CODE:

```
import numpy as np

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

# ReLU activation function
def relu(x):
    return np.maximum(0, x)

# Leaky ReLU activation function
def leaky_relu(x, alpha=0.01):
    return np.maximum(alpha*x, x)

# Softmax activation function
def softmax(x):
    exp_x = np.exp(x - np.max(x, axis=-1, keepdims=True))
    return exp_x / np.sum(exp_x, axis=-1, keepdims=True)
# Test the sigmoid function
x_sigmoid = np.array([[-1, 0, 1], [-2, 2, 3]])
print(sigmoid(x_sigmoid))

# Test the ReLU function
x_relu = np.array([[-1, 0, 1], [-2, 2, 3]])
print(relu(x_relu))

# Test the leaky ReLU function
x_leaky_relu = np.array([[-1, 0, 1], [-2, 2, 3]])
print(leaky_relu(x_leaky_relu))

# Test the softmax function
x_softmax = np.array([[1, 2, 3], [4, 5, 6]])
print(softmax(x_softmax))
```

OUTPUT: (matrix format)

```
[[0.26894142 0.5        0.73105858]
 [0.11920292 0.88079708 0.95257413]]
[[0 0 1]
 [0 2 3]]
[[-0.01  0.    1.  ]
 [-0.02  2.    3.  ]]
[[0.09003057 0.24472847 0.66524096]
 [0.09003057 0.24472847 0.66524096]]
```

CODE:

```
(graph plotting)
import matplotlib.pyplot as plt

# Generate a range of values from -10 to 10
x = np.linspace(-10, 10, 100)

# Compute the activation functions
y_sigmoid = sigmoid(x)
y_relu = relu(x)
y_leaky_relu = leaky_relu(x)
y_softmax = softmax(x)

# Create a new figure
plt.figure(figsize=(14, 7))

# Plot the activation functions
plt.subplot(2, 2, 1)
plt.plot(x, y_sigmoid)
plt.title('Sigmoid')

plt.subplot(2, 2, 2)
plt.plot(x, y_relu)
plt.title('ReLU')

plt.subplot(2, 2, 3)
plt.plot(x, y_leaky_relu)
plt.title('Leaky ReLU')

plt.subplot(2, 2, 4)
plt.plot(x, y_softmax)
plt.title('Softmax')

# Show the plot
plt.tight_layout()
plt.show()
```

OUTPUT: