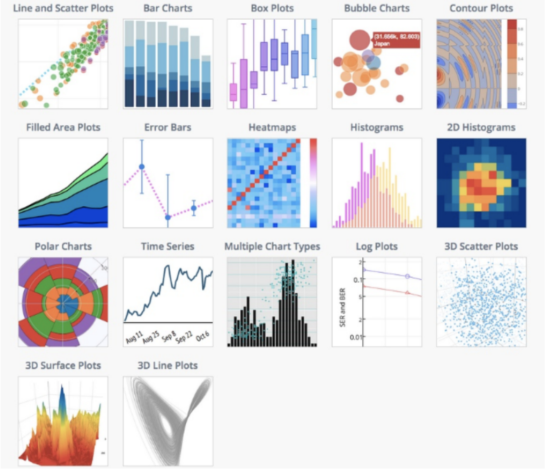
matplotlib
Version 3.1.1



[데이터 시각화 : matplotlib]

matplotlib 그래프 그리기

- 파이썬에서 자료를 차트(chart)나 플롯(plot)으로 시각화(visulaization)하는 패키지이다.
- MATLAB(과학 및 공학 연산을 위한 소프트웨어)의 시각화 기능을 모델링해서 만들어짐
- 몇 줄의 코드로 간단하게 그래프 그리기 가능
- 2차원 선 그래프(plot), 산점도(scatter plot), 막대 그래프(bar chart), 히스토그램(histogram), 파이 그래프(pie chart)등
- 아나콘다 배포판에 포함돼 있음
- matplotlib 홈페이지:http://matplotlib.org/

https://matplotlib.org/gallery.html

- matplotlib 그래프 종류

- 라인 플롯(line plot)
- 스캐터 플롯(scatter plot)
- 컨투어 플롯(contour plot)
- 서피스 플롯(surface plot)
- 바 차트(bar chart)
- 히스토그램(histogram)
- 박스 플롯(box plot)

Simple Plot

- Chart 생성과 출력
- matplotlib의 pyplot 모듈을 이용하여 그래프를 생성

In [ ]:
```
# 그래프 객체의 정보 없이 그래프만 출력 : plt.show()
import matplotlib.pyplot as plt

plt.plot([1, 4, 5, 10])
plt.show()
```

In [ ]:
```
import matplotlib.pyplot as plt
data1 = [10, 14, 19, 20, 25]
plt.plot(data1)
plt.show()
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
# 데이터 준비
X = np.arange(0, 6, 0.1) # 0에서 6까지 0.1 간격으로 생성

Y1 = np.sin(X)
Y2 = np.cos(X)

# 그래프 그리기
plt.plot(X, Y1)
plt.plot(X, Y2)

# 화면에 출력
plt.show()

print(X)
```

1. 선 그래프 (line plot)

- 가장 간단한 플롯은 선을 그리는 라인 플롯(line plot)이다.
- 라인 플롯은 데이터가 시간, 순서 등에 따라 어떻게 변화하는지 보여주기 위해 사용한다.
- plot([x], y, [fmt], data=None, **kwargs)

```
import matplotlib.pyplot as plt
%matplotlib qt # 별도의 팝업창에 그래프 출력
%matplotlib inline # 다시 코드결과 출력 부분에 출력

- 기본적인 그래프 형식
  plt.plot(y)
  plt.plot(y, fmt)
  plt.plot(x, y)
  plt.plot(x, y, fmt)
  fmt = '[color][line_style][marker]'

- 그래프 옵션
  plt.figure() # 그래프창 지정
  plt.subplot(m,n,p) # 멀티그래프
```

속성	설명
color or c	Plot 되는 선의 색상 (blue, green, red, cyan, magenta, yellow, black, white)
label	Plot 되는 데이터의 Label. 범례 표시를 위해 legend() 함수에서 활용 됨.
linestyle or ls	Plot 되는 선의 스타일 (solid : - dashed : -- dashdot : -. dotted : :)
linewidth or lw	Plot 되는 선의 굵기 (float value)

- 기본적인 선 그래프 그리기

In [ ]:
```
data1 = [10, 14, 19, 20, 25]
```

In [ ]:
```
plt.plot(data1)
```

In [ ]:
```
plt.plot(data1)
plt.show()
```

In [ ]:
```
%matplotlib qt
```

In [ ]:
```
plt.plot(data1)
```

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

In [ ]:
```
import numpy as np
x = np.arange(-4.5, 5, 0.5) # 배열 x 생성. 범위: [-4.5, 5), 0.5씩 증가
y = 2*x**2 # 수식을 이용해 배열 x에 대응하는 배열 y 생성
[x,y]
```

In [ ]:
```
plt.plot(x,y)
plt.show()
```

- 여러 그래프 그리기

In [ ]:
```
import numpy as np

x = np.arange(-4.5, 5, 0.5)
y1 = 2*x**2
y2 = 5*x + 30
y3 = 4*x**2 + 10
```

In [ ]:
```
plt.plot(x, y1, x, y2, x, y3)
plt.show()
```

In [ ]:
```
plt.plot(x, y1) # 처음 그리기 함수를 수행하면 그래프 창이 자동으로 생성됨

plt.figure() # 새로운 그래프 창을 생성함
plt.plot(x, y2) # 새롭게 생성된 그래프 창에 그래프를 그림

plt.show()
```

In [ ]:
```
import numpy as np

# 데이터 생성
x = np.arange(-5, 5, 0.1)
y1 = x**2 -2
y2 = 20*np.cos(x)**2 # NumPy에서 cos()는 np.cos()으로 입력

plt.figure(1) # 1번 그래프 창을 생성함
plt.plot(x, y1) # 지정된 그래프 창에 그래프를 그림

plt.figure(2) # 2번 그래프 창을 생성함
plt.plot(x, y2) # 지정된 그래프 창에 그래프를 그림

plt.figure(1) # 이미 생성된 1번 그래프 창을 지정함
plt.plot(x, y2) # 지정된 그래프 창에 그래프를 그림
```

```
plt.figure(2) # 이미 생성된 2번 그래프 창을 지정함
plt.clf() # 2번 그래프 창에 그려진 모든 그래프를 지움
plt.plot(x, y1) # 지정된 그래프 창에 그래프를 그림

plt.show()
```

In [ ]:
```
import numpy as np

# 데이터 생성
x = np.arange(0, 10, 0.1)
y1 = 0.3*(x-5)**2 + 1
y2 = -1.5*x + 3
y3 = np.sin(x)**2 # NumPy에서 sin()은 np.sin()으로 입력
y4 = 10*np.exp(-x) + 1 # NumPy에서 exp()는 np.exp()로 입력

# 2 × 2 행렬로 이뤄진 하위 그래프에서 p에 따라 위치를 지정
plt.subplot(2,2,1) # p는 1
plt.plot(x,y1)

plt.subplot(2,2,2) # p는 2
plt.plot(x,y2)

plt.subplot(2,2,3) # p는 3
plt.plot(x,y3)

plt.subplot(2,2,4) # p는 4
plt.plot(x,y4)

plt.show()
```

- 그래프의 출력 범위 지정하기

In [ ]:
```
import numpy as np

x = np.linspace(-4, 4,100) # [-4, 4] 범위에서 100개의 값 생성

y1 = x**3
y2 = 10*x**2 - 2

plt.plot(x, y1, x, y2)
plt.show()
```

In [ ]:
```
plt.plot(x, y1, x, y2)
plt.xlim(-1, 1)
plt.ylim(-3, 3)
plt.show()
```

2. 그래프 꾸미기

- plot 함수와 Line Styling
- pyplot 모듈 내 plot 함수의 정의

- 출력 형식 지정

In [ ]:
```
import numpy as np
x = np.arange(0, 5, 1)
```

```
y1 = x
y2 = x + 1
y3 = x + 2
y4 = x + 3
```

In [ ]:
```
plt.plot(x, y1, x, y2, x, y3, x, y4)
plt.show()
```

In [ ]:
```
plt.plot(x, y1, 'm', x, y2,'y', x, y3, 'k', x, y4, 'c')
plt.show()
```

In [ ]:
```
plt.plot(x, y1, '-', x, y2, '--', x, y3, ':',  x, y4, '-.')
plt.show()
```

In [ ]:
```
plt.plot(x, y1, 'o', x, y2, '^',x, y3, 's', x, y4, 'd')
plt.show()
```

In [ ]:
```
plt.plot(x, y1, '>--r', x, y2, 's-g', x, y3, 'd:b', x, y4, '-.Xc')
plt.show()
```

- 라벨, 제목, 격자, 범례, 문자열 표시

함수	설명
xlabel, ylabel	x축과 y축에 전체에 대한 label
xticks, yticks	x축과 y축에 대응 값에 대한 label
title	차트의 제목
legend	범례 표시
axis	X축 y축의 최소 및 최대값 설정 [xmin, xmax, ymin, ymax]
annotate	xy point 에 text로 annotation 를 함. annotate(*args, **kwargs)

In [ ]:
```
import numpy as np

x = np.arange(-4.5, 5, 0.5)
y = 2*x**3

plt.plot(x,y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

In [ ]:
```
plt.plot(x,y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph title')
plt.show()
```

In [ ]:
```
plt.plot(x,y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph title')
plt.grid(True) # 'plt.grid()'도 가능
```

In [ ]:
```
# Styling 후의 Chart
# matplotlib의 pyplot 모듈을 이용하여 그래프를 생성

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

```
# 데이터 준비
X = np.arange(0, 6, 0.1) # 0에서 6까지 0.1 간격으로 생성
Y1 = np.sin(X)
Y2 = np.cos(X)

# 그래프 그리기
plt.plot(X, Y1, color="magenta", label="sin", linestyle="dashed", linewidth=
plt.plot(X, Y2, c="yellow", label="cos", ls="dashdot", lw=1.5)
plt.xlabel("x")
plt.ylabel("y")
plt.title("sin & cos")
plt.legend()

# 화면에 출력
plt.show()
```

In [ ]:
```
import numpy as np
x = np.arange(0, 5, 1)
y1 = x
y2 = x + 1
y3 = x + 2
y4 = x + 3

plt.plot(x, y1, '>--r', x, y2, 's-g', x, y3, 'd:b', x, y4, '-.Xc')
plt.legend(['data1', 'data2', 'data3', 'data4'])
plt.show()
```

In [ ]:
```
plt.plot(x, y1, '>--r', x, y2, 's-g', x, y3, 'd:b', x, y4, '-.Xc')
plt.legend(['data1', 'data2', 'data3', 'data4'], loc = 'lower right')
plt.show()
```

In [ ]:
```
plt.plot(x, y1, '>--r', x, y2, 's-g', x, y3, 'd:b', x, y4, '-.Xc')
plt.legend(['data1', 'data2', 'data3', 'data4'], loc = 4)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph title')
plt.grid(True)
```

---------- END ----------