```python
import numpy as np
import random

#Theoretical notes and considerations
#X is my conditional distribution
#Y is the probability of each Gaussian
#X | Y is one of the Gaussians

#Class to model the conditional Gaussian in our example
class ConditionalGaussian:

    #List of means and variances, the parameters of our Gaussians
    #List of probabilities of the random variables
    def __init__(self, means, variances, probabilities):
        if(sum(probabilities) != 1):
            raise ValueError("Probabilities do not respect probabilities axioms")

        if(len(probabilities) != len(means) & len(means) != len(variances) & len(probabilities) != len(variances)):
            raise ValueError("Not ok data")

        self.means = means
        self.variances = variances
        self.probabilities = probabilities

    #E[X] = mean_1 * probability_1 + mean_2 * probability_2 + ... + mean_n * probability_n
    def get_conditional_mean(self):
        return np.dot(self.means, self.probabilities)

    #Var(X) = E[Var(X|Y)] + Var(E[X|Y])
    def get_conditional_variance(self):
        #E[Var(X|Y)] = variance_1 * probability_1 + variance_2 * probability_2 + ... + variance_n * probability_n
        expected_variance = np.dot(self.variances, self.probabilities)

        #Let's call it mu
        computed_mean = self.get_conditional_mean()

        #Var(E[X|Y]) = (mean_1-mu)^2 * probability_1 + (mean_2-mu)^2 * probability_2 + ... + (mean_n-mu)^2 * probability_n
        variance_of_expectation = np.dot(list(map(lambda x: (x-computed_mean)**2, self.means)), self.probabilities)

        return expected_variance + variance_of_expectation

    def sample(self):

        #We use random.choices to choose from which Gaussian to sample.
        #We extract one specific Gaussian basing on the probability.
        index = random.choices(range(len(self.probabilities)), weights=self.probabilities)[0]
        """
        From the documentation:
            [...]
            The period is 2**19937-1.
            It is one of the most extensively tested generators in existence.
            The random() method is implemented in C, executes in a single Python step,
            and is, therefore, threadsafe.
            [...]
        """

        mean = self.means[index]
        variance = self.variances[index]
        return random.gauss(mean, np.sqrt(variance))



myvar = ConditionalGaussian([-2, 4, 10, 15], [2, 1, 3, 2], [0.15, 0.25, 0.35, 0.25])
print("Theoretical mean: " + str(myvar.get_conditional_mean()))
print("Theoretical variance: " + str(myvar.get_conditional_variance()))

#number of samples to draw
N = 1_000_000

samples = []

for i in range(N):
    samples.append(myvar.sample())


empirical_mean_dataset, empirical_var_dataset = np.mean(samples), np.var(samples)
print("Dataset mean: " + str(empirical_mean_dataset), "\nDataset variance: " + str(empirical_var_dataset), "\n")

#Conclusions: both mean and variances computed on the dataset of the samples meet with theoretical evidences
print(f"Absolute error in mean {abs(empirical_mean_dataset-myvar.get_conditional_mean())}")
print(f"Absolute error in variance {abs(empirical_var_dataset-myvar.get_conditional_variance())}")
print(f"Relative error in mean {abs(empirical_mean_dataset-myvar.get_conditional_mean()) / abs(myvar.get_conditional_mean())}")
print(f"Relative error in variance {abs(empirical_var_dataset-myvar.get_conditional_variance()) / abs(myvar.get_conditional_variance())}")
```