Name :- Harshal Ravindra Naik

Roll No :- 33

Div :- A

Incharge :- Dr. Vikas Jadhav Sir

Python Practical (Sem 6)

Practical 1: Turtle Graphics (Unit 1)

1. Initialize the Turtle canvas and draw the following commands: ¶

(a) Move forward 150 pixel.

(b) Rotate clockwise 45 degree.

(c) Move forward 180 pixel without drawing.

(d) Draw circle with radius 200 pixel with green color.

(e) Goto home.

(f) Draw a circle in clockwise direction with fill color blue with radius

40 pixel.

(g) Print the current position of turtle.

(h) Print the current direction of turtle.

(i) clear all screen.

In [1]:
```
from turtle import*
t=Turtle()
t.fd(150)
```

In [2]:
```
t.fd(80)
```

In [3]:
```
t.rt(-45)
```

In [4]:
```
t.fd(180)
```

In [5]:
```
t.circle(200)
```

In [6]:
```
t.home()
```

In [7]:
```
t.begin_fill()
t.fillcolor("blue")
t.circle(-40)
t.end_fill
```

Out[7]:
```
<bound method RawTurtle.end_fill of <turtle.Turtle object at 0x000001A853C615E0
>>
```

In [8]:
```
t.position()
```

Out[8]:
```
(-0.00,0.00)
```

In [9]:
```
t.clear()
```

2. Write a python programme to define a function, which plot a circle of a given radius starting with its current position.

In [11]:
```
from turtle import*
def circle(radius):
    t=Turtle()
    t.circle(radius)
```

3. Draw a letter “W” usin turtle methods with width 3.

In [12]:
```
from turtle import*
t=Turtle()
t.width(3)
t.right(90)
t.fd(150)
t.right(-120)
t.fd(90)
t.right(60)
t.fd(90)
t.left(120)
t.fd(150)
```

4. Draw a circle radius 250 pixel centered at (150,150).

In [13]:
```
from turtle import*
t=Turtle()
t.goto(150,150)
t.circle(250)
```

5. Using random function, draw a random walk with 60 steps which travels forward between 20 and 100 and moves right between 0 and 360randomly.

In [ ]:
```
from turtle import*
import random
count = 0
while count <61:
    count+= 1
    if (turtle.xcor() >-300 and turtle.xcor() <300 and\(turtle.ycor() >-300 and t

            t.fd(random.randit(20,100))
            t.right(random.randit(0,360))
    else:
        t.right(180)
        t.fd(300)
```

6. Draw a circles of different radius with random colors

In [15]:
```
from turtle import*
import random
t=Turtle()
t.pencolor("blue")
t.circle(50)
t.pencolor("red")
t.circle(100)
t.pencolor("yellow")
t.circle(150)
t.pencolor("green")
t.circle(200)
```

7. Draw a random walk with random color having 100 steps using random function.

In [16]:
```
from turtle import*
import random
t=Turtle()
color=["Black","Red","Yellow","grey","blue"]
for i in range(100):
    t.fd(40)
    t.right(random.randint(20,350))
    t.pencolor(color[i%2])
```

8. Use “for” loops to make a turtle draw these regular polygons (regular means all sides the same lengths, all angles the same):

(a) An equilateral triangle

(b) A square

(c) A hexagon (six sides)

(d) An octagon (eight sides)

In [17]:
```
from turtle import*
t = Turtle()
for i in [3,4,6,8]:
    t.circle(200,360,i)
```

Practical 2

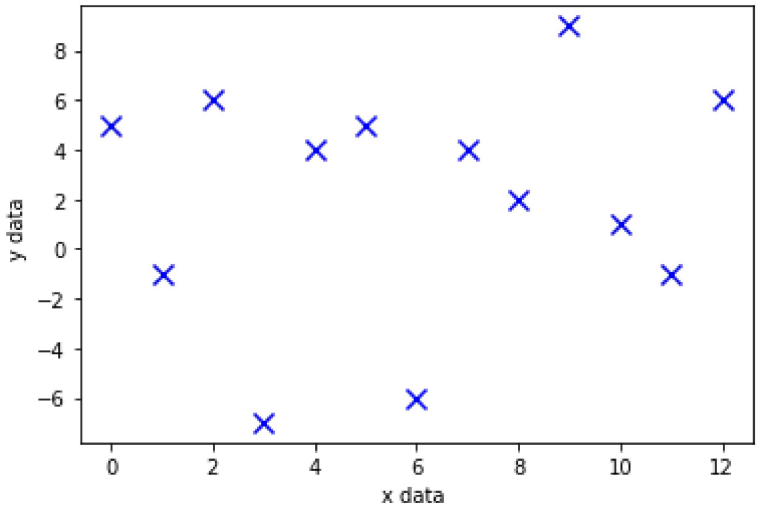
In [21]:
```
# Q1
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
y = [5,-1 , 6, -7, 4, 5, -6, 4,2, 9, 1, -1, 6]

plt.scatter(x , y , c='blue' , marker ='x' , s=100)

plt.xlabel('x data ')
plt.ylabel('y data ')
plt.show
```

Out[21]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



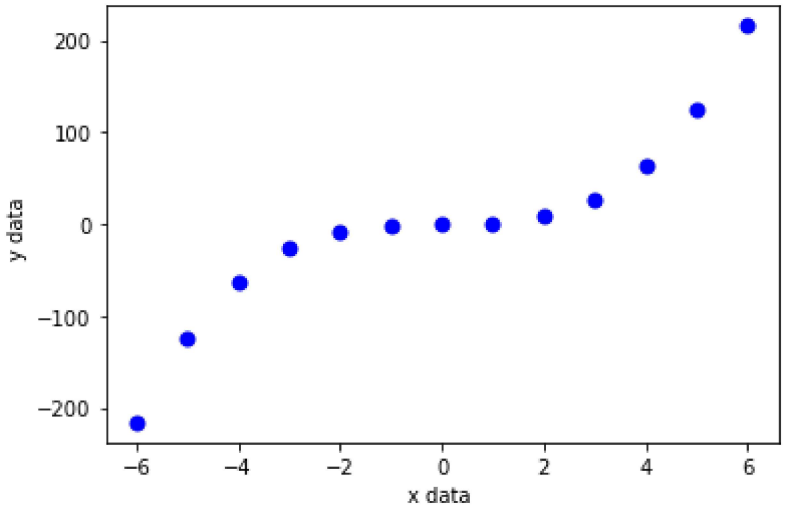
In [22]:
```
# Q2
import matplotlib.pyplot as plt

x =  [-6 , -5 , -4 , -3 ,-2 , -1, 0, 1, 2, 3, 4, 5, 6]
y = [i**3 for i in x]

plt.scatter(x , y , c='blue' , linewidth=2)

plt.xlabel('x data ')
plt.ylabel('y data ')
plt.show
```

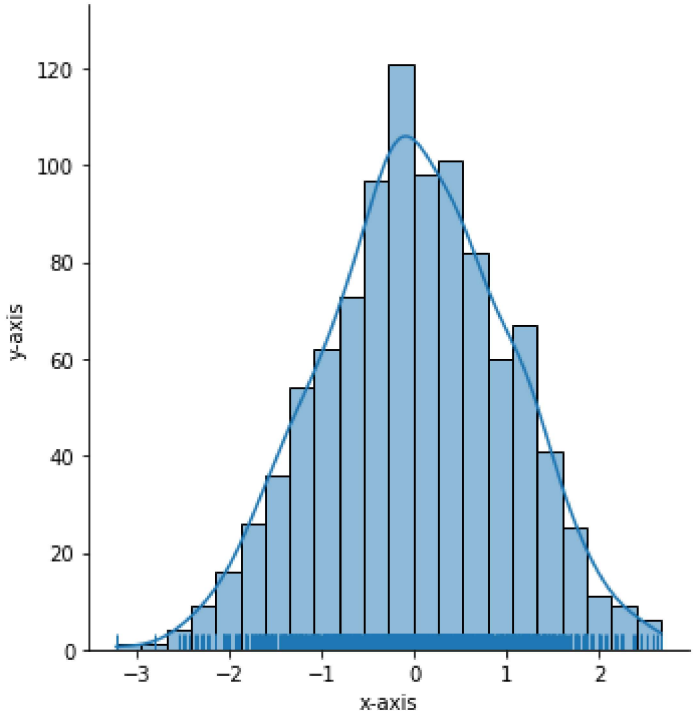
Out[22]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [ ]:
```
# Q3
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df , x="sepal_width", y="sepal_length",color="species",
                 size='petal_lenght' , hover_data=['petal_width'])
fig.show()
```

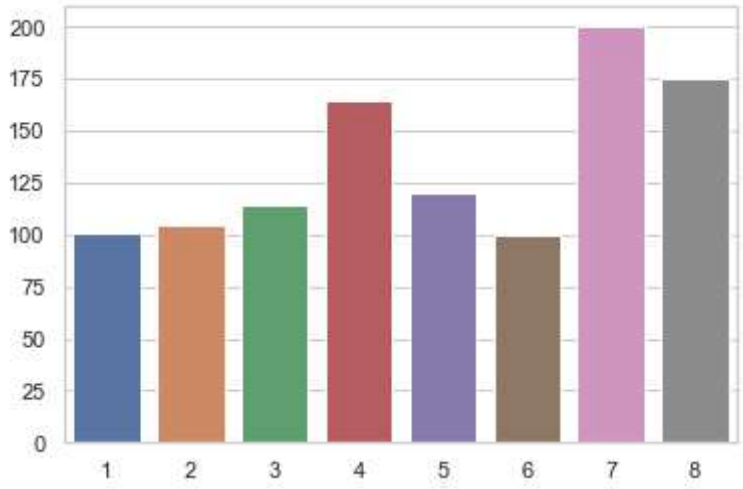
In [24]:
```
#Q4
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

data = np.random.randn(1000)
sns.displot(data , kde=True , rug=True)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```



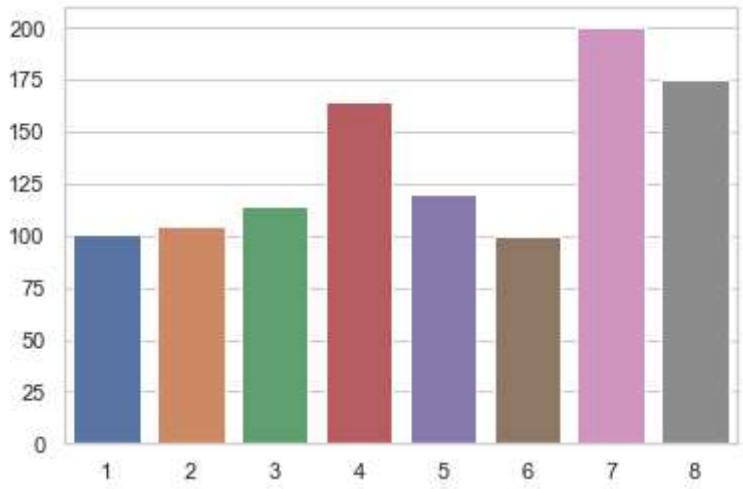
In [1]:
```
#Q5
import seaborn as sns
from pylab import*
sns.set_theme(style="whitegrid")
ax=sns.barplot([1,2,3,4,5,6,7,8],[101,105,114,164,120,100,200,175])
show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [2]:
```
#Q5
import seaborn as sns
from pylab import*
sns.set_theme(style="whitegrid")
ax=sns.barplot([1,2,3,4,5,6,7,8],[101,105,114,164,120,100,200,175])
show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [28]:
```
#Q7
from mayavi import mlab
import numpy as np
x,y,z,value = np.random.random((4,40))
mlab.points3d(x,y,z,value)
```

Out[28]:
```
<mayavi.modules.glyph.Glyph at 0x1a86b1b44a0>
```

Practical 3

In [29]:
```
#Q1. Create dictionary with keys as {1, 2, 3..., 8} and correspoding values

d= {1 : 'one' , 2: 'two' , 3 : 'three' , 4: 'four' , 5 : 'five ' , 6 : 'six' ,
7 : 'seven' , 8: 'eight'}

print(d)

del d[3] , d[5]
d[7] = 's'
d
```

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five ', 6: 'six', 7: 'seven',
8: 'eight'}
```

Out[29]:
```
{1: 'one', 2: 'two', 4: 'four', 6: 'six', 7: 's', 8: 'eight'}
```

In [30]:
```
#Q3. Print all key-value pairs from dictionary:
D = { 'a': 'one' , 'b' : 'two' , 'c' : 'three' , 'd' : 'four'}

for key , value in D.items():
    print(key , value)
```

```
a one
b two
c three
d four
```

In [31]:
```
#Q4. Consider the following dictionary
d = {0 : 'Peter ' , 2 : ' Joseph' , 3 : 'Rickey' ,
     'Emp_ages' :(20 , 33 , 24)}
d.setdefault("Joseph2" , 4)
```

Out[31]:
```
4
```

In [32]:
```
#Q5. Merge the following dictionaries
def Merge(d1 , d2 ):
    return(d2.update(d1))
d1 = {'a' : 1 , 'b' : 2 , 'c' : 3}
d2 = {'d' : 8 , 'e' : 9 , 'f' : 6}

print(Merge(d2,d1))
print(d2)
```

```
None
{'d': 8, 'e': 9, 'f': 6}
```

In [33]:
```
#Q6. Use ** to unpack the dictionary and define a function, which add
#values from dictionary.
d = {'a' : 2 , 'b': 3}
def add(a = 0 ,b = 0 ):
    return a+b
add(**d)
```

Out[33]:
```
5
```

In [ ]:
```
#Q7. Create the ordered dictionary for the data
keys= {0 , 1 , 4 , 5, 6, 3 , 2 }
values= {-9 , -7 , -4 , -6 , -3 , 3 , 4}
from collections import OrderedDict
d = OrderedDict()
for i in keys:
    d[i]=values[i]
...
...
d
```

In [35]:
```
L = [[5 ,3 ,4, 7,8,-6, -8, -3, 0 ], [6,5,6,3] ,[7,4,3,5] ]


x = min(L)
y = max(L)
print(x)
print(y)
```

```
[5, 3, 4, 7, 8, -6, -8, -3, 0]
[7, 4, 3, 5]
```

In [ ]:
```
import heapq
l = [8,-6,-8,-3,0]
heapq.nsmallest(3,1)
```

Practical 4

In [36]:
```
# Q1
L=[[5,5,7,8,-6,-8,-3,0],[6,5,6,3],[7,4,3,5]]
min(L)
```

Out[36]:
```
[5, 5, 7, 8, -6, -8, -3, 0]
```

In [37]:
```
max(L)
```

Out[37]:
```
[7, 4, 3, 5]
```

In [38]:
```
#Q2
import heapq
l = [ 1, 2 , 3, 4 , 5, 6 , 7, 8, 9]
x = 3
heapq.nsmallest(x , l)
```

Out[38]:
```
[1, 2, 3]
```

In [39]:
```
#Q3
adict={'n':1,'s':2,'r':2,'u':2,'t':1}
min(adict)
```

Out[39]:
```
'n'
```

In [40]:
```
max(adict)
```

Out[40]:
```
'u'
```

In [41]:
```
sorted(adict)
```

Out[41]:
```
['n', 'r', 's', 't', 'u']
```

In [42]:
```
#Q4
keys=[0,1,4,5,6,3,2]
values=[-9,-7,-4,-6,-3,3,4]
from collections import OrderedDict
d=OrderedDict()
for i in keys:
    d[i]=values[i]
...
...
d
```

Out[42]:
```
OrderedDict([(0, -9), (1, -7), (4, -3), (5, 3), (6, 4), (3, -6), (2, -4)])
```

In [43]:
```
#Q5
adict={'a':3,'b':5,'c':1}
min(adict)
```

Out[43]:
```
'a'
```

In [44]:
```
max(adict)
```

Out[44]:
```
'c'
```

In [45]:
```
#Q6
adict={'one':'uno','three':'tres','two':'dos'}
min(adict)
```

Out[45]:
```
'one'
```

In [46]:
```
max(adict)
```

Out[46]:
```
'two'
```

In [47]:
```
#Q7
L=[[1,2],[6,5],[6,3],[7,4],[3,5]]
min(L)
```

Out[47]:
```
[1, 2]
```

In [48]:
```
max(L)
```

Out[48]:
```
[7, 4]
```

In [49]:
```
#Q8 Example Python program that finds the largest n elements

# from a Python iterable

import heapq



iterable = [5,3,4,7,8,-6,-8,-3,0]

selectCount = 3

largests = heapq.nlargest(selectCount, iterable)

print(largests)
```

```
[8, 7, 5]
```

Practical 5

#Q1. Disscus the importance of Sympy module in Python.

Ans: The SYMPY module allows us to create two dimensional geometrical object , such as points,lines,polygons,and circles, and gives information about these objects like area of an ellipse , checking for collinearity of a set of points , or finding the intersection between two lines . The primary use case of the module involves objects with numerical values , but it is possible to also use symbolic representation.

In [2]:
```
#Q2 Find the distance between points x and y, y and w, x and z, if x =[1, -1], y
from sympy import *
x=Point(1,-1)
y=Point(-2,4)
z=Point(-1,-1)
w=Point(3,5)
x.distance(y)
```

Out[2]: $\sqrt{34}$

In [3]:
```
y.distance(w)
```

Out[3]: $\sqrt{26}$

In [4]:
```
x.distance(z)
```

Out[4]: 2

Q3. Reflect the given points through respective lines

In [6]:
```
#(a) (-3, 6) , x + 2y = 0
from sympy import *
x,y=symbols('x y')
P=Point(-3,6)
P.reflect(Line(x+2*y))
```

Out[6]: $Point2D\left(-\frac{33}{5}, -\frac{6}{5}\right)$

In [7]:
```
#(b) (2, -6) , 2x + 3y = -1
from sympy import *
x,y=symbols('x y')
P=Point(2,-6)
P.reflect(Line(2*x+3*y+1))
```

Out[7]: $Point2D\left(6, 0\right)$

In [8]:
```
#(c) (5, -2 , x - y = 5
from sympy import *
x,y=symbols('x y')
P=Point(5,-2)
P.reflect(Line(x-y-5))
```

Out[8]: $Point2D\left(3, 0\right)$

In [9]:
```
#(d) (6.3, 3.6) , x - 4y = 1
from sympy import *
x,y=symbols('x y')
P=Point(6.3,3.6)
P.reflect(Line(x-4*y-1))
```

Out[9]: $Point2D\left(\frac{1253}{170}, -\frac{58}{85}\right)$

Q4 Reflect the point P [−3 6] through the line 3/√2x − 2y +4/3 = 0.

In [11]:
```
from sympy import *
x,y=symbols('x y')
P=Point(-3,6)
P.reflect(Line((3/sqrt(2))*x-2*y+(4/3)))
```

Out[11]: $Point2D\left(\frac{3}{17} + \frac{64\sqrt{2}}{17}, \frac{50}{51} - \frac{36\sqrt{2}}{17}\right)$

Q5 Apply each of the following transformations on the point P = [2, −5].

In [12]:
```
#(a) Reflection through X-axis.
from sympy import *
x,y=symbols('x y')
P=Point(2,-5)
#REFLECTION THROUGH X AXIS
P.transform(Matrix([[1,0,0],[0,-1,0],[0,0,1]]))
# SCALING IN X COORDINATE BY FACTOR 4
```

Out[12]: $Point2D\,(2, 5)$

In [14]:
```
#(b) Scaling in X-coordinate by factor 4
P.scale(4,0)
# Scaling in Y-coordinate by factor 5
```

Out[14]: $Point2D\,(8, 0)$

In [15]:
```
#(c) Scaling in Y-coordinate by factor 5.
P.scale(0,5)
#Reflection through the line y = -2x.
```

Out[15]: $Point2D\,(0, -25)$

In [16]:
```
#(d) Reflection through the line y = -2x
P.reflect(Line(2*x+y+0))
#Shearing in Y direction by 2 units
```

Out[16]: $Point2D\left(\frac{14}{5}, -\frac{23}{5}\right)$

In [17]:
```
#(e) Shearing in Y direction by 2 units.
P.transform(Matrix([[1,0,0],[2,1,0],[0,0,1]]))
#Scaling in X and Y direction by 4/5 and 7 units respectively.
```

Out[17]: $Point2D\,(-8, -5)$

In [18]:
```
#(f) Scaling in X and Y direction by 4/5and 7 units respectively
P.scale(4/5,7)
#Shearing in both X and Y direction by -3 and 1 units respectively.
```

Out[18]: $Point2D\left(\frac{8}{5}, -35\right)$

In [19]:
```
#(g) Shearing in both X and Y direction by -3 and 1 units respectively.
P.transform(Matrix([[1,-3,0],[1,1,0],[0,0,1]]))
#Rotation about origin by an angle 45 degrees.
```

Out[19]: $Point2D\,(-3, -11)$

In [20]:
```
#(h) Rotation about origin by an angle 45 degrees.
P.rotate(pi/4)
```

Out[20]: $Point2D\left(\frac{7\sqrt{2}}{2}, -\frac{3\sqrt{2}}{2}\right)$

Q6 Apply each of the following transformations on the point P = [−2, 4]

In [21]:
```
#(a) Scaling in X and Y direction by 7/2and 4 units respectively.
from sympy import *
x,y=symbols('x y')
P=Point(-2,4)
#Scaling in X and Y direction by 7/2and 4 units respectively.
P.scale(7/2,4)
```

Out[21]: $Point2D\left(-7, 16\right)$

In [22]:
```
#(b) Shearing in both X and Y direction by 4 and 7 units respectively.
#Shearing in both X and Y direction by 4 and 7 units respectively.
P.transform(Matrix([[1,4,0],[7,1,0],[0,0,1]]))
```

Out[22]: $Point2D\left(26, -4\right)$

In [23]:
```
#(c) Reflection through the line y = 2x + 3.
#Reflection through the line y = 2x + 3.
P.reflect(Line(-2*x+y-3))
```

Out[23]: $Point2D\left(2, 2\right)$

In [24]:
```
#(d) Shearing in Y direction by 7 units.
#Shearing in Y direction by 7 units.
P.transform(Matrix([[1,0,0,],[7,1,0],[0,0,1]]))
```

Out[24]: $Point2D\left(26, 4\right)$

In [25]:
```
#(e) Rotation about origin by an angle 48 degrees.
#Rotation about origin by an angle 48 degrees.
from math import*
angle=radians(48) #convert degree in radians
P.rotate(angle)
```

Out[25]: $Point2D\left(-\frac{431084051462729}{1000000000000000}, \frac{929869355063}{781250000000}\right)$

In [26]:
```
#(f) Reflection through line 3x + 4y = 5.
#Reflection through line 3x + 4y = 5.
P.reflect(Line(3*x+4*y-5))
```

Out[26]: $Point2D\left(-\frac{16}{5}, \frac{12}{5}\right)$

In [27]:
```
#(g) Scaling in X-coordinate by factor 6.
#Scaling in X-coordinate by factor 6.
P.scale(6,0)
```

Out[27]: $Point2D\left(-12, 0\right)$

In [28]:
```
#(h) Scaling in Y-coordinate by factor 4.1.
#Scaling in Y-coordinate by factor 4.1.
P.scale(0,4.1)
```

Out[28]: $Point2D\left(0, \frac{82}{5}\right)$

Practical 6

Q1. Rotate the line passing through points A{5 1}and B{2 5}.aboutorigin through an angle 2700.

In [30]:
```
from sympy import *
x, y= symbols('x y')
L= Line((5,1) , (2, 5))
L.rotate((3*pi)/2)
```

Out[30]: $Line2D\left(Point2D\left(1, -5\right), Point2D\left(5, -2\right)\right)$

Q2. Rotate the line passing through points A [0 −1] and B [2 −5] about origin through an angle 800.

In [31]:
```
from sympy import *
x, y= symbols('x y')
L= Line((0 , -1 ) , (2, -5))
L.rotate(80)
```

Out[31]: $Line2D\left(Point2D\left(\sin\left(80\right), -\cos\left(80\right)\right), Point2D\left(5\sin\left(80\right) + 2\cos\left(80\right), 2\sin\left(80\right) - 5\cos\left(\right.\right.\right.$

Q3. If the line segment joining the points A[−2 5], B[−4 3] is transformed to the line segment A∗B∗ by the transformation matrix, [T] =2 −2 3 −6 , then find the midpoint and length of A*B*

In [32]:
```
from sympy import *
x,y=symbols('x y')
A=Point(-2,5)
B=Point(-4,3)
A1=A.transform(Matrix([[2,-2,0],[3,-6,0],[0,0,1]]))
B1=B.transform(Matrix([[2,-2,0],[3,-6,0],[0,0,1]]))
L=Segment(A1,B1)
L.midpoint
```

Out[32]: $Point2D\left(6, -18\right)$

Q4. Reflect the line segment joining the points A[5, 2] and B[−3, 4] through the line y = 2x − 1.

In [34]:
```
from sympy import *
x,y=symbols('x y')
A=Point(5,2)
B=Point(-3,4)
L=Segment(A1,B1)
L.reflect(Line(2*x-y-1))
```

Out[34]: $Segment2D\left(Point2D\left(-\frac{133}{5}, -\frac{36}{5}\right), Point2D\left(-\frac{39}{5}, -\frac{28}{5}\right)\right)$

Q5. Rotate the line by 75 degrees having two points (0, 0) and (0, 1). Also find its equation after applying rotation.

In [35]:
```
from sympy import *
L=Line(Point(0,0),Point(0,1))
L.rotate((5*pi)/12)
L1=L.rotate((5*pi)/12)
L1.equation()
```

Out[35]: $x\left(-\frac{\sqrt{6}}{4} + \frac{\sqrt{2}}{4}\right) + y\left(-\frac{\sqrt{6}}{4} - \frac{\sqrt{2}}{4}\right)$

Q6. Rotate the segment by 180 degrees having end points (1, 0) and (2, −1).

In [36]:
```
from sympy import *
L=Segment(Point(1,0),Point(2,-1))
L.rotate(pi)
```

Out[36]: $Segment2D\left(Point2D\left(-1, 0\right), Point2D\left(-2, 1\right)\right)$

Q7. Rotate the ray by 90 degrees having starting point (0, 0) in the

direc tion of (4, 4).

In [37]:
```
from sympy import *
L=Ray(Point(0,0),Point(4,4))
L.rotate(-pi/2)
```

Out[37]: $Ray2D\left(Point2D\left(0,0\right),Point2D\left(4,-4\right)\right)$

Q8. Reflect the line 4x + 3y = 5 through line x + y = 0 and find the equation of reflected line.

In [38]:
```
from sympy import *
L=Line(4*x+3*y-5)
L1=Line(x+y)
Line=L.reflect(L1)
Line.equation()
```

Out[38]: $x+\frac{4y}{3}+\frac{5}{3}$

Q9. Reflect the segment having two endpoints (2, 3),(4, 6) through line 7x + 6y = 3.

In [39]:
```
from sympy import *
P=Point(2,3)
Q=Point(4,6)
S=Segment(P,Q)
x,y=symbols('x y')
L=Line(7*x+6*y-3)
S.reflect(L)
```

Out[39]: $Segment2D\left(Point2D\left(-\frac{236}{85},-\frac{93}{85}\right),Point2D\left(-\frac{514}{85},-\frac{222}{85}\right)\right)$

Q10. Reflect the line segment having starting point (0, 0) in the direction of (2, 4) through line x − 2y = 3

In [40]:
```
from sympy import *
P=Point(0,0)
Q=Point(2,4)
R=Ray(P,Q)
x,y=symbols('x y')
L=Line(x-2*y-3)
R.reflect(L)
```

Out[40]: $Ray2D\left(Point2D\left(\frac{6}{5},-\frac{12}{5}\right),Point2D\left(\frac{28}{5},-\frac{16}{5}\right)\right)$

Practical 7

Q1If the line with points A [2 1], B [4 −1] is transformed by the transformation matrix, [T] = 1 2 2 1, then find the equation of transformed line.

In [41]:
```
from sympy import *
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()
```

Out[41]: $-2x - 2y + 18$

#Q2. Rotate the line passing through points A[1 1]and B [5 5] about origin through an angle 90 degree.

In [42]:
```
L=Line((1,3),(5,5))
L.rotate(pi/2)
```

Out[42]: $Line2D\left(Point2D\left(-3, 1\right), Point2D\left(-5, 5\right)\right)$

Q3If the line segment joining the points A [2 5] , B [4 3] is transformed to the line segment A∗B∗ by the transformation matrix, [T] =2 −3 4 1 , then find the midpoint of A∗B∗.

In [46]:
```
from sympy import*
A=Point(2,5)
B=Point(4,3)
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
```

Out[46]: $Point2D\left(22, 13\right)$

#Q4 Reflect the line segment joining the points A[5 −3] and B[1 4] through the line y = 2x + 3.

In [47]:
```
from sympy import *
x,y=symbols('x y')
A=Point(5,-3)
B=Point(1,4)
L=Segment(A1,B1)
L.reflect(Line(2*x-y+3))
```

Out[47]:
$$Segment2D\left(Point2D\left(-8,27\right),Point2D\left(-\frac{12}{5},\frac{131}{5}\right)\right)$$

Q5. Suppose that the line segment between the points A [1 4] and B [3 6] is transformed to the line segment A∗B∗ using the transformation matrix [T] = 2 −1 1 −3. Find slope of the transformed line segment A∗B

In [48]:
```
from sympy import *
x,y=symbols('x y')
A=Point(1,4)
B=Point(3,6)
A1=A.transform(Matrix([[2,-1,0],[1,3,0],[0,0,1]]))
B1=B.transform(Matrix([[2,-1,0],[1,3,0],[0,0,1]]))
L=Segment(A1,B1)
L.slope
```

Out[48]:
$$\frac{2}{3}$$

#Q6. If the two lines 2x − y = 5 and x + 3y = −1 are transformed us ing the transformation matrix [T] = −2 3 1 1 then find the point of intersection of the traformed line.

In [49]:
```
from sympy import *
x,y=symbols('x y')
L1=Line(2*x-y-5)
L2=Line(x+3*y+1)
P=L1.intersection(L2)
P=P[0]
P.transform(Matrix([[-2,-1,0],[1,3,0],[0,0,1]]))
```

Out[49]:
$$Point2D\left(-5,-5\right)$$

Q7. If we apply shearing on the line 2x + y = 3 in x and y directions by 2 and -3 units respectively, then find the equation of the resulting line.

In [50]:
```
from sympy import *
x, y= symbols('x y')
l = Line(2*x + y - 3 )
point = l.points
p = point[0]
q = point[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])))
li = Line(p1, q1)
li.equation
```

Out[50]:
```
<bound method Line2D.equation of Line2D(Point2D(6, 3), Point2D(3, -2))>
```

Practical 8: Polygons (Unit 4)

Q1 Write a python programme to draw a polygon with vertices (0, 1),(1, 0),(-2, 2),(1, -4) and find its area and perimete

In [1]:
```
from sympy import *
A=Point(0,1)
B=Point(1,0)
C=Point(-2,2)
D=Point(1,-4)
P=Polygon(A,B,C,D)
P.area
```

Out[1]: 4

In [2]:
```
P.perimeter
```

Out[2]: $\sqrt{2}+\sqrt{13}+\sqrt{26}+3\sqrt{5}$

2. Write a python programme to draw a regular polygon with 8 sides and radius 6 centered at origin and find its area and perimeter.

In [3]:
```
from sympy import *
P=Polygon((0,0),6,n=8)
P.area
```

Out[3]: $\dfrac{576-288\sqrt{2}}{-4+4\sqrt{2}}$

In [4]:
```
P.perimeter
```

Out[4]: $48\sqrt{2-\sqrt{2}}$

3. Write a python programme to draw a regular polygon with 6 sides and radius 1 centered at (1, 2) and find its area and perimeter.

In [5]:
```
#Q3
from sympy import *
P=Polygon((1,2),1,n=6)
P.area
```

Out[5]: $\frac{3\sqrt{3}}{2}$

In [6]:
```
P.perimeter
```

Out[6]: 6

4. Write a python programme to draw a regular polygon with 7 sides and radius 1.5 centered at (2, 2) and reflect it through line x − y = 5

In [7]:
```
#Q4
from sympy import *
x,y=symbols('x y')
P=Polygon((2,2),1.5,n=7)
P.reflect(Line(x-y-5))
```

Out[7]: $RegularPolygon\left(Point2D\left(7,-3\right),-1.5,7,\frac{3\pi}{14}\right)$

5. Write a python programme to draw a polygon with vertices (0, 0), (2, 0),(2, 3),(1, 6) and rotate by 180 degrees and find internal angle at each vertex.

In [8]:
```
#Q5
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/3)
```

Out[8]: $Polygon\left(Point2D\left(0,0\right),Point2D\left(1,\sqrt{3}\right),Point2D\left(1-\frac{3\sqrt{3}}{2},\frac{3}{2}+\sqrt{3}\right),Point2D\left($

6. Write a python programme to draw a polygon with vertices (0, 0), (1, 0),(2, 2),(1, 4) and find its area and perimeter.

In [12]:
```
#Q6
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)
x = P.area
y = P.perimeter

print( x  , y)
```

```
4 1 + sqrt(17) + 2*sqrt(5)
```

7. Write a python programme to draw a regular polygon with 4 sides and radius 6 centered at origin and find its area and perimeter.

In [13]:
```
#Q7
from sympy import *
P=Polygon((0,0),6,n=4)
P.area
```

Out[13]: 72

In [15]:
```
P.perimeter
```

Out[15]: $24\sqrt{2}$

8. Write a python programme to draw a regular polygon with 8 sides and radius 2 centered at (−1, 2) and find its area and perimeter.

In [16]:
```
#Q8
from sympy import *
P=Polygon((1,-2),2,n=8)
P.area
```

Out[16]: $\frac{64 - 32\sqrt{2}}{-4 + 4\sqrt{2}}$

In [18]:
```
P.perimeter
```

Out[18]: $16\sqrt{2 - \sqrt{2}}$

9. Write a python programme to draw a regular polygon with 7 sides and radius 6 centered at (−2, 2) and reflect it through line x − 2y = 5

In [19]:
```
#Q9
from sympy import *
x,y=symbols('x y')
P=Polygon((-2,2),6,n=7)
P.reflect(Line(x-2*y-5))
```

Out[19]:
$$RegularPolygon\left(Point2D\left(\frac{12}{5},-\frac{34}{5}\right),-6,7,-\frac{2\pi}{7}+\operatorname{atan}\left(\frac{4}{3}\right)\right)$$

10. Write a python programme to draw a polygon with vertices (0, 0), (−2, 0),(5, 5),(1, −6) and rotate by 180 degrees and find internal angle at each vertex

In [20]:
```
#Q10
from sympy import *
A=Point(0,0)
B=Point(-2,0)
C=Point(5,5)
D=Point(1,-6)
P=Polygon(A,B,C,D)
P.rotate(pi/3)
```

Out[20]:
$$Polygon\left(Point2D\left(0,0\right),Point2D\left(-1,-\sqrt{3}\right),Point2D\left(\frac{5}{2}-\frac{5\sqrt{3}}{2},\frac{5}{2}+\frac{5\sqrt{3}}{2}\right),Poin$$

Practical 9

1. Reflect the pol ABC through the line y = 3, where A[1 0], B[2 − 1], C[−1 3].

In [21]:
```
#Q1
from sympy import *
A=Point(0,0)
B=Point(-2,0)
C=Point(5,5)
D=Point(1,-6)
P=Polygon(A,B,C,D)
P.rotate(pi/3)
```

Out[21]:
$$Polygon\left(Point2D\left(0,0\right),Point2D\left(-1,-\sqrt{3}\right),Point2D\left(\frac{5}{2}-\frac{5\sqrt{3}}{2},\frac{5}{2}+\frac{5\sqrt{3}}{2}\right),Poin$$

2. Rotate the triangle ABC by 90 degree, where A[1 2], B[2 −2], C[−1 2].

In [22]:
```
#Q2
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)
```

Out[22]: $Triangle\left(Point2D\left(-2, 1\right), Point2D\left(2, 2\right), Point2D\left(-2, -1\right)\right)$

3. Find the area and perimeter of the triangle ABC, where A[0 0], B[5 0], C[3 3].

In [23]:
```
#Q3
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
```

Out[23]: $\frac{15}{2}$

In [24]:
```
T.perimeter
```

Out[24]: $\sqrt{13} + 3\sqrt{2} + 5$

4. Find the angle at each vertices of the triangle ABC, where A[0 0], B[2 2], C[0 2].

In [29]:
```
 #Q4
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]
```

Out[29]: $\frac{\pi}{4}$

In [30]:
```
T.angles[B]
```

Out[30]: $\frac{\pi}{4}$