3장. NumPy와 Matplotlib

세부목차

NumPy and MatPlotLib

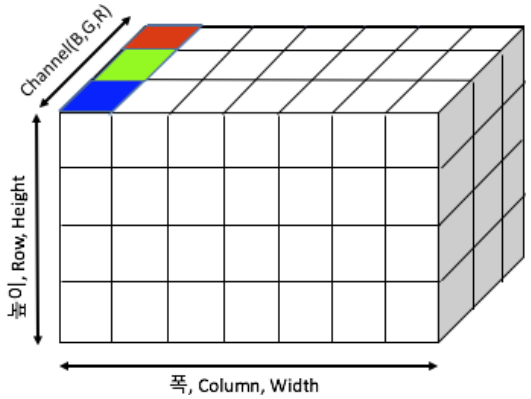
❖ NumPy

- http://www.numpy.org/
- https://docs.scipy.org/doc/numpy/genindex.html
- Fundamental package for sientific computing with Python
  - 강력한 N-차 배열 객체
  - 정교한 Broadcasting 함수
  - C/C++, Fortran 통합 도구
  - 유용한 선형대수, 푸리에 변환, 난수발생 기능
- OpenCV-Python ver 2.3+
  - 이전에는 자체 Internal Data Structure 사용
  - 이미지 데이타 자료구조로 NumPy 채용
  - cv2 패키지로 변경
  - import cv2
  - import numpy as np

NumPy and MatPlotLib

❖ NumPy and Image

- ndarray
  - ndim : n차원 (axes)
  - shape : 각 차원의 크기, tuple
  - size : 전체 요소의 갯수, shape의 곱
  - dtype : 요소의 data type
  - itemsize : 각 요소의 바이트 크기
  - data : 요소를 담고 있는 버퍼, 실제로 사용하지 않음




```
import numpy as np
import cv2

img = cv2.imread('../img/blank_500.jpg')
type(img)     # <class 'numpy.ndarray'>
print(img.ndim)   #3
print(img.shape) #(500,500,3,)
print(img.size) #750000
print(img.dtype) #'uint8'
print(a.itemsize) # 1, 각 용소의 크기
```

❖ NumPy

- 생성 방법
  - 값으로 생성
    - array()
  - 초기 값으로 생성
    - empty(), zeros(), ones(), full()
  - 기존 배열로 생성
    - empty_like(), zeros_like(), ones_like(), full_like()
  - 순차적인 값으로 생성
    - arange()
  - 난수로 생성
    - random.rand(), random.randn()

❖ NumPy

- 값으로 생성
  - numpy.array(list [, dtype]) : 지정한 값들로 NumPy 배열 생성
    - list : 배열 생성에 사용할 값을 갖는 파이썬 리스트 객체
    - dtype : 데이타 타입, 생략하면 값에 의해 자동 결정
      - int8, int16, int32, int64 : 부호 있는 정수
      - uint8, uint16, uint32, uint64 : 부호 없는 정수
      - float16, float32, float64, float128 : 부동 소수점을 갖는 실수
      - complex64, complex128, complex256 : 부동 소수점을 갖는 복소수
      - bool : 불리언

❖ NumPy

- 값으로 생성

```
import numpy as np
a = np.array([1,2,3,4])    # 정수를 갖는 리스트로 생성
b = np.array([[1,2,3,4],    # 2차원 리스트로 생성
                        [5,6,7,8]])
c = np.array([1,2,3.14,4])  # 정수와 소수점이 혼재된 리스트
d = np.array([1,2,3,4], dtype=np.float64)   # dtype을 지정해서 생성
print(a, a.dtype, a.shape)  # --- ①
print(b, b.dtype, b.shape)  # --- ②
print(c, c.dtype, c.shape)  # --- ③
print(d, d.dtype, d.shape)  # --- ④
```

❖ NumPy

- 크기와 초기 값으로 생성
  - np.empty (tuple [, dtype])
    - 모든 요소가 초기화 되지 않은 , tuple 크기
    - np.empty((2,3))
    - np.empty((2,3), dtype=np.int16)
  - np.zeros(tuple [, dtype])
    - 모든 요소가 0이고 tuple 크기
    - np.zeros((3,4))
    - np.zeros((2,3,4) , dtype=np.int16)
  - np.ones (tuple [, dtype])
    - 모든 요소가 1이고, tuple 크기
    - np.ones((3,4))
    - np.ones((2,3,4) , dtype=np.int16)
  - np.full(shape, fill_value [, dtype])
    - fill_value로 초기화된 배열 생성

NumPy and MatPlotLib

NumPy

- 크기로 생성

```
import numpy as np


a = np.zeros( (2,3))
b = np.zeros( (2,3,4), dtype=np.uint8)

c = np.ones( (2,3), dtype=np.float32)
d = np.empty( (2,3))  #not initialized, garbage value
e = np.empty( (2,3), dtype=np.int16)
f  = np.fill(255)

print(a, a.dtype, a.shape)
print(b, b.dtype, b.shape)
print(c, c.dtype, c.shape)
print(d, d.dtype, d.shape)
print(f, f.dtype, f.shape)
```

❖ NumPy

- 기존 배열로 생성
  - np.empty_like(array [, dtype])
    - array와 같은 shape과 dtype의 초기화 되지 않은 배열 생성
  - zeros_like(array [, dtype])
    - 0(영, zero)로 초기화된 array와 같은 shape과 dtype의 배열 생성
  - ones_like(array [, dtype])
    - 1로 초기화된 array와 같은 shape과 dtype의 배열 생성
  - ones_like(array, fill_value [, dtype])
    - fill_value로 초기화된 array와 같은 shape과 dtype의 배열 생성

NumPy and MatPlotLib

❖ NumPy

- 기존 배열로 생성

```
import numpy as np
img = cv2.imread('../img/girl.jpg')

a = np.empty_like(img)
b = np.zeros_like(img)
c = np.ones_like(img)
d = np.full_like(img, 255)

print(a, a.dtype, a.shape)
print(b, b.dtype, b.shape)
print(c, c.dtype, c.shape)
print(d, d.dtype, d.shape)
```

NumPy and MatPlotLib

❖ NumPy

- Sequence와 난수로 생성
  - numpy.arange([start=0,] stop [, step=1, dtype=float64])
  - start : 시작 값
  - stop : 종료 값, 범위에 포함하는 수는 stop-1 까지
  - step : 증가 값
  - dtype : 데이타 타입
- numpy.random.rand([d0 [, d1 [..., dn]]]) :
  - d0, d1..dn : shape, 생략하면 난수 한개 반환
  - 0과 1 사이의 무작위 수
- numpy.random.randn([d0 [, d1 [..., dn]]]) :
  - 평균=0, 분산=1, 표준정규 분포를 따르는 무작위 수

NumPy and MatPlotLib

❖ NumPy

- Sequence와 난수로 생성

```
import numpy as np

a = np.arange(5) b
= np.arange(5.0)
c = np.arange(3,9,2)

e = np.random.rand()
f = np.random.randn()
g = np.random.rand(2,3)
h = np.random.randn(2,3)
```

❖ NumPy

- Data Type 변경
  - a.astype('typename') , a.astype(np.dtype)
  - np.uintXX(), np.intXX(), np.floatXX(), np.complexXX()

```
a = np.arange(10)
print(a, a.dtype)

b = a.astype('float32')
print(b, b.dtype)

c = np.uint8(b)
print(c, c.dtype)

d = c.astype(np.float64)
print(d, d.dtype)
```

❖ NumPy

- 차원 변경
  - np.reshape(array, newshape [, order])
  - arr.reshape(newshape [, order])
    - array : 재정렬할 대상 array
    - newshape : 새로운 행렬크기, tuple
      - -1 : 해당 컬럼은 크기 미지정
    - order : {'C', 'F', 'A'} C:C-like, F:Fotran-like, A:F or C
  - np.ravel(array)
    - 1차원으로 변경
  - arr.T : 전치 배열

NumPy and MatPlotLib

NumPy

- 차원 변경

```
a=np.arange(6) #[0 1 2 3 4 5] (6,)
b=a.reshape(2,3)   #[[0 1 2] [3 4 5]] (2, 3)
c= np.arange(24).reshape(2,3,4)
#[[[ 0  1  2  3]...
 [[12 13 14 15]...
  [20 21 22 23]]] (2, 3, 4)

d = np.arange(100).reshape(-1, 5)
# [[0 1 2 3 4]
   [5 6 7 8 9]
    ...
   [95 96 97 98 99]]

e = np.arange(100).reshape(2, -1)
#[[0 1 2 3 ... 49]
  [50 51 52 ...99]]

f  = np.ravel(c) #[0 1 2 3 4 5 6 7 8 .... 21 22 23]
```

❖ NumPy

- Broadcasting 연산
  - 스케일러와 연산
  - 배열과 연산, 두 배열의 shape이 같아야 한다.

```
a = np.array([10,20,30,40])
b = np.arange(1,5)

c = a + b
d = a - b
e = a * b
f = a / b

g = a + 5
h = a * 2
i = b ** 2
```

```
[10 20 30 40]
[1 2 3 4]

[11 22 33 44]
[ 9 18 27 36]
[ 10  40  90 160]
[ 10.  10.  10.  10.]

[15 25 35 45]
[20 40 60 80]
[ 1  4  9 16]
```

NumPy and MatPlotLib

❖ NumPy

- 행렬 연산
  - 행렬 합 : + (더하기 연산)
  - 행렬 곱 : np.dot(a, b), a.dot(b)
  - 전치 행렬 : a.T , a.transpose()

```
a=np.arange(6).reshape(2,3) #[[0 1 2]
                            #[3 4 5]]
b = np.array([1,2,3])       #[1 2 3]
c = a + b                   #[[1 3 5]
                            #[4 6 8]]
e = np.dot(a , b)           #[ 8 26]
f = a.T                     # [[0 3]
                            #  [1 4]
                            #  [2 5]]
a.dot(b)                    #  [ 8 26]
```

❖ NumPy

- Indexing/Slicing
  - 1차원
    - a[0] : 특정 요소
    - a[1 : 5] : 1~4번째 요소
    - a[5:] : 5번째부터 끝까지
    - a[:5] : 처음 부터 4번째 까지
    - a[:] : 모든 요소
  - 2차원
    - a[2, 3] : 2행 3열 요소
    - a[2:4, 3] : 2~3행의 3열 요소
    - a[2, 1:4] : 2행의 1~3열 요소
    - a[2, :] : 2행의 모든 열 요소
    - a[:, 2] : 모든 행의 2열 요소
    - a[1:3, :] : 1~2행의 모든 열 요소

NumPy and MatPlotLib

- **NumPy**
  - Indexing/Slicing

```
a = np.arange(10) # [0 1 2 3 4 5 6 7 8 9]


print(a[5])   # 5
print(a[1:5])# [1 2 3 4]
print(a[5:]) # [5 6 7 8 9]
print(a[:5]) # [0 1 2 3 4]
print(a[:]). # [0 1 2 3 4 5 6 7 8 9]

b = np.arange(16).reshape(4,-1)

print(b)
print(b[2, 2])
print(b[2, :])
print(b[:, 2])
print(b[1:3, :])
```

❖ NumPy

- Fancy Indexing
  - 조건을 만족하는 데이타만 추출하거나 초기화
  - 조건 연산: > , >=, <, <=, ==, !=
  - Bool type (True/False)
  - a[a > 4]
  - a[a > 4] = 0
  - 색인 배열
    - a[[0,2,4]]
    - b = [1,2,3]
    - a[b]

❖ NumPy

- Fency Indexing

```
a = np.arange(10) #[0 1 2 3 4 5 6 7 8 9]
b = a > 4
print(b) #[False False False False False True True True True True]
print(a[b]) #[5 6 7 8 9]
print(a[a>4])#[5 6 7 8 9]
a[a>4] =0
print(a) # [0 1 2 3 4 0 0 0 0 0]
names = np.array(['John', 'Tom', 'Lee', 'Tom'])
scores = np.random.rand(4,4) * 50 + 50 print(scores)
print(names=='Tom')
print(scores[names=='Tom',   :]) #Tom의 점수만 추출
```

❖ NumPy

- Fency Indexing – 색인 배열

```
>>> a = np.arange(0,100,  10)
array([0,10,20,30,40,50,60,70,80,90])
>>> b = [1,3,5,7]
array([1,3,5,7])
>>> a[b]
array([10,30,50,70])
```

❖ NumPy

- Merge
  - numpy.hstack(arrays) : arrays 배열을 수평으로 병합
  - numpy.vstack(arrays) : arrays 배열을 수직으로 병합
  - numpy.concatenate(arrays, axis=0) : arrays 배열을 지정한 축 기준으로 병합
  - numpy.stack(arrays, axis=0) : arrays 배열을 새로운 축으로 병합
    - arrays : 병합 대상 배열, 튜플

NumPy and MatPlotLib

- **NumPy**
  - Merge

```
>>> a = np.arange(4).reshape(2,2)
>>> a
array([[0, 1],
       [2, 3]])
>>> b = np.arange(10, 14).reshape(2,2)
>>> b
array([[10, 11],
       [12, 13]])
>>> np.vstack( (a,b) )
array([[ 0,  1],
       [ 2,  3],
       [10, 11],
       [12, 13]])
>>> np.hstack( (a,b) )
array([[ 0,  1, 10, 11],
       [ 2,  3, 12, 13]])
>>> np.concatenate((a,b), 0)
array([[ 0,  1],
       [ 2,  3],
       [10, 11],
       [12, 13]])
>>> np.concatenate((a,b), 1)
array([[ 0,  1, 10, 11],
       [ 2,  3, 12, 13]])
```

NumPy and MatPlotLib

- **NumPy**
  - Merge

```
>>> a =np.arange(12).reshape(4,3)
>>> b =np.arange(10, 130, 10).reshape(4,3)
>>> a
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
>>> b
array([[ 10,  20,  30],
       [ 40,  50,  60],
       [ 70,  80,  90],
       [100, 110, 120]])
>>> c =np.stack( (a,b), 0)
>>> c.shape
(2, 4, 3)
>>> c
   array([[[  0,   1,   2],
           [  3,   4,   5],
           [  6,   7,   8],
           [  9,  10,  11]],

          [[ 10,  20,  30],
           [ 40,  50,  60],
           [ 70,  80,  90],
           [100, 110, 120]]])
```

NumPy

- splitting
  - np.hsplit(array, indice) = np.split(axis=1)
  - np.vsplit(array, indice) = np.split(axis=0)
  - np.split(array, indice, axis=0)

```
a = np.arange(12) # [ 0  1  2  3  4  5  6  7  8  9 10 11]

b = np.hsplit(a, 3)
# [array([0, 1, 2, 3]), array([4, 5, 6, 7]), array([ 8,  9, 10, 11])]

c = np.hsplit(a, (3,6))
# [array([0, 1, 2]), array([3, 4, 5]), array([ 6,  7,  8,  9, 10, 11])]
```

❖ NumPy Search

- numpy.where(condition [, t, f]) : 조건에 맞는 요소를 찾기
  - return : 튜플, 검색 조건에 맞는 요소의 인덱스 또는 변경된 값으로 채워진 배열
  - condition : 검색에 사용할 조건식,
  - t, f : 조건에 따라 지정할 값 또는 배열, 배열의 경우 조건에 사용한 배열과 같은 shape
    - t : 조건에 맞는 값에 지정할 값이나 배열
    - f : 조건에 틀린 값에 지정할 값이나 배열
- numpy.nonzero(array) : array에서 요소중에 0(영,zero)가 아닌 요소의 인덱스들을 반환, 튜플
- numpy.all(array [, axis]) : array의 모든 요소가 True 인지 검색
  - array : 검색 대상 배열
  - axis : 검색할 기준 축, 생략하면 모든 요소 검색, 지정하면 축 갯수별로 결과 반환
- numpy.any(array [, axis]) : array의 어느 요소이든 True가 있는지 검색

NumPy and MatPlotLib

- **NumPy**
  - Search

```
>>> a = np.arange(10, 20)
>>> a
array([10, 11, 12, 13, 14, 15, 16, 17,    18, 19])
>>> np.where(a > 15)
(array([6, 7, 8,  9]),)
>>> np.where(a > 15, 1, 0)
array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1])
>>> a
array([10, 11, 12,  13, 14, 15, 16, 17, 18, 19])
>>> np.where(a>15,  99, a)
array([10, 11, 12,  13, 14, 15, 99, 99, 99, 99])
>>> np.where(a>15,  a, 0)
array([ 0,  0,  0,  0,  0,  0, 16, 17, 18, 19])
>>> a
array([10, 11, 12,  13, 14, 15, 16, 17, 18, 19])
```

NumPy and MatPlotLib

NumPy

- Search

```
>>> b = np.arange(12).reshape(3,4)
>>> b

   array([[ 0,   1,   2,  3],
          [ 4,   5,   6,  7],
          [ 8,   9,  10, 11]])
>>> coords = np.where(b>6)
>>> coords
(array([1, 2, 2, 2, 2]), array([3, 0, 1, 2,  3]))
>>> np.stack( (coords[0], coords[1]), -1)
    array([[1, 3],
           [2, 0],
           [2, 1],
           [2, 2],
           [2, 3]])
>>> z = np.array([0,1,2,0,1,2])
>>> np.nonzero(z)
(array([1, 2, 4, 5]),)
```

NumPy and MatPlotLib

- **NumPy**
  - Search

```
>>> a = np.arange(10)
>>> b = np.arange(10)
>>> a
array([0, 1, 2, 3, 4,  5, 6, 7, 8, 9])
>>> b
array([0, 1, 2, 3, 4,  5, 6, 7, 8, 9])
>>> a==b
array([ True,  True,  True,  True,  True,  True,  True,  True,
True,
        True])
>>> np.all(a==b)
True
>>> b[5] = -1
>>> a
array([0, 1, 2, 3, 4,  5, 6, 7, 8, 9])
>>> b
array([ 0,  1,  2,  3,  4, -1,  6,  7,  8,  9])
>>> np.all(a==b)
False
>>> np.where(a==b)
(array([0, 1, 2, 3, 4, 6, 7, 8, 9]),)
>>> np.where(a!=b)
(array([5]),)
>>>
```

❖ NumPy

- 기초통계 함수
  - numpy.sum(array [, axis]) : 배열의 합계 계산
  - numpy.mean(array [, axis]) : 배열의 평균 계산
  - numpy.amin(array [, axis]) : 배열의 최소 값 계산
  - numpy.min(array [, axis]) : numpy.amin()과 동일
  - numpy.amax(array [, axis]) : 배열의 최대 값 계산
  - numpy.max(array [, axis]) : numpy.amax()와 동일
    - array : 계산의 대상 배열
    - axis : 계산 기준 축, 생략하면 모든 요소를 대상

NumPy and MatPlotLib

❖ NumPy

- 기초통계 함수

```
a = np.arange(12).reshape(-1,4)

print(np.sum(a)) #np.sum(-1)
print(np.sum(a, 0))
print(np.sum(a, 1))
print(np.sum(a, (0,)))

print(np.min(a))
print(np.min(a, 0))
print(np.min(a, 1))

print(np.max(a))
print(np.max(a, 0))
print(np.max(a, 1))

print(np.mean(a))
print(np.mean(a, 0))
print(np.mean(a, 1))
```

NumPy and MatPlotLib

❖ NumPy

- Creating Gray Image
  - n x m
  - n : height, rows
  - m : width, columns
  - value : 0~255 (dtype=np.uint8)
- Creating Color Image
  - n x m x 3
  - n : height, rows
  - m : width, columns
  - 3 : color channels, Default = [Blue, Green, Red]
  - value : 0~255 (dtype=np.uint8)

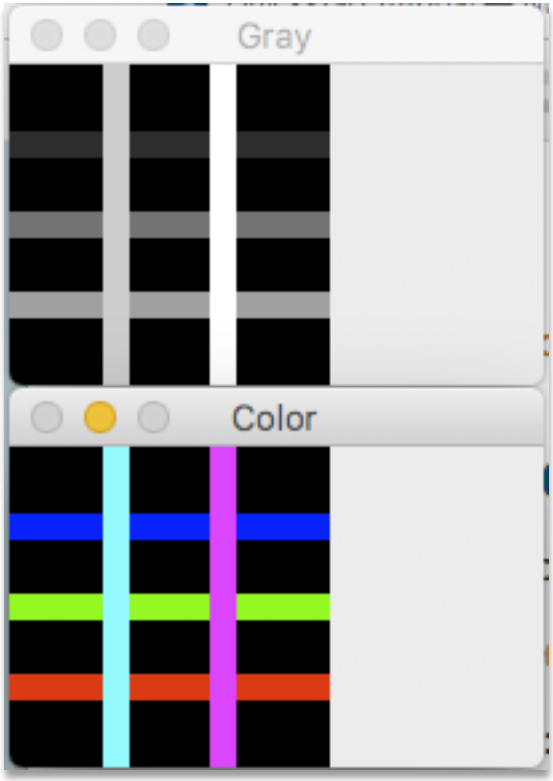
NumPy and MatPlotLib

❖ NumPy

- NumPy Image

[예제3-1] 배열로 체크무늬 그레이 스케일 이미지 생성(np_gray.py)

```
import cv2
import numpy as np

img = np.zeros((120,120), dtype=np.uint8) # 120x120 2차원 배열 생성, 검은색 흑백 이미지
img[25:35, :] = 45 # 25~35행 모든 열에 45 할당
img[55:65, :] = 115 # 55~65행 모든 열에 115 할당
img[85:95, :] = 160 # 85~95행 모든 열에 160 할당
img[:, 35:45] = 205 # 모든행 35~45 열에 205 할당
img[:, 75:85] = 255 # 모든행 75~85 열에 255 할당
cv2.imshow('Gray', img)
if cv2.waitKey(0) & 0xFF == 27:
cv2.destroyAllWindows()
```



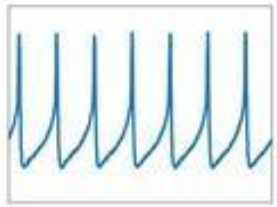
세부목차

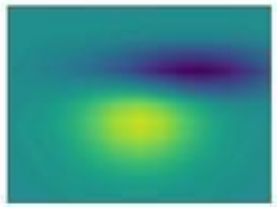
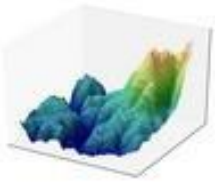
1. NumPy
2. **MatPloLib**

NumPy and MatPlotLib

❖ Matlotlib

- https://matplotlib.org/
- Python 2D plotting library
- 다양한 그래프, 도표 표시
- 한 화면에 여러 이미지 표시
- Installation
  - pip install matplotlib 또는 pip3 install matplotlib
  - sudo apt-get install python3-matplotlib
- checking
  - import matplotlib
  - matplotlib.__version__
- import pyplot
  - import matplotlib.pyplot as plt

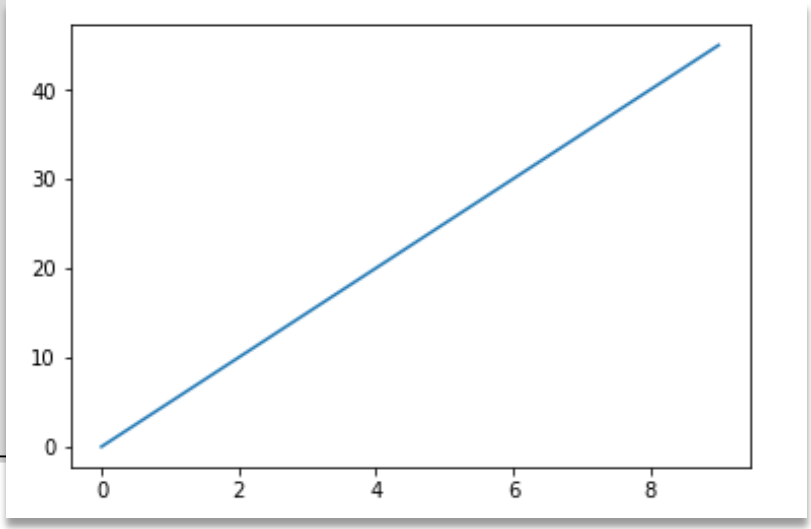


NumPy and MatPlotLib

- **Matplotlib**
  - plt.plt(x, y)
  - plt.show()

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

x = np.arange(10)
y = x * 5
x, y  plt.plot
(x,y)   plt.sh
ow()
```

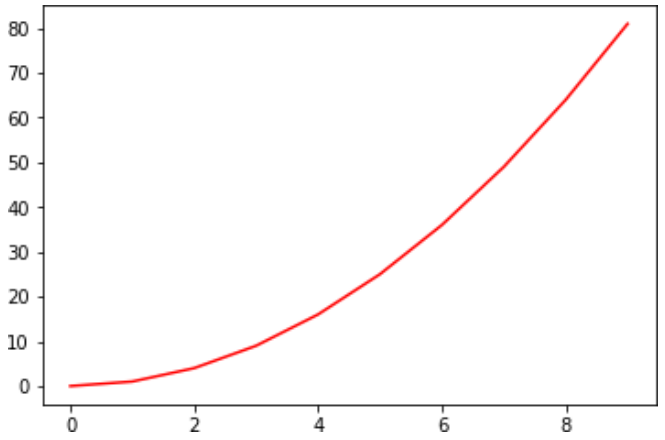


❖ Matplotlib

- color : plt.plot(x, y, 'r')
  - b : blue
  - g : green
  - r : red
  - c : cyan
  - m : magenta
  - y : yellow
  - k : black
  - w : white

[예제 3-5] plot의 색 지정(plt_color.py)

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(10) # 0,1,2,3,4,5,6,7,8,9
y = x **2 # 0,1,4,9,16,25,36,49,64,81
plt.plot(x,y, 'r') # plot 생성 --- ①
plt.show() # plot 화면에 표시
```



[그림3-6] [예제3-5]의 실행 결과

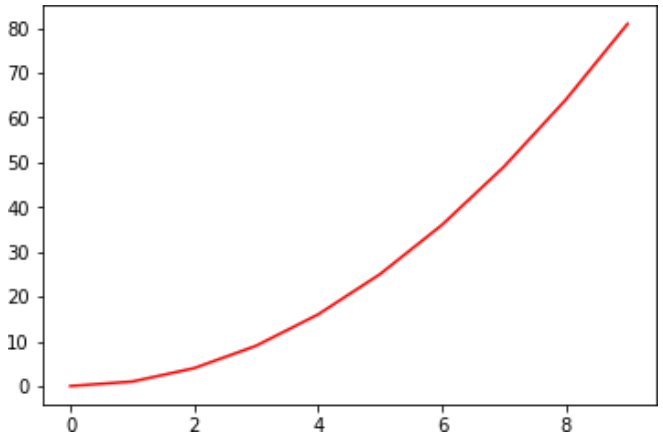
NumPy and MatPlotLib

❖ Matplotlib

- style : plt.plot(x, y, '--g')

char	style	char	style
-	solid(*)	--	dashed
-.	dash-dot	:	dotted
.	point	,	pixel
o	circle	v	triangle dn
^	trangle up	<	traingle left
>	triangle right	1	small tri dn
2	small tri up	3	small tri left
4	small tri rigt	s	square
p	pentagon	*	star
h	hexagon	+	plus
D	diamond	x	ex

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

x = np.arange(10)
y = x**2
plt.plot(x,y,'--g')
plt.show()
```

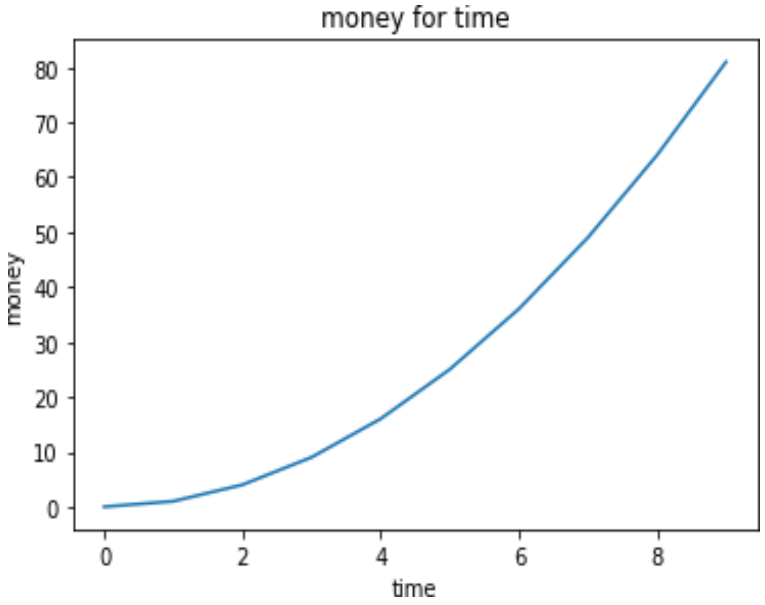


NumPy and MatPlotLib

❖ Matplotlib

- label
  - plt.xlabel('text')
  - plt.ylabel('text')
  - plt.title('text')

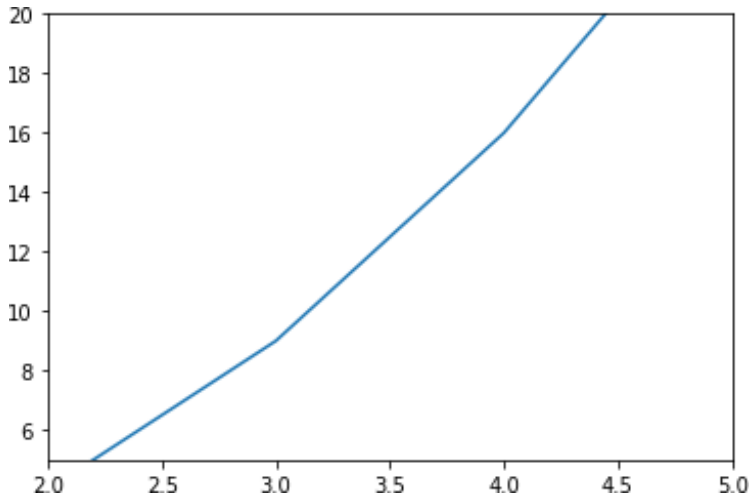
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x **2

plt.plot(x,y)
plt.xlabel('time')
plt.ylabel('money')
plt.title('money for time')
plt.show()
```



❖ Matplotlib

- limit
  - 전체 데이타 중에 특정 영역만 표시
  - plt.xlim(x,y)
  - plt.ylim(x,y)

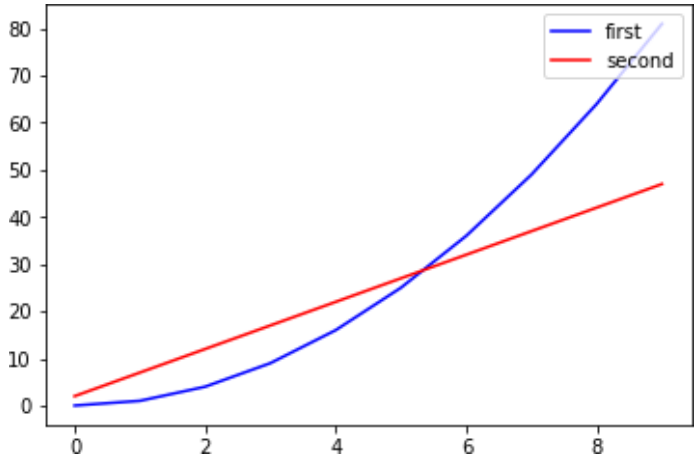
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x **2
plt.plot(x,y)
plt.xlim(2,5)
plt.ylim(5,20)
plt.show()
```



NumPy and MatPlotLib

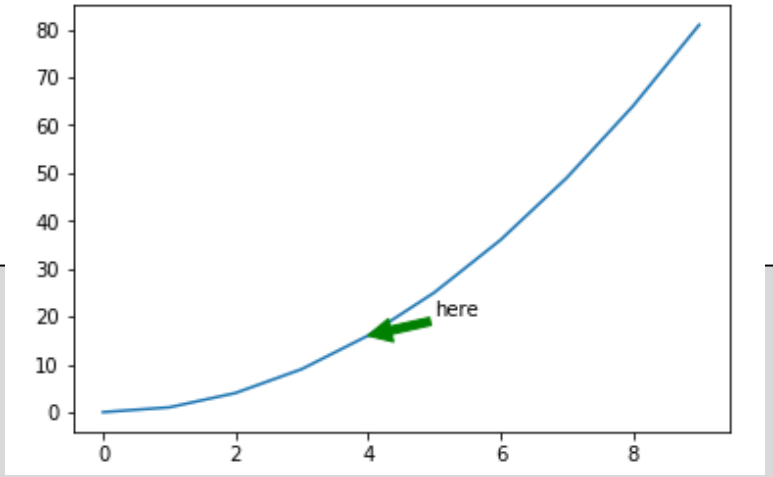
- **Matplotlib**
  - legend
    - 범례
    - plt.plot(x, y, 'b', label='text')
    - plt.legend(loc='upper right')
      - upper left
      - upper center
      - upper right
      - center left
      - center
      - center right
      - lower left
      - lower center
      - lower right
      - best
      - right

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

x = np.arange(10)
y = x **2
y2 = x*5 + 2

plt.plot(x,y, 'b', label='first')
plt.plot(x,y2, 'r', label='second')
plt.legend(loc='upper right')
plt.show()
```



NumPy and MatPlotLib

- **Matplotlib**
  - annotate
    - plt.annotate('text')



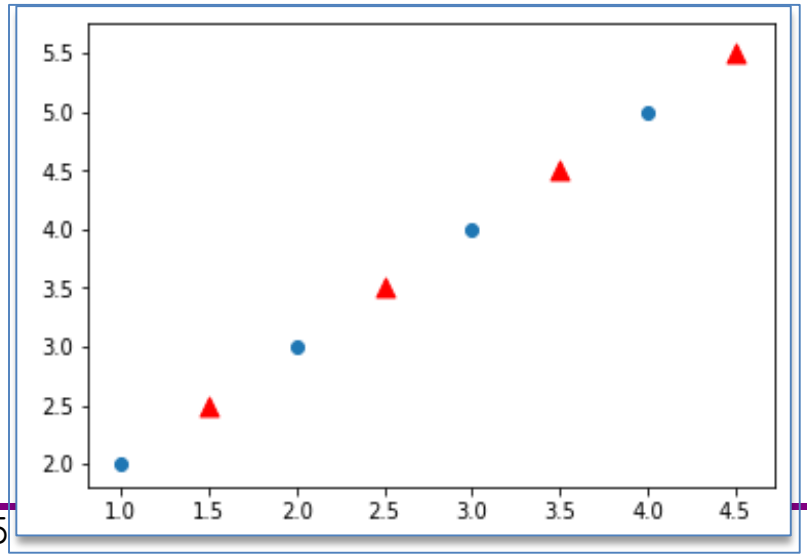

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x**2

plt.plot(x,y)
plt.annotate('here', xy=(4,16), xytext=(5,20), arrowprops={'color':'green'})
plt.show()
```

NumPy and MatPlotLib

Matplotlib

- scatter
  - plt.scatter(x, y, s=None, c=None, market=None)

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(1,5)  y = np.arange(2, 6)
x2 = np.arange(1.5,5.5)  y2 = np.arange(2.5,
6.5)  plt.scatter(x, y)
plt.scatter(x2, y2, s=80, c='r',  marker='^')
plt.show()
```

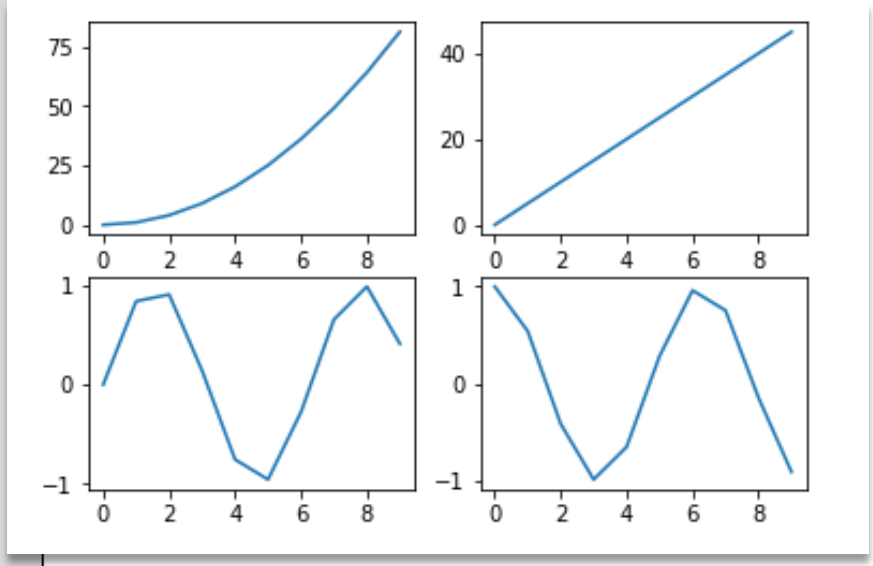


❖ Matplotlib

- subplot
  - plt.subplot(r,c,idx)
  - plt.subplot(3digit)

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(10)
plt.subplot(2,2,1) #2행 2열 중에 1번째
plt.plot(x,x**2)
plt.subplot(2,2,2) #2행 2열 중에 2번째
plt.plot(x,x*5)
plt.subplot(223) #2행 2열 중에 3번째
plt.plot(x, np.sin(x))
plt.subplot(224) #2행 2열 중에 4번째
plt.plot(x,np.cos(x))
plt.show()
```

[예제 3-7] subplot(plt_subplt.py)



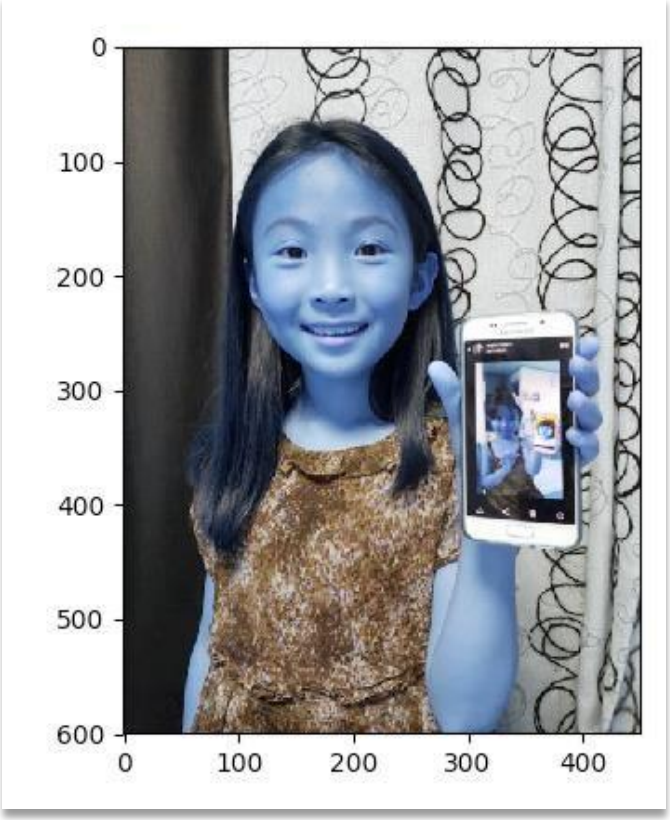
[그림3-8] [예제3-7]의 실행 결과

NumPy and MatPlotLib

❖ Matplotlib

- OpenCV Image
  - plt.imshow(img)
  - OpenCV = GBR, Plot = RGB

```
import cv2
from matplotlib import pyplot as plt
img = cv2.imread('../img/girl.jpg')
plt.imshow(img[:,:,::-1]) # 이미지 컬러 채
널 변경해서 표시 --- ①
plt.xticks([]) # x좌표 눈금 제거 ---②
plt.yticks([]) # y좌표 눈금 제거 ---③
plt.show()
```



[예제 3-9] 컬러 채널을 변경한 이미지 출력(plt_imshow_rgb.py)

❖ Matplotlib

- OpenCV Image
  - change BGR to RGB
    - numpy slicing
    - cv2.split(), cv2.merge()
    - cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
img1 = cv2.imread('../img/model.jpg')
img2 = cv2.imread('../img/model2.jpg')
img3 = cv2.imread('../img/model3.jpg')
```

[예제 3-10] 여러 이미지 동시 출력(plt_imshow_subplot .py)

```
plt.subplot(1,3,1) #1행 3열 중에 1번째
plt.imshow(img1[:,:,(2,1,0)])
plt.xticks([]); plt.yticks([])
plt.subplot(1,3,2) #1행 3열 중에 2번째
plt.imshow(img2[:,:,(2,1,0)])
plt.xticks([]); plt.yticks([])
plt.subplot(1,3,3) #1행 3열 중에 3번째
plt.imshow(img3[:,:,(2,1,0)])
plt.xticks([]); plt.yticks([])
plt.show()
')
```

[예제 3-10] 여러 이미지 동시 출력(plt_imshow_subplot .py)

❖ Matplotlib

- OpenCV Image
  - change BGR to RGB

[그림3-11] [예제3-10]의 실행 결과