่งของ numpy