파이션 Matplotlib

충북대학교 소프트웨어학과
류관희

목 차

❖ **Part 1. Matplotlib 소개**
- 개요 및 설치
- 선, 점, 버블, 막대 그래프 그리기
- 히스토그램 그리기

❖ **Part 2. Matplotlib 격자 등 그리기**
- 선그리기 주석 달기
- 격자 표시, 파이챠트 그리기
- 정규분포 그리기

❖ **Part 3. Matplotlib 고급 그리기**
- 균등분포 그리기
- 산포도 그리기
- 영상 및 애니메이션 그리기
- 경사하강법 그리기

01

Matplotlib 소개

- 개요 및 설치
- 선, 점, 버블, 막대 그래프 그리기
- 히스토그램 그리기

02

Matplotlib 격자 등 그리기

- 선그리기 주석 달기
- 격자 표시, 파이챠트 그리기
- 정규분포 그리기

03

Matplotlib 고급 그리기

- 균등분포 그리기
- 산포도 그리기
- 영상 및 애니메이션 그리기
- 경사하강법 그리기

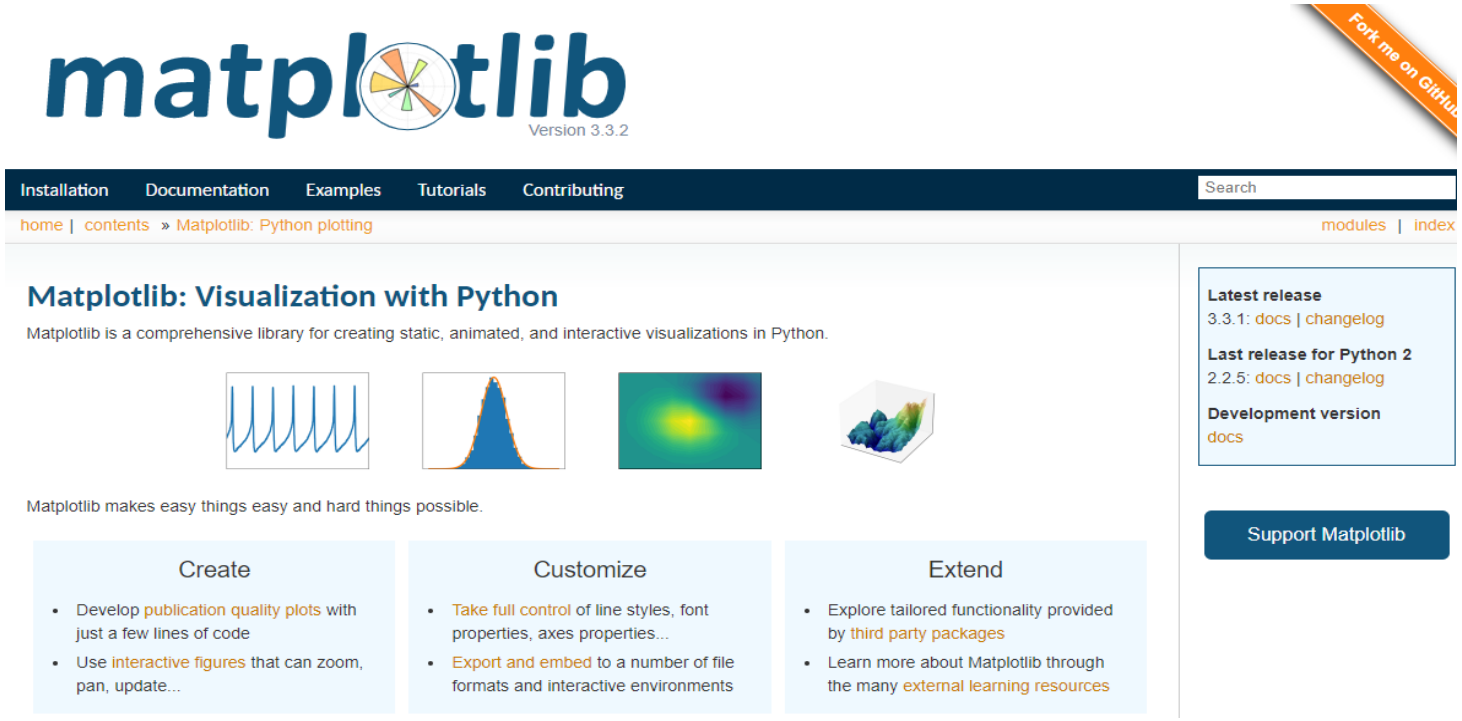
학습목표

- Matplotlib에서 다음과 같은 기능을 공부한다.
  - ✓ 설치하는 방법
  - ✓ 사용하는 방법
  - ✓ 선, 점, 버블 그리기
  - ✓ 막대 그래프 그리기
  - ✓ 히스토그램 그리기

Matplotlib 개요

- Python에서 정적, 애니메이션 및 대화 형 시각화를 만들기위한 포괄적 인 라이브러리입니다.

https://matplotlib.org/



Matplotlib 설치

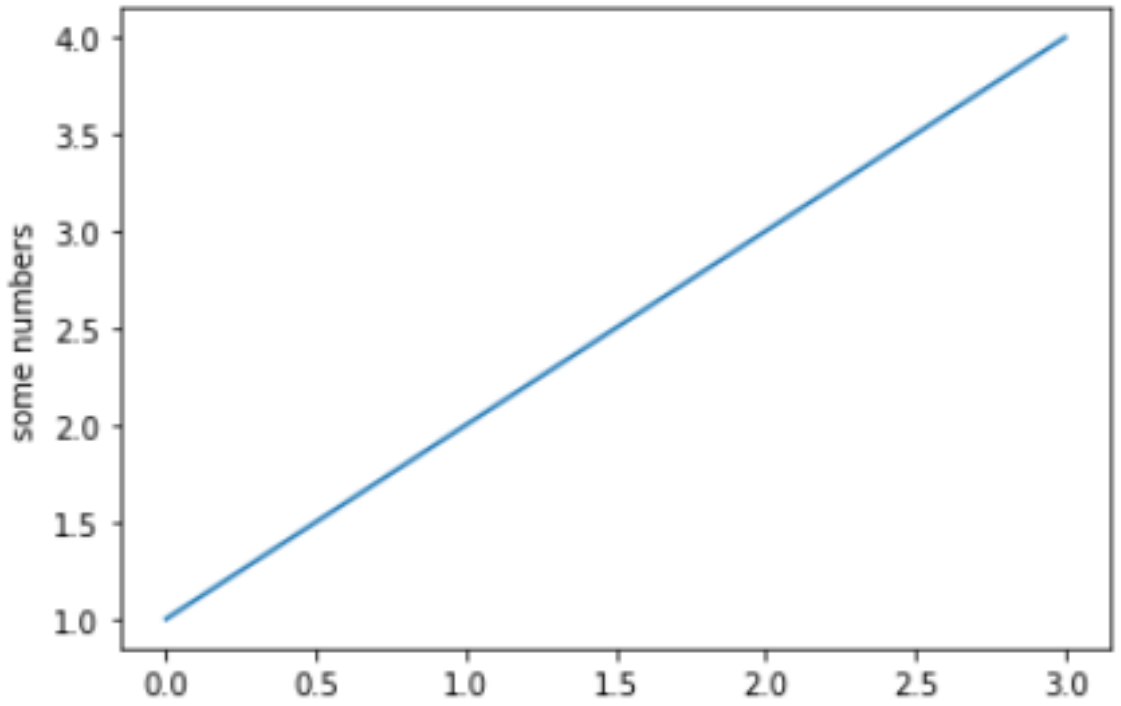
- pip install matplotlib
- import matplotlib.pyplot as plt

Matplotlib 기능

- Lines, bars and markers
- Images, contours and fields
- Subplots, axes and figures
- Statistics
- Pie and polar charts
- Text, labels and annotations
- Pyplot
- Color
- Shapes and collections
- Style sheets
- Axes Grid
- Axis Artist
- Showcase
- Animation
- Event Handling
- 3D plotting

라인 그리기

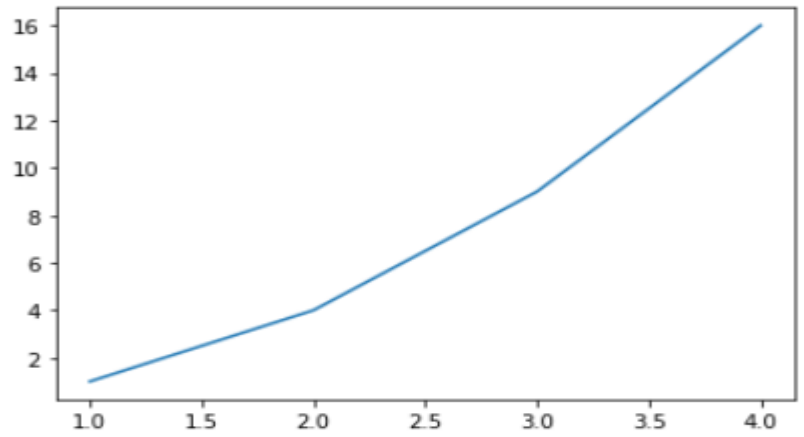
```
1  import matplotlib.pyplot as plt
2  plt.plot([1, 2, 3, 4])
3  plt.ylabel('some numbers')
4  plt.show()
```



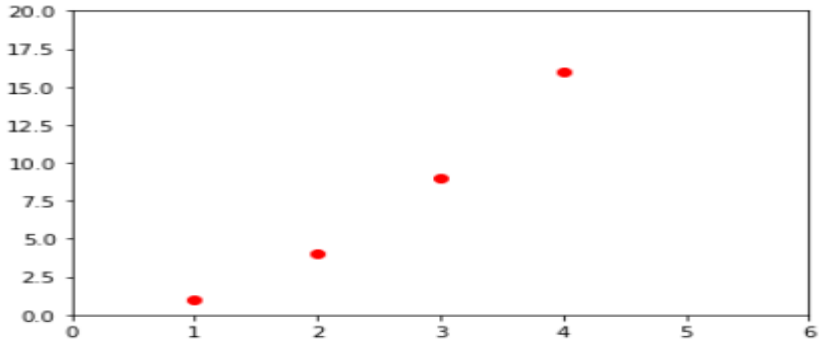
다각라인 및 점 그리기


```
1  plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
```

```
[<matplotlib.lines.Line2D at 0x2bc32a01940>]
```

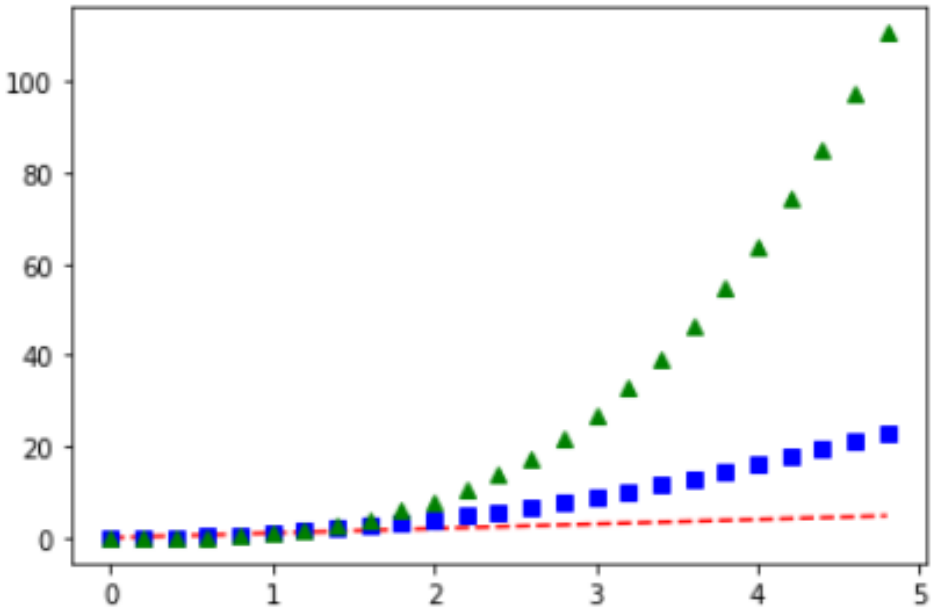



```
1  plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
2  plt.axis([0, 6, 0, 20])
3  plt.show()
```



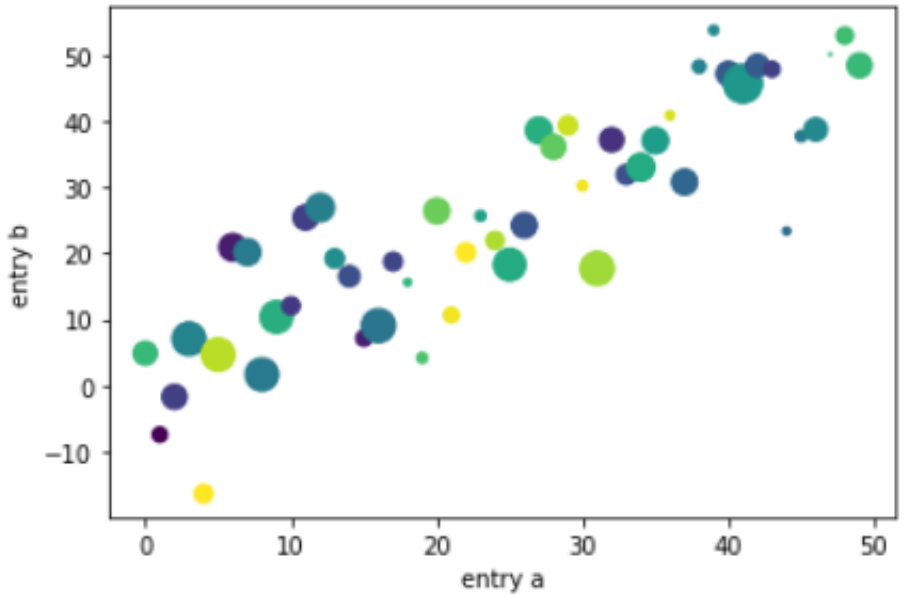
점 그리기


```python
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



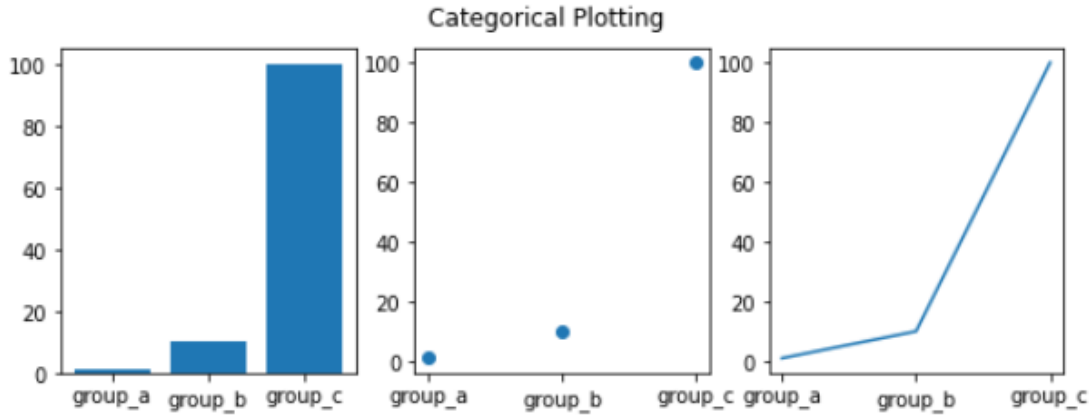
버블 그리기

```
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```



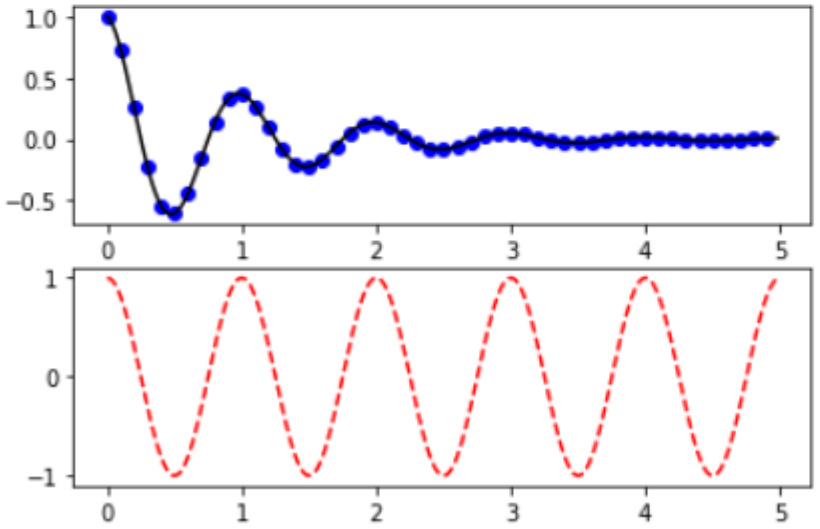
막대 그리기


```
1  names = ['group_a', 'group_b', 'group_c']
2  values = [1, 10, 100]
3
4  plt.figure(figsize=(9, 3))
5
6  plt.subplot(131)
7  plt.bar(names, values)
8  plt.subplot(132)
9  plt.scatter(names, values)
10 plt.subplot(133)
11 plt.plot(names, values)
12 plt.suptitle('Categorical Plotting')
13 plt.show()
```



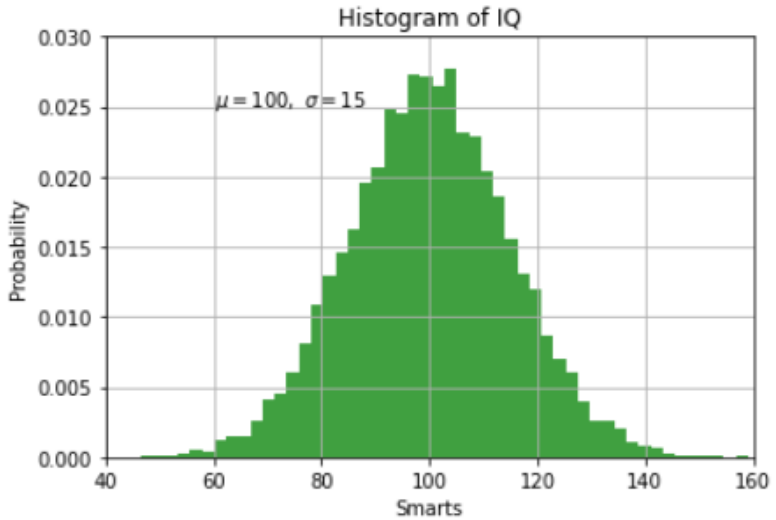
곡선 그리기

```
:
1  def f(t):
2      return np.exp(-t) * np.cos(2*np.pi*t)
3
4  t1 = np.arange(0.0, 5.0, 0.1)
5  t2 = np.arange(0.0, 5.0, 0.02)
6
7  plt.figure()
8  plt.subplot(211)
9  plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
10
11 plt.subplot(212)
12 plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
13 plt.show()
```



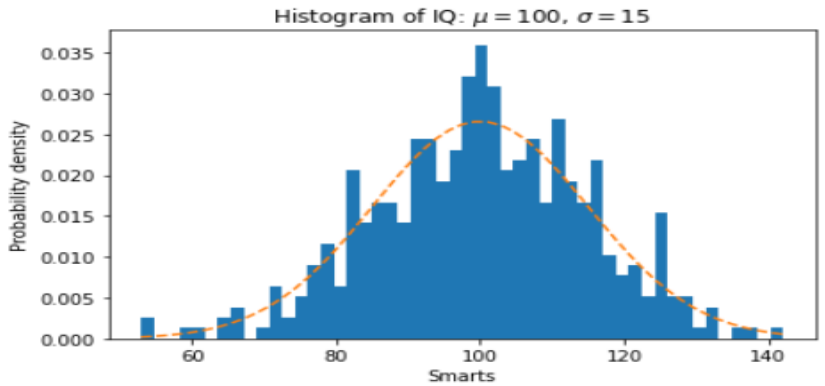
히스토그램 그리기

```
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1, facecolor='g', alpha=0.75)


plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```



히스토그램 그리기


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

np.random.seed(19680801)

# example data
mu = 100  # mean of distribution
sigma = 15  # standard deviation of distribution
x = mu + sigma * np.random.randn(437)
num_bins = 50
fig, ax = plt.subplots()
# the histogram of the data
n, bins, patches = ax.hist(x, num_bins, density=True)
# add a 'best fit' line
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) * np.exp(-0.5 * (1 / sigma * (bins - mu))**2))
ax.plot(bins, y, '--')
ax.set_xlabel('Smarts')
ax.set_ylabel('Probability density')
ax.set_title(r'Histogram of IQ: $\mu=100$, $\sigma=15$')

# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.show()
```



문제풀이

- Matplotlib 기능을 설명하시오.

- Matplotlib에서 제공하는 기능 중에 3가지를 말해 보시오.

요약

- 그림을 그리기 위해 널리 사용되는 Matplotlib을 공부하였음

- Matplotlib에서 제공하는 선 그리기 기능을 공부하였음.

01

Matplotlib 소개

- 개요 및 설치
- 선, 점, 버블, 막대 그래프 그리기
- 히스토그램 그리기

02

Matplotlib 격자 등 그리기

- 선그리기 주석 달기
- 격자 표시, 파이챠트 그리기
- 정규분포 그리기

03

Matplotlib 고급 그리기

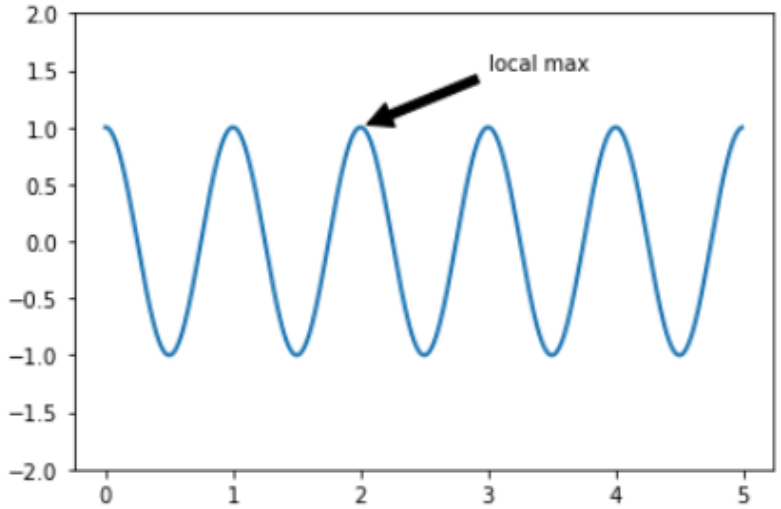
- 균등분포 그리기
- 산포도 그리기
- 영상 및 애니메이션 그리기
- 경사하강법 그리기

학습목표

- Matplotlib에서 다음과 같은 기능을 공부한다.
  - ✓ 주석달기
  - ✓ 격자 표시 그리기
  - ✓ 파이챠트 그리기
  - ✓ 정규 분포 그리기

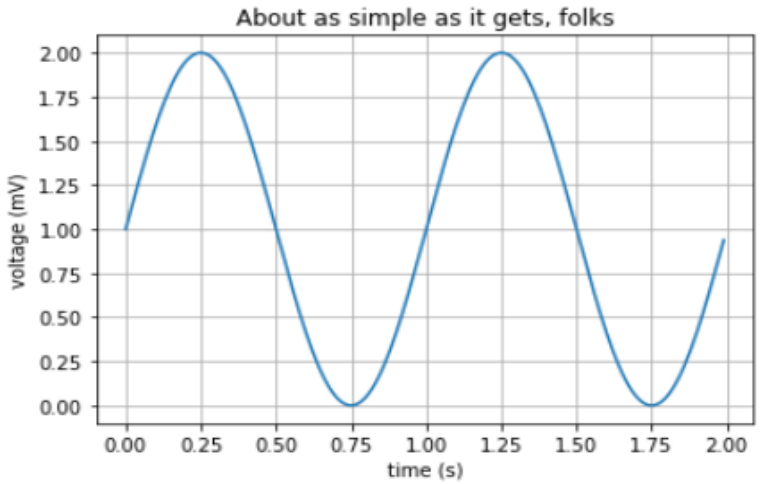
그리기에 주석달기


```
1  ax = plt.subplot(111)
2
3  t = np.arange(0.0, 5.0, 0.01)
4  s = np.cos(2*np.pi*t)
5  line, = plt.plot(t, s, lw=2)
6
7  plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
8               arrowprops=dict(facecolor='black', shrink=0.05),
9               )
10
11 plt.ylim(-2, 2)
12 plt.show()
```



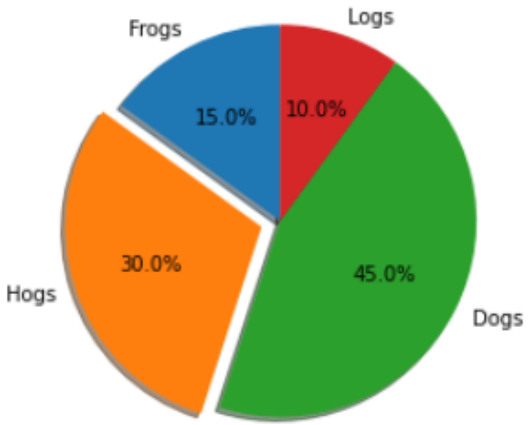
격자 표시하기

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

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)', title='About as simple as it gets, folks')
ax.grid()

fig.savefig("test.png")
plt.show()
```



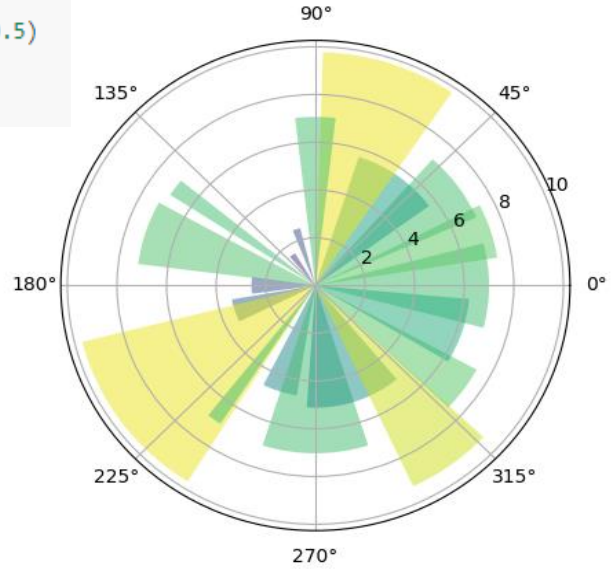
파이차트

```python
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```



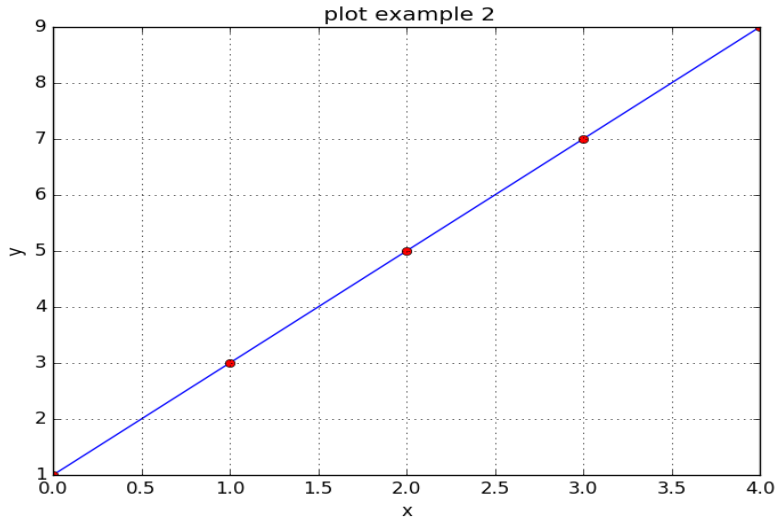
Polar 축에 Bar 챠트 그리기

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


# Fixing random state for reproducibility
np.random.seed(19680801)

# Compute pie slices
N = 20
theta = np.linspace(0.0, 2 * np.pi, N, endpoint=False)
radii = 10 * np.random.rand(N)
width = np.pi / 4 * np.random.rand(N)
colors = plt.cm.viridis(radii / 10.)

ax = plt.subplot(111, projection='polar')
ax.bar(theta, radii, width=width, bottom=0.0, color=colors, alpha=0.5)

plt.show()
```



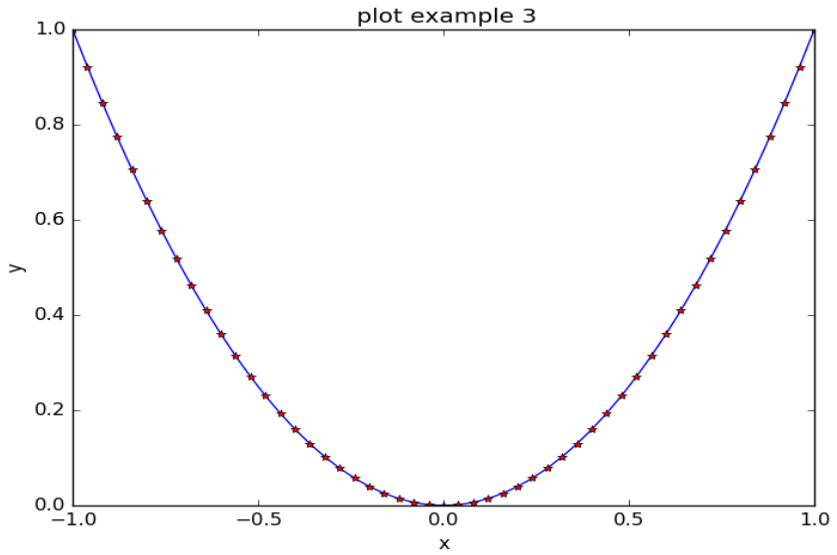
선그리기에 Labeling 추가하기

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(5)
y = np.array([1, 3, 5, 7, 9])
plt.plot(x, y, "b-", x, y, "ro")
plt.xlabel("x")
plt.ylabel("y")
plt.title(" plot example 2")
plt.grid(True)
plt.savefig("line.png")
plt.show()
```



선그리기에 Labeling 추가하기


```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(start = -1, stop = 1, num=51)
y = x**2
plt.plot(x, y, 'b-', x, y, 'r*')
plt.axis([-1, 1, 0, 1])

plt.xlabel("x")
plt.ylabel("y")
plt.title(" plot example 3")
plt.savefig("ex0951.png")
plt.show()
```

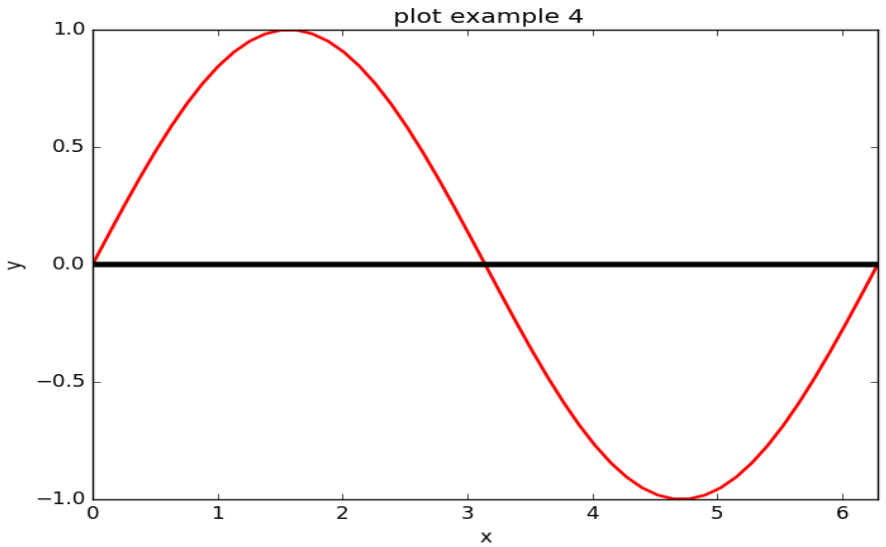


사인 함수 그리기


```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2*np.pi, num=51)
y = np.sin(x)
line = plt.plot(x, y)
xmin, xmax, ymin, ymax = np.amin(x), np.amax(x), -1, 1
plt.axis([xmin, xmax, ymin, ymax])
#plt.xlim(xmin, xmax)
#plt.ylim(ymin, ymax)
plt.plot([xmin, xmax], [0, 0], color='black', linewidth=4.0)
```

사인 함수 그리기


```
plt.setp(line, color='red', linewidth=2.0)
plt.xlabel("x")
plt.ylabel("y")
plt.title(" plot example 4")
plt.savefig(“sine.png")
plt.show()
```

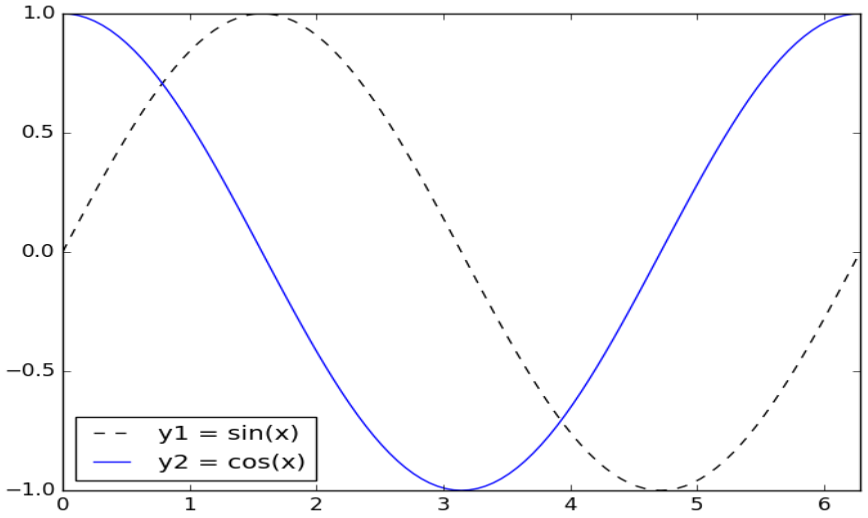


y1=sin(x), y2=cos(x), 범례생성


```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2*np.pi, num=101)
y1, y2 = np.sin(x), np.cos(x)
plt.plot(x, y1, "k--", label = "y1 = sin(x)")
plt.plot(x, y2, "b-", label = "y2 = cos(x)")
xmin, xmax, ymin, ymax = x[0], x[-1], -1, 1
```

y1=sin(x), y2=cos(x), 범례생성

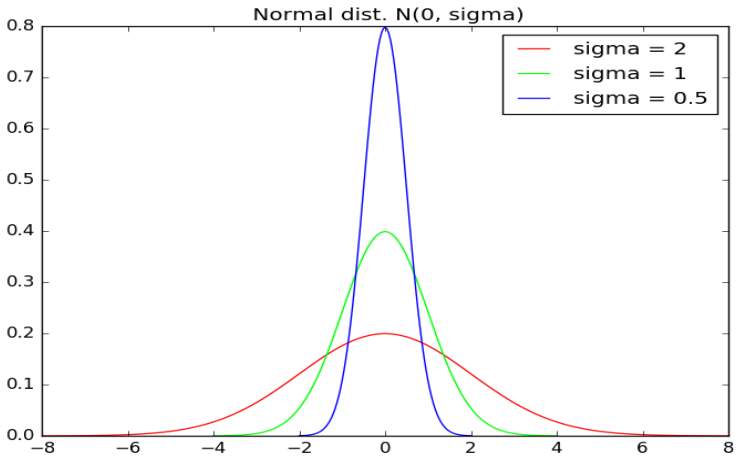

```
plt.axis([xmin, xmax, ymin, ymax])
#plt.xlim(xmin, xmax)
#plt.ylim(ymin, ymax)
plt.legend(loc="best")
plt.savefig(“sincos.png")
plt.show()
```



1D 정규분포 N(0, sigma)


```
mu, sigma= 0, 1
y2 = gauss(mu, sigma, x)
plt.plot(x, y2, color = "#00ff00", label = "sigma = 1")
mu, sigma= 0, 0.5
y3 = gauss(mu, sigma, x)
plt.plot(x, y3, color = "#0000ff", label = "sigma = 0.5")
plt.title(" Normal dist. N(0, sigma)")
plt.legend(loc="best")
plt.savefig(“norm1d.png")
```

1D 정규분포 그래프


Normal dist. N(0, sigma)
sigma = 2
sigma = 1
sigma = 0.5

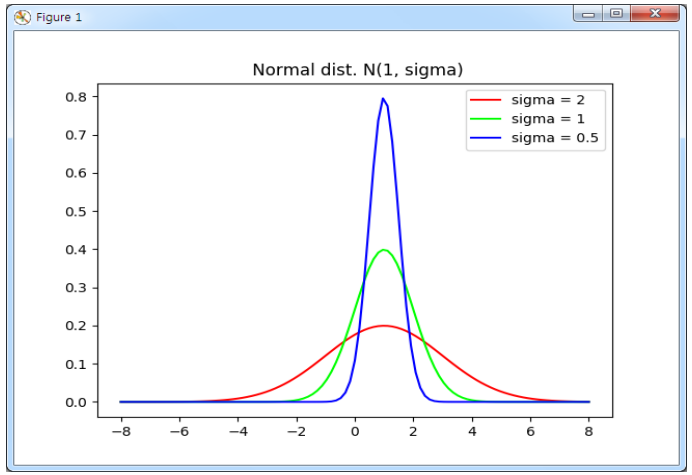


Figure 1
Normal dist. N(1, sigma)
sigma = 2
sigma = 1
sigma = 0.5

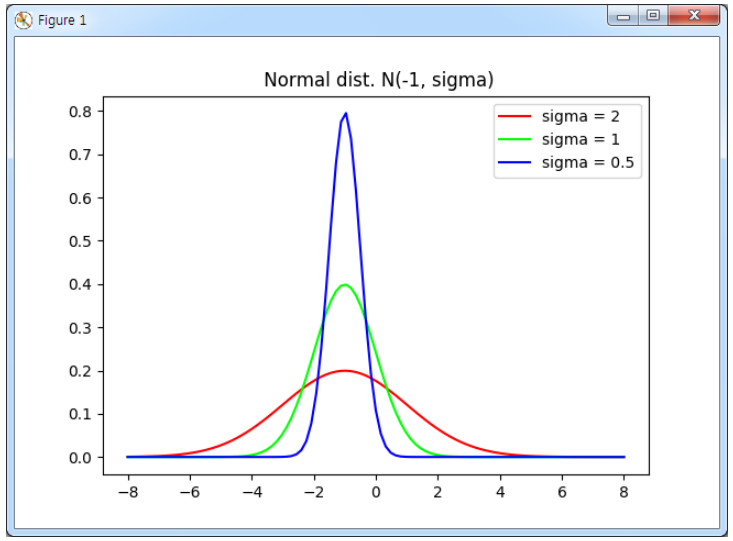


Figure 1
Normal dist. N(-1, sigma)
sigma = 2
sigma = 1
sigma = 0.5

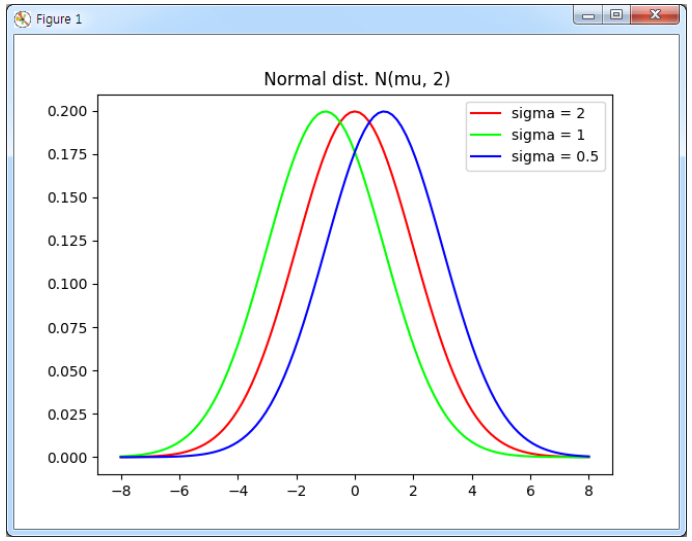
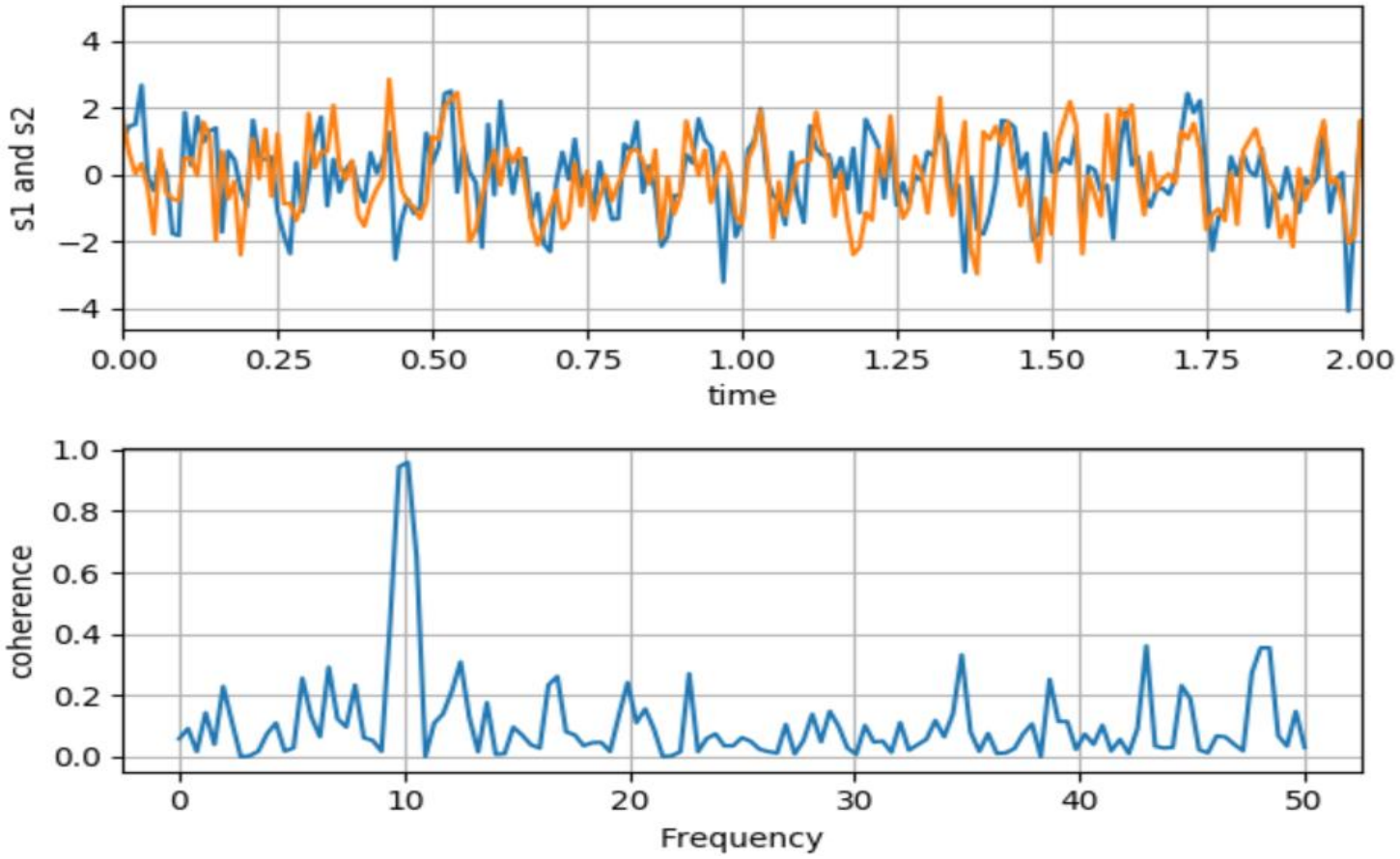


Figure 1
Normal dist. N(mu, 2)
sigma = 2
sigma = 1
sigma = 0.5

두 신호의 응집성 그리기


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

# Fixing random state for reproducibility
np.random.seed(19680801)

dt = 0.01
t = np.arange(0, 30, dt)
nse1 = np.random.randn(len(t))                 # white noise 1
nse2 = np.random.randn(len(t))                 # white noise 2

# Two signals with a coherent part at 10Hz and a random part
s1 = np.sin(2 * np.pi * 10 * t) + nse1
s2 = np.sin(2 * np.pi * 10 * t) + nse2

fig, axs = plt.subplots(2, 1)
axs[0].plot(t, s1, t, s2)
axs[0].set_xlim(0, 2)
axs[0].set_xlabel('time')
axs[0].set_ylabel('s1 and s2')
axs[0].grid(True)

cxy, f = axs[1].cohere(s1, s2, 256, 1. / dt)
axs[1].set_ylabel('coherence')

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

두 신호의 응집성 그리기

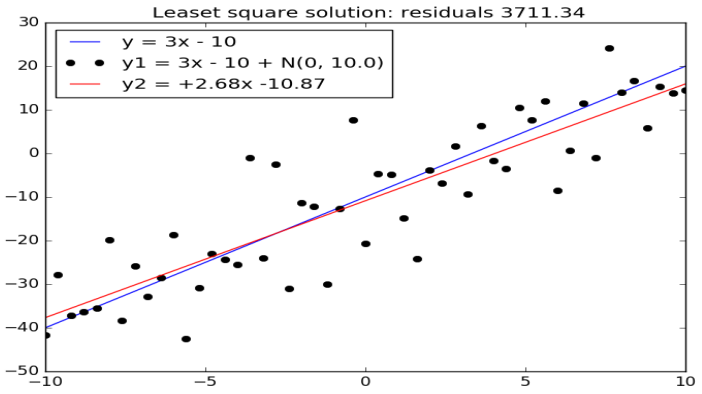

s1 and s2
4
2
0
−2
−4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
time
coherence
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
Frequency

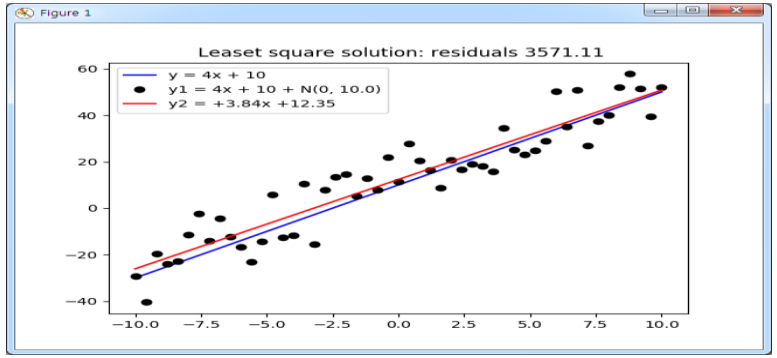
직선의 최소자승해


```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-10.0, 10.0, num=51)
m, c = 3, -10 # Truth
y = m*x + c
e = np.random.normal(0, 10.0, x.size) # noise
y1 = y + e
A = np.vstack([x, np.ones(len(x))]).T
p, residuals, r, s = np.linalg.lstsq(A, y1)
m1, c1 = p  # Estimate
y2 = m1*x + c1
```

직선의 최소자승해

```
plt.plot(x, y,  "b-", label =  "y = 3x - 10")
plt.plot(x, y1, "ko", label =  "y1 = 3x - 10 + N(0, 10.0)")
plt.plot(x, y2, "r-", label =  "y2 = {:+.2f}x {:+.2f}".format(m1, c1))
plt.title(" Leaset square solution: residuals {:.2f}".format(residuals[0]))
plt.legend(loc="best")
plt.savefig( " lineFit.png")
plt.show()
```





문제풀이

- Matplotlib에서 제공하는 Piechart 그리는 함수를 말해 보시오.

- 최소자승해를 구하고 Matplotlib을 이용하여 그해를 그리는 방법을 설명하시오.

요약

- Matplotlib를 사용하여 시각화하는 다양한 방법을 공부하였음.
  - 주석달기
  - 사인함수 그리기
  - 파이차트 그리기
  - 다중 라인 그리기

01

Matplotlib 소개

- 개요 및 설치
- 선, 점, 버블, 막대 그래프 그리기
- 히스토그램 그리기

02

Matplotlib 격자 등 그리기

- 선그리기 주석 달기
- 격자 표시, 파이챠트 그리기
- 정규분포 그리기

03

Matplotlib 고급 그리기

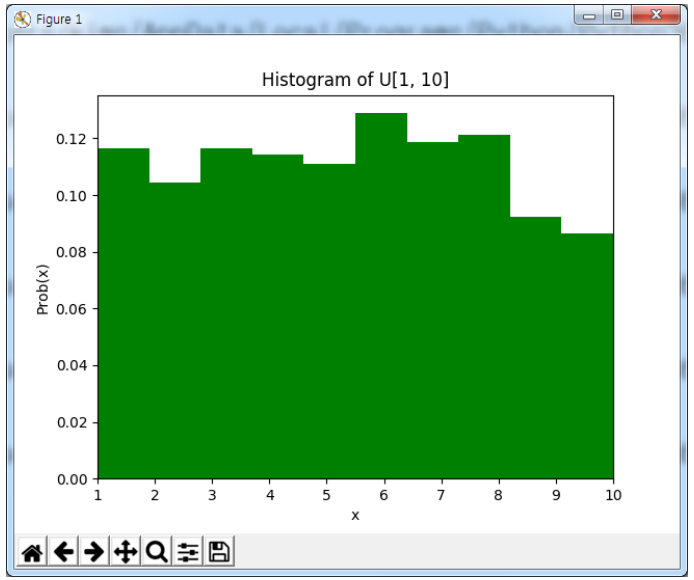
- 균등분포 그리기
- 산포도 그리기
- 영상 및 애니메이션 그리기
- 경사하강법 그리기

학습목표

- Matplotlib에서 다음과 같은 기능을 공부한다.
  - ✓ 균등 분포 그리기
  - ✓ 산포도 그리기
  - ✓ 영상 그리기
  - ✓ 애니메이션
  - ✓ 경사하강법 그리기

균등분포의 히스토그램

```
import numpy as np
import matplotlib.pyplot as plt
x = np.random.random_integers(1, 10, size=1000)
n, bins, patches = plt.hist(x, 10, normed=True, color='g')
print(np.sum(n*np.diff(bins)))
xmin, xmax = np.amin(x), np.amax(x)
plt.xlim(xmin, xmax)
plt.xlabel("x")
plt.ylabel('Prob(x)')
plt.title("Histogram of U[1, 10]")
plt.show()
```

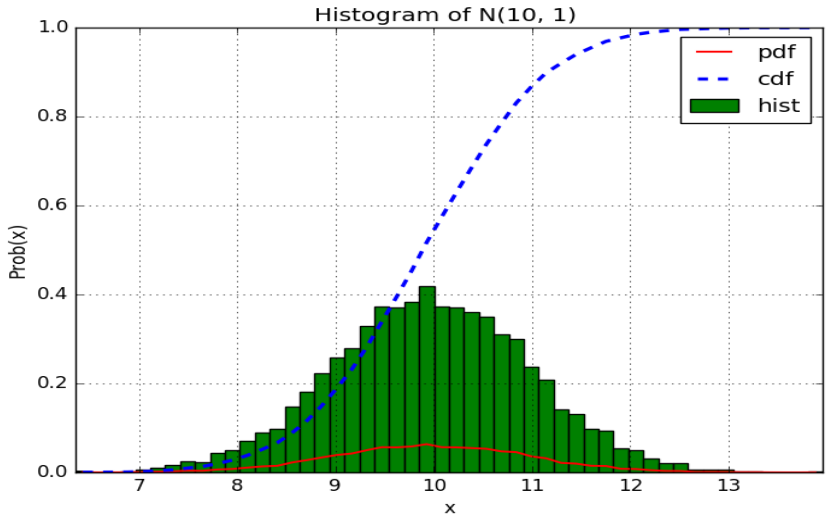


정규분포의 히스토그램


```
import numpy as np
import matplotlib.pyplot as plt
mu, sigma = 10, 1
x = mu + sigma * np.random.randn(10000)
n, bins, patches = plt.hist(x, 50, normed=True, color='g', label = 'hist')
print(np.sum(n*np.diff(bins)))

pdf = n*np.diff(bins)
print(np.sum(pdf))
middle_x = bins[:-1] + np.diff(bins)/2
plt.plot(middle_x, pdf, 'r-', linewidth=1.5, label='pdf')
```

정규분포의 히스토그램

```
cdf = np.cumsum(pdf)
#cdf /= cdf[-1]
plt.plot(middle_x, cdf, 'b--', linewidth=2.5, label='cdf')
xmin, xmax = np.amin(x), np.amax(x)
plt.xlim(xmin, xmax)
plt.xlabel("x")
plt.ylabel('Prob(x)')
plt.title("Histogram of N(10, 1) ")
plt.grid(True)
plt.legend(loc="best")
plt.show()
```

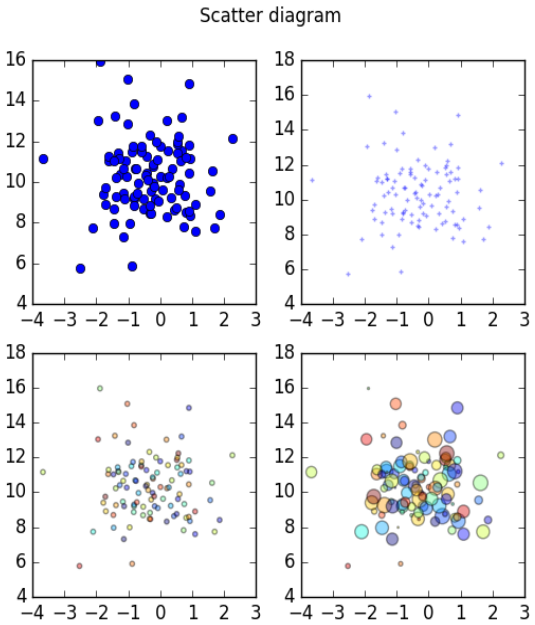


산포도그리기


```
import numpy as np
import matplotlib.pyplot as plt
ax.add_patch( patches.Rectangle( (0.1, 0.4), 0.2, 0.2,
fill=False, linestyle='dashed', linewidth=2))
plt.rc('figure', figsize=(6, 6)) # plt.rcParams['figure.figsize'] = (6, 6)
fig, ax = plt.subplots(2, 2)
fig.suptitle("Scatter diagram")
ax[0][0].plot(x, y, "o")
ax[0][1].scatter(x, y, color='b', s=10, marker='+')
```

산포도 그리기

```
colors = np.random.rand(N)
ax[1][0].scatter(x, y, c=colors, s=10, marker='o', alpha=.4)
size1 = np.random.random_integers(1, 100, size=N)
ax[1][1].scatter(x, y, c=colors, s=size1, marker='o', alpha=.4)
plt.savefig( " scatter.png")
plt.show()
```



영상 그리기

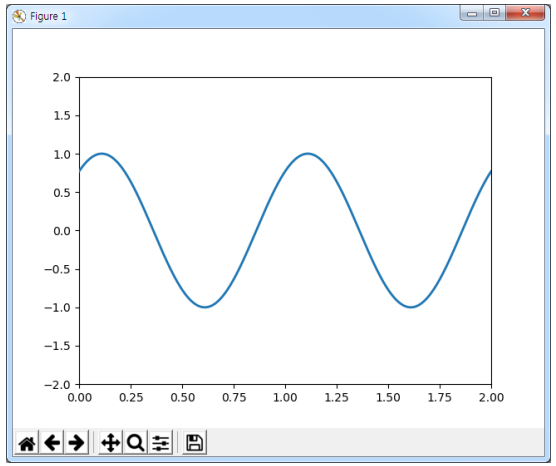
```
import matplotlib.pyplot as plt
import matplotlib.image as img
image = img.imread("Desert.jpg")
print("image.shape =", image.shape)
plt.imshow(image)
img.imsave( " desert1.png " , image)
#plt.savefig( " desert2.png " )
plt.show()
```

애니메이션(sine 그래프)


```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import animation
fig = plt.figure()
ax = plt.axes(xlim=(0, 2), ylim=(-2, 2))
line, = ax.plot([], [], lw=2)
def init():
    line.set_data([], [])
    return line,
```

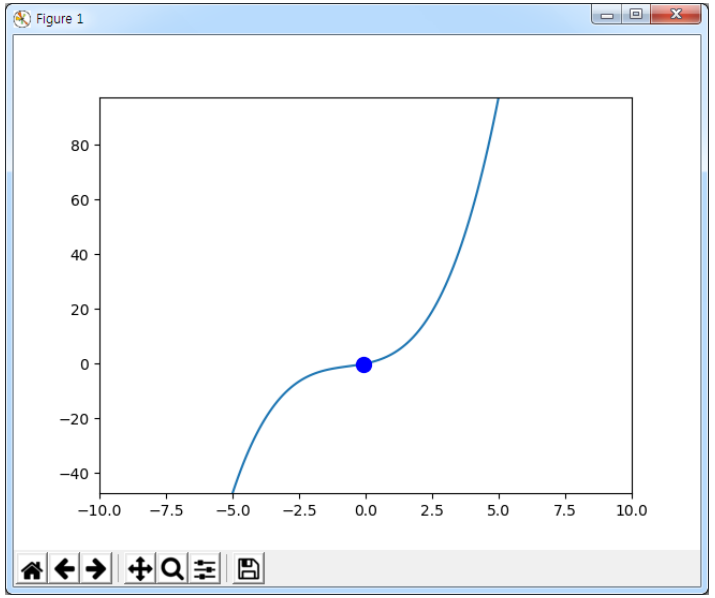
애니메이션(sine 그래프)

```
def animate(i):
##    print("i=", i)
    x = np.linspace(0, 2, 1000)
    y = np.sin(2 * np.pi * (x - 0.01 * i))
    line.set_data(x, y)
    return line,
anim = animation.FuncAnimation(fig, animate, init_func=init,
                               frames=200, interval=20, blit=True)
plt.show()
```



애니메이션(움직이는 원)


```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
fig = plt.figure()
line, = plt.plot([], [], "b-o", markersize= 10,lw=2)
x1 = np.linspace(-5, 5, num=101)
y1 = 0.5*x1**3+x1**2+x1*2
xmax, xmin = 10, -10
ymax, ymin = max(y1), min(y1)
…… 채워 보세요.
```



경사하강법(Gradient Decent Alg.)

Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

  until we hopefully end up at a minimum

경사하강법(Gradient Decent Alg.)

repeat until convergence{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

}

- $:=$: "Assignment Operator (대입 연산자)
- α : Learning Rate (학습 속도)
- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$: Derivative Term (미분 계수)

경사하강법(Gradient Decent Alg.)

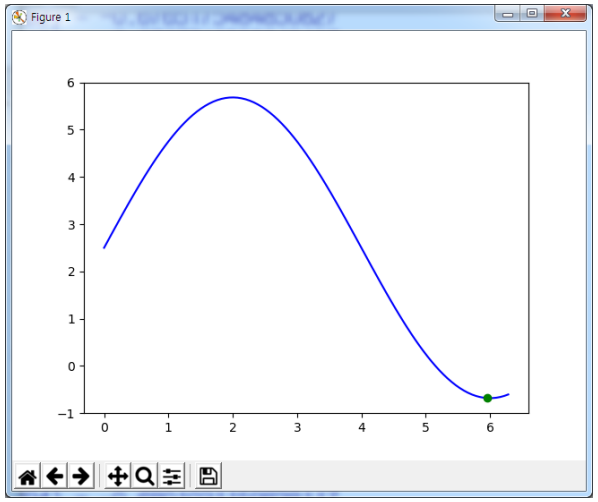

```
import numpy as np
import matplotlib.pyplot as plt
x_old = 0
x_new = 3 # The algorithm starts at x=6
eps = 0.01  # step size
precision = 0.001

def f(x):
##  return x**4 -3*x**3 + 2
##  return (3*x - 2)*(x + 2)**2
  return 2.5 + 10/np.pi * np.sin(np.pi*x/4)
```

경사하강법(Gradient Decent Alg.)


```
##def f_prime(x):
##    return 4 * x**3 - 9 * x**2
def dx(f, x):
  h = 0.001
  return (f(x+h) - f(x))/h
while abs(x_new - x_old) > precision:
    x_old = x_new
##    x_new = x_old - eps * f_prime(x_old)
    x_new = x_old - eps * dx(f, x_old)
    print("f(%s) = %s"%(x_new, f(x_new) ) )
```

경사하강법(Gradient Decent Alg.)


```
x = np.linspace(0, 2*np.pi, num = 101)
plt.plot(x, f(x), color='blue')
plt.plot([x_new], [f(x_new)], "go")
plt.show()
```



#실행결과

f(3.0176846097164773) = 4.719312422396786

f(3.0356129425749163) = 4.686963231516396

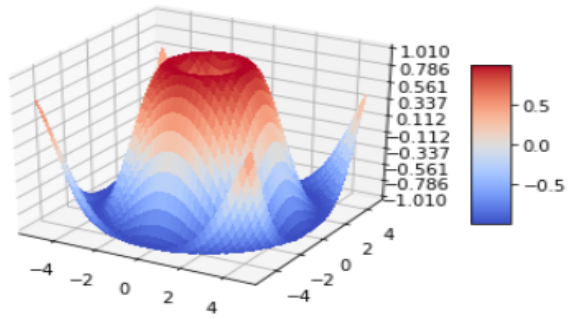
…..

f(5.948920729447565) = -0.6805377353958311

f(5.949913589895848) = -0.6806363195365486

3D 시각화

```python
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator
import numpy as np

fig, ax = plt.subplots(subplot_kw={"projection": "3d"})

# Make data.
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm, linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```



문제풀이

- Matplotlib을 이용하여 정규 분포 히스토그램을 그리기 위한 방법을 설명하시오.

- 경사하강 알고리즘을 이용하여 해를 구할때 왜 시각화를 사용하는지 설명하시오.

요약

- Matplotlib를 사용하여 시각화하는 다양한 방법을 공부하였음.
  - 히스토그램
  - 산포도
  - 경사 하강 알고리즘을 위한 시각화