202001555 지은미

In [52]:
```
from collections import Counter
import math, random

def random_kid():
    return random.choice(["boy","girl"]) #boy, girl 중 랜덤선택

kid_test_list = [random_kid() for i in range(10)]
kid_test_list

both_girls=0
older_girl=0
either_girl=0

random.seed(0)
for _ in range(10000):
    younger=random_kid()
    older=random_kid()
    if older=="girl":
        older_girl+=1
    if older=="girl" and younger=="girl":
        both_girls+=1
    if older=="girl" or younger=="girl":
        either_girl+=1
print(younger)
print("P(both | older):", both_girls/older_girl)
print("P(both | either):", both_girls/either_girl)
```

```
girl
P(both | older): 0.5007089325501317
P(both | either): 0.3311897106109325
```

변형

빨간공, 파란공이 있는 두 개의 상자에서 둘다 빨간공일 확률

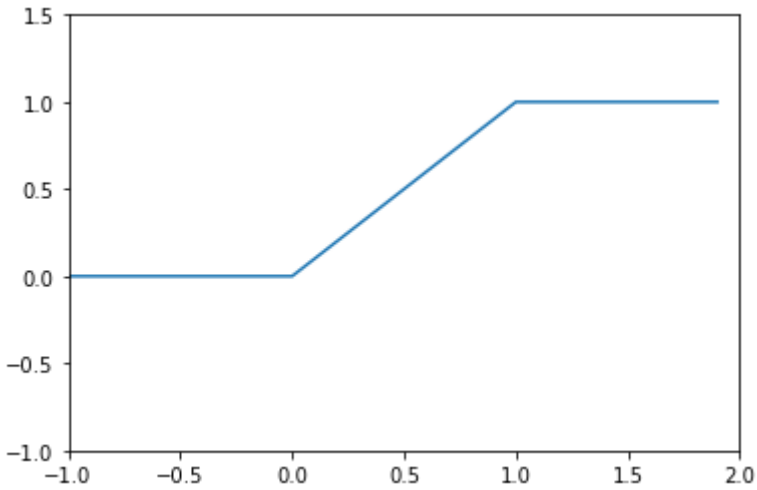
In [53]:
```
from collections import Counter
import math, random

def random_ball():
    return random.choice(["r","b"])

both_red=0
first_red=0
either_red=0

random.seed(0)
for _ in range(10000):
    first=random_ball()
    second=random_ball()
    if second=="r" or first=="r": #빨간공이 한개 이상 나올 확률 +1
        either_red+=1
    if second=="r" and first=="r": # 두 상자에서 빨간공이 나올 경우+1
        both_red+=1
    if first=="r": # 1번 상자에서 빨간공이 나올 경우+1
        first_red+=1

print("P(both|first):",both_red/first_red)#1번 상자에서 빨간공이 나오고 둘다 빨간공일
print("P(both|either):",both_red/either_red)#빨간공이 한개 이상 나오고 둘다 빨간공일
```

```
P(both|first): 0.5070985802839432
```

```
P(both|either): 0.3368756641870351
```

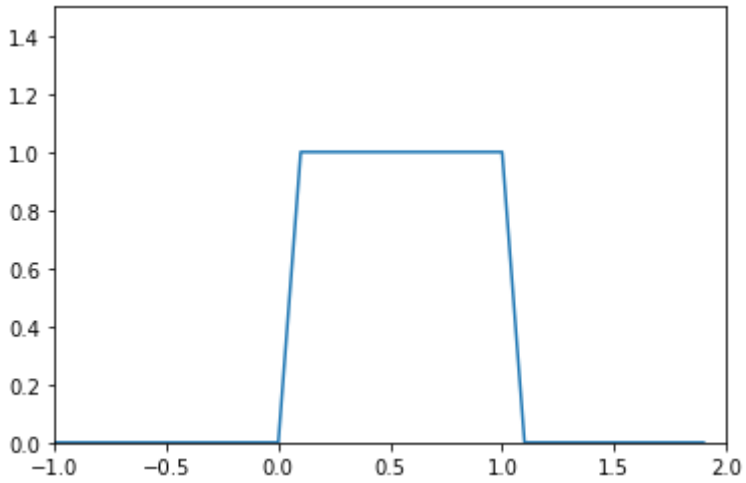
In [66]:
```
#연속 분포
def uniform_pdf(x):
    return 1 if x>=0 and x<1 else 0

def uniform_cdf(x):
    """returns the probability that a uniform random variable is less than x"""
    if x<0:
        return 0
    elif x<1:
        return x
    else:
        return 1

import numpy as np
x=np.arange(-1.0,2.0,0.1)
result_array=np.vectorize(uniform_cdf, otypes=[np.float])(x)

import matplotlib.pyplot as plt
%pylab inline

plt.plot(x,result_array)
plt.axis([-1,2,-1,1.5])
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```



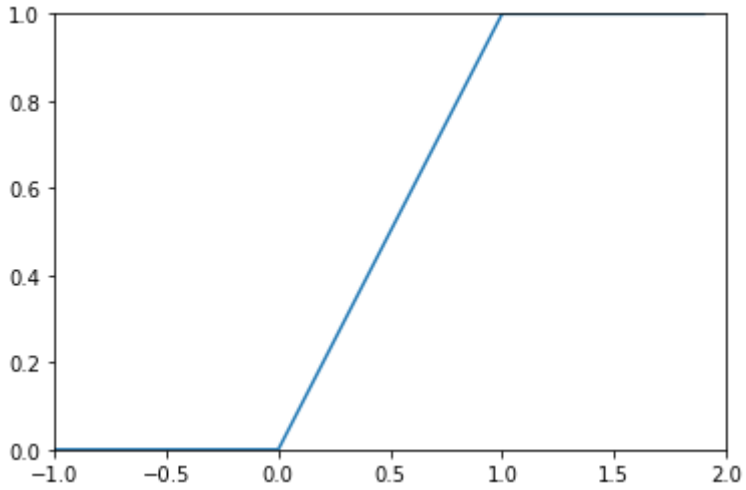
In [55]:
```
#변형

x=np.arange(-1.0,2.0,0.1)

result_array=np.vectorize(uniform_pdf, otypes=[np.float])(x)

import matplotlib.pyplot as plt
%pylab inline

plt.plot(x,result_array)
plt.axis([-1,2,0,1.5])
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```



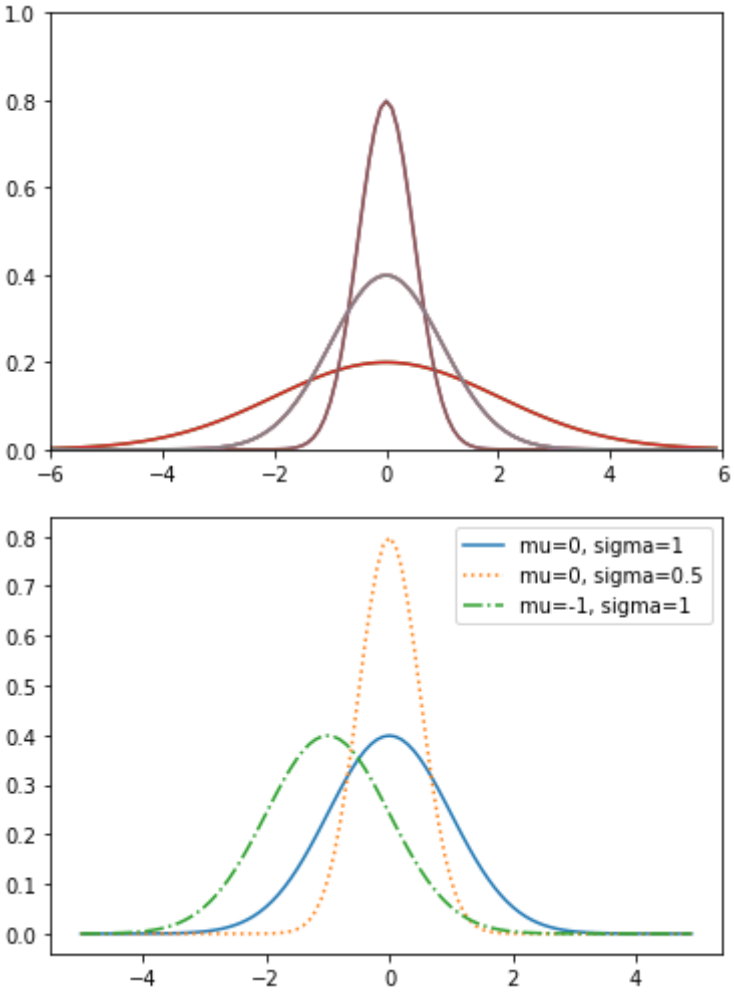
In [56]:
```
#변형

x=np.arange(-1.0,2.0,0.1)

result_array=np.vectorize(uniform_cdf, otypes=[np.float])(x)

import matplotlib.pyplot as plt
%pylab inline

plt.plot(x,result_array)
plt.axis([-1,2,0,1])
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```



In [57]:
```
#정규분포
def normal_pdf(x,mu=0,sigma=1):
    sqrt_two_pi=math.sqrt(2*math.pi)
    return (math.exp(-(x-mu)**2/2/sigma**2)/(sqrt_two_pi*sigma))

for sigma_value in [1,2,0.5,1]:
    x=np.arange(-6.0,6.0,0.1)
    result_array=np.vectorize(normal_pdf,otypes=[np.float])(x,sigma=sigma_value)
    plt.plot(x,result_array)
    plt.plot(x,result_array)

plt.axis([-6,6,0,1])
plt.show()

def plot_normal_pdfs(plt):
    xs=[x/10.0 for x in range(-50,50)]
    plt.plot(xs,[normal_pdf(x,sigma=1) for x in xs],'-',label='mu=0, sigma=1')
```

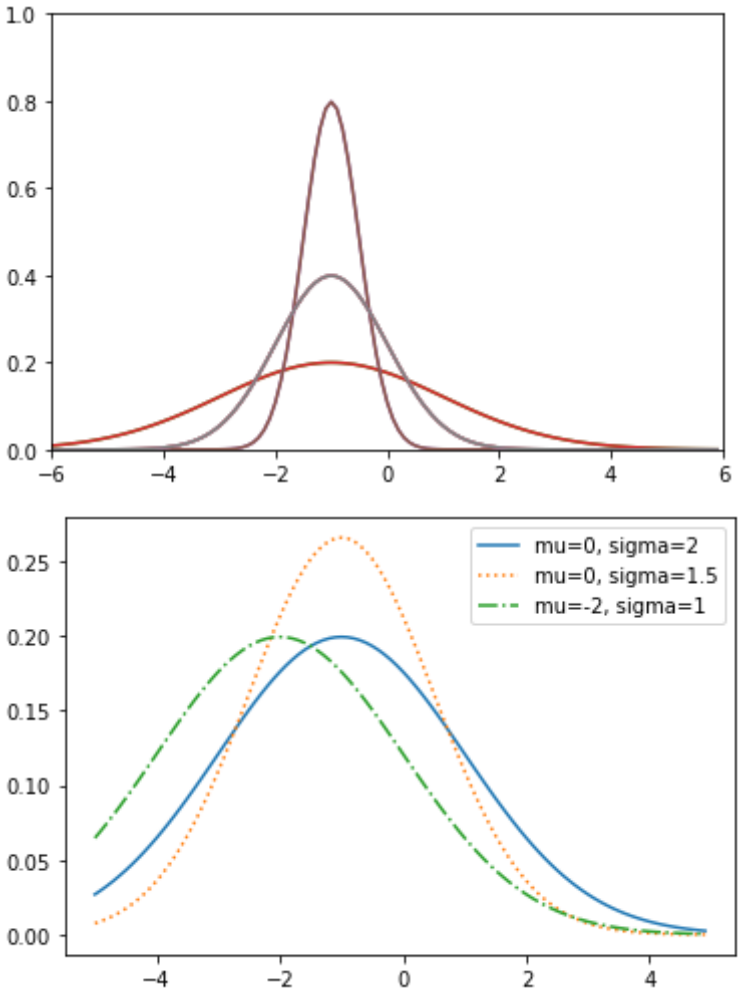
```
    plt.plot(xs,[normal_pdf(x,sigma=0.5) for x in xs],':',label='mu=0, sigma=0.5')
    plt.plot(xs,[normal_pdf(x,mu=0-1) for x in xs],'-.',label='mu=-1, sigma=1')
    plt.legend()
    plt.show()

import matplotlib.pyplot as plt
plot_normal_pdfs(plt)
```



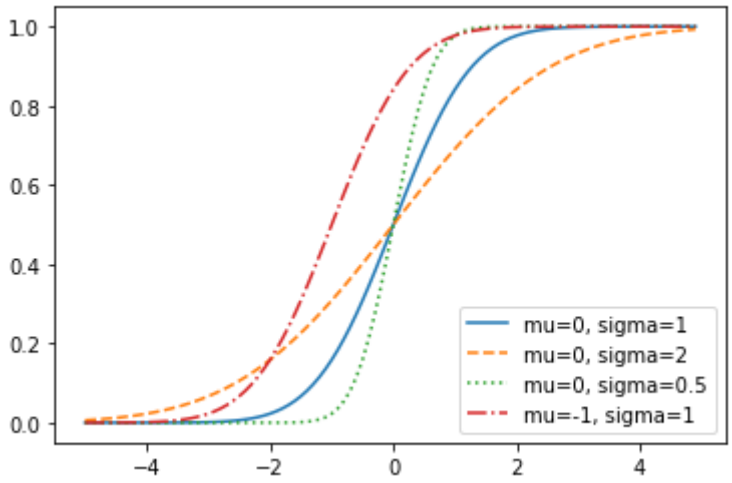
In [58]:

```
#변형
def normal_pdf(x,mu=-1,sigma=2):
    sqrt_two_pi=math.sqrt(2*math.pi)
    return (math.exp(-(x-mu)**2/2/sigma**2)/(sqrt_two_pi*sigma))

for sigma_value in [1,2,0.5,1]:
    x=np.arange(-6.0,6.0,0.1)
    result_array=np.vectorize(normal_pdf,otypes=[np.float])(x,sigma=sigma_value)
    plt.plot(x,result_array)
    plt.plot(x,result_array)

plt.axis([-6,6,0,1])
plt.show()

def plot_normal_pdfs(plt):
    xs=[x/10.0 for x in range(-50,50)]
    plt.plot(xs,[normal_pdf(x,sigma=2) for x in xs],'-',label='mu=0, sigma=2')
    plt.plot(xs,[normal_pdf(x,sigma=1.5) for x in xs],':',label='mu=0, sigma=1.5')
    plt.plot(xs,[normal_pdf(x,mu=0-2) for x in xs],'-.',label='mu=-2, sigma=1')
    plt.legend()
    plt.show()

import matplotlib.pyplot as plt
```

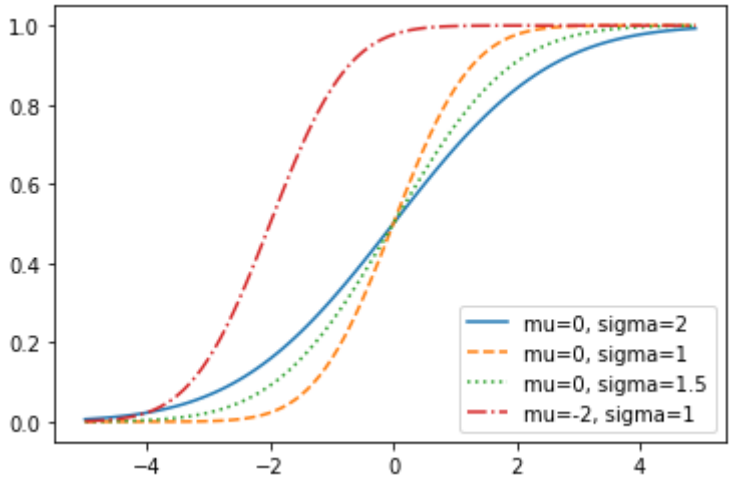
```
plot_normal_pdfs(plt)
```



In [59]:
```
#정규분포
def normal_cdf(x,mu=0,sigma=1):
    return (1+math.erf((x-mu)/math.sqrt(2)/sigma))/2

def plot_normal_cdfs(plt):
    xs=[x/10.0 for x in range(-50,50)]
    plt.plot(xs,[normal_cdf(x,sigma=1) for x in xs],'-',label='mu=0, sigma=1')
    plt.plot(xs,[normal_cdf(x,sigma=2) for x in xs],'--',label='mu=0, sigma=2')
    plt.plot(xs,[normal_cdf(x,sigma=0.5) for x in xs],':',label='mu=0, sigma=0.5')
    plt.plot(xs,[normal_cdf(x,mu=-1) for x in xs],'-.',label='mu=-1, sigma=1')
    plt.legend(loc=4)
    plt.show()

import matplotlib.pyplot as plt
plot_normal_cdfs(plt)
```



In [60]:
```
#변형
def normal_cdf(x,mu=0,sigma=1):
    return (1+math.erf((x-mu)/math.sqrt(2)/sigma))/2

def plot_normal_cdfs(plt):
    xs=[x/10.0 for x in range(-50,50)]
    plt.plot(xs,[normal_cdf(x,sigma=2) for x in xs],'-',label='mu=0, sigma=2')
    plt.plot(xs,[normal_cdf(x,sigma=1) for x in xs],'--',label='mu=0, sigma=1')
    plt.plot(xs,[normal_cdf(x,sigma=1.5) for x in xs],':',label='mu=0, sigma=1.5')
    plt.plot(xs,[normal_cdf(x,mu=-2) for x in xs],'-.',label='mu=-2, sigma=1')
    plt.legend(loc=4)
    plt.show()

import matplotlib.pyplot as plt
plot_normal_cdfs(plt)
```



In [61]:
```
# 정규분포 누적 분포 함수의 역함수
def nomal_cdf(x,mu=0,sigma=1):
    return (1+math.erf((x-mu)/math.sqrt(2)/sigma))/2

def inverse_normal_cdf(p,mu=0,sigma=1,tolerance=0.00001):
    """find approximate inverse using binary search"""

    if mu!=0 or sigma!=1:
        return mu+sigma*inverse_normal_cdf(p,tolerance=tolerance)

    low_z,low_p=-10.0,0
    hi_z, hi_p =10.0,1
    while hi_z-low_z>tolerance:
        mid_z=(low_z+hi_z)/2
        mid_p=normal_cdf(mid_z)
        if mid_p<p:
```

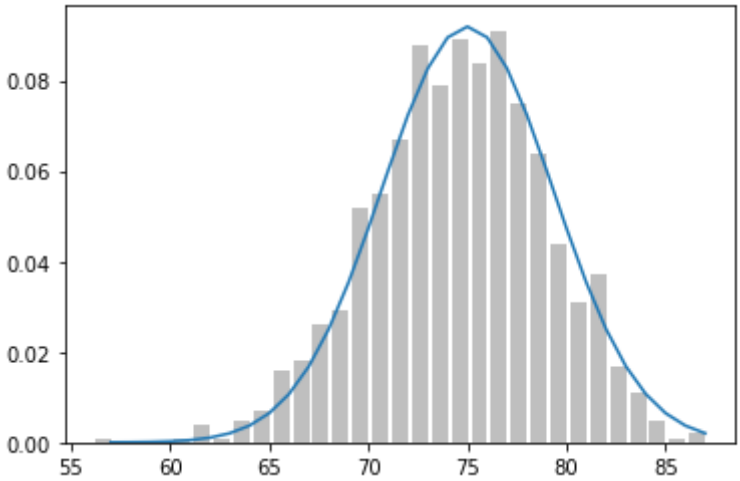
```
            low_z,low_p=mid_z,mid_p
        elif mid_p>p:
            hi_z,hi_p=mid_z,mid_p
        else:
            break

    return mid_z

np.vectorize(inverse_normal_cdf, otypes=[np.float])([0,0.5,0.90,0.95,0.975,1])
# 0%, 50%, 90%, 95%, 97.5%, 100%의 확률일경우 누적분포의 확률변수값
```

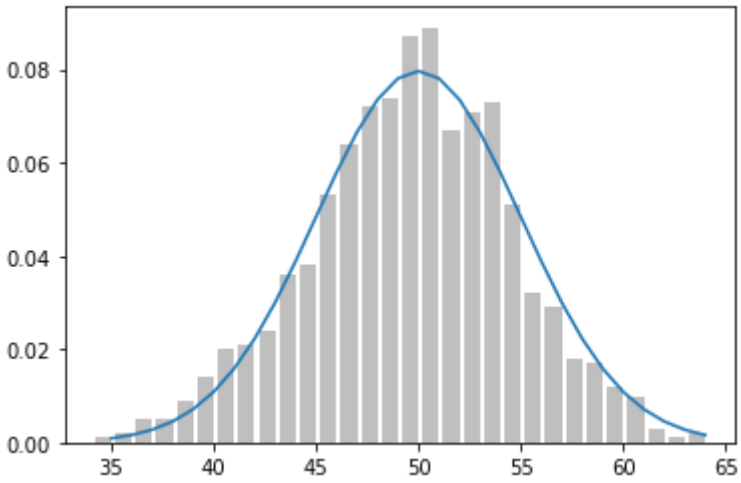
Out[61]:
```
array([-8.75      ,  0.        ,  1.28155708,  1.64484978,  1.95996284,
        8.75      ])
```

In [62]:
```
# 변형
np.vectorize(inverse_normal_cdf, otypes=[np.float])([0,0.25,0.5,0.75,0.975,1])
```

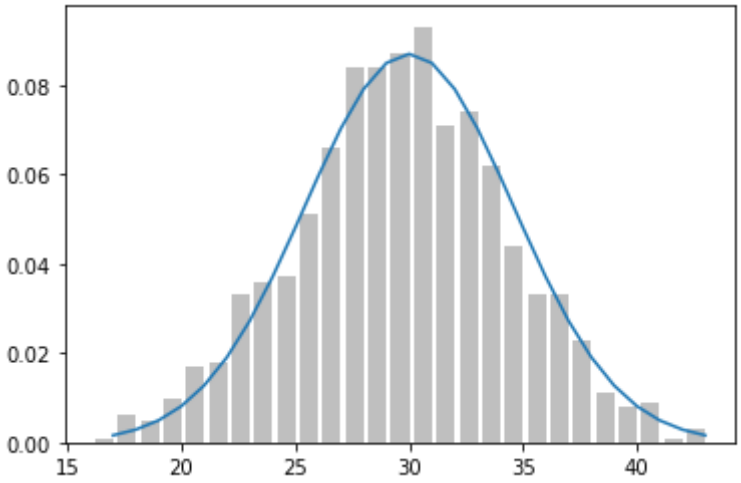
Out[62]:
```
array([-8.75      , -0.67448616,  0.        ,  0.67448616,  1.95996284,
        8.75      ])
```

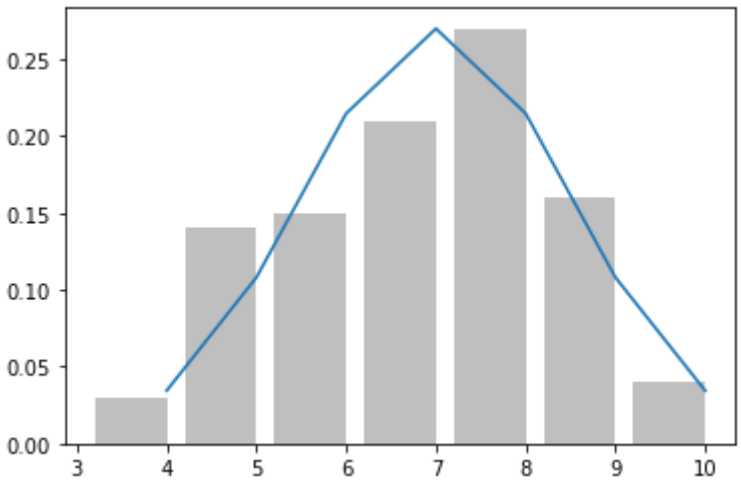
In [63]:
```
#중심극한정리
def bernoulli_trial(p):
    return 1 if random.random()<p else 0

def binomial(p,n):
    return sum([bernoulli_trial(p) for _ in range(n)])

def make_hist(p,n,num_points):

    data=[binomial(p,n) for _ in range(num_points)]

    histogram=Counter(data)
    plt.bar([x-0.4 for x in histogram.keys()],
            [v/num_points for v in histogram.values()],
            0.8,
            color='0.75')

    mu=p*n
    sigma=math.sqrt(n*p*(1-p))

    xs=range(min(data), max(data)+1)
    ys=[normal_cdf(i+0.5, mu, sigma)-normal_cdf(i-0.5,mu,sigma)
       for i in xs]
    plt.plot(xs,ys)
    plt.show()

make_hist(0.75,100,1000)
make_hist(0.50,100,1000)
```





In [64]:
```
#변형

make_hist(0.30,100,1000)
make_hist(0.70,10,100)
```





202001555 지은미