```python
#!/usr/bin/python3

import sys
from random import random
from math import e

if True or input("Is matplotlib installed? (y/n)").strip() == "y":
    import matplotlib.pyplot as plot
    matplotinstalled = True
else:
    print("View the images in the zip folder.")
    matplotinstalled = False

class Perceptron(object):
    learnRate = .1
    bias = .5
    errorLimit = 0.0001
    maxIterations = 1

    def __init__(self, xarray, outputs, graph_results):
        self.xarray = xarray
        [xline.append(Perceptron.bias) for xline in self.xarray]
        # print(self.xarray)
        self.outputs = [o for o in outputs]
        self.graph_results = graph_results
        self.weights = [random() for i in enumerate(self.xarray[0])]
        #self.train()

    @staticmethod
    def sig(h): # also called g
        return 1.0 / (1.0 + e**(-1.0*h))

    @staticmethod
    def dsig(h): # g'
        return Perceptron.sig(h)*(1-Perceptron.sig(h))

    def plot_error(self):
        if matplotinstalled and self.graph_results:
            fig = plot.figure()
            ax = fig.add_subplot(111)
            ax.plot(self.graph_x, self.graph_y)
            ax.set_ylim(0)
            print("Close the plot to continue")
            plot.show()

    def train(self):
        iterations = 0
        # self.weights = [0random() for i in enumerate(self.xarray[0])]
        # print("Starting Weights: ",self.weights)
        self.graph_error = 1
        self.graph_y = []
        self.graph_x = []
        while self.graph_error > Perceptron.errorLimit and iterations <
        Perceptron.maxIterations:
            iterations+=1
            iteration_errors = []
            for xcase,o in zip(self.xarray,self.outputs):
                if not self.classifiesCorrectly(xcase, o):
                    y = sum([w*x for w,x in zip(self.weights,xcase)])
                    graph_error = (o-y)**2
                    iteration_errors.append(graph_error)
                    error = o-y
                    delta_weight = Perceptron.dsig(y)
                    self.weights = [w+Perceptron.learnRate*error*delta_weight*x for w,x
                    in zip(self.weights,xcase)]
                else:
                    error = 0
```

```
66              error_sum = sum(iteration_errors)/len(iteration_errors) if
                len(iteration_errors) > 0 else 0
67              self.graph_x.append(iterations)
68              self.graph_y.append(error_sum)
69              if Perceptron.maxIterations <= 100:
70                  print(iterations, error_sum)
71          self.plot_error()
72
73      def getValue(self, xinput):
74          xinput.append(Perceptron.bias)
75          out = sum([x*w for x,w in zip(xinput, self.weights)])
76          return out > .5
77
78      def classifiesCorrectly(self, xcase, o):
79          value = self.getValue(xcase)
80          return value == bool(o-1)
81
82      def getY(self, x):
83          print('Weights: ', self.weights)
84          wx = self.weights[0]
85          wy = self.weights[1]
86          wb = self.weights[2]
87          return (.5 - wx*x - wb*Perceptron.bias)/wy
88
```