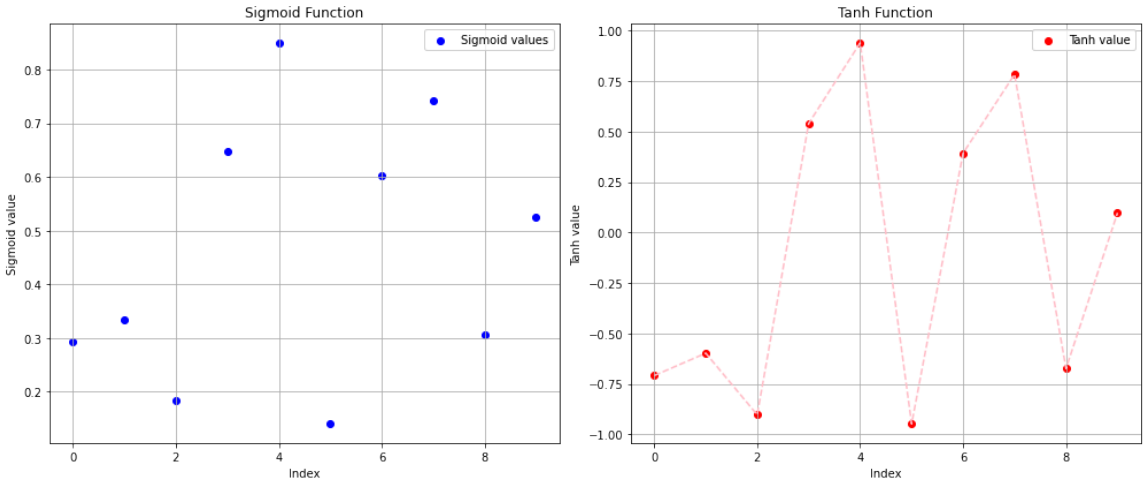
In [23]:
```
import numpy as np
import matplotlib.pyplot as plt
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def tanh(x):
    return np.tanh(x)
random_values = np.random.randn(10)
sigmoid_values = sigmoid(random_values)
tanh_values = tanh(random_values)
indices=np.arange(len(random_values))
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
plt.scatter(indices, sigmoid_values, color='blue', label='Sigmoid values')
plt.title('Sigmoid Function')
plt.xlabel('Index')
plt.ylabel('Sigmoid value')
plt.grid(True)
plt.legend()
plt.subplot(1, 2, 2)
plt.scatter(indices, tanh_values, color='red', label='Tanh value')
plt.plot(indices,tanh_values,color='pink',linestyle='--')
plt.title('Tanh Function')
plt.xlabel('Index')
plt.ylabel('Tanh value')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



In [19]:
```
import numpy as np
def gradient_descent(f,initial_point,learning_rate, iterations):
    point=initial_point.copy()
    history=[]
    for _ in range(iterations):
        gradient=np.array([2*point[0],2*point[1]])
        point-=learning_rate*gradient
        value=f(point[0],point[1])
        history.append((point.copy(),value))
        print("Values of point and objective function value are:{} and {}".
        return point,history
def objective_function(x,y):
        return x**2+y**2+4
initial_point=np.array([1.0,1.0])
learning_rate=0.1
iterations=100
minimum_point,history=gradient_descent(objective_function,initial_point,lea
print("Minimum point(x,y):",minimum_point)
print("Minimum value ofthe function:",objective_function(*minimum_point))
```

```
Values of point and objective function value are:[0.8 0.8] and 5.28
Minimum point(x,y): [0.8 0.8]
Minimum value ofthe function: 5.28
```

In [ ]: