In [1]:

```python
import numpy as np

class Neuron:
    def __init__(self, n_inputs):
        self.weights = np.random.rand(n_inputs)
        self.bias = np.random.rand(1)
    def activate(self, x, activation_function='sigmoid'):
        if activation_function == 'sigmoid':
            return self.sigmoid(x)
        elif activation_function == 'tanh':
            return self.tanh(x)
        elif activation_function == 'relu':
            return self.relu(x)
        else:
            raise ValueError("Unsupported activation function.")
    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))
    def tanh(self, x):
        return np.tanh(x)
    def relu(self, x):
        return np.maximum(0, x)
    def forward(self, inputs, activation_function='sigmoid'):
        linear_combination = np.dot(inputs, self.weights) + self.bias
        return self.activate(linear_combination, activation_function)
data = np.array([[0.5, 1.0], [1.5, 2.0], [-1.0, -0.5], [0.0, 0.0]])
neuron = Neuron(n_inputs=2)
activation_functions = ['sigmoid', 'tanh', 'relu']
results = {}
for activation in activation_functions:
    results[activation] = []
    for input_vector in data:
```

```
        output = neuron.forward(input_vector, activation_function=activation)
        results[activation].append(output)
for activation, outputs in results.items():
    print(f"Outputs with {activation} activation function:")
    print(outputs)
    print()
```

```
Outputs with sigmoid activation function:
[array([0.77120076]), array([0.85978799]), array([0.57870614]), array([0.68527007])]

Outputs with tanh activation function:
[array([0.83820394]), array([0.94818937]), array([0.30721227]), array([0.65161372])]

Outputs with relu activation function:
[array([1.2151038]), array([1.81353019]), array([0.31746423]), array([0.77809812])]
```

In [ ]:

```
1
```