```python
from random import *
import math
import matplotlib.pyplot as plt # Import plot
import numpy as np

tries = 100000 # 10^5 tires
ASpacing = 1 # set the spacing between the first set of parallel lines
BSpacing = 3 # set the spacing between the second set of parallel lines
topAngle = math.atan(ASpacing / BSpacing) # find the top angle in radians
CSpacing = 0.3 * math.sqrt(10) # set the spacing between the thired set of parallel lines
prob = 0.5
longrun = [] # for storing all long-run values
a = 0.1
b = 0.5
length = np.arange(a, b, 0.01)
fig = plt.figure()

def findmean(list):
    v = 0
    for el in list:
        v += float(sum(el)) / max(len(el), 1)
    return v / max(len(list), 1)

while abs(b - a) > 0.015:
    longrun = []
    length = np.arange(a, b, 0.01)
    if abs(b - a) <= 0.1:
        length = np.arange(a, b, 0.001)
    for needleLength in length: # Test length from 0.1 to 0.5
        estimation = [] # for storing the estimations
        hits = 0 # for keeping track of the number of needles that cross a line
        for needle in range(0,tries): # one needle per try
            # Initialize the (x, y) position of one end of the needle at random
            # Assuming the needle is put uniformly across the board
            # z is the diagnol-perspective position
            x = random() * ASpacing
            y = random() * BSpacing
            z = random() * CSpacing

            # Choose the angle that the needle makes with the horizontal in radians
            # we assume it is uniformly distributed in [0,2pi] radians
            angle = random() * 2 * math.pi

            # Use the generated angle and needle length to find (x1, y1) position of the other
            end
            x1 = x + math.cos(angle) * needleLength
            y1 = y + math.sin(angle) * needleLength
            z1 = z + math.cos(angle - topAngle) * needleLength
            # check if the needle cross lines in set A
            if x1 > ASpacing or x1 < 0:
                hits += 1
            # then check if the needle cross lines in set B
            elif y1 > BSpacing or y1 < 0:
                hits += 1
            # finally check if the needle cross lines in set C
            elif z1 > CSpacing or z1 < 0:
                hits += 1

            # For every try, record the estimated probablity
            estimation.append(hits / (needle + 1))
        longrun.append(estimation)
    if findmean(longrun) > prob:
        b = (a + b) / 2
    elif findmean(longrun) <= prob:
        a = (a + b) / 2
```

```
65
66  plt.boxplot(longrun)
67  plt.ylim((prob*0.99, prob*1.01))
68  plt.xticks(range(len(length)), length, rotation=45, fontsize=8)
69  plt.xlabel("Length of the needle")
70  plt.ylabel("Probability found")
71  plt.title("Relationships between the Probability and Needle Length")
72  fig.savefig('p2.png', dpi = fig.dpi)
73  print("The reasonable range of needle length is (" + str(min(length)) + ", " + str(max(length))
    + ").")
74
75  #============= Output=============================================================================#
76  # [Running] python "c:\Users\Johnnia\Desktop\46\Fall 2017\Math381\HW4\tempCodeRunnerFile.py"
77  # The reasonable range of needle length is (0.4, 0.412).
78
79  # [Done] exited with code=0 in 75.832 seconds
80  #=================================================================================================#
```