```
import numpy as np
import random

misc = 0
out = 0
total = 0

for t in range(1000):

    n = 10
    counter = 0

    p1x = np.random.uniform(-1,1) #pick 2 random points and find the equation of line going
    through them
    p1y = np.random.uniform(-1,1)
    p2x = np.random.uniform(-1,1)
    p2y = np.random.uniform(-1,1)
    f = np.array([(-(p2x-p1x)+p1x*(p2y-p1y)),(p1y-p2y),(p2x-p1x)]) #correct f in terms of (w0,
    wx, wy)
    points = np.zeros([n, 3])
    sample = np.zeros([1000, 3])
    pla = np.zeros([n, 5])

    for i in range(n):
        points[i,0] = 1 #set all x0 to 1
        points[i,1] = np.random.uniform(-1,1) #set all x1 to the x-coordinate of point
        points[i,2] = np.random.uniform(-1,1) #set all x2 to the y-coordinate of point
        pla[i,0] = points[i,0] #different form for the PLA implementation later
        pla[i,1] = points[i,1]
        pla[i,2] = points[i,2]
        pla[i,3] = np.sign(np.dot(f,[pla[i,0],pla[i,1],pla[i,2]]))

    for i in range(1000): #setup 1000 sampling points
        sample[i,0] = 1
        sample[i,1] = np.random.uniform(-1,1)
        sample[i,2] = np.random.uniform(-1,1)

    y = np.sign(np.dot(points,f)) #calculate y
    z = np.sign(np.dot(sample,f)) #calculate y for data in sampling points
    w =
    np.dot(np.dot(np.linalg.inv(np.dot(np.matrix.transpose(points),points)),np.matrix.transpose(p
    oints)),y) #calculate w using linear regression
    g = np.sign(np.dot(points,w)) #calculate the output our function predicts
    h = np.sign(np.dot(sample,w)) #same calculation but for sampling points

    clasf = []

    while len(clasf) != n: #repeat PLA until all points are classified
        counter += 1
        clasf = []
        mclasf = []

        for i in range(n):
```

```
            pla[i,4] = np.sign(np.dot(w,[pla[i,0],pla[i,1],pla[i,2]])) #classify according to w
            if pla[i,3] == pla[i,4]: #set point as classified or misclassified
                clasf += [i]
            else:
                mclasf += [i]
        if len(mclasf) == 0:
            break
        w = w +
        pla[mclasf[random.randint(0,len(mclasf)-1)],3]*pla[mclasf[random.randint(0,len(mclasf)-1)
        ],0:3]

    misc += (100-np.sum(np.dot(g,y)))/2 #find the total number of points misclassified by
    regression
    out += (1000-np.sum(np.dot(h,z)))/2 #same but for sampling points

    total += counter #counter for iterations of PLA

print(misc/100000)
print(out/1000000)
print(total/1000)
```