```
"""
Registration : xxxx
Description  : difference formula for f'(a) = [f(a+h)-f(a-h)]/2h (central)
                                            = [f(a+h)-f(a)]/h    (forward)
                                            = [f(a)-f(a-h))/h    (backward)
               Use Sympy for derivative : import sympy as sp
                                          x = sp.symbols('x')
                                          print sp.diff(((4*x**2 + 2*x + 1)/
(x+2*sp.exp(x)))**x,x)
Author       : AKB
"""
import numpy as np
from scipy.misc import derivative
from scipy.special import factorial
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

# Logical case switch for different problems to choose from
prob1=1; prob2=0; prob3=0; prob4=0;

def deriv(f,a,method,h):
  if   method == 'central':  return (f(a+h)-f(a-h))/(2*h)
  elif method == 'forward':  return (f(a+h)-f(a))/h
  elif method == 'backward': return (f(a)-f(a-h))/h
  else:                      raise  ValueError("Wrong choice")

if(prob1):
   # Evaluate f'(x) for f(x)=sin(x), cos(x), exp(x) etc at a point
   h = float(eval(input('Enter h = ')))  # 0.5; check convergence for varying grid size
   x = float(eval(input('Enter the point = '))) # x=0.5
   f = np.sin
   method = 'forward';  dydxf = deriv(f, x, method, h)
   method = 'backward'; dydxb = deriv(f, x, method, h)
   method = 'central';  dydxc = deriv(f, x, method, h)
   print ('Forward difference = ',dydxf, '\n Backward difference = ',dydxb, '\n Central
difference = ',dydxc)

if(prob2):
   # Evaluate f'(x) for f(x)=sin(x), cos(x), exp(x) etc on a line
   h = float(eval(input('Enter h = ')))  # 0.5; check convergence for varying grid size
   x = np.linspace(0,5*np.pi,100)
   f = np.sin
   method = 'forward';  dydxf = deriv(f, x, method, h)
   method = 'backward'; dydxb = deriv(f, x, method, h)
   method = 'central';  dydxc = deriv(f, x, method, h)
   ##=====================================================#
   # Note: use "derivative" module for central difference #
   # x = np.arange(0,5)                                    #
   # dydx = derivative(np.exp, x, h)                       #
   # print dydx                                            #
   ##====================================================##
   dYdx = np.cos(x)

   #Plot
   plt.figure(1)
   plt.plot(x, f(x),  'orange', label='f(x)')
   plt.plot(x, dydxf, 'kx', label='Forward Difference')
   plt.plot(x, dydxb, 'm+', label='Backward Difference')
   plt.plot(x, dydxc, 'r.', label='Central Difference')
   plt.plot(x, dYdx,  'b',  label="True f'(x)")
   plt.title('Derivative of sin(x)', size=16)
   plt.legend(loc='best', prop={'size':12})
   plt.text(12.4,-0.85,'h = '+str(h),size = 20)
   plt.xlabel('x', size = 16); plt.xticks(size = 14)
   plt.ylabel(r'$\frac{df(x)}{dx} = cos(x)$', size = 20); plt.yticks(size = 14)
```

```
    plt.grid()
    #plt.savefig('plot/07_forbackdiff1.pdf')
    plt.show()

if(prob3):
    # Evaluate f'(x) for f(x)=((4x^2+2x+1)/(x+2e^x))^x on a line
    h1, h2 = input('Enter h1 and h2 = ').split()  # 0.005, 0.5
    h1, h2 = float(h1), float(h2)
    x = np.linspace(0, 6, 100)
    def f(x):
        return ((4*x**2 + 2*x + 1)/(x+2*np.exp(x)))**x
    def fp(x):
        return ((1+2*x+4*x**2)/(2*np.exp(x)+x))**x*((x*(-1+4*x**2+np.exp(x)*(2+
               4*(3-2*x)*x)))/((2*np.exp(x)+x)*(1+2*x+4*x**2))+
               np.log((1+2*x+4*x**2)/(2*np.exp(x)+x)))

    dYdx = fp(x) # Theoretical result
    # 1st set
    method = 'forward';  dydxf1 = deriv(f, x, method, h1)
    method = 'backward'; dydxb1 = deriv(f, x, method, h1)
    method = 'central';  dydxc1 = deriv(f, x, method, h1)
    # 2nd set
    method = 'forward';  dydxf2 = deriv(f, x, method, h2)
    method = 'backward'; dydxb2 = deriv(f, x, method, h2)
    method = 'central';  dydxc2 = deriv(f, x, method, h2)

    # Plot
    plt.figure(1)
    plt.subplot(2,1,1)
    plt.plot(x, f(x), 'orange', label='f(x)')
    plt.plot(x, dydxf2, 'kx', label='Forward Difference')
    plt.plot(x, dydxb2, 'm+', label='Backward Difference')
    plt.plot(x, dydxc2, 'r.', label='Central Difference')
    plt.plot(x, dYdx,   'b',  label="True f'(x)")
    plt.suptitle(r'Derivative of $f(x) = (\frac{4x^2 + 2x + 1}{x+2e^x})^x$', size=14)
    plt.legend(loc='best', prop={'size':10})
    plt.text(4.8,-0.6,'h = '+str(h2),size = 14)
    plt.xlabel('x', size = 16); plt.xticks(size = 14)
    plt.ylabel(r'$\frac{df(x)}{dx}$', size = 14); plt.yticks(size = 14)
    plt.grid()

    plt.subplot(2,1,2)
    plt.plot(x, f(x), 'orange', label='f(x)')
    plt.plot(x, dydxf1, 'kx', label='Forward Difference')
    plt.plot(x, dydxb1, 'm+', label='Backward Difference')
    plt.plot(x, dydxc1, 'r.', label='Central Difference')
    plt.plot(x, dYdx,  'b',  label="True f'(x)")
    plt.suptitle(r'Derivative of $f(x) = (\frac{4x^2 + 2x + 1}{x+2e^x})^x$', size=14)
    plt.legend(loc='best', prop={'size':10})
    plt.text(4.5,-0.7,'h = '+str(h1),size = 14)
    plt.xlabel('x', size = 16); plt.xticks(size = 14)
    plt.ylabel(r'$\frac{df(x)}{dx}$', size = 14); plt.yticks(size = 14)
    plt.grid()
    #plt.savefig('plot/07_forbackdiff2.pdf')
    plt.show()

if(prob4):
    # Taylor Series f(x) = 3e^x/(x^2+x+1) : evaluate derivative upto 3rd order at x=0
    # f(x) = f(0) + \sum \frac{f^n(0)}{n!} x^n, n=1,2,3
    h1, h2 = input('Enter h1 and h2 = ').split()  # 0.005, 0.5
    h1, h2 = float(h1), float(h2)
    x = np.linspace(-3, 3, 100)
    def f(x): return 3*np.exp(x)/(x**2 + x + 1)

    # construct the Taylor Polynomial with two different h. order specifies the number
    # of points to use. order must be odd & at least n+1
```

```
a0 = f(0)
h=0.005 # 1st set
a1 = derivative(f,0,h,n=1)
a2 = derivative(f,0,h,n=2)/factorial(2)
a3 = derivative(f,0,h,n=3,order=5)/factorial(3)
P1 = a0 + a1*x + a2*x**2 + a3*x**3
h=0.5   # 2nd set
a1 = derivative(f,0,h,n=1)
a2 = derivative(f,0,h,n=2)/factorial(2)
a3 = derivative(f,0,h,n=3,order=5)/factorial(3)
P2 = a0 + a1*x + a2*x**2 + a3*x**3

#Plot
plt.figure(1)
plt.subplot(2,1,1)
plt.plot(x,f(x),'k-',  label=r'$f(x)$')
plt.plot(x, P2, 'r--', label=r'$3^{rd}$ order polynomial')
plt.xlim([-3,3])
plt.ylim([0,5])
plt.legend(loc='best', prop={'size':16})
plt.text(1.5,0.2,r'$h = 0.5$',size = 20)
plt.xlabel('x', size = 16);
plt.xticks(size = 14)
plt.ylabel(r'$f(x)=3e^x/(x^2+x+1)$', size=16);
plt.yticks(size = 14)
plt.grid()

plt.subplot(2,1,2)
plt.plot(x,f(x),'k-',  label=r'$f(x)$')
plt.plot(x, P1, 'r--', label=r'$3^{rd}$ order polynomial')
plt.xlim([-3,3])
plt.ylim([0,5])
plt.legend(loc='best', prop={'size':16})
plt.text(1.5,0.2,r'$h = 0.005$',size = 20)
plt.xlabel('x', size = 16);
plt.xticks(size = 14)
plt.ylabel(r'$f(x)=3e^x/(x^2+x+1)$', size=16);
plt.yticks(size = 14)
plt.grid()
#plt.savefig('plot/07_Taylor.pdf')
plt.show()
```