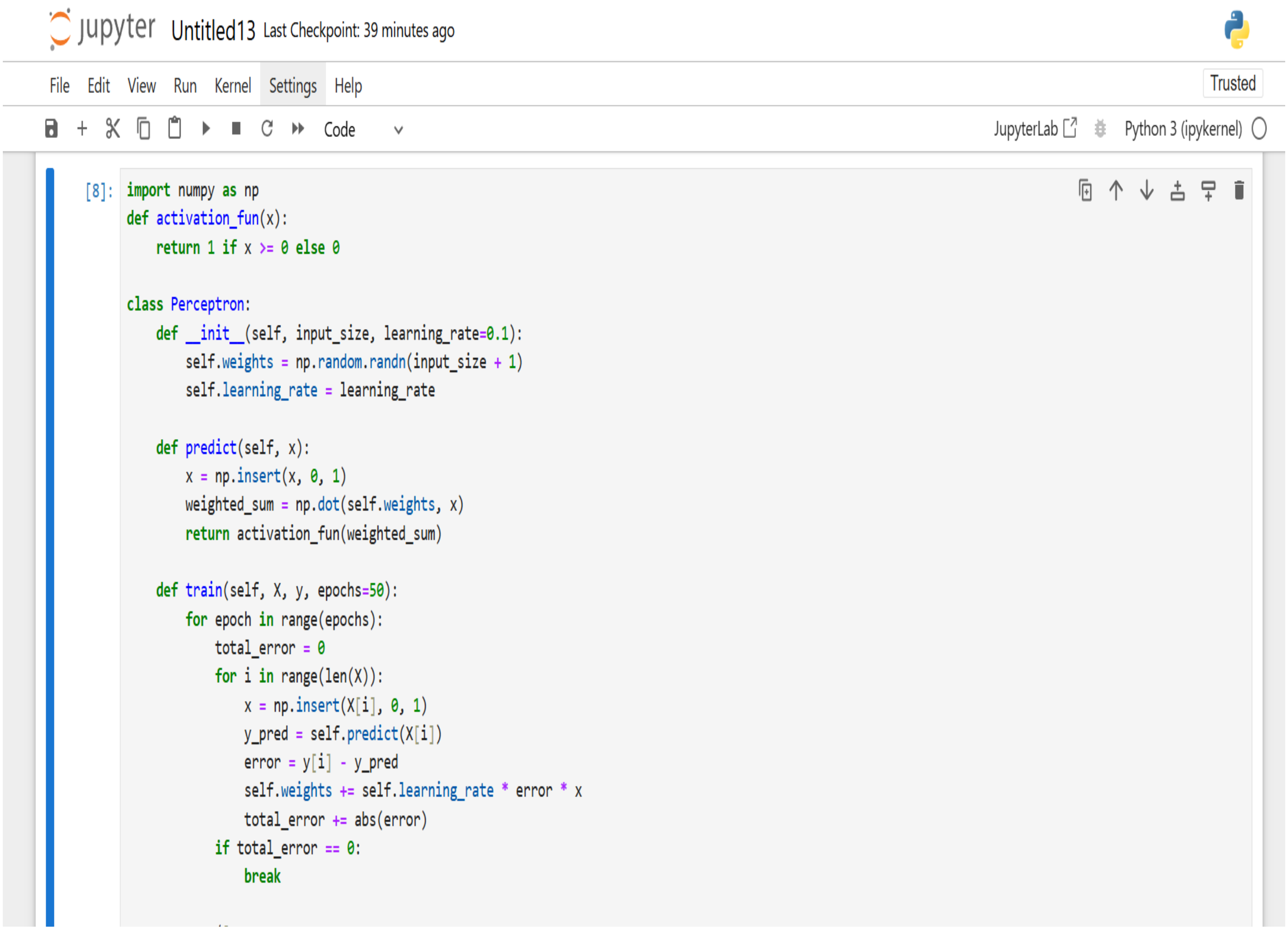
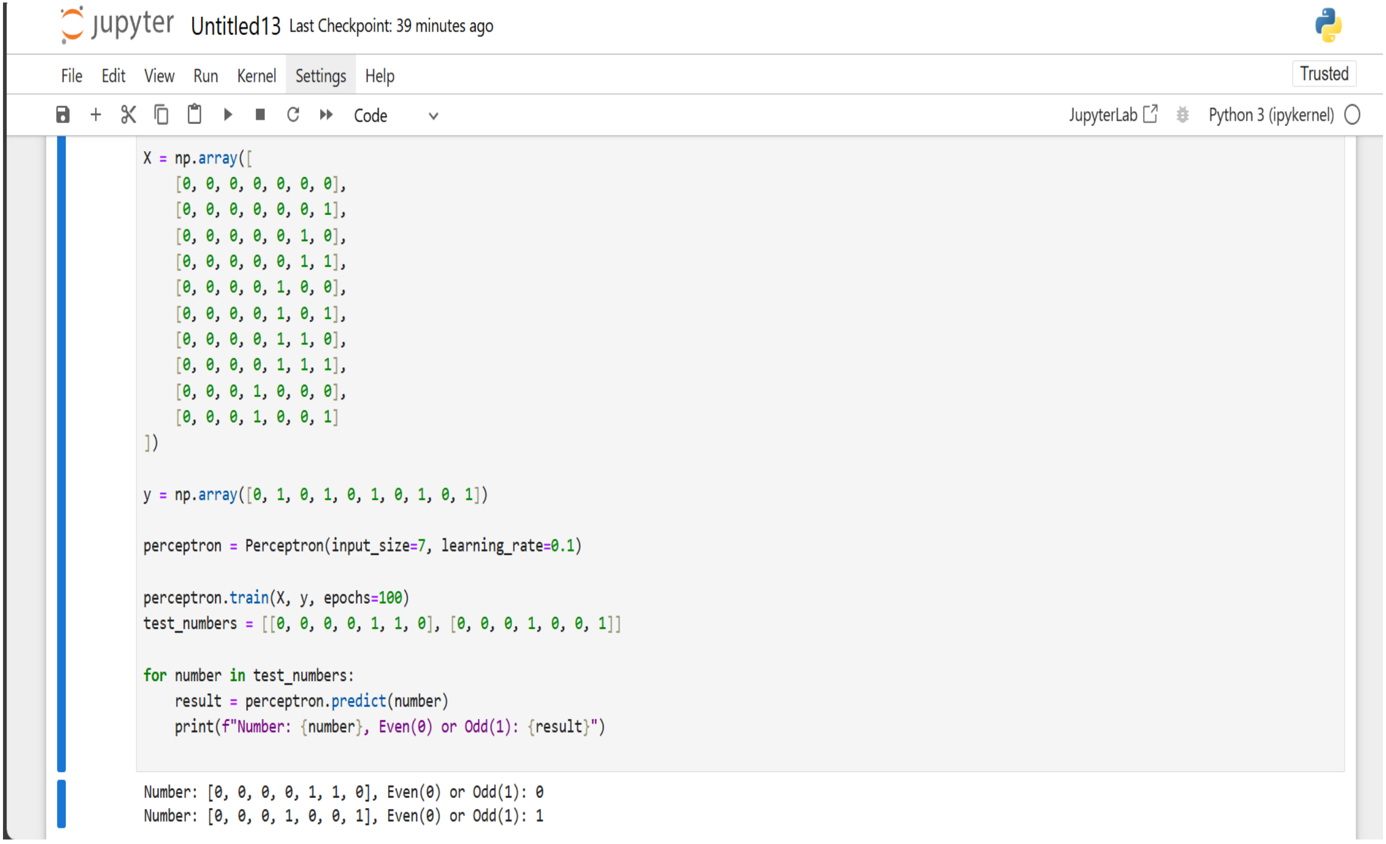


```
import numpy as np
def activation_fun(x):
    return 1 if x >= 0 else 0

class Perceptron:
    def __init__(self, input_size, learning_rate=0.1):
        self.weights = np.random.randn(input_size + 1)
        self.learning_rate = learning_rate

    def predict(self, x):
        x = np.insert(x, 0, 1)
        weighted_sum = np.dot(self.weights, x)
        return activation_fun(weighted_sum)

    def train(self, X, y, epochs=50):
        for epoch in range(epochs):
            total_error = 0
            for i in range(len(X)):
                x = np.insert(X[i], 0, 1)
                y_pred = self.predict(X[i])
                error = y[i] - y_pred
                self.weights += self.learning_rate * error * x
                total_error += abs(error)
            if total_error == 0:
                break
```



```
X = np.array([
    [0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 0, 1, 1],
    [0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 1, 0, 1],
    [0, 0, 0, 0, 1, 1, 0],
    [0, 0, 0, 0, 1, 1, 1],
    [0, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 1]
])

y = np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1])

perceptron = Perceptron(input_size=7, learning_rate=0.1)

perceptron.train(X, y, epochs=100)
test_numbers = [[0, 0, 0, 0, 1, 1, 0], [0, 0, 0, 1, 0, 0, 1]]

for number in test_numbers:
    result = perceptron.predict(number)
    print(f"Number: {number}, Even(0) or Odd(1): {result}")
```

```
Number: [0, 0, 0, 0, 1, 1, 0], Even(0) or Odd(1): 0
Number: [0, 0, 0, 1, 0, 0, 1], Even(0) or Odd(1): 1
```