```
# Logistic Regression from Scratch (Single Program)

import numpy as np

# Sigmoid function
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Logistic Regression training using Gradient Descent
def train_logistic_regression(X, y, learning_rate=0.01, epochs=1000):
    m, n = X.shape
    weights = np.zeros(n)
    bias = 0

    for _ in range(epochs):
        # Linear model
        z = np.dot(X, weights) + bias
        y_pred = sigmoid(z)

        # Compute gradients
        dw = (1 / m) * np.dot(X.T, (y_pred - y))
        db = (1 / m) * np.sum(y_pred - y)

        # Update parameters
        weights -= learning_rate * dw
        bias -= learning_rate * db

    return weights, bias
```

```
# Prediction function
def predict(X, weights, bias):
    z = np.dot(X, weights) + bias
    y_pred = sigmoid(z)
    return [1 if i >= 0.5 else 0 for i in y_pred]

# Sample dataset
X = np.array([
    [2, 3],
    [1, 5],
    [2, 8],
    [5, 8],
    [6, 3],
    [7, 2]
])

y = np.array([0, 0, 0, 1, 1, 1])

# Train the model
weights, bias = train_logistic_regression(X, y, learning_rate=0.1, epochs=1000)

# Make predictions
predictions = predict(X, weights, bias)

# Output results
print("Weights:", weights)
print("Bias:", bias)
print("Predicted Values:", predictions)
print("Actual Values:   ", y.tolist())
```

OUTPUT:-

Weights: [2.25387467 -0.63722277]

Bias: -3.9189816820780523

Predicted Values: [0, 0, 0, 1, 1, 1]

Actual Values: [0, 0, 0, 1, 1, 1]