1 -

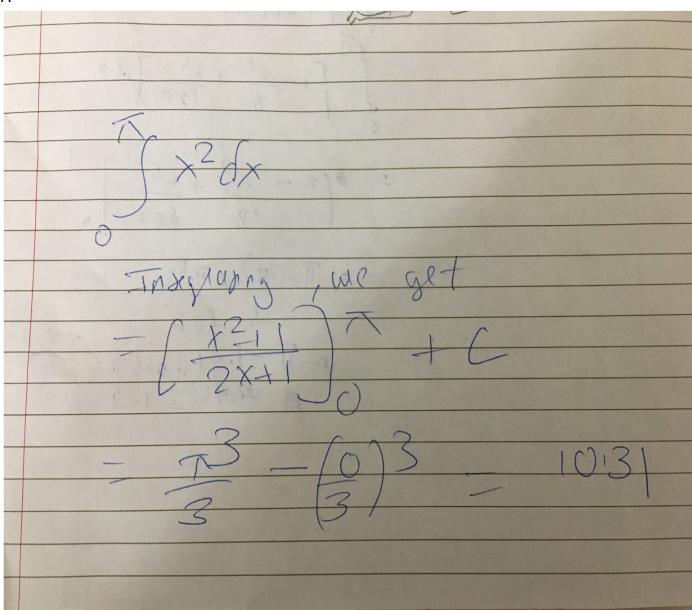
```
import math #importing math library
import random #importing library for random numbers
#the function for the area under the graph
#it changes from graph to graph and depends on the the curve
def f(x): #function defination
  return math.sin(x)**3+math.cos(x)
# define any xmin-xmax interval here! (xmin < xmax)
xmin = 0.0 #lower limit of x
xmax = 3.0 * math.pi #upper limit of x
# find ymin-ymax
numSteps = 1000000 # number of steps
ymin = f(xmin) #the minimum number of steps in the y axis using function f
ymax = ymin
for i in range(numSteps):
  x = xmin + (xmax - xmin) * float(i) / numSteps
  y = f(x)
  if y < ymin: ymin = y
  if y > ymax: ymax = y
# Monte Carlo
rectArea = (xmax - xmin) * (ymax - ymin) #the area of the graph
numPoints = 1000000 # number of points to observe on the graph
ctr = 0 #assigning the counter to zero
for j in range(numPoints): #loop runs from 0 to
number of points
  x = xmin + (xmax - xmin) * random.random() #finding the x coordinate
  y = ymin + (ymax - ymin) * random.random() #random.random to increase randomness
  if math.fabs(y) \leq math.fabs(f(x)):
    if f(x) > 0 and y > 0 and y <= f(x):
       ctr += 1 # area over x-axis is positive
    if f(x) < 0 and y < 0 and y >= f(x):
       ctr -= 1 # area under x-axis is negative
# this is the main part
# we calculate the area here
```

this is the same algorithm we use to calculate the monte carlo approximate numerical solution without function

fnArea = rectArea * float(ctr) / numPoints #calculating the area around the number of points print("Numerical integration = " + str(fnArea)) #printing the numeric value of area

2 -

A -



b) I suck at Calculus - it's too diffcult

$$\int_{0}^{\frac{\pi}{3}} \sin^{4} d^{x}$$

$$\int_{0$$