Problem 1: Noisy Phase-Shifted Sine Waves

Generate two noisy sine waves that represent the levels of two different hormones over time. Each sine wave should have:

- 1000 points.
- A frequency of 1 cycle in 24 hrs.
- A noise level that is normally distributed with a mean of 0 and a standard deviation of 0.1.
- A phase shift for the second hormone, representing a delay in its cycle compared to the first.

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
```

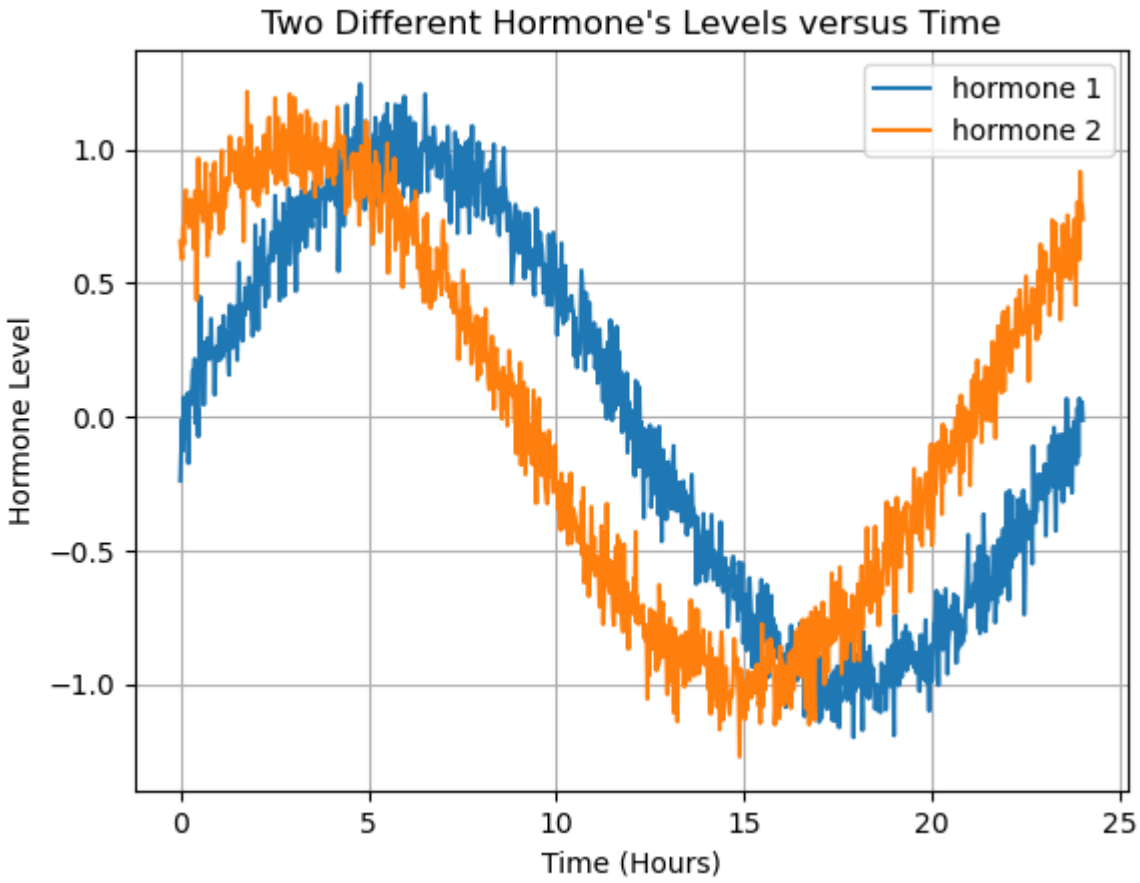
In [18]:
```
# Time vector representing 24 hours
time = np.linspace(0, 24, n_points)

#first sine wave
hormone1 = np.sin(2 * np.pi / 24 * time) + np.random.normal(0, .1, 1000)

#second sine
phase_shift = np.pi / 4
hormone2 = np.sin(2 * np.pi / 24 * time + phase_shift) + np.random.normal(0, .
```

Plot the two sine waves on the same graph over a period of 10 days.

- The x-axis should represent time in hours or days.
- The y-axis should represent the hormone level.
- Include a legend to differentiate between the two hormones.
- Title the graph appropriately.

Ensure your plot has gridlines for better readability.

In [23]:
```
plt.plot(time, hormone1, label = 'hormone 1')
plt.plot(time, hormone2, label = 'hormone 2')
plt.xlabel('Time (Hours)')
plt.ylabel('Hormone Level')
plt.title('Two Different Hormone\'s Levels versus Time')
plt.legend()
plt.grid()
```



Problem 2: Correlated Data and Marginal Distributions

Create a dataset containing 500 pairs of x and y values that are linearly correlated with:

- A correlation coefficient approximately equal to 0.8.
- Normally distributed residuals with a mean of 0 and standard deviation of 1

In [44]:
```
x_mu  = 0 # we set up the mean of the first set of data points
y_mu  = 0 # we set up the mean of the second sample
x_var = 1 # the variance of the first sample
y_var = 1 # the variance of the second sample
cov   = 0.8 # this is the covariance (can be thought of as correlation)
r     = 0.8 # correlation between the two datasets

# we now create the two correlated data sets
x = x_mu + x_var*np.random.randn(500,1)
z = y_mu + y_var*np.random.randn(500,1)

# The following is a little bit of magic that simply helps us create well beha
# correlated datasets with a specific correlation (y).
# It is fine not to completelly follow this step, in the future, we might expl
y = r*x + ((1-r**2)**0.5)*z
```

Generate a scatterplot of the x and y values.

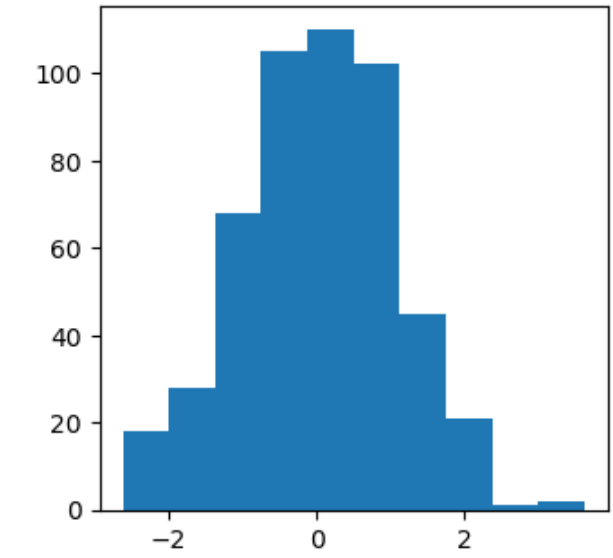
- Label the x-axis as "Variable X".
- Label the y-axis as "Variable Y".

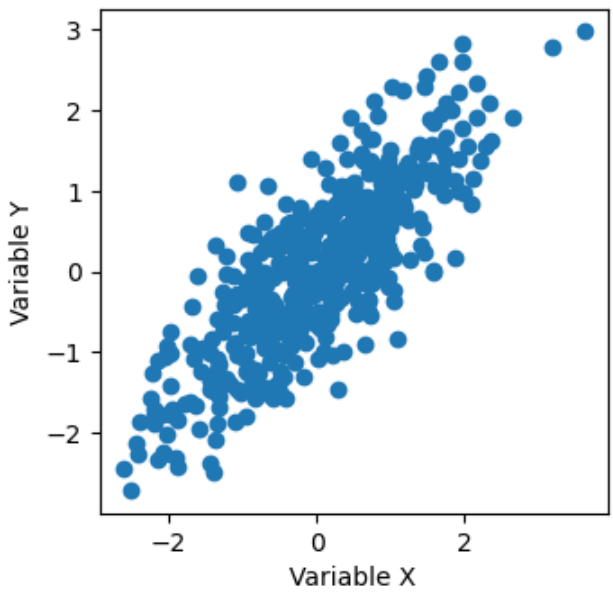
Alongside the scatterplot, create two histograms representing the marginal distributions of x and y.

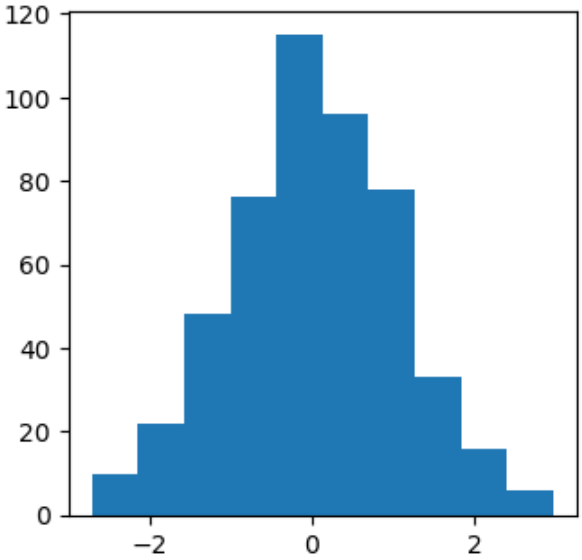
- Position the histogram of x values above the scatterplot.
- Position the histogram of y values to the right of the scatterplot.
- Use subplots to create a clean layout.
- Ensure all plots share their respective axes (x with x, and y with y).

In [51]:
```
plt.figure(figsize=(8, 8))


plt.subplot(223)
plt.scatter(x,y)
plt.xlabel('Variable X')
plt.ylabel('Variable Y')

plt.subplot(221)
plt.hist(x)

plt.subplot(224)
plt.hist(y)
```

Out[51]:
```
(array([ 10.,  22.,  48.,  76., 115.,  96.,  78.,  33.,  16.,   6.]),
 array([-2.71494867, -2.14562089, -1.57629312, -1.00696534, -0.43763757,
         0.13169021,  0.70101798,  1.27034576,  1.83967353,  2.40900131,
         2.97832908]),
 <BarContainer object of 10 artists>)
```


100
80
60
40
20
0
−2
0
2


3
2
1
0
−1
−2
Variable Y
−2
0
2
Variable X


120
100
80
60
40
20
0
−2
0
2