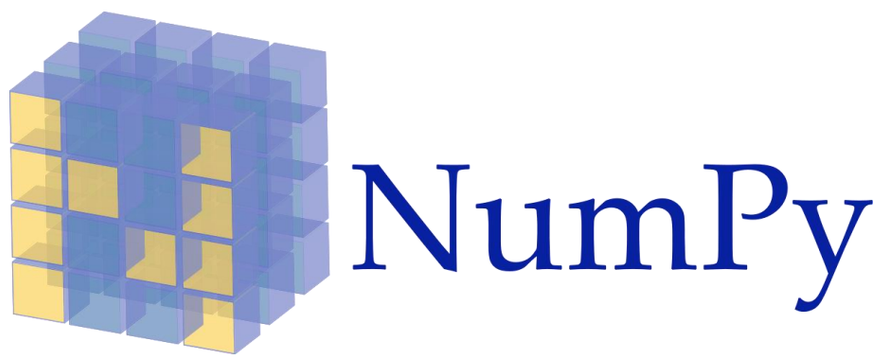
한양대학교

NumPy

Hosung Jo

Day 3

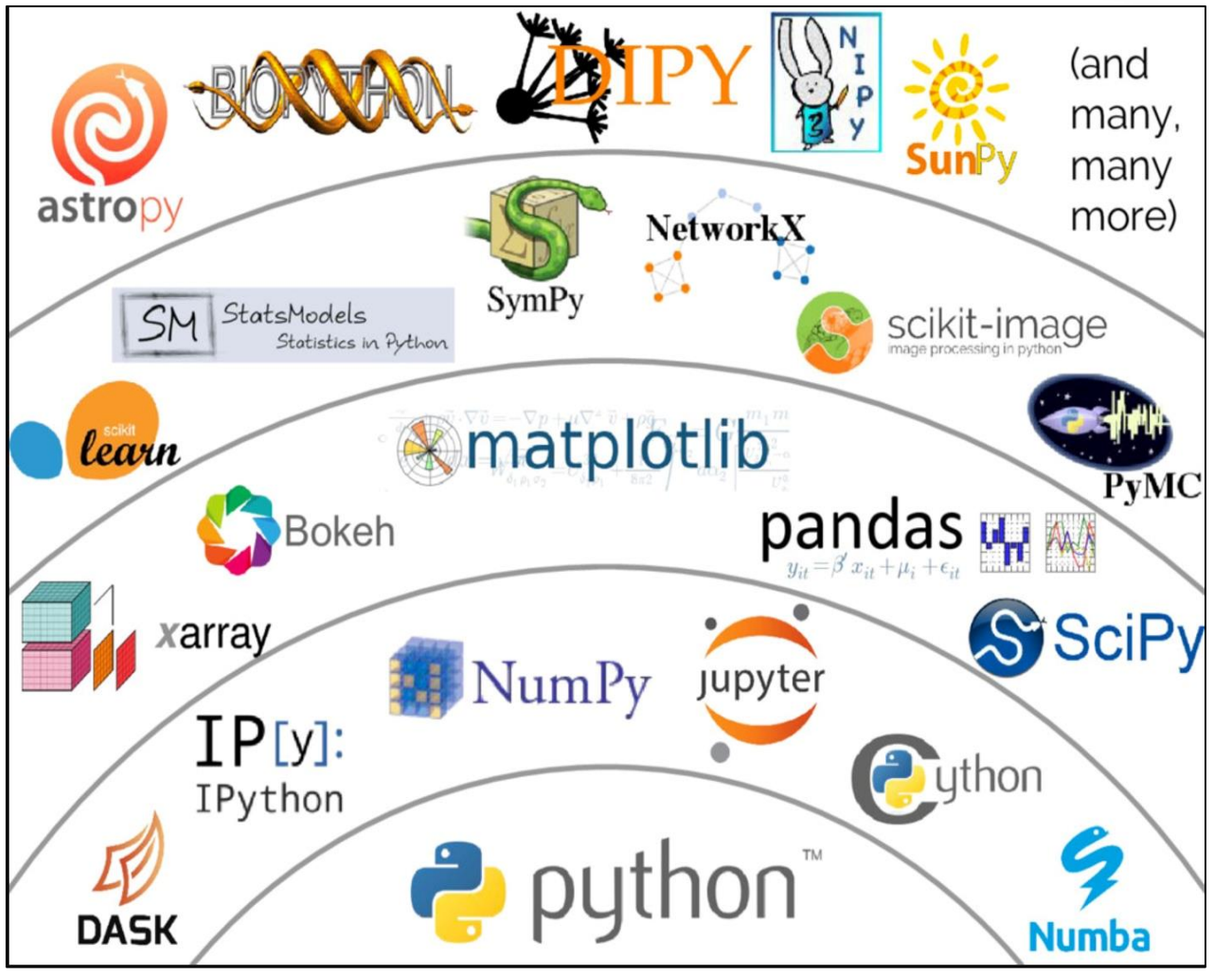
- **Review**
  - Python, NumPy, and ndarray
  - NumPy Basics operations
  - NumPy Functions, Axis



- **Matplotlib**
  - Bar, line, scatter

Python

- **Python**
  - 오픈소스 기반의 하이레벨 컴퓨터 언어
  - 간결하여 이해가 쉽고, 활용 분야가 넓음
- 상호 운용성, 단순성과 동적 특성
- **<u>강력한 Third-party 라이브러리,</u>**
  NumPy, SciPy, Pandas, matplotlib, Scikit-image



NumPy

NumPy = Number + Python

- 파이썬 라이브러리
  - (NumPy v1.18., http://www.numpy.org)
- 행렬이나 다차원 배열 처리, 계산과학 분야의 복잡한 연산을 지원
  - ndarray class
  - 원소단위로 계산하지 않고 배열 단위로 계산을 지원
  - Loops 없이 연산하여 코딩의 양이 줄어듬 →병렬처리 가능 (multi-core)

배열(Arrays) 생성

명령어	설명	사용 예
모듈 임포트	import numpy as np	
np.array	주어진 값을 ndarray를 생성한다.	np.array([1,2,3])
np.ones	행렬을 1로 채워준다.	np.ones((3,6))
np.ones_like	대상과 같은 shape의 행렬을 1로 채워준다.	np.ones_like(ar)
np.zeros	행렬을 0으로 채워준다.	np.zeros((3,6))
np.zero_like	대상과 같은 shape의 행렬을 0으로 채워준다.	np.zeros_like(ar)
np.arange	주어진 범위의 숫자들로 행렬을 만들어준다.	np.arange(1,100,5)
np.linspace	주어진 범위를 개수만큼 분할한다.	np.linspace(1,100,5)
np.logspace	주어진 범위를 개수만큼 로그로 분할한다.	np.logspace(1,100,5)
np.random.random	0~1사이의 실수로 행렬을 생성한다.	np.random.random((1,10))

ndarray 명령어

명령어	설명	사용법
shape	행렬의 모양을 확인	np.shape(ar)
reshape	행렬의 모양을 변경	ar.reshape(3,5)
ndim	행렬의 차수(dimension, 깊이)를 확인	np.ndim(ar)
len	길이를 확인	len(ar), len(ar[0])
type	형식을 확인	type(ar)
T	행렬을 치환	ar.T
Slicing	1차원 배열: 리스트와 동일 N차원 배열: ‘,’ 로 구별하여 연결	ar[1:2] ar[1:, 2:, 3:]

명령어	설명	사용법
min	행렬 또는 부분 행렬에서 최소값을 반환	np.min(ar)
max	행렬 또는 부분 행렬에서 최대값을 반환	np.max(ar)
argmin	최소값의 위치를 반환	np.argmin(ar)
argmax	최대값의 위치를 반환	np.argmax(ar)
sum	행렬 또는 부분 행렬의 합을 반환 axis 설정에 따라 행합 또는 열합을 계산	np.sum(ar) np.sum(ar, axis=0)
mean	sum을 전체 개수로 나눈 값, 평균	np.mean(ar)
median	평균이 아닌 중간 값	np.median(ar)
dot	두 행렬의 벡터 내적	np.dot(ar1, ar2) a@b
var	분산을 반환	np.var(ar)
std	표준편차를 반환	np.std(ar)
sort	정렬을 수행, axis 값에 따라	np.sort(ar) np.sort(ar, axis=1)
shuffling	아이템의 순서를 뒤섞어줌	np.random.shuffle(ar)

Arrays

행렬곱 (내적)

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} (a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}) & (a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32}) \\ (a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31}) & (a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32}) \\ (a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31}) & (a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32}) \\ (a_{41}b_{11} + a_{42}b_{21} + a_{43}b_{31}) & (a_{41}b_{12} + a_{42}b_{22} + a_{43}b_{32}) \end{bmatrix}$$

- 행렬과 행렬의 곱
- 행렬과 스칼라의 곱과 구별, a*b

행렬곱 (내적)

- 가중치 합 (Weighted Sum) $w_1x_1 + \cdots + w_Nx_N = \sum_{i=1}^{N} w_i x_i$
- 가중치 평균 (Weighted average) $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{N}\mathbf{1}_N^T x$
- 유사도 검사 (Similarity check)

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Review Practice

- np.arange와 random.shuffle을 이용하여 100개의 정수가 무작위 순서로 저장되어있는 (10,10) 크기의 행렬을 생성하시오.

- 위에서 생성된 배열을 정렬하고, 가장 작은 값과 가장 큰 값이 행렬의 어디에 있는지 확인해 보시오.

- 다음과 같은 행렬이 주어졌을 때, 전체 합과 평균, 중간값을 구하시오.

```
array([[ 5, 13, 20, 24, 14, 24],
       [30,  6, 19, 30,  4,  2],
       [15,  1, 16, 15,  1,  1],
       [26, 26, 16,  8, 24, 15],
       [ 4, 27, 17,  2, 14, 19]])
```

Array

- **Deep Copy vs. Shallow copy**
  - Deep Copy, 새롭게 생성
  - Shall Copy, 같은 곳을 참조,
    - 복사된 곳에서 값을 수정하면 원본도 수정

Array

- **Deep Copy vs. Shallow copy**
  - Deep Copy, 새롭게 생성
  - Shall Copy, 같은 곳을 참조,
    - 복사된 곳에서 값을 수정하면 원본도 수정
- **A와 B를 실행하여 어느 것이 Deepcopy인지 확인하시오.**

A

```
ar = np.arange(1,10).reshape(3,3)
ar
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
Dar = ar
Dar
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
Dar[0][2] = 1000
Dar
```

```
array([[   1,    2, 1000],
       [   4,    5,    6],
       [   7,    8,    9]])
```

B

```
ar = np.arange(1,10).reshape(3,3)
ar
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
Dar = ar.copy()
Dar
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
Dar[0][2] = 1000
Dar
```

```
array([[   1,    2, 1000],
       [   4,    5,    6],
       [   7,    8,    9]])
```

Array

- **Text File load and save**
  - 행렬을 텍스트파일로 저장하거나 불러오는 기능

```
ar
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
np.savetxt('File10.txt', ar)
```

```
ar2=np.loadtxt('File10.txt')
ar2
```

```
array([[1., 2., 3.],
       [4., 5., 6.],
       [7., 8., 9.]])
```

Array

- **Text File load and save**
  - 행렬을 텍스트파일로 저장하거나 불러오는 기능
  - 0부터 연속된 정수로 이루어진 (2, 10, 10) 행렬A 를 생성하고 텍스트파일에 저장한 뒤 텍스트 파일을 열어 내용을 확인하시오.

```
ar
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
np.savetxt('File IO.txt', ar)
```

```
ar2=np.loadtxt('File IO.txt')
ar2
```

```
array([[1., 2., 3.],
       [4., 5., 6.],
       [7., 8., 9.]])
```

Array

- **Text File load and save**
  - 행렬을 텍스트파일로 저장하거나 불러오는 기능

  - price.txt 와 qnt.txt 내용을 불러온 후 다음을 계산하시오.
    - price에는 각 물건의 하나당 가격이 저장되어 있다.
    - qnt에는 구매할 물건의 개수가 저장되어 있다.
    - 각 물건의 구매 가격을 계산하여 calc.txt라는 파일에 저장하시오.

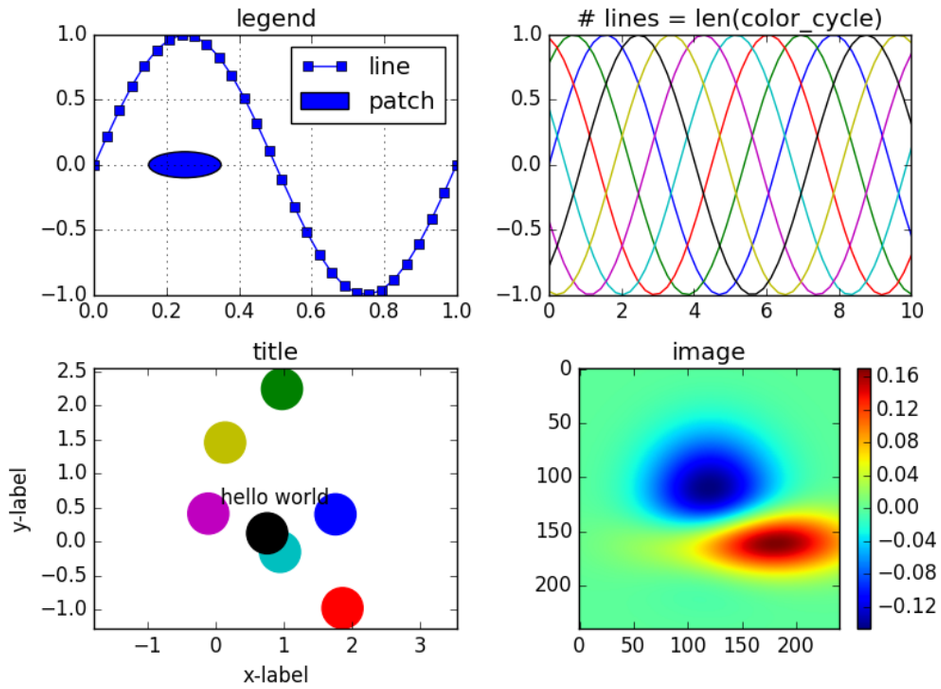
Array

```
array([[  1.,  70.,  85.,  95., 100., 100.],
       [  2., 100.,  85.,  90.,  90.,  95.],
       [  3.,  95.,  95.,  80.,  75.,  75.],
       [  4.,  65.,  75.,  70.,  90., 100.],
       [  5.,  90.,  80.,  85.,  95.,  95.],
       [  6.,  90.,  90.,  75.,  75.,  80.],
       [  7.,  60.,  90.,  80.,  40.,  45.],
       [  8.,  85.,  90.,  75.,  60.,  90.],
       [  9.,  90.,  85.,  85.,  90.,  85.]])
```

- **Text File load and save**
  - 행렬을 텍스트파일로 저장하거나 불
  - score.txt 파일의 내용을 불러온 후 다음을 계산하시오.
    - 자료는 앞에서부터 학번, 영어, 국어, 수학, 체육, 컴퓨터의 시험점수이다.
    - 전체 평균점수가 가장 높은 학생과 가장 낮은 학생은 각각 몇번 학생인가?
    - 국어, 영어, 수학 점수의 평균이 가장 높은 학생은 누구인가?
    - 국어, 영어, 수학 점수에 2배 가중치를 준다면 가장 점수가 높은 학생은 누구인가?

Matplotlib

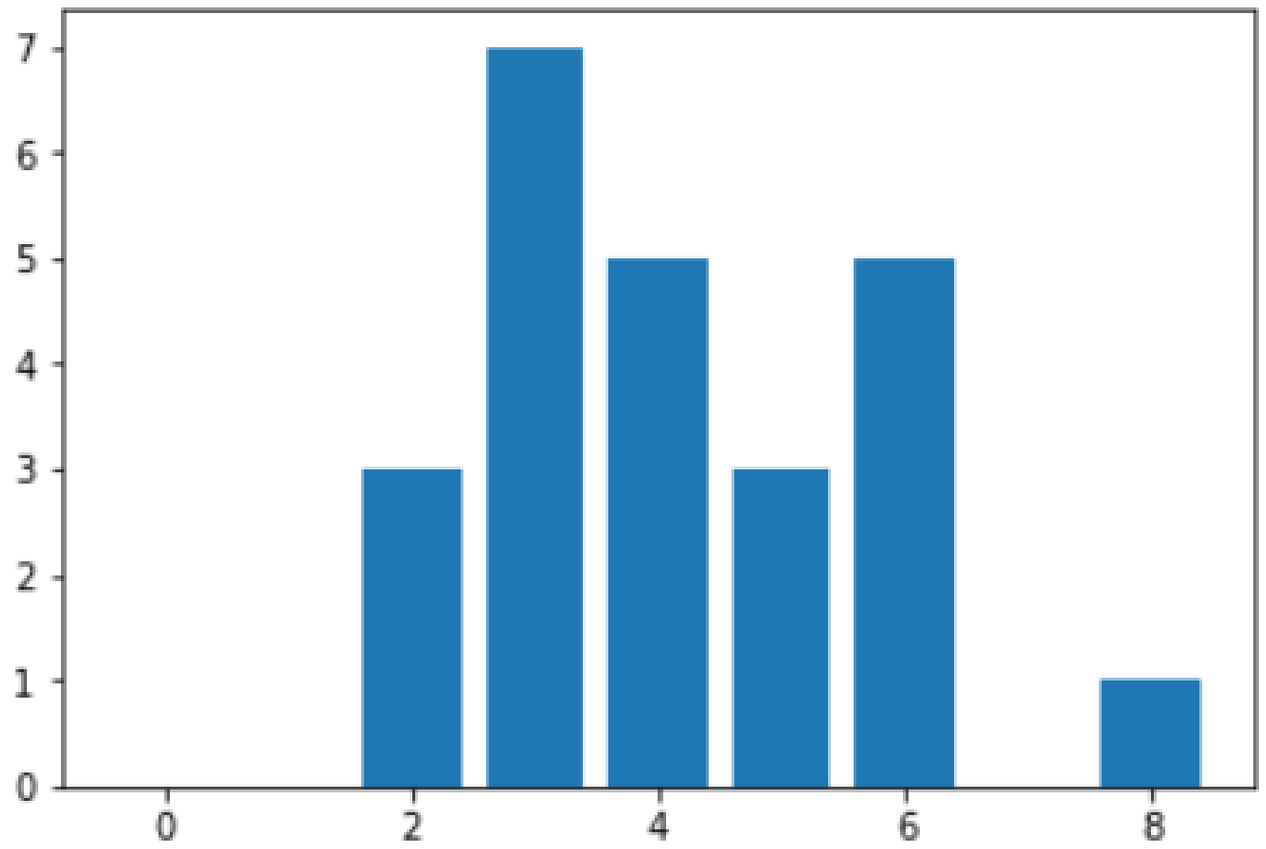
- **Matplotlib**
  - Python에서 matlab과 같은 chart나 plot를 만드는 라이브러리
  - 데이터 시각화를 쉽게 구현
  - 예제는 아래에서 확인가능
    https://matplotlib.org/gallery.html
  - 하위에 pylab 모듈에서 대부분 기능 지원



Matplotlib

- **bar**
  - 막대그래프
  - plt.bar()로 작성한 후 plt.show()로 jupyter 노트북에서 확인 가능

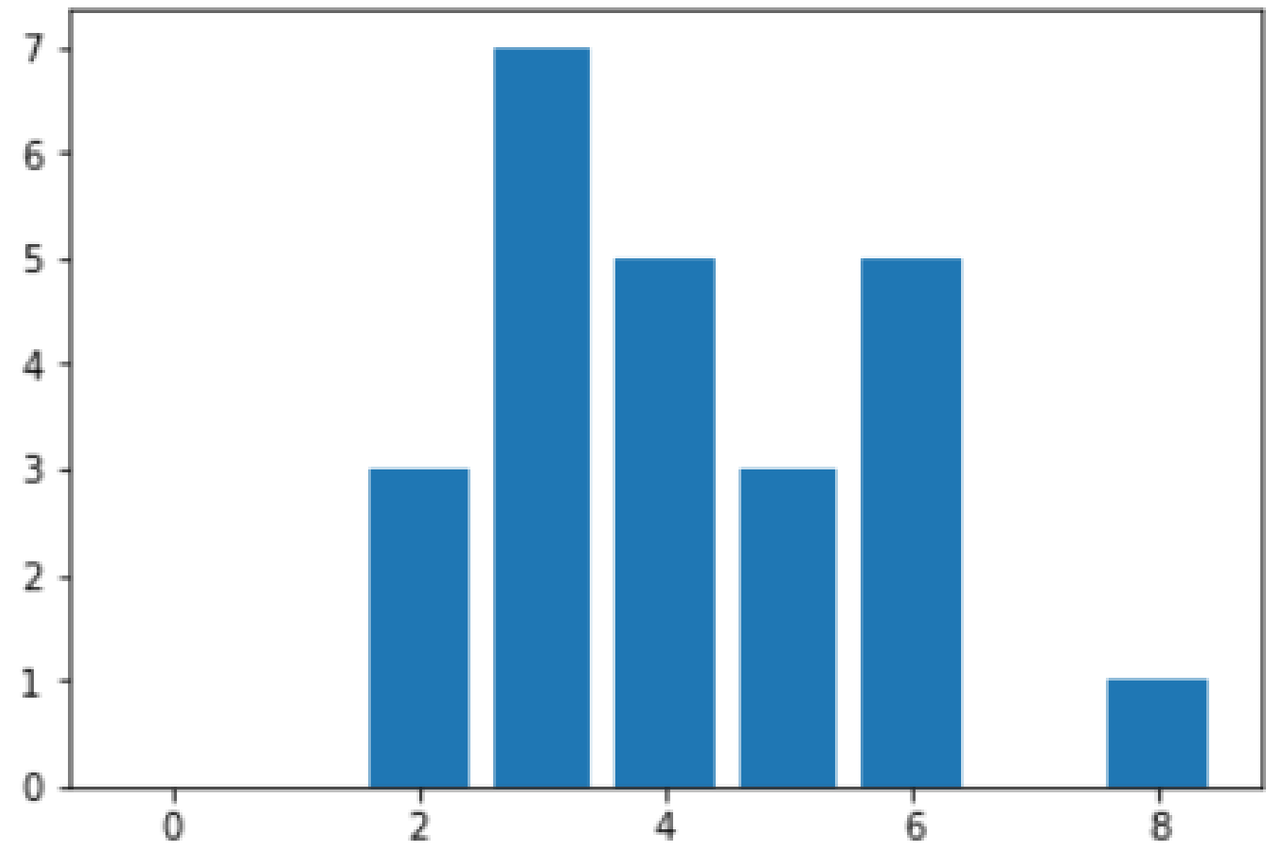
```
y = np.array([0, 0, 3, 7, 5, 3, 5, 0, 1])
x = np.arange(0,len(y))
plt.bar(x,y)
plt.show()
```



Matplotlib

- **bar**
  - 막대그래프
  - plt.bar()로 작성한 후 plt.show()로 jupyter 노트북에서 확인 가능

```
y = np.array([0, 0, 3, 7, 5, 3, 5, 0, 1])
x = np.arange(0,len(y))
plt.bar(x,y)
plt.show()
```



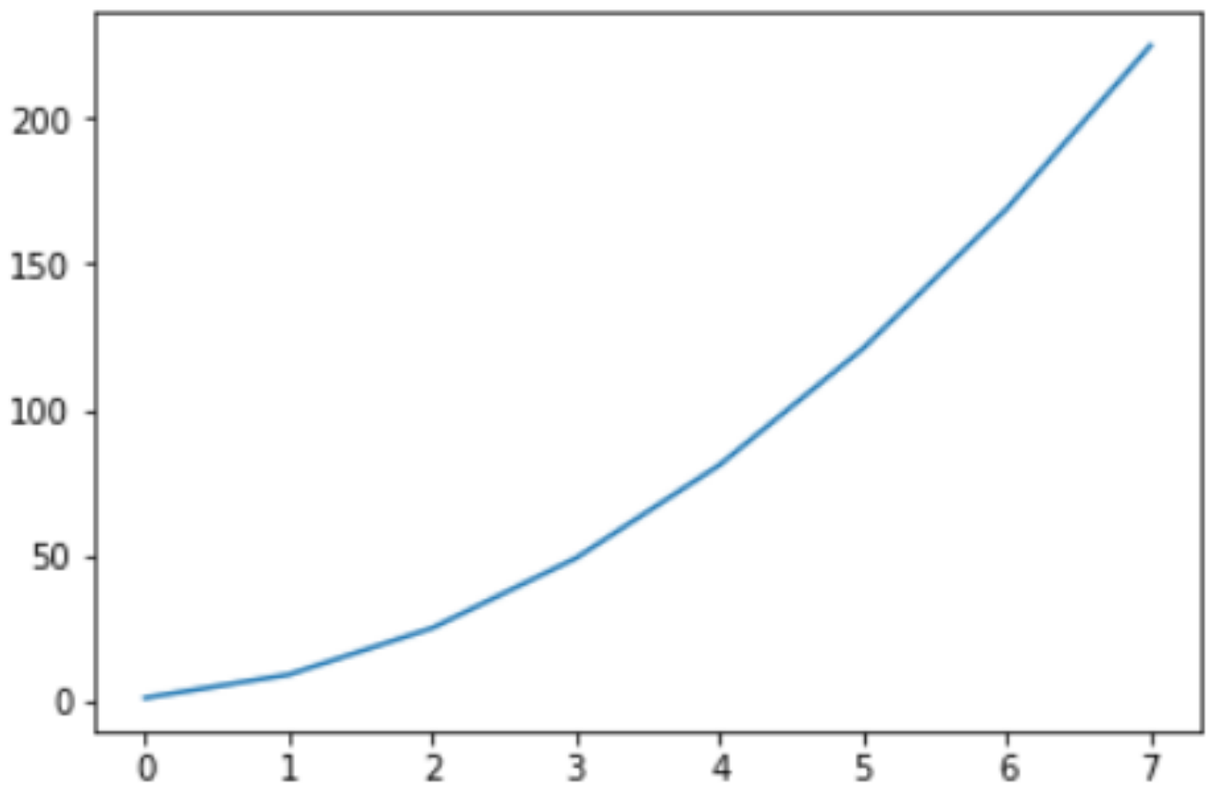
  - result.txt의 학생들의 영어성적을 막대그래프로 그려보시오.

Matplotlib

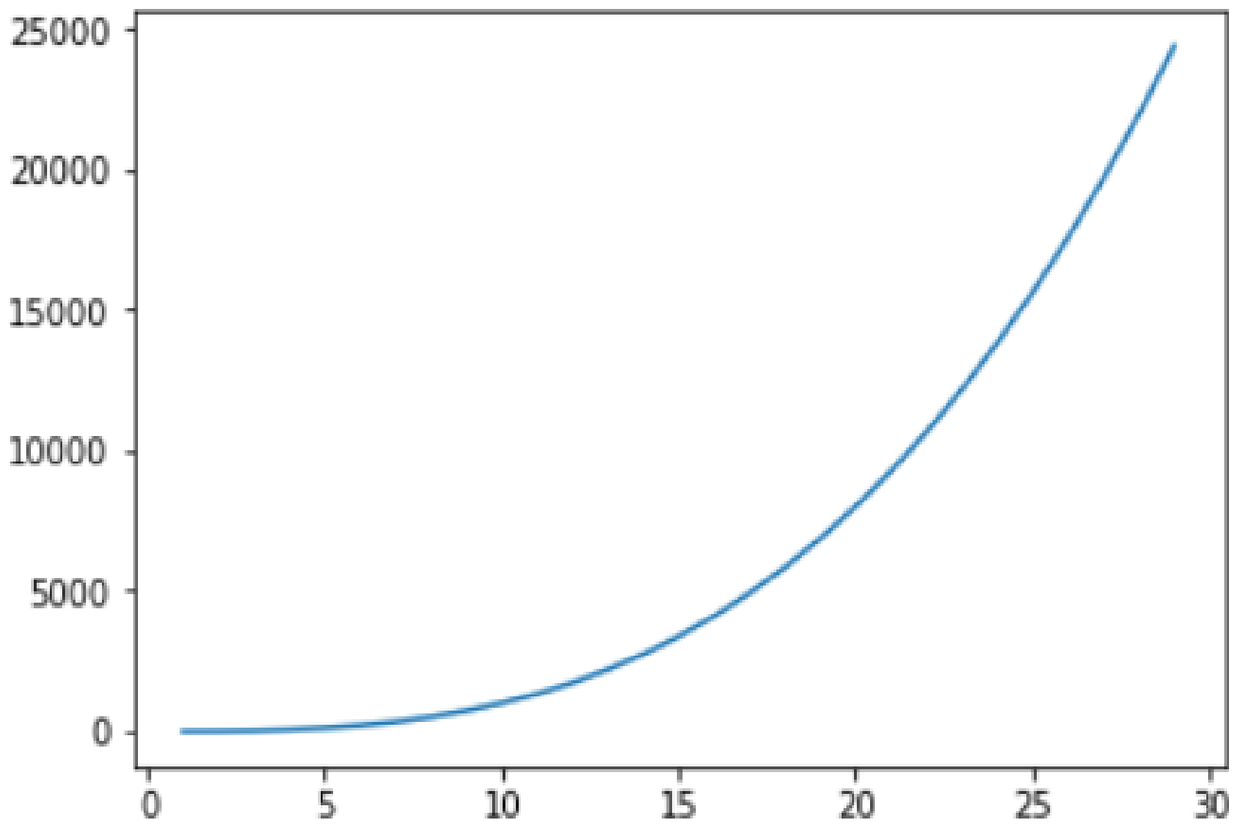
- **line plot**
  - 선형그래프
  - plt.plot()로 작성한 후 plt.show()로 jupyter 노트북에서 확인 가능
  - 선모양, 색깔, 마커 지정 가능

```
import matplotlib as mpl
import matplotlib.pylab as plt
```

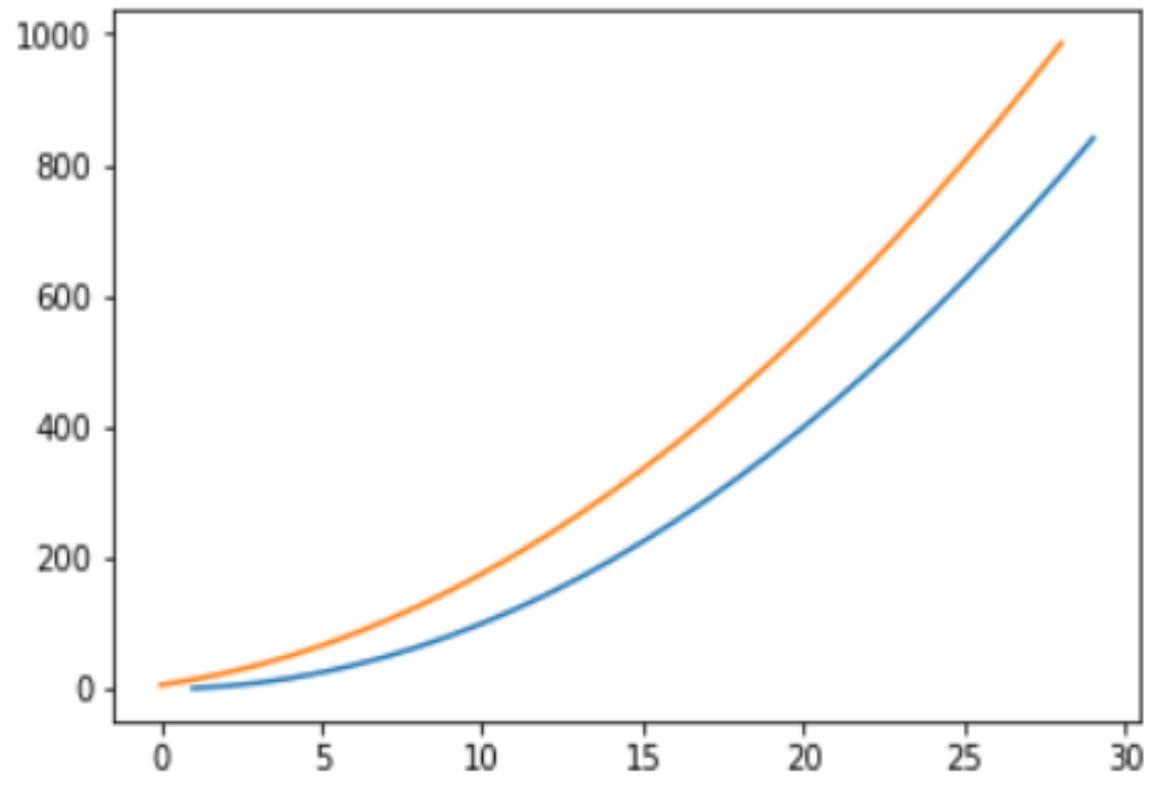
```
plt.plot([1, 9, 25, 49, 81, 121, 13*13, 15*15])
plt.show()
```



```
x = np.arange(1,30)
y = x**3
plt.plot(x,y)
plt.show()
```




```
x = np.arange(1,30)
y = x**2
y2 = x**2+5*x
plt.plot(x, y, y2)
plt.show()
```



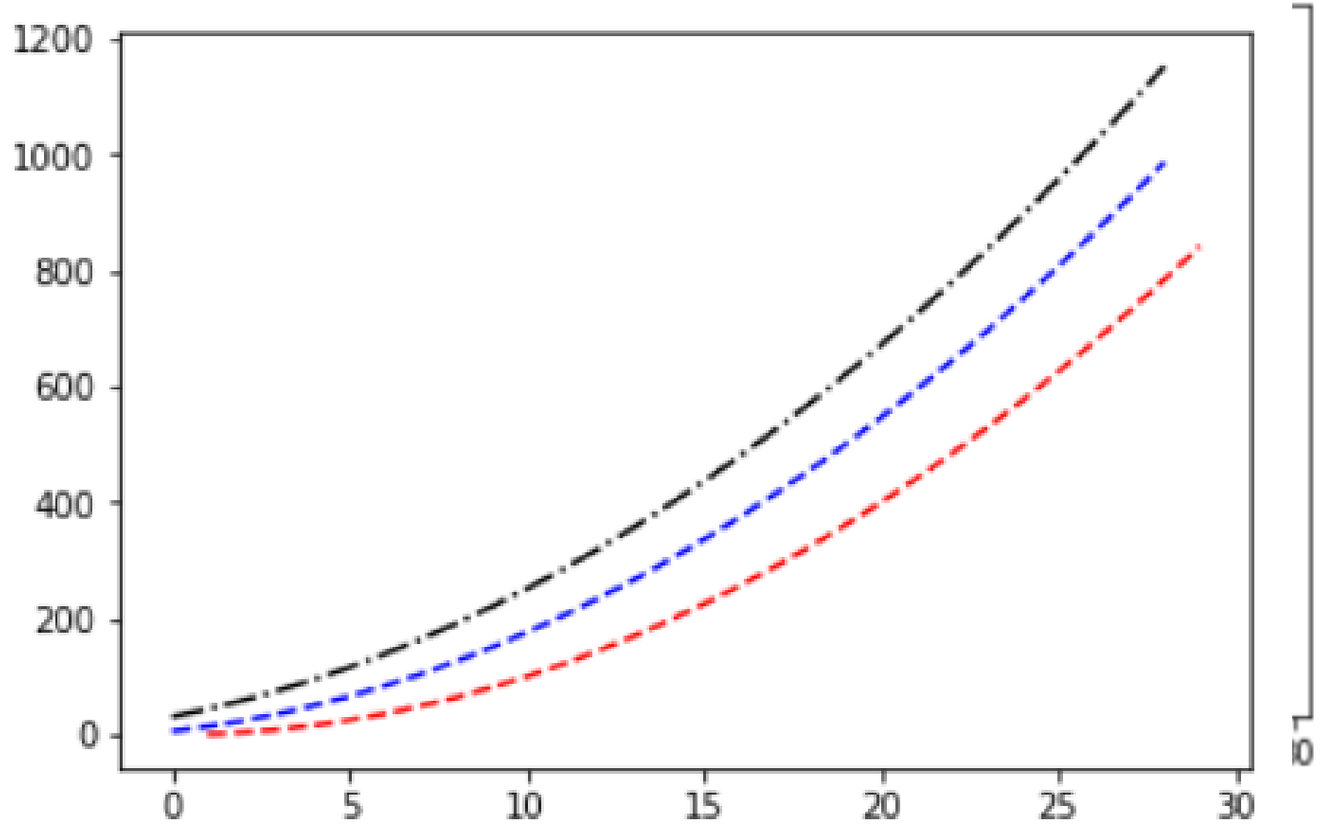
Color

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

LineStyle

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

```
x = np.arange(1,30)
y = x**2
y2 = x**2+5*x
y3 = x**2+10*x+20
plt.plot(x, y,'r—', y2,'b--', y3, 'k-.')
plt.show()
```



Matplotlib

- **line plot**
  - 선형그래프
  - plt.show()로
    jupyter 노트북에서 확인 가능
  - 선모양, 색깔, 마커 지정 가능

- **다음을 그려보시오.**
- **style을 취향대로 변경해보시오.**

(1)

```
x = np.arange(1,10)
y1 = np.random.random(9)
y2 = np.random.random(9)
plt.plot(x, y1, y2)
plt.show()
```

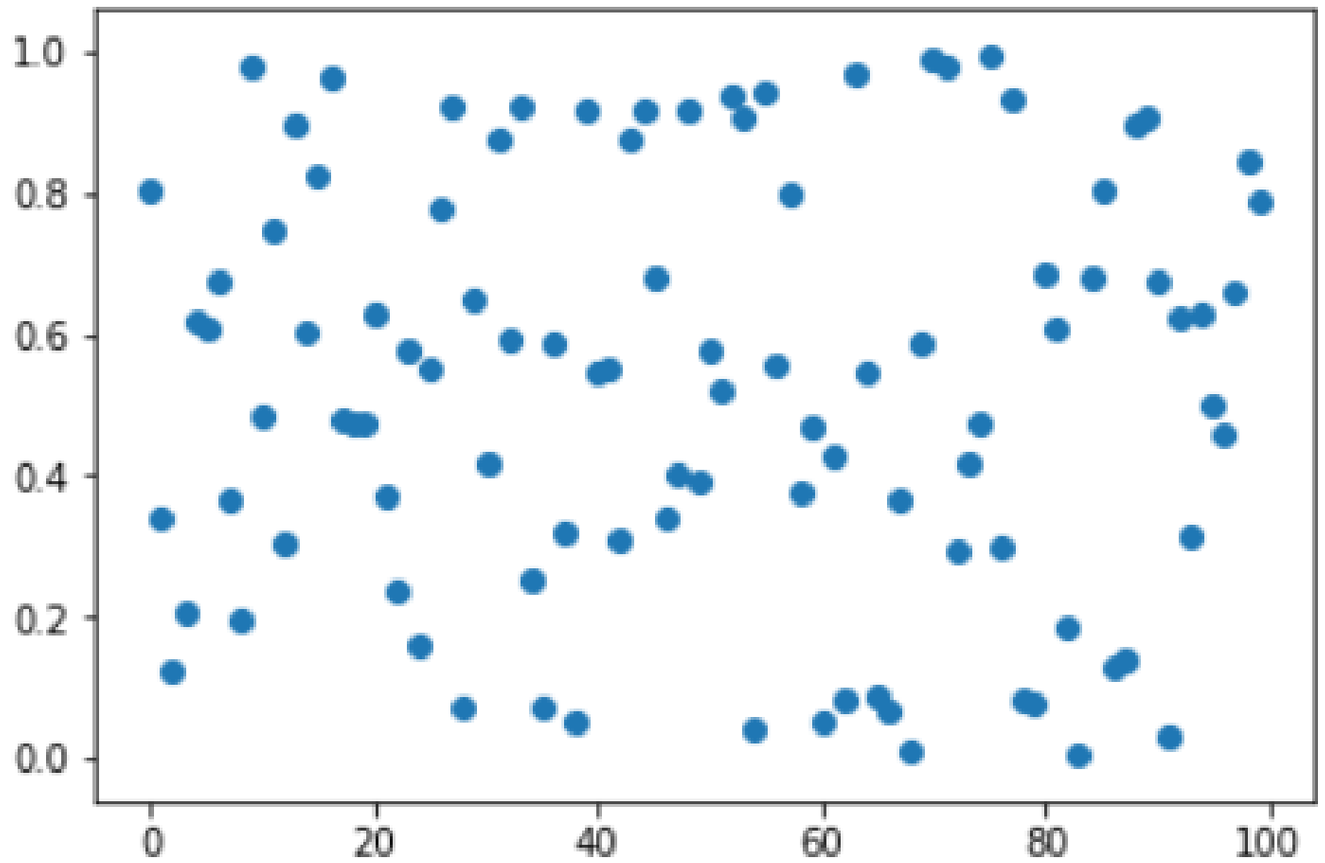
(2)

```
x = np.linspace(-6,6)
y1, y2 = np.sin(x), np.cos(x)
plt.plot(x, y1,'r--')
plt.plot(x, y2,'b--')
plt.show()
```

Matplotlib

- **scatter plot**
  - 분산그래프
  - plt.scatter()로 작성한 후 plt.show()로 jupyter 노트북에서 확인 가능

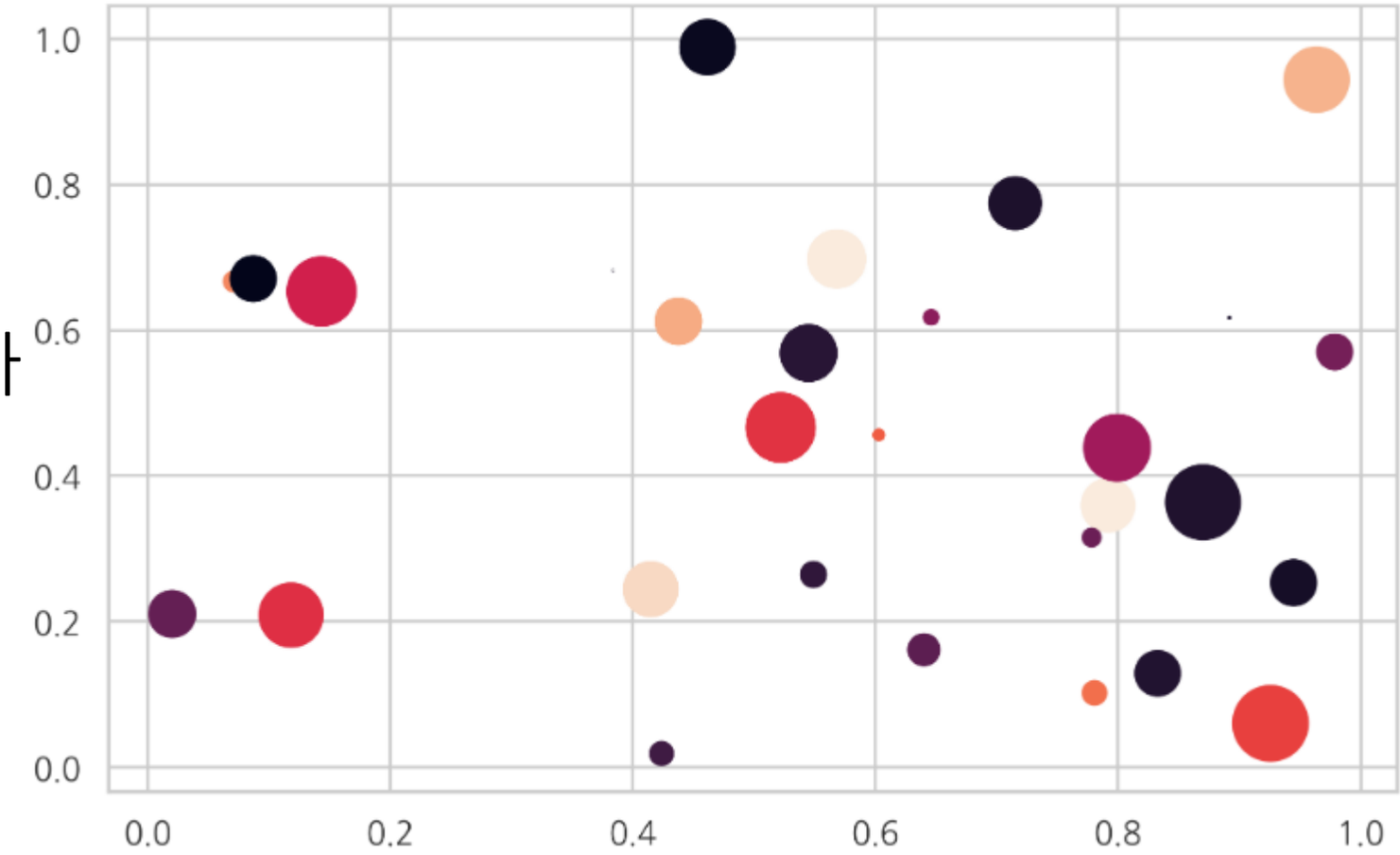
```
x = np.arange(0,100)
y = np.random.random(100)
plt.scatter(x, y)
plt.show()
```



Matplotlib

- **scatter plot**
  - 분산그래프
  - plt.scatter()로 작성한 후 plt.show()로 jupyter 노트북에서 확인 가

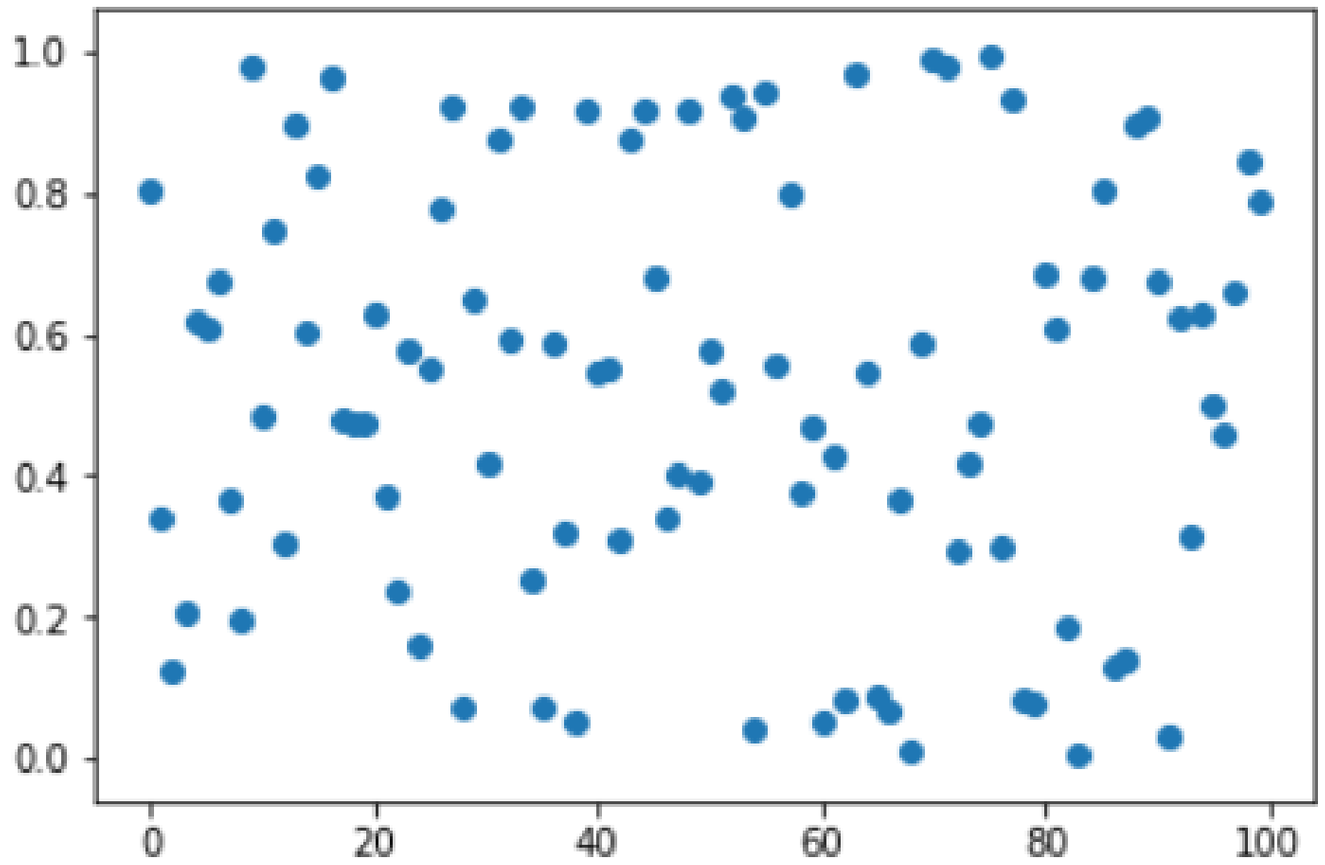
```
x = np.arange(0,100)
y = np.random.random(100)
plt.scatter(x, y)
plt.show()
```



Matplotlib

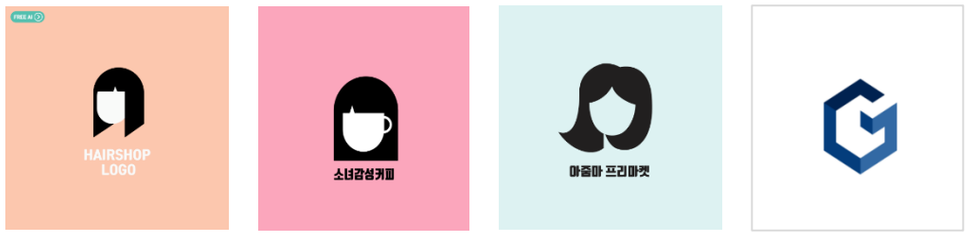
- **scatter plot**
  - 분산그래프
  - plt.scatter()로 작성한 후 plt.show()로 jupyter 노트북에서 확인 가능
  - 좌측과 같이 입력해보고 결과를 확인하시오

```
x = np.arange(0,100)
y = np.random.random(100)
plt.scatter(x, y)
plt.show()
```



Matplotlib

- **image**
  - 2D array로 표현되는 이미지
  - plt.imread()로 이미지를 로드하고 ndarray로 저장
  - plt.imshow()로 내용 확인



- 이미지를 로드하여
  데이타타입, 쉐이프, 차수를 확인하고
  화면에 보이시오.

Matplotlib

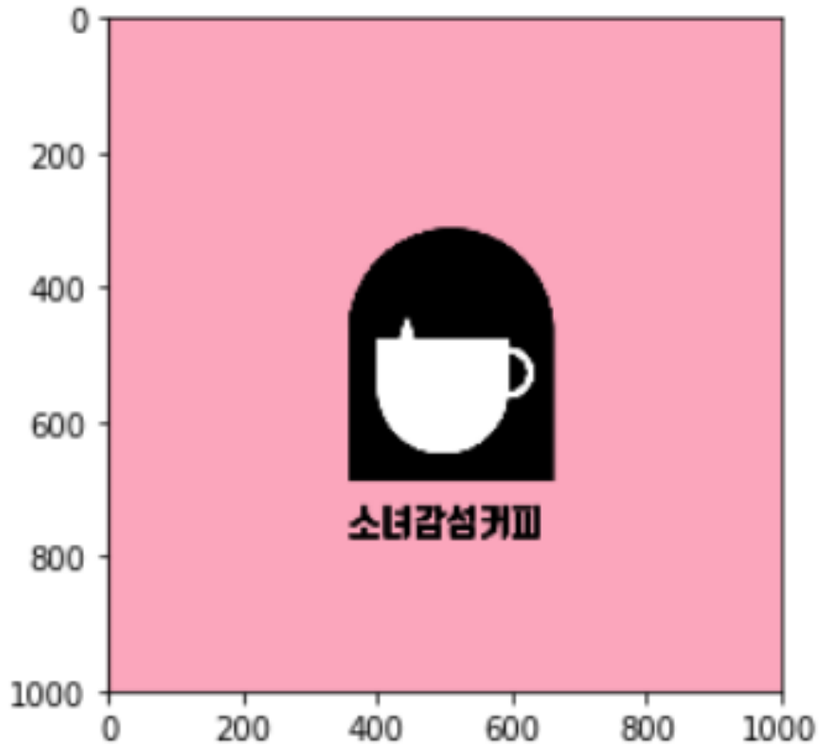
- **image**
  - 2D array로 표현되는 이미지
  - plt.imread()로 이미지를 로드하고 ndarray로 저장
  - plt.imshow()로 내용 확인

```
im1 = plt.imread("pic1.png")
print (type(im1))
print (im1.ndim)
print (im1.shape)
```

```
<class 'numpy.ndarray'>
3
(1000, 1000, 4)
```

```
plt.imshow(im1)
```

```
<matplotlib.image.AxesImage at 0x13094fa4b70>
```



Matplotlib

- **image**
  - 2D array로 표현되는 이미지
  - plt.imread()로 이미지를 로드하고 ndarray로 저장
  - plt.imshow()로 내용 확인

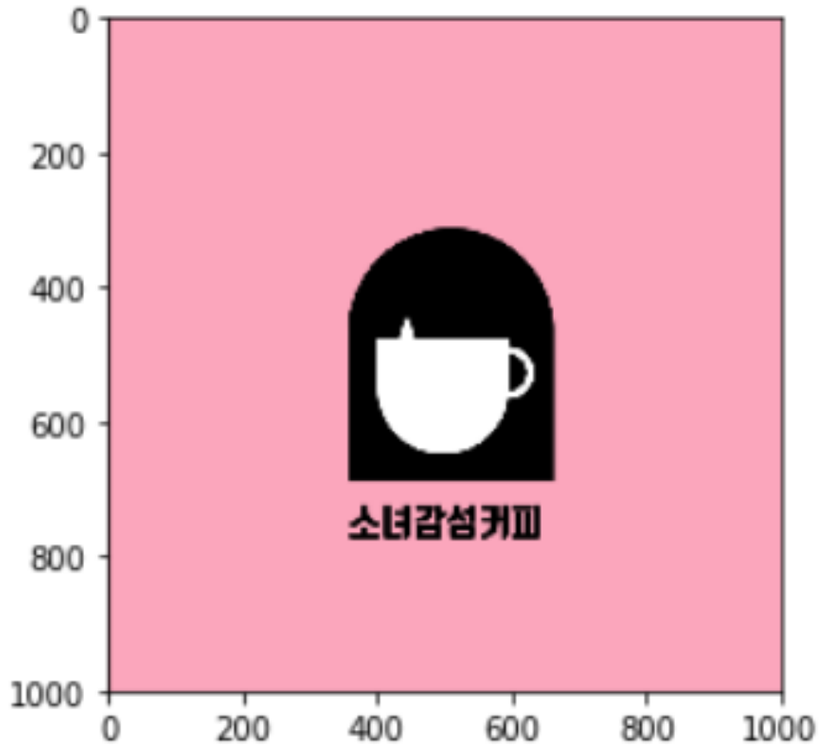
  - 주어진 이미지를 로드하여 데이타타입, 쉐이프, 차수를 확인하고 화면에 보이시오.

```
im1 = plt.imread("pic1.png")
print (type(im1))
print (im1.ndim)
print (im1.shape)
```

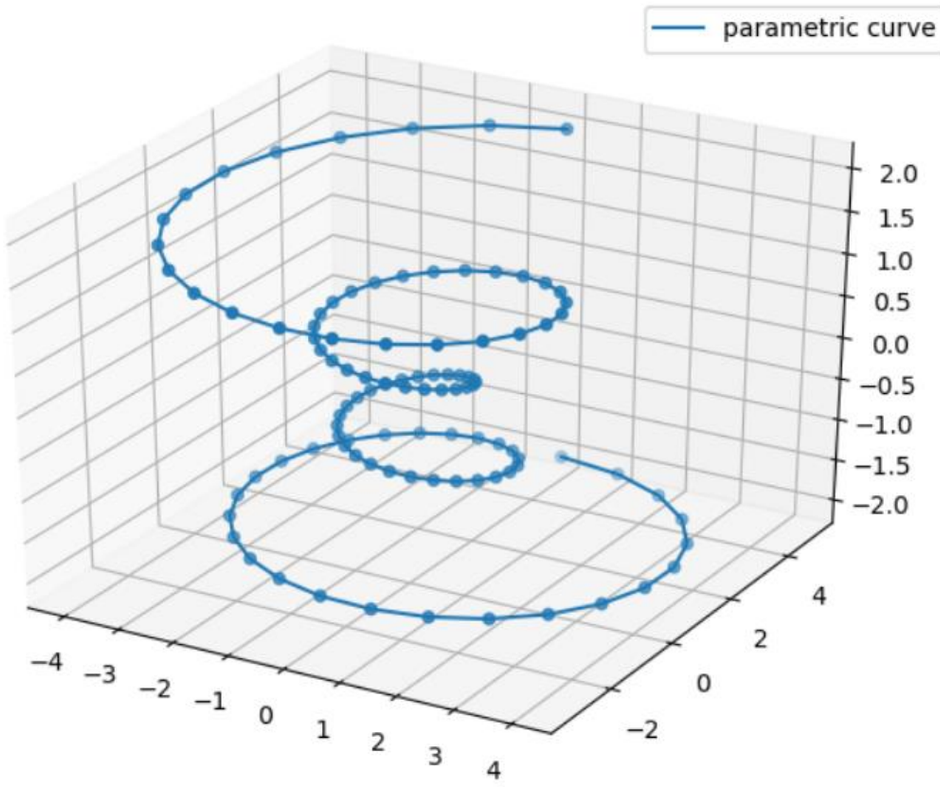
```
<class 'numpy.ndarray'>
3
(1000, 1000, 4)
```

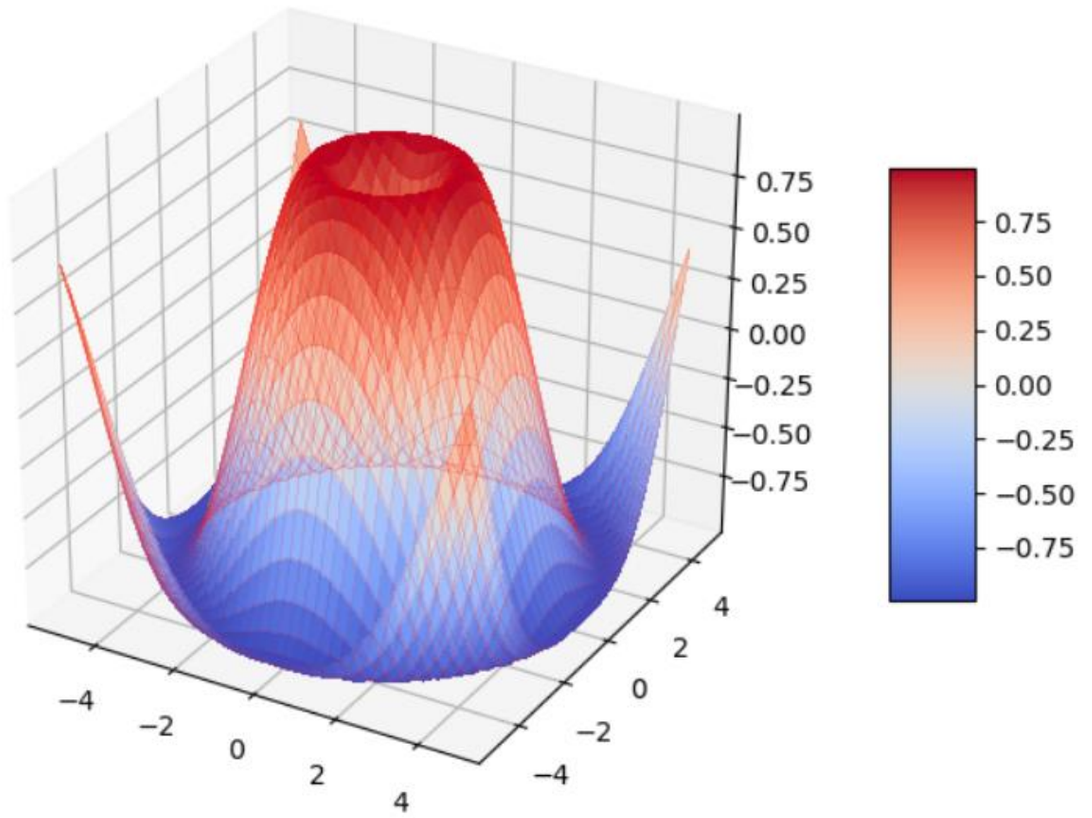
```
plt.imshow(im1)
```

```
<matplotlib.image.AxesImage at 0x13094fa4b70>
```



3D Array Data





Summary

강사소개

- **조호성 (ustog@hanyang.ac.kr)**
  - 한양대학교 SW융합원 SW교육전담교수
  - 한양대학교 전자컴퓨터통신 박사
    - 알고리즘 (정보보호, 문자열매칭, 생물정보학)