

Lecture 3

가상환경 조성 및 Numpy, Matplotlib

Jung-Il Choi

School of Mathematics and Computing (Computational Science and Engineering)



YONSEI UNIVERSITY

Contents

- 가상 환경 조성(Virtual environments)

- Python은 다양한 패키지와 모듈을 사용하여 프로그래밍을 하는 Interpreter 언어
- 패키지와 모듈의 셋업과 사용법을 익히는 것이 중요!

- 패키지와 모듈

- Numpy
- Matplotlib

1. 가상환경 조성

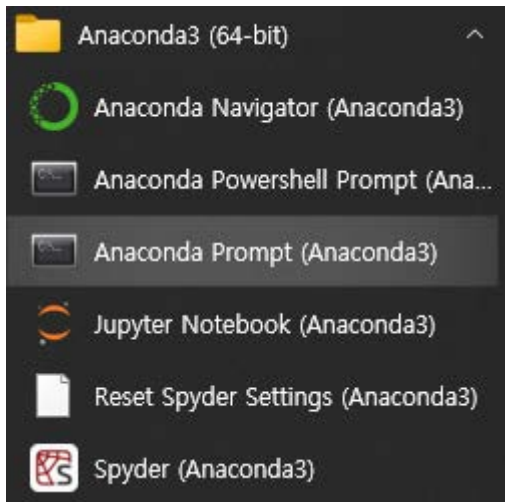
- 가상 환경 조성의 필요성

- Python은 다양한 모듈과 패키지를 활용하여 프로그래밍을 하는 인터프리터 언어이다.
- 개별 프로그래밍에 필요한 모듈과 패키지는 다른 영역에서 개발되고 있으며, 상황에 따라 원하는 모듈과 패키지에 따라서 서로 Version이 안 맞아 충돌하는 경우가 생긴다.
- 특정 프로젝트를 수행함에 있어서 개별 환경을 조성하여, 패키지를 관리하여야 한다.
- 본 수업에서는 Anaconda3 파이썬 배포판을 위주로 설명한다.

1. 가상환경 조성

- 가상 환경 조성

- Anaconda Prompt 실행 (> conda search python을 입력하면 아나콘다에서 지원하는 Python 버전을 확인할 수 있다.)



```
Anaconda Prompt (Anaconda3)

# Name      Version      Build      Channel
python      2.7.13       h1b6d89f_16 pkgs/main
python      2.7.13       h9912b81_15 pkgs/main
python      2.7.13       hb034564_12 pkgs/main
python      2.7.14       h2765ee6_18 pkgs/main
python      2.7.14       h3e68818_15 pkgs/main
python      2.7.14       h4084c39_22 pkgs/main
python      2.7.14       h4a10d90_30 pkgs/main
python      2.7.14       h4a10d90_31 pkgs/main
python      2.7.14       h59f5a59_20 pkgs/main
python      2.7.14       h819644d_16 pkgs/main
python      2.7.14       h8c3f1cb_23 pkgs/main
python      2.7.15       h2880e7c_2  pkgs/main
python      2.7.15       h2880e7c_3  pkgs/main
python      2.7.15       h2880e7c_4  pkgs/main
python      2.7.15       hcb6e200_15 pkgs/main
python      2.7.15       hcb6e200_5  pkgs/main
python      2.7.15       hcb6e200_7  pkgs/main
python      2.7.15       he216670_0  pkgs/main
python      2.7.16       hcb6e200_0  pkgs/main
python      2.7.17       h930f6bb_0  pkgs/main
python      2.7.18       hcb6e200_0  pkgs/main
python      2.7.18       hfb89ab9_0  pkgs/main
python      3.5.4       h1357f44_23 pkgs/main
python      3.5.4       hc495aa9_21 pkgs/main
python      3.5.4       hd3c4935_11 pkgs/main
python      3.5.4       hdec4e59_20 pkgs/main
python      3.5.4       hedc2606_15 pkgs/main
python      3.5.5       h0c2934d_0  pkgs/main
python      3.5.5       h0c2934d_1  pkgs/main
```

1. 가상환경 조성

- 가상 환경 조성

- “>conda create -name (or -n) 가상환경이름 python = 버전” 을 통해 가상 환경을 만들 수 있다.

→ Ex) conda create -n python_class python=3.9

✓ Anaconda3를 이용하지 않고 직접 필요한 Python 버전을 깔아 환경을 조성할 수 있다.

- (python version 3 >) “<python -m venv 가상환경이름”

```
Anaconda Prompt (Anaconda3) - conda create -n python_class python=3.9

The following packages will be downloaded:

package                                build                                size
-----                                -
ca-certificates-2023.01.10             haa95532_0                          121 KB
certifi-2022.12.7                      py39haa95532_0                      149 KB
openssl-1.1.1t                         h2bbff1b_0                          5.5 MB
pip-23.0.1                             py39haa95532_0                      2.7 MB
python-3.9.16                          h6244533_2                          19.5 MB
setuptools-65.6.3                     py39haa95532_0                      1.1 MB
sqlite-3.40.1                          h2bbff1b_0                          889 KB
tzdata-2022g                           h04d1e81_0                          114 KB
wheel-0.38.4                           py39haa95532_0                      83 KB
-----                                -
Total:                                30.1 MB

The following NEW packages will be INSTALLED:

ca-certificates pkgs/main/win-64::ca-certificates-2023.01.10-haa95532_0 None
certifi         pkgs/main/win-64::certifi-2022.12.7-py39haa95532_0 None
openssl         pkgs/main/win-64::openssl-1.1.1t-h2bbff1b_0 None
pip             pkgs/main/win-64::pip-23.0.1-py39haa95532_0 None
python          pkgs/main/win-64::python-3.9.16-h6244533_2 None
setuptools      pkgs/main/win-64::setuptools-65.6.3-py39haa95532_0 None
sqlite         pkgs/main/win-64::sqlite-3.40.1-h2bbff1b_0 None
tzdata         pkgs/main/noarch::tzdata-2022g-h04d1e81_0 None
vc             pkgs/main/win-64::vc-14.2-h21ff451_1 None
vs2015_runtime pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2 None
wheel          pkgs/main/win-64::wheel-0.38.4-py39haa95532_0 None
wincertstore   pkgs/main/win-64::wincertstore-0.2-py39haa95532_2 None

Proceed ([y]/n)? y_
```

1. 가상환경 조성

- 가상 환경 조성

- “> conda env list” 를 통해 현재 만들어져 있는 가상 환경의 목록을 볼 수 있다.

```
# conda environments:
#
base                  *  C:\Anaconda3
python_class          C:\Users\...\.conda\envs\python_class
```

- 마찬가지로, “>conda env remove -name(or -n) 가상환경이름” 을 통해 만든 가상환경을 제거할 수 있다.

→ Ex) conda env remove python_class

- “>conda activate 가상환경이름”을 통해 구축한 가상 환경을 활성화 할 수 있다.

```
(base) C:\Users\...>conda activate python_class
[python_class] C:\Users\...>_
```

- “>conda deactivate”을 통해 가상 환경에서 나갈 수 있다.

```
(python_class) C:\Users\...>conda deactivate
(base) C:\Users\...>_
```

1. 가상환경 조성

- 가상 환경 조성

- 가상 환경 구축이 완료되었다면, 필요한 라이브러리 목록을 설치해야 한다. 가상 환경에서의 라이브러리 설치에 “pip” 을 이용한다.
- “pip”을 이용하기 전에 “>pip install --upgrade pip”으로 pip를 최신버전으로 유지한다.
- “>pip --version”을 통해 버전을 확인 할 수 있다.

```
(python_class) C:\Users\₩₩₩>pip --version
pip 23.0.1 from C:\Users\₩₩₩\AppData\Local\Programs\Python\Python39\python.exe\lib\site-packages\pip (python 3.9)
```

- “>pip install 라이브러리 이름”을 통해 필요한 라이브러리를 가상 환경 내에서 설치할 수 있다.
→ Ex) pip install numpy scipy matplotlib

```
(python_class) C:\Users\₩₩₩>pip install numpy scipy matplotlib
Collecting numpy
  Downloading numpy-1.24.2-cp39-cp39-win_amd64.whl (14.9 MB)
    ----- 14.9/14.9 MB 9.4 MB/s eta 0:00:00
Collecting scipy
  Downloading scipy-1.10.1-cp39-cp39-win_amd64.whl (42.5 MB)
    ----- 42.5/42.5 MB 8.8 MB/s eta 0:00:00
Collecting matplotlib
  Downloading matplotlib-3.7.1-cp39-cp39-win_amd64.whl (7.6 MB)
    ----- 7.6/7.6 MB 9.6 MB/s eta 0:00:00
Collecting fonttools>=4.22.0
```

1. 가상환경 조성

- 가상 환경 조성

- “>conda list” 또는 “>pip list”를 통해 가상 환경 내에 설치한 라이브러리 목록을 확인할 수 있다.
- “>conda remove 라이브러리 이름”을 통해 설치한 라이브러리를 삭제할 수 있다.
- 원하는 라이브러리를 모두 설치하면 가상 환경 조성이 완료 된다.

```
(python_class) C:\Users\W...>conda list
# packages in environment at C:\Users\W...\.conda\envs\python_class:
#
# Name                        Version      Build                Channel
ca-certificates              2023.01.10   haa95532_0
certifi                      2022.12.7    py39haa95532_0
contourpy                    1.0.7        pypi_0              pypi
cyclor                       0.11.0       pypi_0              pypi
fonttools                    4.39.0       pypi_0              pypi
importlib-resources          5.12.0       pypi_0              pypi
kiwisolver                   1.4.4        pypi_0              pypi
matplotlib                   3.7.1        pypi_0              pypi
numpy                        1.24.2       pypi_0              pypi
openssl                      1.1.1t       h2b6ff1b_0
packaging                    23.0         pypi_0              pypi
pillow                       9.4.0        pypi_0              pypi
pip                          23.0.1       py39haa95532_0
pyparsing                    3.0.9        pypi_0              pypi
python                       3.9.16       h6244533_2
python-dateutil              2.8.2        pypi_0              pypi
scipy                        1.10.1       pypi_0              pypi
setuptools                   65.6.3       py39haa95532_0
six                          1.16.0       pypi_0              pypi
sqlite                       3.40.1       h2b6ff1b_0
tzdata                       2022g        h04d1e81_0
vc                           14.2         h21ff451_1
vs2015_runtime               14.27.29016  h5e58377_2
wheel                        0.38.4       py39haa95532_0
wincertstore                 0.2          py39haa95532_2
zipp                         3.15.0       pypi_0              pypi
```


1. 가상환경 조성

• 가상 환경 조성

- 조성한 가상 환경을 그대로 다른 가상 환경에서 적용하고 싶다면
“>conda env export > conda_requirements.txt” 를 통해 설치한 패키지 목록을 저장할 수 있다.
- conda_requirements.txt는 가상환경을 활성화한 폴더에 만들어진다.
- “>conda env create -f conda_requirements.txt” 를 이용하면 동일한 가상 환경을 구축할 수 있다.
- “>pip freeze > requirements.txt”를 통해 pip을 통해 install한 패키지 목록만 따로 저장할 수 있다.
 - ➔ 단, python 버전은 따로 표기가 되지 않기 때문에, python 버전을 따로 명시해 줄 필요가 있다.
- “>pip install -r requirements.txt를 통해 동일한 환경을 구축 할 수 있다.

```
conda_requirements.txt
1  name: python_class
2  channels:
3      - http://conda.anaconda.org/gurobi
4      - defaults
5  dependencies:
6      - ca-certificates=2023.01.10=h1a95532_0
7      - certifi=2022.12.7=py39h1a95532_0
8      - openssl=1.1.1t=h2bbff1b_0
9      - pip=23.0.1=py39h1a95532_0
10     - python=3.9.16=h6244533_2
11     - setuptools=65.6.3=py39h1a95532_0
12     - sqlite=3.40.1=h2bbff1b_0
13     - tzdata=2022g=h04d1e81_0
14     - vc=14.2=h21ff451_1
15     - vs2015_runtime=14.27.29016=h5e58377_2
16     - wheel=0.38.4=py39h1a95532_0
17     - wincertstore=0.2=py39h1a95532_2
18     - pip:
19         - contourpy==1.0.7
20         - cycycler==0.11.0
21         - fonttools==4.39.0
22         - importlib-resources==5.12.0
23         - kiwisolver==1.4.4
24         - matplotlib==3.7.1
25         - numpy==1.24.2
26         - packaging==23.0
27         - pillow==9.4.0
28         - pyparsing==3.0.9
29         - python-dateutil==2.8.2
30         - scipy==1.10.1
31         - six==1.16.0
32         - zipp==3.15.0
33  prefix: C:\Users\...\conda\envs\python_class
```

1. 가상환경 조성

- 가상 환경 조성

- 구축한 가상 환경을 Jupyter notebook에서 실행하려면 먼저 **가상환경 내에 jupyter notebook**을 설치해야한다.

→ “>pip install jupyter notebook”

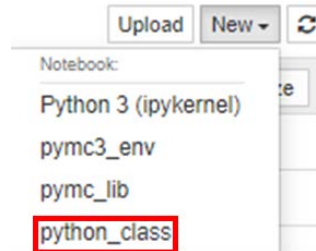
- 이후 가상 환경을 Kernel에 연결 한다.

→ “>python -m ipykernel install --user --name 가상환경이름 --display-name 커널명”

✓ Ex) >python -m ipykernel install --user --name python_class

→ 커널명은 jupyter notebook과 연동시 표기되는 이름이므로 “--display-name 커널명” 은 생략해도 무방하다.

- 이후 Jupyter notebook을 실행하고, new 탭에서 연결한 가상 환경을 선택하면 연동이 완료된다.



2. 패키지와 모듈

- 패키지와 모듈 정의

- 모듈

- 모듈은 파이썬 코드 파일. 즉, .py 확장자를 가진 파일을 말한다. 모듈은 관련 함수, 변수 및 클래스의 집합을 포함할 수 있다. 다른 파이썬 스크립트에서 코드를 재사용 할 목적으로 만든다.

- 패키지

- 패키지는 모듈의 계층적인 구성을 지원한다. 패키지는 하나 이상의 모듈을 포함할 수 있으며, 패키지에는 __init__.py 파일이 있고, 패키지는 관련된 모듈을 그룹화하여 더 큰 프로젝트를 구성하는 데 사용된다.

- 라이브러리

- 라이브러리는 여러 모듈 및 패키지의 모음이다. 많은 프로그래밍 언어에서 라이브러리는 기본 제공되며, 이는 표준 라이브러리(Standard library)라고 한다. 또한, 사용자가 작성한 코드를 라이브러리 형태로 공유할 수도 있다.

2. 패키지과 모듈

- 패키지와 모듈 불러오기

- 사용하려는 패키지 또는 모듈을 선언하는 기본적인 형태

```
from some_library import some_package, module
```

- Ex)

```
from math import atan, pi, sin, cos, log
```

- 라이브러리에 있는 모든 패키지/모듈 선언하기

```
from some_library import *
```

- 이렇게 라이브러리에서 직접 패키지와 모듈을 선언하면 편리하지만 추후 코드가 길어지고 선언하는 라이브러리가 증가하면 변수/함수명 선언에 충돌이 발생할 수 있다. (Name conflicts)
- Prefix를 써서 불러오는 형태가 선호된다.

```
✓ import numpy as np  
  from scipy.linalg import inv  
  import matplotlib.pyplot as plt
```

```
A=np.array([[1,2,3],[4,5,6]])  
print(A)  
✓ 0.0s  
[[1 2 3]  
 [4 5 6]]
```

2. 패키지와 모듈

- **Numpy**

- 기본적인 과학계산 패키지
- 배열(Array) 자료형을 이용한 수치계산 최적화
- 각종 선형대수, 푸리에 변환, 난수 관련 모듈
- 다른 과학계산 패키지와의 연동이 편리 (ex) Scipy)
- 오픈 소스(무료)

2. 패키지와 모듈

- **Numpy**

- 필요한 모듈/함수가 있으면, User Guide를 적극 참조
- 배열 생성하기

→ **numpy.arange** <https://numpy.org/doc/stable/reference/generated/numpy.arange.html>

```
numpy.arange([start, ]stop, [step, ]dtype=None, *, Like=None)
```

“[start,] : default 값 존재 (0), [step,] : default 값 존재 (1), [start, stop) half-open 구간에서 일정한 간격(step)의 배열 생성”

```
np.arange(0, 1, 0.2)
✓ 0.0s
array([0. , 0.2, 0.4, 0.6, 0.8])
```

→ **numpy.linspace** <https://numpy.org/doc/stable/reference/generated/numpy.linspace.html>

```
numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)
```

“num : default 값 존재 (50), [start, stop] closed 구간에서 일정한 간격의 (num)개의 배열 생성”

```
np.linspace(0, 2*np.pi, 4)
✓ 0.0s
array([0. , 2.0943951 , 4.1887902 , 6.28318531])
```

2. 패키지와 모듈

- Numpy

- 배열 생성하기

- N-dimensional 배열(행렬)도 생성 가능
- 필요한 Attribute (ex) .shape) 가 있으면 User Guide 적극 참조
(<https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html>)

모든 원소가 0인 $n \times m$ 행렬 생성

```
A = np.zeros((2,3))
print(A.shape)
print(A)

✓ 0.0s

(2, 3)
[[0. 0. 0.]
 [0. 0. 0.]]
```

$n \times n$ 대각행렬 생성

```
A = np.diag((1,2,3,4))
print(A.shape)
print(A)

✓ 0.0s

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

모든 원소가 1인 $n \times m$ 행렬 생성

```
A = np.ones((3,2))
print(A.shape)
print(A)

✓ 0.0s

(3, 2)
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

단위 $n \times n$ 행렬 생성

```
A=np.eye(3)
print(A)

✓ 0.0s

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

```
a=np.arange(10)
print(a)
a=a.reshape((2,5)) # 1X10 array to 2X5 matrix
print(a)
print(a.ndim)      # 2 dimension
print(a.size)      # 10 elements
a=a.T              # transpose
print(a)
a.dtype            # data type

✓ 0.0s
```



```
[0 1 2 3 4 5 6 7 8 9]
[[0 1 2 3 4]
 [5 6 7 8 9]]
2
10
[[0 5]
 [1 6]
 [2 7]
 [3 8]
 [4 9]]

dtype('int32')
```

2. 패키지와 모듈

- Numpy

- 난수 생성하기

→ <https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html>

```
np.random.random((2,3))  
✓ 0.0s  
array([[0.86925323, 0.43688411, 0.82949868],  
       [0.59514665, 0.22652891, 0.75682662]])
```

[0, 1] 사이의 난수를 2X3 행렬로 생성

- I/O

```
a=np.random.normal(size=(3,3))  
  
np.savetxt("a_out.txt", a)  
# save to file  
b = np.loadtxt("a_out.txt")  
# read from file  
print(a)  
print(b)  
✓ 0.0s  
[[-1.86107637  0.27582315 -0.04186183]  
 [ 2.00079254 -0.78191699  1.48089965]  
 [ 0.75850862  1.80576736 -0.06448941]]  
[[-1.86107637  0.27582315 -0.04186183]  
 [ 2.00079254 -0.78191699  1.48089965]  
 [ 0.75850862  1.80576736 -0.06448941]]
```

```
np.random.normal(loc=1.0, scale=2.0, size=(2,2))  
✓ 0.0s  
array([[ 0.37641539,  2.39084927],  
       [-0.74078059, -1.65686366]])
```

Mean이 1이고 Std가 2인 정규분포에서 난수를 2X2 행렬로 생성
(default는 표준 정규분포)

```
≡ a_out.txt  
≡ a_out.txt  
1 | -1.861076368342400089e+00  2.758231518970046814e-01 -4.186182921091661374e-02  
2 |  2.000792543451720640e+00 -7.819169889001940099e-01  1.480899646593767160e+00  
3 |  0.7585086188321534806e-01  1.805767361135816262e+00 -6.448941202344560253e-02
```


2. 패키지와 모듈

- **Numpy**

- Arithmetic operators : elementwise application

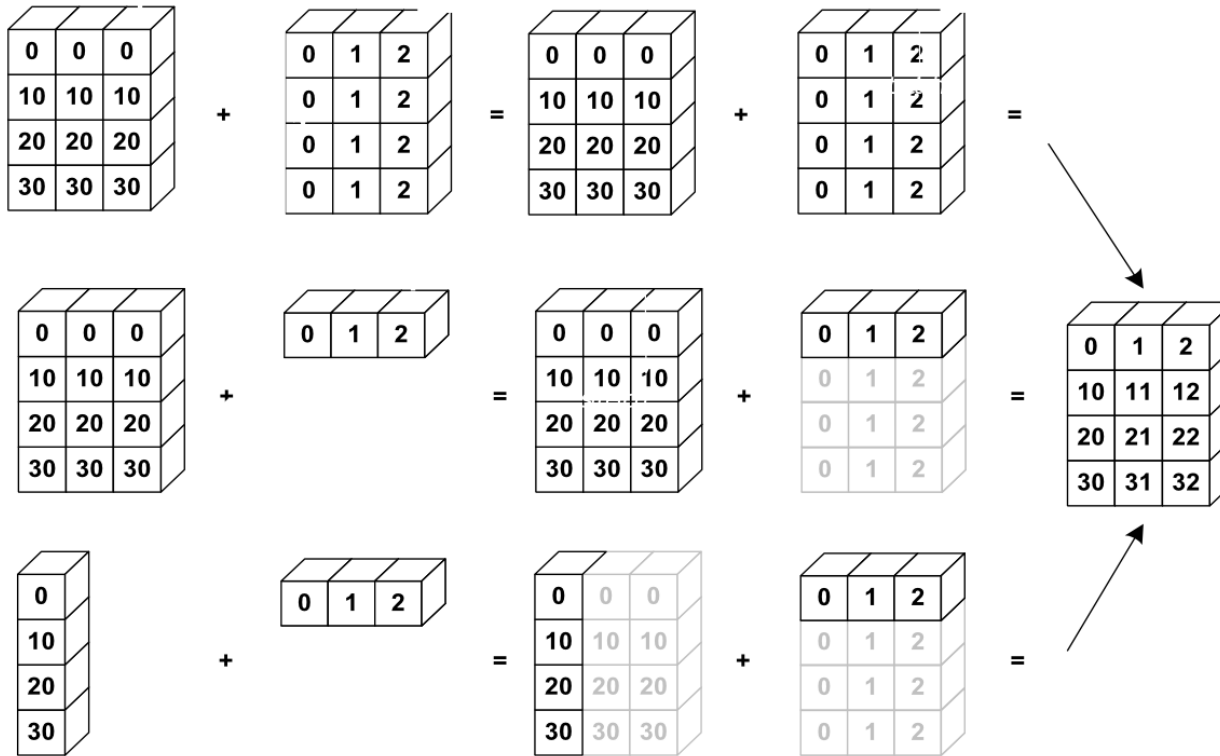
```
a = np.arange(4)
print(a)                # array([0, 1, 2, 3])
b = np.array([2, 3, 2, 4])
print(a * b)            # array([ 0, 3, 4, 12])
print(b - a)            # array([2, 2, 0, 1])
c = [2, 3, 4, 5]
print(a * c)            # array([ 0, 3, 8, 15])
```

- ➔ 2개의 배열을 계산할 때, Numpy는 shape을 비교하여 계산의 성립 여부를 먼저 조사한다.
- ➔ 대수 계산이 성립하려면,
 - ✓ 배열의 Shape이 같을 때,
 - ✓ 배열의 Shape이 다르지만, 둘 중 하나가 1차원 배열이고 사이즈가 다른 하나의 행/열 중 하나와 같을 때
 - ✓ 둘 다 1차원 배열이고, 행/열 Shape이 다를 때 (즉, $1 \times n$ 과 $m \times 1$)
 - ✓ N-dimensional 배열과 1개 상수

2. 패키지와 모듈

- Numpy

→ 배열에 상수에 대한 Arithmetic operator를 적용하면 모든 element에 모두 적용된다.



```
A = np.ones((3,3))
print (3 * A - 1)
```

✓ 0.0s

```
[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]
```

Numpy 배열 계산 성립 경우와 결과

2. 패키지와 모듈

- Numpy

- Vector operators

- Inner product (`np.inner(u, v)`)

- Outer product (`np.outer(u, v)`)

- Matrix multiplication (`np.dot(A, B)`)

```
u = np.array([1.2, 2.5, -3.1])
v = np.array([-1.4, 1.3, 5.1])
print(np.inner(u,v))
print(np.outer(u,v))

✓ 0.0s

-14.239999999999998
[[ -1.68  1.56  6.12]
 [ -3.5   3.25 12.75]
 [ 4.34 -4.03 -15.81]]
```

```
A = np.ones((3, 2))
B = np.ones((2, 3))
print(A)
print(B)

✓ 0.0s

[[1. 1.]
 [1. 1.]
 [1. 1.]]
[[1. 1. 1.]
 [1. 1. 1.]]
```

```
np.dot(A, B)

✓ 0.0s

array([[2., 2., 2.],
       [2., 2., 2.],
       [2., 2., 2.]])
```

```
np.dot(B, A)

✓ 0.0s

array([[3., 3.],
       [3., 3.]])
```

```
np.dot(B.T, A.T)

✓ 0.0s

array([[2., 2., 2.],
       [2., 2., 2.],
       [2., 2., 2.]])
```

행렬 연산 법칙에 위배 되면 오류가 발생

```
np.dot(A, B.T)

⊗ 0.0s

-----
ValueError                                Traceback (most recent call last)
Cell In[46], line 1
----> 1 np.dot(A, B.T)

File <__array_function__ internals>:200, in dot(*args, **kwargs)
ValueError: shapes (3,2) and (3,2) not aligned: 2 (dim 1) != 3 (dim 0)
```

2. 패키지와 모듈

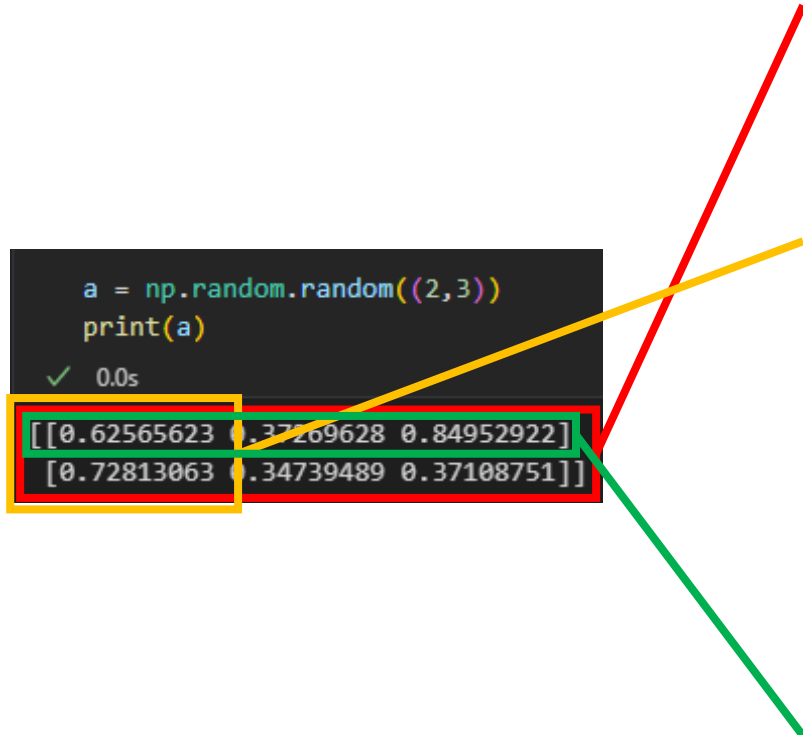
- Numpy

- Operation along axes

```
a = np.random.random((2,3))
print(a)
```

✓ 0.0s

```
[[0.62565623 0.37269628 0.84952922]
 [0.72813063 0.34739489 0.37108751]]
```



```
a.sum()
```

✓ 0.0s

3.2944947506479982

```
a.min()
```

✓ 0.0s

0.3473948894744653

```
a.sum(axis=0) # column sum
```

✓ 0.0s

array([1.35378685, 0.72009117, 1.22061672])

```
a.max(axis=0)
```

✓ 0.0s

array([0.72813063, 0.37269628, 0.84952922])

```
a.cumsum()
```

✓ 0.0s

array([0.62565623, 0.99835251, 1.84788173, 2.57601235, 2.92340724, 3.29449475])

```
a.cumsum(axis=1) # cumulative row sum
```

✓ 0.0s

array([[0.62565623, 0.99835251, 1.84788173],
 [0.72813063, 1.07552552, 1.44661302]])

Column → row

2. 패키지와 모듈

- Numpy

- Slicing

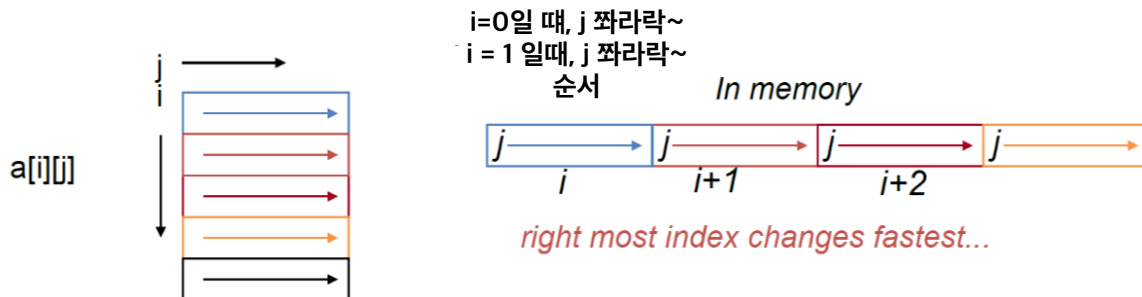
- Python의 indexing은 0부터 시작한다
- A[row, column]
- start:stop은 Half-open 구간 [start:stop)을 카운트 한다.

```
a = np.random.random((4,5))
a[2, :]          # third row, all columns
a[1:3]           # 2nd, 3rd row, all columns
a[:, 2:4]        # all rows, columns 3 and 4
```

```
[[0.58289928 0.07051389 0.19787544 0.30504009 0.54614797]
 [0.20414595 0.78362425 0.95001806 0.66601481 0.99929904]
 [0.33826978 0.56581613 0.79373313 0.70502512 0.6454779 ]
 [0.09374427 0.29420426 0.84452776 0.60183001 0.04206279]]
```

- Iterating order

- Python에서는 C와 동일하게 I, J, K 순서로 메모리에 접근한다



```
b = 0
for i in range(256):
    for j in range(256):
        for k in range(256):
            b += a[i, j, k]
```

✓ 4.3s

```
b = 0
for k in range(256):
    for j in range(256):
        for i in range(256):
            b += a[i, j, k]
```

✓ 5.9s

2. 패키지와 모듈

- **Numpy**

- 그 밖의 유용한 함수 및 속성(Attribute)들

- “np.where(“array | condition”) : 배열 내에서 조건에 맞는 자료의 인덱스를 반환한다.
 - ✓ Ex) np.where(array>value) : array 내부에서 value보다 값이 큰 수들의 인덱스를 저장한다.
- “np.array(List, dtype=float)” : 리스트를 배열로 전환
- “array.tolist()” : 배열을 리스트로 전환
- “np.append(array, value)” : 배열 끝에 값 추가
- “array.astype(dtype)” : 배열 원소의 데이터 타입 변환
 - ✓ (정수 = int, 실수 = float, 복소수 = complex)
- “np.column_stack((array1, array2))” : array1, array2를 Column 방향으로 합친다.
- “np. row_stack((array1, array2))” : array1, array2를 Row 방향으로 합친다.
- “np.linalg.inv(2d_square_array)” : 정사각행렬의 역행렬을 구한다.

- And so on...

2. 패키지와 모듈

- **Matplotlib**

- 다양한 자료를 시각화 할 수 있는 Matplotlib 라이브러리

```
import matplotlib.pyplot as plt
```

- rcParams를 이용한 그래프 옵션 설정
 - Figure(Canvas) 크기 설정 [가로, 세로]
 - Figure 내 글씨 크기 설정
 - Figure 내 글씨체 설정
 - LaTeX 사용 여부
 - 그래프 상자 선 두께 설정
 - 그래프 내부 선 두께 설정(추후 조정 가능)
 - 그래프 눈금 방향 설정(x, y)
 - 그래프 눈금 Major/Minor 설정 여부 및 길이/두께 설정
 - 그래프 눈금 (위/아래) 추가 설정

```
plt.rcParams['figure.figsize'] = [7,5]
plt.rcParams['font.size'] = 15
plt.rcParams['font.family'] = 'Times New Roman'
plt.rcParams['text.usetex'] = False
plt.rcParams['axes.linewidth'] = 2
plt.rcParams['lines.linewidth'] = 2
plt.rcParams['xtick.direction'] = 'in'
plt.rcParams['ytick.direction'] = 'in'
plt.rcParams['xtick.minor.visible']=True
plt.rcParams['ytick.minor.visible']=True
plt.rcParams['xtick.major.size']= 7
plt.rcParams['ytick.major.size']= 7
plt.rcParams['xtick.minor.size']= 3.5
plt.rcParams['ytick.minor.size']= 3.5
plt.rcParams['xtick.major.width']= 1.5
plt.rcParams['ytick.major.width']= 1.5
plt.rcParams['xtick.minor.width']= 1.5
plt.rcParams['ytick.minor.width']= 1.5
plt.rcParams['xtick.top'] = True
plt.rcParams['ytick.right'] = True
```

2. 패키지와 모듈

• Matplotlib

• 간단한 선 그래프 (“plt.plot()”)

→ Default는 실선

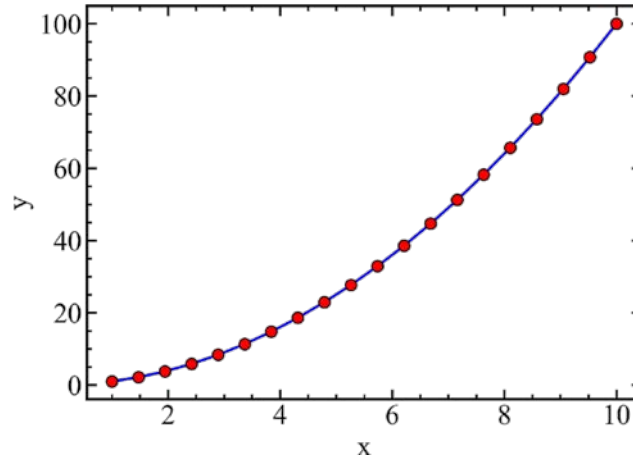
```
x = np.linspace(1, 10, 20)
y = x*x
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

→ 선의 색과 종류, 표식자 바꾸기

```
plt.plot(x,y,'k-') # 검은색 실선
plt.plot(x,y,'r:',lw=3) # 빨간색 점선
plt.plot(x,y,'b--') # 파란색 점선
plt.plot(x,y,'y-.') # 노란색 일점쇄선
plt.plot(x,y,color='blue',linestyle='--')
```

→ “mec” : 표식자 테두리 색, “mfc” : 표식자 내부 색, “ms” : 표식자 크기

```
plt.plot(x,y,'ro') # 빨간색 원
plt.plot(x,y,'bs') # 파란색 사각형
plt.plot(x,y,'y.') # 노란색 점
plt.plot(x,y,'ro-') # 빨간색 원과 실선
plt.plot(x,y,'bo-',mec='k',mfc='r',ms=8)
# 파란색 선과 빨간색 원, 검은색 테두리
```



기호	표식자	기호	표식자
^	삼각형	v	역삼각형
>	오른쪽 삼각형	<	왼쪽 삼각형
o	원	s	사각형
p	오각형	h	육각형
D	마름모	8	팔각형
+	더하기 기호	x	곱하기 기호
.	작은 점	*	별표

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html

2. 패키지와 모듈

- Matplotlib

- 간단한 선 그래프

→ 로그 스케일로 그래프를 그릴 때,

✓ 방법 1

```
plt.semilogx(x, y)    # x축 로그 스케일  
plt.semilogy(x, y)    # y축 로그 스케일  
plt.loglog(x, y)      # x, y축 로그 스케일
```

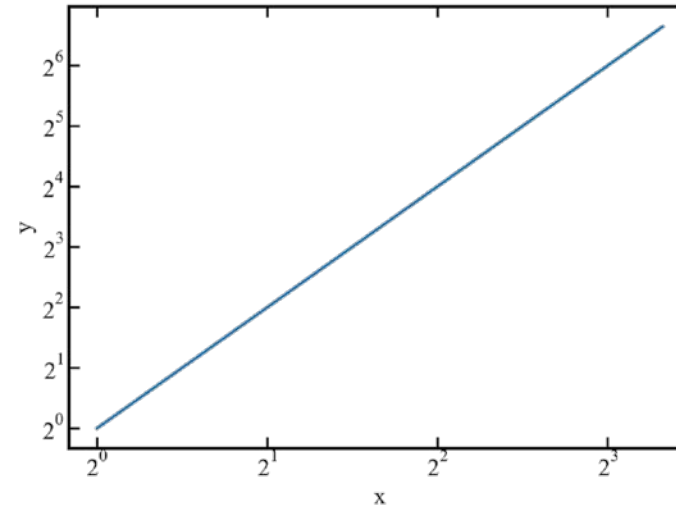
✓ 방법 2

```
plt.plot(x,y)  
plt.xscale('log')  
plt.yscale('log')
```

→

```
plt.plot(x,y)  
plt.xscale('log', base=2)  
plt.yscale('log', base=2)
```

* Data에 대한 정보를 알 수 없음
* data를 찍어주는게 좋다
* x, y 사이즈 커야함...!!

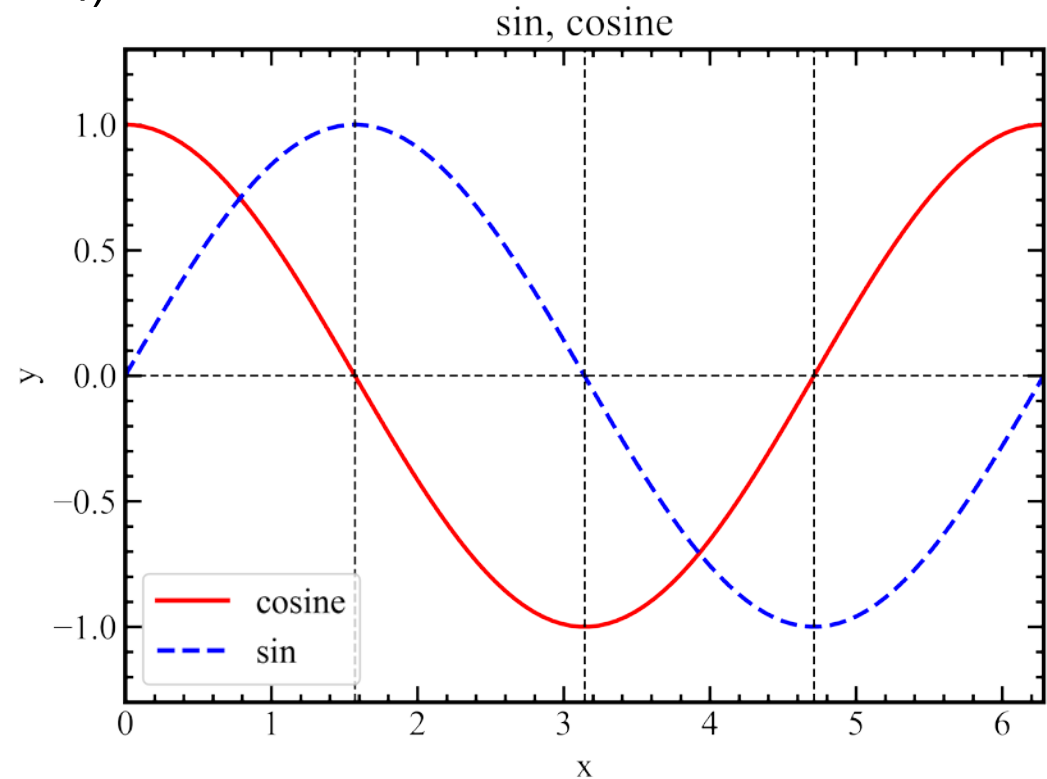


2. 패키지와 모듈

- **Matplotlib**

- 2개 이상의 그래프 동시에 그리기 (같은 Figure 안에)

```
x=np.linspace(0, 2*np.pi, 100)
y1=np.cos(x)
y2=np.sin(x)
plt.plot(x, y1, 'r-',label='cosine')
plt.plot(x, y2, 'b--',label='sin' )
plt.xlim(0, 2*np.pi)
plt.ylim(-1.3, 1.3)
plt.xlabel('x')
plt.ylabel('y')
plt.title('sin, cosine')
plt.axhline(y=0, color='k', linewidth=1, linestyle='--')
for i in range(4):
    plt.axvline(x=i*np.pi/2, color='k', linewidth=1, linestyle='--')
plt.minorticks_on()
plt.legend()
plt.savefig('example2.png', dpi=600, transparent=True)
plt.show()
```



- “plt.axhline() : 수평선 그리기”
- “plt.axvline() : 수직선 그리기”
- “plt.plot(x, y, label=”) → “plt.legend()” **범례** 추가하기 (위치, 폰트 사이즈 등 수정 가능)
- “plt.savefig(“filename”) → 그래프 저장하기

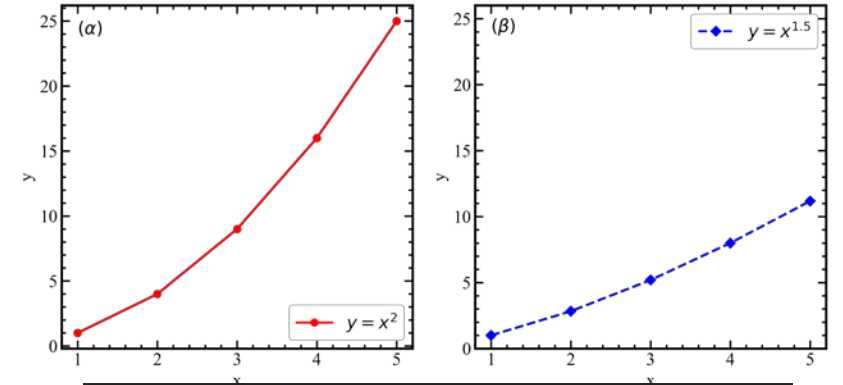
https://matplotlib.org/stable/api/as_gen/matplotlib.pyplot.legend.html

2. 패키지와 모듈

- **Matplotlib**

- 2개 이상의 그래프 동시에 그리기 (다른 여러 개의 Figure)

- ➔ 그래프 여러 개를 그리는 방법은 다양하다.
- ➔ 기본적으로 “plt.subplot()”을 이용한다.
- ➔ “plt.figure(id, figsize=(가로, 세로 (inch)))”
 - ✓ 먼저 그래프를 그릴 Canvas의 크기 지정한다
- ➔ “plt.subplot(000)”
 - ✓ 앞 00으로 총 Canvas를 나눈다.
 - ✓ 뒤 0로 그래프를 위치시킬 순서를 지정한다.
 - Ex) 121 이면 세로 1개 가로 2개에 첫번째라는 의미



```
x=np.arange(1,6)
y1=x*x
y2=x**1.5

plt.figure(1, figsize=(10, 5))
plt.subplot(121)
plt.plot(x, y1, 'ro-', label=r'$y=x^2$')
plt.xlabel('x')
plt.ylabel('y')
plt.text(1, 24, r'$(\alpha)$')
plt.legend(loc='lower right')

plt.subplot(122)
plt.plot(x, y2, 'bd--', label=r'$y=x^{1.5}$')
plt.xlabel('x')
plt.ylabel('y')
plt.ylim(0, 26)
plt.text(1, 24, r'$(\beta)$')
plt.legend(loc='upper right')
plt.tight_layout()
plt.savefig('example3.png', dpi=600, transparent=True)
plt.show()
```

2. 패키지와 모듈

- **Matplotlib**

- 2개 이상의 그래프 동시에 그리기 (다른 여러 개의 Figure)
 - ➔ 다른 방식으로 가능하다.

- ✓ “sharex, sharey”
 - 축 공유
- ✓ Axes의 속성으로 그래프를 그릴 경우
 - “set_”를 붙이는 경우가 있으니 주의
 - User Guide 참조

https://matplotlib.org/stable/api/axes_api.html

```
fig, axes=plt.subplots(2, 2, sharex=True, sharey=True, figsize=(10, 10))

fig.suptitle('Example4', fontsize=25)

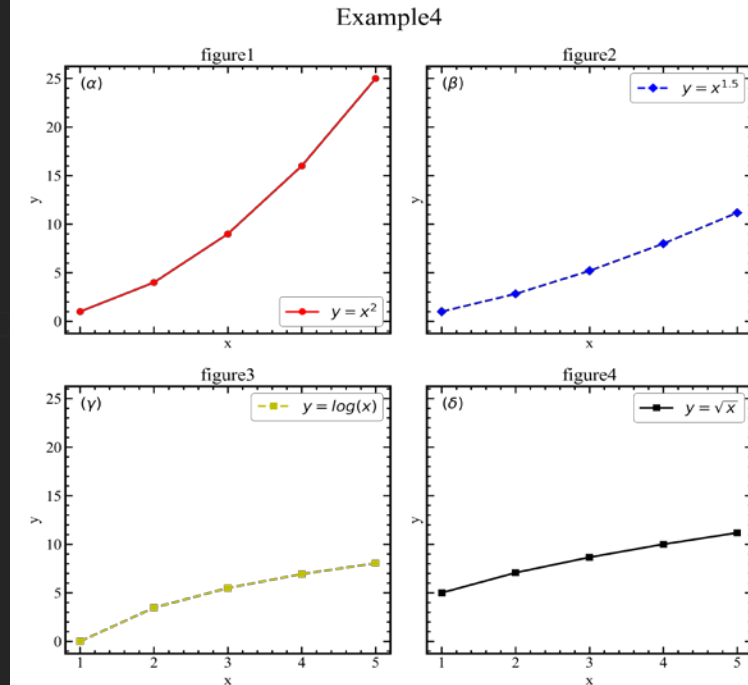
axes[0,0].set_title('figure1')
axes[0,0].plot(x, y1, 'ro-', label=r'$y=x^2$')
axes[0,0].set_xlabel('x')
axes[0,0].set_ylabel('y')
axes[0,0].text(1, 24, r'$(\alpha)$')
axes[0,0].legend(loc='lower right')

axes[0,1].set_title('figure2')
axes[0,1].plot(x, y2, 'bd--', label=r'$y=x^{1.5}$')
axes[0,1].set_xlabel('x')
axes[0,1].set_ylabel('y')
axes[0,1].text(1, 24, r'$(\beta)$')
axes[0,1].legend(loc='upper right')

axes[1,0].set_title('figure3')
axes[1,0].plot(x, y3, 'ys--', label=r'$y=\log(x)$')
axes[1,0].set_xlabel('x')
axes[1,0].set_ylabel('y')
axes[1,0].text(1, 24, r'$(\gamma)$')
axes[1,0].legend(loc='upper right')

axes[1,1].set_title('figure4')
axes[1,1].plot(x, y4, 'ks-', label=r'$y=\sqrt{x}$')
axes[1,1].set_xlabel('x')
axes[1,1].set_ylabel('y')
axes[1,1].text(1, 24, r'$(\delta)$')
axes[1,1].legend(loc='upper right')

plt.tight_layout()
plt.savefig('example4.png', dpi=600, transparent=True)
plt.show()
```



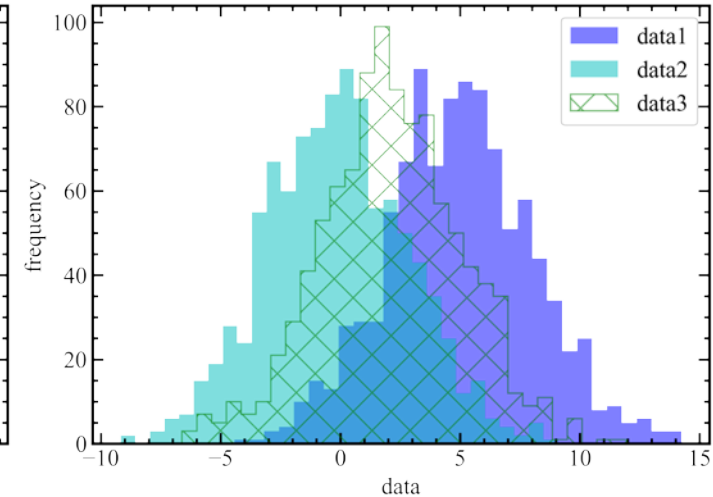
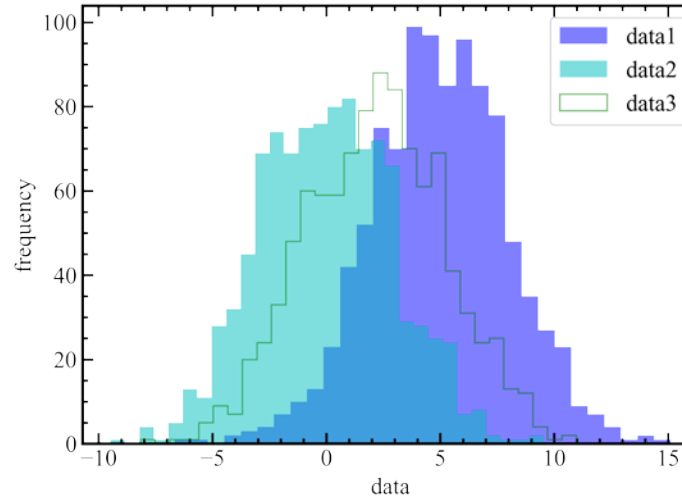
2. 패키지와 모듈

• Matplotlib

• 히스토그램 그리기

→ “plt.hist()”

- ✓ “bins” = bin 개수 (계급 수)
- ✓ “color” = 히스토그램 색
- ✓ “alpha” = 투명도
 - (0 : fully transparent, 1: not transparent)
- ✓ “histtype” = 히스토그램 타입
 - Default는 “bar” 타입(전부 색이 칠해짐)
 - “step”은 윤곽선만
- ✓ “rwidth”: 막대 간격 조정
- ✓ “hatch”: 히스토그램 빗금 채우기
 - “x”는 x자형 빗금을 채워준다. (/ , + , * 등 사용가능)
- ✓ 추가적으로 “density=True”를 하면 Normalized 히스토그램을 그려준다.



```
data1 = np.random.normal(5.0, 3.0, 1000)
data2 = np.random.normal(0.0, 3.0, 1000)
data3 = np.random.normal(2.0, 3.0, 1000)

plt.hist(data1, bins=30, color='b', alpha=0.5, histtype='bar', rwidth=1, label='data1')
plt.hist(data2, bins=30, color='c', alpha=0.5, histtype='bar', rwidth=1, label='data2')
plt.hist(data3, bins=30, color='g', alpha=0.5, histtype='step', rwidth=1, label='data3')

plt.xlabel('data')
plt.ylabel('frequency')
plt.legend()
plt.savefig('example5.png', dpi=600, transparent=True)
plt.show()
```

2. 패키지와 모듈

- Matplotlib

- Image visualization (“plt.imshow()”, “plt.pcolormesh()”)

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
```

```
x = np.arange(-3, 3.1, 0.1)
y = np.arange(-3, 3.1, 0.1)
X, Y = np.meshgrid(x, y)
Z1 = np.exp(-X**2-Y**2)
Z2 = np.exp(-(X-1)**2-(Y-1)**2)
Z = Z1-Z2

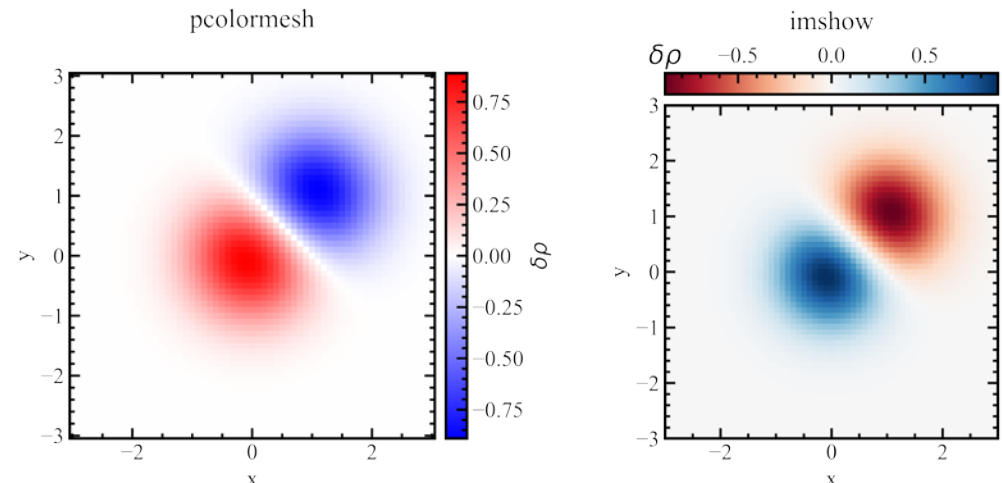
fig, axs = plt.subplots(1, 2, figsize=(10, 5))

axs[0].set_title('pcolormesh', y=1.1)
im=axs[0].pcolormesh(x, y, Z, cmap='bwr')
axs[0].set_xlabel('x')
axs[0].set_ylabel('y')
axs[0].set_aspect(aspect=1)
divider=make_axes_locatable(axs[0])
cax=divider.append_axes("right", size=0.2, pad=0.1)
cbar=plt.colorbar(im, cax=cax)
cbar.set_label(r'$\delta\rho$')

axs[1].set_title('imshow', y=1.2)
im=axs[1].imshow(Z, cmap='RdBu', origin='lower', extent=[-3, 3, -3, 3])
axs[1].set_xlabel('x')
axs[1].set_ylabel('y')
divider=make_axes_locatable(axs[1])
cax=divider.append_axes("top", size=0.2, pad=0.1)
cbar=plt.colorbar(im, cax=cax, orientation='horizontal')
cbar.ax.xaxis.set_ticks_position('top')
cbar.ax.set_title(r'$\delta\rho$', x=0, y=1)

plt.tight_layout()
plt.savefig('example6.png', dpi=600, transparent=True)
plt.show()
```

- ✓ “np.meshgrid()” = (x_i, y_j) 의 격자를 만드는 함수, 2차원 배열로 값을 생성
- ✓ “plt.imshow()” or “plt.pcolormesh()” = contour image를 생성
 - **plt.pcolormesh()**은 격자점을 직접 입력, 일반 이미지를 불러올 땐 격자점을 따로 생성해야해서 번거롭지만, **비균등 격자점에 대해서도 사용할 수 있는 장점**이 있다.
- ✓ “make_axes_locatable()” = colorbar를 그리기 위한 새로운 축 cax를 subplot 내에 생성
- ✓ “.append_axes()”를 통해 위치와 크기를 부여한다.
- ✓ “plt.colorbar()”를 통해 해당하는 contour image의 cax에 색을 부여한다.



2. 패키지와 모듈

- Matplotlib

- Image visualization (“plt.contourf()” and “plt.contour()”)

- “plt.contourf()” = 색을 채운 등고선을 그려준다. 격자점과 값을 모두 입력해야 한다.

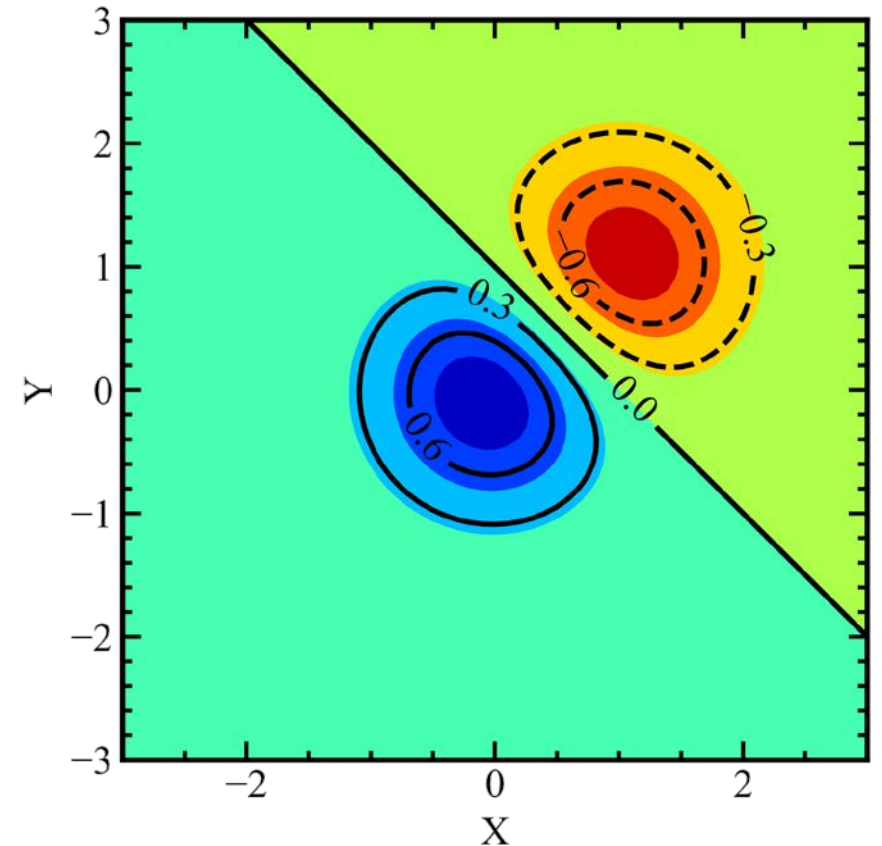
- ✓ “cmap”을 통해 contour의 색을 바꿀 수 있다.

- ✓ <https://matplotlib.org/stable/tutorials/colors/colormaps.html>

- “plt.contour()” = “levels”에 할당된 등고선을 그려준다.

```
lvs=np.linspace(-0.6, 0.6, 5)

fig=plt.figure(1, figsize=(5,5))
plt.contourf(X,Y,Z, cmap='jet_r')
plt.xlabel('X')
plt.ylabel('Y')
cs=plt.contour(X,Y,Z,levels=lvs, colors='k')
plt.clabel(cs)
plt.savefig('example7.png', dpi=600, )
plt.show()
```



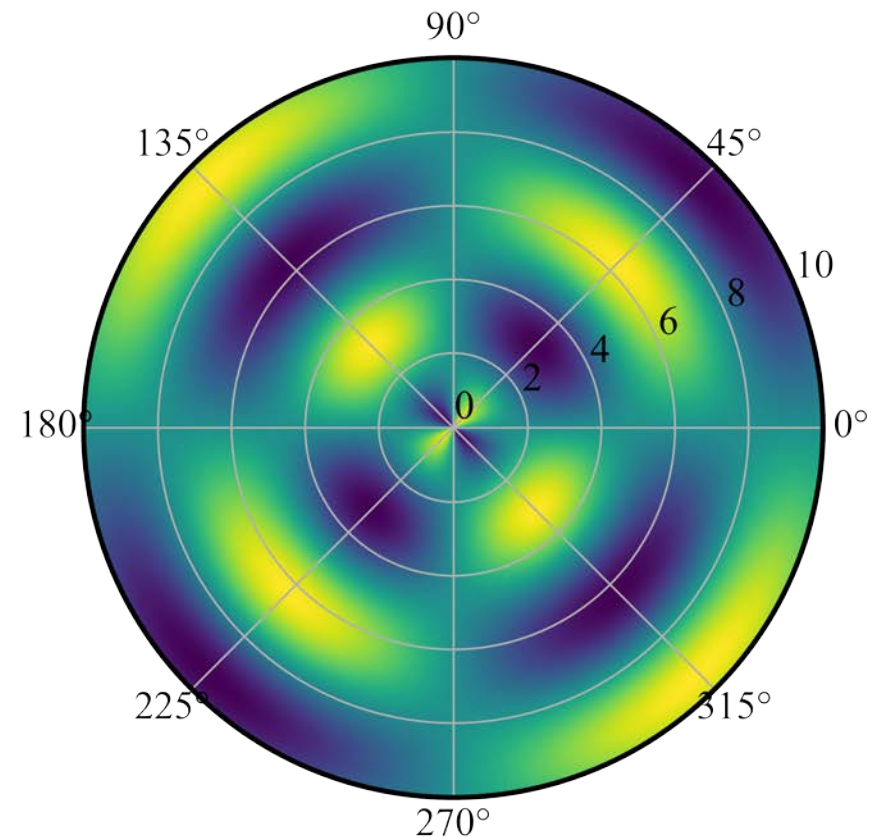
2. 패키지와 모듈

- **Matplotlib**

- Image visualization (“plt.subplot(projection=”)”)
 - “plt.pcolormesh”를 이용하면 비균등(non-uniform) 격자점을 그리는 데에도 유용하다.

```
theta = np.linspace(0, 2*np.pi, 361)
r = np.linspace(0, 10, 201)
R, Theta = np.meshgrid(r, theta)
data = np.sin(2*Theta)*np.cos(R)

plt.subplot(projection='polar')
plt.pcolormesh(theta, r, data.T)
plt.savefig('example8.png', dpi=600, transparent=True)
plt.show()
```



2. 패키지와 모듈

- Numpy

<https://numpy.org/devdocs/reference/index.html>

- Matplotlib

<https://matplotlib.org/stable/gallery/index.html#>

Q&A *Thanks for listening*

