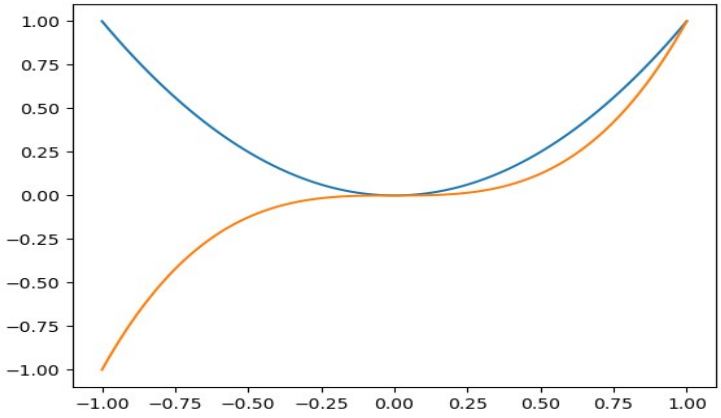
Slip no1

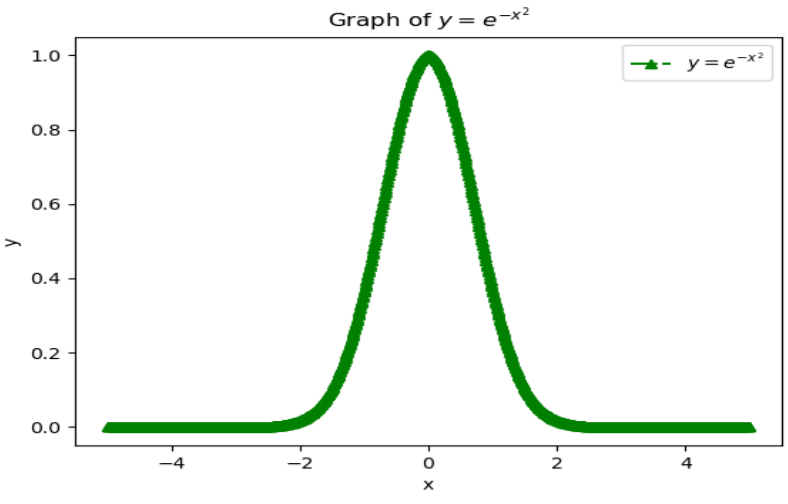
Q no 1
a)

```
>>> from pylab import *
>>> import numpy as np
>>> x=np.linspace(-1,1,100)
>>> f=x**2
>>> g=x**3
>>> plot(x,f)
[<matplotlib.lines.Line2D object at
0x7fa52aea49d0>]
>>> plot(x,g)
[<matplotlib.lines.Line2D object at
0x7fa52aea4c40>]
>>> show()
```



Qb)

```
>>> import numpy as np
>>> x=np.linspace(-5,5,1000)
>>> y=np.exp(-x**2)
>>> plot(x,y,"-.^g",label="$y=e^{-x^2}$")
[<matplotlib.lines.Line2D object at 0x7fbd422f89d0>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('Graph of $y=e^{-
x^2}$')
Text(0.5, 1.0, 'Graph of
$y=e^{-x^2}$')
>>> legend()
<matplotlib.legend.Legend
object at 0x7fbd596d0f70>
>>> show()
```



Q2)a)

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(5,3)
>>> B=Point(1,4)
>>> S=Segment(A,B)
>>> S.reflect(Line(x-y+1))
Segment2D(Point2D(2, 6), Point2D(3, 2))
>>>
```

Q2b)

```
>>> A=Point(0,0)
>>> B=Point(2,0)
>>> C=Point(2,3)
>>> D=Point(1,6)
>>> P=Polygon(A,B,C,D)
>>> P.rotate(pi)
Polygon(Point2D(0, 0), Point2D(-2, 0), Point2D(-2, -3), Point2D(-1, -6))
>>>
```

Q2)C)

```
>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(5,0)
>>> C=Point(3,3)
>>> T=Triangle(A,B,C)
>>> T.area
15/2
>>> T.perimeter
sqrt(13) + 3*sqrt(2) + 5
>>>
```

Q3)

a)i)

```
>>> from pulp import*
>>> model=LpProblem(name="Small-problem",sense=LpMaximize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model += (4*x+6*y <= 24)
>>> model += (5*x+3*y <= 15)
>>> model += 150*x+75*y
>>> model
Small-problem:
MAXIMIZE
150*x + 75*y + 0
SUBJECT TO
_C1: 4 x + 6 y <= 24

_C2: 5 x + 3 y <= 15
```

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/002063f8abc0457e9b0920157d83f3d5-pulp.mps max timeMode elapsed branch printingOptions all solution /tmp/002063f8abc0457e9b0920157d83f3d5-pulp.sol (default strategy 1)
At line 2 NAME MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
0 Obj -0 Dual inf 225 (2)
0 Obj -0 Dual inf 225 (2)
1 Obj 450
Optimal - objective value 450
Optimal objective 450 - 1 iterations time 0.042
Option for printingOptions changed from normal to all
Total time (CPU seconds): 0.12 (Wallclock seconds): 0.04

1
>>> x.value()
3.0
>>> y.value()
0.0

Q3)b)1

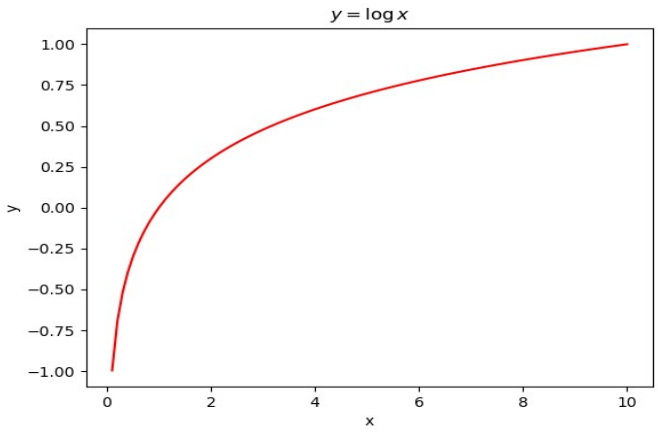
>>> from sympy import*
>>> P=Point(3,-1)
1)
>>> P.transform(Matrix([[1,0,0],[0,-1,0],[0,0,1]]))
Point2D(3, 1)
2)
>>> P.scale(2,0)
Point2D(6, 0)
3)
>>> P.scale(0,1.5)
Point2D(0, -3/2)
4)
>>> x,y=symbols('x,y')
>>> P.reflect(Line(x-y))

```
Point2D(-1, 3)
>>>
```

Slip no :- 2

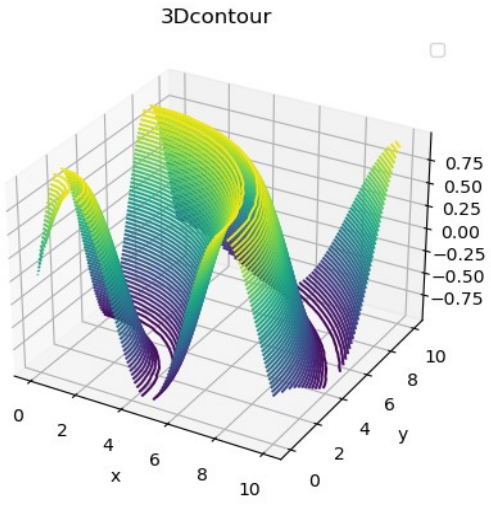
Q.1..a)

b)..>>> from pylab import *
>>> import numpy as np
>>> from math import *
>>> x=np.linspace(0,10,100)
>>> y=np.log10(x)
<stdin>:1: RuntimeWarning: divide by zero encountered in log10
>>> plot(x,y,'r')
[<matplotlib.lines.Line2D object at 0x7f91712be2c0>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('$y=\logx$')
Text(0.5, 1.0, '$y=\\logx$')
>>> show()
>>>



>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from pylab import*
>>> def f(x,y):
... return np.sin(np.sqrt(x**2+y**2))
...
>>> x=np.linspace(0,10,30)
>>> y=np.linspace(0,10,30)
>>> x,y=np.meshgrid(x,y)
>>> z=f(x,y)
>>> ax=axes(projection='3d')
>>> ax.contour3D(x,y,z,50)
<matplotlib.contour.QuadContourSet object at 0x7fe53b0ecca0>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('3Dcontour')
Text(0.5, 0.92, '3Dcontour')

>>> legend()
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



<matplotlib.legend.Legend object at 0x7fe53b0ecca0>
>>> show()
>>>

Q2. b)

>>> from sympy import*
>>> P=Polygon((1,2),1,n=6)
>>> P.area
3*sqrt(3)/2
>>> P.perimeter
6

c)..
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(6,0)
>>> c=Point(4,4)
>>> T=Triangle(A,B,c)
>>> T.area
12
>>> T.perimeter
2*sqrt(5) + 4*sqrt(2) + 6
>>>

Q3)..a)..1)

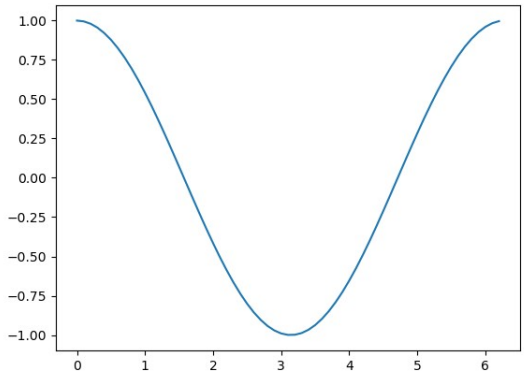
```
>>> from pulp import*

>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model+=(x+y<=7)
>>> model+=(2*x+5*y<=1)
>>> model+=x+y
>>> model
NoName:
MAXIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x + y <= 7

_C2: 2 x + 5 y <= 1

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/c6b2c407f807453fb8d3bfebc25e9195-pulp.mps max timeMode elapsed
branch printingOptions all solution /tmp/c6b2c407f807453fb8d3bfebc25e9195-pulp.sol (default
strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
0  Obj -0 Dual inf 1.9999998 (2)
0  Obj -0 Dual inf 1.9999998 (2)
2  Obj 0.5
Optimal - objective value 0.5
Optimal objective 0.5 - 2 iterations time 0.002
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.08   (Wallclock seconds):       0.03

1

>>> model.objective.value()
0.5
>>> x.value()
0.5
```

```
>>> y.value()
0.0
```

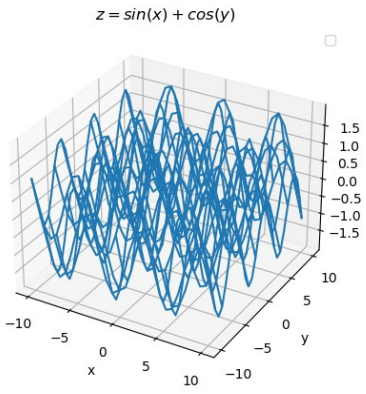
b)..

```
>>> from sympy import *
>>> P=Point(4,-2)
>>> P.transform(Matrix([[-1,1,0],[0,1,0],[0,0,1]]))
Point2D(-4, 2)
>>> P.scale(3,0)
Point2D(12, 0)
>>> P.scale(0,2.5)
Point2D(0, -5)
>>> x,y=symbols('x y')
>>> P.reflect(Line(x+y+0))
Point2D(2, -4)
```

Slip no.3

`````kkw@kkw-HP-ProDesk-400-G4-SFF:~$ **python3**
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.

Q1. a]
>>> **import matplotlib.pyplot as plt**
>>> import numpy as np
>>> >>> from pylab import*
>>> x=np.linspace(0*pi,2*pi,100)
>>> f=np.cos(x)
>>> plot(x,f)
[<matplotlib.lines.Line2D object at 0x77e3c39691b0>]
>>> show()




Q1. c]

>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from pylab import*
>>> def f(x,y):
...  return np.sin(x)+np.sin(y)
...
>>> x = np.linspace(-10,10,30)
>>> y = np.linspace(-10,10,30)
>>> X,Y=np.meshgrid(x,y)
>>> Z = f(X, Y)
>>> ax = axes(projection='3d')
>>> ax.plot_wireframe(X,Y,Z,rstride=2,cstride=2)
<mpl_toolkits.mplot3d.art3d.Line3DCollection object at 0x71896cbcfeb0>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('$z=sin(x)+cos(y)$')
`````

Text(0.5, 0.92, '$z=sin(x)+cos(y)$')
>>> legend()
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x718984b66500>
>>> show()



Q2_b

Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from sympy import*
A=>>> A=Point(2,1)
>>> B=Point(4,-1)
>>> A1=A.transform(Matrix([[1,2,0],[2,1,0],[0,0,1]]))
>>> B1=B.transform(Matrix([[1,2,0],[2,1,0],[0,0,1]]))
>>> L=Line(A1,B1)
>>> L.equation()
-2*x - 2*y + 18

Q2_c

>>> S=Segment(Point(0,0),Point(10,10))
>>> S.midpoint
Point2D(5, 5)
>>>

Q3_A_i

>>> from pulp import*
>>> model = LpProblem(sense=LpMinimize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model +=(x+y>=5)
>>> model +=(x>=4)
>>> model +=(y<=2)

```
>>> model +=3.5*x+2*y
>>> model
NoName:
MINIMIZE
3.5*x + 2*y + 0.0
SUBJECT TO
_C1: x + y >= 5

_C2: x >= 4

_C3: y <= 2

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/834ec6cb8ee4482b83ad829370f19a0c-pulp.mps timeMode elapsed
branch printingOptions all solution /tmp/834ec6cb8ee4482b83ad829370f19a0c-pulp.sol
(default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 15 RHS
At line 19 BOUNDS
At line 20 ENDATA
Problem MODEL has 3 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 1 (-2) rows, 2 (0) columns and 2 (-2) elements
0  Obj 14 Primal inf 0.999999 (1)
1  Obj 16
Optimal - objective value 16
After Postsolve, objective 16, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 16 - 1 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.03

1
>>> model.objective.value()
16.0
>>> x.value
<bound method LpVariable.value of x>
>>> y.value
<bound method LpVariable.value of y>
>>>
```

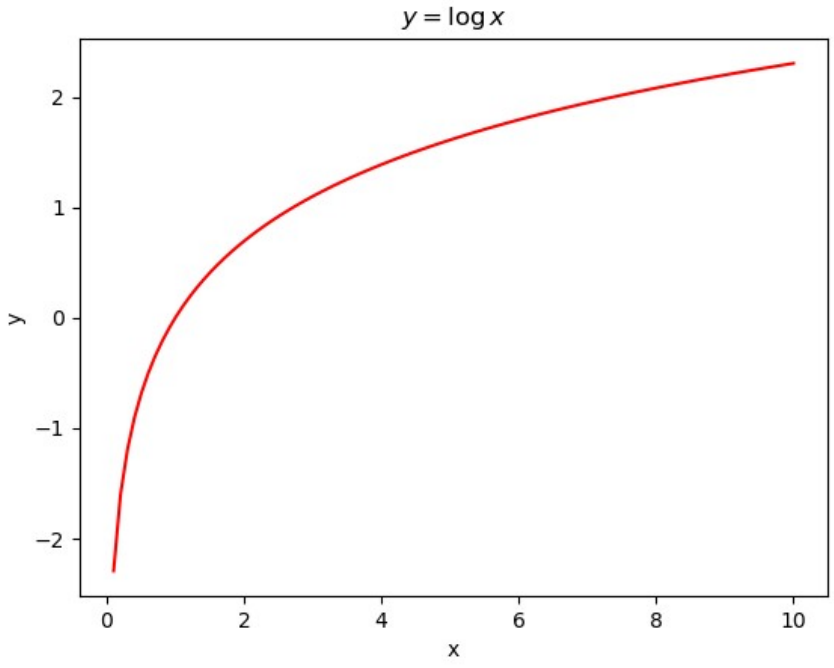
Q3 b .i]

```
>>> from sympy import*
>>> P=Point(4,-2)
>>> P.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
Point2D(-4, -2)
>>> P.scale(3,0)
Point2D(12, 0)
>>> P.scale(0,2.5)
Point2D(0, -5)
>>> x,y=symbols('x,y')
>>> P.reflect(Line(x+y+0))
Point2D(2, -4)
```

Slip-4

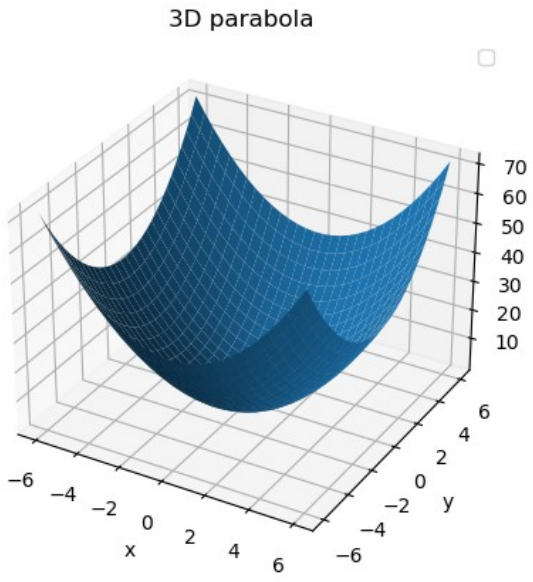
(a)

```
>>> from pylab import*
>>> import numpy as np
>>> from math import*
>>> x = np.linspace(0, 10, 100)
>>> y = np.log(x)
<stdin>:1: RuntimeWarning: divide by zero encountered in log
>>> plot(x, y, 'r')
[<matplotlib.lines.Line2D object at 0x7796d2e81090>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('$y=\logx$')
Text(0.5, 1.0, '$y=\\logx$')
>>> show()
```



(c)

```
kkw@kkw-HP-ProDesk-400-G4-SFF:~$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from pylab import*
>>> import numpy as np
>>> from math import*
>>> x = np.linspace(0, 10, 100)
>>> y = np.log(x)
<stdin>:1: RuntimeWarning: divide by zero encountered in log
>>> plot(x, y, 'r')
[<matplotlib.lines.Line2D object
at 0x7796d2e81090>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('$y=\logx$')
Text(0.5, 1.0, '$y=\\logx$')
>>> show()
```



Q.2-(b)

```
 >>> from sympy import*
>>> A = Point(0, 0)
>>> B = Point(2, 0)
>>> C = Point(2, 3)
>>> D = Point(1, 6)
>>> P =Polygon(A, B, C, D)
>>> P.rotate(pi)
Polygon(Point2D(0, 0), Point2D(-2, 0), Point2D(-2, -3), Point2D(-1, -6))
```

(c)

```
>>> from sympy import*
>>> L = Line(Point(2, 3), Point(4, 3))
>>> L.equation
<bound method Line2D.equation of Line2D(Point2D(2, 3), Point2D(4, 3))>
>>> L.coefficients
(0, 1, -3)
```

Q-3(a)-1

```
>>> from pulp import *
>>> model = LpProblem(name="small-problem", sense=LpMaximize)
>>> x = LpVariable(name="x", lowBound=0)
>>> y = LpVariable(name="y", lowBound=0)
>>> model += (4 * x + 6 * y <= 24)
>>> model += (5 * x + 3 * y <= 15)
>>> model += 150 * x + 75 * y
>>> model
small-problem:
MAXIMIZE
150*x + 75*y + 0
SUBJECT TO
_C1: 4 x + 6 y <= 24

_C2: 5 x + 3 y <= 15

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/afb4c6af376343b2944880e467300c0b-pulp.mps max
timeMode elapsed branch printingOptions all solution
/tmp/afb4c6af376343b2944880e467300c0b-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 7 COLUMNS
```

```
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
0  Obj -0 Dual inf 225 (2)
0  Obj -0 Dual inf 225 (2)
1  Obj 450
Optimal - objective value 450
Optimal objective 450 - 1 iterations time 0.002
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.08

1
>>> model.objective.value()
450.0
>>> x.value()
3.0
>>> y.value()
0.0
```

(b)-2

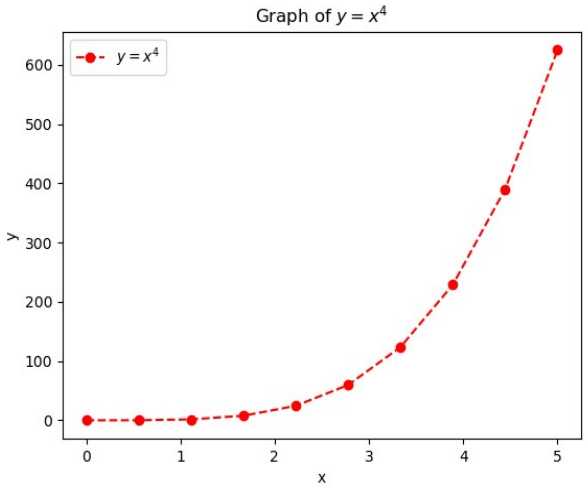
```
>>> from sympy import*
>>> A=Point(4,-1)
>>> B=Point(3,0)
>>> S=Segment(A,B)
>>> S=S.rotate(pi)
>>> S=S.scale(3,0)
>>> points=S.points
>>> p=points[0]
>>> q=points[1]
>>> p1=p.transform(Matrix([[0,1,0],[1,0,0],[0,0,1]]))
>>> q1=q.transform(Matrix([[0,1,0],[1,0,0],[0,0,1]]))
>>> Segment(p1,q1)
Segment2D(Point2D(0, -12), Point2D(0, -9))
>>> A.transform(Matrix([[1, 9, 0], [0, 1, 0],[0, 0, 1]]))
```

Point2D(4, 35)

slip no 7

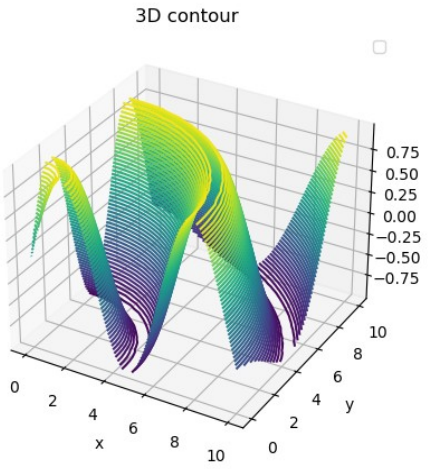
Q1)a)

```
>>> from pylab import*
>>> import numpy as np
>>> x=np.linspace(0,5,10)
>>> y=x**4
>>> plot(x,y,"--or",label="$y=x^4$")
[<matplotlib.lines.Line2D object at 0x7157cd387d30>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y’)
>>> title('Graph of $y=x^4$')
Text(0.5, 1.0, 'Graph of $y=x^4$')
>>> legend()
<matplotlib.legend.Legend object at 0x7157ce3d9ea0>
>>> show()
>>>
```



b)

```
>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from pylab import*
>>> def f(x,y):
...   return np.sin(np.sqrt(x**2+y**2))
...
>>> x=np.linspace(0,10,30)
>>> y=np.linspace(0,10,30)
>>> X,Y=np.meshgrid(x,y)
>>> Z = f (X,Y)
>>> ax=axes(projection='3d')
>>> ax.contour3D(X,Y,Z,50)
<matplotlib.contour.QuadContourSet object at 0x79e8b9ad0f70>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('3D contour')
Text(0.5, 0.92, '3D contour')
>>> legend()
No artists with labels found to put in legend.
Note that artists whose label start with an
underscore are ignored when legend() is called
with no argument.
<matplotlib.legend.Legend object at
0x79e89f9d4100>
>>> show()
>>>
```



Q2) a)

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(5,3)
>>> B=Point(1,4)
>>> S=Segment(A,B)
>>> S.reflect(Line(x-y+1))
Segment2D(Point2D(2, 6), Point2D(3, 2))
>>>
>>>
```

b)

```
>>> from sympy import*
>>> P=Point(5,2)
>>> Q=Point(5,-2)
>>> R=Point(5,0)
>>> Point.is_collinear(P,Q,R)
True
>>>
```

Q3)a)i)

```
>>> from pulp import*
>>> model=LpProblem(sense=LpMinimize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model += (x+y>=5)
>>> model += (x>=4)
>>> model += (y<=2)
>>> model += 3.5*x+2*y
>>> model
NoName:
MINIMIZE
3.5*x + 2*y + 0.0
SUBJECT TO
_C1: x + y >= 5

_C2: x >= 4

_C3: y <= 2

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/f792b507e8d340be9dfbc2b5dc864df8-pulp.mps timeMode elapsed branch printingOptions all solution /tmp/f792b507e8d340be9dfbc2b5dc864df8-pulp.sol (default strategy 1)
```

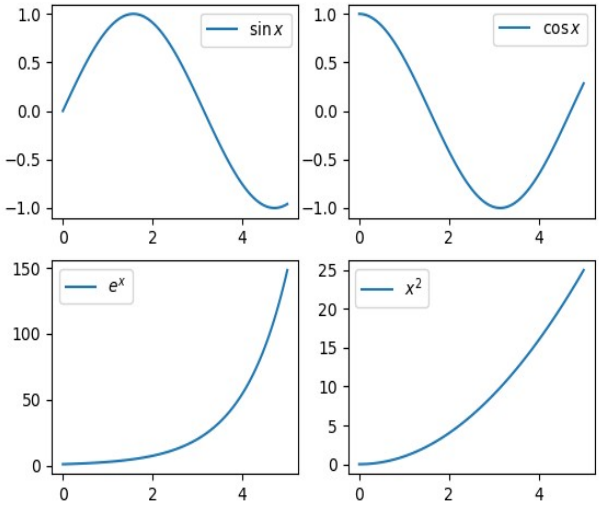
```
At line 2 NAME          MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 15 RHS
At line 19 BOUNDS
At line 20 ENDATA
Problem MODEL has 3 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 1 (-2) rows, 2 (0) columns and 2 (-2) elements
0  Obj 14 Primal inf 0.999999 (1)
1  Obj 16
Optimal - objective value 16
After Postsolve, objective 16, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 16 - 1 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.03   (Wallclock seconds):       0.02

1
>>> model.objective.value()
16.0
>>> x.value()
4.0
>>> y.value()
1.0
>>>

Q3)b)1)
>>> from sympy import*
>>> P=Point(4,-2)
1)
>>> P.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
Point2D(-4, -2)
2)
>>> P.scale(5,0)
Point2D(20, 0)
3)
>>> P.rotate(5/2)
Point2D(-100381508698991/50000000000000, 399617580750969/100000000000000)
4)
>>> P.transform(Matrix([[1,7/2,0],[0,1,0],[0,0,1]]))
Point2D(4, 12)
>>>
```

slip no 8

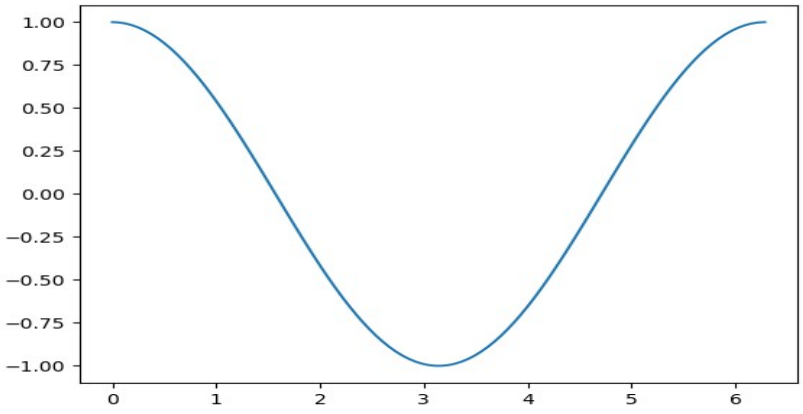
Q1
a)

```
>>> from pylab import *
>>> import numpy as np
>>> from math import*
>>> x=np.linspace(0,5,100)
>>> y1=np.sin(x)
>>> y2=np.cos(x)
>>> y3=np.exp(x)
>>> y4=x**2
>>> subplot(2,2,1)
<AxesSubplot:>
>>> plot(x,y1,label="$\sinx$")
[<matplotlib.lines.Line2D object at 0x7fbd422b52d0>]
>>> legend()
<matplotlib.legend.Legend object at 0x7fbd422f9300>
>>> subplot(2,2,2)
<AxesSubplot:>
>>> plot(x,y2,label="$\cosx$")
[<matplotlib.lines.Line2D object at 0x7fbd3d3d0d60>]
>>> legend()
<matplotlib.legend.Legend object at 0x7fbd422b6920>
>>> subplot(2,2,3)
<AxesSubplot:>
>>> plot(x,y3,label="$e^x$")
[<matplotlib.lines.Line2D object at 0x7fbd3d3ff220>]
>>> legend()
<matplotlib.legend.Legend object at 0x7fbd3d3d1de0>
>>> subplot(2,2,4)
<AxesSubplot:>
>>> plot(x,y4,label="$x^2$")
[<matplotlib.lines.Line2D object at 0x7fbd3d44d690>]
>>> legend()
<matplotlib.legend.Legend object at 0x7fbd3d3fffd0>
>>> show()
```

>>>



b)>>> from pylab import*
>>> import numpy as np
>>> x= np.linspace(0*pi,2*pi,100)
>>> f=np.cos(x)
>>> plot(x,f)
[<matplotlib.lines.Line2D object at
0x72a2aa537e50>]
>>> show()
>>>



q2)any two
b)>>> from sympy import *
>>>
t1=Triangle(Point(0,0),Point(4,0),Point(1,4))
>>> t1.is_scalene()
True
>>>

c)>>> from sympy import *
>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(6,0)
>>> C=Point(4,4)
>>> T=Triangle(A,B,C)
>>> T.area
12
>>> T.perimeter
2*sqrt(5) + 4*sqrt(2) + 6
>>>

Q3)
1)>>> from pulp import*
>>> model=LpProblem(name="Small-problem",sense=LpMaximize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model += (4*x+6*y <= 24)
>>> model += (5*x+3*y <= 15)
>>> model += 150*x+75*y
>>> model
Small-problem:
MAXIMIZE
150*x + 75*y + 0
SUBJECT TO
_C1: 4 x + 6 y <= 24

_C2: 5 x + 3 y <= 15

VARIABLES
x Continuous
y Continuous
>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/002063f8abc0457e9b0920157d83f3d5-pulp.mps max timeMode elapsed branch printingOptions all solution /tmp/002063f8abc0457e9b0920157d83f3d5-pulp.sol (default strategy 1)
At line 2 NAME MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
0 Obj -0 Dual inf 225 (2)
0 Obj -0 Dual inf 225 (2)
1 Obj 450
Optimal - objective value 450
Optimal objective 450 - 1 iterations time 0.042
Option for printingOptions changed from normal to all
Total time (CPU seconds): 0.12 (Wallclock seconds): 0.04

1
>>> x.value()
3.0
>>> y.value()
0.0

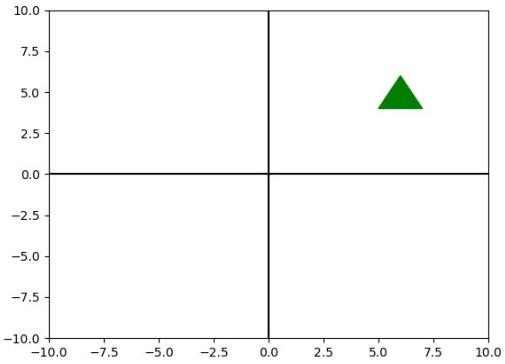
q3)b)

```
>>> from sympy import*
>>> P=Point(4,-2)
>>> P.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
Point2D(-4, -2)
>>> P.scale(3,0)
Point2D(12, 0)
>>> P.rotate(pi/3.14)
Point2D(2*sin(50*pi/157) + 4*cos(50*pi/157), -2*cos(50*pi/157) + 4*sin(50*pi/157))
>>> P.rotate(pi)
Point2D(-4, 2)
>>> P.transform(Matrix([[1,-2,0],[4,1,0],[0,0,1]]))
Point2D(-4, -10)
>>> P.transform(Matrix([[1,-2,0],[,0],[0,0,1]]))
```

Slip No 9

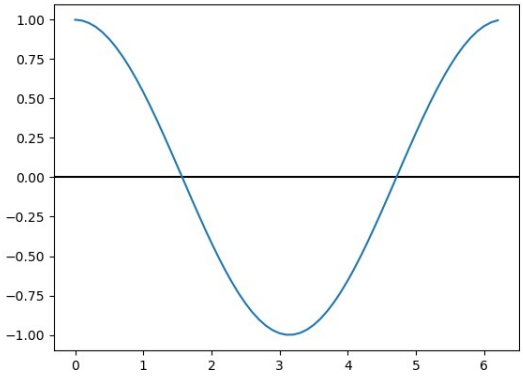
Q1)a)

```
>>> import matplotlib.pyplot as plt
>>> plt.axhline(y=0, color='black')
<matplotlib.lines.Line2D object at 0x7fbc2fd7f010>
>>> plt.axvline(x=0, color='black')
<matplotlib.lines.Line2D object at 0x7fbc45b7ab90>
>>> x = [5, 7, 6]
>>> y = [4, 4, 6]
>>> plt.fill(x, y, color='green')
[<matplotlib.patches.Polygon object at
0x7fbc2fd7f6a0>]
>>> plt.xlim(-10, 10)
(-10.0, 10.0)
>>> plt.ylim(-10, 10)
(-10.0, 10.0)
>>> plt.show()
```



c)using python plot the graph of function f(x)=cos(x) on the interval[0,2pi].

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> x=np.arange(0,2*(np.pi),0.1)
>>> y=np.cos(x)
>>> plt.plot(x,y)
[<matplotlib.lines.Line2D object at 0x736740656110>]
>>> plt.show()
```



Q_2_b

```
>>> import sympy
>>> A = sympy.Point(0, 7)
>>> B = sympy.Point(5, 2)
>>> AB = sympy.Segment(A, B)
>>> AB_length = AB.length
>>> AB_midpoint = AB.midpoint
>>> print("Ponit A:",A)
Ponit A: Point2D(0, 7)
```

```
>>> print("Ponit B:",B)
Ponit B: Point2D(5, 2)
>>> print("Line segment AB:", AB)
Line segment AB: Segment2D(Point2D(0, 7), Point2D(5, 2))
>>> print("Length of line segment AB:", AB_length)
Length of line segment AB: 5*sqrt(2)
>>> print("Midpoint of line segment AB:", AB_midpoint)
Midpoint of line segment AB: Point2D(5/2, 9/2)
```

Q_2c)

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(5,0)
>>> C=Point(3,3)
>>> T=Triangle(A,B,C)
>>> T . area
15/2
>>> T . perimeter
sqrt(13) + 3*sqrt(2) + 5
```

Q3)a)

2)

```
>>> from pulp import *
>>> model = LpProblem(sense= LpMaximize)
>>> x = LpVariable(name="x", lowBound=0)
>>> y = LpVariable(name="y", lowBound=0)
>>> z = LpVariable(name="z", lowBound=0)
>>> w = LpVariable(name="w", lowBound=0)
>>> model += (4*x+6*y-5*z-4*x >= -20)
>>> model += (-3*x-2*y+4*z+w <= 10)
>>> model += (-8*x-3*y+3*z+2*w <= 20)
>>> model += 4*x+y+3*z+5*w
>>> model
NoName:
MAXIMIZE
5*w + 4*x + 1*y + 3*z + 0
SUBJECT TO
_C1: 0 x + 6 y - 5 z >= -20

_C2: w - 3 x - 2 y + 4 z <= 10

_C3: 2 w - 8 x - 3 y + 3 z <= 20

VARIABLES
w Continuous
x Continuous
y Continuous
z Continuous
```

```
>>> model.solve()
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --cpxlp /tmp/bbc259170cfd4edba73649b8c6cab5d9-pulp.lp -o
/tmp/bbc259170cfd4edba73649b8c6cab5d9-pulp.sol
Reading problem data from '/tmp/bbc259170cfd4edba73649b8c6cab5d9-pulp.lp'...
3 rows, 4 columns, 10 non-zeros
8 lines were read
GLPK Simplex Optimizer 5.0
3 rows, 4 columns, 10 non-zeros
Preprocessing...
3 rows, 4 columns, 10 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  8.000e+00  ratio =  8.000e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 3
*     0: obj =  -0.000000000e+00 inf =   0.000e+00 (4)
*     2: obj =   5.000000000e+01 inf =   0.000e+00 (2)
LP HAS UNBOUNDED PRIMAL SOLUTION
glp_simplex: unable to recover undefined or non-optimal solution
If you need actual output for non-optimal solution, use --nopresol
Time used:   0.0 secs
Memory used: 0.0 Mb (39693 bytes)
Writing basic solution to '/tmp/bbc259170cfd4edba73649b8c6cab5d9-pulp.sol'...
-3
```

Q3 B)

1)

```
1) >>> P=Point(-2,4)
>>> x,y=symbols('x y')
>>> P.transform(Matrix([[1,0,0],[7,1,0],[0,0,1]]))
Point2D(26, 4)

2) >>> P.scale(7/2,7)
Point2D(-7, 28)

3) >>> P.transform(Matrix([[1,4,0],[7,1,0],[0,0,1]]))
Point2D(26, -4)

4) >>> from math import *
>>> angle=radians(60)
>>> P . rotate(angle)
Point2D(-17856406460551/4000000000000, 267949192431123/1000000000000000)
```

Slip no 10

```
kkw@kkw-HP-ProDesk-400-G4-SFF:~$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

Q 2. a]

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(1,1)
>>> B=Point(2,-2)
>>> C=Point(1,2)
>>> T=Triangle(A,B,C)
>>> T.rotate(pi/2)
Triangle(Point2D(-1, 1), Point2D(2, 2), Point2D(-2, 1))
```

b]

```
>>> A=Point(0,0)
>>> B=Point(2,0)
>>> C=Point(2,3)
>>> D=Point(1,6)
>>> P=Polygon(A,B,C,D)
>>> P.rotate(pi)
Polygon(Point2D(0, 0), Point2D(-2, 0), Point2D(-2, -3), Point2D(-1, -6))
```

c]

```
>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(5,0)
>>> C=Point(3,3)
>>> T=Triangle(A,B,C)
>>> T.area
15/2
>>> T.perimeter
sqrt(13) + 3*sqrt(2) + 5
```

Q 3.a .i]

```
>>> from pulp import*
>>> model=LpProblem(name="small-problem",sense=LpMaximize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model +=(x-y>=1)
>>> model +=(x+y>=2)
>>> model +=x+y
>>> model
small-problem:
MAXIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x - y >= 1

_C2: x + y >= 2

VARIABLES
```

```
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/35d9395e02084db3b7bcf55b0c98c52b-pulp.mps max timeMode elapsed
branch printingOptions all solution /tmp/35d9395e02084db3b7bcf55b0c98c52b-pulp.sol (default
strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve thinks problem is unbounded
Analysis indicates model infeasible or unbounded
0  Obj -0 Primal inf 2.9999998 (2) Dual inf 1.9999998 (2)
1  Obj 2e+10
1  Obj 1e+11
Dual infeasible - objective value 1e+11
DualInfeasible objective 1e+11 - 1 iterations time 0.002

Result - Linear relaxation unbounded

Enumerated nodes:           0
Total iterations:           0
Time (CPU seconds):         0.00
Time (Wallclock Seconds):   0.05

Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.05

-2
>>> x.value()
0.0
>>> y.value()
0.0


Q 3.b. i ]
>>> from sympy import*
>>> P=Point(-2,4)
>>> P.transform(Matrix([[1,0,0],[0,-1,0],[0,0,1]]))
Point2D(-2, -4)
>>> P.scale()
Point2D(-2, 4)
```

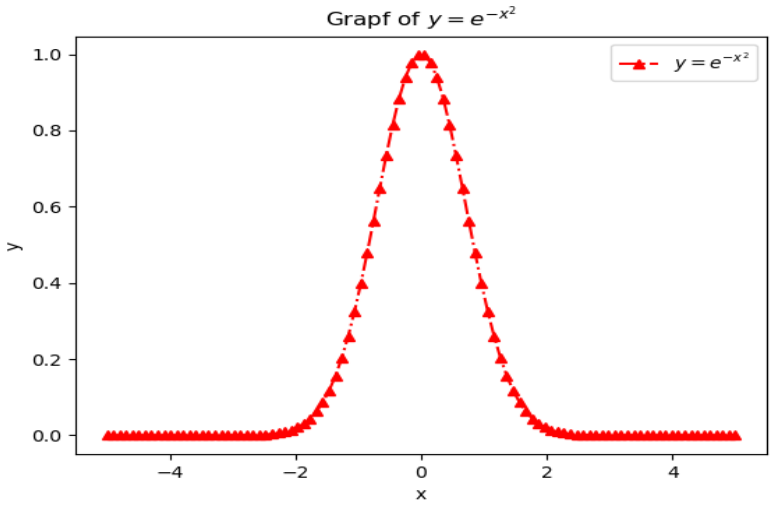
```
>>> P.transform(Matrix([[4,0,0],[0,1,0],[0,0,1]]))
Point2D(-8, 4)
>>>
>>> from math import*
>>> angle=radius(30)
```

slip-11

Q.1

(b)

```
>>> from pylab import*
>>> import numpy as np
>>> x = np.linspace(-5,5,100)
>>> y = np.exp(-x**2)
>>> plot(x, y,"-.^r",label="$y=e^{-x^2}$")
[<matplotlib.lines.Line2D object at 0x7fbf0a659120>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('Grapf of $y=e^{-x^2}$')
Text(0.5, 1.0, 'Grapf of $y=e^{-
x^2}$')
>>> legend()
<matplotlib.legend.Legend object at
0x7fbf0a61ae90>
>>> show()
```



Q.2-(a)

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=point(1,0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'point' is not defined. Did you mean: 'print'?
>>> A=Point(1,0)
>>> B=Point(2,-1)
>>> C=Point(-1,3)
>>> T=Triangle(A,B,C)
>>> P=Point(0,3)
>>> Q=Point(1,3)
>>> L=Line(P,Q)
>>> T.reflect(L)
Triangle(Point2D(1, 6), Point2D(2, 7), Point2D(-1, 3))
```

(b)

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(1,2)
>>> B=Point(2,-2)
>>> C=Point(-1,2)
>>> T=Triangle(A,B,C)
>>> T.rotate(pi/2)
Triangle(Point2D(-2, 1), Point2D(2, 2), Point2D(-2, -1))
```

Q.3(a)-1

```
> from pulp import*
>>> model = LpProblem(sense= LpMinimize)
>>> x = LpVariable(name="x", lowBound=0)
>>> y = LpVariable(name="y", lowBound=0)
>>> model += (x >= 6)
>>> model += (y >= 6)
>>> model += (x + y >= 11)
>>> model += x+y
>>> model
NoName:
MINIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x >= 6

_C2: y >= 6

_C3: x + y >= 11

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
```

Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/abc14133bece40a186ae7ed1a5022fc3-pulp.mps timeMode elapsed branch printingOptions all solution /tmp/abc14133bece40a186ae7ed1a5022fc3-pulp.sol (default strategy 1)
At line 2 NAME MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 15 RHS
At line 19 BOUNDS
At line 20 ENDATA
Problem MODEL has 3 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 0 (-3) rows, 0 (-2) columns and 0 (-4) elements
Empty problem - 0 rows, 0 columns and 0 elements
Optimal - objective value 12
After Postsolve, objective 12, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 12 - 0 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds): 0.01 (Wallclock seconds): 0.00

1

(b)-1

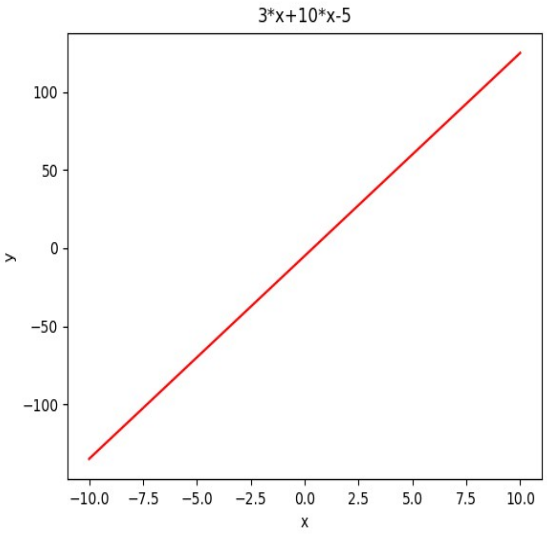
```
>>> from sympy import*
>>> P=Point(-2, 4)
(1)
>>> P.transform(Matrix([[1,0,0],[7,1,0],[0,0,1]]))
Point2D(26, 4)
(2)
>>> P.scale(7/2,4)
Point2D(-7, 16)
(3)
>>> P.transform(Matrix([[1,2,0],[4,1,0],[0,0,1]]))
```

```
Point2D(14, 0)
(4)
>>> from math import*
>>> angle=radians(45)
>>> P.rotate(angle)
Point2D(-53033008588991/12500000000000,
14142135623731/10000000000000)
>>>
```

slip no 12

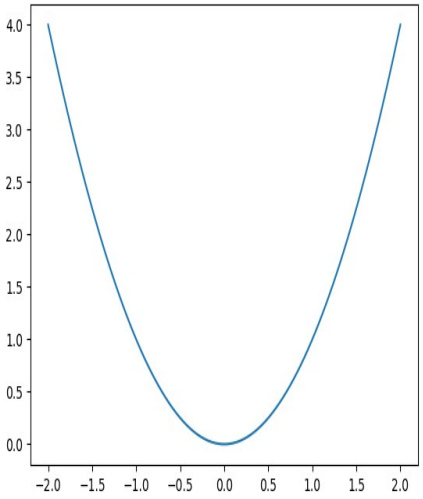
Q1)a

```
>>> from pylab import*
>>> import numpy as np
>>> from math import*
>>> x=np.linspace(-10,10,100)
>>> y=3*x+10*x-5
>>> plot(x,y,'r')
[<matplotlib.lines.Line2D object at
0x797e77ad8940>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('3*x+10*x-5')
Text(0.5, 1.0, '3*x+10*x-5')
>>> show()
>>>
```



c)>>> from pylab import*
>>> import numpy as np
>>> x=np.linspace(-2,2,100)

```
>>> y=x**2
>>> plot(x,y)
[<matplotlib.lines.Line2D object at 0x797e747189a0>]
>>> show()
>>>
```



Q2)

```
a)>>> from sympy import*
>>> S=Segment(Point(1,0),Point(2,-1))
>>> S.rotate(pi)
Segment2D(Point2D(-1, 0), Point2D(-2, 1))
>>>


b)>>> P=Polygon((0,0),5,n=8)
>>> P.area
(400 - 200*sqrt(2))/(-4 + 4*sqrt(2))
>>> P.perimeter
40*sqrt(2 - sqrt(2))
>>>
c)>>> x,y=symbols('X,Y')
>>> X=Point(1,2)
>>> Y=Point(2,-2)
>>> Z=Point(-1,2)
>>> T=Triangle(X,Y,Z)
>>> T.area
-4
>>> T.perimeter
sqrt(17) + 7
>>>
```

Q3)a)

```
1)>>> from pulp import*
>>> model=LpProblem(sense=LpMinimize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model +=(x+y>=5)
>>> model +=(x>=4)
>>> model +=(y<=2)
```

```
>>> model +=3.5*x+2*y
>>> model
NoName:
MINIMIZE
3.5*x + 2*y + 0.0
SUBJECT TO
_C1: x + y >= 5

_C2: x >= 4

_C3: y <= 2

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/87877d0bcc1d4fff958324e9d486f189-pulp.mps timeMode elapsed branch
printingOptions all solution /tmp/87877d0bcc1d4fff958324e9d486f189-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 15 RHS
At line 19 BOUNDS
At line 20 ENDATA
Problem MODEL has 3 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 1 (-2) rows, 2 (0) columns and 2 (-2) elements
0  Obj 14 Primal inf 0.999999 (1)
1  Obj 16
Optimal - objective value 16
After Postsolve, objective 16, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 16 - 1 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.01

1
>>> model.objective.value()
16.0
>>> x.value()
4.0
>>> y.value()
1.0
>>>
```

Q3)b)

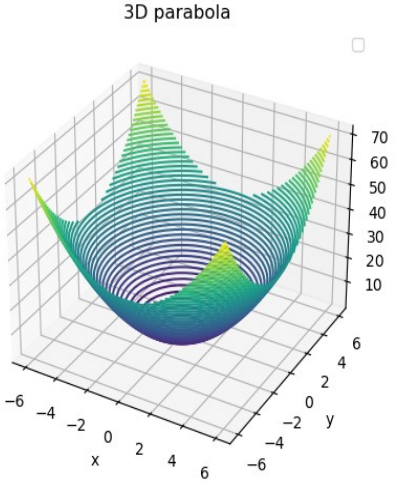
```
1)>>> from sympy import*
>>> P=Point(-2,4)
1)
>>> P.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
Point2D(2, 4)
2)
>>> P.scale(6,0)
Point2D(-12, 0)
3)
>>> P.scale(0,4.1)
Point2D(0, 82/5)
4)
>>> P.transform(Matrix([[1,7/2,0],[0,1,0],[0,0,1]]))
Point2D(-2, -3)
>>>
```

slip 13)

Q1)

a)>>> from pylab import*
>>> import numpy as np
>>> x=np.linspace(-1,1,100)
>>> f=x**2
>>> g=x**3
>>> plot(x,f)
[<matplotlib.lines.Line2D object at 0x713c87757730>]
>>> plot(x,g)
[<matplotlib.lines.Line2D object at 0x713c877577f0>]
>>> show()
>>>

b)>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from pylab import*
>>> def f(x,y):
... return x**2+y**2
...
>>> x=np.linspace(-6,6,30)
>>> y=np.linspace(-6,6,30)
>>> X,Y=np.meshgrid(x,y)
>>> Z=f(X,Y)
>>> ax=axes(projection='3d')
>>> ax.contour3D(X,Y,Z,50)
<matplotlib.contour.QuadContourSet object at 0x713c84696890>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')



```
Text(0.5, 0.5, 'y')
>>> title('3D parabola')
Text(0.5, 0.92, '3D parabola')
>>> legend()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x713c84696890>
>>> show()
>>>
```

Q2)

```
a)>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(1,0)
>>> B=Point(2,-1)
>>> C=Point(-1,3)
>>> T=Triangle(A,B,C)
>>> P=Point(0,3)
>>> Q=Point(1,3)
>>> L=Point(P,Q)
>>> L=Line(P,Q)
>>> T.reflect(L)
Triangle(Point2D(1, 6), Point2D(2, 7), Point2D(-1, 3))
>>>
```

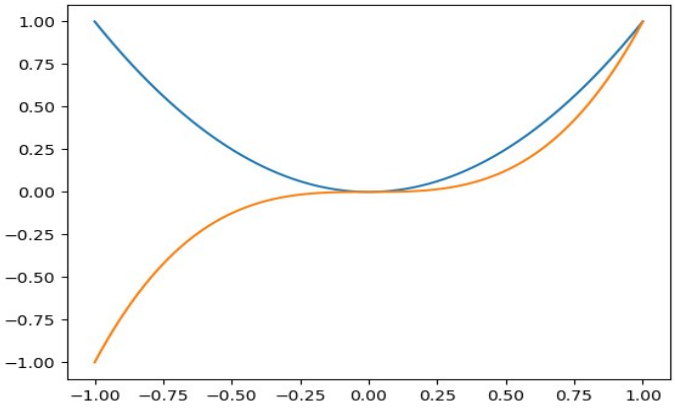
```
b)>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(5,0)
>>> C=Point(3,3)
>>> T=Triangle(A,B,C)
>>> T.area
15/2
>>> T.perimeter
sqrt(13) + 3*sqrt(2) + 5
>>>
Q3)a)1)>>> from pulp import*
>>> model=LpProblem(sense=LpMinimize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model+=(x+y<=7)
>>> model+=(2*x+5*y<=1)
>>> model+=5*x+3*y
>>> model
NoName:
MINIMIZE
5*x + 3*y + 0
SUBJECT TO
_C1: x + y <= 7

_C2: 2 x + 5 y <= 1

VARIABLES
```

```
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/f29929be56c448e39dca3452ca6afe2a-pulp.mps timeMode elapsed branch
printingOptions all solution /tmp/f29929be56c448e39dca3452ca6afe2a-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 0 (-2) rows, 0 (-2) columns and 0 (-4) elements
Empty problem - 0 rows, 0 columns and 0 elements
Optimal - objective value 0
After Postsolve, objective 0, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 0 - 0 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.01   (Wallclock seconds):       0.00

1
>>> model.objective.value()
0.0
>>> x.value()
0.0
>>> y.value()
0.0
>>>
Q3)b)1)
>>> from sympy import*
>>> P=Point(-2,4)
1)
>>> P.transform(Matrix([[1,0,0],[7,1,0],[0,0,1]]))
Point2D(26, 4)
2)
>>> P.scale(7/2,4)
Point2D(-7, 16
3)
>>> P.transform(Matrix([[1,4,0],[7,1,0],[0,0,1]]))
Point2D(26, -4)
>>>
4)
>>> from math import*

>>> angle=radians(48)
```

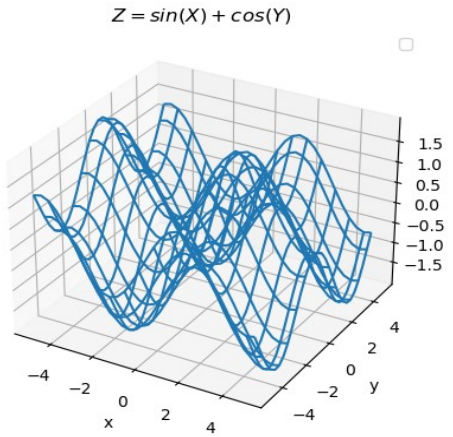
```
>>> P.rotate(angle)
Point2D(-431084051462729/100000000000000, 929869355063/781250000000)
>>>
```

```
>>> P.rotate(angle)
Point2D(-431084051462729/100000000000000, 929869355063/781250000000)
>>>
```

slip 14




```
Q1)a)>>> from pylab import*
>>> import numpy as np
>>> x=np.linspace(-1,1,1000)
>>> f=x**2
>>> g=x**3
>>> plot(x,f)
[<matplotlib.lines.Line2D object at
0x703d534689d0>]
>>> plot(x,g)
[<matplotlib.lines.Line2D object at 0x703d53468c40>]
>>> show()
>>>

c)>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from pylab import*
>>> def f(X,Y):
...  return np.sin(X)+np.cos(Y)
...
>>> x=np.linspace(-5,5,30)
>>> y=np.linspace(-5,5,30)
>>> X,Y=np.meshgrid(x,y)
>>> Z=f(X,Y)
>>> ax = axes(projection='3d')
>>> ax.plot_wireframe(X,Y,Z,rstride=2,cstride=2)
<mpl_toolkits.mplot3d.art3d.Line3DCollection object at 0x70b59038d690>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('$Z=sin(X)+cos(Y)$')
Text(0.5, 0.92, '$Z=sin(X)+cos(Y)$')
>>> legend()
<matplotlib.legend.Legend object at
0x70b592d07e80>
>>> show()
>>>
Q2)a)>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(5,3)
```



```
>>> B=Point(1,4)
>>> S=Segment(A,B)
>>> S.reflect(Line(x-y+1))
Segment2D(Point2D(2, 6), Point2D(3, 2))
>>>
```

b)
```
>>> A=Point(0,0)
>>> B=Point(2,0)
>>> C=Point(2,3)
>>> D=Point(1,6)
>>> P=Polygon(A,B,C,D)
>>> P.rotate(pi)
Polygon(Point2D(0, 0), Point2D(-2, 0), Point2D(-2, -3), Point2D(-1, -6))
>>>
```

c)
```
>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(5,0)
>>> C=Point(3,3)
>>> T=Triangle(A,B,C)
>>> T.area
15/2
>>> T.perimeter
sqrt(13) + 3*sqrt(2) + 5
>>>
```

Q3)a)
```
>>> from pulp import*
>>> model=LpProblem(name="Small-problem",sense=LpMaximize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model += (4*x+6*y <= 24)
>>> model += (5*x+3*y <= 15)
>>> model += 150*x+75*y
>>> model
Small-problem:
MAXIMIZE
150*x + 75*y + 0
SUBJECT TO
_C1: 4 x + 6 y <= 24

_C2: 5 x + 3 y <= 15

VARIABLES
x Continuous
y Continuous

>>> model.solve()
```

```
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/002063f8abc0457e9b0920157d83f3d5-pulp.mps max timeMode elapsed
branch printingOptions all solution /tmp/002063f8abc0457e9b0920157d83f3d5-pulp.sol (default
strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
0  Obj -0 Dual inf 225 (2)
0  Obj -0 Dual inf 225 (2)
1  Obj 450
Optimal - objective value 450
Optimal objective 450 - 1 iterations time 0.042
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.12   (Wallclock seconds):       0.04

1
>>> x.value()
3.0
>>> y.value()
0.0


Q3)b)1)
>> from sympy import*
>>> P=Point(2,-3)
1)
>>> P.transform(Matrix([[1,0,0],[0,-1,0],[0,0,1]]))
Point2D(2, 3)
2)
>>> P.scale(2,0)
Point2D(4, 0)
3)
>>> P.scale(0,1.5)
Point2D(0, -9/2)
4)
>>> x,y=symbols('x,y')
>>> P.reflect(Line(x-y+0))
Point2D(-3, 2)
>>>
```

Slip no :- 15

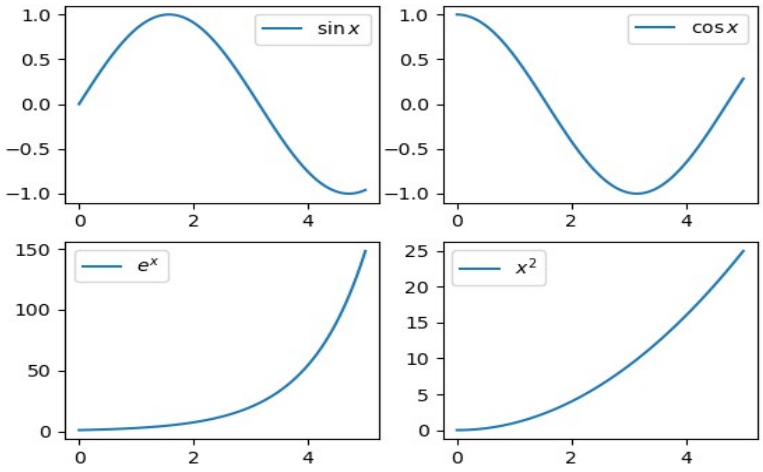
Q1)..
a)...

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(5,0)
>>> C=Point(3,3)
>>> T=Triangle(A,B,C)
>>> T.area
15/2
```

b)...

```
>>> from pylab import*
>>> import numpy as np
>>> from math import *
>>> x=np.linspace(0,5,100)
>>> y1=np.sin(x)
>>> y2=np.cos(x)
>>> y3=np.exp(x)
>>> y4=x**2
>>> subplot(2,2,1)
<AxesSubplot:>
>>> plot(x,y1,label="$\sin x$")
[<matplotlib.lines.Line2D object at 0x7fc4c3c5bac0>]
>>> legend()
<matplotlib.legend.Legend object at 0x7fc4c3ca2bf0>
>>> subplot(2,2,2)
<AxesSubplot:>
>>> plot(x,y2,label="$\cos x$")
[<matplotlib.lines.Line2D object at 0x7fc4bb18d000>]
>>> legend()
<matplotlib.legend.Legend object at 0x7fc4c3c592d0>
>>> subplot(2,2,3)
<AxesSubplot:>
>>> plot(x,y3,label="$e^ x$")
[<matplotlib.lines.Line2D object at 0x7fc4bb1b7460>]
```

```
>>> legend()
<matplotlib.legend.Legend object at 0x7fc4c3ceaec0>
>>> subplot(2,2,4)
<AxesSubplot:>
>>> plot(x,y4,label="$x^2$")
[<matplotlib.lines.Line2D object at 0x7fc4bb2018d0>]
>>> legend()
<matplotlib.legend.Legend object at 0x7fc4bb1b79a0>
>>> show
<function show at 0x7fc4c43d1d80>
>>> show()
```



Q2)...

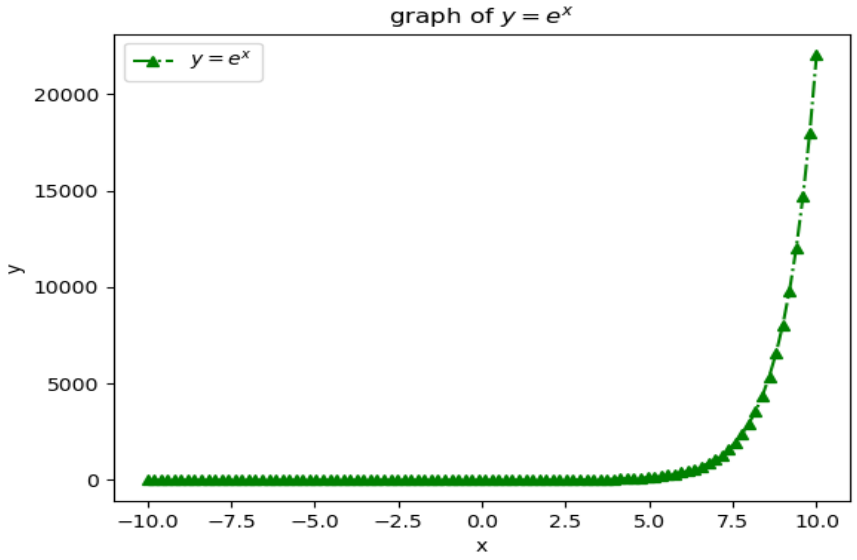
a)..

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(1,2)
>>> B=Point(2,-2)
>>> C=Point(-1,2)
>>> T=Triangle(A,B,C)
>>> T.rotate(pi)
Triangle(Point2D(-1, -2), Point2D(-2, 2), Point2D(1, -2))
```

b)...

```
 >>> from pylab import*
```

```
>>> import numpy as np
>>> x=np.linspace(-10,10,100)
>>> y=np.exp(x)
>>> plot(x,y,"-.^g",label="$y=e^{x}$")
[<matplotlib.lines.Line2D object at 0x7fc34ddf6260>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('graph of $y=e^{x}$')
Text(0.5, 1.0, 'graph of $y=e^{x}$')
>>> legend()
<matplotlib.legend.Legend object at 0x7fc34ddaf040>
>>> show()
```



Q3)...a)..
1)..

```
>>> from pulp import*
>>> model=LpProblem(sense=LpMinimize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model+=(x+y>=5)
>>> model+=(x>=4)
>>> model+=(y<=2)
>>> model+= 3.5*x+2*y
```

```
>>> model
NoName:
MINIMIZE
3.5*x + 2*y + 0.0
SUBJECT TO
_C1: x + y >= 5

_C2: x >= 4

_C3: y <= 2

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/ecbb0dcab23d411980e98be47356f5c8-pulp.mps
timeMode elapsed branch printingOptions all solution
/tmp/ecbb0dcab23d411980e98be47356f5c8-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 15 RHS
At line 19 BOUNDS
At line 20 ENDATA
Problem MODEL has 3 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 1 (-2) rows, 2 (0) columns and 2 (-2) elements
0  Obj 14 Primal inf 0.999999 (1)
1  Obj 16
Optimal - objective value 16
After Postsolve, objective 16, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 16 - 1 iterations time 0.072, Presolve 0.00
Option for printingOptions changed from normal to all
```

Total time (CPU seconds): 0.08 (Wallclock seconds): 0.02

1
>>> model.objective.value()
16.0
>>> x.value()
4.0
>>> y.value()
1.0

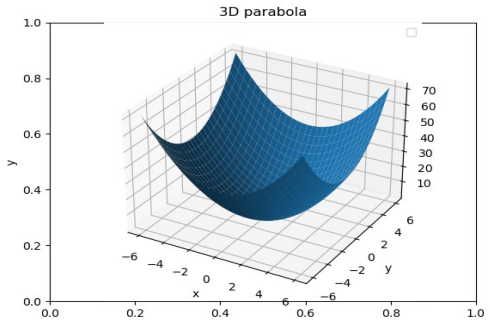
b)...
>>> from sympy import*
>>> P=Point(-2,4)
1)...>>> x,y=symbols('x,y')
   >>> P.reflect(Line(-x+y-1))
   Point2D(3, -1)

2)..>>> P.scale(0,1.5)
        Point2D(0, 6)
3)..>>> P.transform(Matrix([[1,2,0],[0,1,0],[0,0,1]]))
         Point2D(-2, 0)
4)..>>> P.rotate(pi/4)
        Point2D(-3*sqrt(2), sqrt(2))

Q1.A

Slip No.17

```
>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from pylab import*
>>> def f(x, y):
...  return x ** 2 + y **2
...
>>> x=np.linspace(-6,6,30)
>>> y=np.linspace(-6,6,30)
>>> x,y=np.meshgrid(x,y)
>>> z=f(x,y)
>>> ax=axes(projection='3d')
>>> ax.plot_surface(x,y,z)
<mpl_toolkits.mplot3d.art3d.Poly3DCollection object at 0x70799de07b50>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('3D parabola')
Text(0.5, 0.92, '3D parabola')
>>> legend()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x70799ca0bac0>
>>> show()
```



Q1.C

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(-5,2)
>>> B=Point(1,3)
>>> S=Segment(A,B)
>>> S.reflect(Line(x-y))
Segment2D(Point2D(2, -5), Point2D(3, 1))
>>>
```

Q2.A

```
>>> from sympy import*
>>> S=Segment(Point(1,0),Point(2,-1))
>>> S.rotate(pi)
Segment2D(Point2D(-1, 0), Point2D(-2, 1))
```

Q2.C

```
>>> from sympy import*
>>> A=Point(0,0)
>>> B=Point(1,0)
>>> C=Point(2,2)
>>> D=Point(1,4)
>>> P=Polygon(A,B,C,D)
>>> P.area
4
>>> P.perimeter
1 + sqrt(17) + 2*sqrt(5)
>>>
```

Q3.A.1

```
>>> from pulp import*
>>> model=LpProblem(name="small-problem",sense=LpMaximize)
>>> x=LpVariable(name='x',lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> z=LpVariable(name="z",lowBound=0)
>>> w=LpVariable(name="w",lowBound=0)
>>> model += (4*x+6*y-5*z-4*x >=-20)
>>> model += (-8*x-3*y+3*z+2*w <=20)
>>> model += 4*x+y+3*z+5*w
>>> model
small-problem:
MAXIMIZE
5*w + 4*x + 1*y + 3*z + 0
SUBJECT TO
_C1: 0 x + 6 y - 5 z >= -20

_C2: 2 w - 8 x - 3 y + 3 z <= 20

VARIABLES
w Continuous
x Continuous
y Continuous
z Continuous
```

Q3.B.1

1)

```
>>> from sympy import*
>>> P=Point(3,-1)
```

2)

```
>>> P.scale(0,1.5)
Point2D(0, -3/2)
```

3)

```
>>> P.transform(Matrix([[1,-2,0],[1,4,0],[0,0,1]]))
Point2D(2, -10)
```

4)

```
>>> from math import*
>>> angle=radians(30)
>>> P.rotate(angle)
Point2D(77451905283833/25000000000000, 633974596215561/1000000000000000)
```

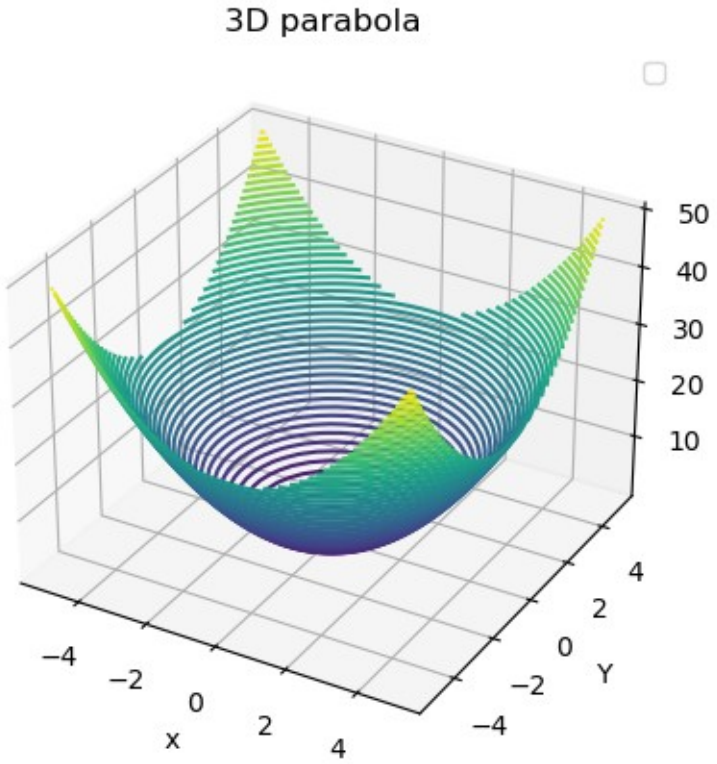
>>>

Slip No:18

Q.1

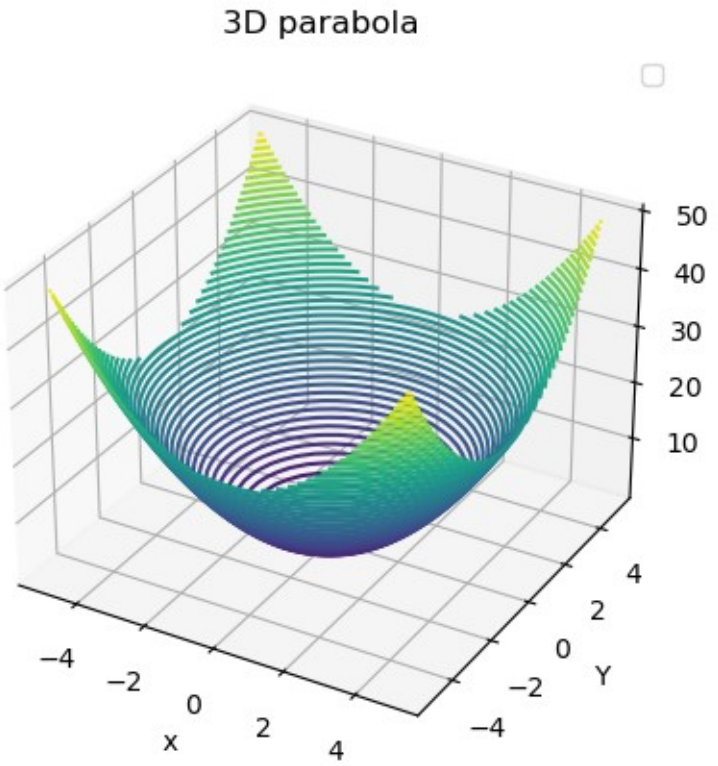
(C)

```
>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from pylab import*
>>> def f(x,y):
...     return x**2+y**2
...
>>> x=np.linspace(-5,5,30)
>>> y=np.linspace(-5,5,30)
>>> X,Y=np.meshgrid(x,y)
>>> Z=f(X,Y)
>>> ax=axes(projection='3d')
>>> ax.contour3D(X,Y,Z,50)
<matplotlib.contour.QuadContourSet object at 0x7fe51798ab30>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('Y')
Text(0.5, 0.5, 'Y')
>>> title('3D parabola')
Text(0.5, 0.92, '3D parabola')
>>> legend()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x7fe518d4db40>
>>> show()
```



(b)

```
from pylab import*
>>> import numpy as np
>>> x=np.linspace(-10,10,100)
>>> y=2*x**2-4*x+5
>>> plot(x,y,"-m",label="$y=y=2*x**2-4*x+5")
[<matplotlib.lines.Line2D object at 0x7f7940ee9360>]
>>> plot(x,y,"- -m",label="$y=2*x**2-4*x+5$")
[<matplotlib.lines.Line2D object at 0x7f7940f1c760>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('Graph of $y=2*x**2-4*x+5$')
Text(0.5, 1.0, 'Graph of $y=2*x**2-4*x+5$')
>>> legend()
<matplotlib.legend.Legend object at 0x7f7941d914e0>
>>> show()
```



Q2

(b)

```
>>> x,y=symbols('x,y')
>>> A=Point(1,2)
>>> B=Point(2,-2)
>>> c=Point(-1,2)
>>> T=Triangle(A,B,c)
>>> T.rotate(pi/2)
Triangle(Point2D(-2, 1), Point2D(2, 2), Point2D(-2, -1))
```

Q 3. A.

(a)

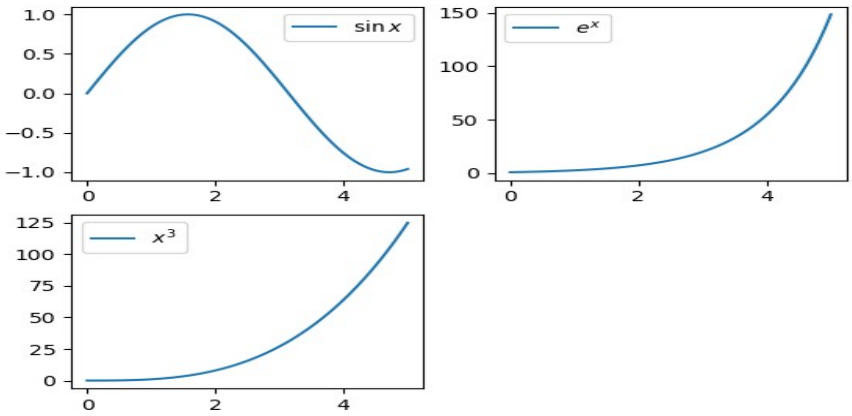
```
>>> from pulp import*
>>> model=LpProblem(sense=LpMinimize)
>>> x = LpVariable(name="x",lowBound=0)
>>> y = LpVariable(name="y",lowBound=0)
>>> model+=(x>=6)
>>> model+=(y>=6)
>>> model+=(x+y<=11)
>>> model+=x+y
>>> model
NoName:
MINIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x >= 6

_C2: y >= 6

_C3: x + y <= 11

VARIABLES
x Continuous
y Continuous

>>> model.solve
<bound method LpProblem.solve of NoName:
MINIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x >= 6

_C2: y >= 6

_C3: x + y <= 11

VARIABLES
x Continuous
y Continuous
```

Q 3. B
(b)

```
>>> from sympy import*
>>> P=Point(3,-1)
>>> P.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
Point2D(-3, -1)
>>> P.scale(1/2,3)
Point2D(3/2, -3)
>>> P.transform(Matrix([[1,4,0],[7,1,0],[0,0,1]]))
Point2D(-4, 11)
>>> from math import*
>>> angle=radians(60)
>>> P.rotate(angle)
Point2D(59150635094611/25000000000000, 52451905283833/25000000000000)
```

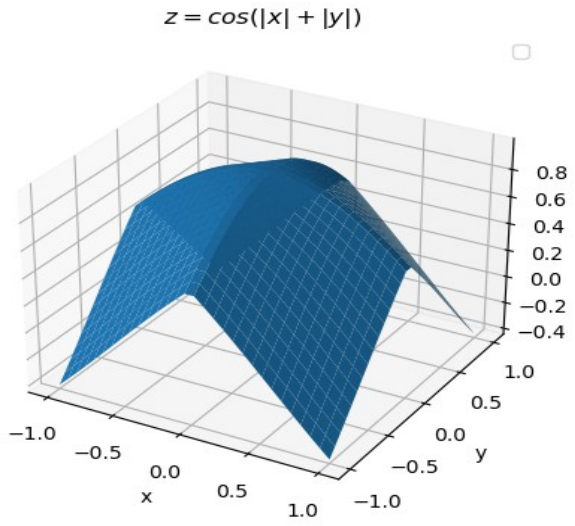
Slip No :-19

Q1 a)>>> from pylab import*
>>> import numpy as np
>>> from math import*
>>> x=x=np.linspace(0,5,100)
>>> y1=np.sin(x)
>>> y2=np.exp(x)
>>> y3=x**3
>>> subplot(2,2,1)
<AxesSubplot:>
>>> plot(x,y1,label="$\sin x$")
[<matplotlib.lines.Line2D object at 0x7634ef114910>]
>>> legend()
<matplotlib.legend.Legend object at 0x7634ef1148b0>
>>> subplot(2,2,2)
<AxesSubplot:>
>>> plot(x,y2,label="e^x")
[<matplotlib.lines.Line2D object at 0x7634ef146da0>]
>>> legend()
<matplotlib.legend.Legend object at 0x7634ef147a90>
>>> subplot(2,2,3)
<AxesSubplot:>
>>> plot(x,y3,label="x^3")
[<matplotlib.lines.Line2D object at 0x7634e60c45e0>]
>>> legend()
<matplotlib.legend.Legend object at 0x7634ef147af0>
>>> show()



b)>>> import numpy as np
>>> from math import*
>>> from pylab import*
>>> def f(X,Y):
... return np.cos(abs(X)+abs(Y))
...
>>> x=np.linspace(-1,1,30)
>>> y=np.linspace(-1,1,30)
>>> X,Y=np.meshgrid(x,y)

```
>>> Z=f(X,Y)
>>> ax=axes(projection='3d')
>>> ax.plot_surface(X,Y,Z)
<mpl_toolkits.mplot3d.art3d.Poly3DCollection object at 0x7e174bcb2020>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('$z=cos(|x|+|y|)$')
Text(0.5, 0.92, '$z=cos(|x|+|y|)$')
>>> legend()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore
are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x7e174df7c2b0>
>>> show()
```



Q2 a)
```
>>>from sympy import*
>>> R=Ray(Point(0,0),Point(4,4))
>>> R.rotate(-pi/2)
Ray2D(Point2D(0, 0), Point2D(4, -4))
```

b)
```
>>> x,y=symbols('x,y')
>>> A=Point(1,0)
>>> B=Point(2,-1)
>>> C=Point(-1,3)
>>> T=Triangle(A,B,C)
>>> P=Point(0,3)
>>> Q=Point(1,3)
>>> L=Line(P,Q)
>>> T.reflect(L)
Triangle(Point2D(1, 6), Point2D(2, 7), Point2D(-1, 3))
```

c)
```
>>> A=Point(0,0)
>>> B=Point(1,0)
>>> C=Point(2,2)
>>> D=Point(1,4)
```

```
>>> P=Polygon(A,B,C,D)
>>> P.area
4
>>> P.perimeter
1 + sqrt(17) + 2*sqrt(5)
```

Q3 a)i)

```
>>> from pulp import*
>>> model=LpProblem(sense=LpMaximize)
>>> x=LpVariable(name="x",lowBound=0)
>>> x=LpVariable(name="y",lowBound=0)
>>> x=LpVariable(name="y",lowBound=0)
>>>
>>>
>>> from pulp import*
>>> model=LpProblem(sense=LpMaximize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> z=LpVariable(name="z",lowBound=0)
>>> model += (2*x+3*y<=8)
>>> model += (2*y+5*z<=10)
>>> model += (3*x+2*y+4*z<=15)
>>> model += 3*x+5*y+4*z
>>> model
NoName:
MAXIMIZE
3*x + 5*y + 4*z + 0
SUBJECT TO
_C1: 2 x + 3 y <= 8

_C2: 2 y + 5 z <= 10

_C3: 3 x + 2 y + 4 z <= 15

VARIABLES
x Continuous
y Continuous
z Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/5bbc6e5487e84759b1298fc6cdffb153-pulp.mps max timeMode elapsed branch printingOptions all solution /tmp/5bbc6e5487e84759b1298fc6cdffb153-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 19 RHS
At line 23 BOUNDS
```

```
At line 24 ENDATA
Problem MODEL has 3 rows, 3 columns and 7 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 3 (0) rows, 3 (0) columns and 7 (0) elements
0  Obj -0 Dual inf 13 (3)
0  Obj -0 Dual inf 13 (3)
3  Obj 18.658537
Optimal - objective value 18.658537
Optimal objective 18.65853659 - 3 iterations time 0.042
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.12   (Wallclock seconds):       0.04

1
>>> model.objective.value()
18.658536500000004
>>> x.value()
2.1707317
>>> y.value()
1.2195122
>>> z.value()
1.5121951
>>>
```

b)i)

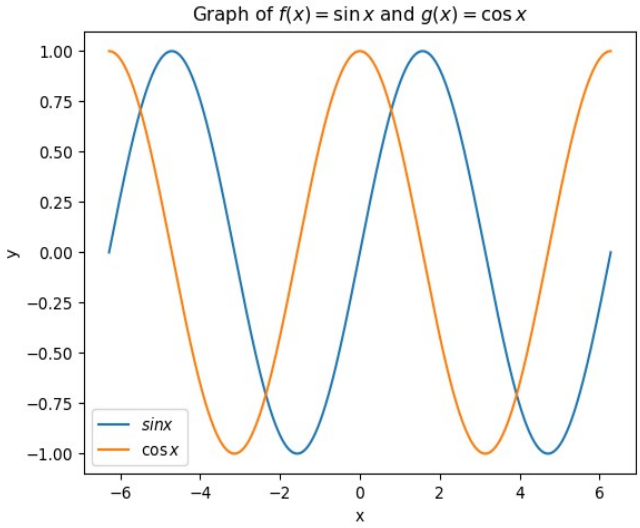
```
I)>>> from sympy import*
>>> P=Point(-2,4)
>>> x,y=symbols('x y')
>>> from math import*
>>> angle=radians(48)
>>> P.rotate(angle)
Point2D(-431084051462729/100000000000000, 929869355063/781250000000)

II)>>> P.scale(2,0)
Point2D(-4, 0)

III)>>> P.reflect(Line(-2*x+y+3))
Point2D(34/5, -2/5)

Iv)>>> P.transform(Matrix([[1,7,0],[0,1,0],[0,0,1]]))
Point2D(-2, -10)
```

ggggggggg

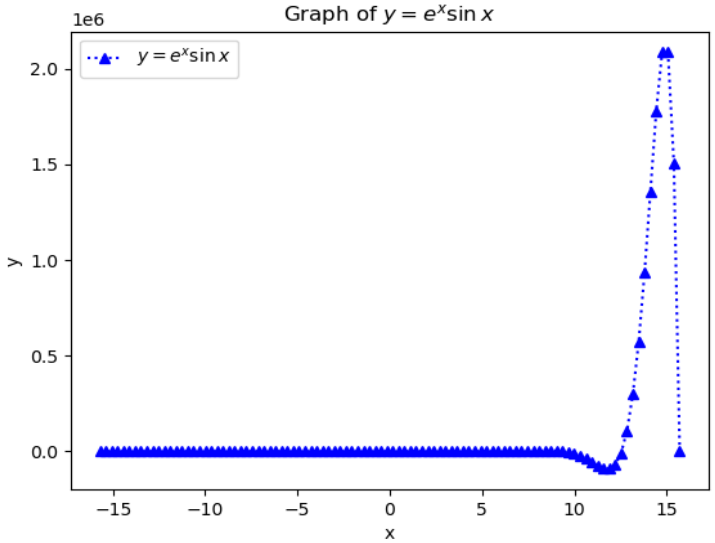
slip no 20

Q1)a)>>> from pylab import*
>>> import numpy as np
>>> from math import*

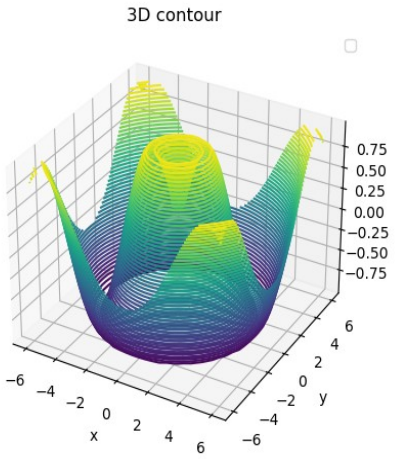
```
>>> x=np.linspace(-2*pi,2*pi,1000)
>>> f=np.sin(x)
>>> g=np.cos(x)
>>> plot(x,f,label="$sin x$")
[<matplotlib.lines.Line2D object at 0x71ce58f67ee0>]
>>> plot(x,g,label="$\cos x$")
[<matplotlib.lines.Line2D object at 0x71ce58fa8130>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('Graph of $f(x)=\sin x$ and $g(x)=\cos x$')
Text(0.5, 1.0, 'Graph of $f(x)=\\sin x$ and $g(x)=\\cos
x$')
>>> legend()
<matplotlib.legend.Legend object at 0x71ce58f67eb0>
>>> show()
>>>
```




```
b)>>> from pylab import *
>>> import numpy as np
>>> from math import*
>>> x=np.linspace(-5*pi,5*pi,100)
>>> y=np.exp(x)*np.sin(x)
>>> plot(x,y,":^b",label="$y=e^x \sin x$")
[<matplotlib.lines.Line2D object at 0x71ce593730d0>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('Graph of $y=e^x \sin x$')
Text(0.5, 1.0, 'Graph of $y=e^x \\sin x$')
>>> legend()
<matplotlib.legend.Legend object at 0x71ce59416c20>
>>> show()
```



c)>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from pylab import*
>>> def f(x,y):
... return np.sin(np.sqrt(x**2+y**2))
...
>>> x=np.linspace(-6,6,30)
>>> y=np.linspace(-6,6,30)
>>> X,Y=np.meshgrid(x,y)
>>> Z = f(X,Y)
>>> ax=axes(projection='3d')
>>> ax.contour3D(X,Y,Z,50)
<matplotlib.contour.QuadContourSet object at
0x71ce59003880>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('3D contour')
Text(0.5, 0.92, '3D contour')
>>> legend()
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x71ce591dbbe0>
>>> show()
>>>



Q2)a)>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(-5,2)
>>> B=Point(3,-4)
>>> S=Segment(A,B)
>>> S.reflect(Line(2*x-y-1))
Segment2D(Point2D(27/5, -16/5), Point2D(-21/5, -2/5))
>>>

b)>>> A=Point(0,0)
>>> B=Point(-2,0)
>>> C=Point(5,5)
>>> D=Point(1,-6)
>>> P=Polygon(A,B,C,D)
>>> P.area
-45/2
>>> P.perimeter
2 + sqrt(37) + sqrt(74) + sqrt(137)
>>>

Q3)a)i)>>> from pulp import*
>>> model=LpProblem(sense=LpMaximize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)

```
>>> model +=(x-y>=1)
>>> model +=(x+y>=2)
>>> model +=x+y
>>> model
NoName:
MAXIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x - y >= 1

_C2: x + y >= 2

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/88b40a8143214bb3b2986c3cbead56e4-pulp.mps max timeMode elapsed branch printingOptions all solution /tmp/88b40a8143214bb3b2986c3cbead56e4-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve thinks problem is unbounded
Analysis indicates model infeasible or unbounded
0  Obj -0 Primal inf 2.9999998 (2) Dual inf 1.9999998 (2)
1  Obj 2e+10
1  Obj 1e+11
Dual infeasible - objective value 1e+11
DualInfeasible objective 1e+11 - 1 iterations time 0.002

Result - Linear relaxation unbounded

Enumerated nodes:           0
Total iterations:           0
Time (CPU seconds):         0.01
Time (Wallclock Seconds):   0.00

Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.01   (Wallclock seconds):       0.00

-2
```

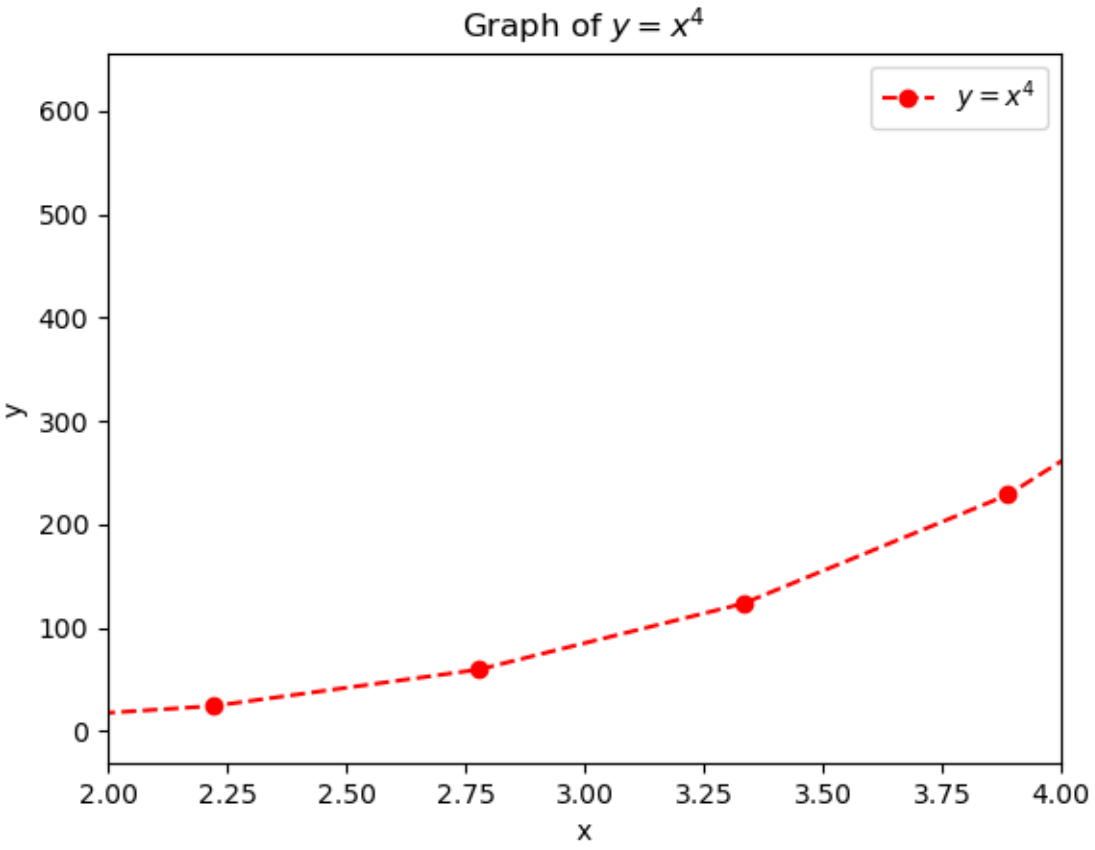
```
>>> model.objective.value()
0.0
>>> x.value()
0.0
>>> y.value()
0.0
>>>
```

Q3)b)i)
```
>>> from sympy import*
>>> P=Point(-2,4)
```
1)
```
>>> from math import*
>>> angle=radians(48)
>>> P.rotate(angle)
Point2D(-431084051462729/100000000000000, 929869355063/781250000000)
```
2)
```
>>> P.scale(2,0)
Point2D(-4, 0)
```
3)
```
>>> x,y=symbols('x,y')
>>> P.reflect(Line(-2*x+y+3))
Point2D(34/5, -2/5)
>>>
```
4)
```
>> P.transform(Matrix([[1,7,0],[0,1,0],[0,0,1]]))
Point2D(-2, -10)
>>>
```

Slip 21

Q1) a)

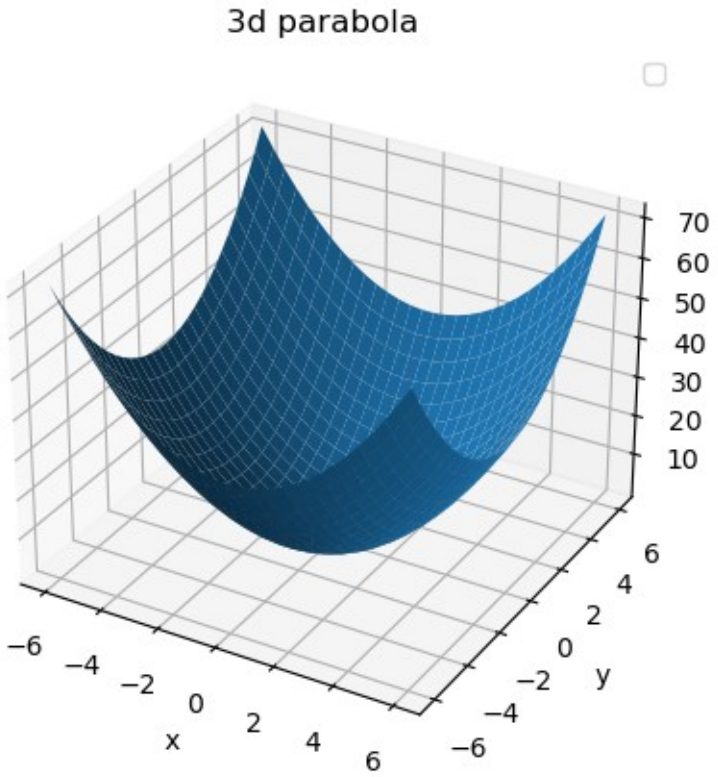
```
>>> from pylab import *
>>> import numpy as np
>>> x=np.linspace(0,5,10)
>>> y=x**4
>>> plot(x,y,"--or",label="$y=x^4$")
[<matplotlib.lines.Line2D object at 0x727bee706110>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> xlim([2,4])
(2.0, 4.0)
>>> title('Graph of $y=x^4$')
Text(0.5, 1.0, 'Graph of $y=x^4$')
>>> legend()
<matplotlib.legend.Legend object at 0x727bee6c3e80>
>>> show()
```



b)

```
>>> import numpy as np
>>> from pylab import *
```

```
>>> def f(x,y):
...     return x**2+y**2
...
>>> x=np.linspace(-6,6,30)
>>> y=np.linspace(-6,6,30)
>>> x,y=np.meshgrid(x,y)
>>> z=f(x,y)
>>> ax=axes(projection='3d')
>>> ax.plot_surface(x,y,z)
<mpl_toolkits.mplot3d.art3d.Poly3DCollection object at 0x7e5ebe88dba0>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('3d parabola')
Text(0.5, 0.92, '3d parabola')
>>> legend()
>>> show()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x7e5ebfc3fdf0>
```




```
>>> A=Point(2,5)
>>> B=Point(4,73)
>>> A1=A.transform(Matrix([[2,3,0],[4,1,0],[0,0,1]]))
```

```
>>> B1=B.transform(Matrix([[2,3,0],[4,1,0],[0,0,1]]))
>>> L=Segment(A1,B1)
>>> L.midpoint
Point2D(162, 48)
```
c)
```
>>> x,y=symbols('x,y')
>>> l=Line(2*x+y-3)
>>> points=l.points
>>> p=points[0]
>>> q=points[1]
>>> p1=p.transform(Matrix([[1,-3,0],[2,1,0],[0,0,1]]))
>>> q1=q.transform(Matrix([[1,-3,0],[2,1,0],[0,0,1]]))
>>> l1=Line(p1,q1)
>>> l1.equation()
5*x - 3*y – 21
```

Q3)a)
```
>>> from pulp import *
>>> model=LpProblem(sense=LpMinimize)
>>> x=LpVariable(name="x",lowBound=0)
>>> model+=(x+y<=3)
>>> model+=(x-y>=2)
>>> model+=4*x+2*y
>>> model
NoName:
MINIMIZE
4*x + 2*y + 0
SUBJECT TO
_C1: x + y <= 3

_C2: x - y >= 2

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/b197b89da6004e2494b6d40172b1c950-pulp.mps timeMode elapsed branch printingOptions all solution /tmp/b197b89da6004e2494b6d40172b1c950-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
```

Option for timeMode changed from cpu to elapsed
Presolve 0 (-2) rows, 0 (-2) columns and 0 (-4) elements
Empty problem - 0 rows, 0 columns and 0 elements
Optimal - objective value 8
After Postsolve, objective 8, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 8 - 0 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds): 0.13 (Wallclock seconds): 0.03

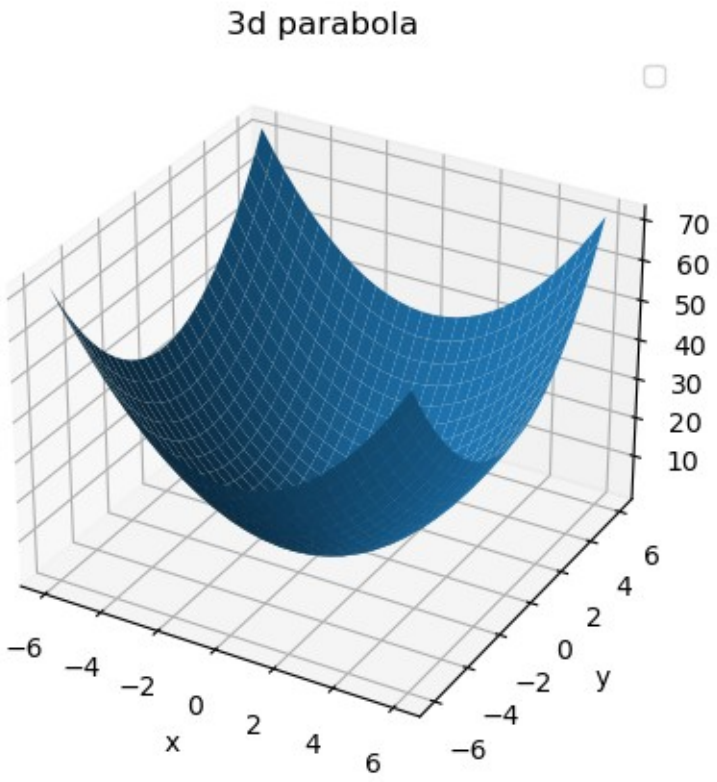
1
b)
i)

```
>>> P=Point(-2,4)
>>> x,y=symbols('x,y')
>>> P.reflect(Line(3*x+4*y-5))
Point2D(-16/5, 12/5)
>>> P.scale(6,0)
Point2D(-12, 0)
>>> P.scale(0,4.1)
Point2D(0, 82/5)
>>> P.reflect(Line(-2*x+y-3))
Point2D(2, 2)
```

Slip 22

Q 1) b)

```
>>> import numpy as np
>>> from pylab import *
>>> def f(x,y):
...     return x**2+y**2
...
>>> x=np.linspace(-6,6,30)
>>> y=np.linspace(-6,6,30)
>>> x,y=np.meshgrid(x,y)
>>> z=f(x,y)
>>> ax=axes(projection='3d')
>>> ax.plot_surface(x,y,z)
<mpl_toolkits.mplot3d.art3d.Poly3DCollection object at 0x7e5ebe88dba0>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('3d parabola')
Text(0.5, 0.92, '3d parabola')
>>> legend()
>>> show()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x7e5ebfc3fdf0>
```



c)

```
>>> from pylab import *
>>> import numpy as np
>>> x=np.linspace(-5,5,100)
>>> y=x*np.sin(1/(x**2))
>>> plot(x,y,label="$y=x \sin (\frac{1}{x^2})$")
[<matplotlib.lines.Line2D object at 0x7e5eb4920c40>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('$y=x \sin (\frac{1}{x^2})$')
Text(0.5, 1.0, '$y=x \\sin (\x0crac{1}{x^2})$')
>>> legend(loc=4)
<matplotlib.legend.Legend object at 0x7e5eb49003d0>
>>> show()
```

Q2) a)

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(2,2)
>>> C=Point(0,2)
>>> T=Triangle(A,B,C)
>>> T.angles[A]
pi/4
```

```
>>> T.angles[B]
pi/4
>>> T.angles[C]
pi/2
```

b)

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> P=Point(3,6)
>>> P.reflect(Line(x-2*y+4))
Point2D(5, 2)
```

c)

```
>>> from sympy import*
>>> x,y=symbols('x,y')
>>> A=Point(0,0)
>>> B=Point(5,0)
>>> C=Point(3,3)
>>> T=Triangle(A,B,C)
>>> T.area
15/2
>>> T.perimeter
sqrt(13) + 3*sqrt(2) + 5
```

Q3) a)

ii)

```
>>> from pulp import*
>>> model=LpProblem(sense=LpMinimize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model+=(x+y<=11)
>>> model+=(x>=6)
>>> model+=(y>=6)
>>> model+=x+y
>>> model
NoName:
MINIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x + y <= 11

_C2: x >= 6

_C3: y >= 6

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
```

Build Date: Feb 14 2022

command line - cbc /tmp/2131362542a64d8caa8924d37c6a012f-pulp.mps timeMode elapsed branch printingOptions all solution /tmp/2131362542a64d8caa8924d37c6a012f-pulp.sol (default strategy 1)
At line 2 NAME MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 15 RHS
At line 19 BOUNDS
At line 20 ENDATA
Problem MODEL has 3 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve determined that the problem was infeasible with tolerance of 1e-08
Analysis indicates model infeasible or unbounded
0 Obj 0 Primal inf 12 (2)
2 Obj 12 Primal inf 0.9999999 (1)
Primal infeasible - objective value 12
PrimalInfeasible objective 12 - 2 iterations time 0.002

Result - Linear relaxation infeasible

Enumerated nodes: 0
Total iterations: 0
Time (CPU seconds): 0.00
Time (Wallclock Seconds): 0.09

Option for printingOptions changed from normal to all
Total time (CPU seconds): 0.00 (Wallclock seconds): 0.09

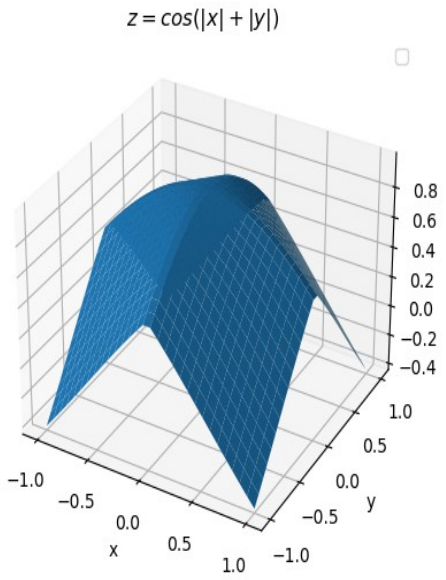
-1

b)
```
>>> x,y=symbols('x,y')
>>> A=Point(1,1)
>>> B=Point(1,4)
>>> T.rotate(pi/2)
Triangle(Point2D(0, 0), Point2D(0, 5), Point2D(-3, 3))
>> from sympy import*
>>> x=Point(0,0)
>>> y=Point(1,0)
>>> x.distance(y)
1
>> P=Point(3,4)
>>> P.transform(Matrix([[1,3,0],[0,1,0],[0,0,1]]))
Point2D(3, 13)
>>>x=Point(-2,5)
```

Slip no :- 23

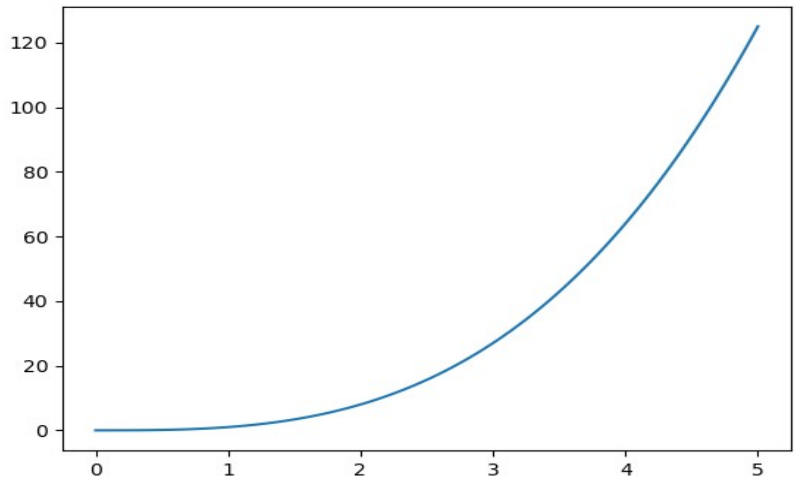
Q1)
..
c)..

```
>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from math import *
>>> from pylab import *
>>> def f(x,y):
...     return np.cos(abs(x)+abs(y))
...
>>> x=np.linspace(-1,1,30)
>>> y=np.linspace(-1,1,30)
>>> x,y=np.meshgrid(x,y)
>>> z=f(x,y)
>>> ax=axes(projection='3d')
>>> ax.plot_surface(x,y,z)
<mpl_toolkits.mplot3d.art3d.Poly3DCollection object at 0x7f9551d5ab60>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('$z=cos(|x|+|y|)$')
Text(0.5, 0.92, '$z=cos(|x|+|y|)$')
>>> legend()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x7f9593cd0d00>
>>> show()
```



b)..
1)..

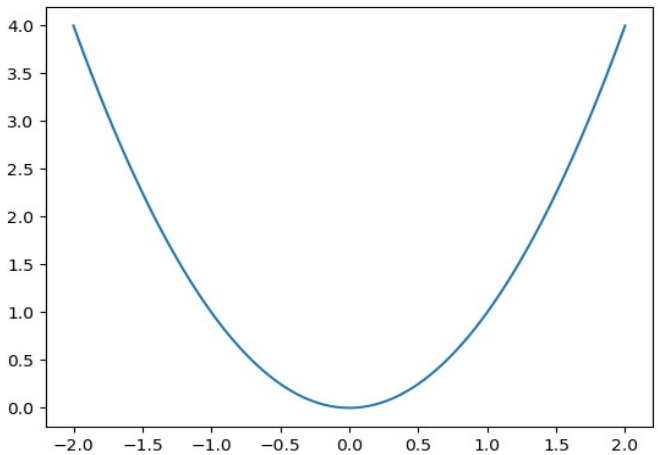
```
>>> from pylab import *
>>> import numpy as np
>>> x=np.linspace(0,5,100)
>>> y=x**3
>>> plot(x,y)
[<matplotlib.lines.Line2D object at 0x7f9551b5ab90>]
>>> show()
>>>
```



2)..

```
>>> from pylab import *
>>> import numpy as np
>>> x=np.linspace(-2,2,100)
>>> y=x**2
>>> plot(x,y)
[<matplotlib.lines.Line2D object at 0x7f9226ab57e0>]
>>> show()
```

>>>



Q2)..
b)..

```
>>> from sympy import *
>>> A=Point(0,0)
>>> B=Point(1,0)
>>> C=Point(2,2)
>>> D=Point(1,4)
>>> p=polygon(A,B,C,D)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'polygon' is not defined
>>> p=Polygon(A,B,C,D)
>>> p.area
4
```

c)

```
>>> from sympy import *
>>> A=Points(0,1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Points' is not defined
>>> A=Point(0,1)
>>> B=Point(-5,0)
>>> C=Point(-3,3)
>>> T=Triangle(A,B,C)
>>> T.area
-13/2
>>> T.Perimeter
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Triangle' object has no attribute 'Perimeter'. Did you mean: 'perimeter'?
>>> T.perimeter
sqrt(26) + 2*sqrt(13)
>>>
```

Q3)..

a)..1)..

```
>>> from pulp import *
>>> model=LpProblem(name="small_Problem",sense=LpMaximize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> z=LpVariable(name="z",lowBound=0)
>>> model +=(2*x+3*y<=8)
>>> model +=(2*x+5*y<=10)
>>> model +=(3*x+2*y+4*z<=15)
>>> model +=3*x+5*y+4*z
>>> model
```

```
small_Problem:
MAXIMIZE
3*x + 5*y + 4*z + 0
SUBJECT TO
_C1: 2 x + 3 y <= 8

_C2: 2 x + 5 y <= 10

_C3: 3 x + 2 y + 4 z <= 15

VARIABLES
x Continuous
y Continuous
z Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/7db7c7d3756a4eca87ba14cb188b49e6-pulp.mps max timeMode elapsed
branch printingOptions all solution /tmp/7db7c7d3756a4eca87ba14cb188b49e6-pulp.sol (default
strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 19 RHS
At line 23 BOUNDS
At line 24 ENDATA
Problem MODEL has 3 rows, 3 columns and 7 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 3 (0) rows, 3 (0) columns and 7 (0) elements
0  Obj -0 Dual inf 13 (3)
0  Obj -0 Dual inf 13 (3)
2  Obj 21
Optimal - objective value 21
Optimal objective 21 - 2 iterations time 0.062
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.06   (Wallclock seconds):       0.02

1
>>> model.objective.value()
21.0
>>> x.value()
0.0
>>> y.value()
2.0
>>> z.value()
2.75
```

b)..1)

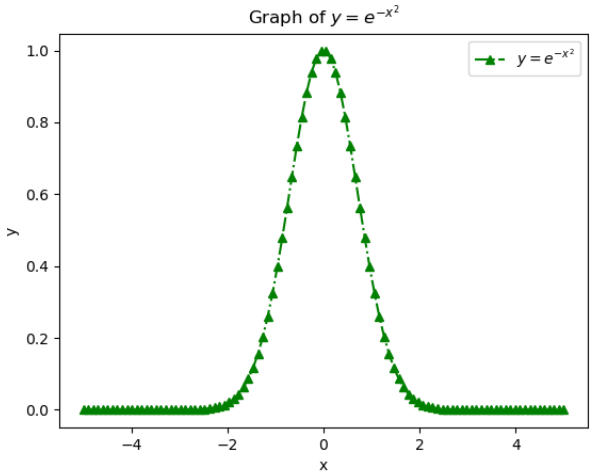
```
>>> from sympy import *
>>> P=Point(3,-1)
1)....P.transform(Matrix([[1,0,0],[0,1,0],[0,0,1]]))
      Point2D(3, -1)
2)....P.rotate(pi/4)
      Point2D(2*sqrt(2), sqrt(2))
3)....P.scale(8,0)
      Point2D(24, 0)
4)...P.transform(Matrix([[1,2,0],[0,1,0],[0,0,1]]))
     Point2D(3, 5)
```

Slip no 24

```
kkw@kkw-HP-ProDesk-400-G4-SFF:~$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

Q 1.a]

```
>>> from pylab import*
>>> import numpy as np
>>> x=np.linspace(-5, 5,100)
>>> y=np.exp(-x**2)
>>> plot(x,y,"-.^g",label="$y=e^{-x^2}$")
[<matplotlib.lines.Line2D object at 0x7e39f877c4c0>]
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0, 0.5, 'y')
>>> title('Graph of $y=e^{-x^2}$')
Text(0.5, 1.0, 'Graph of $y=e^{-x^2}$')
>>> legend()
<matplotlib.legend.Legend object at 0x7e39f873e230>
>>> show()
```



Q 2. c]

```
>>> from sympy import*
>>> A=Point(0,0)
>>> B=Point(1,0)
>>> C=Point(2,2)
>>> D=Point(1,4)
>>> P=Polygon(A,B,C,D)
>>> P.area
4
>>> P.perimeter
```

```
1 + sqrt(17) + 2*sqrt(5)
>>>
```

Q 3. a . i]

```
>>> from pulp import*
>>> model = LpProblem(sense=LpMinimize)
>>> x = LpVariable(name="x",lowBound=0)
>>> y = LpVariable(name="y",lowBound=0)
>>> model += (x+y>=5)
>>> model += (x>=4)
>>> model += (y<=2)
>>> model += 3.5*x+2*y
>>> model
NoName:
MINIMIZE
3.5*x + 2*y + 0.0
SUBJECT TO
_C1: x + y >= 5

_C2: x >= 4

_C3: y <= 2

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/8f554aaa21454bb1ae5f556d660c8048-pulp.mps timeMode elapsed
branch printingOptions all solution /tmp/8f554aaa21454bb1ae5f556d660c8048-pulp.sol (default
strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 15 RHS
At line 19 BOUNDS
At line 20 ENDATA
Problem MODEL has 3 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 1 (-2) rows, 2 (0) columns and 2 (-2) elements
0  Obj 14 Primal inf 0.999999 (1)
1  Obj 16
Optimal - objective value 16
After Postsolve, objective 16, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 16 - 1 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
```

Total time (CPU seconds): 0.00 (Wallclock seconds): 0.08

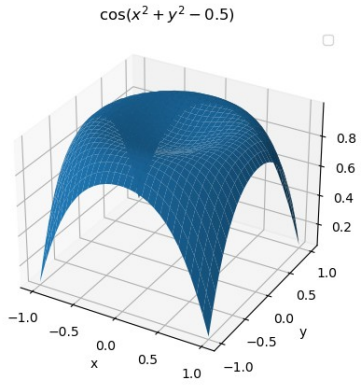
```
1
>>> model.objective.value()
16.0
>>> x.value()
4.0
>>> y.value()
1.0
```

Q 3. b. I]

```
>>> from sympy import*
>>> P=Point(3,-1)
>>> P.transform(Matrix([[1,0,0],[0,1,0],[0,0,1]]))
Point2D(3, -1)
>>> P.scale(2,0)
Point2D(6, 0)
>>> P.scale(0,1.5)
Point2D(0, -3/2)
>>> x,y=symbols('x y')
>>> P.reflect(Line(x+y+0))
Point2D(1, -3)
```

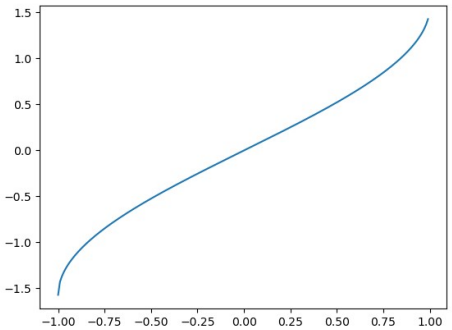
slip_25

Q_1_a




```
>>> from mpl_toolkits import mplot3d
>>> import numpy as np
>>> from pylab import *
>>> def f(X,Y):
...     return np.cos(X**2+Y**2-0.5)
...
>>> x = np.linspace(-1,1,30)
>>> y = np.linspace(-1,1,30)
>>> X, Y = np.meshgrid(x, y)
>>> Z = f(X, Y)
>>> ax = axes(projection='3d')
>>> ax.plot_surface(X,Y,Z)
<mpl_toolkits.mplot3d.art3d.Poly3DCollection object at 0x7fb1068c4370>
>>> xlabel('x')
Text(0.5, 0, 'x')
>>> ylabel('y')
Text(0.5, 0.5, 'y')
>>> title('$\cos (x^2+y^2-0.5)$')
Text(0.5, 0.92, '$\\cos (x^2+y^2-0.5)$')
>>> legend()
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
<matplotlib.legend.Legend object at 0x7fb1065dfaf0>
>>> show()
```

Q_1_b




```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> x=np.arange(-1,1,0.01)
>>> y=np.arcsin(x)
>>> plt.plot(x,y)
[<matplotlib.lines.Line2D object at 0x7fd240be21a0>]
>>> plt.show()
```

Q_2_a

```
>>> from sympy import *
>>> S=Segment(Point(1,0),Point(2,-1))
>>> S.rotate(pi)
Segment2D(Point2D(-1, 0), Point2D(-2, 1))
```

Q_2_c

```
>>> from sympy import *
>>> t1=Triangle(Point(0,0),Point(4,0),Point(1,4))
>>> t1.is_scalene()
True
```

Q_3_a_i

```
>> from pulp import *
>>> model = LpProblem(name="small-problem",sense=LpMaximize)
>>> x=LpVariable(name="x",lowBound=0)
>>> y=LpVariable(name="y",lowBound=0)
>>> model += (4*x+6*y<=24)
>>> model += (5*x+3*y<=15)
>>> model += 150*x+75*y
>>> model
small-problem:
MAXIMIZE
150*x + 75*y + 0
SUBJECT TO
_C1: 4 x + 6 y <= 24

_C2: 5 x + 3 y <= 15

VARIABLES
x Continuous
y Continuous

>>> model.solve()
Welcome to the CBC MILP Solver
Version: 2.10.7
Build Date: Feb 14 2022

command line - cbc /tmp/02d47fa4a55a4812995fd0cb781a3294-pulp.mps max timeMode elapsed branch printingOptions all solution /tmp/02d47fa4a55a4812995fd0cb781a3294-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 14 RHS
At line 17 BOUNDS
At line 18 ENDATA
Problem MODEL has 2 rows, 2 columns and 4 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
0  Obj -0 Dual inf 225 (2)
0  Obj -0 Dual inf 225 (2)
1  Obj 450
Optimal - objective value 450
Optimal objective 450 - 1 iterations time 0.032
Option for printingOptions changed from normal to all
```

Total time (CPU seconds): 0.07 (Wallclock seconds): 0.02

1
>>> model.objective.value()
450.0
>>> x.value()
3.0
>>> y.value()
0.0
>>>

Q_3 _b_i
>>> from sympy import *
>>> P=Point(-2,4)
i->
>>> P.transform(Matrix([[1,0,0],[0,1,0],[0,0,1]]))
Point2D(-2, 4)
ii>
>> P.scale(6,0)
Point2D(-12, 0)
iii>
>> P.transform(Matrix([[1,4,0],[0,1,0],[0,0,1]]))
Point2D(-2, -4)
iv>
>>> from math import *
>>> angle=radians(30)
>>> P.rotate(angle)
Point2D(-46650635094611/12500000000000, 9856406460551/4000000000000)
>>>