```python
#Created April 29 3:16 PM
@author: markanthonyviray

import numpy as np

import matplotlib.pyplot as plt

# Generate synthetic data
np.random.seed(0)
n = 100
x = np.linspace(0, 10, n)
true_slope = 2
true_intercept = 1
y = true_slope * x + true_intercept + np.random.normal(0, 1, n)

# Bayesian linear regression (shortest approach)
X = np.vstack([np.ones(n), x]).T  # Design matrix
y = y[:, np.newaxis]  # Reshape y to be a column vector

# Compute posterior parameters analytically
XTX = np.dot(X.T, X)
XTy = np.dot(X.T, y)
posterior_cov = np.linalg.inv(XTX + np.eye(2))  # Using a small identity matrix for regularization
posterior_mean = np.dot(posterior_cov, XTy)

# Generate posterior predictive samples
num_samples = 1000
beta_samples, alpha_samples = np.random.multivariate_normal(posterior_mean.flatten(), posterior_cov,
y_pred_samples = np.outer(beta_samples, x) + alpha_samples[:, np.newaxis]

# Plot observed data and posterior predictive distribution
plt.figure(figsize=(10, 6))
plt.scatter(x, y.flatten(), label='Observed data')
plt.plot(x, true_slope * x + true_intercept, label='True regression line', color='red', linestyle='--
plt.plot(x, posterior_mean[1] * x + posterior_mean[0], label='Posterior mean regression line', color=
plt.fill_between(x, np.percentile(y_pred_samples, 2.5, axis=0), np.percentile(y_pred_samples, 97.5, a
plt.xlabel('x')
plt.ylabel('y')
plt.title('Bayesian Linear Regression (Shortest Approach)')
plt.legend()
plt.show()
```