In [5]:

```
#program 1
#create  and implement a basic neuron model with the computational framework integrate essential elements
#like input node,weight parameters,bias and an activation function(included but not limited to sigmoid
#hyperbolic,tangent and relu)
#to construct a comprehensive representation of neuron then evaluate and observe how each activation
#function influences the network behavior and effectiveness in handling different types of data
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([0.76437077]), array([0.90764893]), array([0.38085829]), array([0.58370398])]

Outputs with tanh activation function:
[array([0.82643783]), array([0.97950702]), array([-0.4509617]), array([0.32568836])]

Outputs with relu activation function:
[array([1.17679348]), array([2.28526036]), array([0.]), array([0.33799737])]
```

In [ ]: