```
#This code estimates the area under the curve of a function using Monte Carlo methods
#python libraries used for this code
import numpy as np
import matplotlib.pyplot as plt
import random
import math
#start_time = time.time()
N_iterations = 100000 #define n number of iterations - higher number returns closer approximation
Under_curve_x = [] #define empty array called circle to later store (x,y) variables/dots inside circle radius
Under_curve_y = []
In_domain_x = []
In_domain_y = []
area_inside_curve = 0
outside_curve = 0
#create random number generator
for i in range(N_iterations):
    #using random.uniform because it returns random floating point numbers within defined range
    x = random.uniform(0,(2*math.pi)) #random variables for x in range 0 2pi
    y = random.uniform(0, 4.8) #random variables for y in range 0, 4.8
    #adding x,y points to array and counter if pass selection criteria
    if (y) <= (x*math.sin(x))*(math.sin(x)):
       Under_curve_x.append(x)
       Under_curve_y.append(y)
       area_inside_curve += 1.0
    else:
        In_domain_x.append(x)
        In_domain_y.append(y)
    outside_curve += 1.0
Area_estimate = ((float(area_inside_curve)/float(outside_curve)) * 2 * np.pi * 4.8)
print(Area_estimate)
Area = np.pi * np.pi
print(Area)
error = Area_estimate - Area
print(error)
#Plot dots inside the circle
plt.figure(figsize=(5, 5)) #sets size of figure
#plot under curve
plt.scatter(Under_curve_x, Under_curve_y, marker="*", alpha=1.0, color = "purple", label="Inside Area Points")
#plot outside curve
plt.scatter(In_domain_x, In_domain_y, marker="*", alpha=1.0, color = "blue", label="Outside Area Points")
plt.xlabel("Random X-Values")
plt.ylabel("Random Y-Values")
plt.legend(loc="upper left")
#plt.savefig() #set path to save figure
```