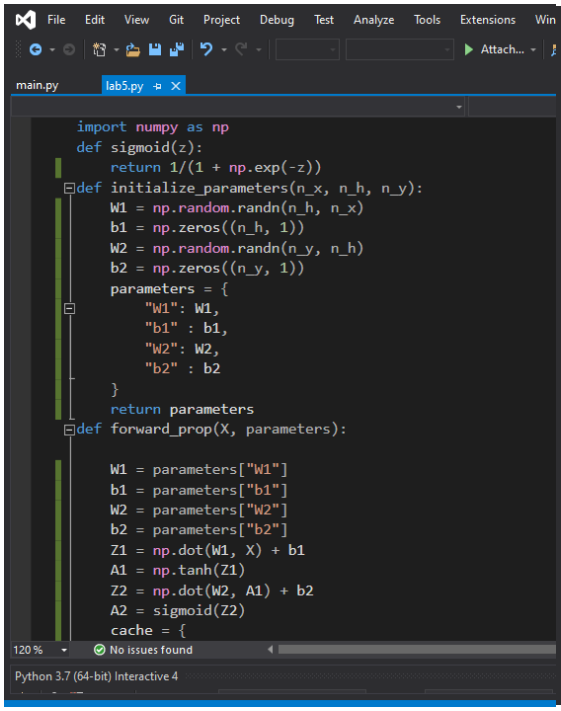
LAB5-AI
NIDA-RAMZAN CS-121

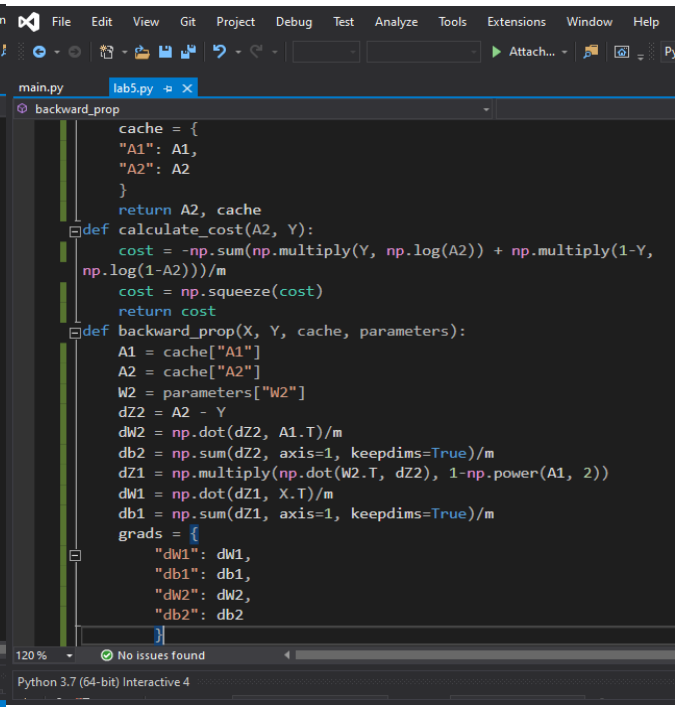
LAB: 05

XOR:

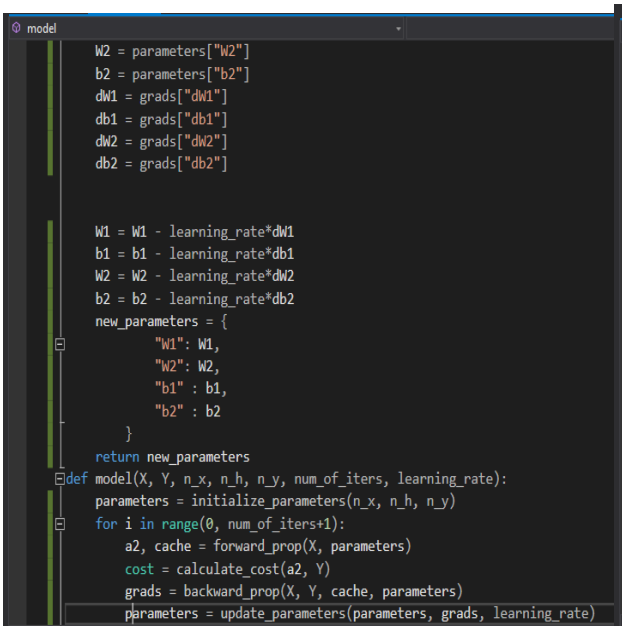


```python
import numpy as np
def sigmoid(z):
    return 1/(1 + np.exp(-z))
def initialize_parameters(n_x, n_h, n_y):
    W1 = np.random.randn(n_h, n_x)
    b1 = np.zeros((n_h, 1))
    W2 = np.random.randn(n_y, n_h)
    b2 = np.zeros((n_y, 1))
    parameters = {
        "W1": W1,
        "b1" : b1,
        "W2": W2,
        "b2" : b2
    }
    return parameters
def forward_prop(X, parameters):

    W1 = parameters["W1"]
    b1 = parameters["b1"]
    W2 = parameters["W2"]
    b2 = parameters["b2"]
    Z1 = np.dot(W1, X) + b1
    A1 = np.tanh(Z1)
    Z2 = np.dot(W2, A1) + b2
    A2 = sigmoid(Z2)
    cache = {
```

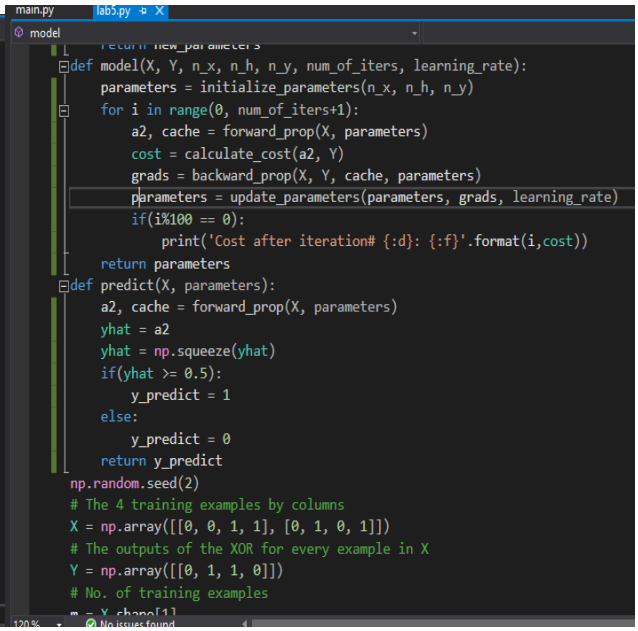


```python
    cache = {
    "A1": A1,
    "A2": A2
    }
    return A2, cache
def calculate_cost(A2, Y):
    cost = -np.sum(np.multiply(Y, np.log(A2)) + np.multiply(1-Y,
np.log(1-A2)))/m
    cost = np.squeeze(cost)
    return cost
def backward_prop(X, Y, cache, parameters):
    A1 = cache["A1"]
    A2 = cache["A2"]
    W2 = parameters["W2"]
    dZ2 = A2 - Y
    dW2 = np.dot(dZ2, A1.T)/m
    db2 = np.sum(dZ2, axis=1, keepdims=True)/m
    dZ1 = np.multiply(np.dot(W2.T, dZ2), 1-np.power(A1, 2))
    dW1 = np.dot(dZ1, X.T)/m
    db1 = np.sum(dZ1, axis=1, keepdims=True)/m
    grads = {
        "dW1": dW1,
        "db1": db1,
        "dW2": dW2,
        "db2": db2
        }
```



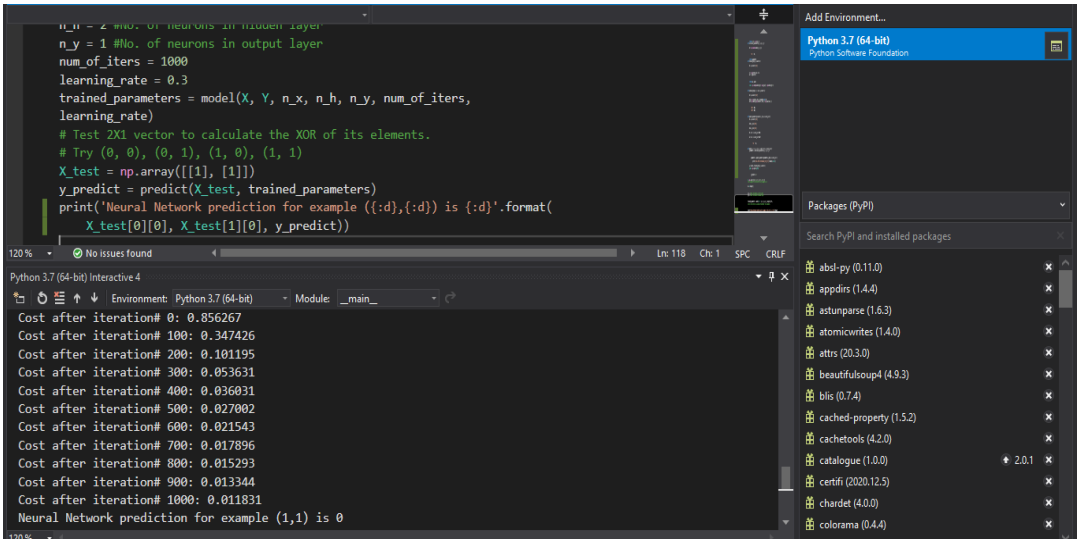
```python
    W2 = parameters["W2"]
    b2 = parameters["b2"]
    dW1 = grads["dW1"]
    db1 = grads["db1"]
    dW2 = grads["dW2"]
    db2 = grads["db2"]


    W1 = W1 - learning_rate*dW1
    b1 = b1 - learning_rate*db1
    W2 = W2 - learning_rate*dW2
    b2 = b2 - learning_rate*db2
    new_parameters = {
            "W1": W1,
            "W2": W2,
            "b1" : b1,
            "b2" : b2
        }
    return new_parameters
def model(X, Y, n_x, n_h, n_y, num_of_iters, learning_rate):
    parameters = initialize_parameters(n_x, n_h, n_y)
    for i in range(0, num_of_iters+1):
        a2, cache = forward_prop(X, parameters)
        cost = calculate_cost(a2, Y)
        grads = backward_prop(X, Y, cache, parameters)
        parameters = update_parameters(parameters, grads, learning_rate)
```



```python
def model(X, Y, n_x, n_h, n_y, num_of_iters, learning_rate):
    parameters = initialize_parameters(n_x, n_h, n_y)
    for i in range(0, num_of_iters+1):
        a2, cache = forward_prop(X, parameters)
        cost = calculate_cost(a2, Y)
        grads = backward_prop(X, Y, cache, parameters)
        parameters = update_parameters(parameters, grads, learning_rate)
        if(i%100 == 0):
            print('Cost after iteration# {:d}: {:f}'.format(i,cost))
    return parameters
def predict(X, parameters):
    a2, cache = forward_prop(X, parameters)
    yhat = a2
    yhat = np.squeeze(yhat)
    if(yhat >= 0.5):
        y_predict = 1
    else:
        y_predict = 0
    return y_predict
np.random.seed(2)
# The 4 training examples by columns
X = np.array([[0, 0, 1, 1], [0, 1, 0, 1]])
# The outputs of the XOR for every example in X
Y = np.array([[0, 1, 1, 0]])
# No. of training examples
```

LAB5-AI
NIDA-RAMZAN CS-121



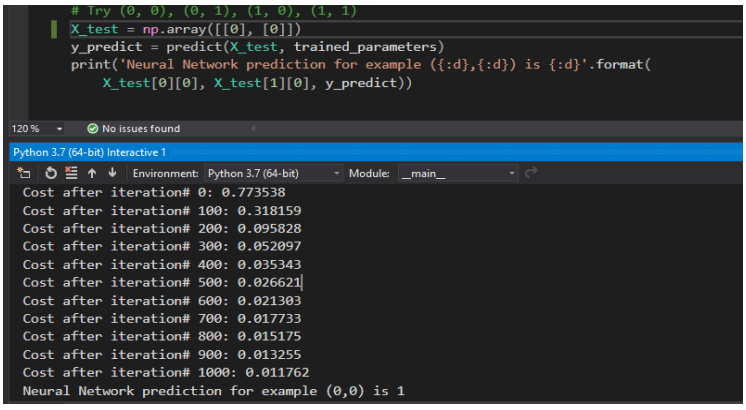
XNOR:

Changes I/p:

```
np.random.seed(2)
# The 4 training examples by columns
X = np.array([[0, 0, 1, 1], [0, 1, 0, 1]])
# The outputs of the XOR for every example in X
Y = np.array([[1, 0, 0, 1]])
# No. of training examples
m = X.shape[1]
```

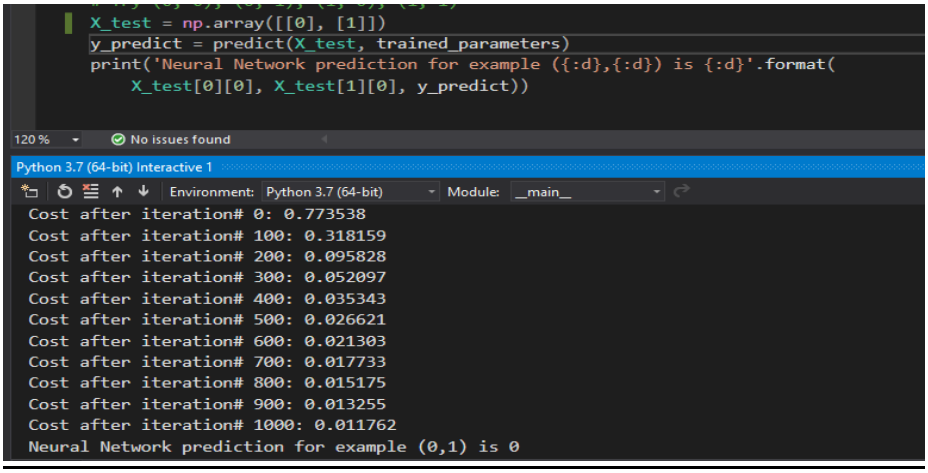
Given training patterns:

CASE: 00



LAB5-AI
NIDA-RAMZAN CS-121

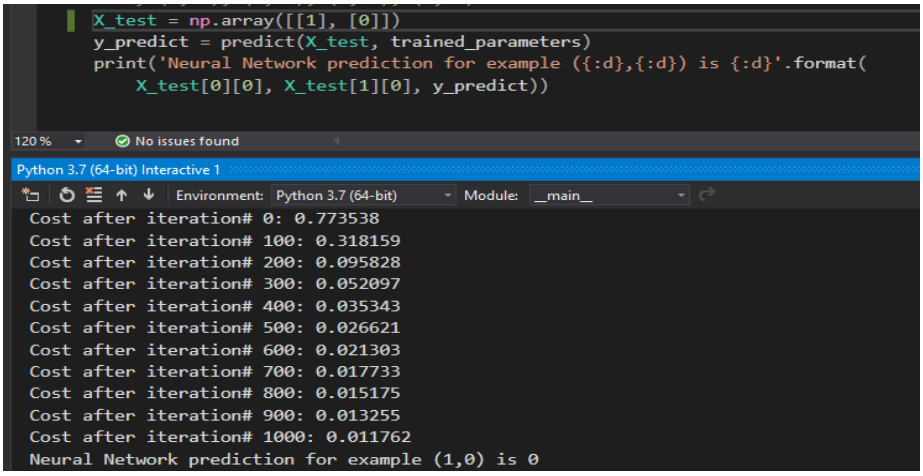
CASE:01



```
X_test = np.array([[0], [1]])
y_predict = predict(X_test, trained_parameters)
print('Neural Network prediction for example ({:d},{:d}) is {:d}'.format(
    X_test[0][0], X_test[1][0], y_predict))
```

```
120 %  No issues found
Python 3.7 (64-bit) Interactive 1
Environment: Python 3.7 (64-bit)  Module: __main__
Cost after iteration# 0: 0.773538
Cost after iteration# 100: 0.318159
Cost after iteration# 200: 0.095828
Cost after iteration# 300: 0.052097
Cost after iteration# 400: 0.035343
Cost after iteration# 500: 0.026621
Cost after iteration# 600: 0.021303
Cost after iteration# 700: 0.017733
Cost after iteration# 800: 0.015175
Cost after iteration# 900: 0.013255
Cost after iteration# 1000: 0.011762
Neural Network prediction for example (0,1) is 0
```

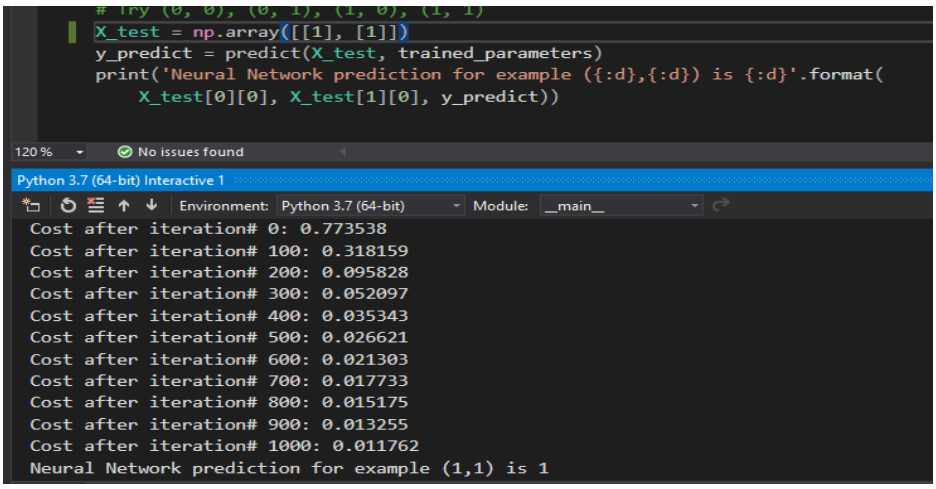
CASE:10



```
X_test = np.array([[1], [0]])
y_predict = predict(X_test, trained_parameters)
print('Neural Network prediction for example ({:d},{:d}) is {:d}'.format(
    X_test[0][0], X_test[1][0], y_predict))
```

```
120 %  No issues found
Python 3.7 (64-bit) Interactive 1
Environment: Python 3.7 (64-bit)  Module: __main__
Cost after iteration# 0: 0.773538
Cost after iteration# 100: 0.318159
Cost after iteration# 200: 0.095828
Cost after iteration# 300: 0.052097
Cost after iteration# 400: 0.035343
Cost after iteration# 500: 0.026621
Cost after iteration# 600: 0.021303
Cost after iteration# 700: 0.017733
Cost after iteration# 800: 0.015175
Cost after iteration# 900: 0.013255
Cost after iteration# 1000: 0.011762
Neural Network prediction for example (1,0) is 0
```

CASE:11



```
# Try (0, 0), (0, 1), (1, 0), (1, 1)
X_test = np.array([[1], [1]])
y_predict = predict(X_test, trained_parameters)
print('Neural Network prediction for example ({:d},{:d}) is {:d}'.format(
    X_test[0][0], X_test[1][0], y_predict))
```

```
120 %  No issues found
Python 3.7 (64-bit) Interactive 1
Environment: Python 3.7 (64-bit)  Module: __main__
Cost after iteration# 0: 0.773538
Cost after iteration# 100: 0.318159
Cost after iteration# 200: 0.095828
Cost after iteration# 300: 0.052097
Cost after iteration# 400: 0.035343
Cost after iteration# 500: 0.026621
Cost after iteration# 600: 0.021303
Cost after iteration# 700: 0.017733
Cost after iteration# 800: 0.015175
Cost after iteration# 900: 0.013255
Cost after iteration# 1000: 0.011762
Neural Network prediction for example (1,1) is 1
```