

**Creating class for neuron which consists of computation for the weighted sum of inputs and ReLU activation function:**

```
In [ ]: class Neuron:
    def __init__(self, input_list, weight_list):
        self.input_list = input_list
        self.weight_list = weight_list

    # A function for computing weighted sum of the inputs
    def sum(self):
        total = 0
        for input_value, weight_value in zip(self.input_list, self.weight_list):
            total += input_value * weight_value
        return total

    # Activation function (ReLU)
    def rectified(output):
        if output > 0:
            return output
        else:
            return 0
```

```
In [ ]: input_list = [5, 7, 2, 6, 7] # 5 inputs
weight_list = [0.1, 0.2, 0.3, 0.5, 0.6] # 5 weights

result = Neuron(input_list, weight_list)
add_unit = result.sum() # computing their add unit
print("Inputs: ", input_list) # printing the inputs
print("Weights: ", weight_list) # printing the weights
print("Add Unit: ", add_unit) # printing the add unit
with_act_function = Neuron.rectified(add_unit) # applying the ReLU activation function
print("With ReLU Activation: ", with_act_function) # printing the value when ReLU activation is applied

Inputs: [5, 7, 2, 6, 7]
Weights: [0.1, 0.2, 0.3, 0.5, 0.6]
Add Unit: 9.7
With ReLU Activation: 9.7
```

**The Rectified Linear Unit (ReLU) outputs  $x$  if it is positive. However, if it is negative or 0, the function outputs 0.**

```
In [ ]: from random import randint

weight_list_2 = [] # random weights
n=5

for i in range(n):
    weight_list_2.append(randint(1,10))

result = Neuron(input_list, weight_list_2) # updating the weight list
add_unit = result.sum() # computing the add unit with new random weights
print("Random Weights: ", weight_list_2) # printing the weights
print("Add Unit: ", add_unit) # printing the add unit
with_act_function = Neuron.rectified(add_unit) # applying the ReLU activation function
print("With ReLU Activation: ", with_act_function) # printing the value when ReLU activation is applied

Random Weights: [9, 4, 6, 4, 8]
Add Unit: 165
With ReLU Activation: 165
```

FEED FORWARD NEURAL NETWORK									
Input		Weight		Neuron		Random Weight		Weight	
5	0.1			Add Unit	9.7	9			
7	0.2			Activation Function	9.7	4			
2	0.3			Neuron		6			
6	0.5			Add Unit	9.7	4		Add Unit	165
7	0.6			Activation Function	9.7	8		Activation Function	165
				Neuron					
				Add Unit	9.7				
				Activation Function	9.7				
				Neuron					
				Add Unit	9.7				
				Activation Function	9.7				
				Neuron					
				Add Unit	9.7				
				Activation Function	9.7				

***To check if the outputs are correct, I manually computed it using the excel. It shows similar result which means that code for feed forward neural network successfully computed the add unit with an activation function.***

Google colab link: <https://colab.research.google.com/drive/1x4aQL1BwjzT-bX5Mqaf8uonIcGlnHntS?usp=sharing>