Experiment-3

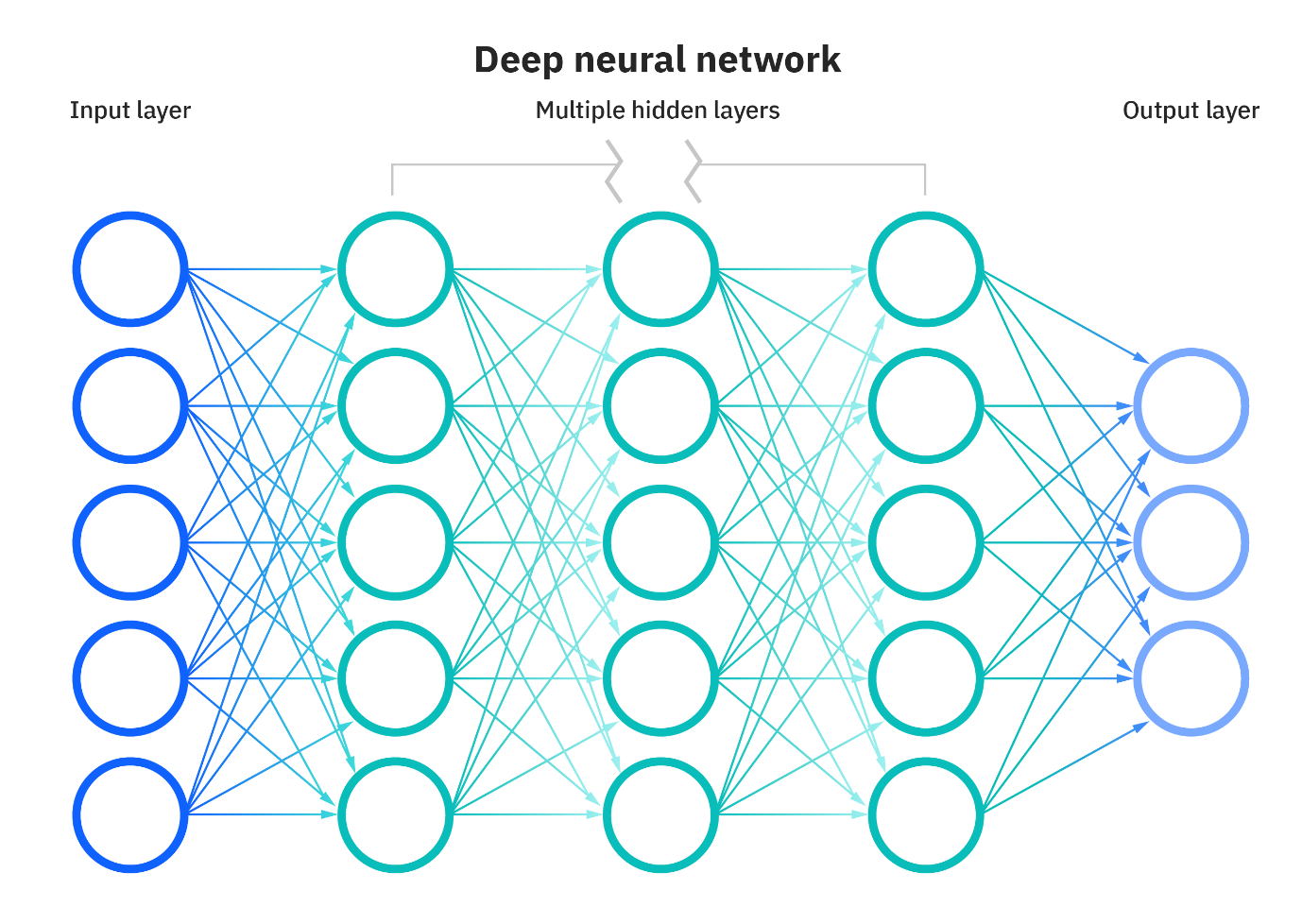
Aim:- To Create First Neural Network

Neural Network:- It is an artificial network or model which is inspired by human brain neural network. It take one or more input to provide desire output.

There are two types of Neural Network:-

|  |  |
| --- | --- |
| Biological Neural Network | Artificial Neural Network |
| BNN is made up of dendrites, synapse ,axon and cell body. | ANN is made up of input, weight, output and hidden layer. |
| It is simple but low speed. | It is complex but high speed. |
| It is connected in network. | It is organised in layers. |

**Artificial Neural Network:-**



Source:- IBM

Input Layer:- It receive raw data as an input.

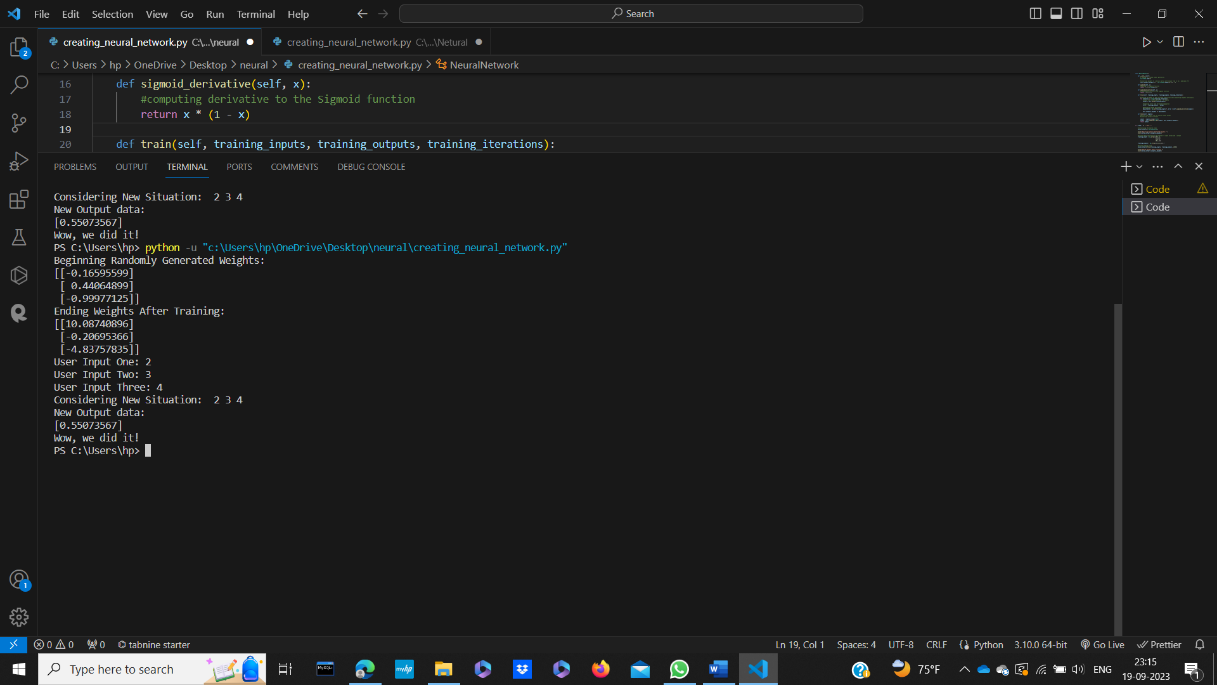
Hidden Layer:- It is intermediary layer between the input and output layer. A simple neural network can contain multiple hidden layer. This layer is use to perform computation on input data by providing weight and activation function.

Output Layer:- This is the final layer of the neural network.

**Code:-**

|  |
| --- |
| import numpy as np  class NeuralNetwork():    def \_\_init\_\_(self):  # seeding for random number generation  np.random.seed(1)    #converting weights to a 3 by 1 matrix with values from -1 to 1 and mean of 0  self.synaptic\_weights = 2 \* np.random.random((3, 1)) - 1  def sigmoid(self, x):  #applying the sigmoid function  return 1 / (1 + np.exp(-x))  def sigmoid\_derivative(self, x):  #computing derivative to the Sigmoid function  return x \* (1 - x)  def train(self, training\_inputs, training\_outputs, training\_iterations):    #training the model to make accurate predictions while adjusting weights continually  for iteration in range(training\_iterations):  #siphon the training data via the neuron  output = self.think(training\_inputs)  #computing error rate for back-propagation  error = training\_outputs - output    #performing weight adjustments  adjustments = np.dot(training\_inputs.T, error \* self.sigmoid\_derivative(output))  self.synaptic\_weights += adjustments  def think(self, inputs):  #passing the inputs via the neuron to get output  #converting values to floats    inputs = inputs.astype(float)  output = self.sigmoid(np.dot(inputs, self.synaptic\_weights))  return output  if \_\_name\_\_ == "\_\_main\_\_":  #initializing the neuron class  neural\_network = NeuralNetwork()  print("Beginning Randomly Generated Weights: ")  print(neural\_network.synaptic\_weights)  #training data consisting of 4 examples--3 input values and 1 output  training\_inputs = np.array([[0,0,1],  [1,1,1],  [1,0,1],  [0,1,1]])  training\_outputs = np.array([[0,1,1,0]]).T  #training taking place  neural\_network.train(training\_inputs, training\_outputs, 15000)  print("Ending Weights After Training: ")  print(neural\_network.synaptic\_weights)  user\_input\_one = str(input("User Input One: "))  user\_input\_two = str(input("User Input Two: "))  user\_input\_three = str(input("User Input Three: "))    print("Considering New Situation: ", user\_input\_one, user\_input\_two, user\_input\_three)  print("New Output data: ")  print(neural\_network.think(np.array([user\_input\_one, user\_input\_two, user\_input\_three])))  print("Wow, we did it!") |

**Output:-**

****