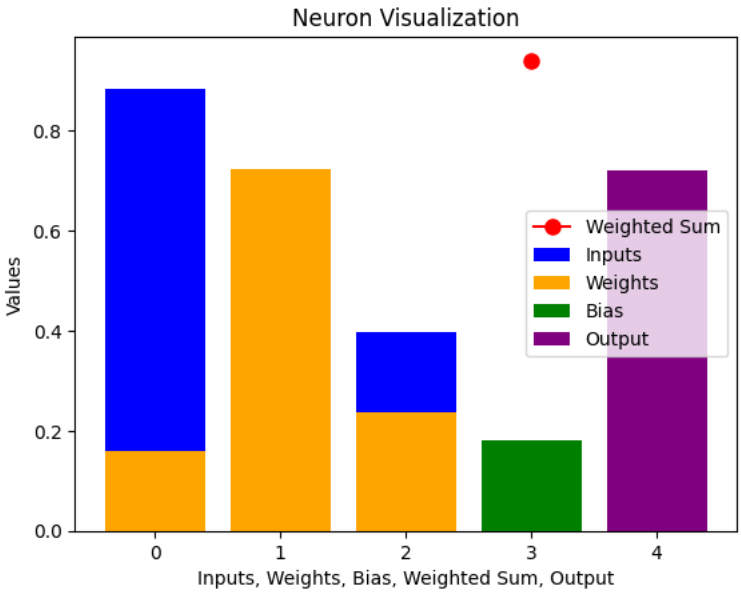
```
class Neuron:
    def __init__(self, input_size, activation_function='sigmoid'):
        self.weights = np.random.rand(input_size)
        self.bias = np.random.rand(1)
        self.activation_function = activation_function

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def step_function(self, x):
        return 1 if x >= 0 else 0

    def activate(self, x):
        if self.activation_function == 'sigmoid':
            return self.sigmoid(x)
        elif self.activation_function == 'step':
            return self.step_function(x)
        else:
            raise ValueError("Invalid activation function")

    def forward(self, inputs):
        weighted_sum = np.dot(inputs, self.weights) + self.bias
        return self.activate(weighted_sum)


input_size = 3
neuron = Neuron(input_size, activation_function='sigmoid')


inputs = np.random.rand(input_size)

output = neuron.forward(inputs)

print(f"Inputs: {inputs}")
print(f"Weights: {neuron.weights}")
print(f"Bias: {neuron.bias}")
print(f"Output: {output}")
```

```
Inputs: [0.13111627 0.21614359 0.28100589]
Weights: [0.55249074 0.62142993 0.62699614]
Bias: [0.45381713]
Output: [0.69778353]
```


Neuron Visualization
Values
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
Inputs, Weights, Bias, Weighted Sum, Output
Weighted Sum
Inputs
Weights
Bias
Output