import numpy as np

import math

def f(x):

    return (math.cos(2\*x) + math.exp(x\*\*2) + 3) / (4 + math.sin(4\*x))

def SimpsonRuleApprox(x\_0, x\_2, h):

    return (h/3)\*(f(x\_0) + 4\*f((x\_0 + x\_2)/2) + f(x\_2))

def SimpsonRuleComposite(a, n, h):

    total = 0

    for j in range(0, n):

        total += SimpsonRuleApprox(a + j\*h, h\*(1+j), h)

    total = total / 2

    return total

def abs\_error\_specific(computed):

    true = 2.5289360549014543

    return abs(true - computed)

def R(k, j, a, b):

*# Break out condition*

    if j == 1:

        return R\_1(k, a, b)

*# Recursive formula for Rhomerg greater than 1*

    val = R(k, j - 1, a, b) + (4\*\*(j - 1) - 1)\*\*(-1) \* \

            (R(k, j - 1, a, b) - R(k - 1, j - 1, a, b))

    return val

def R\_1(k, a, b):

*# Defining h*

    h = b - a

*# Finding middle sum*

    total = 0

    for i in range(0, 2\*\*(k-1)-1):

        total += f(a + (i+1)\*(h / 2\*\*(k - 1)))

*# Caculating R*

    R = (h/2\*\*(k))\*(f(a) + f(b) + 2\*total)

    return R

def f\_transform(y, a1, a2):

    x = a1 + a2\*y

    return a2\*f(x)

def interval\_transform(a, b):

    a\_arr = np.array([[1, -1], [1, 1]])

    b\_vec = np.array([a, b])

    a1, a2 = np.linalg.solve(a\_arr, b\_vec)

    return a1, a2

def LegrengeRoots(n):

    roots = np.genfromtxt(

        "C:/Users/user/Documents/School/2019-2020/APPM 4650/Roots.csv", delimiter=",")

    roots[0][0] = 1

    nodes = np.zeros((n - 1))

    weights = np.zeros((n - 1))

    ind = -1

    for i in range(0, roots.shape[0]):

        if roots[i, 0] == n:

            ind = i

    nodes = roots[ind:ind + n, 1]

    weights = roots[ind:ind + n, 2]

    return nodes, weights

def Guassian(a, b, n):

    a1, a2 = interval\_transform(a, b)

    nodes, weights = LegrengeRoots(n)

    f\_arr = np.zeros((len(nodes)))

    for i in range(0, len(nodes)):

        f\_arr[i] = f\_transform(nodes[i], a1, a2)

    I = np.dot(weights, f\_arr)

    return I

def NinePtNewtonCotes(x\_0, x\_8):

    h = (x\_8 - x\_0) / 9

    I = (4\*h/14175)\*(989\*f(x\_0) + 5888\*f(x\_0 + h) - 928 \*

                  f(x\_0 + 2\*h) + 10496\*f(x\_0 + 3\*h) - 4540\*f(x\_0 + 4\*h) +

                  10496\*f(x\_0 + 5\*h) - 928\*f(x\_0 + 6\*h) + 5888\*f(x\_0 + 7\*h) + 989\*f(x\_0 + 8\*h))

    return I

def main():

*# Bounds*

    a = -1

    b = 1

*# Simpsons Composite =============================================*

    n = 8

    h = (b - a) / n

    Sim\_approx = SimpsonRuleComposite(a, n, h)

    print('Simpsons Composite:', Sim\_approx,'| Error:', abs\_error\_specific(Sim\_approx))

*# Romberg Integration =============================================*

    Rom\_approx = R(4,4, a, b)

    print('Romberg:', Rom\_approx,'| Error:', abs\_error\_specific(Rom\_approx))

*# Guassian Quadrature ================================================*

    n = 9

    Guass\_approx = Guassian(a, b, n)

    print('Guassian:', Guass\_approx,'| Error:', abs\_error\_specific(Guass\_approx))

*# Nine Pt Newton Cotes ================================================*

    Nine\_NC\_approx = NinePtNewtonCotes(a, b)

    print('Nine Pt Newton Cotes:', Nine\_NC\_approx,'| Error:', abs\_error\_specific(Nine\_NC\_approx))

if \_\_name\_\_ == "\_\_main\_\_":

    main()