빅데이터 실습

9주차 3차시

데이터 시각화 - matplotlib 실습 [2]

데이터 시각화
matplotlib 실습 [2]

학습개요

1/ backend의 개념 및 종류

2/ Matplotlib 관련 설정 변경하기

backend의 개념 및 종류

Matplotlib 관련 설정 변경하기

01

backend의 개념 및 종류

backend 개념

Matplotlib

- ✓ 오래된 파이썬 데이터 시각화 라이브러리
- ✓ 매우 다양한 usecase에 사용됨
- ✓ 다양한 usecase에 하나의 동작 방식으로는 한계가 있기 때문에 backend라는 개념을 사용

backend option

◎ Inline

✔ output 창에 그대로 표현

Run Code

e를 실행하여 설정한다.

%로 시작하는 명령어는 Cell Magic 명령어로써, 파이썬 코드가 아니라 Jupyter에게 특정 기능을 수행하도록 하는 명령이다. 즉, %matplotlib inline 명령은 Jupyter에게 matplotlib 그래프를 출력 영역에 표시할 것을 지시하는 명령이다.

```
In [43]: %matplotlib inline
```

```
In [ ]:
```

만약 그래프를 생성한 이후, interactive하게 그래프를 다루고 싶은 경우에는 backend를 interactive backend로 설정하면 된다.
예를 들어, **%matplotlib nbagg**를 실행하여 **nbagg**으로 설정을 변경할 수 있다.
보다 다양한 설정 옵션을 확인하려면 여기 를 참고하면 된다.

아래와 같이 nbagg로 설정한 후, 그래프를 생성하면 output 창에 몇몇 버튼과 함께 그래프가 그려진다.
버튼들을 활용하여 zoon-in, zoom-out이 가능하다.
그래프를 그린 이후, set_title(), set_xlabel() 함수를 수행하면 현재 그래프에 적용되는 것을 확인할 수 있으며, 오른쪽 상단에 있는 파란색 버튼을 클릭하면 interactive 모드가 종료된다.

backend option

◎ %로 시작하는 명령어

- ✔ Cell Magic 명령어
- ✔ Jupyter에게 특정 기능을 수행하도록 하는 명령어

코드가 아니라 Jupyter에게 특정 기능을

Jupyter에게 matplotlib 그래프를 출력 영

역에 표시할 것을 지시하는 명령이다.

```
In [43]: %matplotlib inline

In [ ]:

In [ ]:
```

만약 그래프를 생성한 이후, interactive하게 그래프를 다루고 싶은 경우에는 backend를 interactive backend로 설정하면 된다.
예를 들어, **%matplotlib nbagg**를 실행하여 **nbagg**으로 설정을 변경할 수 있다.
보다 다양한 설정 옵션을 확인하려면 여기 를 참고하면 된다.

Matplotlib으로 그래프를 그릴 때는 출력 창에 그래프를 인라인으로 바로 표시하라고 명령하는 것입니다.

backend option

◎ interactive하게 그래프를 다루고 싶은 경우 interactive backend 제공

In [ ]:

만약 그래프를 생성한 이후, interactive하게 그래프를 다루고 싶은 경우에는 backend를 interactive backend로 설정하면 된다.
예를 들어, **%matplotlib nbagg**를 실행하여 **nbagg**으로 설정을 변경할 수 있다.
보다 다양한 설정 옵션을 확인하려면 여기 를 참고하면 된다.

아래와 같이 nbagg로 설정한 후, 그래프를 생성하면 output 창에 몇몇 버튼과 함께 그래프가 그려진다.
버튼들을 활용하여 zoon-in, zoom-out이 가능하다.
그래프를 그린 이후, set_title(), set_xlabel() 함수를 수행하면 현재 그래프에 적용되는 것을 확인할 수 있으며, 오른쪽 상단에 있는 파란색 버튼을 클릭하면 interactive 모드가 종료된다.

In [ ]:

In [ ]:

In [ ]:

backend option

◎ 다양한 그래프를 그리는 방식들의 backend 제공

matplotlib
Version 3.1.2

Fork me on GitHub

This tutorial covers some basic usage patterns and best-practices to help you get started with Matplotlib.

`matplotlib` has an extensive codebase that can be daunting to many new users. However, most of matplotlib can be understood with a fairly simple conceptual framework and knowledge of a few important points.

Plotting requires action on a range of levels, from the most general (e.g., 'contour this 2-D array') to the most specific (e.g., 'color this screen pixel red'). The purpose of a plotting package is to assist you in visualizing your data as easily as possible, with all the necessary control -- that is, by using relatively high-level commands most of the time, and still have the ability to use the low-level commands when needed.

Therefore, everything in matplotlib is organized in a hierarchy. At the top of the hierarchy is the matplotlib "state-machine environment" which is provided by the `matplotlib.pyplot` module. At this level, simple functions are used to add plot elements (lines, images, text, etc.) to the current axes in the current figure.

> **Note**
>
> Pyplot's state-machine environment behaves similarly to MATLAB and should be most familiar to users with MATLAB experience.

The next level down in the hierarchy is the first level of the object-oriented interface, in which pyplot is used only for a few functions such as figure creation, and the user explicitly creates and keeps track of the figure and axes objects. At this level, the user uses pyplot to create figures, and through those figures, one or more axes objects can be created. These axes objects are then used for most plotting actions.

For even more control -- which is essential for things like embedding matplotlib plots in GUI applications -- the pyplot level may be dropped completely, leaving a purely object-oriented approach.

```
# sphinx_gallery_thumbnail_number = 3
import matplotlib.pyplot as plt
import numpy as np
```

backend option

◎ 다양한 그래프를 그리는 방식들의 backend 제공

- ✔ inline으로 output창이 바로 표현해 주는 방식
- ✔ interactive하게 표현해 주는 방식

File Edit View Insert Cell Kernel Widgets Help

만약 그래프를 생성한 이후, interactive하게 그래프를 다루고 싶은 경우에는 backend를 interactive backend로 설정하면 된다.
예를 들어, **%matplotlib nbagg**를 실행하여 **nbagg**으로 설정을 변경할 수 있다.
보다 다양한 설정 옵션을 확인하려면 여기 를 참고하면 된다.

아래와 같이 nbagg로 설정한 후, 그래프를 생성하면 output 창에 몇몇 버튼과 함께 그래프가 그려진다.
버튼들을 활용하여 zoon-in, zoom-out이 가능하다.
그래프를 그린 이후, set_title(), set_xlabel() 함수를 수행하면 현재 그래프에 적용되는 것을 확인할 수 있으며, 오른쪽 상단에 있는 파란색 버튼을 클릭하면 interactive 모드가 종료된다.

In [ ]:

In [ ]:

In [ ]:

backend option

◎ nbagg

In [ ]:

만약 그래프를 생성한 이후, interactive하게 그래프를 다루고 싶은 경우에는 backend를 interactive backend로 설정하면 된다.
예를 들어, **%matplotlib nbagg**를 실행하여 **nbagg**으로 설정을 변경할 수 있다.
보다 다양한 설정 옵션을 확인하려면 여기 를 참고하면 된다.

아래와 같이 nbagg로 설정한 후, 그래프를 생성하면 output 창에 몇몇 버튼과 함께 그래프가 그려진다.
버튼들을 활용하여 zoon-in, zoom-out이 가능하다.
그래프를 그린 이후, set_title(), set_xlabel() 함수를 수행하면 현재 그래프에 적용되는 것을 확인할 수 있으며, 오른쪽 상단에 있는 파란색 버튼을 클릭하면 interactive 모드가 종료된다.

In [44]:
```
%matplotlib nbagg
```

In [ ]:
```
plt.plot(sr)
```

In [ ]:

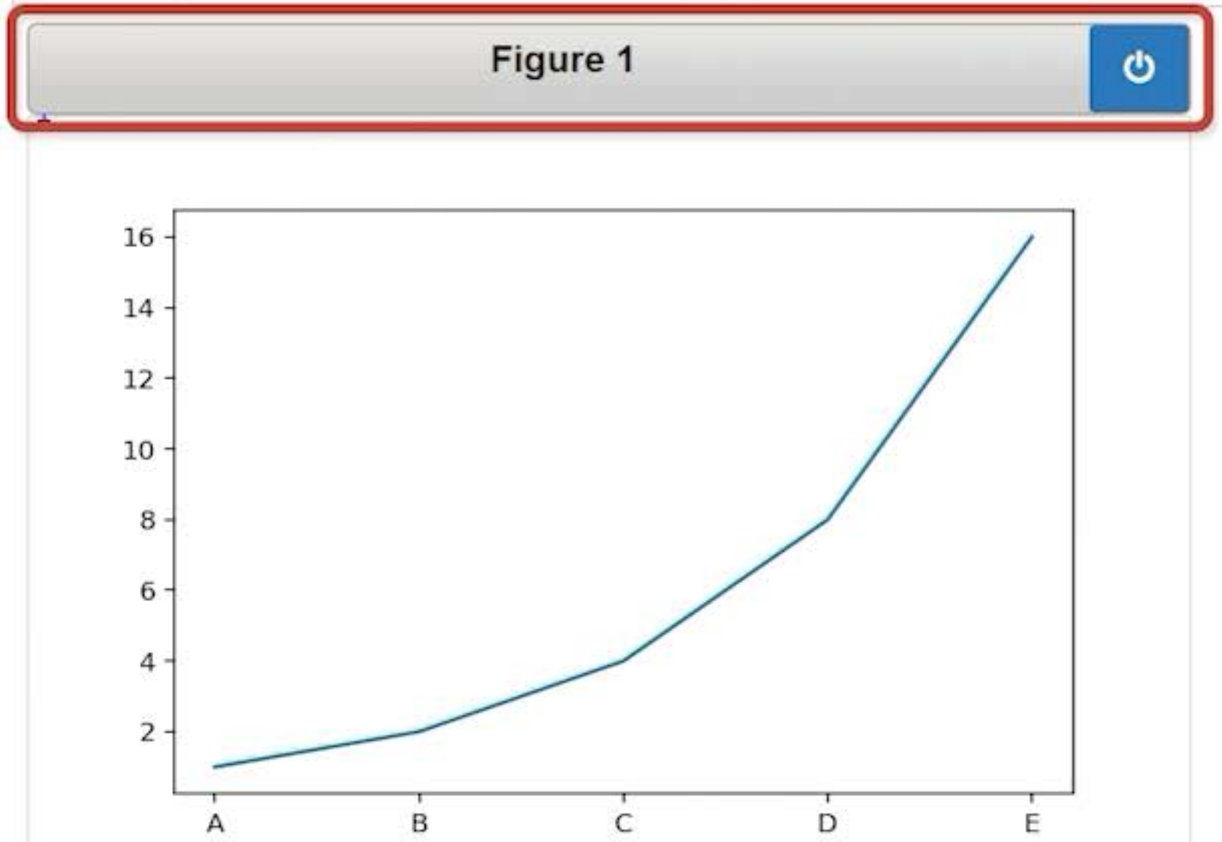
backend option

◎ nbagg

Run Code

In [44]:
```
%matplotlib nbagg
```

In [45]:
```
plt.plot(sr)
```

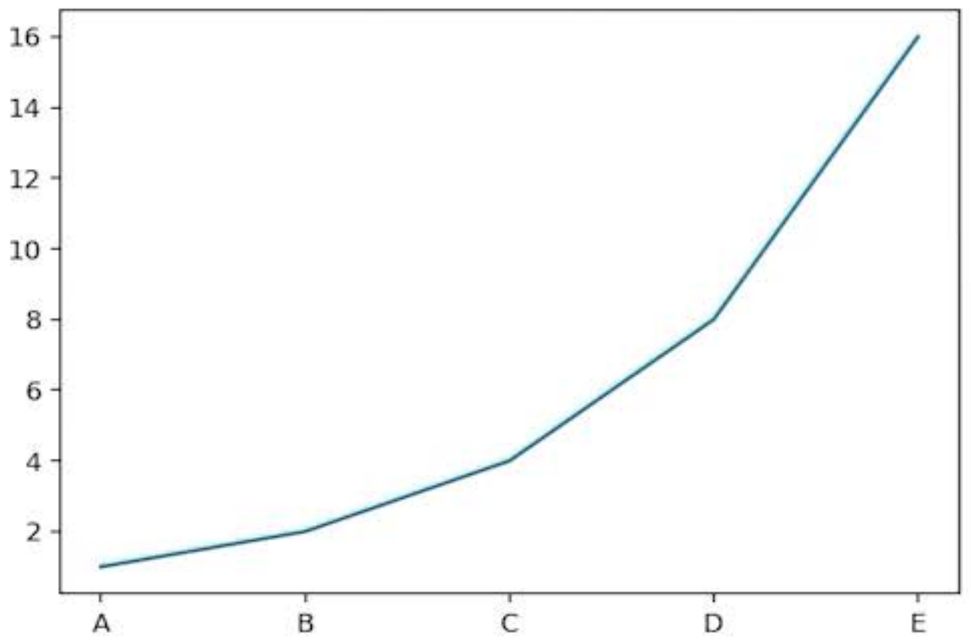


backend option

◎ nbagg

Run Code

Figure 1



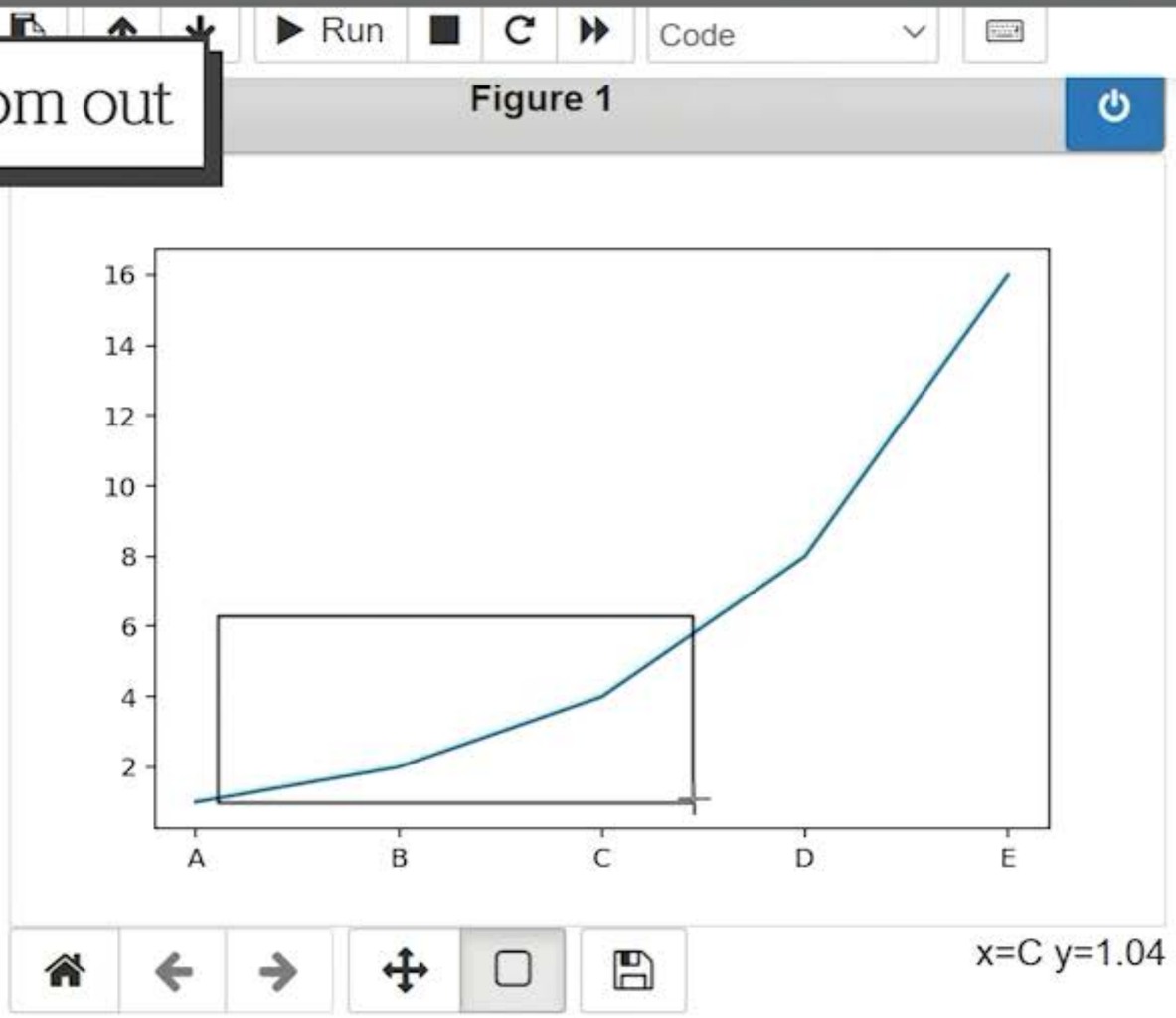
Left button pans, Right button zooms x/y fixes axis, CTRL fixes aspect

Out[45]: [<matplotlib.lines.Line2D at 0x2015b130910>]

backend option

◎ nbagg

✔ Zoom in, Zoom out



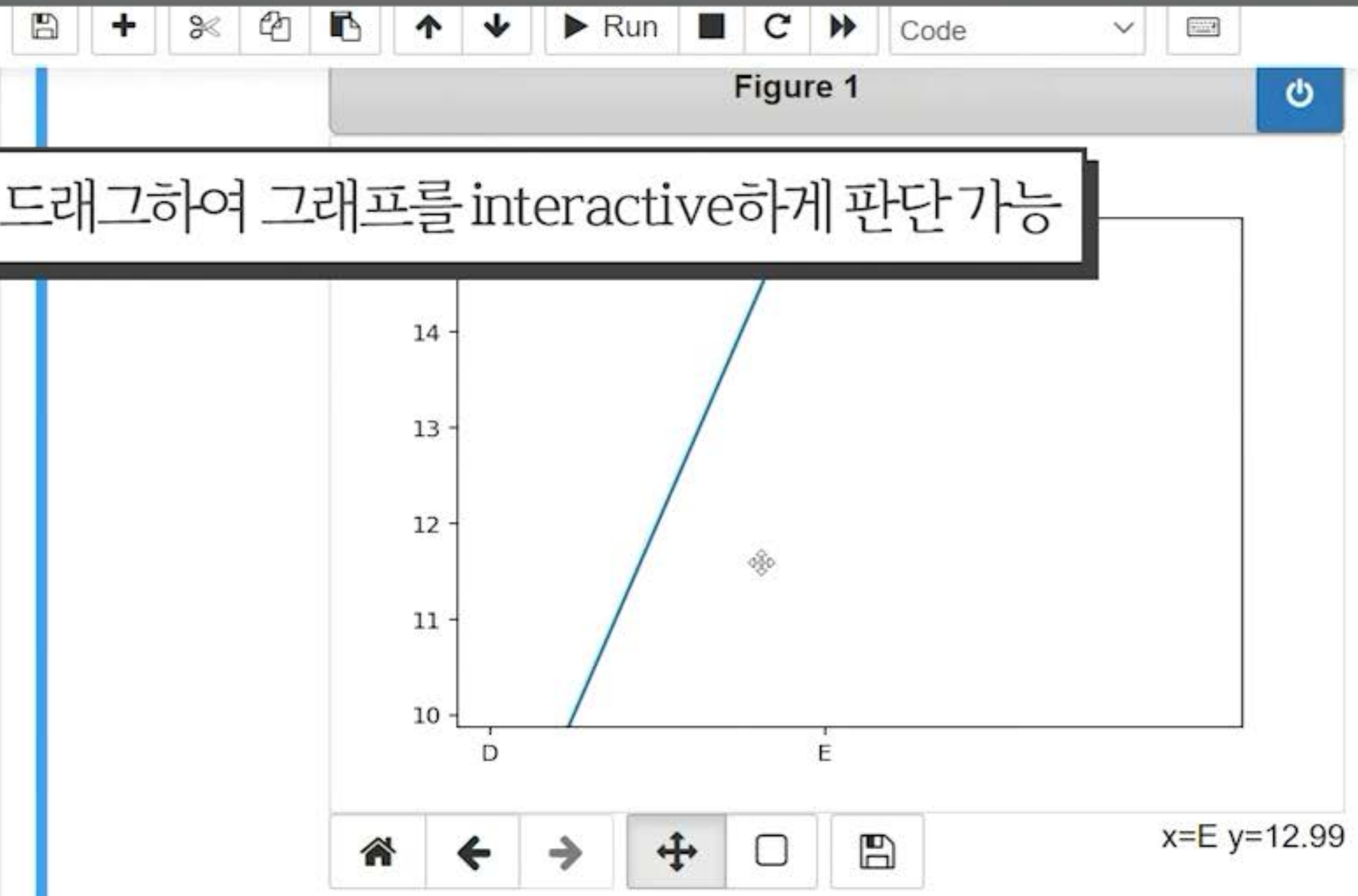
Out[45]: [<matplotlib.lines.Line2D at 0x2015b130910>]

In [ ]:

backend option

◎ nbagg

✔ 드래그하여 그래프를 interactive하게 판단 가능



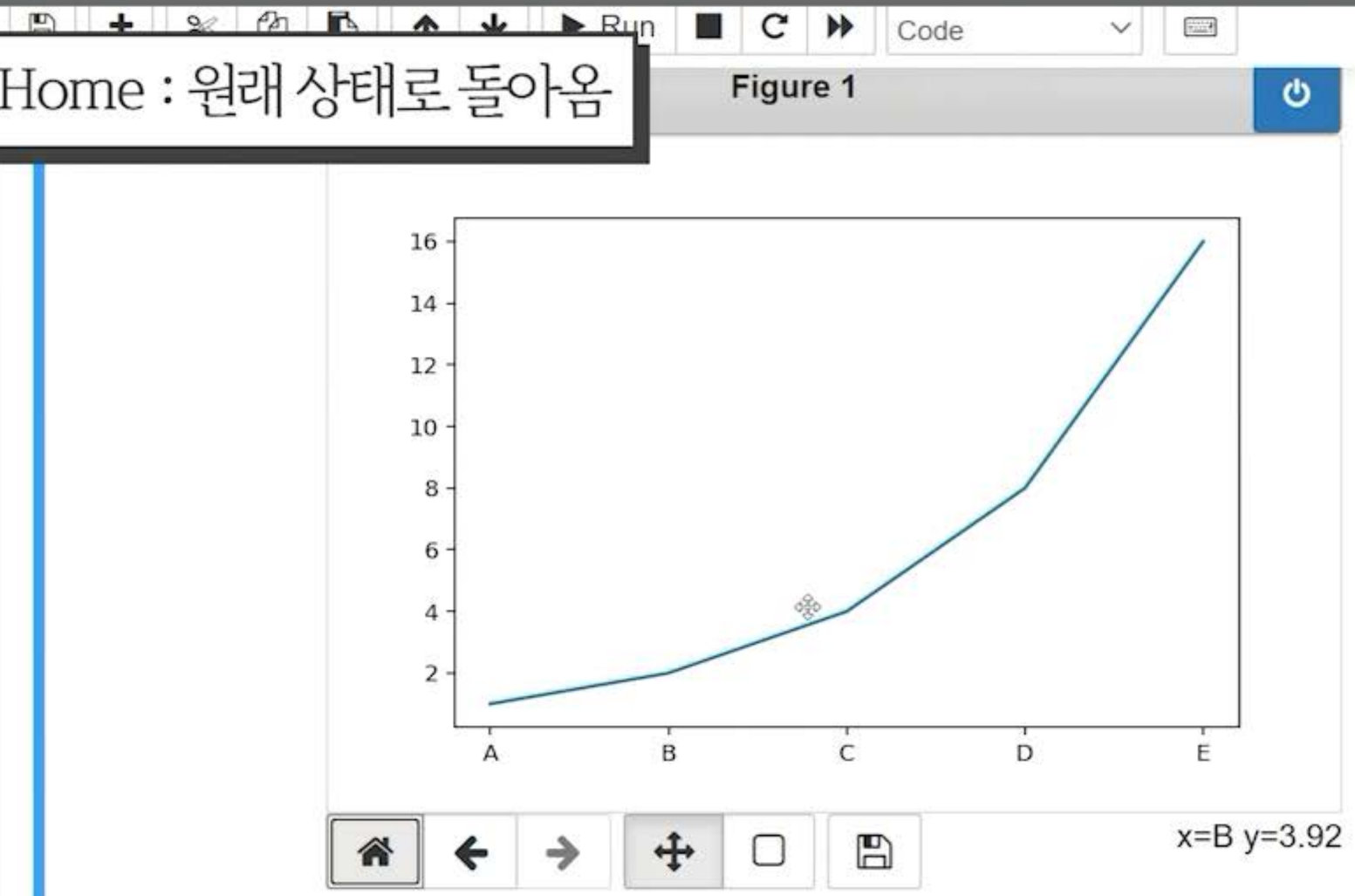
Out[45]: [<matplotlib.lines.Line2D at 0x2015b130910>]

In [ ]:

backend option

◎ nbagg

✔ Home : 원래 상태로 돌아옴



Out[45]: [<matplotlib.lines.Line2D at 0x2015b130910>]

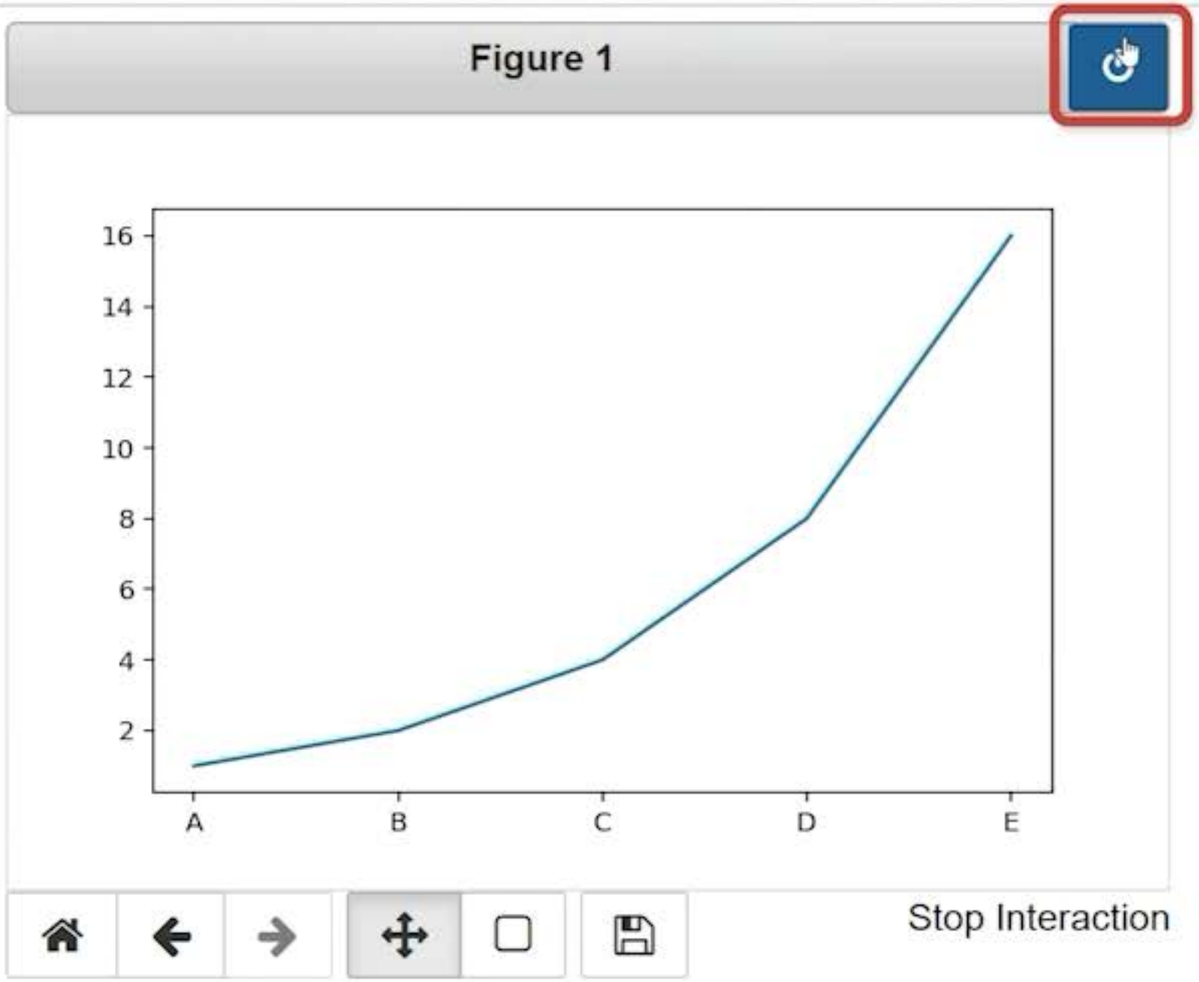
In [ ]:

backend option

◎ nbagg

Run Code

In [45]:
```
plt.plot(sr)
```



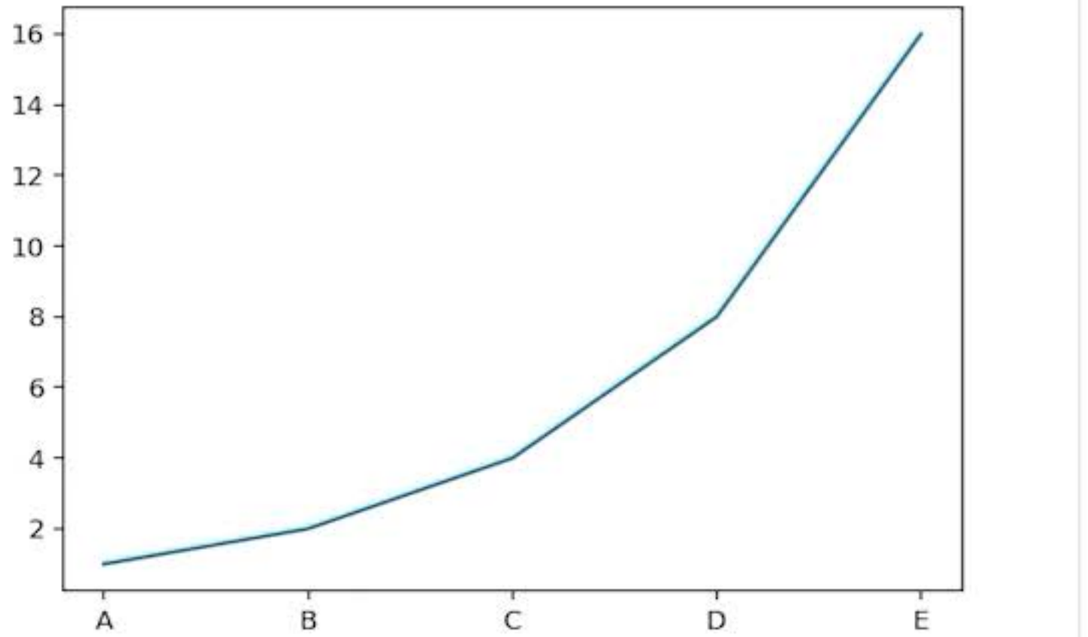
backend option

◎ nbagg

Run Code

In [45]:
```
#plt.plot(sr)
fig, ax = plt.subplots()
ax.plot(sr)
```

Figure 1

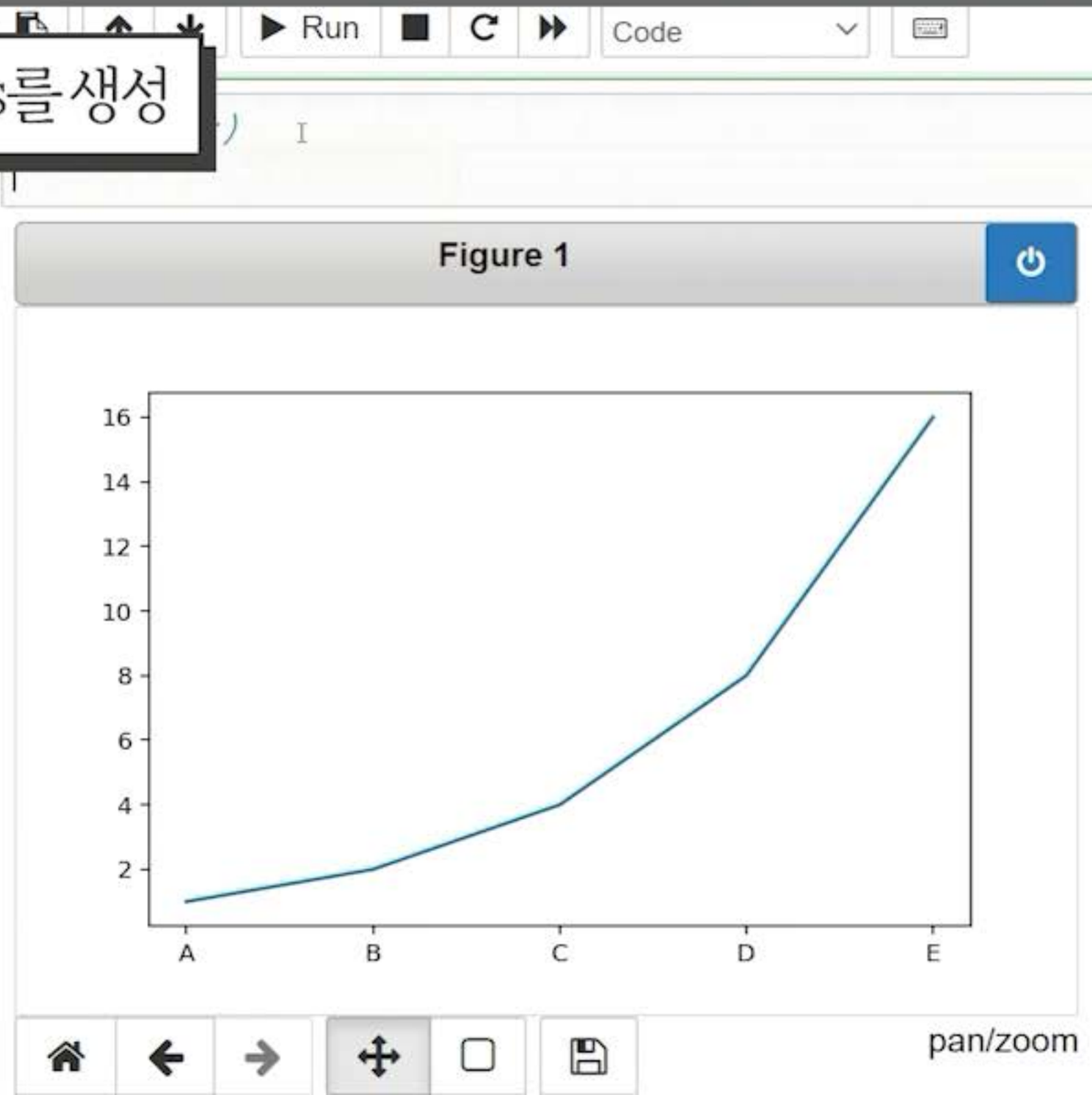


pan/zoom

backend option

◎ nbagg

✔ figure와 axes를 생성

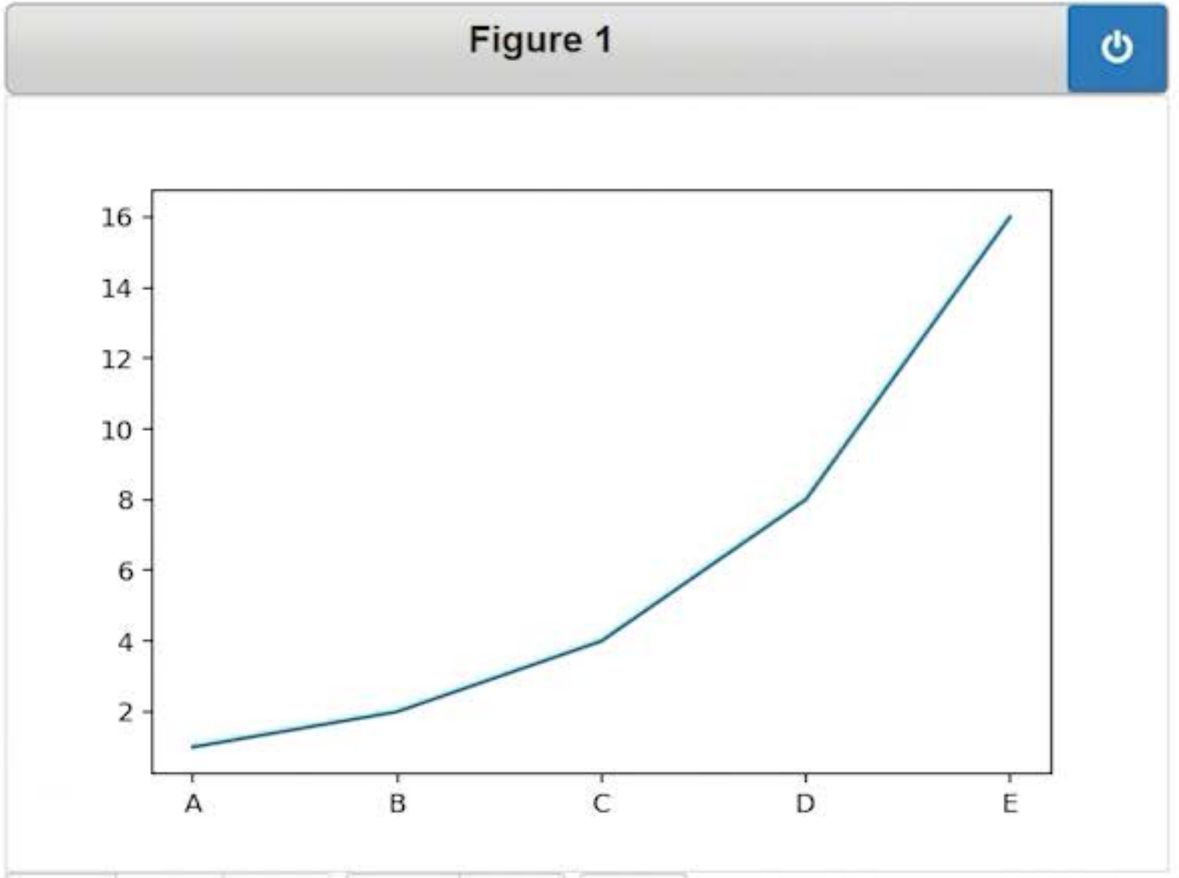


backend option

◎ nbagg

✔ 데이터들을 드래그하여 상세하게 볼 수 있음

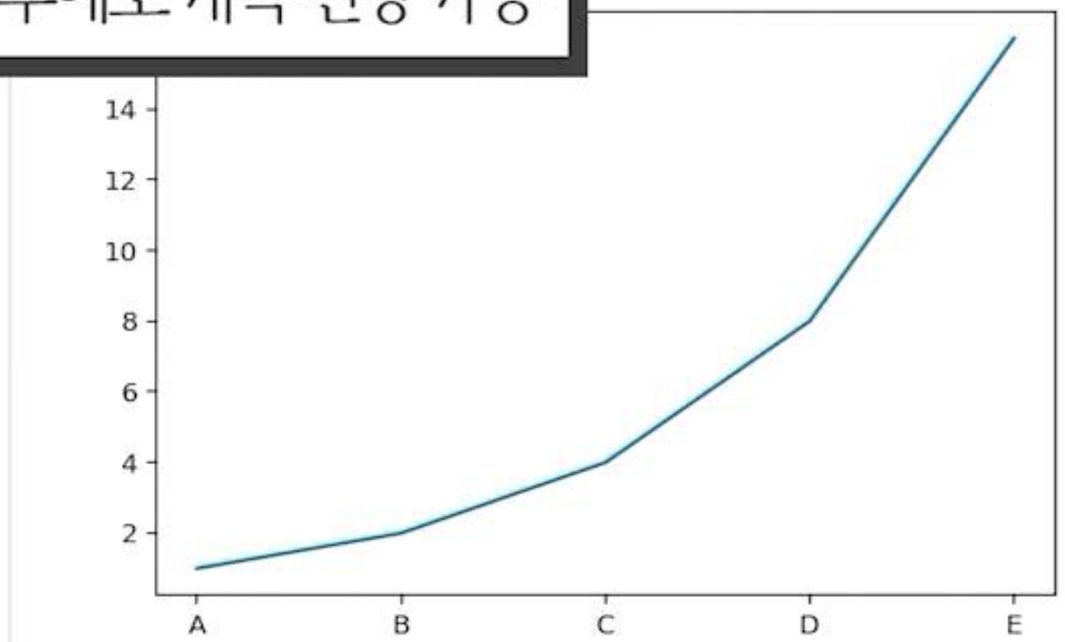
```
ax.plot(sr)
```



backend option

◎ nbagg

✔ 그래프를 그린 후에도 계속 변경 가능

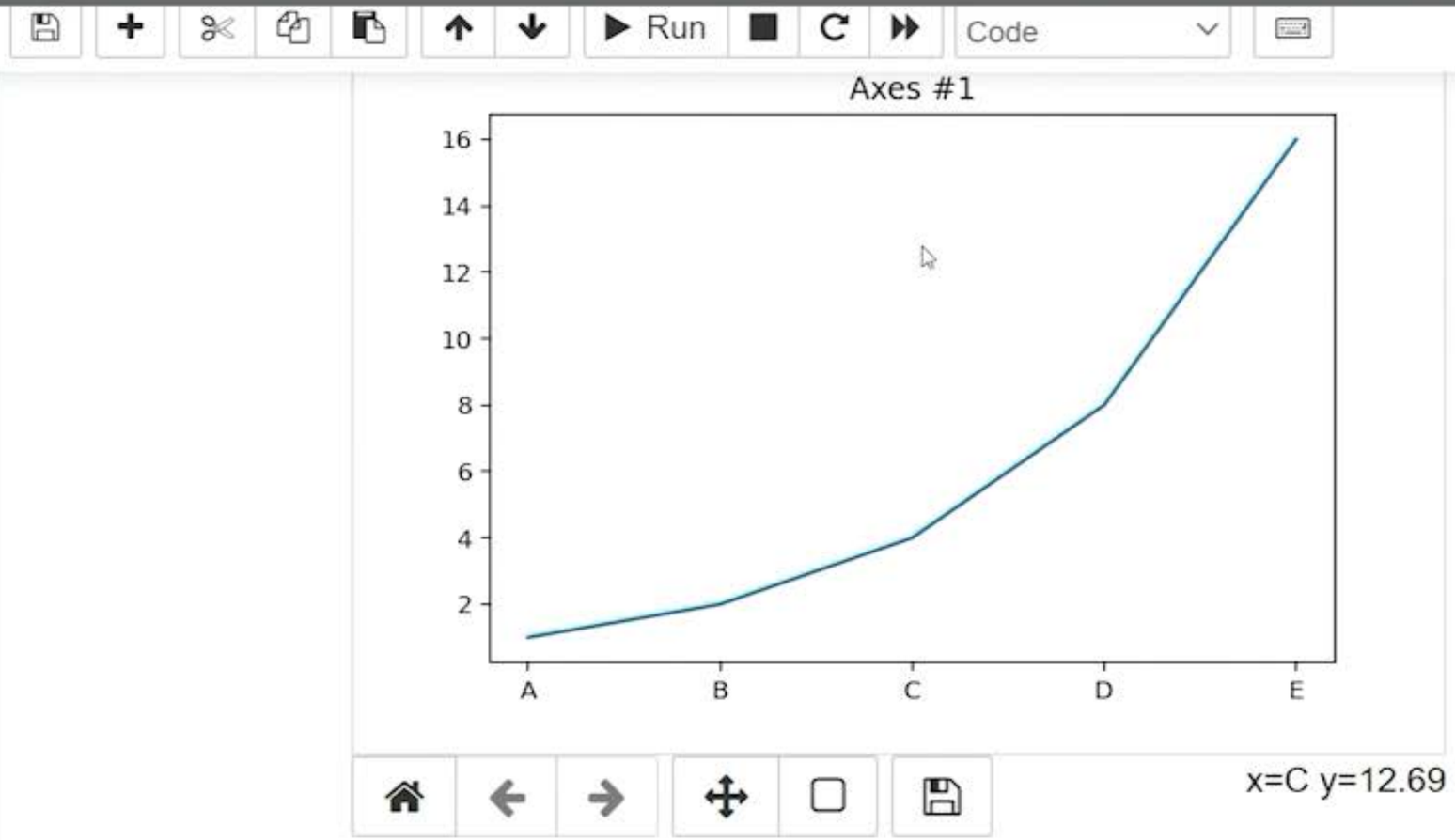


Out[46]: [<matplotlib.lines.Line2D at 0x2015b5c7b50>]

In [ ]:

backend option

◎ nbagg




```
Out[46]: [<matplotlib.lines.Line2D at 0x2015b5c7b50>]

In [47]: ax.set_title('Axes #1')
Out[47]: Text(0.5, 1.0, 'Axes #1')
```

backend option

◎ nbagg

Jupyter Notebook (Anaconda3)

```
otlib)와 황좌의 게임 데이터 분석.ipynb is not trusted
[I 12:09:02.364 NotebookApp] Saving file at /Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각
화(matplotlib)와 황좌의 게임 데이터 분석.ipynb
[W 12:09:02.368 NotebookApp] Notebook Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각화(matpl
otlib)와 황좌의 게임 데이터 분석.ipynb is not trusted
[I 12:11:02.354 NotebookApp] Saving file at /Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각
화(matplotlib)와 황좌의 게임 데이터 분석.ipynb
[W 12:11:02.359 NotebookApp] Notebook Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각화(matpl
otlib)와 황좌의 게임 데이터 분석.ipynb is not trusted
[I 12:13:02.351 NotebookApp] Saving file at /Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각
화(matplotlib)와 황좌의 게임 데이터 분석.ipynb
[W 12:13:02.353 NotebookApp] Notebook Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각화(matpl
otlib)와 황좌의 게임 데이터 분석.ipynb is not trusted
[I 12:15:02.354 NotebookApp] Saving file at /Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각
화(matplotlib)와 황좌의 게임 데이터 분석.ipynb
[W 12:15:02.354 NotebookApp] Notebook Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각화(matpl
otlib)와 황좌의 게임 데이터 분석.ipynb is not trusted
[I 12:17:02.317 NotebookApp] Saving file at /Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각
화(matplotlib)와 황좌의 게임 데이터 분석.ipynb
[W 12:17:02.319 NotebookApp] Notebook Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각화(matpl
otlib)와 황좌의 게임 데이터 분석.ipynb is not trusted
[I 12:19:02.365 NotebookApp] Saving file at /Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각
화(matplotlib)와 황좌의 게임 데이터 분석.ipynb
[W 12:19:02.367 NotebookApp] Notebook Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각화(matpl
otlib)와 황좌의 게임 데이터 분석.ipynb is not trusted
[I 12:21:02.346 NotebookApp] Saving file at /Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각
화(matplotlib)와 황좌의 게임 데이터 분석.ipynb
[W 12:21:02.350 NotebookApp] Notebook Documents/빅데이터실습/kmooc용 실습자료_2021/[W9,10] (촬영 전) 데이터 시각화(matpl
otlib)와 황좌의 게임 데이터 분석.ipynb is not trusted
```

E

Out[46]: [<matplotlib.lines.Line2D at 0x2015b5c7b50>]

In [47]:
```
ax.set_title('Axes #1')
```

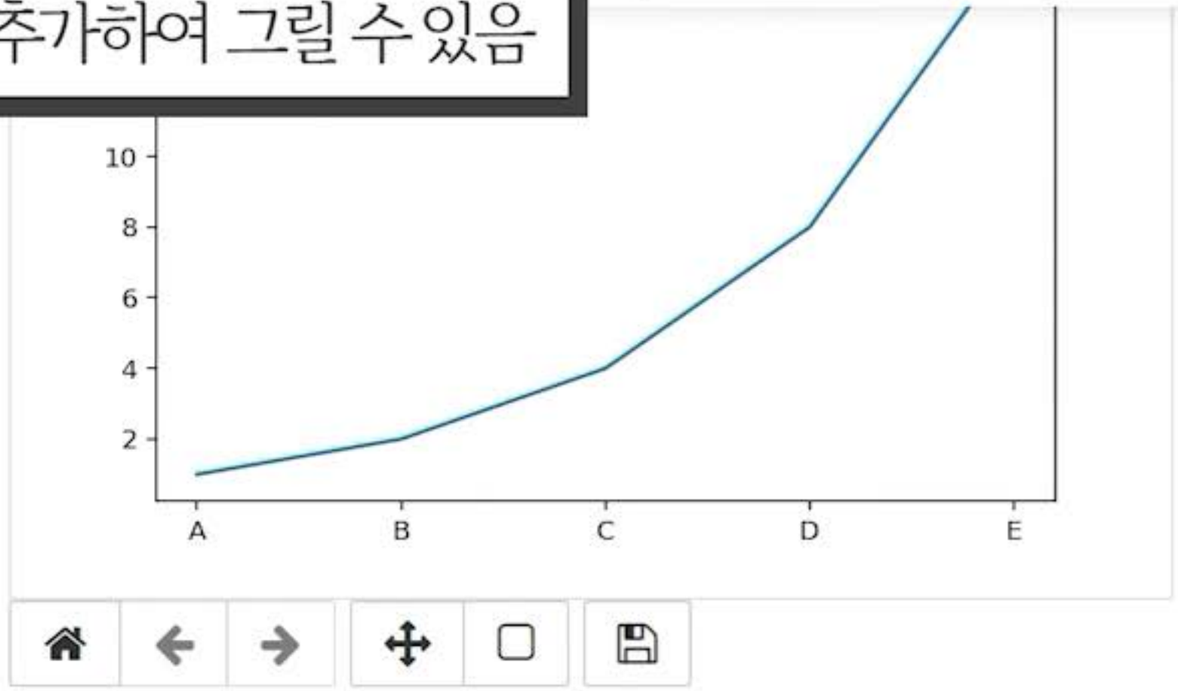
Out[47]: Text(0.5, 1.0, 'Axes #1')

In [ ]:

backend option

◎ nbagg

✔ bar graph를 추가하여 그릴 수 있음



Out[46]: [<matplotlib.lines.Line2D at 0x2015b5c7b50>]

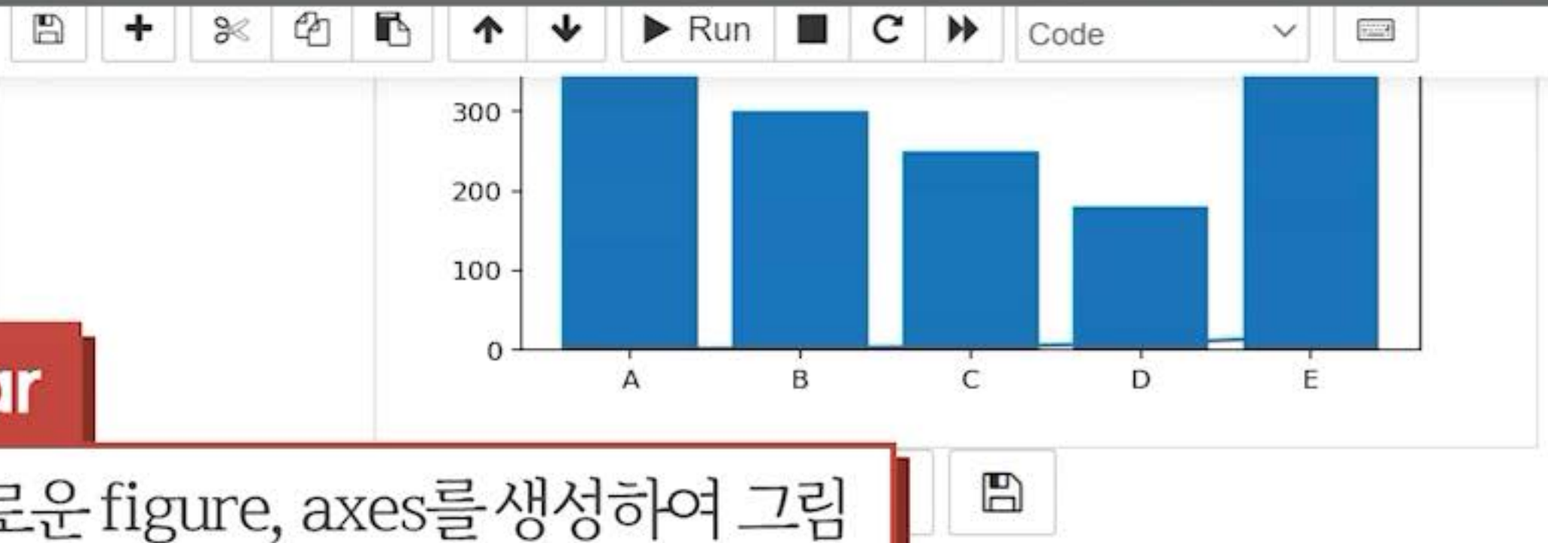
In [47]:
```
ax.set_title('Axes #1')
```

Out[47]: Text(0.5, 1.0, 'Axes #1')

In [ ]:
```
ax.b
```

backend option

◎ nbagg



bar

새로운 figure, axes를 생성하여 그림

Out[46]: [<matplotlib.lines.Line2D at 0x2015b5c7b50>]

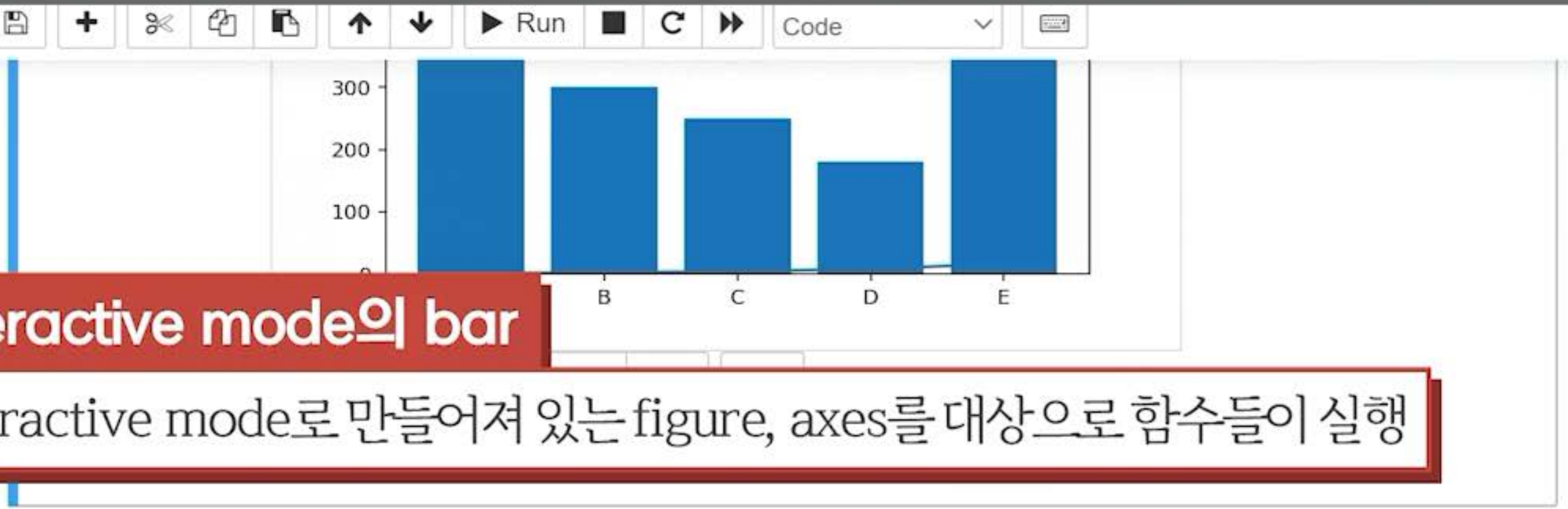
```
In [47]: ax.set_title('Axes #1')
Out[47]: Text(0.5, 1.0, 'Axes #1')
```

```
In [48]: ax.bar(sr2.index, sr2.values)
Out[48]: <BarContainer object of 5 artists>
```

1.4 matplotlib 설정 변경

backend option

◎ nbagg



interactive mode의 bar

interactive mode로 만들어져 있는 figure, axes를 대상으로 함수들이 실행

```
In [47]: ax.set_title('Axes #1')
Out[47]: Text(0.5, 1.0, 'Axes #1')

In [48]: ax.bar(sr2.index, sr2.values)
Out[48]: <BarContainer object of 5 artists>
```

1.4 matplotlib 설정 변경

02

Matplotlib 관련 설정 변경하기

matplotlib 설정 stylesheet 변경

◎ 장점

- ✔ 많은 기능을 안정적으로 제공
- ✔ 많은 레퍼런스(reference)
- ✔ 두터운 사용자층

1.4 matplotlib 설정 변경

1.4.1 stylesheet 변경

matplotlib을 통해 데이터 시각화할 때 적용되는 스타일을 변경 가능하며, 사용가능한 스타일시트 목록 및 샘플은 여기 에서 확인 가능하다.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

matplotlib 설정 stylesheet 변경

◎ 단점

✔ 세련되지 않은 그래프

Run Code

```
In [47]: e('Axes #1')
Out[47]: Text(0.5, 1.0, 'Axes #1')

In [48]: ax.bar(sr2.index, sr2.values)
Out[48]: <BarContainer object of 5 artists>
```

1.4 matplotlib 설정 변경

1.4.1 stylesheet 변경

matplotlib을 통해 데이터 시각화할 때 적용되는 스타일을 변경 가능하며, 사용가능한 스타일시트 목록 및 샘플은 여기 에서 확인 가능하다.

```
In [ ]:
```

seaborn이나 bokeh, plotly, ggplot 같은 다양한 최신 라이브러리들은
조금 더 세련된 그래프를 그릴 수 있습니다.

matplotlib 설정 stylesheet 변경

◎ 사용가능한 스타일 목록

Run Code

In [47]:
```
ax.set_title('Axes #1')
```

Out[47]:
```
Text(0.5, 1.0, 'Axes #1')
```

In [48]:
```
ax.bar(sr2.index, sr2.values)
```

Out[48]:
```
<BarContainer object of 5 artists>
```

1.4 matplotlib 설정 변경

1.4.1 stylesheet 변경

matplotlib을 통해 데이터 시각화할 때 적용되는 스타일을 변경 가능하며, 사용가능한 스타일시트 목록 및 샘플은 여기 에서 확인 가능하다.

In [49]:
```
# 사용가능한 스타일 목록
plt.style.available
```

Out[49]:
```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
```

matplotlib 설정 stylesheet 변경

◎ 사용가능한 스타일 목록

✔ default, bmh, classic, dack_background, ggplot 등을 많이 사용

```
'_classic_test_patch',
'bmh',
'classic',
'dark_background',
'fast',
'fivethirtyeight',
'ggplot',
'grayscale',
'seaborn',
'seaborn-bright',
'seaborn-colorblind',
'seaborn-dark',
'seaborn-dark-palette',
'seaborn-darkgrid',
'seaborn-deep',
'seaborn-muted',
'seaborn-notebook',
'seaborn-paper',
'seaborn-pastel',
'seaborn-poster',
'seaborn-talk',
'seaborn-ticks',
```

matplotlib 설정 stylesheet 변경

◎ 스타일을 변경하는 방법

✔ plt.style.use('classic') : classic 스타일을 선택

```
'seaborn-dark',
'seaborn-dark-palette',
'seaborn-darkgrid',
'seaborn-deep',
'seaborn-muted',
'seaborn-notebook',
'seaborn-paper',
'seaborn-pastel',
'seaborn-poster',
'seaborn-talk',
'seaborn-ticks',
'seaborn-white',
'seaborn-whitegrid',
'tableau-colorblind10']
```

In [50]:
```
%matplotlib inline
```

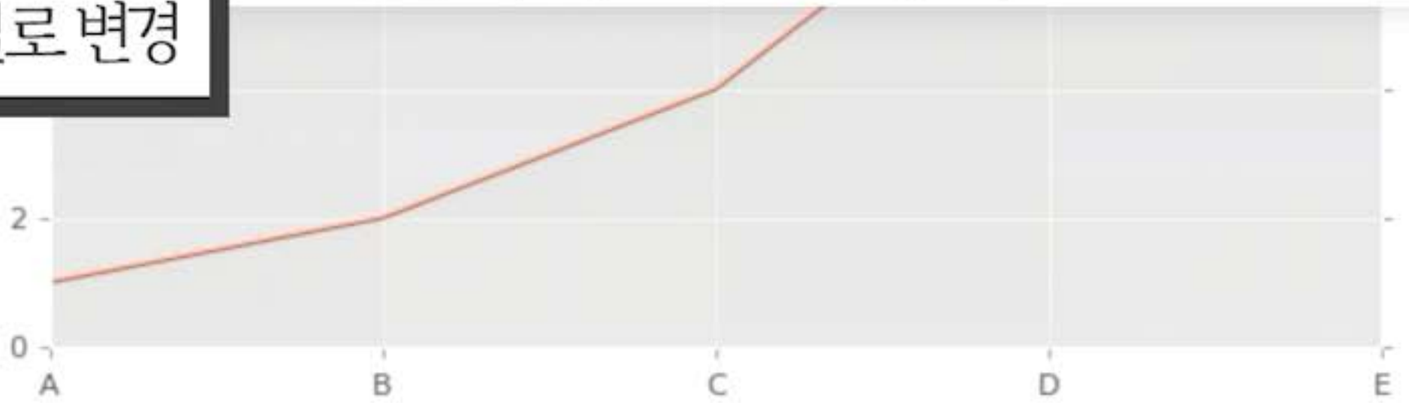
In [ ]:
```
plt.style.use('classic')
plt.plot(sr)
```

In [ ]:

matplotlib 설정 stylesheet 변경

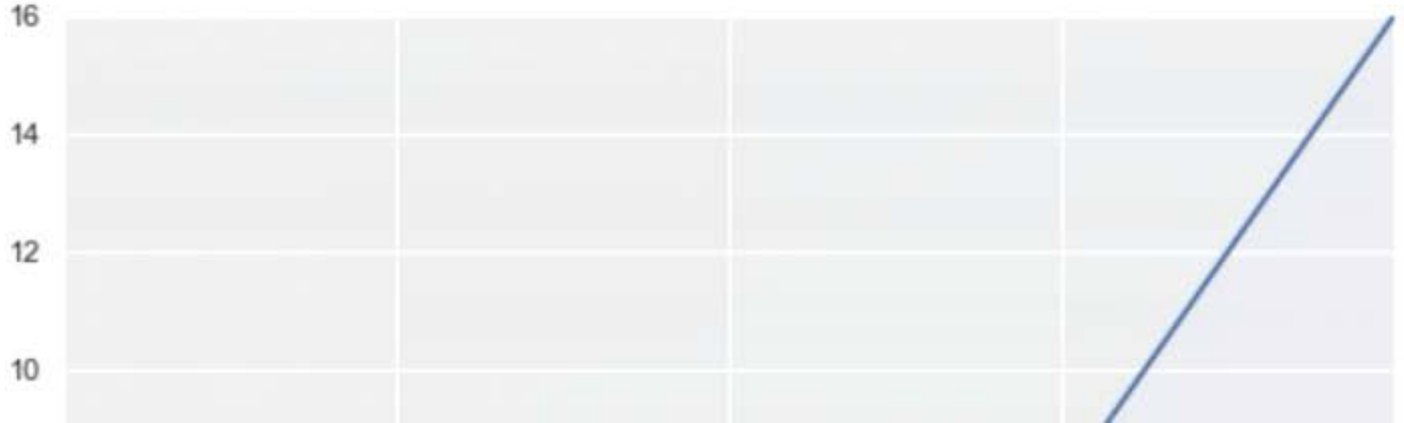
◎ 스타일을 변경하는 방법

✔ seaborn 스타일로 변경



In [53]:
```
plt.style.use('seaborn')
plt.plot(sr)
```

Out[53]: [<matplotlib.lines.Line2D at 0x20159616b50>]



matplotlib 설정 stylesheet 변경

◎ 스타일을 변경하는 방법

✔ seaborn-dark-palette 스타일로 변경

Run Code

C D E

In [54]:
```
plt.style.use('seaborn-dark-palette')
plt.plot(sr)
```

Out[54]: [<matplotlib.lines.Line2D at 0x201595d4940>]

matplotlib 설정 stylesheet 변경

◎ 스타일을 변경하는 방법

✔ dark_background 스타일로 변경

```
alette')
plt.plot(sr)
```

...

In [55]:
```
plt.style.use('dark-background')
plt.plot(sr)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
C:WProgramDataWAnaconda3WlibWsite-packagesWmatplotlibWstyleWcore.py in use(style)
    120             try:
--> 121                 rc = rc_params_from_file(style, use_default_template=False)
    122                 _apply_style(rc)

C:WProgramDataWAnaconda3WlibWsite-packagesWmatplotlibW__init__.py in rc_params_from
_file(fname, fail_on_error, use_default_template)
    873     """
--> 874     config_from_file = _rc_params_in_file(fname, fail_on_error=fail_on_error)
    875

C:WProgramDataWAnaconda3WlibWsite-packagesWmatplotlibW__init__.py in _rc_params_in_
file(fname, transform, fail_on_error)
    802     rc_temp = {}
```

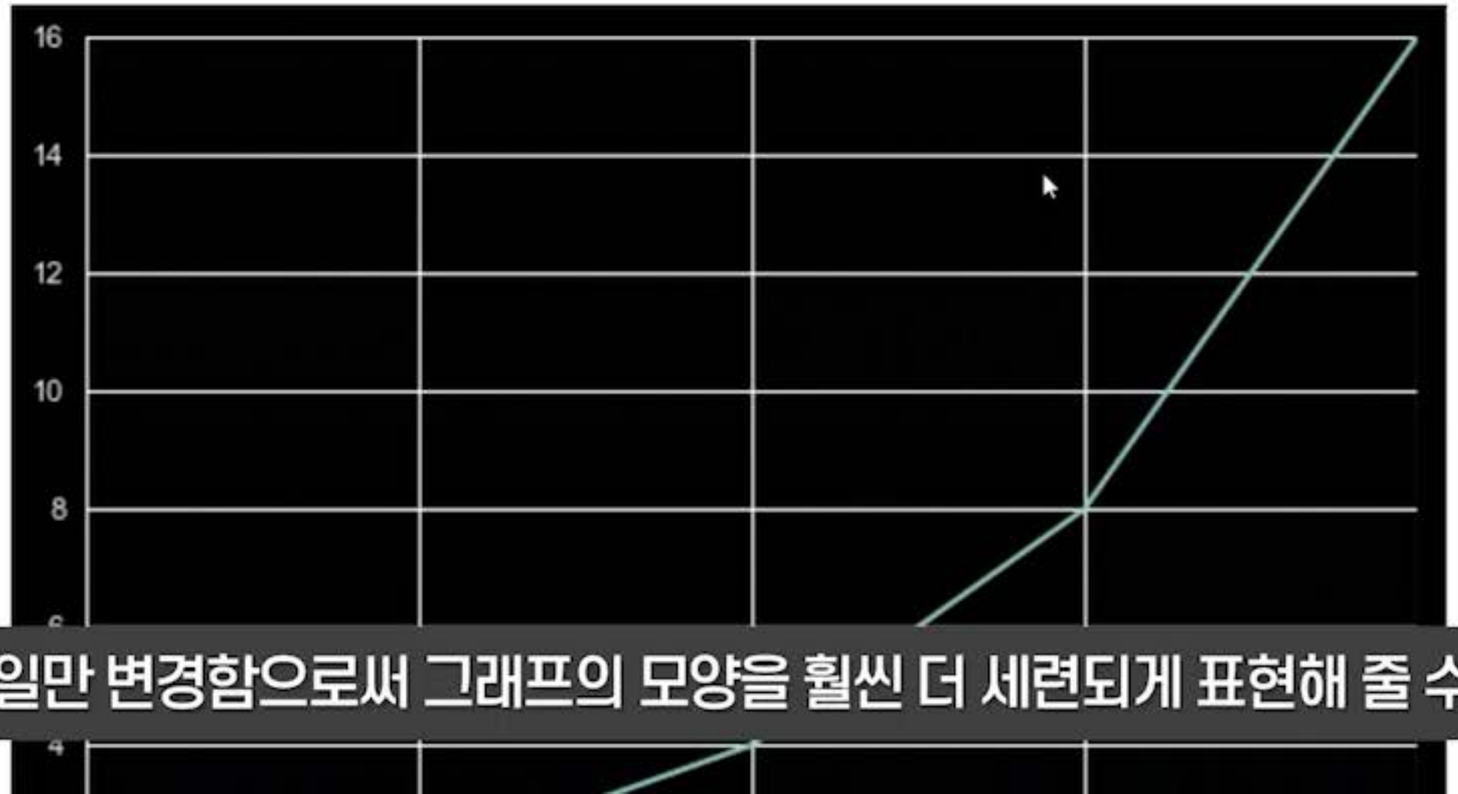
matplotlib 설정 stylesheet 변경

◎ 스타일을 변경하는 방법

Run Code

```
In [56]: plt.style.use('dark_background')
         plt.plot(sr)
```

```
Out[56]: [<matplotlib.lines.Line2D at 0x2015c1ec850>]
```

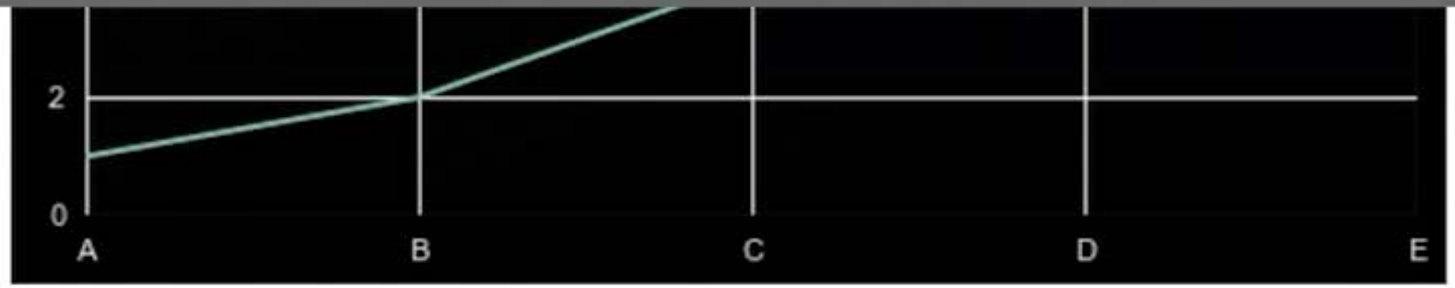


간단하게 스타일만 변경함으로써 그래프의 모양을 훨씬 더 세련되게 표현해 줄 수 있습니다.

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ figure 크기, axes 폰트사이즈, 색상, 스타일 등 속성 설정 가능

◎ 파라미터 수정을 통해 나만의 커스터마이징된 환경 사용 가능



In [ ]:

1.4.2 ***matplotlibrc*** 파일 수정을 통한 파라미터 수정

- matplotlibrc 파일을 통해 figure 크기, axes의 폰트사이즈, 색상, 스타일 등 matplotlib의 모든 속성(property)들을 설정 가능하다.
- 파일을 변경한 후 저장하면, 이후에는 변경된 설정이 계속 사용된다.
- matplotlibrc 파일을 수정한 후에는 jupyter notebook을 재시작해야 수정 내용이 반영된다.

matplotlibrc 파일의 위치를 찾으려면 matplotlib.matplot.lib_fname()를 수행하면 된다.

In [ ]:

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 영구적으로 저장하기

✔ jupyter notebook을 실행하면 matplotlib과 관련된 파라미터들은 matplotlibrc 파일의 내용으로 설정됨

In [ ]:

1.4.2 ***matplotlibrc*** 파일 수정을 통한 파라미터 수정

- matplotlibrc 파일을 통해 figure 크기, axes의 폰트사이즈, 색상, 스타일 등 matplotlib의 모든 속성(property)들을 설정 가능하다.
- 파일을 변경한 후 저장하면, 이후에는 변경된 설정이 계속 사용된다.
- matplotlibrc 파일을 수정한 후에는 jupyter notebook을 재시작해야 수정 내용이 반영된다.

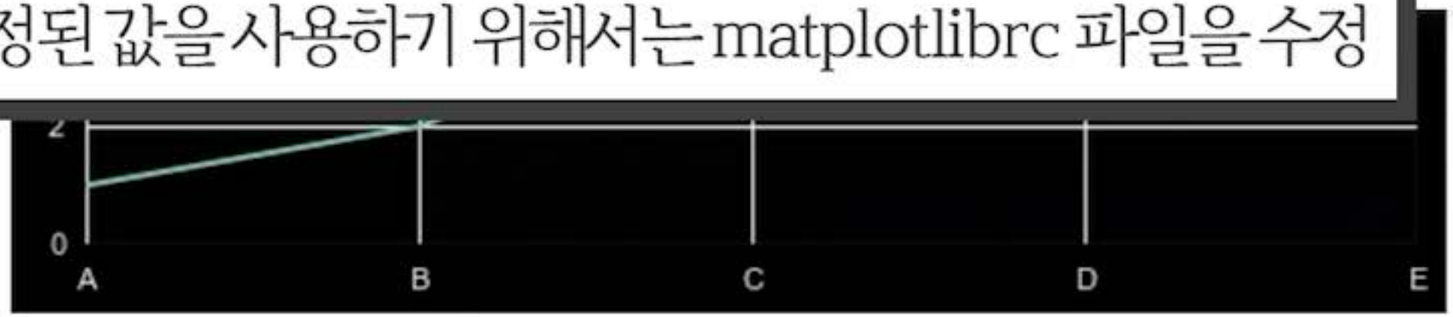
matplotlibrc 파일의 위치를 찾으려면 matplotlib.matplot.lib_fname()를 수행하면 된다.

In [ ]:

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 영구적으로 저장하기

✔ 영구적으로 설정된 값을 사용하기 위해서는 matplotlibrc 파일을 수정



In [ ]:

1.4.2 *matplotlibrc* 파일 수정을 통한 파라미터 수정

- matplotlibrc 파일을 통해 figure 크기, axes의 폰트사이즈, 색상, 스타일 등 matplotlib의 모든 속성(property)들을 설정 가능하다.
- 파일을 변경한 후 저장하면, 이후에는 변경된 설정이 계속 사용된다.
- matplotlibrc 파일을 수정한 후에는 jupyter notebook을 재시작해야 수정 내용이 반영된다.

matplotlibrc 파일의 위치를 찾으려면 matplotlib.matplot.lib_fname()를 수행하면 된다.

In [ ]:

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 영구적으로 저장하기

- ✔ matplotlibrc 파일에는 matplotlib의 모든 속성을 설정할 수 있음
- ✔ 이 파일을 저장하면 컴퓨터를 껐다 켜도 변경된 설정을 계속 사용 가능

주의하기

- ✔ jupyter server를 재시작해야 수정된 내용이 반영됨

In [ ]:

조, 색상, 스타일 등 matplotlib의 모든 속성(property)들을 설정 가능하다.

- 파일을 변경한 후 저장하면, 이후에는 변경된 설정이 계속 사용된다.
- matplotlibrc 파일을 수정한 후에는 jupyter notebook을 재시작해야 수정 내용이 반영된다.

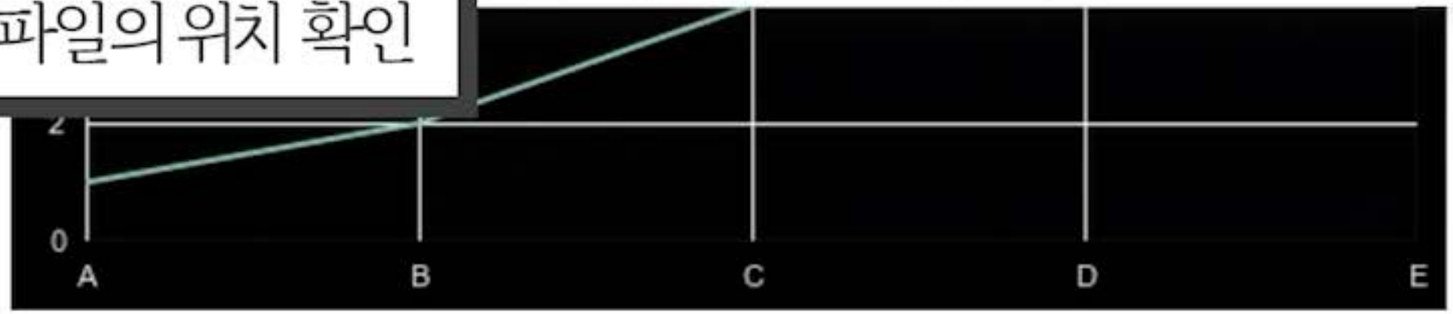
matplotlibrc 파일의 위치를 찾으려면 matplotlib.matplot.lib_fname()를 수행하면 된다.

In [ ]:

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 영구적으로 저장하기

✔ matplotlibrc 파일의 위치 확인



In [ ]:

1.4.2 ***matplotlibrc*** 파일 수정을 통한 파라미터 수정

- matplotlibrc 파일을 통해 figure 크기, axes의 폰트사이즈, 색상, 스타일 등 matplotlib의 모든 속성(property)들을 설정 가능하다.
- 파일을 변경한 후 저장하면, 이후에는 변경된 설정이 계속 사용된다.
- matplotlibrc 파일을 수정한 후에는 jupyter notebook을 재시작해야 수정 내용이 반영된다.

matplotlibrc 파일의 위치를 찾으려면 matplotlib.matplot.lib_fname()를 수행하면 된다.

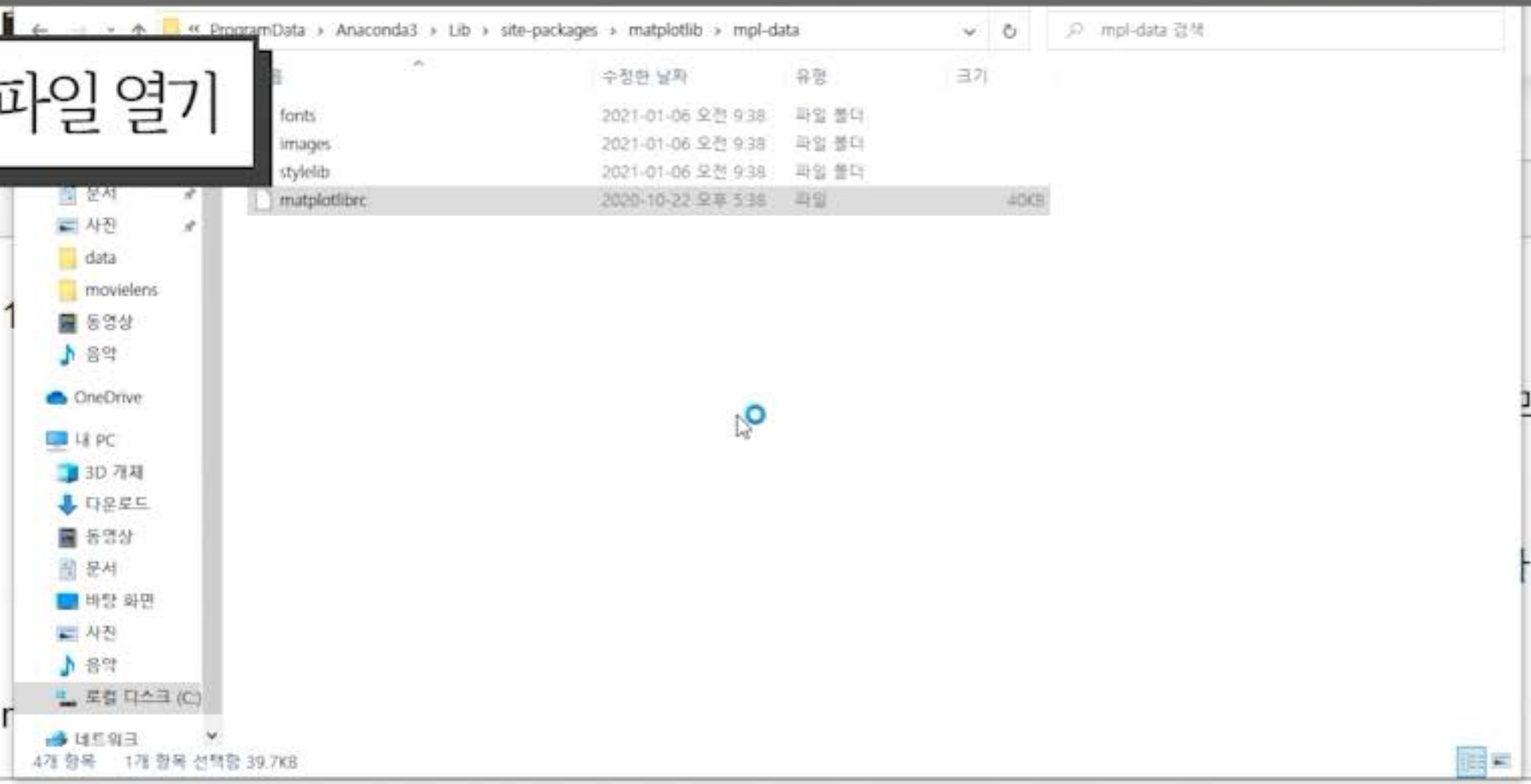
matplotlib_fname
matplotlib

In [ ]: matplotlib.matplot

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 영구적으로 저장하기

✔ matplotlibrc 파일 열기




```
In [57]: matplotlib.matplotlib_fname()
Out[57]: 'C:WWProgramDataWWAnaconda3WWlibWWsite-packagesWWmatplotlibWWmpl-dataWWmatplotlibrc'
```

1.4.3 matplotlib.rcParams을 통한 동적 설정 변경

- matplotlib의 설정 정보는 **matplotlib.rcParams**에 저장되어 있으며, rcParams는 ***사전(Dict)*** 타입니다.

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 영구적으로 저장하기

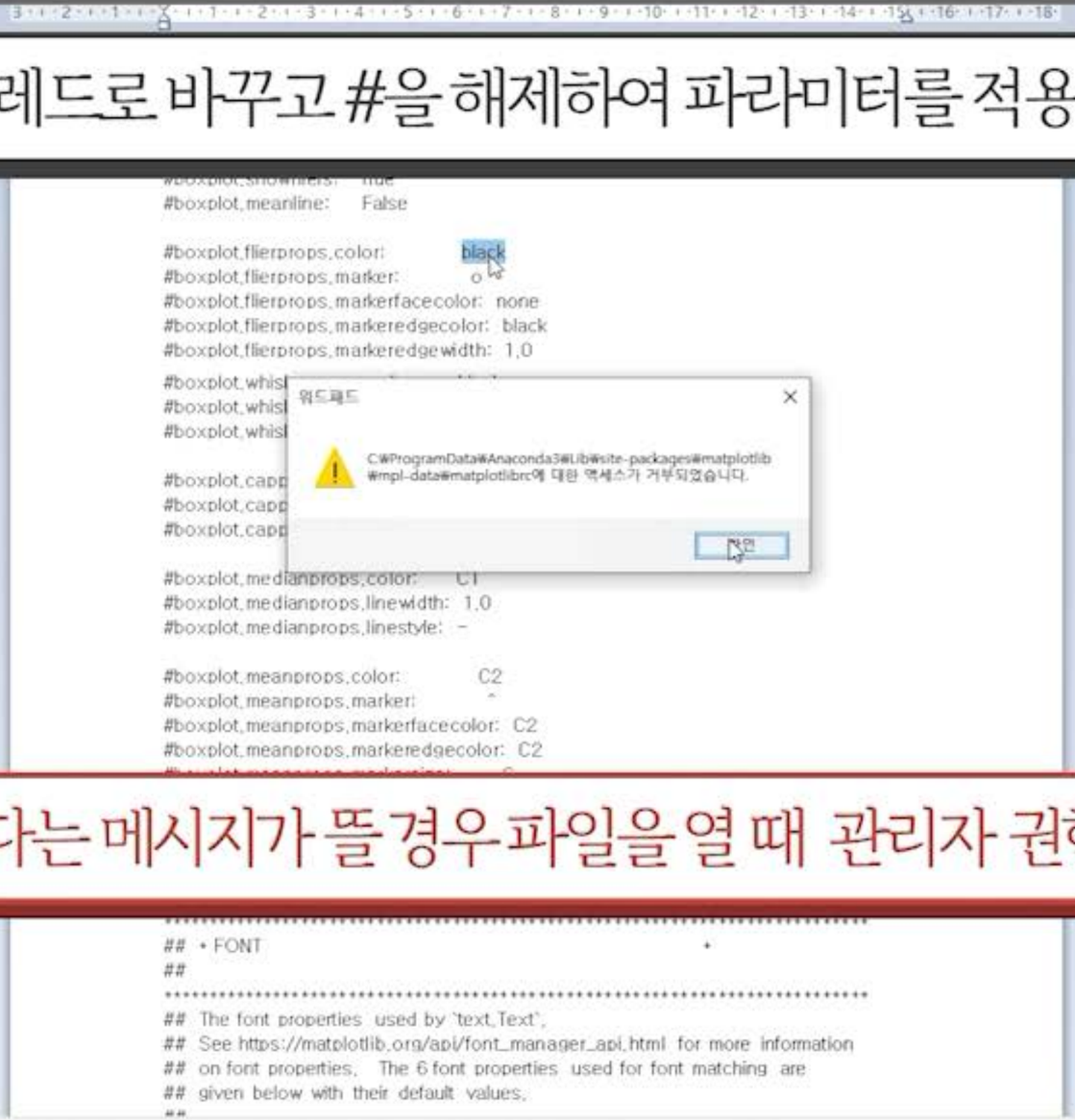
- ✔ matplotlibrc 파일 열기
- ✔ 변경하고 싶은 내용은 #부분을 해제하거나 수정 가능

```
## The three standard dash patterns.  These are scaled by the linewidth.
#lines.dashed_pattern: 3.7, 1.6
#lines.dashdot_pattern: 6.4, 1.6, 1, 1.6
#lines.dotted_pattern: 1, 1.65
#lines.scale_dashes: True

## * PATCHES                                                          *
##
## Patches are graphical objects that fill 2D space, like polygons or circles.
## See https://matplotlib.org/api/artist_api.html#module-matplotlib.patches
## for more information on patch properties.
#patch.linewidth:        1     # edge width in points.
#patch.facecolor:        C0
#patch.edgecolor:        black  # if forced, or patch is not filled
#patch.force_edgecolor:  False  # True to always use edgecolor
#patch.antialiased:      True   # render patches in antialiased (no jaggies)


##
## * HATCHES                                                          *
##
#hatch.color:     black
#hatch.linewidth: 1.0


##
## * BOXPLOT                                                          *
##
#boxplot.notch:       False
#boxplot.vertical:    True
#boxplot.whiskers:    1.5
#boxplot.bootstrap:   None
```

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 영구적으로 저장하기

✔ 컬러값을 블랙에서 레드로 바꾸고 #을 해제하여 파라미터를 적용



주의하기

✔ 액세스가 거부되었다는 메시지가 뜰 경우 파일을 열 때 관리자 권한으로 실행하면 됨

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 동적으로 바로 적용하기

✔ 설정된 값들은 matplotlib.rcPrams에 저장됨

Out[57]: 'C:WWProgramDataWWAnaconda3WWlibWWsite-packagesWWmatplotlibWWmpl-dataWWmatplotlibrc'

1.4.3 matplotlib.rcParams을 통한 동적 설정 변경

- matplotlib의 설정 정보는 **matplotlib.rcParams**에 저장되어 있으며, rcParams는 ***사전(Dict)*** 타입니다.
- rcParams 변경을 통해 동적으로 설정이 변경가능하며, 변경 즉시 반영된다.

In [58]:
```
matplotlib.rcParams
```

Out[58]:
```
RcParams({'_internal.classic_mode': True,
          'agg.path.chunksize': 0,
          'animation.avconv_args': [],
          'animation.avconv_path': 'avconv',
          'animation.bitrate': -1,
          'animation.codec': 'mpeg4',
          'animation.convert_args': [],
          'animation.convert_path': 'convert',
          'animation.embed_limit': 20.0,
          'animation.ffmpeg_args': [],
```

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 동적으로 바로 적용하기

- ✔ 파라미터의 이름이 key 값이 되고, 기본값이 value로 저장됨
- ✔ matplotlib에 있는 모든 파라미터들에 대한 정보가 저장되어 있음

반영된다.

```
            'axes.prop_cycle': cycler('color', ['#8dd3c7', '#feffb3', '#bfbbd9', '#fa8
    174', '#81b1d2', '#fdb462', '#b3de69', '#bc82bd', '#ccebc4', '#ffed6f']),
            'axes.spines.bottom': True,
            'axes.spines.left': True,
            'axes.spines.right': True,
            'axes.spines.top': True,
            'axes.titlecolor': 'auto',
            'axes.titlelocation': 'center',
            'axes.titlepad': 5.0,
            'axes.titlesize': 12.0,
            'axes.titleweight': 'normal',
            'axes.titley': 1.0,
            'axes.unicode_minus': True,
            'axes.xmargin': 0.0,
            'axes.ymargin': 0.0,
            'axes3d.grid': True,
            'backend': 'module://ipykernel.pylab.backend_inline',
            'backend_fallback': True,
```

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 동적으로 바로 적용하기

타입니다.

- rcParams 변경을 통해 동적으로 설정이 변경가능하며, 변경 즉시 반영된다.

```
In [59]: matplotlib.rcParams['font.family']
Out[59]: ['sans-serif']
```

```
In [ ]: matplotlib.rcParams['font.family'] = 'Malgun Gothic'
```

```
In [ ]:
```

```
In [ ]:
```

2. Pandas의 plot() 함수를 통한 시각화

- pandas의 대표적인 데이터타입인 Series와 DataFrame은 plot() 함수를 제공하며 이를 통해 여러 차트를 그릴 수 있다.
- 내부적으로는 Series.plot()과 DataFrame.plot() 모두 matplotlib를 사용하며, 파라미터 인자에 따라 적절한 함수가 호출된다.
- matplotlib은 plot()은 line graph, bar()는 bar graph, hist()는 히스토그램 등 차트의 종류에 따

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 동적으로 바로 적용하기

✔ stylesheet 변경은 관련 파라미터들이 일괄적으로 변경된 것임

In [53]:
```
plt.style.use('seaborn')
plt.plot(sr)
```

Out[53]: [<matplotlib.lines.Line2D at 0x20159616b50>]

matplotlib 설정 matplotlibrc 파일 수정을 통한 파라미터 수정

◎ 동적으로 바로 적용하기

- ✔ 한글로 사용된 데이터를 시각화하면 한글이 깨지는 경우 발생
- ✔ 윈도우의 경우에는 Malgun Gothic으로 변경
- ✔ ios인 경우에는 AppleGothic으로 설정

rcParams는 *사전(Dict)*

가능하며, 변경 즉시 반영된다.

```
In [60]: matplotlib.rcParams['font.family'] = 'Malgun Gothic'
         # ios인 경우에는 AppleGothic으로 변경

In [61]: matplotlib.rcParams['font.family']
Out[61]: ['Malgun Gothic']

In [62]: plt.style.use('ggplot')
         plt.plot(sr)
Out[62]: [<matplotlib.lines.Line2D at 0x2015bfafbe0>]
```

학습완료

1/ backend의 개념 및 종류

2/ Matplotlib 관련 설정 변경하기