In [1]:

```
from pylab import*
```

In [3]:

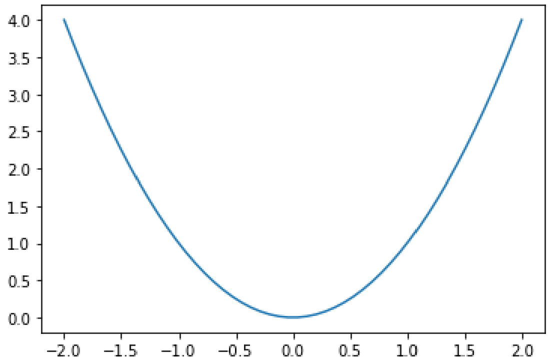
```
import numpy as np
```

In [4]:

```
x=np.linspace(-2,2)
y=x**2
```

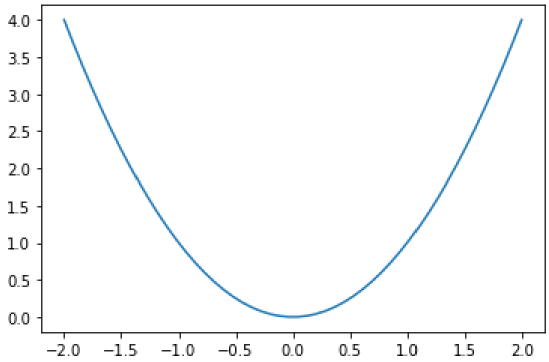
In [5]:

```
plt.plot(x,y)
plt.show()
```



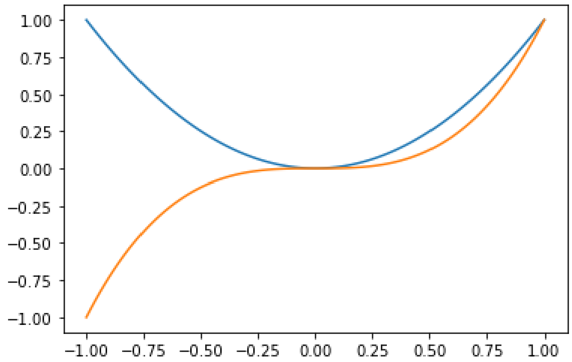
In [6]:

```
x=np.linspace(-2,2)
y=x**2
plt.plot(x,y)
plt.show()
```



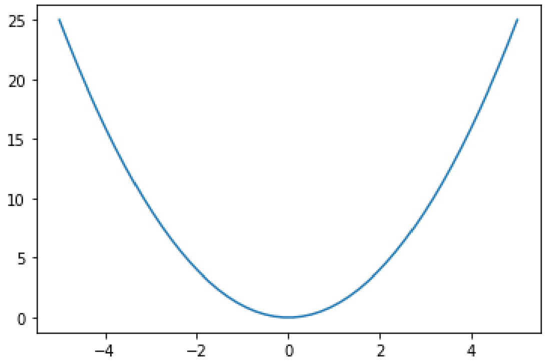
In [7]:

```
x=np.linspace(-1,1,100)
f=x**2
g=x**3
plot(x,f)
plot(x,g)
show()
```



In [8]:

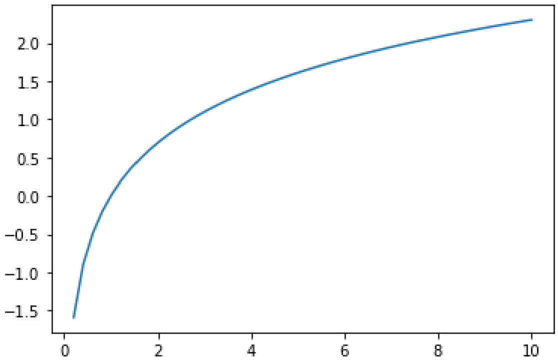
```
x=np.linspace(-5,5)
y=x**2
plt.plot(x,y)
plt.show()
```



In [9]:

```
x=np.linspace(0,10)
y=np.log(x)
plt.plot(x,y)
plt.show()
```

```
/tmp/ipykernel_3760/572656475.py:2: RuntimeWarning: divide by zero encountered in log
  y=np.log(x)
```

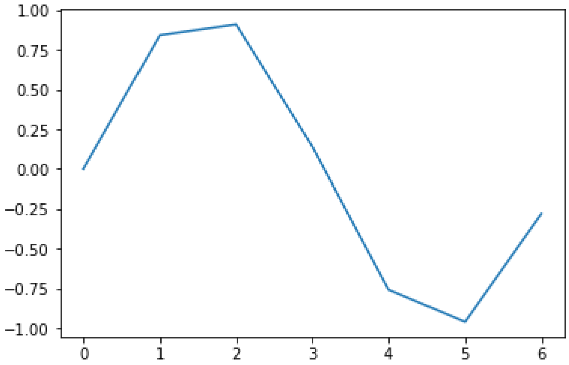


In [ ]:

```
trignometric
```

In [15]:

```
x=np.arange(0,2*(np.pi))
y=np.sin(x)
plt.plot(x,y)
plt.show()
```



In [16]:

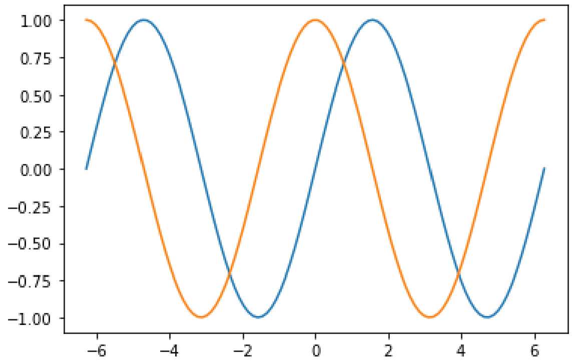
```
from math import*
```

In [ ]:

```

```

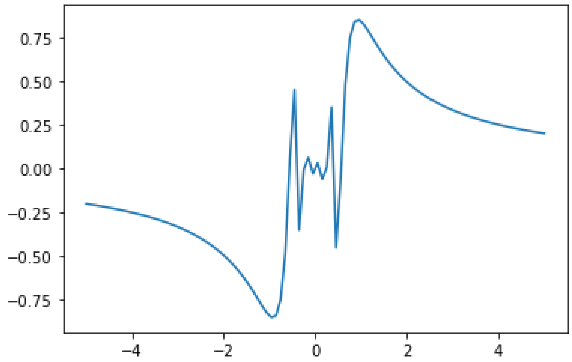
In [17]:

```
x=np.linspace(-2*pi,2*pi,100)
f=np.sin(x)
g=np.cos(x)
plt.plot(x,f)
plt.plot(x,g)
plt.show()
```



In [18]:

```
x=np.linspace(-5,5,100)
f=x*np.sin(1/x**2)
plot(x,f)
show()
```

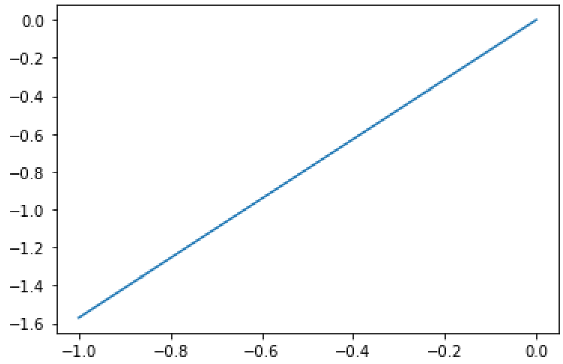


In [ ]:

```
inverse trignometric function
```

In [21]:

```
x=np.arange(-1,1)
f=np.arcsin(x)
plt.plot(x,f)
plt.show()
```

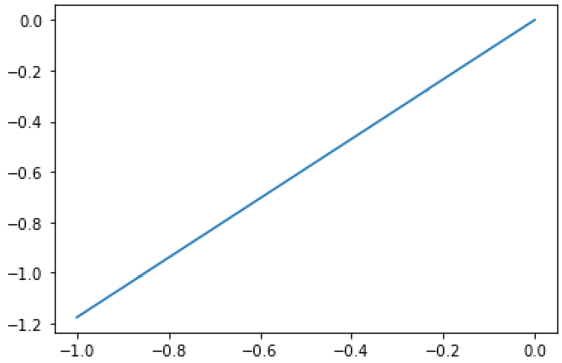


In [ ]:

```
hyperbolic
```

In [22]:

```
x=np.arange(-1,1)
f=np.sinh(x)
plt.plot(x,f)
plt.show()
```

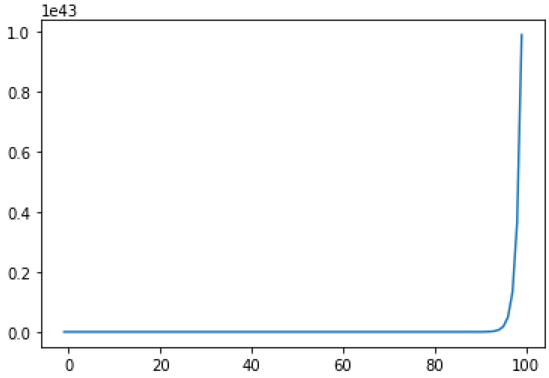


In [ ]:

```
exponential
```

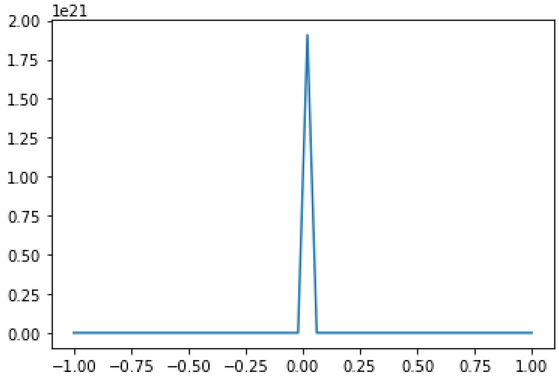
In [23]:

```
x=np.arange(-1,100)
y=np.exp(x)
plt.plot(x,y)
plt.show()
```



In [39]:

```
x=np.linspace(-1,1)
f=np.exp(1/x)
plt.plot(x,f)
plt.show()
```



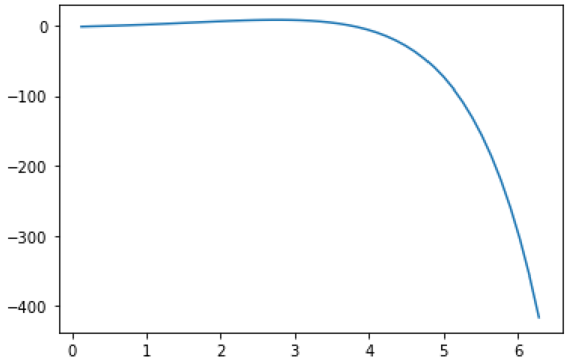
In [ ]:

```
combination LIAT
```

In [27]:

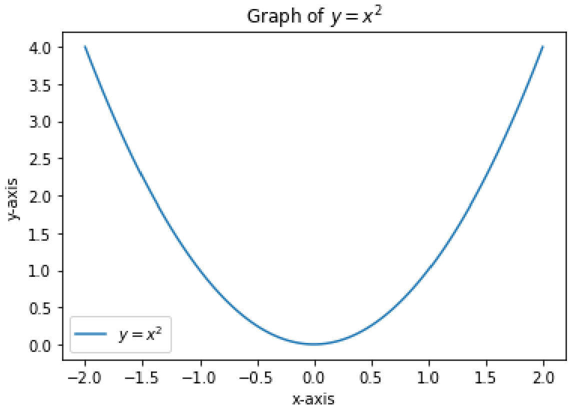
```
x=np.linspace(0,2*(np.pi))
y=np.sin(x)-np.exp(x)+3*x**2+np.log10(x)
plt.plot(x,y)
plt.show()
```

```
/tmp/ipykernel_3760/3520526475.py:2: RuntimeWarning: divide by zero encountered in log10
  y=np.sin(x)-np.exp(x)+3*x**2+np.log10(x)
```



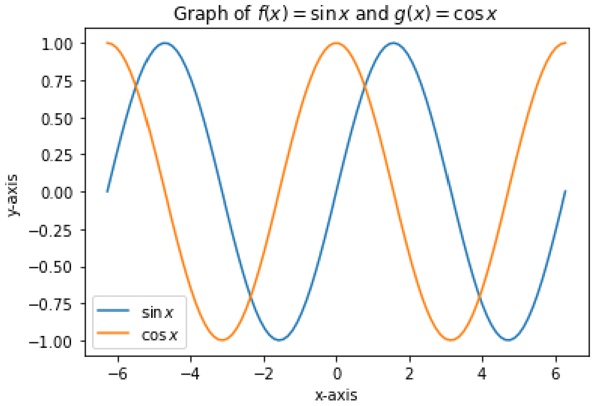
In [32]:

```
x=np.linspace(-2,2,100)
y=x**2
plot(x,y,label="$y=x^2$")
xlabel('x-axis')
ylabel('y-axis')
title('Graph of $y=x^2$')
legend()
show()
```



In [38]:

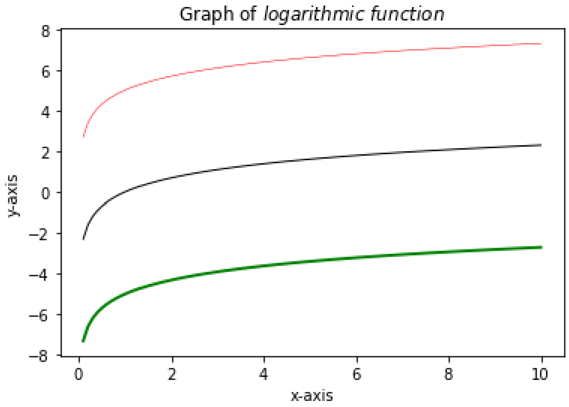
```
x=np.linspace(-2*np.pi,2*np.pi,100)
f=np.sin(x)
g=np.cos(x)
plot(x,f,label="$\sinx$")
plot(x,g,label="$\cosx$")
xlabel('x-axis')
ylabel('y-axis')
title('Graph of $ f(x)=\sinx$ and $g(x)=\cosx$')
legend()
show()
```



In [49]:

```
x=np.linspace(0,10,100)
y1=np.log(x)+5
y2=np.log(x)
y3=np.log(x)-5
plot(x,y1,'r',lw=0.5)
plot(x,y2,'k',lw=1)
plot(x,y3,'g',lw=2)
xlabel('x-axis')
ylabel('y-axis')
title('Graph of $logarithmic$ $function$')
show()
```

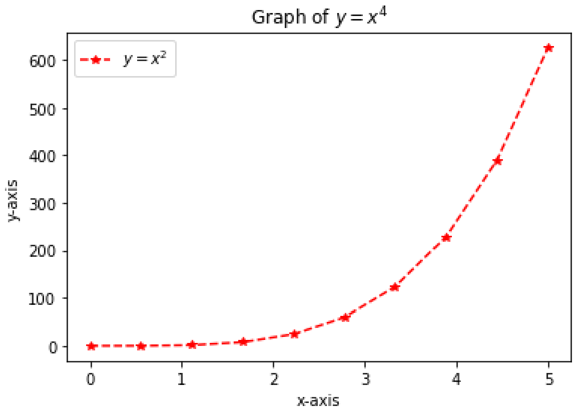
```
/tmp/ipykernel_3760/2921509125.py:2: RuntimeWarning: divide by zero encountered in log
  y1=np.log(x)+5
/tmp/ipykernel_3760/2921509125.py:3: RuntimeWarning: divide by zero encountered in log
  y2=np.log(x)
/tmp/ipykernel_3760/2921509125.py:4: RuntimeWarning: divide by zero encountered in log
  y3=np.log(x)-5
```



In [59]:

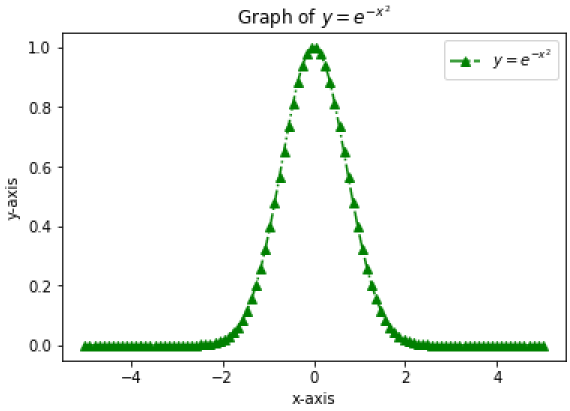
```
x=np.linspace(0,5,10)
y=x**4
plot(x,y,"--or",marker='*',label="$y=x^2$")
xlabel('x-axis')
ylabel('y-axis')
title('Graph of $y=x^4$')
legend()
show()
```

```
/tmp/ipykernel_3760/2990936332.py:3: UserWarning: marker is redundantly defined by the 'marke
r' keyword argument and the fmt string "--or" (-> marker='o'). The keyword argument will take
precedence.
  plot(x,y,"--or",marker='*',label="$y=x^2$")
```



In [60]:

```
x=np.linspace(-5,5,100)
y=np.exp(-x**2)
plot(x,y,"-.^g",label="$y=e^{-x^2}$")
xlabel('x-axis')
ylabel('y-axis')
title('Graph of $y=e^{-x^2}$')
legend()
show()
```

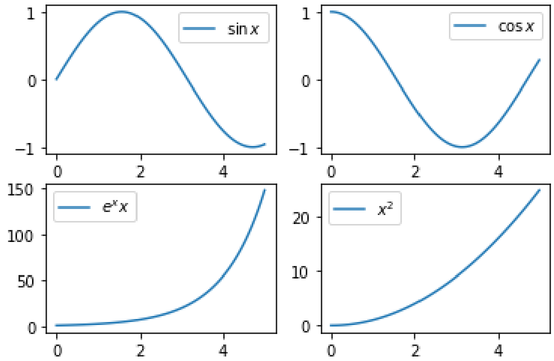


In [ ]:

```
subplot
```

In [65]:

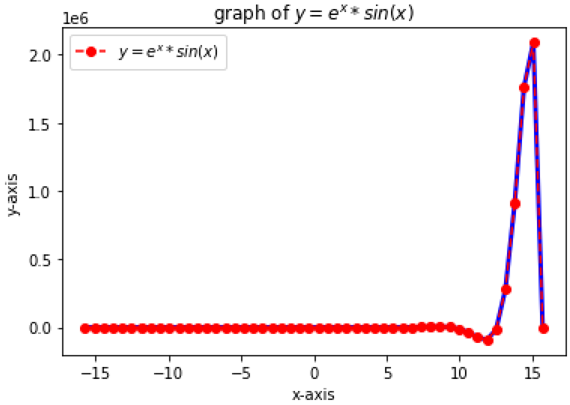
```
x=np.linspace(0,5,100)
y1=np.sin(x)
y2=np.cos(x)
y3=np.exp(x)
y4=x**2
subplot(2,2,1)
plot(x,y1,label="$\sin x$")
legend()
subplot(2,2,2)
plot(x,y2,label="$\cosx$")
legend()
subplot(2,2,3)
plot(x,y3,label="$e^xx$")
legend()
subplot(2,2,4)
plot(x,y4,label="$x^2$")
legend()
show()
```



In [3]:

```
from pylab import*
import numpy as np
x=np.linspace(-5*pi,5*pi)
y=e**x*sin(x)
plot(x,y,'b',lw=3)
plot(x,y,"--or",marker='o',label="$y=e^x*sin(x)$")
xlabel('x-axis')
ylabel('y-axis')
title('graph of $y=e^x*sin(x)$')
legend()
show()
```

```
/tmp/ipykernel_4051/3979224822.py:6: UserWarning: marker is redundantly defined by the 'marke
r' keyword argument and the fmt string "--or" (-> marker='o'). The keyword argument will take
precedence.
  plot(x,y,"--or",marker='o',label="$y=e^x*sin(x)$")
```

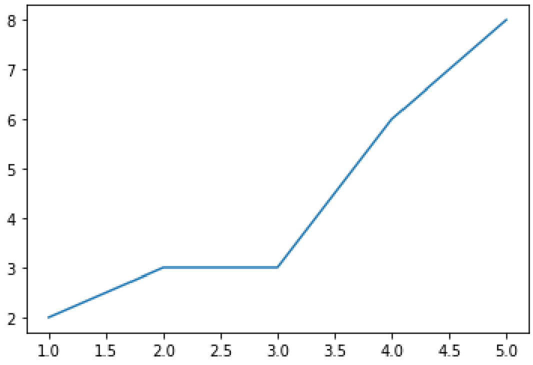


In [ ]:

```
line graph
```

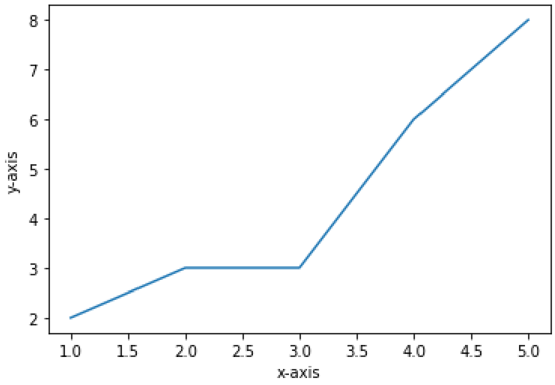
In [5]:

```
import matplotlib.pyplot as plt
x1=[1,2,3,4,5]
y1=[2,3,3,6,8]
plt.plot(x1,y1)
plt.show()
```



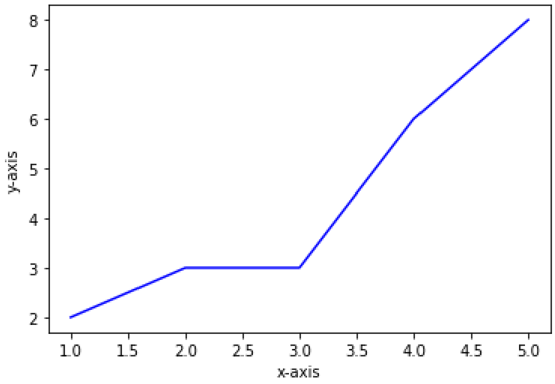
In [7]:

```
import matplotlib.pyplot as plt
x1=[1,2,3,4,5]
y1=[2,3,3,6,8]
plt.plot(x1,y1)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend
plt.show()

```



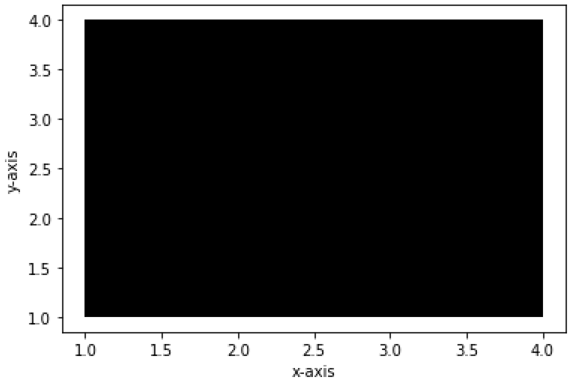
In [9]:

```
import matplotlib.pyplot as plt
x1=[1,2,3,4,5]
y1=[2,3,3,6,8]
plt.plot(x1,y1,'b')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend
plt.show()
```



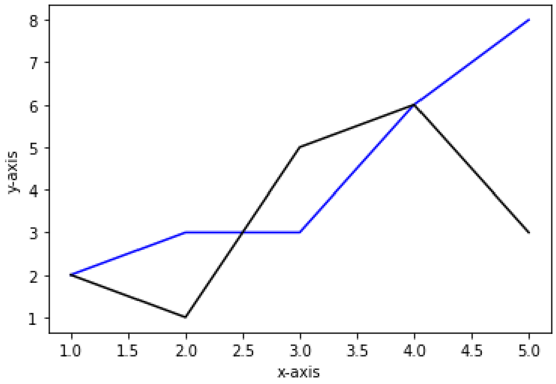
In [14]:

```
import matplotlib.pyplot as plt
x1=[1,1,4,4,1]
y1=[1,4,4,1,1]
plt.fill(x1,y1,'k')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend
plt.show()
```



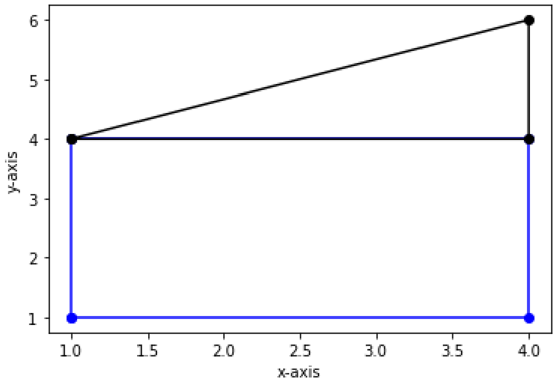
In [16]:

```
import matplotlib.pyplot as plt
x1=[1,2,3,4,5]
y1=[2,3,3,6,8]
plt.plot(x1,y1,'b')
x2=[1,2,3,4,5]
y2=[2,1,5,6,3]
plt.plot(x2,y2,'k')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend
plt.show()
```



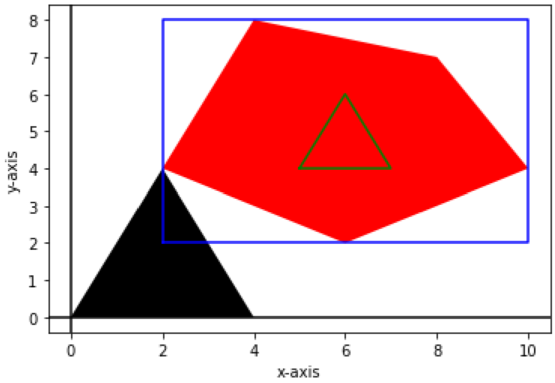
In [22]:

```

x1=[1,1,4,4,1]
y1=[1,4,4,1,1]
plt.plot(x1,y1,'b',marker='o')
x2=[1,4,4,1]
y2=[4,6,4,4]
plt.plot(x2,y2,'k',marker='o')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend
plt.show()
```



In [40]:

```

import matplotlib.pyplot as plt
plt.axhline(y=0,color='k')
plt.axvline(x=0,color='k')
x1=[5,7,6,5]
y1=[4,4,6,4]
plt.plot(x1,y1,'g')
x2=[2,10,10,2,2]
y2=[2,2,8,8,2]
plt.plot(x2,y2,'b')
x3=[6,10,8,4,2,6]
y3=[2,4,7,8,4,2]
plt.fill(x3,y3,'r')
x4=[0,4,2,0]
y4=[0,0,4,0]
plt.fill(x4,y4,'k')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend
plt.show()
```



In [ ]:

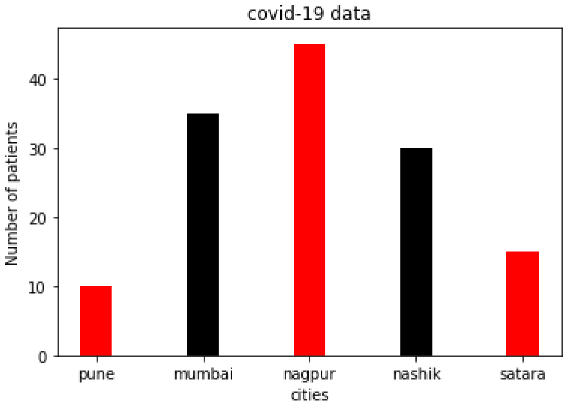
```
bar graph
```

In [41]:

```
import matplotlib.pyplot as plt
```

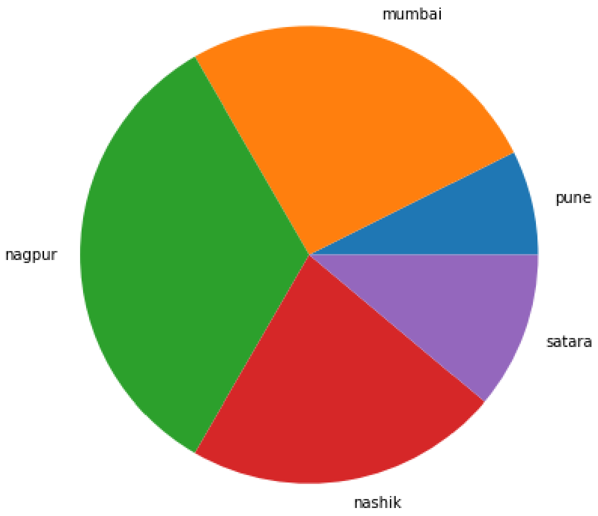
In [48]:

```
left=[1,2,3,4,5]
height=[10,35,45,30,15]
tick_label=['pune','mumbai','nagpur','nashik','satara']
plt.bar(left,height,tick_label=tick_label,width=0.3,color=['red','black'])
plt.xlabel('cityes')
plt.ylabel('Number of patients')
plt.title('covid-19 data')
plt.show()
```



In [58]:

```
left=[1,2,3,4,5]
height=[600,4000,2000,1500,700]
tick_label=['clothing','food','rent','petrol','mis']
plt.bar(left,height,tick_label=tick_label,width=0.3,color=['blue','red',])
plt.xlabel('item')
plt.ylabel('expendenditure in RS.')
plt.title('data')
plt.show()
```



In [ ]:

```

```

In [ ]:

```
pie chart
```

In [ ]:

```
3.practicle
```

In [3]:

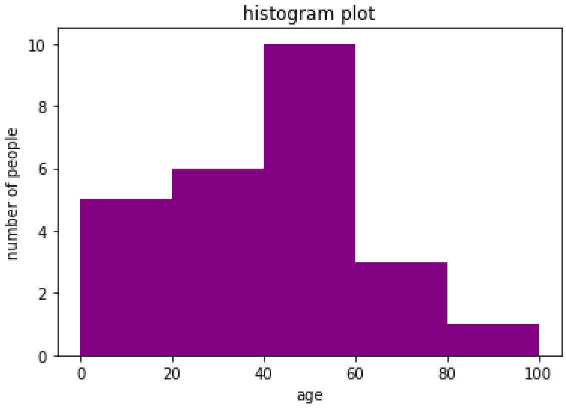
```
import numpy as np
from pylab import*
```

In [ ]:

```
draw histogram for following data
2,5,70,40,30,45,50,45,43,40,44,60,7,13,57,18,90,77,32,21,20,40,45,32,38
```

In [14]:

```
ages=[2,5,70,40,30,45,50,45,43,40,44,60,7,13,57,18,90,77,32,21,20,40,45,32,38]
range=(0,100)
bins=5
plt.hist(ages,bins,range,color='purple',histtype='bar',rwidth=1)
plt.xlabel('age')
plt.ylabel('number of people')
plt.title('histogram plot')
plt.show()
```

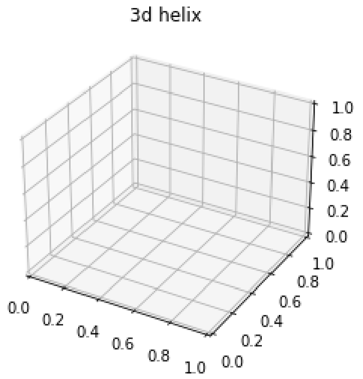


In [21]:

```
from mpl_toolkits import mplot3d
import numpy as np
from pylab import*
fig=plt.figure()
ax=plt.axes(projection='3d')
plt.title('3d helix')
```

Out[21]:

```
Text(0.5, 0.92, '3d helix')
```

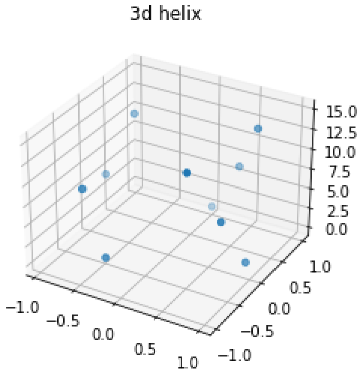


In [24]:

```
ax=plt.axes(projection='3d')
zvalue=np.linspace(0,15,10)
xvalue=np.sin(zvalue)
yvalue=np.cos(zvalue)
ax.scatter3D(xvalue,yvalue,zvalue,'red')
plt.title('3d helix')

```

Out[24]:

```
Text(0.5, 0.92, '3d helix')
```

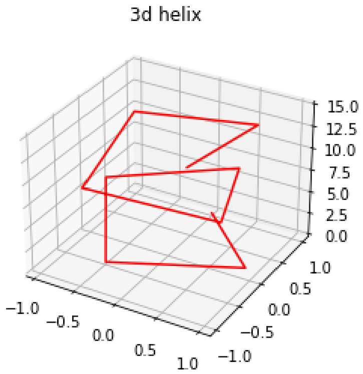


In [25]:

```
ax=plt.axes(projection='3d')
zvalue=np.linspace(0,15,10)
xvalue=np.sin(zvalue)
yvalue=np.cos(zvalue)
ax.plot3D(xvalue,yvalue,zvalue,'red')
plt.title('3d helix')

```

Out[25]:

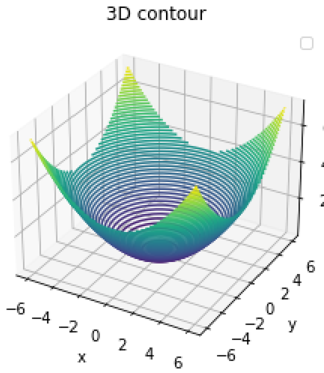
```
Text(0.5, 0.92, '3d helix')
```



In [26]:

```
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=plt.axes(projection='3d')
ax.contour3D(x,y,z,50)
xlabel('x')
ylabel('y')
title('3D contour')
legend()
show()
```

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



In [ ]:

```

```

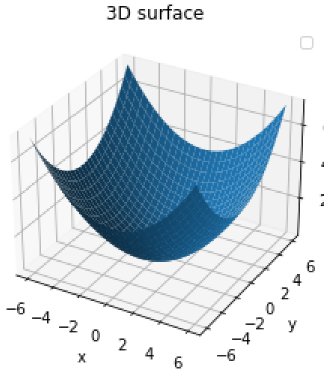
In [ ]:

```
surface and wireframe
```

In [34]:

```
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=plt.axes(projection='3d')
ax.plot_surface(x,y,z)
xlabel('x')
ylabel('y')
title('3D surface')
legend()
show()
```

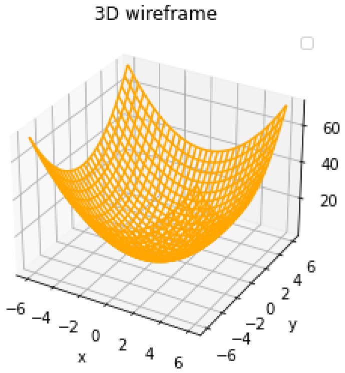
```
No artists with labels found to put in legend.  Note that artists whose label start with an un
derscore are ignored when legend() is called with no argument.
```



In [33]:

```
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=plt.axes(projection='3d')
ax.plot_wireframe(x,y,z,color='orange')
xlabel('x')
ylabel('y')
title('3D wireframe')
legend()
show()
```

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



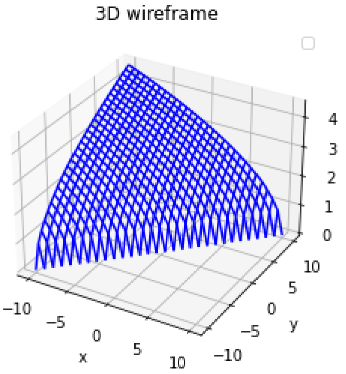
In [ ]:

```
f(x,y)sqrt(y-x)
(e^x)*cos(y)

```

In [37]:

```
def f(x,y):
    return np.sqrt(y-x)
x=np.linspace(-10,10,30)
y=np.linspace(-10,10,30)
x,y=np.meshgrid(x,y)
z=f(x,y)
ax=plt.axes(projection='3d')
ax.plot_wireframe(x,y,z,color='blue')
xlabel('x')
ylabel('y')
title('3D wireframe')
legend()
show()
```

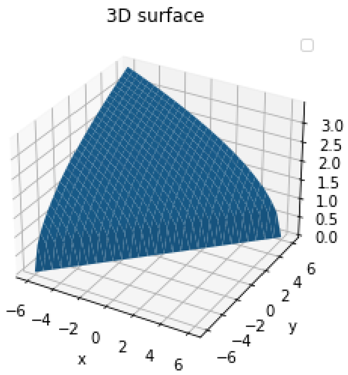
```
/tmp/ipykernel_4014/1299934333.py:2: RuntimeWarning: invalid value encountered in sqrt
  return np.sqrt(y-x)
No artists with labels found to put in legend.  Note that artists whose label start with an un
derscore are ignored when legend() is called with no argument.
```



In [38]:

```
def f(x,y):
    return np.sqrt(y-x)
x=np.linspace(-6,6,30)
y=np.linspace(-6,6,30)
x,y=np.meshgrid(x,y)
z=f(x,y)
ax=plt.axes(projection='3d')
ax.plot_surface(x,y,z)
xlabel('x')
ylabel('y')
title('3D surface')
legend()
show()
```

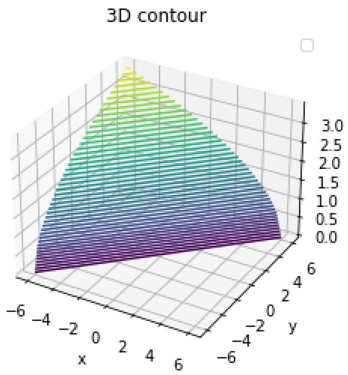
```
/tmp/ipykernel_4014/3915295867.py:2: RuntimeWarning: invalid value encountered in sqrt
  return np.sqrt(y-x)
No artists with labels found to put in legend.  Note that artists whose label start with an un
derscore are ignored when legend() is called with no argument.
```



In [39]:

```
def f(x,y):
    return np.sqrt(y-x)
x=np.linspace(-6,6,30)
y=np.linspace(-6,6,30)
x,y=np.meshgrid(x,y)
z=f(x,y)
ax=plt.axes(projection='3d')
ax.contour3D(x,y,z,50)
xlabel('x')
ylabel('y')
title('3D contour')
legend()
show()
```

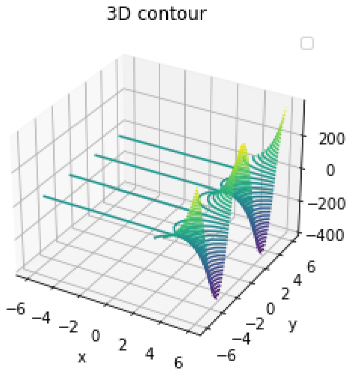
```
/tmp/ipykernel_4014/2678960116.py:2: RuntimeWarning: invalid value encountered in sqrt
  return np.sqrt(y-x)
No artists with labels found to put in legend.  Note that artists whose label start with an un
derscore are ignored when legend() is called with no argument.
```



In [42]:

```
def f(x,y):
    return (np.e**x)*cos(y)
x=np.linspace(-6,6,30)
y=np.linspace(-6,6,30)
x,y=np.meshgrid(x,y)
z=f(x,y)
ax=plt.axes(projection='3d')
ax.contour3D(x,y,z,50)
xlabel('x')
ylabel('y')
title('3D contour')
legend()
show()
```

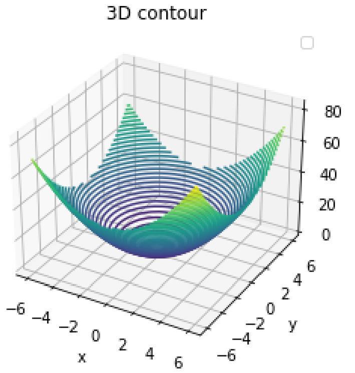
```
No artists with labels found to put in legend.  Note that artists whose label start with an un
derscore are ignored when legend() is called with no argument.
```



In [44]:

```
def f(x,y):
    return (x-y/1+(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=plt.axes(projection='3d')
ax.contour3D(x,y,z,50)
xlabel('x')
ylabel('y')
title('3D contour')
legend()
show()
```

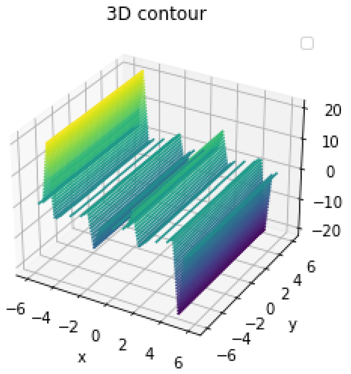
```
No artists with labels found to put in legend.  Note that artists whose label start with an un
derscore are ignored when legend() is called with no argument.
```



In [45]:

```python
def f(x,y):
    return np.tan(x)
x=np.linspace(-6,6,30)
y=np.linspace(-6,6,30)
x,y=np.meshgrid(x,y)
z=f(x,y)
ax=plt.axes(projection='3d')
ax.contour3D(x,y,z,50)
xlabel('x')
ylabel('y')
title('3D contour')
legend()
show()
```

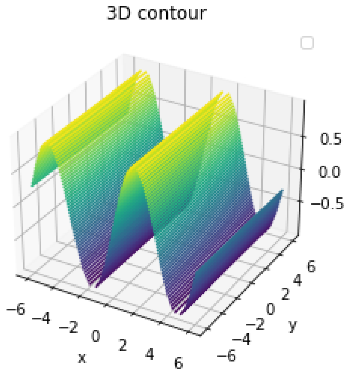
```
No artists with labels found to put in legend.  Note that artists whose label start with an un
derscore are ignored when legend() is called with no argument.
```



In [46]:

```python
def f(x,y):
    return np.sin(x)
x=np.linspace(-6,6,30)
y=np.linspace(-6,6,30)
x,y=np.meshgrid(x,y)
z=f(x,y)
ax=plt.axes(projection='3d')
ax.contour3D(x,y,z,50)
xlabel('x')
ylabel('y')
title('3D contour')
legend()
show()
```

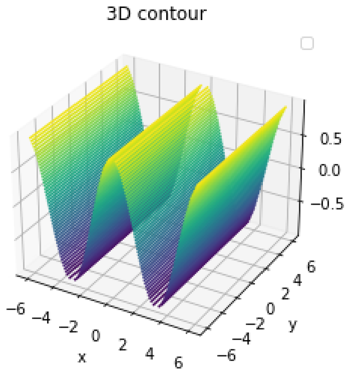
```
No artists with labels found to put in legend.  Note that artists whose label start with an un
derscore are ignored when legend() is called with no argument.
```



In [47]:

```python
def f(x,y):
    return np.cos(x)
x=np.linspace(-6,6,30)
y=np.linspace(-6,6,30)
x,y=np.meshgrid(x,y)
z=f(x,y)
ax=plt.axes(projection='3d')
ax.contour3D(x,y,z,50)
xlabel('x')
ylabel('y')
title('3D contour')
legend()
show()
```

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



In [ ]:

```python
4.practicle
```

In [1]:

```python
from pulp import*
```

In [9]:

```python
model=LpProblem(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
```

In [10]:

```python
model
```

Out[10]:

```
NoName:
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
```

In [11]:

```
model.solve()
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /home/mcs229/anaconda3/lib/python3.9/site-packages/pulp/solverdir/cbc/linux/64/
cbc /tmp/9443997a1dbb4b249fbbc22c046867b6-pulp.mps max timeMode elapsed branch printingOptions
all solution /tmp/9443997a1dbb4b249fbbc22c046867b6-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):       0.01
```

Out[11]:

```
1
```

In [12]:

```
model.objective.value()
```

Out[12]:

```
450.0
```

In [13]:

```
x.value()
```

Out[13]:

```
3.0
```

In [14]:

```
y.value()
```

Out[14]:

```
0.0
```

In [33]:

```
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
```

In [34]:

```
model
```

Out[34]:

```
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
```

In [35]:
```
model.solve()
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /home/mcs229/anaconda3/lib/python3.9/site-packages/pulp/solverdir/cbc/linux/64/
cbc /tmp/229ed7d3bfa0411cb373b58c90bdf295-pulp.mps timeMode elapsed branch printingOptions all
solution /tmp/229ed7d3bfa0411cb373b58c90bdf295-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):       0.00
```

Out[35]:
```
1
```

In [36]:
```
model.objective.value()
```

Out[36]:
```
16.0
```

In [37]:
```
x.value()
```

Out[37]:
```
4.0
```

In [39]:
```
y.value()
```

Out[39]:
```
1.0
```

In [45]:
```
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
```

In [46]:
```
model
```

Out[46]:
```
NoName:
MINIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x >= 6

_C2: y >= 6

_C3: x + y <= 11

VARIABLES
x Continuous
y Continuous
```

In [47]:
```
model.solve()
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /home/mcs229/anaconda3/lib/python3.9/site-packages/pulp/solverdir/cbc/linux/64/
cbc /tmp/7b559323363d4d30806545b72815fd95-pulp.mps timeMode elapsed branch printingOptions all
solution /tmp/7b559323363d4d30806545b72815fd95-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):   0.00

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

Out[47]: -1

model.objective.value()

In [44]:
```
model.objective.value()
```

Out[44]: 12.0

In [48]:
```
model=LpProblem(sense=LpMaximize)
x=LpVariable(name="x",lowBound=0)
y=LpVariable(name="y",lowBound=0)
model+=(x>=1)
model+=(y>=2)
model+=x+y
```

In [49]:
```
model
```

Out[49]:
```
NoName:
MAXIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x >= 1

_C2: y >= 2

VARIABLES
x Continuous
y Continuous
```

In [50]:
```
model.solve()
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /home/mcs229/anaconda3/lib/python3.9/site-packages/pulp/solverdir/cbc/linux/64/
cbc /tmp/1e332a122eca4b5a9873431ab8333175-pulp.mps max timeMode elapsed branch printingOptions
all solution /tmp/1e332a122eca4b5a9873431ab8333175-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 7 COLUMNS
At line 12 RHS
At line 15 BOUNDS
At line 16 ENDATA
Problem MODEL has 2 rows, 2 columns and 2 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)
Dual infeasible - objective value 2e+10
DualInfeasible objective 2e+10 - 0 iterations time 0.002

Result - Linear relaxation unbounded

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

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

Out[50]: -2

In [ ]:
```
6 practical
```

In [1]:
```
from sympy import*
```

In [2]:
```
x=Point(0,0)
```

In [3]:
```
x
```

Out[3]: Point2D(0, 0)

In [74]:
```
y=Point(2,2)

```

In [6]:
```
y


```

Out[6]: Point2D(2, 2)

In [75]:
```
z=Point(-1,-1)


```

In [76]:
```

w=Point(3,4)

```

In [77]:
```
z
```

Out[77]: Point2D(−1, −1)

In [84]:
```
w
```

Out[84]: Point2D(3, 4)

In [85]:
```
Point.is_collinear(x,y)
```

Out[85]: True

In [80]:
```
Point.is_collinear(x,y,z)
```

```
/home/mcs229/anaconda3/lib/python3.9/site-packages/sympy/geometry/point.py:312: UserWarning: D
imension of (-1, -1) needs to be changed from 2 to 3.
  return [Point(i, **kwargs) for i in points]
```

Out[80]:
```
True
```

In [86]:
```
Point.is_collinear(x,y,z,w)
```

```
/home/mcs229/anaconda3/lib/python3.9/site-packages/sympy/geometry/point.py:312: UserWarning: D
imension of (3, 4) needs to be changed from 2 to 3.
  return [Point(i, **kwargs) for i in points]
```

Out[86]:
```
False
```

In [23]:
```
Point.are_coplanar(x,y,z,w)
```

Out[23]:
```
True
```

In [58]:
```
p=Point(0,0,0)

```

In [59]:
```
q=Point(2,2,2)

```

In [60]:
```

r=Point(-1,-1,-1)

```

In [61]:
```

s=Point(3,4,-7)
```

In [62]:
```
p
```

Out[62]: $\mathrm{Point3D}(0, 0, 0)$

In [63]:
```
q
```

Out[63]: $\mathrm{Point3D}(2, 2, 2)$

In [64]:
```
r
```

Out[64]: $\mathrm{Point3D}(-1, -1, -1)$

In [65]:
```
s
```

Out[65]: $\mathrm{Point3D}(3, 4, -7)$

In [66]:
```
Point.is_collinear(p,q,r,s)
```

Out[66]:
```
False
```

In [67]:
```
Point.is_collinear(p,q)
```

Out[67]:
```
True
```

In [68]:
```
Point.is_collinear(p,q,r)
```

Out[68]:
```
True
```

In [69]:
```
Point.is_collinear(p,q,r,s)
```

Out[69]:
```
False
```

In [70]:
```
p.distance(q)
```

Out[70]: $2\sqrt{3}$

In [71]:
```
x.distance(y)
```

Out[71]: $2\sqrt{3}$

In [82]:
```
x.scale(2,2)
```
Out[82]: Point3D(0, 0, 0)

In [81]:
```
y.scale(3,1)
```
Out[81]: Point2D(6, 2)

In [83]:
```
w.scale(1,4)
```
Out[83]: Point2D(3, 16)

In [ ]:
```
reflection:-
```

In [88]:
```
x=Point(3,4)
```

In [90]:
```
x.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
```
Out[90]: Point2D(−3, 4)

In [91]:
```
x=Point(3,-4)
x.transform(Matrix([[1,0,0],[3,1,0],[0,0,1]]))
```
Out[91]: Point2D(−4, 0)

In [96]:
```
x=Point(1,4)
x.transform(Matrix([[1,0,0],[0,-1,0],[0,0,1]]))
```
Out[96]: Point2D(1, −4)

In [97]:
```
y=Point(-4,3)
y.transform(Matrix([[1,0,0],[0,-1,0],[0,0,1]]))
```
Out[97]: Point2D(−4, −3)

In [98]:
```
x=Point(2.3,4)
x.transform(Matrix([[-1,0,0],[0,-1,0],[0,0,1]]))
```
Out[98]: $\text{Point2D}\left(-\frac{23}{10}, -4\right)$

In [99]:
```
y=Point(3,3)
y.transform(Matrix([[-1,0,0],[0,-1,0],[0,0,1]]))
```
Out[99]: Point2D(−3, −3)

In [ ]:
```
y=x
```

In [101]:
```
x=Point(5,4)
x.transform(Matrix([[0,1,0],[1,0,0],[0,0,1]]))
```
Out[101]: Point2D(4, 5)

In [102]:
```
y=Point(3,-2)
y.transform(Matrix([[0,1,0],[1,0,0],[0,0,1]]))
```
Out[102]: Point2D(−2, 3)

In [ ]:
```
y=-x
```

In [103]:
```
x=Point(5,4)
x.transform(Matrix([[0,-1,0],[-1,0,0],[0,0,1]]))
```
Out[103]: Point2D(−4, −5)

In [105]:
```
y=Point(3,-2)
y.transform(Matrix([[0,-1,0],[-1,0,0],[0,0,-1]]))
```
Out[105]: Point2D(2, −3)

In [ ]:

```
reflection through line
```

In [107]:

```
x,y=symbols('x,y')
a=Point(3,6)
a.reflect(Line(x+y))
```

Out[107]: $\mathrm{Point2D}(-6, -3)$

In [109]:

```
x,y=symbols('x,y')
a=Point(2,6)
a.reflect(Line(2*x+y+1))
```

Out[109]: $\mathrm{Point2D}\left(-\frac{34}{5}, \frac{8}{5}\right)$

In [110]:

```
x,y=symbols('x,y')
a=Point(0,-2)
a.reflect(Line(x+y-5))
```

Out[110]: $\mathrm{Point2D}(7, 5)$

In [ ]:

```
shearing
```

In [ ]:

```
in x direction
```

In [111]:

```
x=Point(3,-4)
x.transform(Matrix([[1,0,0],[3,1,0],[0,0,1]]))
```

Out[111]: $\mathrm{Point2D}(-9, -4)$

In [ ]:

```
y direction
```

In [113]:

```
y=Point(3,-1)
y.transform(Matrix([[1,7,0],[0,1,0],[0,0,1]]))
```

Out[113]: $\mathrm{Point2D}(3, 20)$

In [ ]:

```
both direction
```

In [114]:

```
x=Point(1,4)
x.transform(Matrix([[1,0,0],[-3,1,0],[0,0,1]]))
```

Out[114]: $\mathrm{Point2D}(-11, 4)$

In [ ]:

```
**rotation
```

In [115]:

```
x=Point(1,4)
x.rotate(pi/2)
```

Out[115]: $\mathrm{Point2D}(-4, 1)$

In [116]:

```
y=Point(-4,7)
y.rotate(pi/4)
```

Out[116]: $\mathrm{Point2D}\left(-\frac{11\sqrt{2}}{2}, \frac{3\sqrt{2}}{2}\right)$

In [119]:

```
from math import*
```

In [120]:

```
z=Point(-2,5)
angle=radians(75)
z.rotate(angle)
```

Out[120]: $\mathrm{Point2D}\left(-\frac{267363361082519}{50000000000000}, -\frac{637756427065533}{1000000000000000}\right)$

In [ ]:

```
**linees
```

In [121]:

```
l=Line(Point(2,3),Point(4,1))
```

In [122]:

```
l
```

Out[122]: Line2D(Point2D(2, 3), Point2D(4, 1))

In [125]:

```
x,y=symbols('x,y')
l=Line(2*x+3*y-4)
l
```

Out[125]: $\mathrm{Line2D}\left(\mathrm{Point2D}\left(0, \frac{4}{3}\right), \mathrm{Point2D}\left(1, \frac{2}{3}\right)\right)$

In [ ]:

```
line segment
```

In [126]:

```
s=Segment((0,0),(0,1))
s
```

Out[126]: Segment2D(Point2D(0, 0), Point2D(0, 1))

In [ ]:

```
ray
```

In [127]:

```
r=Ray((0,0),(3,1))
r
```

Out[127]: Ray2D(Point2D(0, 0), Point2D(3, 1))

In [ ]:

```
ex1.

```

In [130]:

```
y=Point(4,3)
y.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
```

Out[130]: Point2D(−4, 3)

In [132]:

```
x=Point(4,3)
x.transform(Matrix([[3,0,0],[0,1,0],[0,0,1]]))
```

Out[132]: Point2D(12, 3)

In [133]:

```
y=Point(4,3)
y.transform(Matrix([[1,0,0],[0,3.2,0],[0,0,1]]))
```

Out[133]: $\mathrm{Point2D}\left(4, \frac{48}{5}\right)$

In [134]:

```
x=Point(4,3)
x.transform(Matrix([[0,-1,0],[-1,0,0],[0,0,1]]))
```

Out[134]: Point2D(−3, −4)

In [135]:

```
y=Point(4,3)
y.transform(Matrix([[1,3,0],[0,1,0],[0,0,1]]))
```

Out[135]: Point2D(4, 15)

In [142]:

```
x=Point(4,3)
x.transform(Matrix([[1,2,0],[3/2,1,0],[0,0,1]]))
```

Out[142]: Point2D(6, 6)

In [144]:

```
x=Point(4,3)
x.transform(Matrix([[1,1,0],[-3,1,0],[0,0,1]]))
```

Out[144]: Point2D(−5, 7)

In [ ]:

```
practical 7
```

In [3]:

```
from sympy import*
```

In [4]:

```
p=Point(0,1)
q=Point(2,3)
l=Line(p,q)
l
```

Out[4]: Line2D(Point2D(0, 1), Point2D(2, 3))

In [5]:

```
p=Point(0,1)
q=Point(2,3)
l=Segment(p,q)
l
```

Out[5]: Segment2D(Point2D(0, 1), Point2D(2, 3))

In [6]:

```
p=Point(0,1)
q=Point(2,3)
l=Ray(p,q)
l
```

Out[6]: Ray2D(Point2D(0, 1), Point2D(2, 3))

In [36]:

```
l1=Line((0,1),(2,3))
l1
```

Out[36]: Line2D(Point2D(0, 1), Point2D(2, 3))

In [18]:

```
l2=Line((2,3),(1,3))
l2
```

Out[18]: Line2D(Point2D(2, 3), Point2D(1, 3))

In [19]:

```
l1.angle_between(l2)
```

Out[19]: $\frac{3\pi}{4}$

In [20]:

```
l1.intersection(l2)
```

Out[20]:

```
[Point2D(2, 3)]
```

In [21]:

```
l1.points
```

Out[21]:

```
(Point2D(0, 1), Point2D(2, 3))
```

In [22]:

```
l1.rotate(pi)
```

Out[22]: Line2D(Point2D(0, −1), Point2D(−2, −3))

In [23]:

```
l2.rotate(pi)
```

Out[23]: Line2D(Point2D(−2, −3), Point2D(−1, −3))

In [24]:

```
l1.length
```

Out[24]: ∞

In [25]:

```
l1.slope
```

Out[25]: 1

In [32]:

```
l3=Segment((0,1),(2,3))
l3
```

Out[32]: Segment2D(Point2D(0, 1), Point2D(2, 3))

In [33]:

```
l3.midpoint
```

Out[33]: Point2D(1, 2)

In [37]:

```
l1.equation()
```

Out[37]: $-2x + 2y - 2$

In [38]:
```
l1.coefficients
```
Out[38]:
```
(-2, 2, -2)
```

In [ ]:
```
eqn of transform line using transform matrix
```

In [39]:
```
A=Point(4,9)
B=Point(-2,1)
A1=A.transform(Matrix([[2,0,0],(0,2,0),(0,0,1)]))
B1=B.transform(Matrix([[2,0,0],(0,2,0),(0,0,1)]))
l=Line(A1,B1)
l.equation()
```
Out[39]: $16x - 12y + 88$

In [ ]:
```
practicle 8
```

In [5]:
```
from sympy import*
```

In [6]:
```
p1,p2,p3,p4,p5=[(0,0),(1,0),(5,1),(0,1),(3,0)]
p=Polygon(p1,p2,p3,p4,p5)
p
```
Out[6]: Polygon(Point2D(1, 0), Point2D(5, 1), Point2D(0, 1), Point2D(3, 0))

In [8]:
```
q=Polygon((0,0),1,n=5)
q
```
Out[8]: RegularPolygon(Point2D(0, 0), 1, 5, 0)

In [9]:
```
p.area
```
Out[9]:

$$\frac{\frac{25}{2} - \frac{5\sqrt{5}}{2}}{4\sqrt{5 - 2\sqrt{5}}}$$

In [14]:
```
p1,p2,p3,p4=map(Point,[(0,0),(1,0),(5,1),(0,1)])
p=Polygon(p1,p2,p3,p4)
p.angles[p1]
```
Out[14]: $\frac{\pi}{2}$

In [15]:
```
p.angles[p2]
```
Out[15]:

$$\operatorname{acos}\left(-\frac{4\sqrt{17}}{17}\right)$$

In [17]:
```
p.angles[p3]
```
Out[17]:

$$\operatorname{acos}\left(\frac{4\sqrt{17}}{17}\right)$$

In [18]:
```
p.angles[p4]
```
Out[18]: $\frac{\pi}{2}$

In [ ]:
```
convexpoly
```

In [19]:
```
p.is_convex()
```
Out[19]:
```
True
```

In [20]:
```
s=Polygon((0,0),(5,5),(0,10),(20,0))
s.is_convex()
```
Out[20]:
```
False
```

In [21]:
```
p.perimeter
```

Out[21]: $\sqrt{17} + 7$

In [23]:
```
s.perimeter
```

Out[23]: $10\sqrt{2} + 20 + 10\sqrt{5}$

In [ ]:
```
center of
```

In [24]:
```
p=RegularPolygon(Point(0,0),5,3)
p.center
```

Out[24]: $\text{Point2D}(0, 0)$

In [ ]:
```
reflection
```

In [28]:
```
p=RegularPolygon(Point(0,0),5,3)
x,y=symbols('x,y')
l=Line(x-y)
p.reflect(l)
```

Out[28]: $\text{RegularPolygon}\left(\text{Point2D}(0, 0), -5, 3, \frac{\pi}{2}\right)$

In [31]:
```
from math import*
```

In [ ]:
```
rotation
```

In [33]:
```
p=RegularPolygon(Point(0,0),5,3)
p.rotate(pi/4)
```

Out[33]: $\text{RegularPolygon}(\text{Point2D}(0, 0), 5, 3, 0.785398163397448)$

In [37]:
```
p=RegularPolygon(Point(0,0),5,3)
angle=radians(270)
p.rotate(angle)
```

Out[37]: $\text{RegularPolygon}\left(\text{Point2D}(0, 0), 5, 3, 4.71238898038469 - \frac{4\pi}{3}\right)$

In [ ]:
```
scaling
```

In [38]:
```
p=RegularPolygon(Point(0,0),1,4)
p.scale(2,2)
```

Out[38]: $\text{RegularPolygon}(\text{Point2D}(0, 0), 2, 4, 0)$

In [39]:
```
p.scale(2,3)
```

Out[39]: $\text{Polygon}(\text{Point2D}(2, 0), \text{Point2D}(0, 3), \text{Point2D}(-2, 0), \text{Point2D}(0, -3))$

In [ ]:
```
triangles
```

In [41]:
```
p1,p2,p3=[(0,0),(1,0),(5,1)]
t=Polygon(p1,p2,p3)
t
```

Out[41]: $\text{Triangle}(\text{Point2D}(0, 0), \text{Point2D}(1, 0), \text{Point2D}(5, 1))$

In [42]:
```
t=Triangle(p1,p2,p3)
t
```

Out[42]: $\text{Triangle}(\text{Point2D}(0, 0), \text{Point2D}(1, 0), \text{Point2D}(5, 1))$

In [43]:
```
Triangle(sss=(3,4,5))
```

Out[43]: $\text{Triangle}(\text{Point2D}(0, 0), \text{Point2D}(3, 0), \text{Point2D}(3, 4))$

In [48]:
```
Triangle(asa=(30,1,30))
```

Out[48]:

$$\mathrm{Triangle}\left(\mathrm{Point2D}(0,0), \mathrm{Point2D}(1,0), \mathrm{Point2D}\left(\frac{1}{2}, \frac{\sqrt{3}}{6}\right)\right)$$

In [46]:
```
Triangle(sas=(1,45,2))
```

Out[46]:

$$\mathrm{Triangle}\left(\mathrm{Point2D}(0,0), \mathrm{Point2D}(2,0), \mathrm{Point2D}\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)\right)$$

In [ ]:
```
isosceles
```

In [50]:
```
t=Triangle(Point(0,0),Point(4,0),Point(2,4))
t.is_isosceles()
```

Out[50]:
```
True
```

In [51]:
```
t=Triangle(Point(0,0),Point(4,0),Point(4,3))
t.is_right()
```

Out[51]:
```
True
```

In [52]:
```
t=Triangle(Point(0,0),Point(4,0),Point(1,4))
t.is_scalene()
```

Out[52]:
```
True
```

In [ ]:
```

```