```
from sympy import* #Call Library of sympy
x = symbols('x')   #Make x a symbol
f = x**2 *exp(-x)    #Function to take derivative

#df = diff(f, x,1)    #diff(f,x,1) is used to take first derivative of f w.r.t x
df2 = diff(f, x,2)    #diff(f,x,1) is used to take nth derivative of f w.r.t x

print(df2)
```

```
(x**2 - 4*x + 2)*exp(-x)
```

```
# another procedure for finding  derivative
y=x**2 *exp(-x)
derivative_y=y.diff(x) #differentiate y w.r.t x
derivative2 = derivative_y.diff(x)
print(derivative2)
```

```
x**2*exp(-x) - 4*x*exp(-x) + 2*exp(-x)
```

```
from numpy import array
```

```
df1=lambdify(x,df2) #now numpy function
print(df1(1)) #evaluated at x=1
y= array([1,2,3,4]) #evaluated at array of 1,2,3,4
print(df1(y))
```

```
-0.36787944117144233
[-0.36787944 -0.27067057 -0.04978707  0.03663128]
```

```
# code of backward difference formula.

import numpy as np
from tabulate import tabulate


def backward_diff(x, y):

    # Compute the step size h
    h = x[1] - x[0]
    data=[]

    # Compute the backward difference approximation
    bdf = np.zeros_like(y)
    bdf[0] = (y[1] - y[0]) / h  # use forward difference at the first point
    for i in range(len(y) - 1):
        bdf[i] = (y[i] - y[i-1]) / h
        data.append([x[i],y[i],bdf[i]])
    data.append([x[0],y[0],bdf[0]])

    print(tabulate(data,headers=['x','f(x)','df(x)/dx'],tablefmt="github"))
```

```
    return


# example to run above code
x=[0.2,0.4,0.6,0.8]
y=[3,3.9,3.98,4.2]
backward_diff(x, y)
```

```
|   x |   f(x) |   df(x)/dx |
|-----|--------|------------|
| 0.2 |   3    |       -6   |
| 0.4 |   3.9  |        4.5 |
| 0.6 |   3.98 |        0.4 |
| 0.2 |   3    |       -6   |
```

```
def fivePoint(x, y):
    if len(x) < 5:
      raise Exception("Less than five points given")
    # Compute the step size h
    data=[]
    h = x[1] - x[0]

    # Compute the forward difference approximation
    tp = np.zeros_like(y)
    tp[0]=(-25*y[0]+48*y[1]-36*y[2]+16*y[3]-3*y[4])/(12*h) #five point endpoint (left end)
    tp[-1]=(25*y[-1]-48*y[-2]+36*y[-3]-16*y[-4]+3*y[-5])/(12*h) #five point endpoint (righ
    data.append([x[0],y[0],tp[0]])
    midpoint=int((0+len(x))/2)

    for i in range(1,midpoint):
          tp[i] = (y[i+1] - y[i-1]) / (2*h)
          data.append([x[i],y[i],tp[i]])
    tp[midpoint] = (y[0] - 8*y[1] + 8*y[3] - y[4]) / (12*h)
    data.append([x[2],y[2],tp[2]])

    for i in range(midpoint+1,len(x)-1):
          tp[i] = (y[i+1] - y[i-1]) / (2*h)
          data.append([x[i],y[i],tp[i]])
    data.append([x[-1],y[-1],tp[-1]])

    print(tabulate(data,headers=['x','f(x)','df(x)/dx'],tablefmt="github"))
    return


x=[0.2,0.4,0.6,0.8,1.0]
y=[3,3.9,3.98,4.2,5.0]
fivePoint(x,y)
```

```
|   x |   f(x) |   df(x)/dx |
|-----|--------|------------|
| 0.2 |   3    |   8.8      |
| 0.4 |   3.9  |   2.45     |
| 0.6 |   3.98 |   0.166667 |
| 0.8 |   4.2  |   2.55     |
| 1   |   5    |   5.53333  |
```

```
import math
def f(x):
  return(x**2*ln(x))
def f1(x):
  return (exp(2*x)*sin(3*x))

def simp_rule(f, a, b, n):
  if n%2 != 0:
   raise Exception("n is an odd number")
  h=(b-a)/n
  x=[a+i*h for i in range(n+1)]
  y=[f(xi) for xi in x]
  f1 = y[0] + y[-1]
  for i in range(1,n):
    if i%2==0:
       f1 = f1 + 2*y[i]
    else:
       f1 = f1 + 4*y[i]
  f1= f1 * h/3
  return f1
```

```
import math
#n=2
simp=simp_rule(f, 1, 1.5,2) # simple sinmpson 1/3 rule
print(simp)
#n=4
comp=simp_rule(f,1,1.5,4)# composite simpson 1/3 rule
print(comp)

x = symbols('x')
f = x**2*ln(x)
I_actual = float(integrate(f, (x,1,1.5)))
print(I_actual)
print(I_actual-simp)
print(I_actual-comp)
```

```
0.192245307413098
0.192258460445610
0.19225935773279607
1.40503196976449e-5
8.97287186302220e-7
```

```
simp=simp_rule(f1, 0, 2,2)
print(simp) #simple simpson 1/3 rule
comp=simp_rule(f1,0,2,8)
print(comp) #composite simpson 1/3 rule with n=8

x = symbols('x')
f = exp(2*x)*sin(3*x)

I_actual = float(integrate(f, (x,0,2)))
print(I_actual)
print(I_actual-simp)
```

```
print(I_actual-simp)
print(I_actual-comp)
```

```
-3.69486488910236
-14.1833415614467
-14.213977129862522
-10.5191122407602
-0.0306355684158266
```

```
x = symbols('x')
f = x**4
def errorBoundSimp(f,l,u):
    d4f = diff(f, x,4)
    abs_max_ddf=max(abs(d4f.subs(x,l)),abs(d4f.subs(x,u)))
    h=(u-l)/2
    Error_bound=h**5*abs_max_ddf/90
    return(Error_bound,abs_max_ddf)

errorBoundSimp(f,0.5,1)
```

```
(0.000260416666666667, 24)
```

```
x = symbols('x')
f = 2/(x-4)
def errorBoundSimp(f,l,u):
    d4f = diff(f, x,4)
    abs_max_ddf=max(abs(d4f.subs(x,l)),abs(d4f.subs(x,u)))
    h=(u-l)/2
    Error_bound=h**5*abs_max_ddf/90
    return(Error_bound,abs_max_ddf)

errorBoundSimp(f,0,0.5)
```

```
(9.91650304436643e-7, 0.0913904920568811)
```

0s completed at 11:07 PM