```
import numpy as np
import scipy.stats

# Set a random seed for reproducibility
np.random.seed(0)

# Define ensemble spread in state space
# We have two dimensions:
#   The first dimension is observed
mu1   = 0
sd1   = 0.05
#   The second dimension is hidden
mu2   = 4129  # Does not really matter
sd2   = 1.00     # !!! Adjust this value as required !!!

# Now define the covariance matrix
# let's assume both values are positively correlate with a factor of 0.75
ensemble_cov = np.asarray(
      [[1*sd1*sd1,    0.75*sd1*sd2],
       [0.75*sd2*sd1, 1*sd2*sd2    ]])

# Now draw the ensemble
ensemble_size   = 32
samples     = scipy.stats.multivariate_normal.rvs(
      mean    = [mu1,mu2],
      cov     = ensemble_cov,
      size    = ensemble_size)

# Define the observed state
y      = np.asarray([0.3]) # This lies well outside of the ensemble spread of the first state

# Define the state observation covariance matrix
R      = np.asarray([0.1]).reshape((1,1))

# Define the data extraction matrix
H      = np.asarray([1,0]).reshape((1,2))

# Now estimate the covariance matrix from the ensemble
cov_estimated = np.cov(samples.T)

# If you want to test the effect of no correlation, uncomment the part below
#cov_estimated[1,0] = 0
#cov_estimated[0,1] = 0

# Calculate the Kalman gain
K   = np.linalg.multi_dot((
      cov_estimated,
      H.T,
      np.linalg.inv(
         np.linalg.multi_dot((
            H,
            cov_estimated,
            H.T)) + R)))
```

```
# Apply the update step
samples_updated = np.zeros(samples.shape)

for sample in range(ensemble_size):
    samples_updated[sample,:] = samples[sample,:] + np.dot(K, y -
np.dot(H,samples[sample].reshape((2,1))))[:,0]

print('Mean of update for observed variable: '+str(np.mean(np.sqrt((samples_updated[:,0]-
samples[:,0])**2))))
print('Mean of update for hidden variable  : '+str(np.mean(np.sqrt((samples_updated[:,1]-
samples[:,1])**2))))
```