Logistic Regression

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt

# --- Step 1: Dataset ---
x = np.array([1, 2, 3, 4, 5])
y = np.array([0, 0, 0, 1, 1])
m = len(y)

# --- Step 2: Sigmoid function ---
def hT(z):
    return 1 / (1 + np.exp(-z))

# --- Step 3: Initialize parameters ---
b0 = 0.0  # intercept
b1 = 0.0  # slope

# --- Step 4: Learning rate and iterations ---
A = 0.1
iterations = 10  # you can increase for better convergence

# --- Step 5: Gradient Descent Loop ---
for it in range(iterations):
    z = b0 + b1 * x
    p = hT(z)

    # Compute Cost / Log Loss
    JB = -(1/m) * np.sum(y * np.log(p) + (1 - y) * np.log(1 - p))

    # Compute Gradients
    GB0 = (1/m) * np.sum(p - y)
    GB1 = (1/m) * np.sum((p - y) * x)

    # Update Parameters
    b0 = b0 - A * GB0
    b1 = b1 - A * GB1

    # Print iteration info
    print(f"Iteration {it+1}: JB={JB:.4f}, b0={b0:.4f}, b1={b1:.4f}")

# --- Step 6: Predictions ---
x_curve = np.linspace(x.min(), x.max(), 100)
z_curve = b0 + b1 * x_curve
p_curve = hT(z_curve)

# --- Step 7: Plot ---
plt.scatter(x, y, label='Original Data', color='blue')
plt.plot(x_curve, p_curve, label='Learned Probability Curve', color='red')
plt.xlabel('x')
plt.ylabel('Probability')
```

```
plt.title('Logistic Regression (Gradient Descent)')
plt.legend()
plt.grid(True)
plt.show()
```

```
Iteration 1: JB=0.6931, b0=-0.0100, b1=0.0300
Iteration 2: JB=0.6842, b0=-0.0220, b1=0.0525
Iteration 3: JB=0.6782, b0=-0.0354, b1=0.0698
Iteration 4: JB=0.6737, b0=-0.0497, b1=0.0833
Iteration 5: JB=0.6699, b0=-0.0647, b1=0.0943
Iteration 6: JB=0.6665, b0=-0.0801, b1=0.1034
Iteration 7: JB=0.6634, b0=-0.0958, b1=0.1112
Iteration 8: JB=0.6603, b0=-0.1117, b1=0.1181
Iteration 9: JB=0.6573, b0=-0.1277, b1=0.1243
Iteration 10: JB=0.6544, b0=-0.1437, b1=0.1301
```