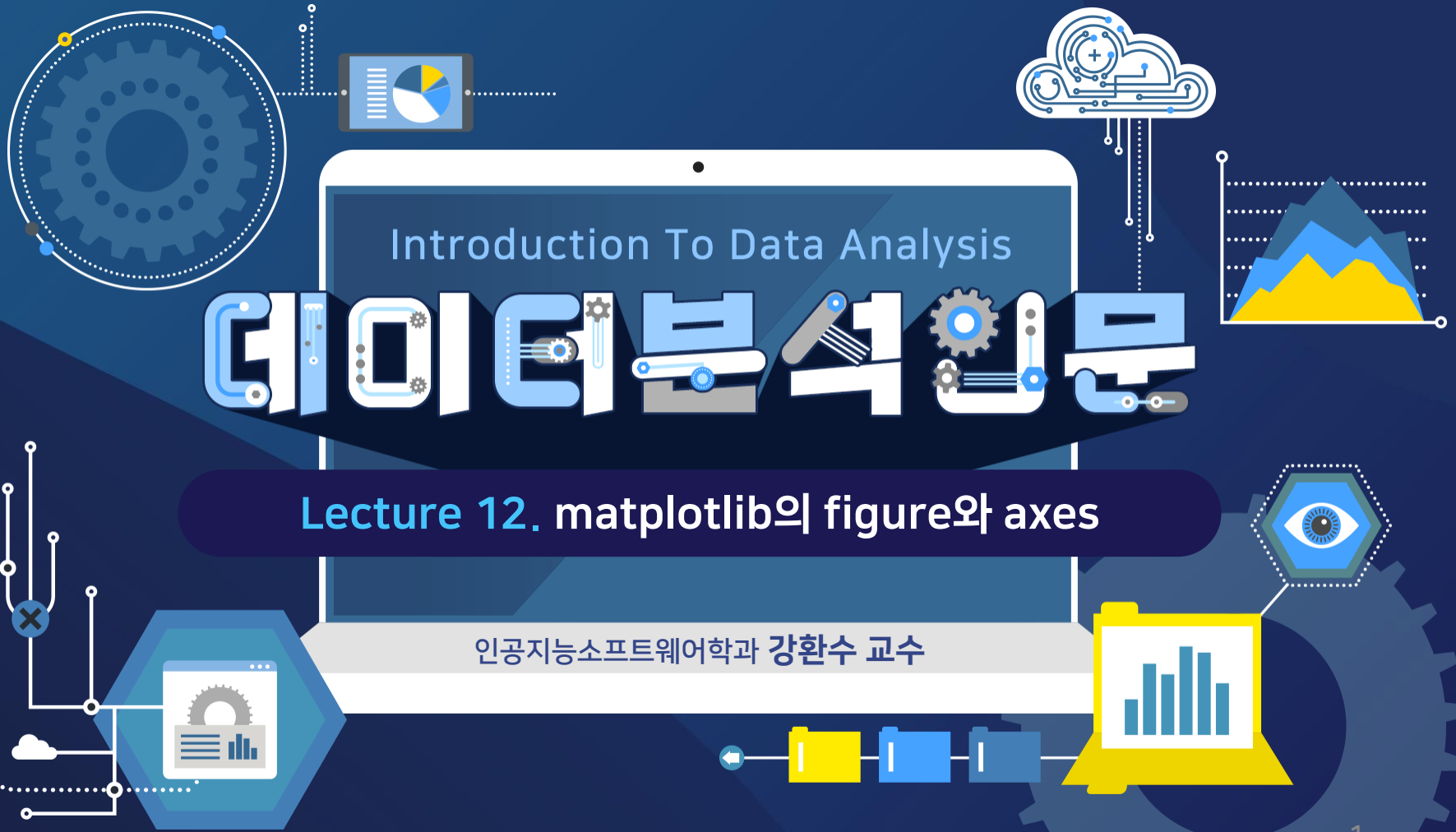

Introduction To Data Analysis
데이터분석입문
Lecture 12. matplotlib의 figure와 axes
인공지능소프트웨어학과 강환수 교수

학습개요

- 도화지인 figure와 도화지 크기 지정
- 부분 그림을 그리기 위한 figure와 axes
- 격자(gird) 형태의 부분 그림 설정
- 부분 그림을 그리기 위한 기초

학습목표

- 도화지인 figure에서 크기를 지정할 수 있다.
- 격자 형태의 부분 그림에서 위치를 알고 그래프를 그릴 수 있다.
- 다음을 사용해 부분 그림을 그릴 수 있다.

  plt.subplot() · plt.subplots() · fig.add_subplot()

LESSON 01

figure와 subplot

figure 개요

figure 개요

- 그려지는 그래프가 figure 객체 내에 존재하며, 그림을 그리는 종이라고 이해

fig = plt.figure(figsize=(10, 6))

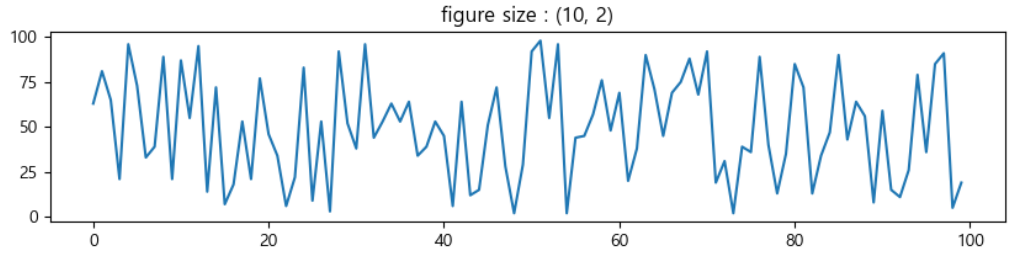
- figure 명령을 사용하여 그 반환 값으로 figure 객체를 얻음
  - 일반적인 plot 명령 등을 실행하면 자동으로 figure를 생성해 주기 때문에 일반적으로는 figure 명령을 잘 사용하지 않음
- figure 명령을 명시적으로 사용하는 경우
  - Jupyter 노트북에서 그림의 크기를 설정할 경우
- figsize=(10, 6)
  - 그림 크기: figsize 인수로 설정(크기와 비율을 지정)
  - 단위: inch

figure 개요

- w, h = 10, 2
- f1 = plt.figure(figsize=(w, h))

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

w, h = 10, 2
f1 = plt.figure(figsize=(w, h))
plt.plot([random.randint(1, 100) for i in range(100)])

plt.title(f"figure size : ({w}, {h})")
plt.show()
```

✓ 0.1s Python



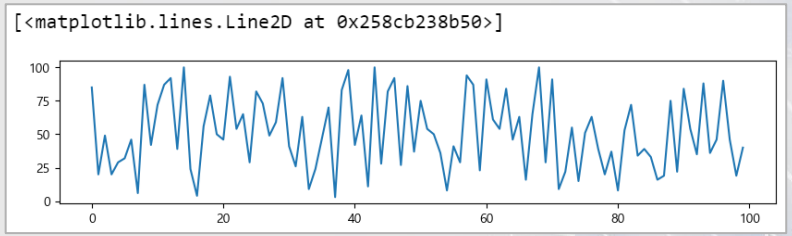
show()

그림 그리기(rendering)

- 시각화 명령을 실제로 차트로 렌더링(rendering)
- 마우스 움직임 등의 이벤트를 기다리라는 지시

주피터 노트북

- 셀 단위로 플롯 명령을 자동 렌더링 해주므로 show() 구문이 필요 없음
- show() 구문이 없다면 (원하지 않은)
  - plot() 등의 이전 구문의 반환 값이 표시

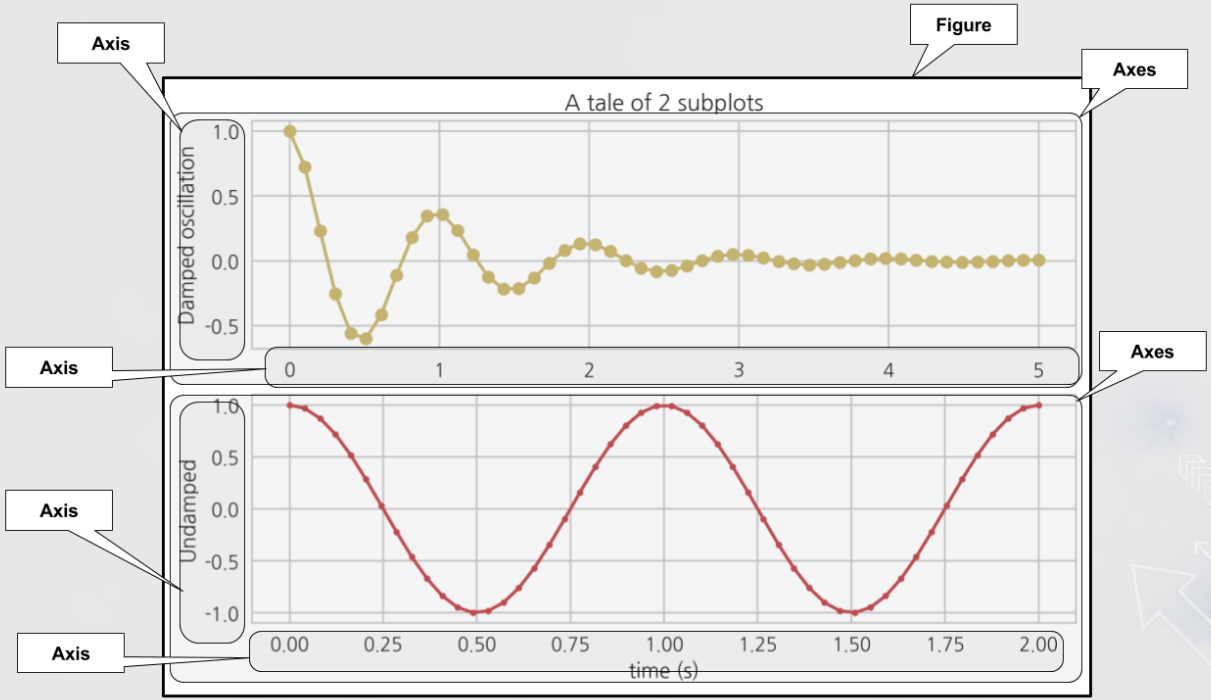


- 일반 파이썬 인터프리터
  - show()가 필요
  - 일반적인 코드가 되도록 항상 마지막에 실행

figure 구조

figure: 그림이 그려지는 캔버스나 종이

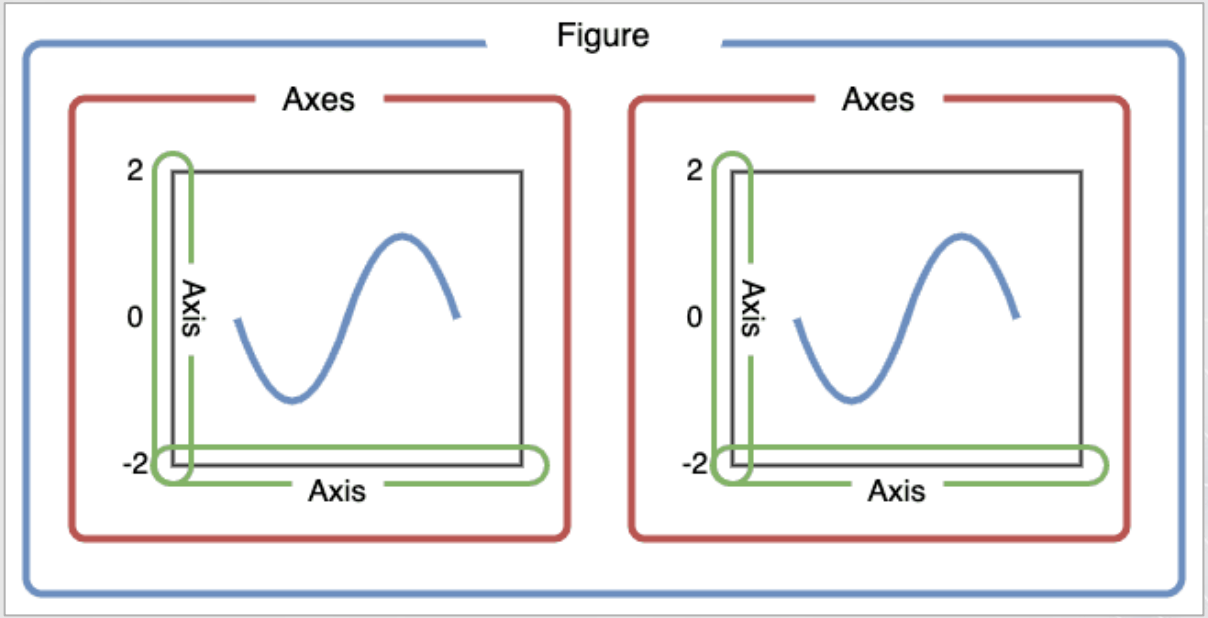
- Axes: figure 내부의 부분 그림
- Axis: 가로축이나 세로축 등의 축



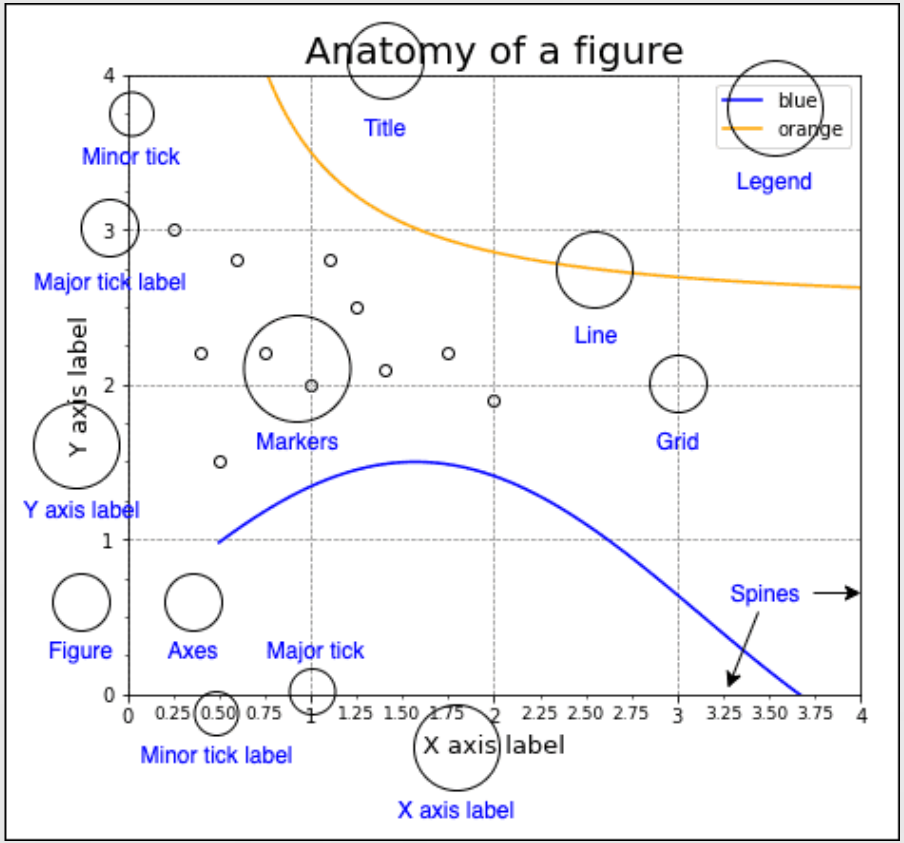
figure와 axes

격자(grid) 형태의 Axes 객체들을 생성

- figure 안에 있는 각각의 부분 그림은 Axes 라고 불리는 객체
  - figure 내부에 행렬(matrix) 형태의 여러 그림이 있으며 이를 axes라 함



figure의 다양한 요소



LESSON 02

figure 내부 부분 그림 axes

부분 그림 axes에 그래프 그리는 방법 기초

plt.subplot()

- 가장 낮은 수준의 접근 방법이지만 가장 많이 사용되는 방법 중 하나

plt.subplots()

- 가장 간단하며 가독성이 좋은 방법

fig.add_subplot()

- figure 객체로 서브플롯의 접근에 사용

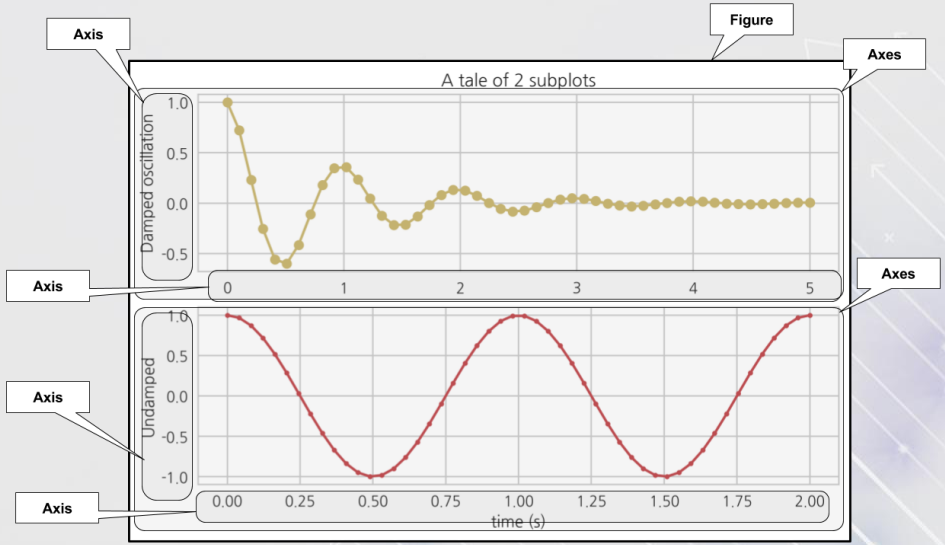
plt.subplot

matplotlib.pyplot 모듈의 subplot() 함수

- 여러 개의 그래프(axes)를 하나의 그림(figure)에 나타내도록
  - 그리드(grid) 형태의 Axes 객체들을 생성

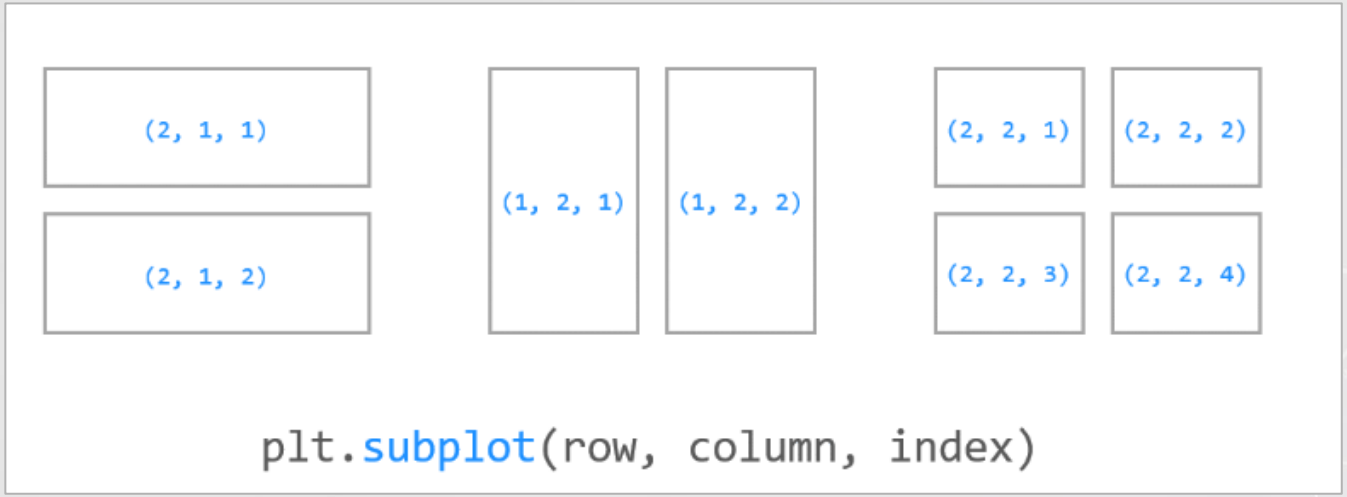
ax1 = plt.subplot(2, 1, 1)

- subplot(m, n, number)
- ax1 = plt.subplot('211') 로도 가능
  - 단 단위만 가능
- 세 개의 인수
  - m, n
    - 전체 그리드 행렬의 모양을 지시하는 두 숫자
  - number
    - 인수가 네 개 중 어느 것인지를 의미하는 숫자
    - 첫번째 부분그림을 가리키는 숫자가 0이 아니라 1임에 주의



plt.subplot(row, column, index)

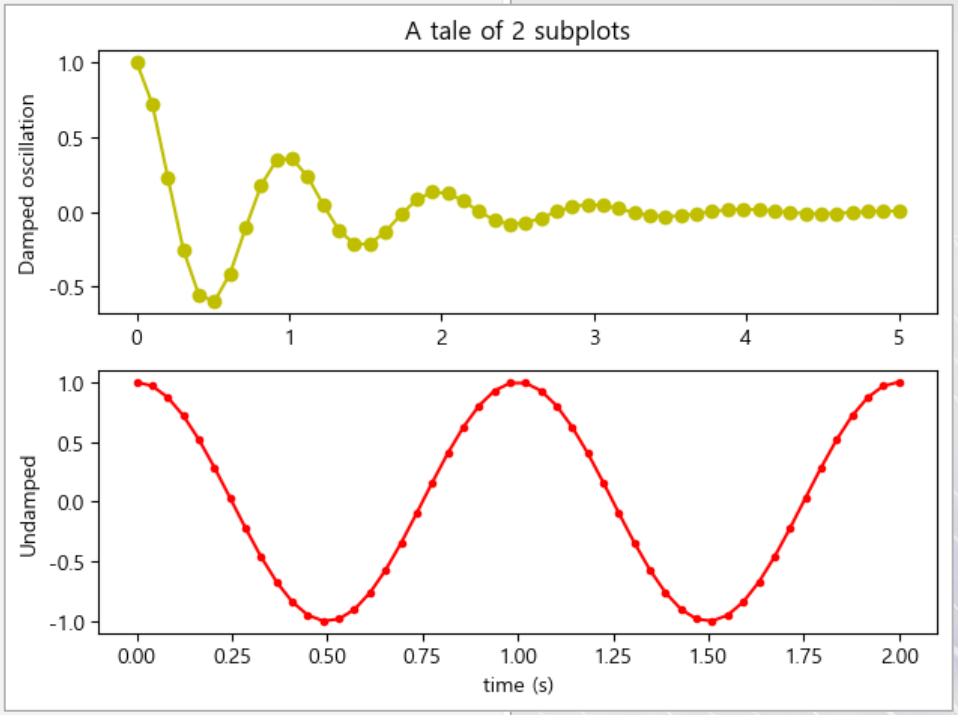
가로, 세로, 현재



부분그림 2행 1열

```
import matplotlib.pyplot as plt

x1 = np.linspace(0.0, 5.0) # num=50 기본 값
x2 = np.linspace(0.0, 2.0) # num=50 기본 값
y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

ax1 = plt.subplot(2, 1, 1)
plt.plot(x1, y1, 'yo-')
plt.title('A tale of 2 subplots')
plt.ylabel('Damped oscillation')

ax2 = plt.subplot(2, 1, 2)
plt.plot(x2, y2, 'r.-')
plt.xlabel('time (s)')
plt.ylabel('Undamped')

plt.tight_layout()
plt.show()
```

✓ 0.2s Python



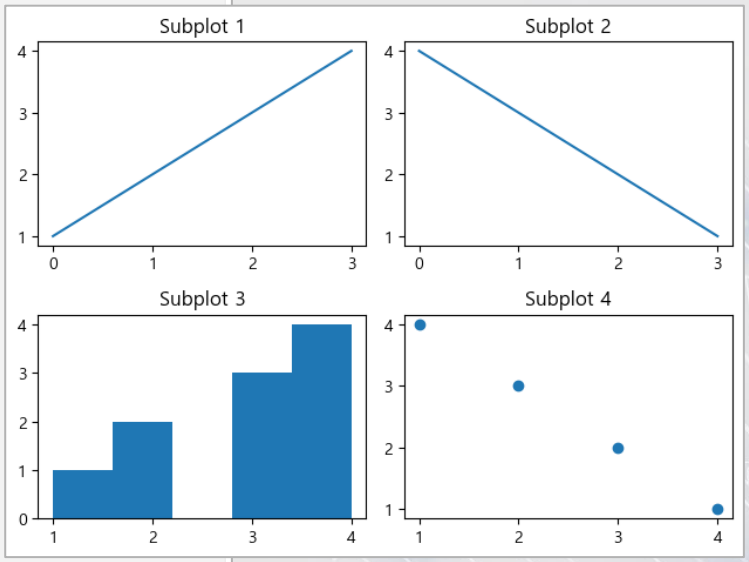
plt.subplots() 개요

plt.subplots(rows, columns) 함수

- 여러 개의 axes을 한 번에 생성하는 간단한 방법을 제공
- plt.subplots()는 행과 열의 개수를 지정하고 모든 axes을 반환

```python
import matplotlib.pyplot as plt

fig, axes = plt.subplots(2, 2)  # 2행 2열의 서브플롯

axes[0, 0].plot([1, 2, 3, 4])
axes[0, 0].set_title("Subplot 1")

axes[0, 1].plot([4, 3, 2, 1])
axes[0, 1].set_title("Subplot 2")

axes[1, 0].hist([1, 2, 2, 3, 3, 3, 4, 4, 4, 4], bins=5)
axes[1, 0].set_title("Subplot 3")

axes[1, 1].scatter([1, 2, 3, 4], [4, 3, 2, 1])
axes[1, 1].set_title("Subplot 4")

plt.tight_layout()
plt.show()
```

0.6s Python



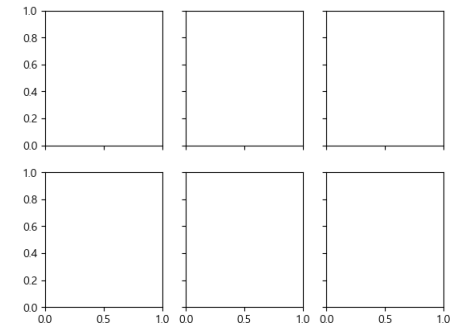
plt.subplots(m, n)

메소드

- matplotlib.pyplot.subplots(nrows=1, ncols=1, sharex=False, sharey=False, squeeze=True, subplot_kw=None, gridspec_kw=None, **fig_kw)
- 특정한 배치에 맞추어 여러 개의 서브플롯을 포함하는 figure를 생성
  - 인자 sharex=True, sharey=True
    - 각 축의 인자를 하나로 공유

```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(2, 3, sharex=True, sharey=True)
plt.show()
```

0.4s Python

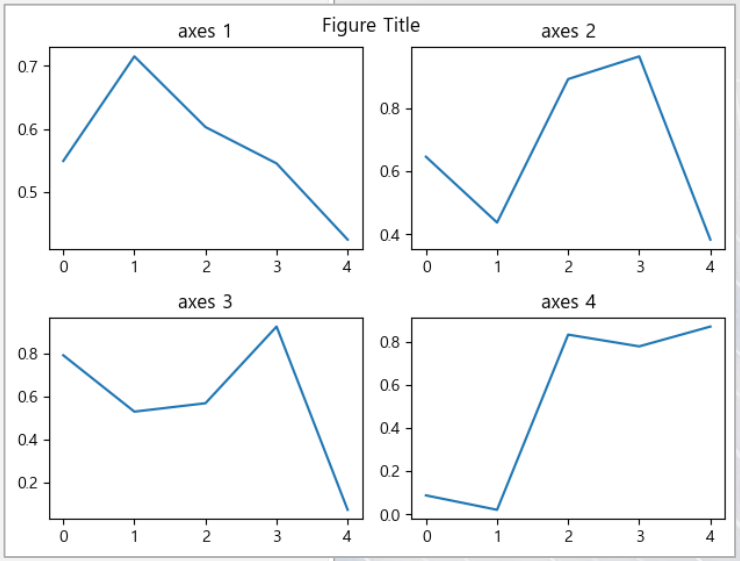


plt.subplots() 전체 제목과 부 제목

- axes[0, 0].set_title("axes 1")
- plt.suptitle('Figure Title')

```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(2, 2)
np.random.seed(0)
axes[0, 0].plot(np.random.rand(5)) # [0, 1) 난수 5개
axes[0, 0].set_title("axes 1")
axes[0, 1].plot(np.random.rand(5))
axes[0, 1].set_title("axes 2")
axes[1, 0].plot(np.random.rand(5))
axes[1, 0].set_title("axes 3")
axes[1, 1].plot(np.random.rand(5))
axes[1, 1].set_title("axes 4")

plt.tight_layout()
plt.suptitle('Figure Title')
plt.show()
```

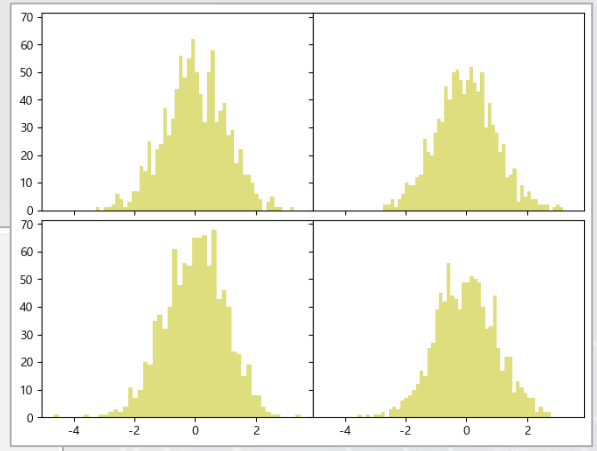
✓ 0.4s Python



부분 그림 간의 간격 조절하기

plt.subplots_adjust(wspace=0, hspace=.05)

- axes 간의 간격을 설정
  - 비율로 명시, .05은 5%의 공간 비우기




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

fig, axes = plt.subplots(2, 2, sharex=True, sharey=True, figsize=(8, 6))

for i in range(2):
    for j in range(2):
        axes[i, j].hist(np.random.randn(1000), bins=50, color='y', alpha=0.5)

print(plt.rcParams['figure.subplot.wspace'])
print(plt.rcParams['figure.subplot.hspace'])

# 적정한 공간 비우기, 코드없는 것과 동일(기본 값): wspace=.2, hspace=.2
plt.subplots_adjust(wspace=0, hspace=.05)
plt.show()
```

✓ 0.5s Python

```
0.2
0.2
```

fig.add_subplot() 개요

fig.add_subplot() 메서드

- axes을 하나씩 추가할 수 있도록 figure 객체에 속한 메서드
- axes을 생성하고 반환하며, 인덱스를 이용하여 위치를 지정

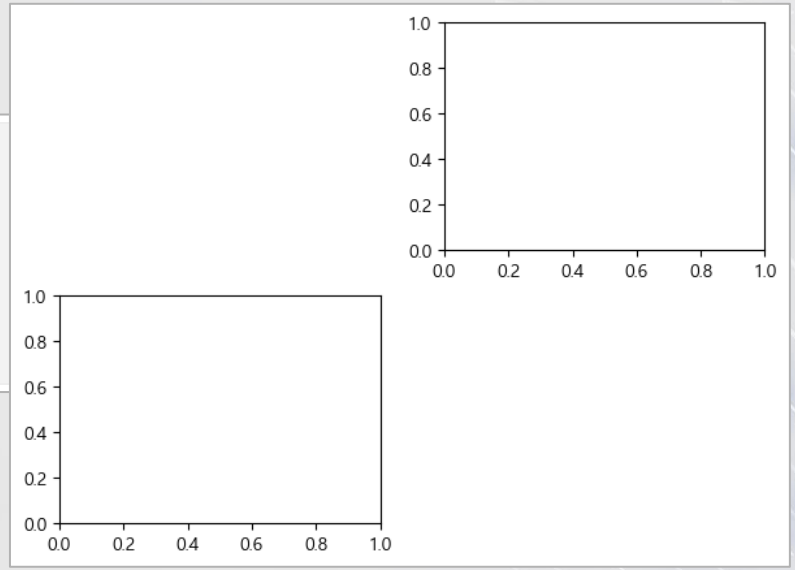
fig.add_subplot() 활용

Axes: 내부의 부분 그림

- ax1 = fig.add_subplot(2, 2, 1)
  - 크기 2 x 2
  - 4개의 그림 중 1번 그림
- ax2 = fig.add_subplot(2, 2, 2)
- ax3 = fig.add_subplot(2, 2, 3)

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(7, 5))
# ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
```

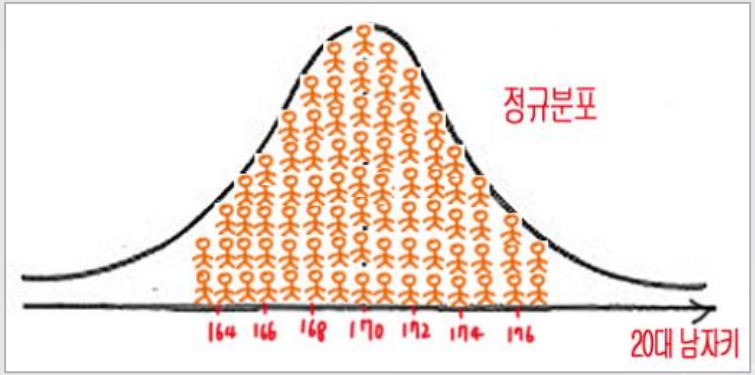
✓ 0.2s



표준 정규 분포

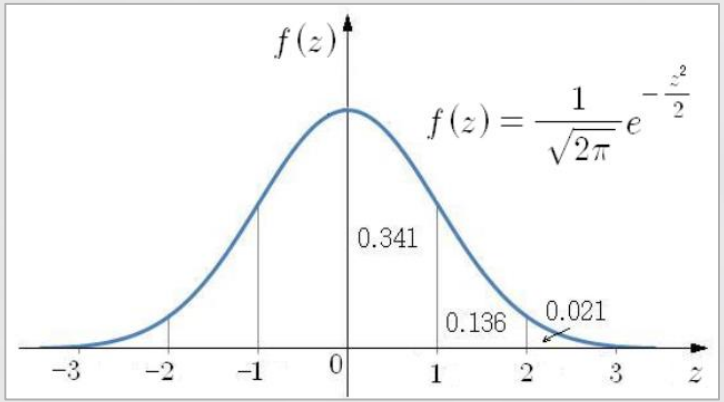
정규 분포

- 가상적인 분포
  - 우리나라 20대 남자키 분포



표준 정규 분포

- 평균 0, 표준편차 1

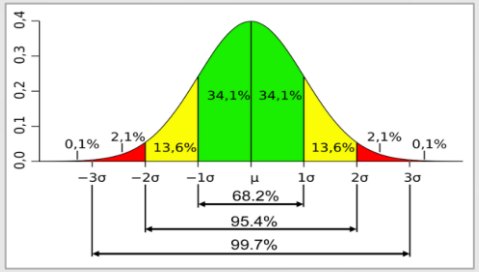


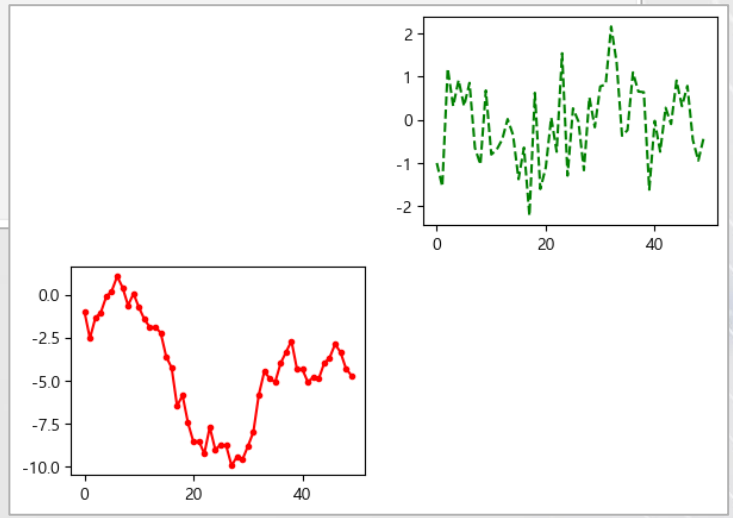
np.random.randn()

표준 정규 분포의 난수를 발생

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

fig = plt.figure(figsize=(7, 5))
data = np.random.randn(50) # 표준 정규분포의 난수 50개

ax2 = fig.add_subplot(2, 2, 2)
plt.plot(data, 'g--')
ax3 = fig.add_subplot(2, 2, 3)
plt.plot(data.cumsum(), 'r.-')
plt.show()
```

✓ 0.1s





SUMMARY

학습정리

도화지인 figure에서 크기를 지정

```
w, h = 10, 2
f1 = plt.figure(figsize=(w, h))
```

plt.subplot()

```
ax1 = plt.subplot(2, 1, 1)
plt.plot(x1, y1, 'yo-’)
…
ax2 = plt.subplot(2, 1, 2)
plt.plot(x2, y2, 'r.-’)
…
```

plt.subplots()

```
fig, axes = plt.subplots(2, 2)  # 2행 2열의 서브플롯

axes[0, 0].plot([1, 2, 3, 4])
axes[0, 0].set_title("Subplot 1")

axes[0, 1].plot([4, 3, 2, 1])
axes[0, 1].set_title("Subplot 2")
```

fig.add_subplot()

```
fig = plt.figure(figsize=(7, 5))
# ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
```