In [15]:
```
import numpy as np
```

In [27]:
```
class activation_fcn(object):
    def __init__(self):
        self.functions = {
    'linear': self.linear,
    'sigmoid': self.logistic,
    'logistic': self.logistic,
    'tanh': self.tanh,
    'relu': self.relu
    }
    def output(self, layer, name):
        if name in self.functions:
            return self.functions[name](layer)
        else:
            sys.exit(f"Error: Activation function '{name}' not found.")
    def linear(self, layer):
        return layer['activation_potential']

    def logistic(self, layer):
        return 1 / (1 + np.exp(-layer['activation_potential']))

    def tanh(self, layer):
        return np.tanh(layer['activation_potential'])

    def relu(self, layer):
        return np.maximum(0, layer['activation_potential'])
```

In [28]:
```
if __name__ == "__main__":
    # Tworzymy przykładową warstwę z potencjałem aktywacyjnym
    layer = {'activation_potential': np.array([-2, -1, 0, 1, 2])}

    # Tworzymy obiekt klasy activation_fcn
    activation = activation_fcn()

    # Testujemy różne funkcje aktywacji
    print("Linear:", activation.output(layer, 'linear'))
    print("Sigmoid:", activation.output(layer, 'sigmoid'))
    print("Tanh:", activation.output(layer, 'tanh'))
    print("ReLU:", activation.output(layer, 'relu'))
```

```
Linear: [-2 -1  0  1  2]
Sigmoid: [0.11920292 0.26894142 0.5        0.73105858 0.88079708]
Tanh: [-0.96402758 -0.76159416  0.          0.76159416  0.96402758]
ReLU: [0 0 0 1 2]
```

In [ ]:
```

```