In [35]:

```
import numpy as np
```

In [36]:

```
class Perceptron:
    def __init__(self, input_size):

        self.weights = np.zeros(input_size)
        self.bias = 0

    def predict(self, inputs):

        summation = np.dot(inputs, self.weights.T) + self.bias
        return 1 if summation >= 0 else 0

    def train(self, inputs, labels, learning_rate, epochs):
        for epoch in range(epochs):
            for input, label in zip(inputs, labels):
                prediction = self.predict(input)
                self.weights += learning_rate * (label - prediction) * input
                self.bias += learning_rate * (label - prediction)
```

In [49]:

```
def ascii_to_binary(ascii_value):
    binary_representation=format(ascii_value,'08b')
    return [int(bit) for bit in binary_representation]
```

In [50]:

```
training_data = []
labels = []

for i in range(10):
    ascii_value = ord(str(i))
    binary_representation = ascii_to_binary(ascii_value)
    training_data.append(binary_representation)
    labels.append(i % 2)
```

In [51]:

```
perceptron=Perceptron(input_size=8)
```

In [52]:

```
learning_rate = 0.1
epochs = 1000
perceptron.train(np.array(training_data), np.array(labels), learning_rate, epochs)
```

In [53]:

```
def ip():
    while True:
        test_number=int(input("Enter Number:"))
        test_number2=test_number%10
        test_input = np.array(ascii_to_binary(ord(str(test_number2))))
        prediction = perceptron.predict(test_input)
        print(f"The perceptron predicts {test_number} is {'odd' if prediction == 1 else '
```

In [54]:

```
ip()
```

```
The perceptron predicts 4 is even.
The perceptron predicts 7 is odd.
```