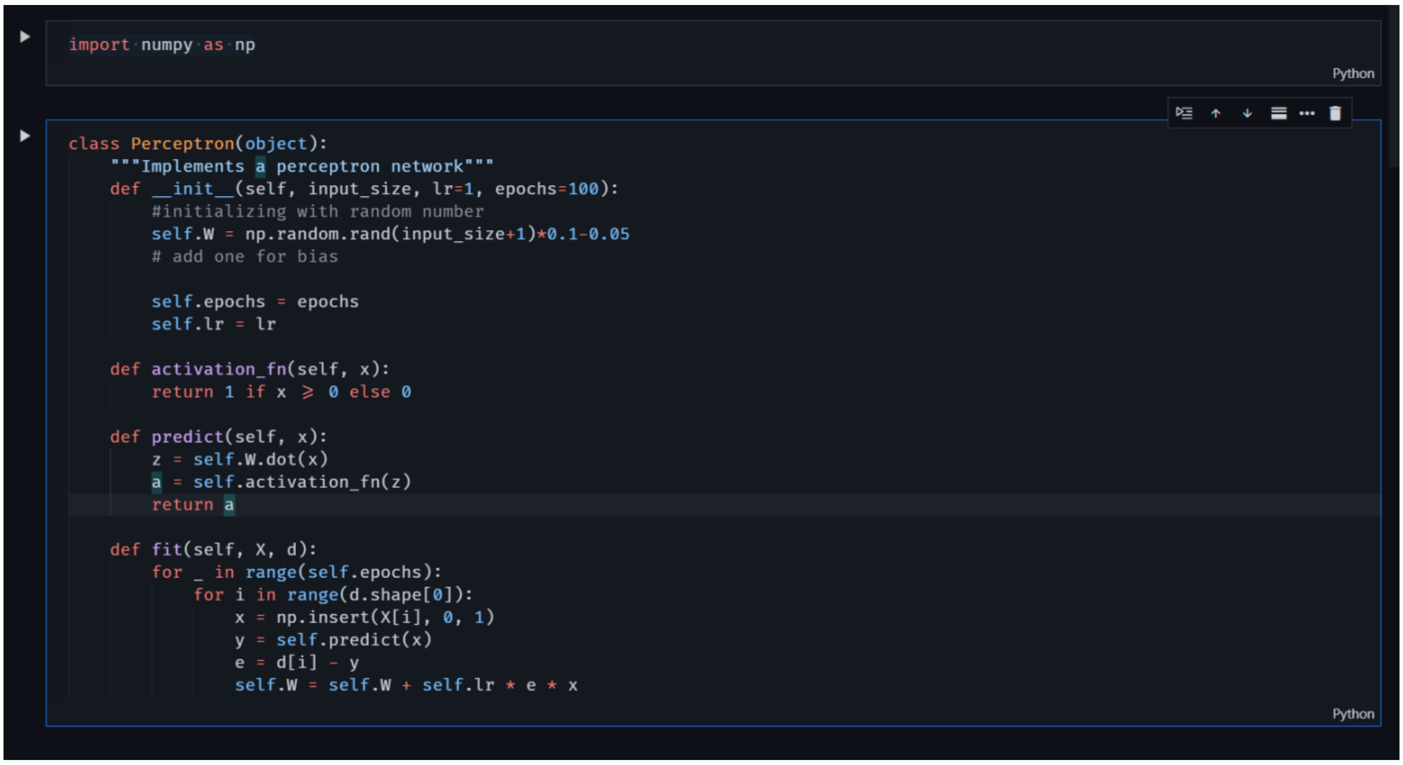
LAB – 5 – SINGLE LAYER PERCEPTRON - LAB

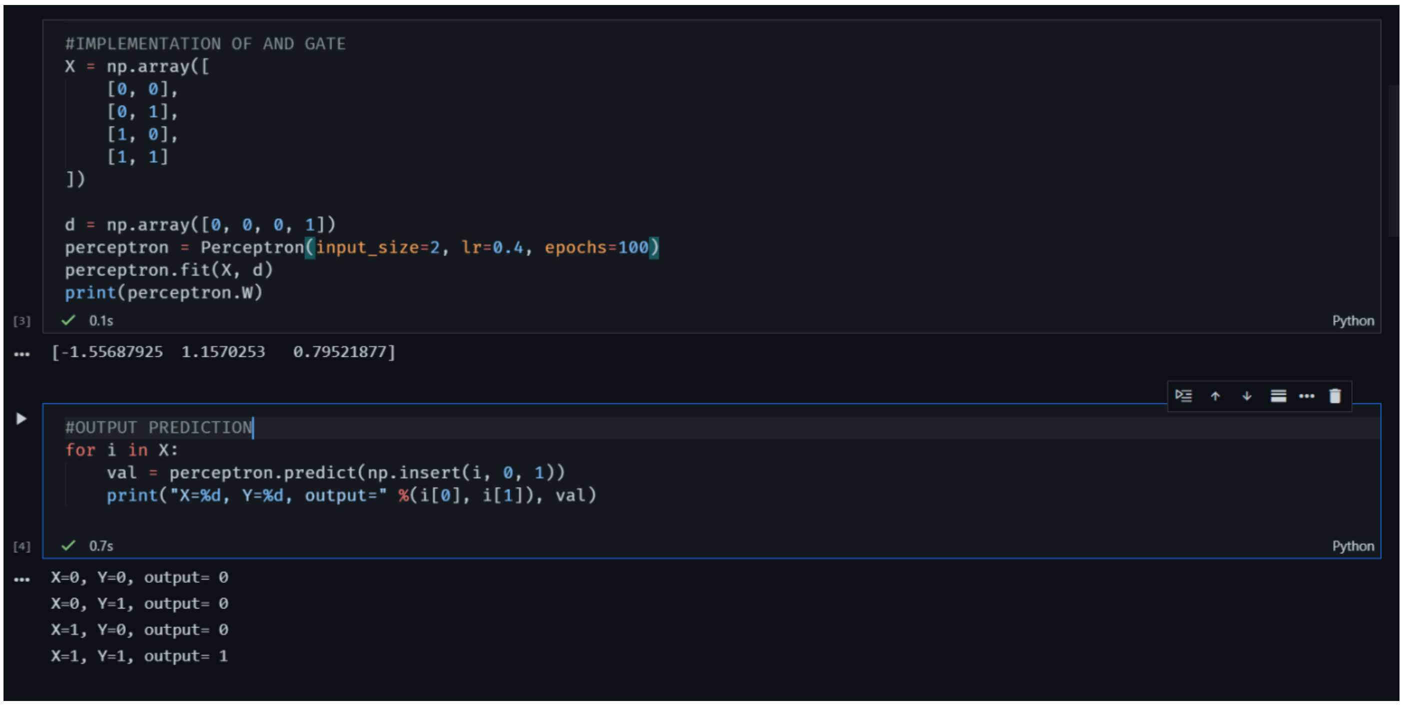
CODE:-



```
import numpy as np
```

```
class Perceptron(object):
    """Implements a perceptron network"""
    def __init__(self, input_size, lr=1, epochs=100):
        #initializing with random number
        self.W = np.random.rand(input_size+1)*0.1-0.05
        # add one for bias

        self.epochs = epochs
        self.lr = lr

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

    def predict(self, x):
        z = self.W.dot(x)
        a = self.activation_fn(z)
        return a

    def fit(self, X, d):
        for _ in range(self.epochs):
            for i in range(d.shape[0]):
                x = np.insert(X[i], 0, 1)
                y = self.predict(x)
                e = d[i] - y
                self.W = self.W + self.lr * e * x
```

AND GATE:-



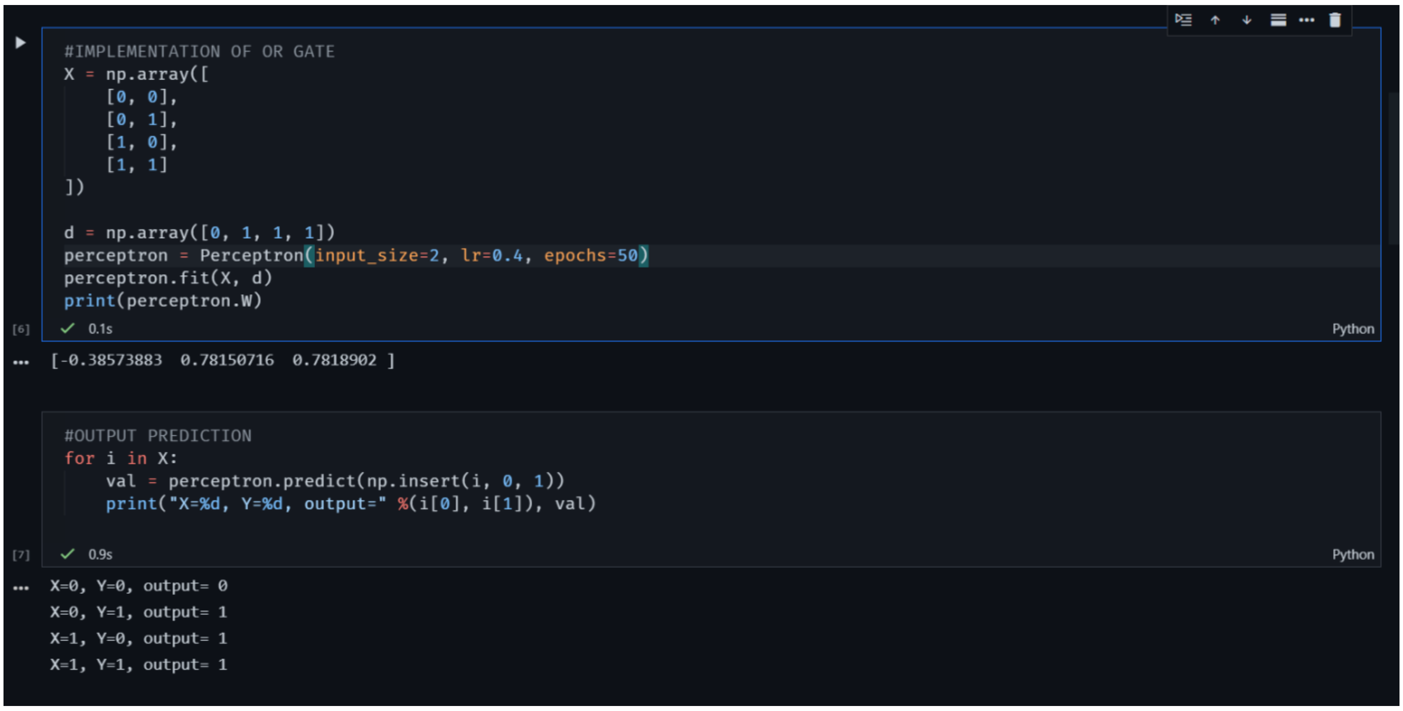
```
#IMPLEMENTATION OF AND GATE
X = np.array([
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
])

d = np.array([0, 0, 0, 1])
perceptron = Perceptron(input_size=2, lr=0.4, epochs=100)
perceptron.fit(X, d)
print(perceptron.W)
```

```
[-1.55687925  1.1570253   0.79521877]
```

```
#OUTPUT PREDICTION
for i in X:
    val = perceptron.predict(np.insert(i, 0, 1))
    print("X=%d, Y=%d, output=" %(i[0], i[1]), val)
```

```
X=0, Y=0, output= 0
X=0, Y=1, output= 0
X=1, Y=0, output= 0
X=1, Y=1, output= 1
```

OR GATE



```
#IMPLEMENTATION OF OR GATE
X = np.array([
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
])

d = np.array([0, 1, 1, 1])
perceptron = Perceptron(input_size=2, lr=0.4, epochs=50)
perceptron.fit(X, d)
print(perceptron.W)
```

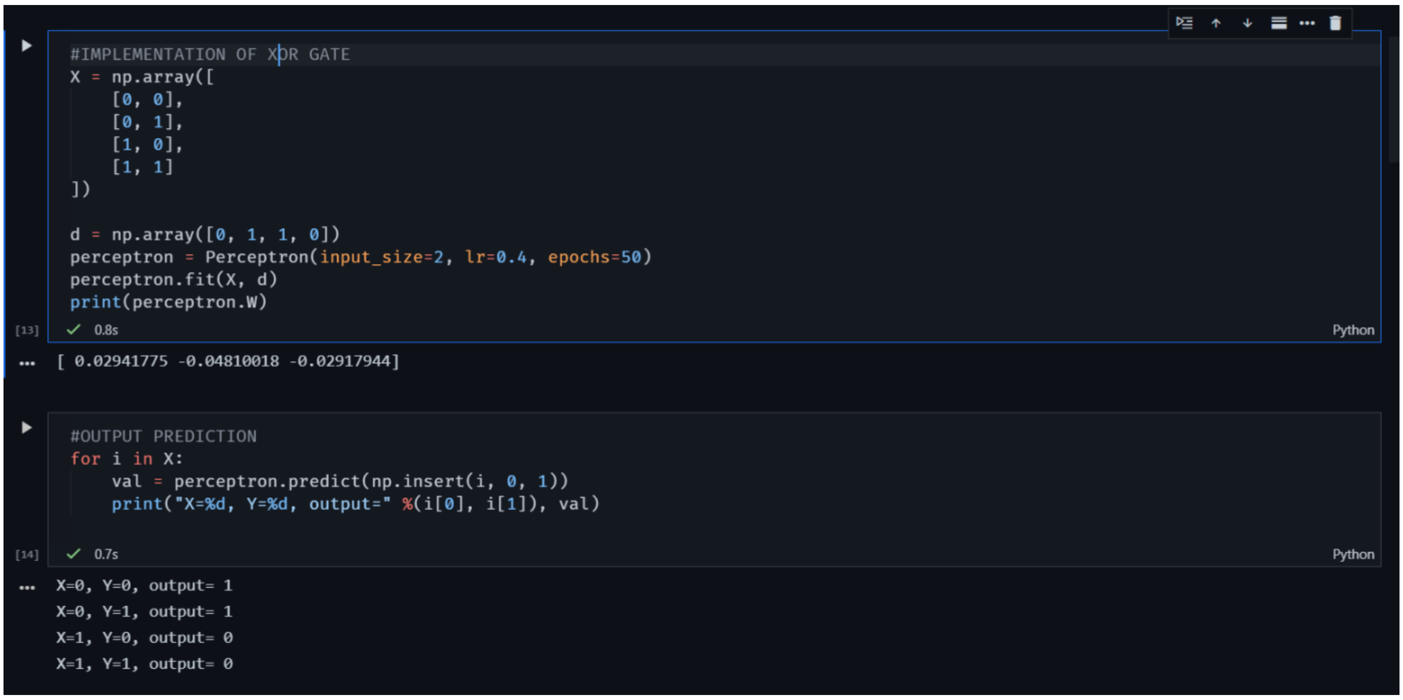
```
[-0.38573883  0.78150716  0.7818902 ]
```

```
#OUTPUT PREDICTION
for i in X:
    val = perceptron.predict(np.insert(i, 0, 1))
    print("X=%d, Y=%d, output=" %(i[0], i[1]), val)
```

```
X=0, Y=0, output= 0
X=0, Y=1, output= 1
X=1, Y=0, output= 1
X=1, Y=1, output= 1
```

BOTH AND , OR ARE LINEARLY SEPARABLE HENCE CAN USE A SINGLE LAYER PERCEPTRON TO MAKE IT LEARN.

XOR GATE:-



```
#IMPLEMENTATION OF XOR GATE
X = np.array([
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
])

d = np.array([0, 1, 1, 0])
perceptron = Perceptron(input_size=2, lr=0.4, epochs=50)
perceptron.fit(X, d)
print(perceptron.W)
```

```
[ 0.02941775 -0.04810018 -0.02917944]
```

```
#OUTPUT PREDICTION
for i in X:
    val = perceptron.predict(np.insert(i, 0, 1))
    print("X=%d, Y=%d, output=" %(i[0], i[1]), val)
```

```
X=0, Y=0, output= 1
X=0, Y=1, output= 1
X=1, Y=0, output= 0
X=1, Y=1, output= 0
```

Here we can see that we can't use our single-layer perceptron to model an XOR gate.