```
"""
Registration : xxxx
Description  : Limiting representation of Dirac Delta function
Author       : AKB
"""

import numpy as np
from scipy.integrate import quad, simps
from scipy.special import airy, jn, factorial, laguerre, hermite
import warnings
warnings.filterwarnings("ignore")

#Logical switch for different representations to choose from:
rep1,  rep2,  rep3,  rep4,  rep5,  rep6,  rep7,  rep8   = 1, 1, 1, 1, 1, 1, 1, 1
rep9,  rep10, rep11, rep12, rep13, rep14, rep15, rep16  = 1, 1, 1, 1, 1, 1, 1, 1

#x1,x2,a=input("Enter the integration limits, x1,x2,a and N")
x1, x2, a = 0.5, 1.0, 2.0
ll, ul = (a-x1), (a+x2)
N = 10000; x = np.linspace(-10.0, 10.0, N)

# If delta representation multiplied with x^2 and integrated will return a^2
def f(x): return x**2

print("The results for each representation of the delta function (for
x1=",x1,",x2=",x2,",a=",a,") are given below:-")

#delta(x-a)=lim_(eps\tendsto 0) (1/2)theta(1+x-a)theta(1-x+a)eps|(x-a)|^(eps-1)
if(rep1):
    def delta1(x):
        eps=0.001
        return 0.5*np.heaviside(1+x-a,1.0)*np.heaviside(1-x+a,1.0)*eps*(abs(x-
a))**(eps-1)*f(x)
    I,acc=quad(delta1,ll,ul)
    print("Representation01: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(l\tendsto\inf) (2l+1)!!/(2^(l+1)*l!) (1-(x-a)^2)^l
if(rep2):
    def delta2(x):
        l=0.01
        return factorial(factorial(2*l+1))/(2**(l+1)*factorial(l))*(1-(x-a)**2)**l*f(x)
    I,acc=quad(delta2,ll,ul)
    print("Representation02: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0) (1/2*eps)e^(-abs(x)/eps)
if(rep3):
    def delta3(x):
        eps=0.001
        return np.exp(-abs(x-a)/eps)/(2*eps)*f(x)
    I, acc=quad(delta3,ll,ul)
    print("Representation03: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0)1/2*sqrt(pi*eps) exp(-(x-a)^2/4*eps)
if(rep4):
    def delta4(x):
        eps=0.001
        return np.exp(-(x-a)**2/(4.0*eps))/(np.sqrt(4.0*np.pi*eps))*f(x)
    I,acc=quad(delta4,ll,ul)
    print("Representation04: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


# delta(x-a)=lim_(eps\tendsto 0)1/sqrt(2*pi*i*eps) exp(i*(x-a)^2/2*eps)
if(rep5):
```

```
    def delta5(x):
        eps = 0.001
        return 1/np.sqrt((2*np.pi*1.0j*eps))*np.e**(1.0j*(x-a)**2/(2*eps))*f(x)

    I,acc=quad(delta5,ll,ul)
    print("Representation05: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0)1/pi*(x-a) sin((x-a)/eps)
if(rep6):
    def delta6(x):
        eps=0.01
        return (np.sin((x-a)/eps)/(np.pi*(x-a)))*f(x)
    I,acc=quad(delta6,ll,ul)
    print("Representation06: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(n\tendsto\inf) 1/2pi sin((n+1/2)(x-a))/sin((x-a)/2)
if(rep7):
    def delta7(x):
        n=100
        return (np.sin((n+0.5)*(x-a))/np.sin((x-a)/2))/(2*np.pi)*f(x)
    I,acc=quad(delta7,ll,ul,
points=a)                                                    #'points=a' excludes the
point x=a from the domain of integration
    print("Representation07: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with
accuracy=",acc)  # since there is a singularity at x=a


#delta(x-a)=lim_(eps\tendsto 0) [2*eps*cosh^2((x-a)/eps)]^-1
if(rep8):
    def delta8(x):
        eps=0.01
        return 1/(2*eps*(np.cosh((x-a)/eps))**2)*f(x)
    I,acc=quad(delta8,ll,ul)
    print("Representation08: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(p\tendsto\inf) log|coth(p(x-a))|
if(rep9):
    def delta9(x):
        p=2.3
        return np.log(abs(1/np.tanh(p*(x-a))))*f(x)
    I,acc=quad(delta9,ll,ul, points=a)
    print("Representation09: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0) 1/eps Ai((x-a)/eps)
if(rep10):
    def delta10(x):
        eps=0.1
        return (airy((x-a)/eps)[0]/eps)*f(x)
    I,acc=quad(delta10,ll,ul)
    print("Representation10: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0) 1/eps j_(1/eps)((x-a+1)/eps)
if(rep11):
    def delta11(x):
        eps=0.0016
        return jn(1/eps,(x-a+1)/eps)*f(x)/eps
    I,acc=quad(delta11,ll,ul)
    print("Representation11: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0) |1/eps exp(-(x-a)^2/eps)L_n(2*(x-a)/eps)|
```

```
if(rep12):
    def delta12(x):
        eps, n = 1.25, 2   # n=Order of Laguerre Polynomial
        return np.abs((1.0/eps)*np.exp(-(x-a)**2/eps)*laguerre(n)(2*(x-a)/eps))*f(x)
    I,acc=quad(delta12,ll,ul)
    print("Representation12: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0) 1/eps*sinh(x/eps)
if(rep13):
    def delta13(x):
        eps = 0.2
        return (1.0/np.sinh((x-a)/eps))*f(x)/eps
    I,acc=quad(delta13,ll,ul)
    print("Representation13: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0) |1/sqrt(eps*pi) exp(-(x-a)^2/eps)(-1/sqrt(eps))^n*H_n((x-
a)/sqrt(eps))|
if(rep14):
    def delta14(x):
        eps, n = 0.01, 1  # n=Order of Hermite Polynomial
        return -1.0/(np.sqrt(np.pi*eps))*np.exp(-(x-a)**2/eps)*(-1.0/
np.sqrt(eps))**n*hermite(n)((x-a)/np.sqrt(eps))*f(x)
    I,acc=quad(delta14,ll,ul)
    print("Representation14: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0)1/pi eps/((x-a)^2+eps^2)
if(rep15):
    def delta15(x):
        eps=0.001
        return eps/(np.pi*((x-a)**2+eps**2))*f(x)
    I,acc=quad(delta15,ll,ul)
    print("Representation15: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#delta(x-a)=lim_(eps\tendsto 0)1/2 eps*|x-a|^(eps-1)
if(rep16):
    def delta16(x):
        eps=0.001
        return 0.5*eps*(np.abs(x-a))**(eps-1)*f(x)
    I,acc=quad(delta16,ll,ul)
    print("Representation16: Int_",ll,"^",ul," x^2 delta(x) dx =",I," with accuracy=",acc)


#------------------RESULTS--------------------#
'''
The results for each representation of the delta function (for x1= 0.5 ,x2= 1.0 ,a= 2.0 )
are given below:-
Representation01: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 3.99992618334305  with accuracy=
1.2761562029339757e-08
Representation02: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 3.9222356027428975  with accuracy=
7.044957506252558e-09
Representation03: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 4.000002000000023  with accuracy=
1.8460248440987925e-08
Representation04: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 4.002000000000001  with accuracy=
1.7122039197411215e-09
Representation05: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 3.977049282889976  with accuracy=
0.00044401018247540306
Representation06: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 3.961669294270765  with accuracy=
6.417255619598747e-13
Representation07: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 3.9559172668356237  with accuracy=
1.1631910375219448e-09
Representation08: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 4.000082246703354  with accuracy=
4.173830539001833e-09
```

```
Representation09: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 4.25521049330483  with accuracy=
3.7125857943465235e-13
Representation10: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 4.093852274807035  with accuracy=
5.3754297511815355e-12
Representation11: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 3.997833767397869  with accuracy=
1.6956608728499267e-08
Representation12: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 3.987816323572433  with accuracy=
1.0663348965186618e-09
Representation13: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 4.094084101531259  with accuracy=
1.7932322293745528e-12
Representation14: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 4.000000000174033  with accuracy=
1.074231062927064e-08
Representation15: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 3.9975392914604457  with accuracy=
5.368096318749804e-10
Representation16: Int_ 1.5 ^ 3.0  x^2 delta(x) dx = 3.99992618334305  with accuracy=
1.2761562029339757e-08
'''
```