Question 1.

a.

import math

import numpy as np

def main (a, b, maxiter, tol):

m = 1

x = np.linspace(a, b, m+1)

y = f(x)

approx = np.trapz(y,x)

print(" m integral approximation")

print(m, end="")

print(" "\*12, end="")

print('%.10f' % approx)

i = 1

n = 1

while (i < maxiter):

m = 2 \*\* n

n = n + 1

oldapprox = approx

x = np.linspace (a, b, m+1)

y = f(x)

approx = np.trapz(y,x)

print(m, end="")

print(" "\*12, end="")

print('%.10f' % approx)

if (np.abs((approx - oldapprox)/approx) < tol):

return

i = i + 1

print("Did not converge in",end="")

print(maxiter, end="")

print("iterations")

Text

Description automatically generated

b.

Part 1

Using code from part A with following added:

def f(i):

ans = []

for x in i:

y = (x \* (math.cos(1/x)))

ans.append(y)

return ans

if \_\_name\_\_ == '\_\_main\_\_':

main(0.1, 3, 20, 0.00001)

A screenshot of a computer

Description automatically generated with medium confidence

Output in terminal:

Puneets-MacBook:A6 puneetgrewal$ python3 trap\_1b\_part1.py

m integral approximation

1 3.9888973448

2 3.7902074408

4 3.5976493493

8 3.4808457876

16 3.4678411685

32 3.4856113710

64 3.4877924488

128 3.4870325249

256 3.4867926880

512 3.4867333190

1024 3.4867185769

Text

Description automatically generated

Part 2

def f(i):

ans = []

for x in i:

y = (((math.e)\*\*(3\*x))\*(math.sin(((x+1)\*\*0.5)+1)))

ans.append(y)

return ans

if \_\_name\_\_ == '\_\_main\_\_':

main(-1, 1, 20, 0.0000001)

A screenshot of a computer

Description automatically generated with medium confidence

Output in terminal:

Puneets-MacBook:A6 puneetgrewal$ python3 trap\_1b\_part2.py

m integral approximation

1 13.3970553517

2 7.6078251027

4 5.6929741681

8 5.1698664471

16 5.0360666322

32 5.0024583324

64 4.9940594943

128 4.9919647293

256 4.9914430366

512 4.9913133379

1024 4.9912811709

2048 4.9912732205

4096 4.9912712651

8192 4.9912707877

Text

Description automatically generated

Question 2.

By using trap formula and code from part 1 and following function to approximate I from 0.02 to 1.

def f(i):

ans = []

for x in i:

y = ((math.log((1/x)))\*\*0.5)

ans.append(y)

return ans

if \_\_name\_\_ == '\_\_main\_\_':

main(0.02, 1, 20, 0.000001)

A screenshot of a computer

Description automatically generated with medium confidence

Output in terminal:

m integral approximation

1 0.9691628984

2 0.8866635642

4 0.8555515331

8 0.8452356655

16 0.8424329168

32 0.8419278867

64 0.8419495931

128 0.8420224566

256 0.8420664315

512 0.8420867040

1024 0.8420950651

2048 0.8420983205

4096 0.8420995464

8192 0.8420999985

Hence, 0.8420999985 is used as the value.

The 3 quadrature points used in the Newton Cotes formula are 0, 0.01 and 0.02 since we are approximating from 0 to 0.02.

Using h = (b-a)/(n+2), I calculated **0.0437**.

Adding that to 0.8420999985 gives 0.88580 in which 3 significant digits are correct from the original value of 0.886227 after rounding off.