CODE:

```
import numpy as np
import matplotlib.pyplot as plt

# Define the perceptron class
class Perceptron:
    def __init__(self, learning_rate=0.1, num_epochs=100):
        self.learning_rate = learning_rate
        self.num_epochs = num_epochs

    # Train the perceptron on the input data
    def train(self, X, y):
        num_samples, num_features = X.shape
        self.weights = np.zeros(num_features + 1)
        X = np.hstack((X, np.ones((num_samples, 1))))
        for epoch in range(self.num_epochs):
            for i in range(num_samples):
                y_pred = np.sign(np.dot(X[i], self.weights))
                if y_pred != y[i]:
                    self.weights += self.learning_rate * y[i] * X[i]

    # Predict the output for a given input
    def predict(self, X):
        X = np.hstack((X, np.ones((X.shape[0], 1))))
        return np.sign(np.dot(X, self.weights))

# Define the input data
X = np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 2], [6, 3]])
y = np.array([-1, -1, -1, 1, 1, 1])

# Create and train the perceptron
perceptron = Perceptron(learning_rate=0.1, num_epochs=100)
perceptron.train(X, y)

# Plot the input data and the decision boundary
plt.scatter(X[y == -1, 0], X[y == -1, 1], color='blue', marker='o', label='-1')
plt.scatter(X[y == 1, 0], X[y == 1, 1], color='red', marker='x', label='1')
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min, y_max, 0.1))
Z = perceptron.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.3, levels=[-1, 0, 1], colors=['blue', 'red'])
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xlabel('X1')
plt.ylabel('X2')
plt.legend(loc='upper left')
plt.show()
```

OUTPUT: