ECL 201 SCIENTIFIC COMPUTING LABORATORY

FINAL LAB REPORT

Lab 4: Numerical Differentiation and Integration

EXPERIMENT 1

AIM: To realize the functions sin t, cos t, sinht and cosht for the vector t = [0, 10] with increment 0.01.

CODE:

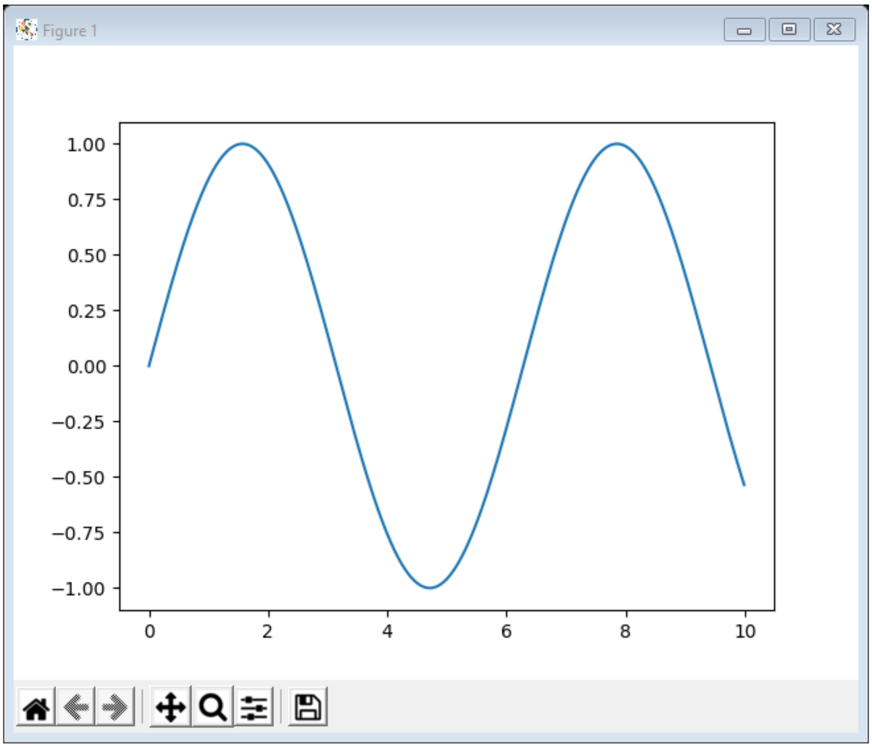
```
#sint
import numpy as np                      #importing numpy
import matplotlib.pyplot as plot        #importing matplotlib.pyplot
t = np.arange(0, 10, 0.01)              #time period and increment
amp = np.sin(t)                         #amplitude
plot.plot(t, amp)
plot.show()                             #plot the function

#sinht
import numpy as np
import matplotlib.pyplot as plot
t = np.arange(0, 10, 0.01)
amp = np.sinh(t)
plot.plot(t, amp)
plot.show()

#cost
import numpy as np
import matplotlib.pyplot as plot
t = np.arange(0, 10, 0.01)
amp = np.cos(t)
plot.plot(t, amp)
plot.show()

#cosht
import numpy as np
import matplotlib.pyplot as plot
t = np.arange(0, 10, 0.01)
amp = np.cosh(t)
plot.plot(t, amp)
plot.show()
```

RESULT:

test35.py

C: > Users > acer > OneDrive > Desktop > python > test35.py > ...

```
#sint
import numpy as np                  #importing numpy
import matplotlib.pyplot as plot    #importing matplotlib.pyplot
t = np.arange(0, 10, 0.01)          #time period and increment
amp = np.sin(t)                     #amplitude
plot.plot(t, amp)
plot.show()                         #plot the function

```

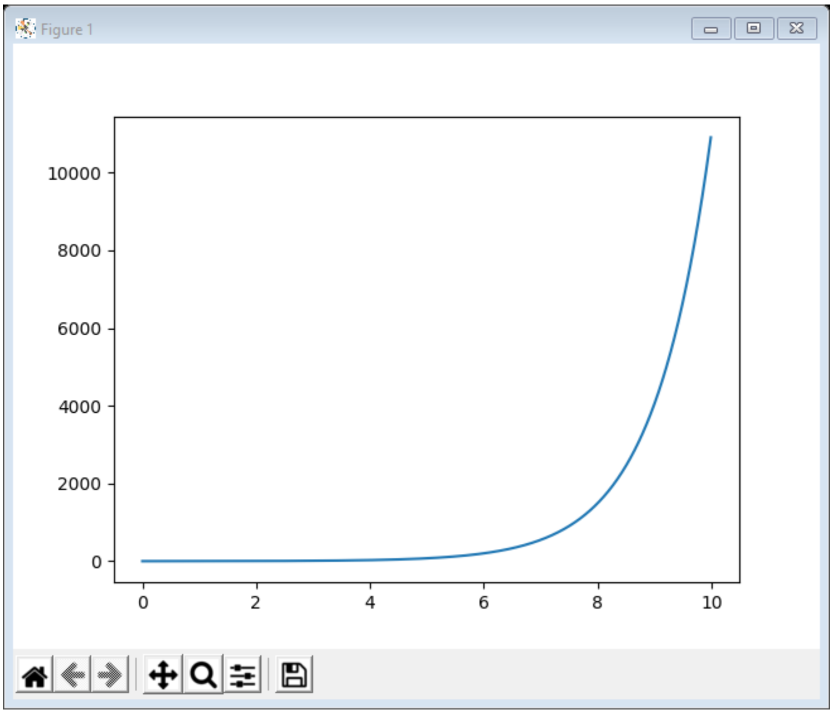


---

test35.py

C: > Users > acer > OneDrive > Desktop > python > test35.py > ...

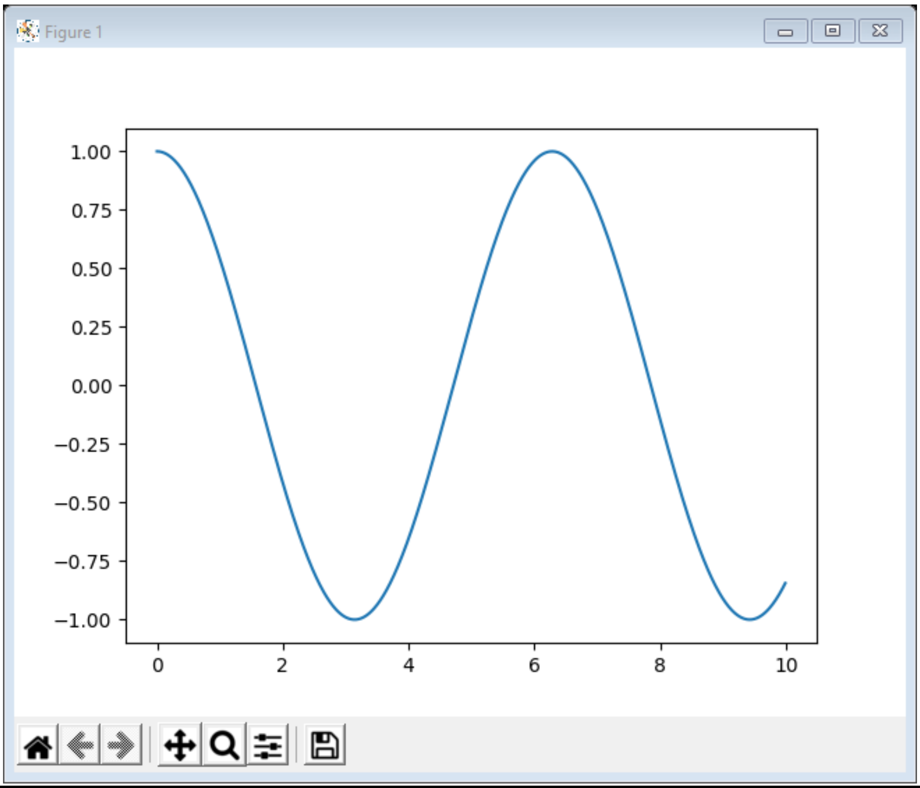
```
#sinht
import numpy as np
import matplotlib.pyplot as plot
t = np.arange(0, 10, 0.01)
amp = np.sinh(t)
plot.plot(t, amp)
plot.show()
```



test35.py

C: > Users > acer > OneDrive > Desktop > python > test35.py > ...

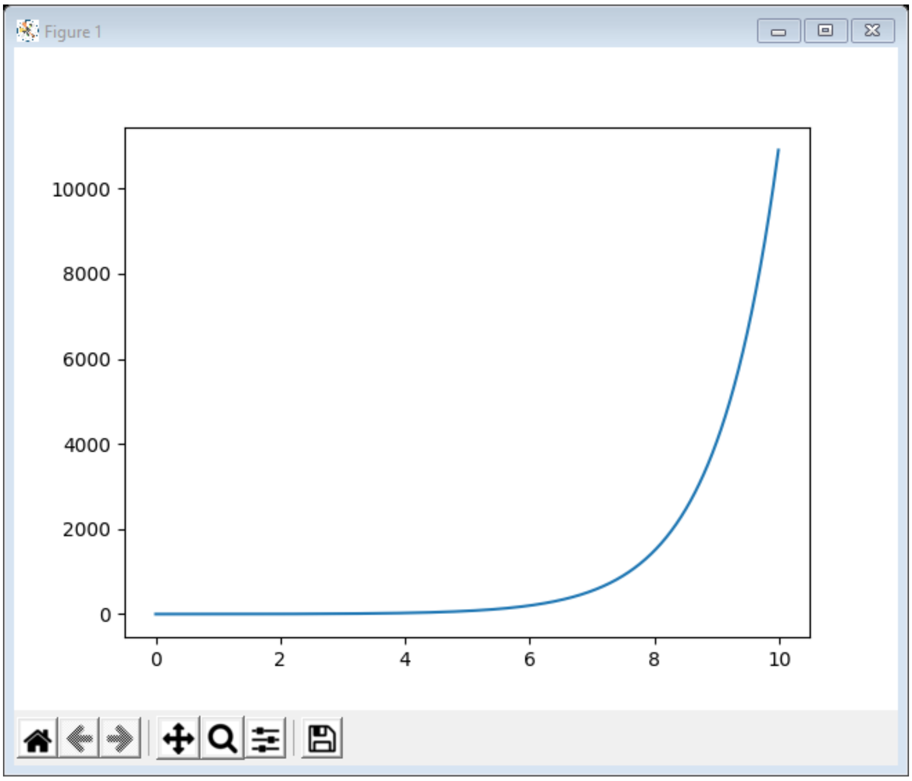
```python
#cost
import numpy as np
import matplotlib.pyplot as plot
t = np.arange(0, 10, 0.01)
amp = np.cos(t)
plot.plot(t, amp)
plot.show()
```



test35.py

C: > Users > acer > OneDrive > Desktop > python > test35.py > ...

```python
#cosht
import numpy as np
import matplotlib.pyplot as plot
t = np.arange(0, 10, 0.01)
amp = np.cosh(t)
plot.plot(t, amp)
plot.show()
```



EXPERIMENT 2

AIM: To compute the first and second derivatives of above functions using built in tools such as grad and plot the derivatives.

CODE:

P.T.0

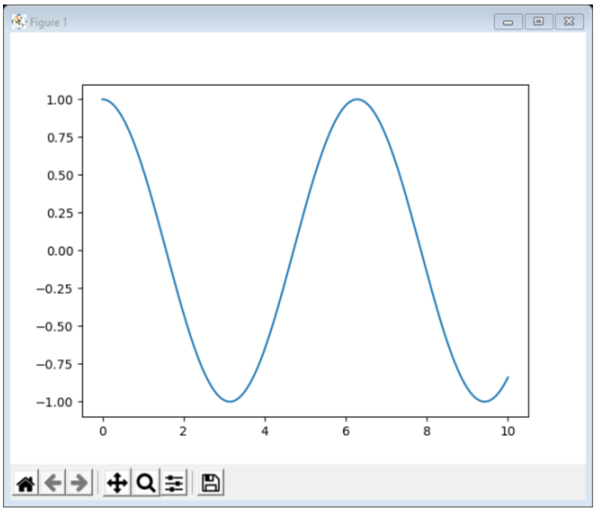
```
#sinx
import numpy as np                                    #import numpy
from scipy.interpolate import InterpolatedUnivariateSpline
import matplotlib.pyplot as plt                       #import matplotlib.pyplot
x= np.arange(0, 10.01,0.01)                           #defining range and increment of x
si = np.sin(x)                                        #function
co=np.cos(x)
sih=np.sinh(x)
coh=np.cosh(x)
f1 = InterpolatedUnivariateSpline(x, si)              #first derivative
dfdx = f1.derivative()
plt.plot(x, dfdx(x))
plt.show()                                            #plot function
dfdx2=dfdx.derivative()                               #second derivative
plt.plot(x, dfdx2(x))
plt.show()                                            #plot function

#sinhx
import numpy as np
from scipy.interpolate import InterpolatedUnivariateSpline
import matplotlib.pyplot as plt
x= np.arange(0, 10.01,0.01)
sih=np.sinh(x)
f1 = InterpolatedUnivariateSpline(x, sih)
dfdx = f1.derivative()
plt.plot(x, dfdx(x))
plt.show()
dfdx2=dfdx.derivative()
plt.plot(x, dfdx2(x))
plt.show()

#cosx
import numpy as np
from scipy.interpolate import InterpolatedUnivariateSpline
import matplotlib.pyplot as plt
x= np.arange(0, 10.01,0.01)
co=np.cos(x)
f1 = InterpolatedUnivariateSpline(x, co)
dfdx = f1.derivative()
plt.plot(x, dfdx(x))
plt.show()
dfdx2=dfdx.derivative()
plt.plot(x, dfdx2(x))
plt.show()

#coshx
import numpy as np
from scipy.interpolate import InterpolatedUnivariateSpline
import matplotlib.pyplot as plt
x= np.arange(0, 10.01,0.01)
coh=np.cosh(x)
f1 = InterpolatedUnivariateSpline(x, si)
dfdx = f1.derivative()
plt.plot(x, dfdx(x))
```

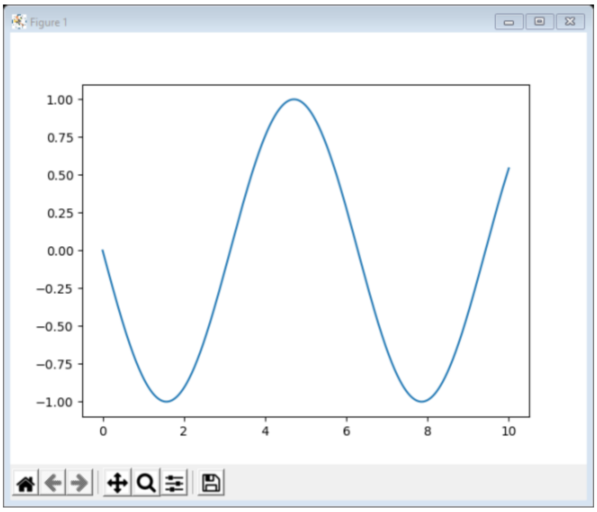
```
plt.show()
dfdx2=dfdx.derivative()
plt.plot(x, dfdx2(x))
plt.show()
```

RESULT:

1. sin(x)

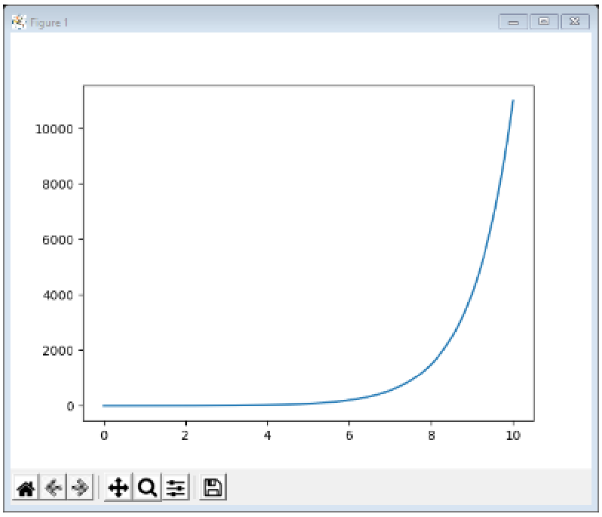
```
test40.py ×
C: > Users > acer > OneDrive > Desktop > python > test40.py > ...
import numpy as np                                          #import numpy
from scipy.interpolate import InterpolatedUnivariateSpline  #import InterpolatedUnivariateSpline
import matplotlib.pyplot as plt                             #import matplotlib.pyplot
x= np.arange(0, 10.01,0.01)                                 #defining range and increment of x
si = np.sin(x)                                              #function
f1 = InterpolatedUnivariateSpline(x, si)
dfdx = f1.derivative()                                      #first derivative
plt.plot(x, dfdx(x))
plt.show()                                                  #plot function
dfdx2=dfdx.derivative()                                     #second derivative
plt.plot(x, dfdx2(x))
plt.show()                                                  #plot function
```

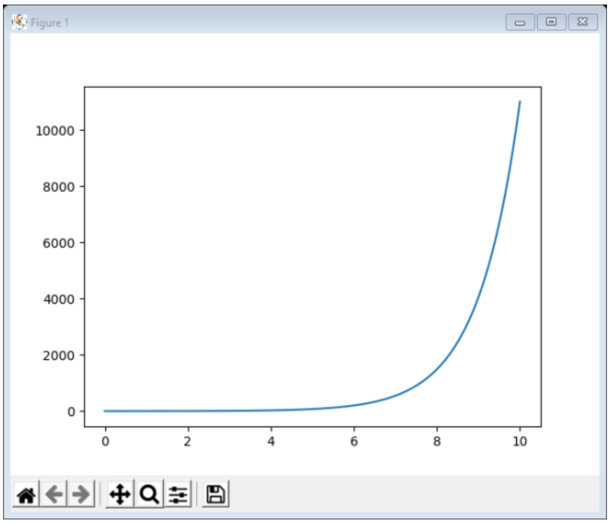




2.sinh(x)

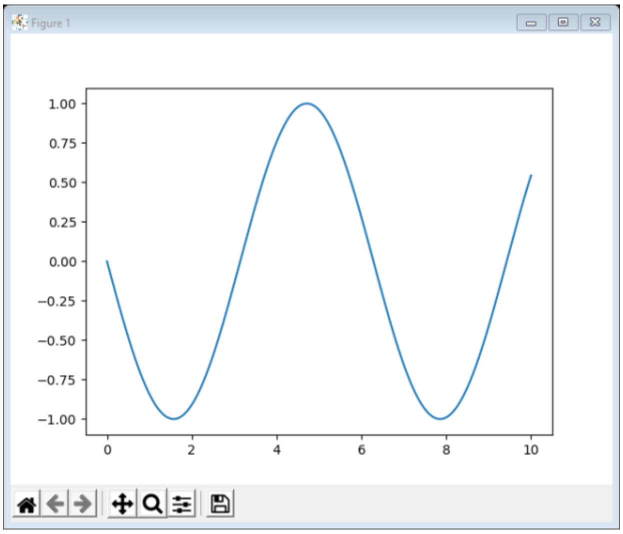
```
test40.py ×
C: > Users > acer > OneDrive > Desktop > python > test40.py > ...
import numpy as np                                          #import numpy
from scipy.interpolate import InterpolatedUnivariateSpline  #import InterpolatedUnivariateSpline
import matplotlib.pyplot as plt                             #import matplotlib.pyplot
x= np.arange(0, 10.01,0.01)                                 #defining range and increment of x
sih=np.sinh(x)                                              #function
f1 = InterpolatedUnivariateSpline(x, sih)
dfdx = f1.derivative()                                      #first derivative
plt.plot(x, dfdx(x))
plt.show()                                                  #plot function
dfdx2=dfdx.derivative()                                     #second derivative
plt.plot(x, dfdx2(x))
plt.show()                                                  #plot function
```

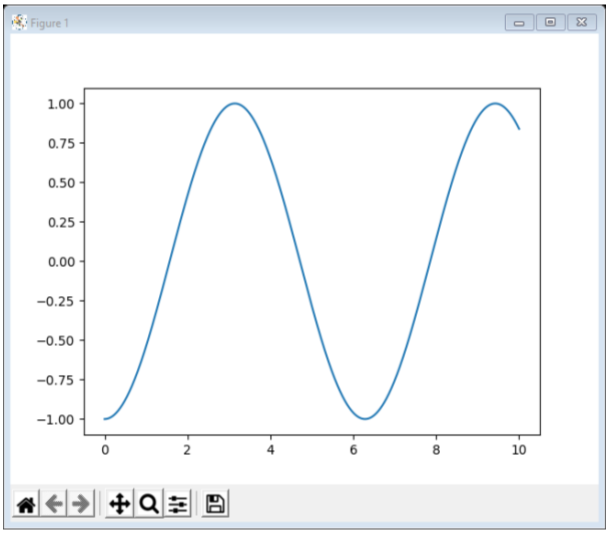




3.cos(x)

```
test40.py ●
C: > Users > acer > OneDrive > Desktop > python > test40.py > ...
import numpy as np                                                 #import numpy
from scipy.interpolate import InterpolatedUnivariateSpline         #import InterpolatedUnivariateSpline
import matplotlib.pyplot as plt                                    #import matplotlib.pyplot
x= np.arange(0, 10.01,0.01)                                        #defining range and increment of x
co=np.cos(x)                                                       #function
f1 = InterpolatedUnivariateSpline(x, co)
dfdx = f1.derivative()                                             #first derivative
plt.plot(x, dfdx(x))
plt.show()                                                         #plot function
dfdx2=dfdx.derivative()                                            #second derivative
plt.plot(x, dfdx2(x))
plt.show()                                                         #plot function
```



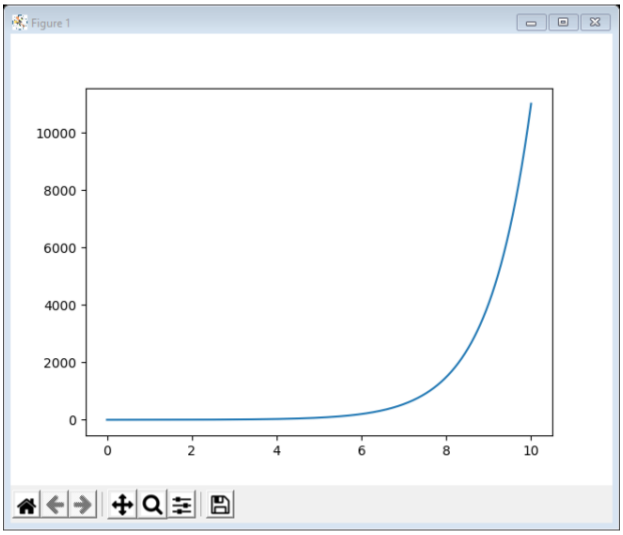


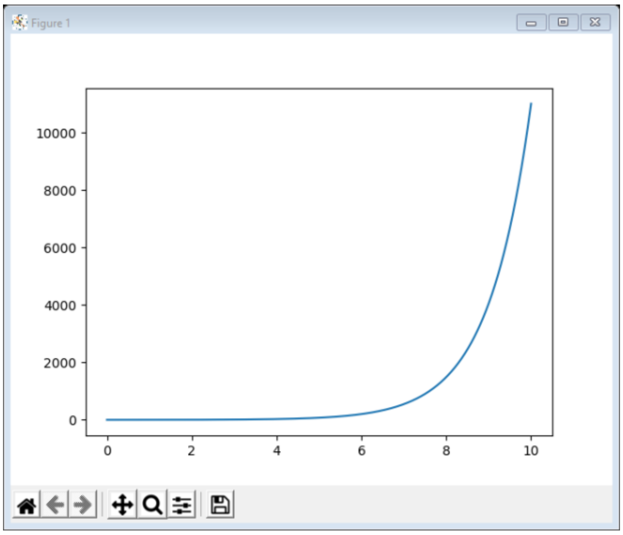
4.cosh(x)

test40.py

C: > Users > acer > OneDrive > Desktop > python > test40.py > ...

```python
import numpy as np                                          #import numpy
from scipy.interpolate import InterpolatedUnivariateSpline  #import InterpolatedUnivariateSpline
import matplotlib.pyplot as plt                             #import matplotlib.pyplot
x= np.arange(0, 10.01,0.01)                                 #defining range and increment of x
coh=np.cosh(x)                                              #function
f1 = InterpolatedUnivariateSpline(x, coh)
dfdx = f1.derivative()                                      #first derivative
plt.plot(x, dfdx(x))
plt.show()                                                  #plot function
dfdx2=dfdx.derivative()                                     #second derivative
plt.plot(x, dfdx2(x))
plt.show()                                                  #plot function
```





P.T.O.

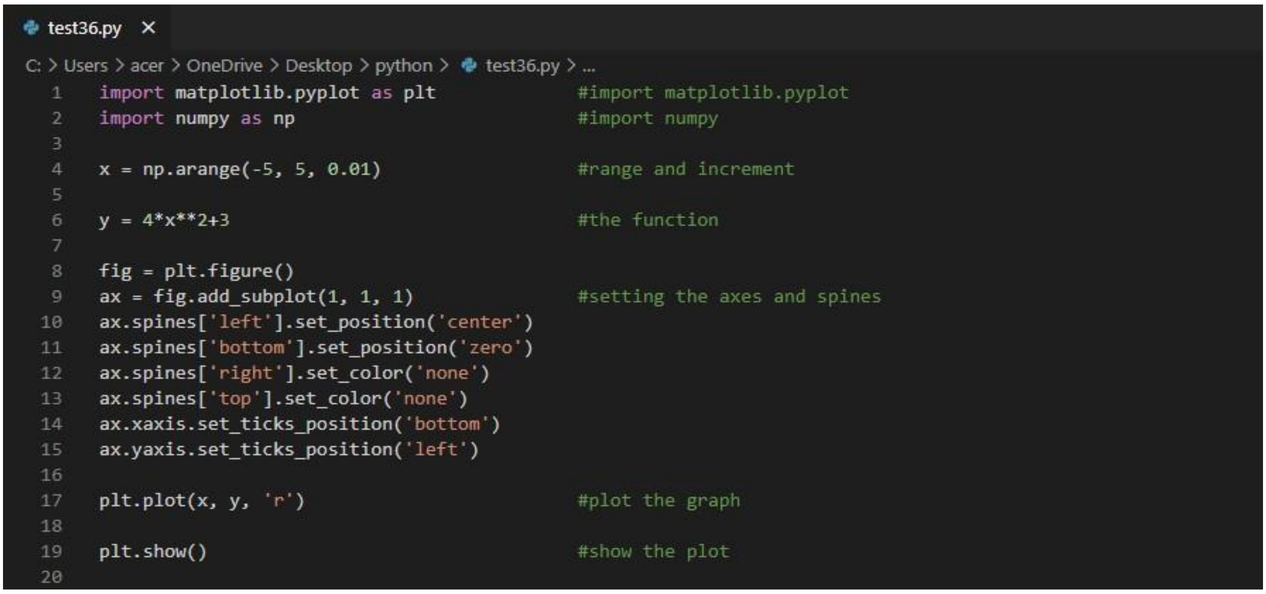
EXPERIMENT 3

AIM: To realize the function f(t) = 4t^2+ 3 and plot it for the vector t = [-5 5] with increment 0.01

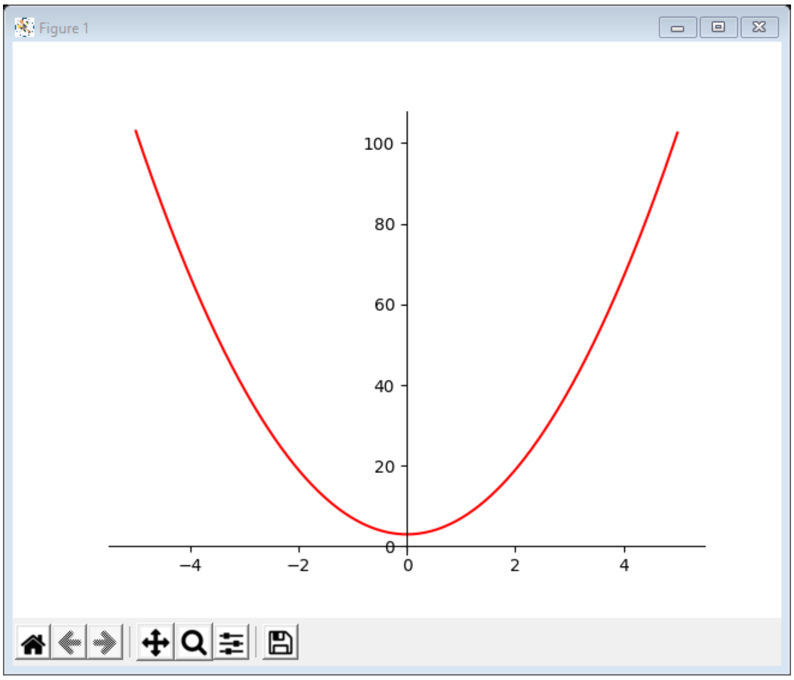
CODE:

```
import matplotlib.pyplot as plt                    #import matplotlib.pyplot
import numpy as np                                 #import numpy

x = np.arange(-5, 5, 0.01)                         #range and increment

y = 4*x**2+3                                       #the function

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)                      #setting the axes and spines
ax.spines['left'].set_position('center')
ax.spines['bottom'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set ticks position('left')

plt.plot(x, y, 'r')                                #plot the graph

plt.show()                                         #show the plot
```

RESULT:




```
import matplotlib.pyplot as plt                    #import matplotlib.pyplot
import numpy as np                                 #import numpy

x = np.arange(-5, 5, 0.01)                         #range and increment

y = 4*x**2+3                                       #the function

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)                      #setting the axes and spines
ax.spines['left'].set_position('center')
ax.spines['bottom'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

plt.plot(x, y, 'r')                                #plot the graph

plt.show()                                         #show the plot
```



EXPERIMENT 4

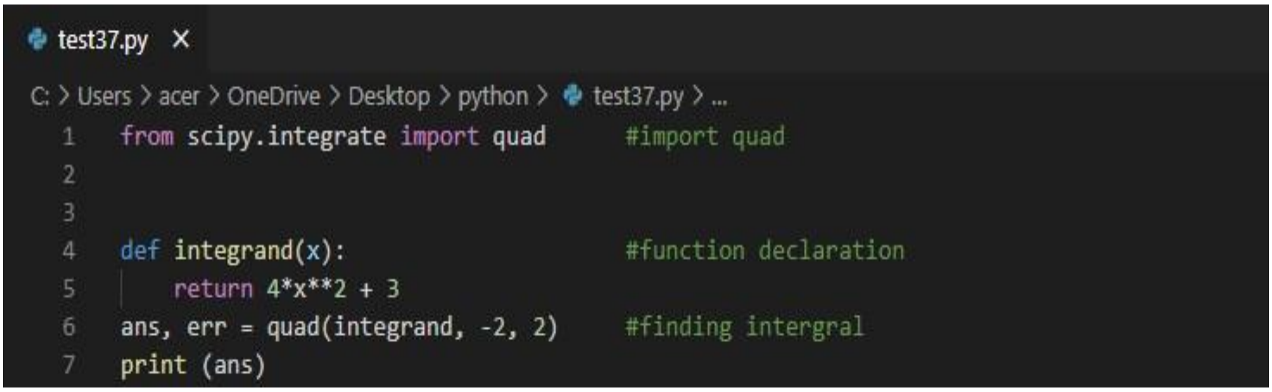
AIM: To compute the integral of f(t) in the limit -2 to 2 where f(t) = 4t^2+ 3

CODE:

```
from scipy.integrate import quad       #import quad


def integrand(x):                      #function declaration
    return 4*x**2 + 3

ans, err = quad(integrand, -2, 2)      #finding intergral
print (ans)
```

RESULT:

Integral of the given function = 33.33



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\acer> & E:/python.exe c:/Users/acer/OneDrive/Desktop/python/test37.py
33.333333333333336
```

ECL 201 SCIENTIFIC COMPUTING LABORATORY

PRELIMINARY LAB REPORT

Lab 4: Numerical Differentiation and Integration

1. Write a short note on different hyperbolic functions such as sinh, cosh, tanh

In python programming language, there are some of the built-in functions which are defined in math module – they can be used for hyperbolic calculations (these functions are analogs of trigonometric functions those are based on hyperbolas instead of circles.), python has following Hyperbolic functions, which are used to various purposes.

- **math.acosh(x) method** is a library method of **math** module, it is used to get the hyperbolic arc cosine of the given number in radians, it accepts a number and returns hyperbolic arc cosine.
  Parameter(s)**:** x – is the number whose hyperbolic arc cosine to be calculated.
  Return value**:** float – it returns a float value that is the hyperbolic arc cosine value of the number x
- **math.asinh(x) method** is a library method of **math** module, it is used to get the hyperbolic arc sine of given number in radians, it accepts a number and returns hyperbolic arc sine.
  Parameter(s): x – is the number whose hyperbolic arc sine to be calculated.
  Return value: float – it returns a float value that is the hyperbolic arc sine value of the number x.
- **math.atanh(x) method** is a library method of **math** module, it is used to get the hyperbolic arc tangent of given number in radians, it accepts a number and returns hyperbolic arc tangent.
  Parameter(s): x – is the number whose hyperbolic arc tangent to be calculated.

Return value: float – it returns a float value that is the hyperbolic arc tangent value of the number x.

- **math.cosh(x) method** is a library method of **math** module, it is used to get the hyperbolic cosine of given number in radians, it accepts a number and returns hyperbolic cosine.
  Parameter(s): x – is the number whose hyperbolic cosine to be calculated.
  Return value: float – it returns a float value that is the hyperbolic cosine value of the number x
- **math.sinh(x) method** is a library method of **math** module, it is used to get the hyperbolic sine of given number in radians, it accepts a number and returns hyperbolic sine.
  Parameter(s): x – is the number whose hyperbolic sine to be calculated.
  Return value: float – it returns a float value that is the hyperbolic sine value of the number x
- **math.tanh() method** is a library method of **math** module, it is used to get the hyperbolic tangent of given number in radians, it accepts a number and returns hyperbolic tangent.
  Parameter(s): x – is the number whose hyperbolic tangent to be calculated.
  Return value: float – it returns a float value that is the hyperbolic tangent value of the number x

2. Write the algorithm/ flowchart for the experiments

EXPERIMENT 1

AIM: To realize the functions sin t, cos t, sinht and cosht for the vector t = [0, 10] with increment 0.01.

ALGORITHM:

1. START
2. Import numpy

3. Import matplot.pyplot
4. Declare time period t within limit [0,10] with an increment of 0.01
   t = numpy.arange(0, 10, 0.01)
5. Declare amplitude for the required function with respect to time t[i.e. amp = numpy.sin(t)]
6. Declare plot function to plot t and amp
7. Declare show function to display the plot
8. END

EXPERIMENT 2

AIM: To compute the first and second derivatives of above functions using built in tools such as grad and plot the derivatives.

ALGORITHM:

1. START
2. Declare the function fibsum
3. Input the number(n)
4. Define the function fibsum
   - If n<=0, return 0
   - fib =[0] * (n+1)
   - fib[1] = 1
   - sum = fib[0] + fib[1]
   - for i in range(2,n+1) :
     - fib[i] = fib[i-1] + fib[i-2]
     - sum = sum + fib[i]
   - return sum
5. Call the function and print the sum
6. END

EXPERIMENT 3

AIM: To realize the function f(t) = 4t^2+ 3 and plot it for the vector t = [-5 5] with increment 0.01

ALGORITHM:

1. START
2. Import numpy
3. Import matplot.pyplot
4. Declare x within limit [-5,5] with an increment of 0.01
5. Declare y as the given function
6. Declare function for plot colour,scale and alignment
7. Declare plot function for plotting
8. Declare show function to display the plot
9. END

EXPERIMENT 4

AIM: To compute the integral of f(t) in the limit -2 to 2 where f(t) = 4t^2+ 3

ALGORITHM:

1. START
2. from scipy.integrate import quad
3. Define function integrand(t) whose return value is the given function f(t)
4. Applying quad function in the given limit
5. Obtained value saved in ans
6. Print ans
7. END