Random variables

A random variable X is a variable that can take on a given set of values, called the sample space (or the support) S_X, where the likelihood of the values in S_X is determined by the probability mass/density function.

$$P(X \in A) = \begin{cases} \sum_{x \in A} f(x), & \text{for discrete r.v. } X \\ \int_{A \cap S_X} f(x)dx, & \text{for continuous r.v. } X \end{cases}$$

Examples:

- Suppose $X \sim \text{Binomial}(n, p)$: X denotes the number of successes in a sequence of n independent experiments.
  - The support $S_X = \{0, 1, \ldots, n\}$.
  - The probability mass function

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x \in S_X.$$

- Suppose $X \sim N(\mu, \sigma^2)$.
  - The support $S_X = (-\infty, \infty)$.
  - The probability density function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad x \in S_X.$$

PDF/PMF

Take $X \sim N(\mu, \sigma^2)$ as an example, where $\mu = 1, \sigma^2 = 4$. We will use the function `norm.pdf()` to determine the value at a certain x.

In [1]:

```
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt

mu = 1
sigma = 2
x0 = 0.5
print('The pdf of N(1, 4) at x = 0.5 is %.3f' % norm.pdf(x0, loc = mu, scale = s
# %.4f: rounded to 4 decimal places
```

```
The pdf of N(1, 4) at x = 0.5 is 0.193
```

A more general expression:

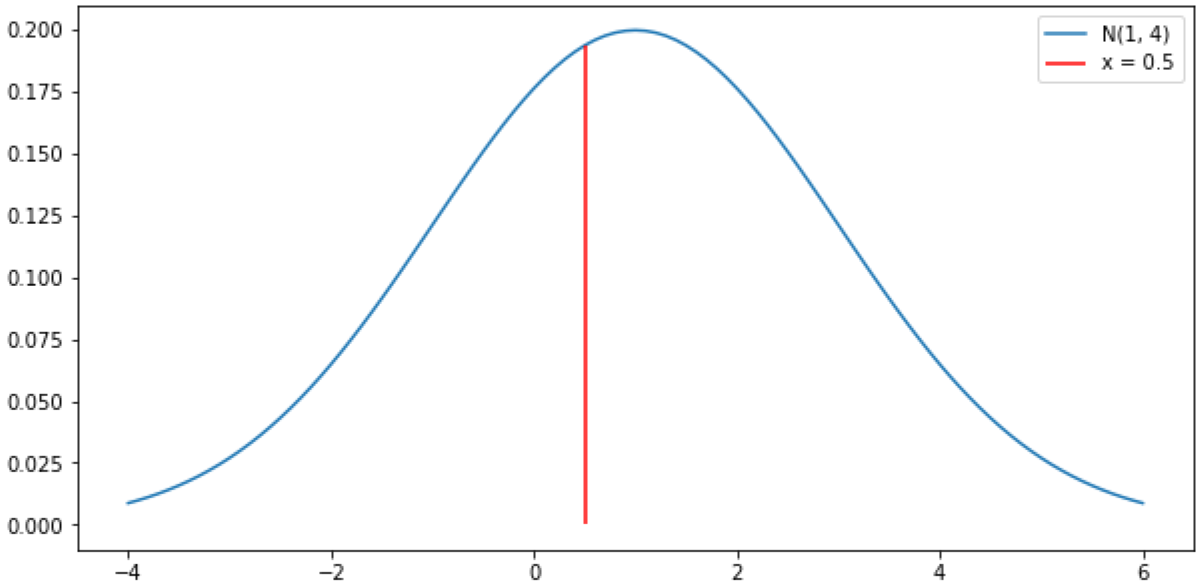
In [2]:

```
print('The pdf of N({}, {}) at x = {} is {:.3f}'.format(mu, sigma**2, x0, norm.p
```

```
The pdf of N(1, 4) at x = 0.5 is 0.193
```

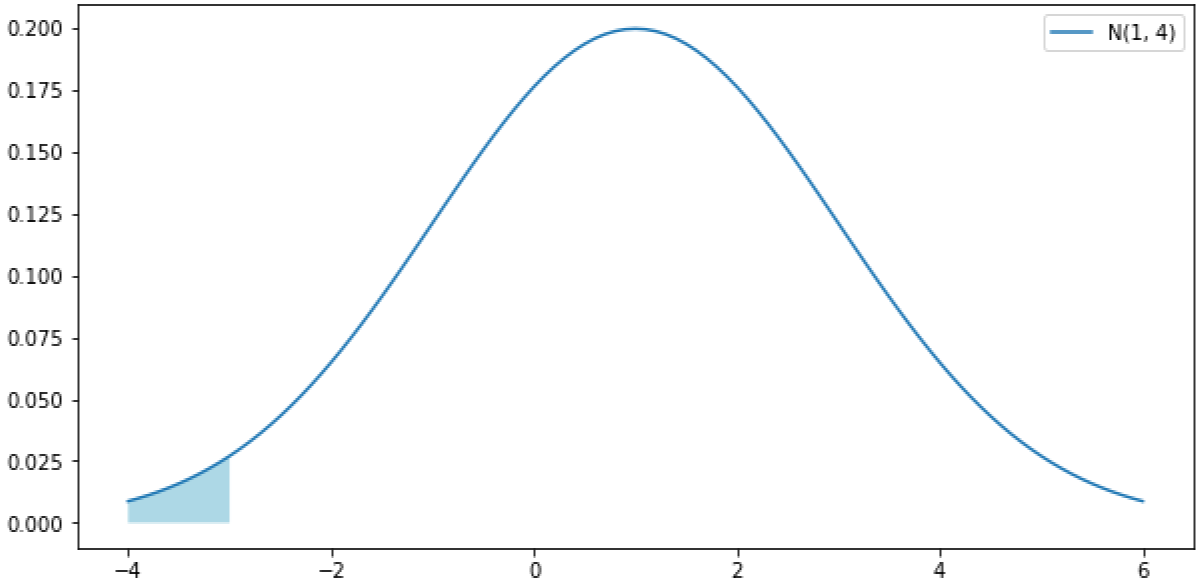
Next, we will depict the probability density curve.

In [3]:
```
x = np.linspace(-4, 6, 101)

plt.figure(figsize = (10,5))
plt.plot(x, norm.pdf(x, mu, sigma), label = 'N({}, {})'.format(mu, sigma ** 2))
plt.vlines(x = x0, ymin = 0, ymax = norm.pdf(x0, mu, sigma), color = 'red', labe
# add a vertical line at x = 0.5
plt.legend()
plt.show()
```



The probability $P(X \leq -3)$ can be computed via the function `norm.cdf()`

In [4]:
```
plt.figure(figsize = (10,5))
plt.plot(x, norm.pdf(x, mu, sigma), label = 'N({}, {})'.format(mu, sigma ** 2))
x_sub = x[x <= -3] # x values at most -3
plt.fill_between(x_sub, norm.pdf(x_sub, mu, sigma), 0, facecolor="lightblue")
# Shade the area between the pdf and line y=0 for x <= -3
plt.legend()
plt.show()
```



In [5]:

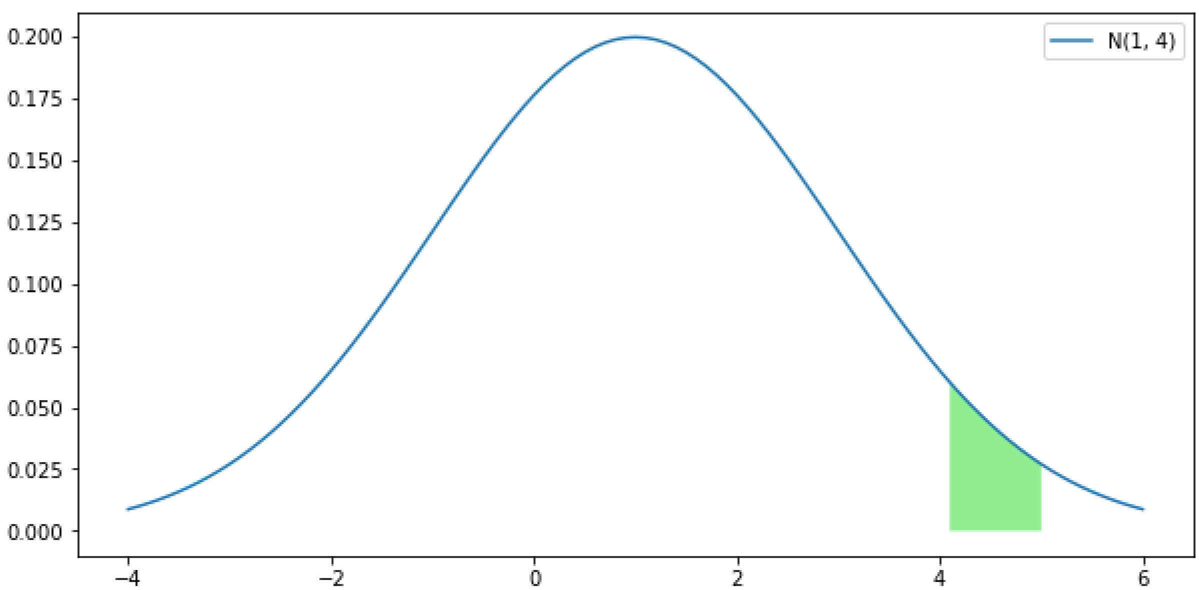
```
print('The probability that X <= -3 is %.3f' % norm.cdf(-3, loc = mu, scale = si
```

```
The probability that X <= -3 is 0.023
```

Similarly, we can compute the probability $P(4 < X \leq 5) = P(X \leq 5) - P(X \leq 4)$.

In [6]:

```
plt.figure(figsize = (10,5))
plt.plot(x, norm.pdf(x, mu, sigma), label = 'N({}, {})'.format(mu, sigma ** 2))
x_sub = x[np.where((x>4) & (x<=5))] # x values between 4 and 5
plt.fill_between(x_sub, norm.pdf(x_sub, mu, sigma), 0, facecolor="lightgreen")
# Shade the area between the pdf and line y=0 for 4 < x <= 5
plt.legend()
plt.show()

print('The probability that 4 < X <= 5 is %.3f' % (norm.cdf(5, mu, sigma) - norm
```




```
The probability that 4 < X <= 5 is 0.044
```

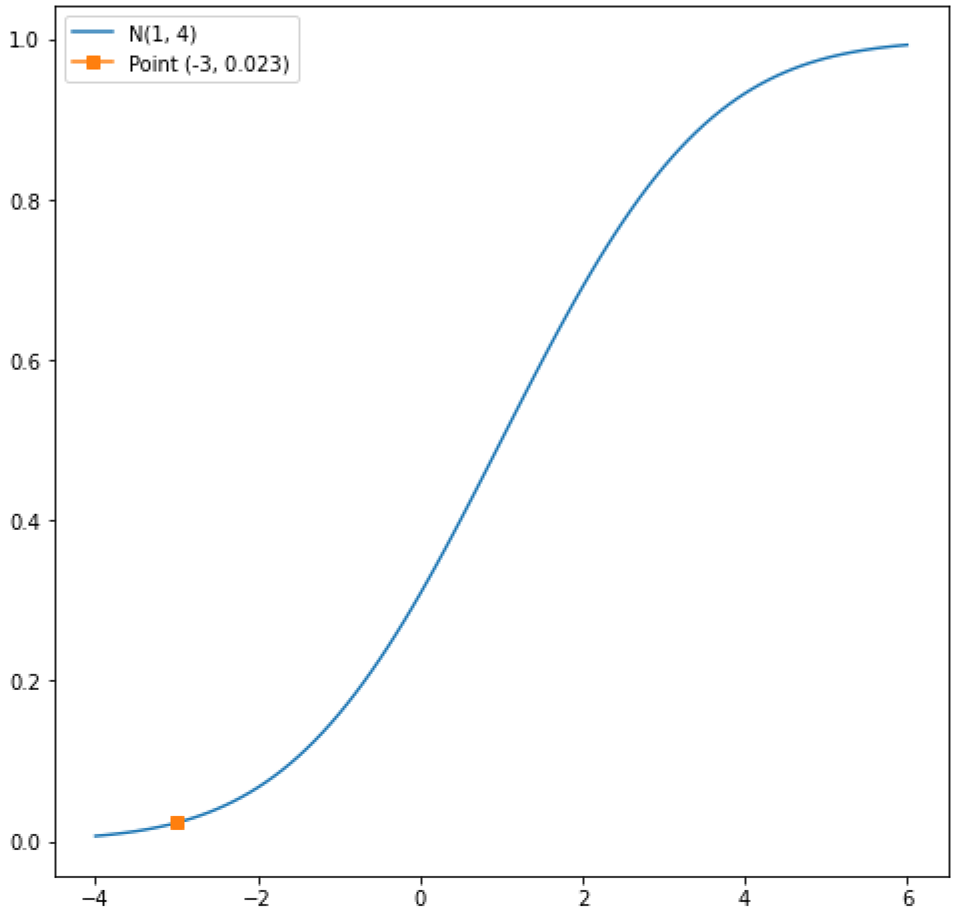
CDF

The cumulative distribution function $F_X(x)$ is given by

$$F_X(x) = P(X \leq x).$$

For the aforementioned X, the curve of CDF is:

In [7]:
```
plt.figure(figsize = (8,8))
plt.plot(x, norm.cdf(x, mu, sigma), label = 'N({}, {})'.format(mu, sigma ** 2))
plt.plot(-3, norm.cdf(-3, mu, sigma), marker='s', label = 'Point (-3, 0.023)')
plt.legend()
plt.show()
```



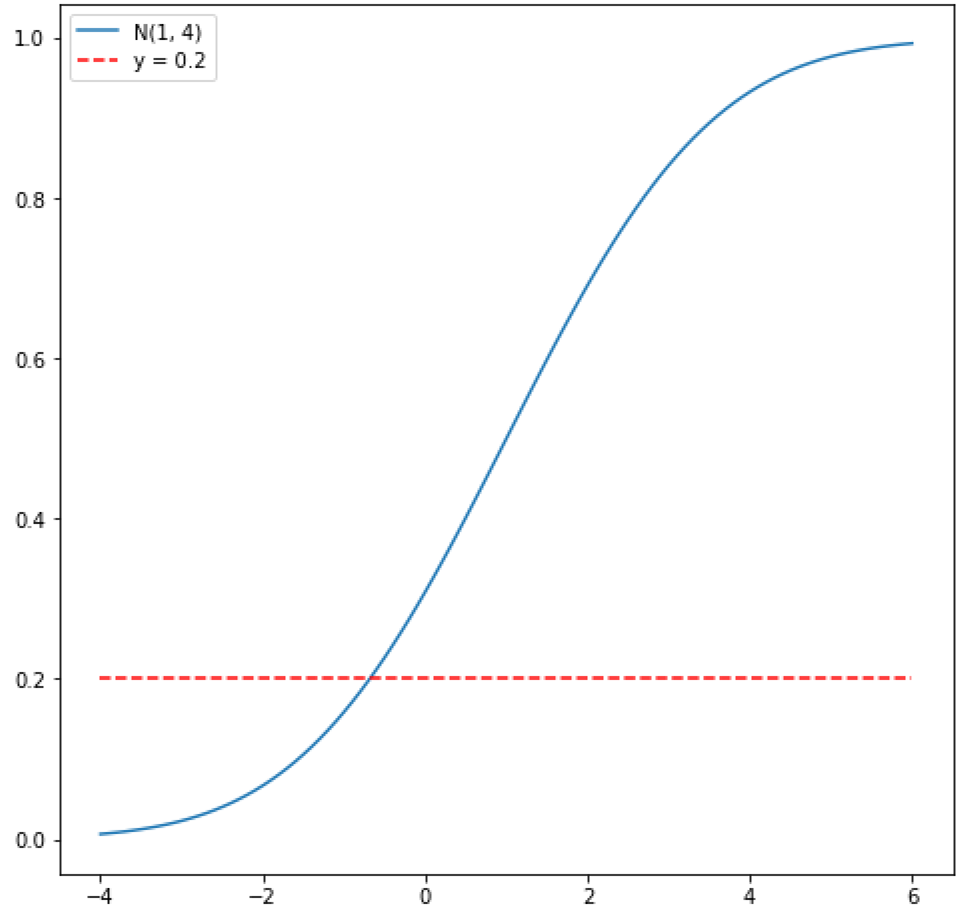
Quantile

The α-quantile of $F_X(\cdot)$ for $\alpha \in (0,1)$ is the value q_α such that

$$q_\alpha = \inf\{x \in \mathbb{R} : F_X(x) \geq \alpha\}.$$

For instance, given $\alpha = 0.2$, the quantile $q^X_{0.2}$ is shown below.

In [8]:
```
plt.figure(figsize = (8,8))
plt.plot(x, norm.cdf(x, mu, sigma), label = 'N({}, {})'.format(mu, sigma ** 2))
plt.hlines(y =  0.2, xmin = -4, xmax = 6, label = 'y = 0.2', linestyle = '--', c
plt.legend()
plt.show()
```



In [9]:
```
print("The 0.2-quantile of X is %.3f" % norm.ppf(0.2, mu, sigma))
# the x-coordinate of the point of intersection
```

```
The 0.2-quantile of X is -0.683
```

We have used `norm.pdf()`, `norm.cdf()` , and `norm.ppf()` so far. When we do monte carlo simulation next time, we will also use `norm.rvs()` .

Standard Normal Distribution

Let $Z \sim N(0,1)$. Its CDF is denoted by

$$\Phi(x) = F_Z(x) = P(Z \leq x).$$

Accordingly, its α-quantile q_α^Z is

$$q_\alpha^Z = \Phi^{-1}(\alpha).$$

Suppose $X \sim N(\mu, \sigma^2)$. For every $\alpha \in (0,1)$, we have

$$q_\alpha^X = \mu + \sigma q_\alpha^Z.$$

Proof.

$$\begin{aligned}\alpha &= F_X(q_\alpha^X) \\ &= P(X \le q_\alpha^X) \\ &= P\left(\frac{X-\mu}{\sigma} \le \frac{q_\alpha^X - \mu}{\sigma}\right) \\ &= P\left(Z \le \frac{q_\alpha^X - \mu}{\sigma}\right) \\ &= P(Z \le q_\alpha^Z),\end{aligned}$$

which implies

$$\frac{q_\alpha^X - \mu}{\sigma} = q_\alpha^Z \quad \text{or} \quad q_\alpha^X = \mu + \sigma q_\alpha^Z.$$

This identity is useful when we evaluate the value-at-risk given normally distributed simple returns.

An illustration:

In [10]:

```
alpha = 0.2
q_alpha_Z = norm.ppf(0.2, 0, 1)
print("The 0.2-quantile of X is: %.3f" % q_alpha_Z)
print('The right-hand side of the above equation is %.3f' % (mu + sigma * q_alph
# the same as the alpha-quantile of X
```

```
The 0.2-quantile of X is: -0.842
The right-hand side of the above equation is -0.683
```

Student's t-distribution

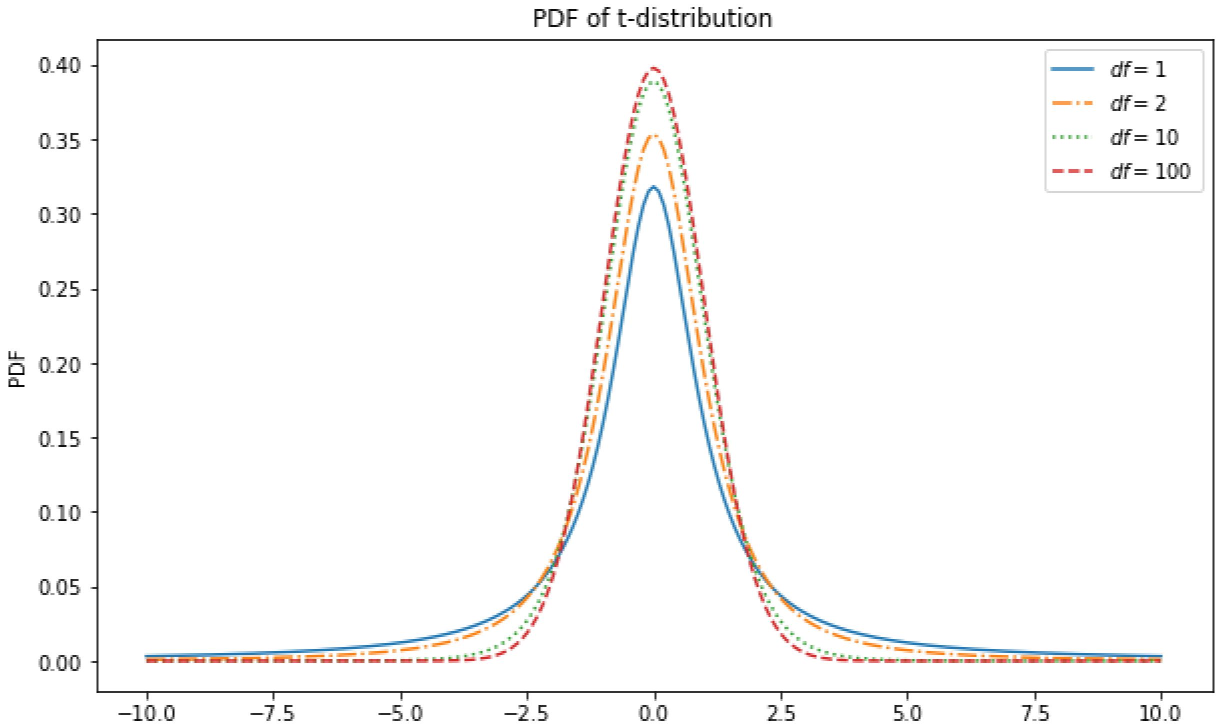
If Z is a standard normal random variable, U_v is a Chi-square random variable with degree of freedom v, and Z and U_v are independent, then

$$T = \frac{Z}{\sqrt{U_v/v}} \sim t(v)$$

is a Student-t random variable with degrees of freedom v.

In [11]:

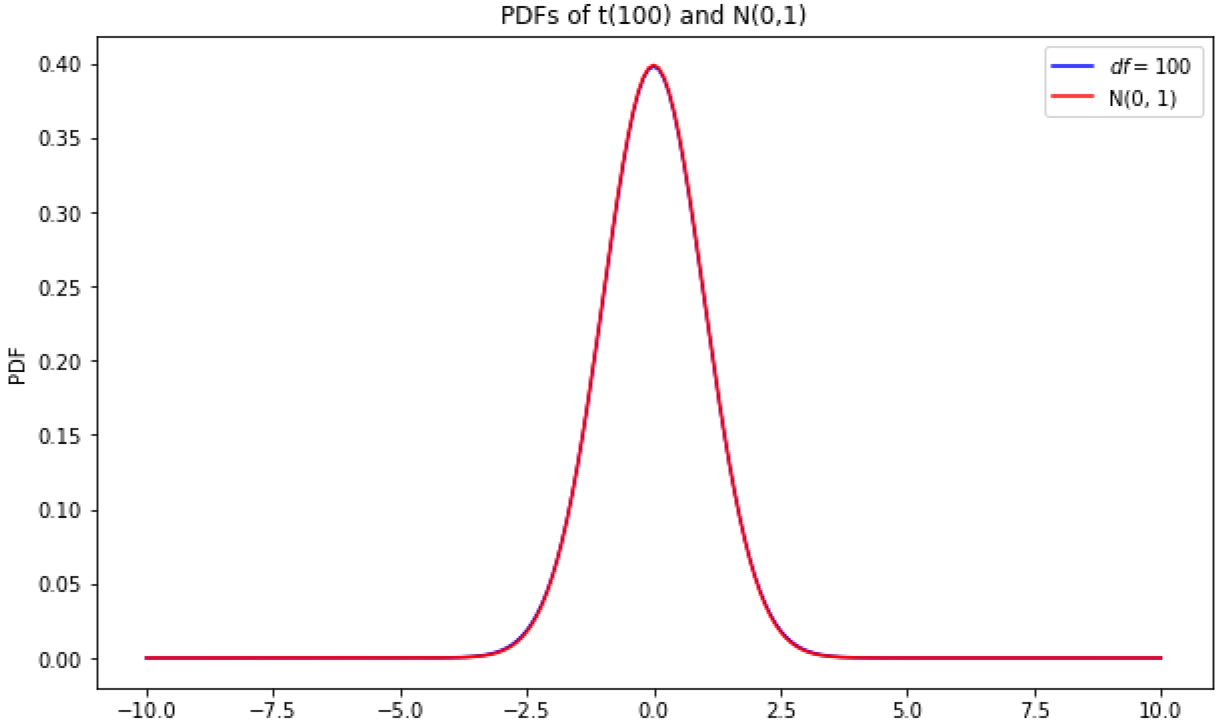
```
from scipy.stats import t

x = np.linspace(-10, 10, 201)
plt.figure(figsize=(10,6))
plt.plot(x, t.pdf(x, df = 1), label = '$df = 1$')
plt.plot(x, t.pdf(x, df = 2), label = '$df = 2$', linestyle='-.')
plt.plot(x, t.pdf(x, df = 10), label = '$df = 10$', linestyle=':')
plt.plot(x, t.pdf(x, df = 100), label = '$df = 100$', linestyle='--')
# plt.plot(x, norm.pdf(x, 0, 1), label = 'N(0, 1)', color = 'red')
plt.ylabel('PDF')
plt.legend()
plt.title('PDF of t-distribution')
plt.show()
```



We can view $N(0,1)$ as $t(\infty)$.

In [12]:

```
x = np.linspace(-10, 10, 201)
plt.figure(figsize=(10,6))
plt.plot(x, t.pdf(x, df = 100), label = '$df = 100$', color = 'blue')
plt.plot(x, norm.pdf(x, 0, 1), label = 'N(0, 1)', color = 'red')
plt.ylabel('PDF')
plt.legend()
plt.title('PDFs of t(100) and N(0,1)')
plt.show()
# almost the same
```



Student-t random variables have fatter tails than normal random variables.

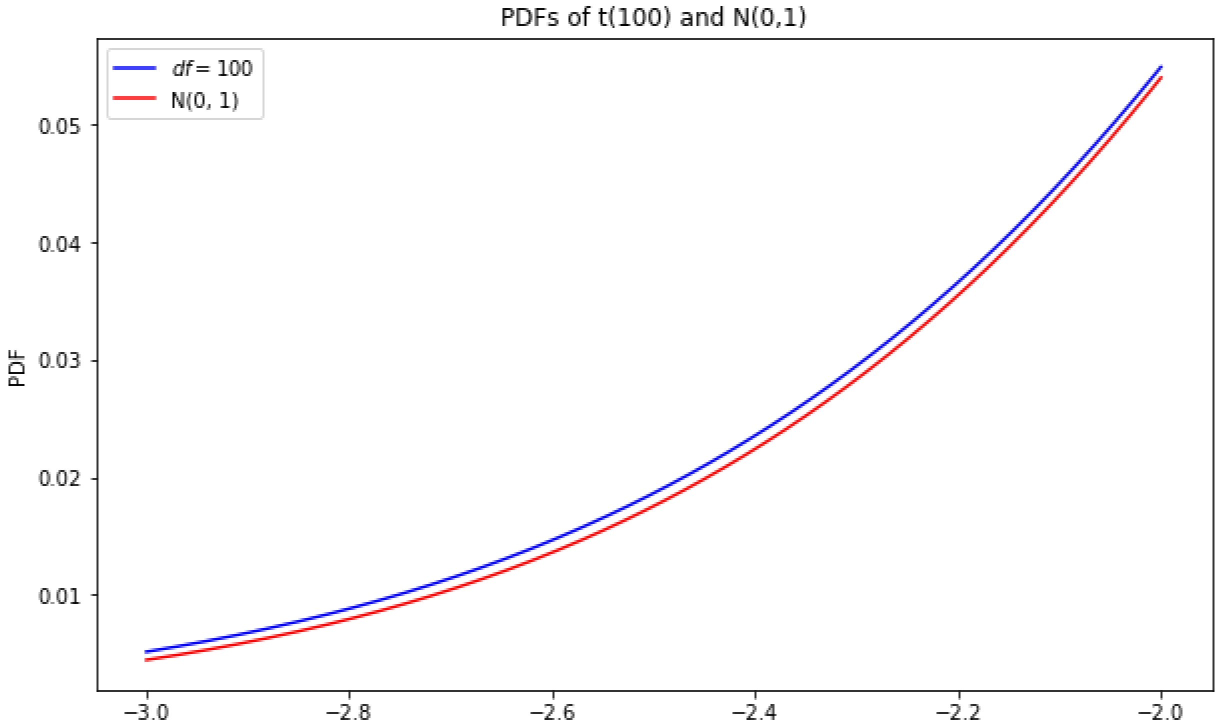
In [13]:

```
print(t.cdf(-2, df = 100)) # larger tail probability that T <= -2
print(norm.cdf(-2, 0, 1))
print("=============")
print(1 - t.cdf(3, df = 100)) # larger tail probability that T > 3
print(1 - norm.cdf(3, 0, 1))
```

```
0.02410608936556682
0.022750131948179195
=============
0.0017039576716647575
0.0013498980316301035
```

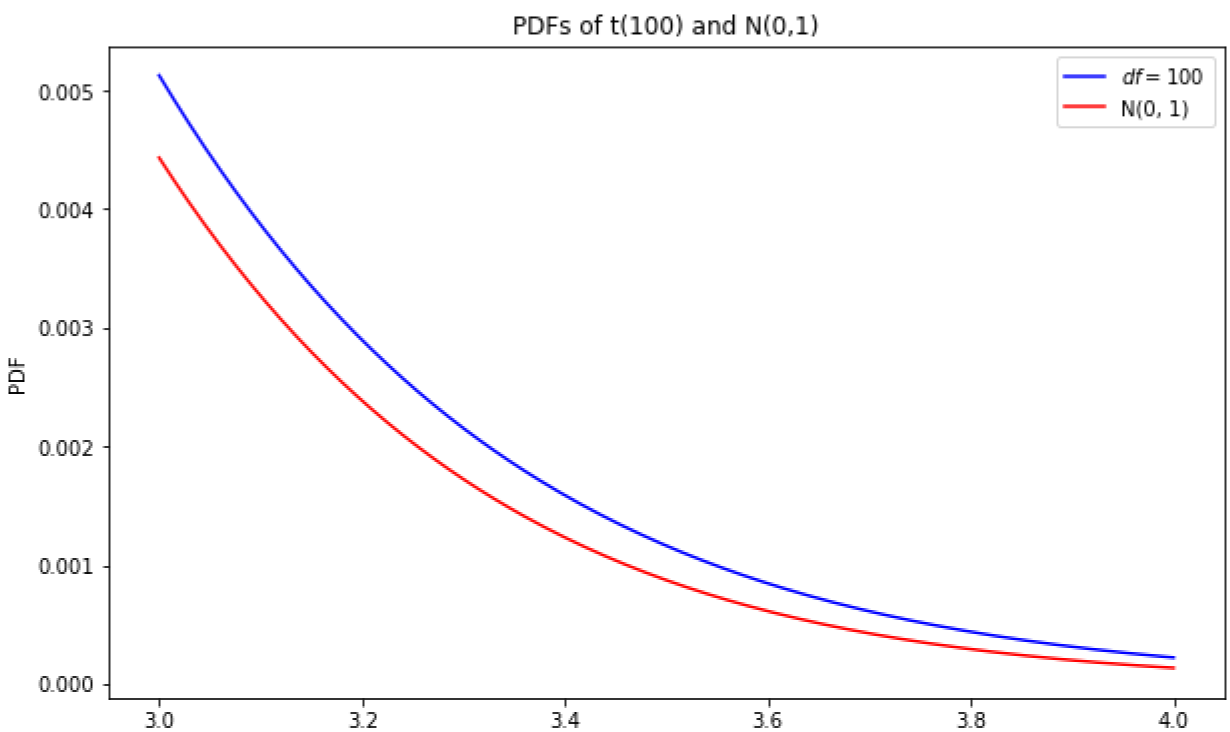
In [14]:

```
# consider a smaller area: -3 <= x <= -2
x = np.linspace(-3, -2, 201)
plt.figure(figsize=(10,6))
plt.plot(x, t.pdf(x, df = 100), label = '$df = 100$', color = 'blue')
plt.plot(x, norm.pdf(x, 0, 1), label = 'N(0, 1)', color = 'red')
plt.ylabel('PDF')
plt.legend()
plt.title('PDFs of t(100) and N(0,1)')
plt.show()
# for any given x, P(X <= x) < P(T <= x) for X \sim N(0, 1) and T \sim t(100)
```



In [15]:

```
# consider a smaller area: 3 <= x <= 4
x = np.linspace(3, 4, 201)
plt.figure(figsize=(10,6))
plt.plot(x, t.pdf(x, df = 100), label = '$df = 100$', color = 'blue')
plt.plot(x, norm.pdf(x, 0, 1), label = 'N(0, 1)', color = 'red')
plt.ylabel('PDF')
plt.legend()
plt.title('PDFs of t(100) and N(0,1)')
plt.show()
# for any given x, P(X > x) < P(T > x) for X \sim N(0, 1) and T \sim t(100)
```



Similarly, for $\alpha \leq 0.05$, the α-quantiles of $X \sim N(0,1)$ are larger than those of $T \sim t(100)$.

In [16]:
```
alpha_seq = np.linspace(0.01, 0.10, 11)
quantile_X = norm.ppf(alpha_seq)
quantile_T = t.ppf(alpha_seq, df = 100)
np.c_[quantile_X, quantile_T]
```

Out[16]:
```
array([[-2.32634787, -2.36421737],
       [-2.07485473, -2.10273178],
       [-1.91103565, -1.93351855],
       [-1.78661337, -1.805534  ],
       [-1.68494077, -1.70126894],
       [-1.59819314, -1.61252215],
       [-1.52203624, -1.5347629 ],
       [-1.45380636, -1.46521271],
       [-1.39174378, -1.40203936],
       [-1.33462229, -1.34396814],
       [-1.28155157, -1.29007476]])
```

Random vectors

Consider the random vector

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}\right).$$

Suppose that $\mu_1 = \mu_2 = 0, \sigma_1^2 = 1, \sigma_2^2 = 4.$

- The marginal distributions are also normal distributions
- The correlation ρ measures the strength of association between two variables and the direction of the relationship

In [17]:
```
from scipy.stats import multivariate_normal

cov_val = [-0.8, 0, 0.8]

# mean of the bivaraite normal
mu = np.array([0,0])
sigma = np.array([1, 4])
rho = 0
# covariance matrix
cov = np.array([[sigma[0], rho * np.sqrt(sigma[0] * sigma[1])], [rho * np.sqrt(s
cov
```

Out[17]:
```
array([[1., 0.],
       [0., 4.]])
```

In [18]:
```
np.array([[1, 0], [0, 4]]) # a simpler form
```

Out[18]:
```
array([[1, 0],
       [0, 4]])
```

In [19]:

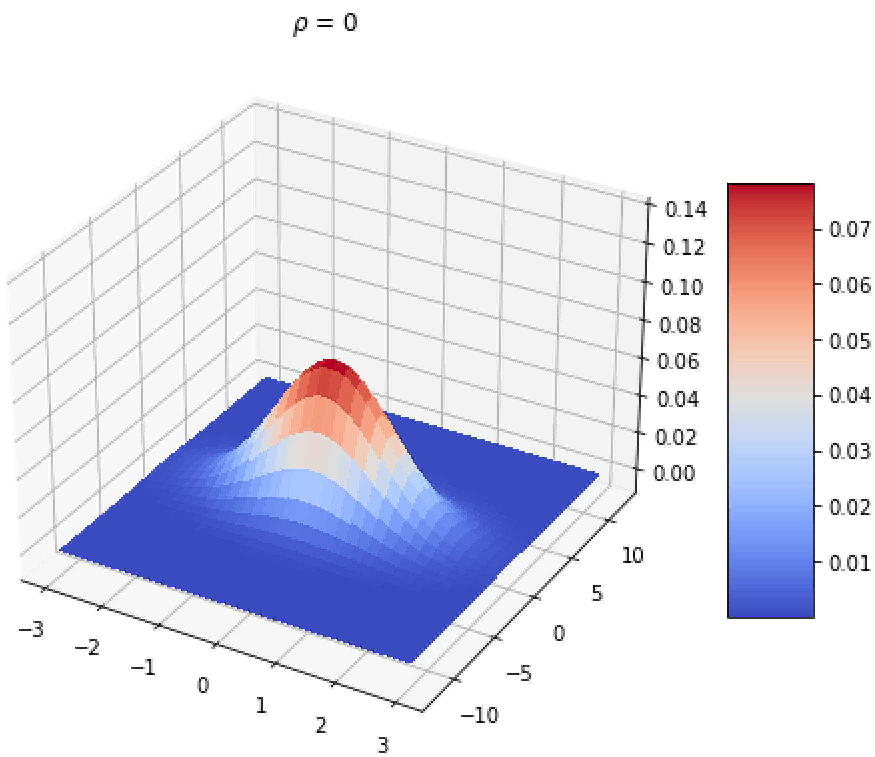
```
N = 100
x = np.linspace(-3*sigma[0], 3*sigma[0], num=N+1)
y = np.linspace(-3*sigma[1], 3*sigma[1], num=N+1)
X, Y = np.meshgrid(x,y)
# np.meshgrid function is used to create a rectangular grid out of two given one
print(X[0:5, 0:5])

pdf = np.zeros(X.shape) # a 101 time 101 matrix
distr = multivariate_normal(mean = mu, cov = cov)
for i in range(N+1):
    for j in range(N+1):
        pdf[i, j] = distr.pdf([X[i,j], Y[i,j]])
```

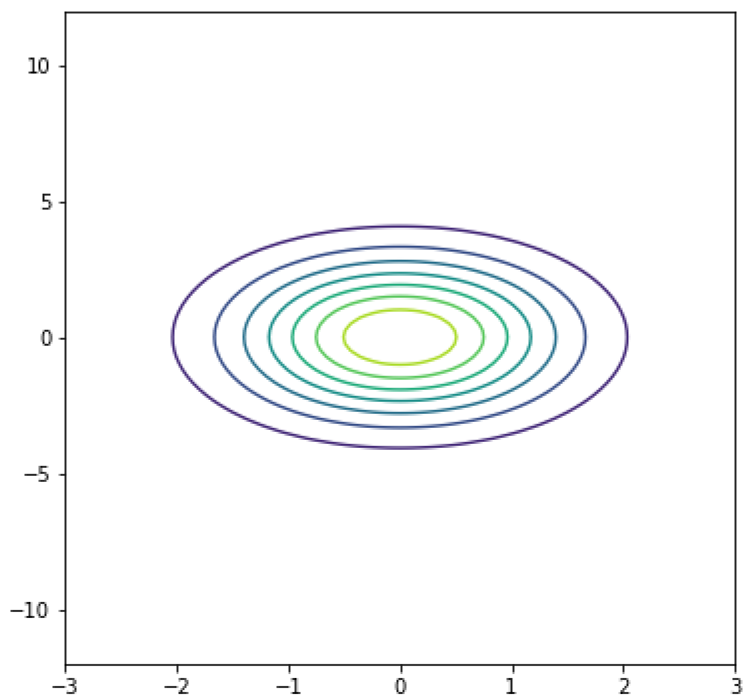
```
[[-3.   -2.94 -2.88 -2.82 -2.76]
 [-3.   -2.94 -2.88 -2.82 -2.76]
 [-3.   -2.94 -2.88 -2.82 -2.76]
 [-3.   -2.94 -2.88 -2.82 -2.76]
 [-3.   -2.94 -2.88 -2.82 -2.76]]
```

To have similar graphs associated with different values of correlation coefficients, we can define a function of ρ.

In [20]:

```
def bivariate_norm(rho, mu = mu, sigma = sigma):
    cov = np.array([[sigma[0], rho * np.sqrt(sigma[0] * sigma[1])], [rho * np.sq
    N = 100
    x = np.linspace(-3*sigma[0], 3*sigma[0], num=N+1)
    y = np.linspace(-3*sigma[1], 3*sigma[1], num=N+1)
    X, Y = np.meshgrid(x,y)

    pdf = np.zeros(X.shape) # a 101 time 101 matrix
    distr = multivariate_normal(mean = mu, cov = cov)
    for i in range(N+1):
        for j in range(N+1):
            pdf[i, j] = distr.pdf([X[i,j], Y[i,j]])
    return X, Y, pdf

from matplotlib import cm

fig = plt.figure(figsize = (8, 8))
ax = fig.add_subplot(111, projection='3d') #nrows, ncols, index
surf = ax.plot_surface(X, Y, pdf, cmap=cm.coolwarm, linewidth=0, antialiased=Fal
# Customize the z axis.
ax.set_zlim(-0.01, 0.14)
ax.zaxis.set_major_formatter('{x:.02f}')

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.title(r'$\rho$ = {}'.format(rho))
plt.show()

fig = plt.figure(figsize = (6, 6))
ax2 = fig.add_subplot(111) #nrows, ncols, index
ax2.contour(X, Y, pdf)
plt.show()
```





We further put together the code inside a function depending on ρ.

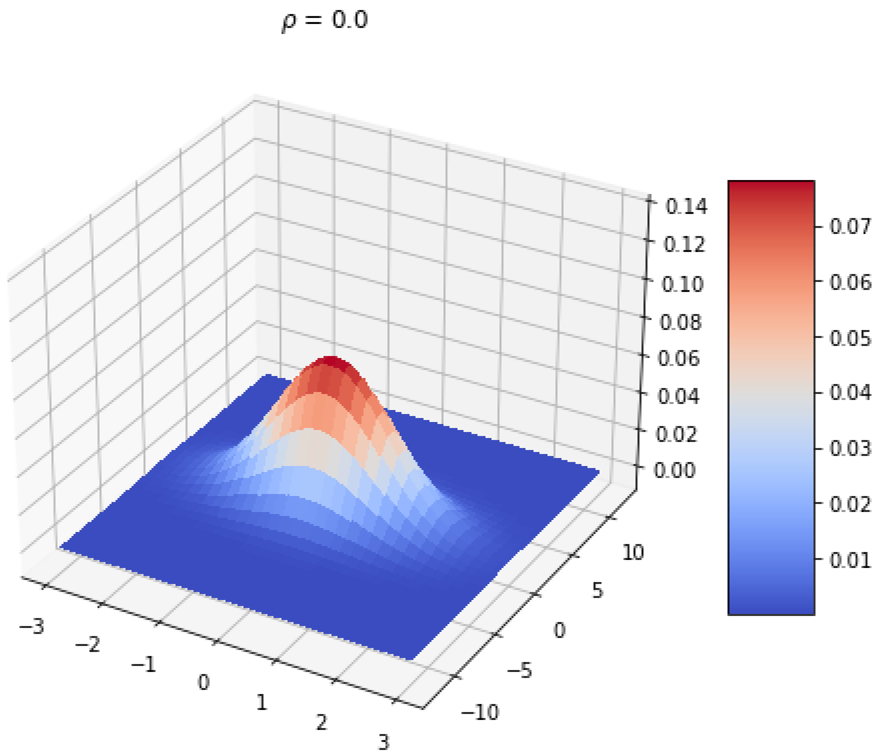
In [21]:

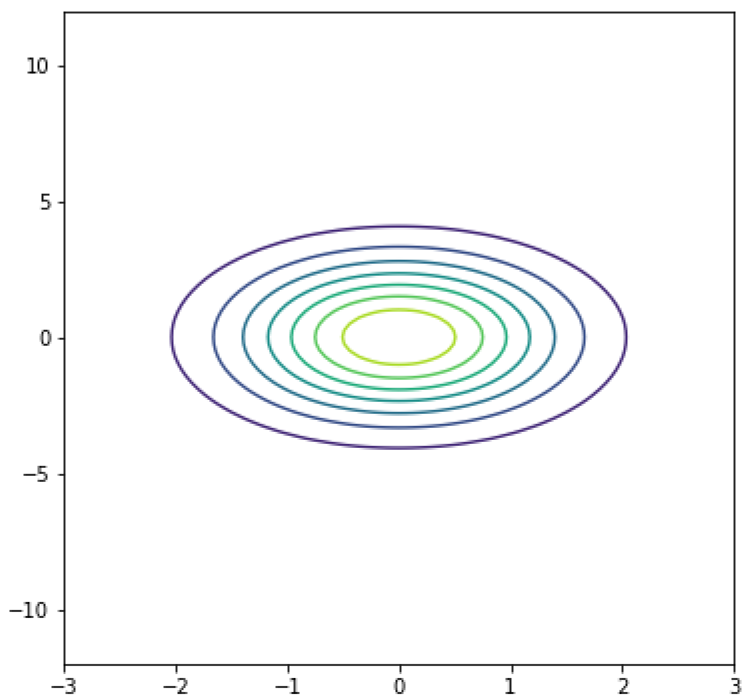
```
def plot(rho, X, Y, pdf):
    fig = plt.figure(figsize = (8, 8))
    ax = fig.add_subplot(111, projection='3d') #nrows, ncols, index
    surf = ax.plot_surface(X, Y, pdf, cmap=cm.coolwarm, linewidth=0, antialiased
    # Customize the z axis.
    ax.set_zlim(-0.01, 0.14)
    ax.zaxis.set_major_formatter('{x:.02f}')

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

```
    plt.title(r'$\rho$ = {}'.format(rho))
    plt.show()

    fig = plt.figure(figsize = (6, 6))
    ax2 = fig.add_subplot(111) #nrows, ncols, index
    ax2.contour(X, Y, pdf)
    plt.show()
```

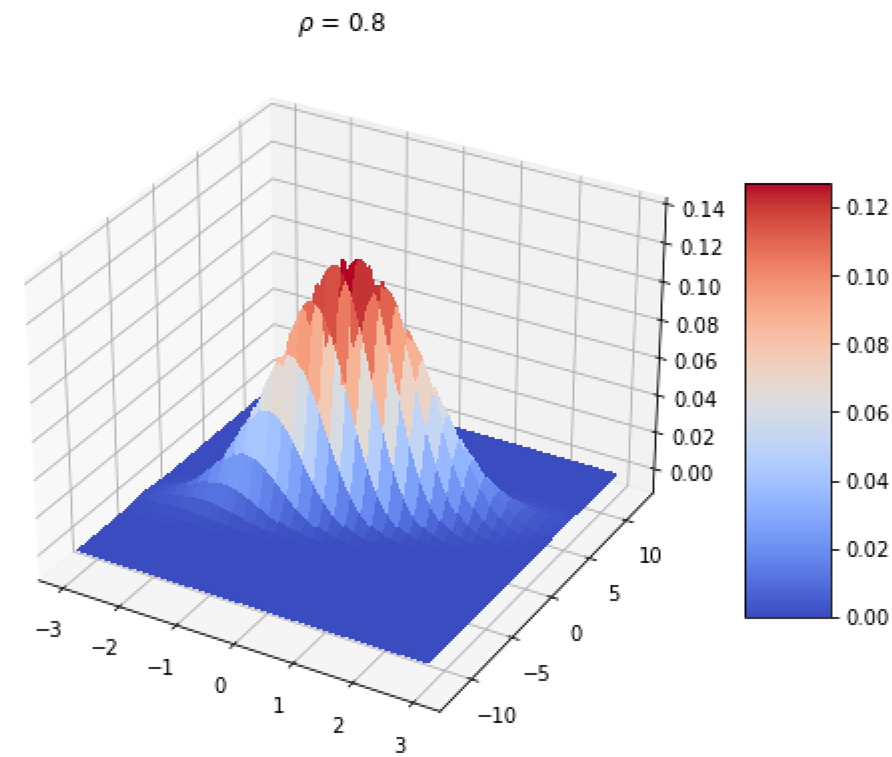
In [22]:

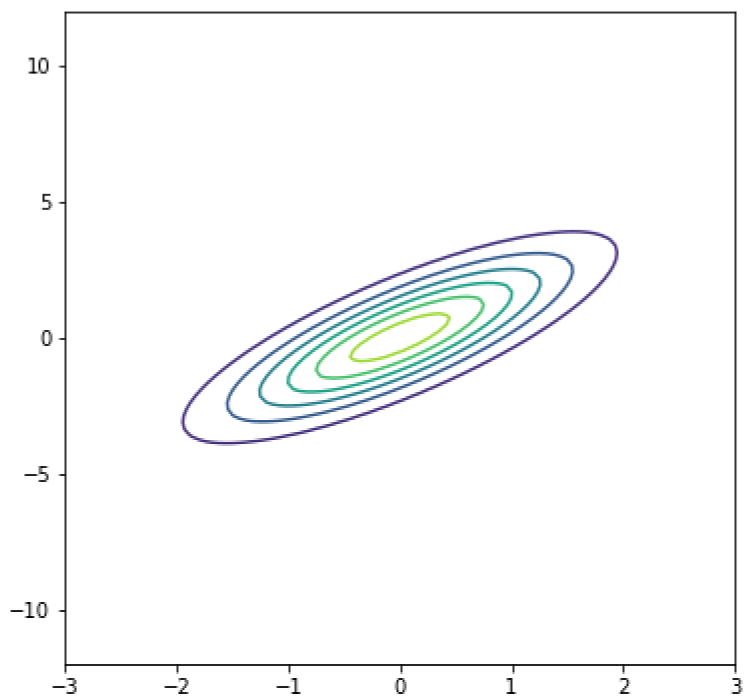
```
rho = 0.0
X, Y, pdf = bivariate_norm(rho)

plot(rho, X, Y, pdf)
```





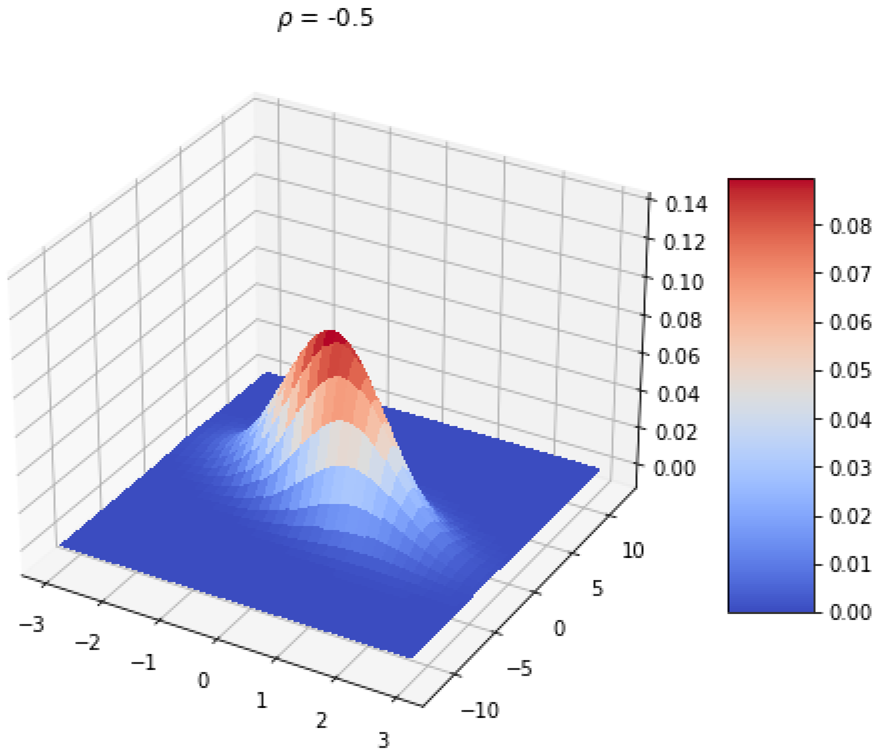
In [23]:
```
rho = 0.8
X, Y, pdf = bivariate_norm(rho)

plot(rho, X, Y, pdf)
```

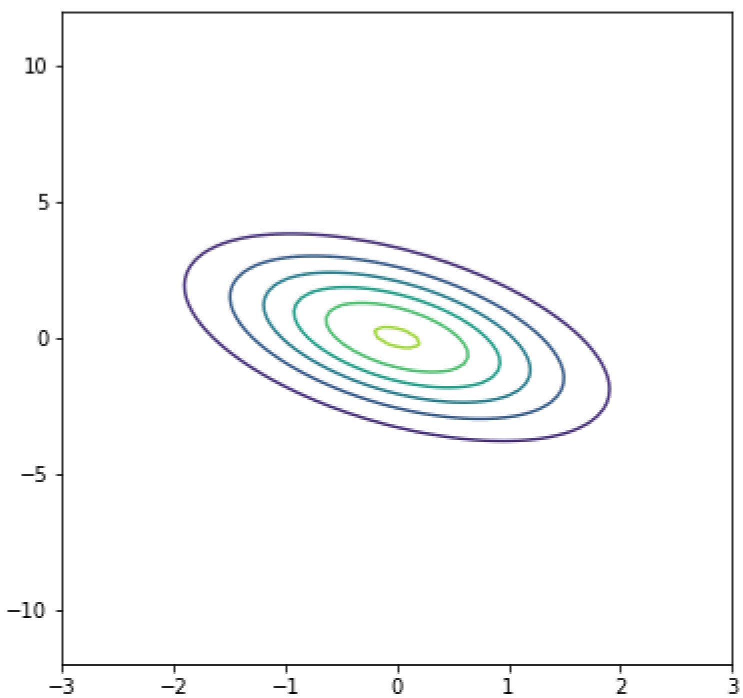




In [24]:

```
rho = -0.5
X, Y, pdf = bivariate_norm(rho)

plot(rho, X, Y, pdf)
```





Suppose that we are interested in $P(-1 \le X_1 \le 1, -2 \le X_2 \le 3)$, where

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0.5 \\ 0.5 & 4 \end{pmatrix}\right).$$

Note that

$$\begin{aligned} P(-1 \le X_1 \le 1, -2 \le X_2 \le 3) = P(X_1 \le 1, X_2 \le 3) - P(X_1 \le -1, X_2 \le 3) \\ - P(X_1 \le 1, X_2 \le -2) + P(X_1 \le -1, X_2 \le -2) \end{aligned}$$

In [25]:

```
mu = np.array([0, 0])
cov = np.array([[1, 0.5], [0.5, 4]])
diff = multivariate_normal.cdf(np.array([1,3]), mean = mu, cov = cov) \
    - multivariate_normal.cdf(np.array([-1,3]), mean = mu, cov = cov) \
    - multivariate_normal.cdf(np.array([1,-2]), mean = mu, cov = cov) \
    + multivariate_normal.cdf(np.array([-1,-2]), mean = mu, cov = cov)
print("The target probability:", diff)
```

```
The target probability: 0.5354662279197304
```