SELECTED SOLUTIONS

CHAPTER 6

Exercise Set 6.1.1

1. There is no error message. It simply draws the axes but not the point.
2. **plot**([-5,-4,-3,-2,-1,0,1,2,3,4,5],[5,4,3,2,1,0,1,2,3,4,5],'go')
3. A. xmin = 0, xmax = 4, xscl = .5, ymin = 50, ymax = 100, yscl = 1

   B. No it is not a function.

   C. A more informative scaling would be **axis**([0,5,0,100]).
4. Scale the axes at [-20,20] x [-20,20] and draw the xy axes in black in the middle of the screen.

```
from matplotlib import pyplot as plt

plt.axis([-20,20,-20,20])
plt.plot([-20,20],[0,0], color='k')
plt.plot([0,0],[-20,20],color='k')
plt.show()
```

5. Write a program that will plot the following ordered pairs, connected with blue lines:

(1,1000), (2,750), (3, 1200), (4, 1450), (5, 1600), (6, 2000)

Label the x-axis as 'Month', the y-axis as 'Revenue', and the title of the graph 'Six Month Projection'.

```
from matplotlib import pyplot as plt

x_values = [1,2,3,4,5,6]
y_values = [1000,750,1200,1450,1600,2000]
plt.plot(x_values,y_values)                ## creates the plot object
plt.xlabel('Month')
plt.ylabel('Revenue')
plt.title('Six Month Projection')
plt.show()
```

Exercise Set 6.2.1

1. Slope between (4, 7) and (-2, 3) is 2/3 (positive)

2. Slope between $\left(\frac{1}{3},\frac{2}{5}\right)$ and $\left(\frac{5}{6},\frac{1}{5}\right)$ is -2/5 (negative)
3. Slope between (7, -8) and (8, -8) is undefined (vertical)
4. Slope between (2, 4) and (3, -4) is -8 (negative)
5. $m_1 = 2$ and $m_2 = -2$ are neither
6. $m_1 = \frac{3}{9}$ and $m_2 = -3$ are perpendicular
7. $m_1 = 0$ and $m_2 =$ undefined are perpendicular

Exercise Set 6.2.2

1. The program does not work with fractions. Modify:

```
from fractions import Fraction
p1 = input('x1,y1? ')
x1,y1 = [Fraction(i) for i in p1.split(',')]
p2 = input('x2,y2? ')
x2,y2 = [Fraction(i) for i in p2.split(',')]
m1 = Fraction(y2-y1,x2-x1)
print('m=',m1)
```

2. No the program does not work when x1 = x2; you end up with an undefined slope. Modify:

```
from fractions import Fraction
p1 = input('x1,y1? ')
x1,y1 = [Fraction(i) for i in p1.split(',')]
p2 = input('x2,y2? ')
x2,y2 = [Fraction(i) for i in p2.split(',')]
if x1==x2:
   print('slope is undefined')
else:
    m1 = Fraction(y2-y1,x2-x1)
    print('m=',m1)
```

3. Write a program that asks the user to input the slopes of two lines, m1 and m2. Compare the slopes. Output whether they are parallel, perpendicular, or neither. *(Hint: a pair of slopes m_1 and m_2 are opposite reciprocals if their product is -1.)*

```
from fractions import Fraction
m1 = Fraction(input('m1? '))
m2 = Fraction(input('m2 '))
```

```
if m1==m2:
   print('parallel')
elif m1*m2==-1:
   print('perpendicular')
else:
   print('neither')
```

4. Write a program (or modify Program 6.2.1) that asks the user to input two ordered pairs (x_1,y_1) and (x_2,y_2). Calculate the slope of the line through the two points, and also calculate and output the parallel and perpendicular slopes.

```
from fractions import Fraction
p1 = input('x1,y1? ')
x1,y1 = [Fraction(i) for i in p1.split(',')]
p2 = input('x2,y2? ')
x2,y2 = [Fraction(i) for i in p2.split(',')]
if x1==x2:
   print('slope is undefined')
   print('parallel slope is also undefined')
   print('perpendicular slope is zero')
elif y1==y2:
    print('slope is zero')
    print('parallel slope is also zero')
    print('perpendicular slope is undefined')
else:
    m1 = Fraction(y2-y1,x2-x1)
    print('m=',m1)
    print('parallel slope is m=',m1)
    m2 = -1/m1
    print('perpendicular slope is m=',m2)
```

Exercise Set 6.2.3

1. y = (-5/2)x – 12
2. y = (5/8)x + ½
3. y = 8
4. x = 5/4
5. y = (2/3)x + 1/3
6. y = (3/7)x – 3
7. y = 5

8. y = -7x + 24

9. y = 8

10. x = 0

11. b = y - mx

12. y = -0.7x + 154

13. Write a program to calculate target heart rate based on age.

```
x = int(input("Age? "))
print("Your target heart rate for moderate activity is")
print(-0.7*x + 154)
```

Exercise Set 6.2.4

1. m = 4/5, b = -53/5

2. Append the following statement to the program:

```
print(x1, y1, x2, y2)
```

3. When x1 = x2, it results in an undefined slope which is a vertical line. The error message can be circumvented with the following changes:

```
from matplotlib import pyplot as plt
from random import randint
from fractions import Fraction

x1 = randint(-10,10)
y1 = randint(-10,10)
x2 = randint(-10,10)
y2 = randint(-10,10)

plt.plot([x1,x2],[y1,y2])
if x1==x2:
   heading = 'x ='+str(x1)
else:
   m = Fraction(y2-y1,x2-x1)
   b = y1 - m*x1
   heading = 'y = ('+str(m)+')x + '+str(b)

plt.title(heading)
plt.show()
```

4. When y1 =y2, it results in a zero slope which is a horizontal line.

5. Modify the program so that the user can enter values for x1,y1 and x2,y2.

```
from matplotlib import pyplot as plt
from fractions import Fraction

p1 = input('x1,y1? ')
x1,y1 = [Fraction(i) for i in p1.split(',')]
p2 = input('x2,y2? ')
x2,y2 = [Fraction(i) for i in p2.split(',')]

plt.plot([x1,x2],[y1,y2])
m = Fraction(y2-y1,x2-x1)

b = y1 - m*x1
heading = 'y = ('+str(m)+')x + '+str(b)

plt.title(heading)
plt.show()
```

6. Replace

plt.plot([x1,x2],[y1,y2])

with:

```
colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'w']
rc = randint(0, 7)
plt.plot([x1,x2],[y1,y2],c=colors[rc])
```

Exercise Set 6.2.5

1. Modify the program to prompt the user to enter xmin, xmax.

```
from matplotlib import pyplot as plt
from sympy import Symbol, sympify

x = Symbol('x')
domain = input('xmin, xmax? ')
xmin, xmax = [int(k) for k in domain.split(',')]
x_list = range(xmin,xmax+1)
y_list = [ ]                        # initialize y_list
f_str = input('function? ')
f = sympify(f_str)

# For each a in x_list,  find f(a) and assign to y_list
y_list = [ f.subs({x:a}) for a in x_list ]
```

```
plt.plot(x_list,y_list,'o')
plt.show()
```

2. Modify the program so that the title of the graph is "f(x) = mx + b over [xmin,xmax]" in which the m,b,xmin and xmax reflect the actual values that were input.

Append the following 2 lines:

```
title="f(x) = "+str(m)+"x +"+str(b)+" over ["+str(xmin)+","+str(xmax)+"]"
plt.title(title)
```

3. If 3x-17 is input, and a domain of [-25, 25], what are the coordinates of the left and right endpoints?

f(-25) = -92 so the left endpoint is (-25, -92).

f(25) = 58, so the right endpoint is (25, 58).

4. Modify the program so that a line is drawn instead of discrete dots.

Change:

```
x_list = range(xmin,xmax+1)
```
to:
```
x_list = [xmin,xmax+1]
```

and change:
```
plt.plot(x_list,y_list,'o')
```

to:
```
plt.plot(x_list,y_list)
```

5. CHALLENGE: Modify the program so that the user is also prompted to input a point. Draw a line through that point, perpendicular to the given line.

Append the following lines to the program.

```
:
:
# parallel line
m,b=Poly(f).coeffs()
point=input('x,y? ')
x,y = [int(k) for k in point.split(',')]
b2 = y-m*x
y_list2 = [ ]
y_list2 = [ m*a+b2 for a in x_list ]
plt.plot(x_list, y_list2)
parallel = '(' + str(m)+')*x+'+str(b2)
title = 'Parallel Line is f(x)=' + parallel
plt.title(title)
```

```
plt.show()
```

Exercise Set 6.2.6

1. What are the ymin and ymax of $f(x) = \frac{1}{10}(x^3 - 6x^2 - 27x + 140)$?
   The minimum occurs at f(-10) = -119. The maximum occurs at f(10) = 27.

2. plt.**text**(10, -10, 'Lisa')

3. Output vertex if quadratic.

Append the following lines:

```
if degree(f)==2:
  a,b,c = Poly(f).all_coeffs()
  vx = -b/(2*a)
  vy= f.subs({x:vx})
  plt.plot([vx],[vy],'o')
  ordered_pair = "("+str(round(vx,0)) + "," + str(round(vy,0)) + ")"
  plt.text(vx,vy,ordered_pair)
```

4. Modify program **function_plot** to draw the x- and y-axes.

   Append the following lines:

```
ymin = min(y_list)
ymax = max(y_list)
plt.plot([xmin,xmax],[0,0],c='k')  # plot the x-axis
plt.plot([0,0],[ymin,ymax],c='k')  # plot the y-axis
plt.plot(x_list,y_list)            # plot the function
```

5. Modify program **function_plot** to determine if a point is on the graph.

Append the following lines:

```
point = input('point? ')
a, b = [int(k) for k in point.split(',')]
plt.plot([a],[b],'o')
p = '('+str(a)+','+str(b)+')'
plt.text(a,b,p)
if f.subs({x:a})==b:
  print(p,'is on the graph')
else:
  print(p,'is not on the graph')
plt.show()
```

Exercise Set 6.2.7

1.

Exercise Set 6.3.1

1. $3\sqrt{10}$

2. $5\sqrt{10}$

3. The lengths of the 3 sides are 72.1, 130,108.2. Yes it is a right triangle because the slope from (0,0)-(60,40) is 2/3 and the slope from (0,0) to (-60,90) is -3/2.

4. Write a function **distance**(x1,y1,x2,y2) that calculates and returns the distance between two points. Call the function from a main program. Output the answer as both a decimal and a simplified radical expression.

```
from sympy import *
import math

def distance(x1,y1,x2,y2):
   d = math.sqrt((x2-x1)**2+(y2-y1)**2)
   return d                          # returns float value using math sqrt()

# main program
x = Symbol('x')
point1 = input('x1,y1? ')
x1,y1 = [int(i) for i in point1.split(',') ]
point2 = input('x2,y2? ')
x2,y2 = [int(i) for i in point2.split(',') ]

r = (x2-x1)**2 + (y2-y1)**2      # radicand
rad = sqrt(r)                    # simplified radical using SymPy sqrt()
pprint(rad)
print(distance(x1,y1,x2,y2))
```

5. Modify the main program you wrote in #4 so that it also graphs the line segment between the two points. Output the distance as a title, rounded to 1 decimal point.

```
from matplotlib import pyplot as plt
import math
```

```
def distance(x1,y1,x2,y2):
   d = math.sqrt((x2-x1)**2+(y2-y1)**2)
   return d

# main program
point1 = input("Point 1? ")
point2 = input("Point 2? ")

x1,y1 = [int(i) for i in point1.split(',') ]
x2,y2 =[int(i) for i in point2.split(',')]

plt.plot([x1,x2],[y1,y2])

d = distance(x1,y1,x2,y2)
d = round(d,1)

plt.title("Distance is "+str(d))
plt.show()
```

6. 35.9

Exercise Set 6.3.2

1. A. $3\sqrt{22} \approx 14.07$

   B. $\sqrt{690} \approx 26.27$

2. Modify program 3dplot so that it also prints the distance between the two points.

```
from sympy import *
from matplotlib import pyplot as plt

ax = plt.axes(projection="3d")        # declare ax as an alias for plt.axes
ax.plot3D([0,10],[0,10],[0,10])
ax.set_xlabel('x axis')
ax.set_ylabel('y axis')
ax.set_zlabel('z axis')
r = (10-0)**2 + (0-10)**2  + (10-0)**2
rad = sqrt(r)
title = 'Distance is '+str(rad)
plt.title(title)
plt.show()
```

3. Modify program 3dplot so that it allows the user to input any two ordered triples.

```
from matplotlib import pyplot as plt
```

```
from sympy import *

ax = plt.axes(projection="3d")        # declare ax as an alias for plt.axes
point1 = input("Point 1? ")
point2 = input("Point 2? ")

x1,y1,z1 = [int(i) for i in point1.split(',') ]
x2,y2,z2 =[int(i) for i in point2.split(',')]

ax.plot3D([x1,x2],[y1,y2],[z1,z2])
ax.set_xlabel('x axis')
ax.set_ylabel('y axis')
ax.set_zlabel('z axis')
r = (x2-x1)**2 + (y2-y1)**2  + (z2-z1)**2
rad = sqrt(r)
title = 'Distance is '+str(rad)
plt.title(title)
plt.show()
```

4.

Exercise Set 6.4.1

1. *A.* $\left(-\frac{9}{2},\frac{19}{2}\right)$

   B. $\left(\frac{49}{2},-\frac{9}{2}\right)$

2. (17, -11)

3. a = 3 → AC = 10

4. Write a program that calculates the midpoint of a line segment.

```
from fractions import Fraction
point1 = input("Point 1? ")
point2 = input("Point 2? ")

x1,y1 = [int(i) for i in point1.split(',') ]
x2,y2 =[int(i) for i in point2.split(',')]

xm = Fraction(x1+x2,2)
ym = Fraction(y1+y2,2)
print(xm, ',', ym)
```

5. Write a program that calculates the midpoint of a line segment. Draw the line segment and label the distance at the coordinate of the midpoint.

```
from matplotlib import pyplot as plt
from math import sqrt

def distance(x1,y1,x2,y2):
   d = sqrt((x2-x1)**2+(y2-y1)**2)
   return d

def midpoint(x1,y1,x2,y2):
   xm = (x1+x2)/2
   ym = (y1+y2)/2
   return xm,ym

# main program
point1 = input("Point 1? ")
point2 = input("Point 2? ")

x1,y1 = [int(i) for i in point1.split(',') ]
x2,y2 =[int(i) for i in point2.split(',')]
plt.plot([x1,x2],[y1,y2])

d = distance(x1,y1,x2,y2)
d = round(d,1)

xm,ym = midpoint(x1,y1,x2,y2)
plt.text(xm,ym,d)
plt.show()
```

6. Write a program that asks the user to input one endpoint (x1,y1) and the midpoint (m1,m2) then calculates and outputs the coordinates of the other endpoint (x2,y2).

```
from matplotlib import pyplot as plt

point1 = input("Point 1? ")
midpoint = input("Midpoint? ")

x1,y1 = [float(i) for i in point1.split(',') ]
xm,ym =[float(i) for i in midpoint.split(',')]
x2 = 2*xm - x1
y2 = 2*ym - y1

plt.plot([x1,x2],[y1,y2])

plt.text(x1,y1,point1)
```

```
p2text = str(x2)+','+str(y2)
plt.text(x2,y2,p2text)

plt.text(xm,ym,midpoint)
plt.show()
```

7. Write a program that draws a square, then connects the midpoints of the square.

```
from matplotlib import pyplot as plt

plt.axis([-15,15,-15,15])
plt.plot([-10,10,10,-10,-10],[10,10,-10,-10,10],c='k')
plt.plot([0,10,0,-10,0],[10,0,-10,0,10],c='r')
plt.show()
```

8. The effect is setting the aspect ratio of the x- and y- axes the same (so that the square looks more like a square and less like a rectangle).

9.

Exercise Set 6.4.2

1. midpoint: (1/2, 3, 5/2); distance = $5\sqrt{2}$

2. midpoint: (9/2, ½, ½); distance = $\sqrt{435}$

3. Modify program 3dplot so that it calculates and labels the midpoint of the line segment.

```
from matplotlib import pyplot as plt

ax = plt.axes(projection="3d")        # declare ax as an alias for plt.axes
point1 = input("Point 1? ")
point2 = input("Point 2? ")

x1,y1,z1 = [int(i) for i in point1.split(',') ]
x2,y2,z2 =[int(i) for i in point2.split(',')]

ax.plot3D([x1,x2],[y1,y2],[z1,z2])
ax.set_xlabel('x axis')
ax.set_ylabel('y axis')
ax.set_zlabel('z axis')
xm, ym, zm = (x1+x2)/2, (y1+y2)/2, (z1+z2)/2
mp = '('+str(xm)+','+str(ym)+','+str(zm)+')'
ax.text(xm,ym,zm,mp)
plt.show()
```

Exercise Set 6.5.1

1. $x^2 + y^2 = 12$

2. $(x - 2)^2 + (y + 3)^2 = 16$

3. Center: (2, -7); Radius: 6

4. $x^2 + y^2 + 10x + 4y - 20 = 0$

5. Center: (0, 2); Radius: $\sqrt{13}$

6. $(x - 3)^2 + (y + 2)^2 = 25$

7. $(x - 2)^2 + (y - 2)^2 = 5$

8. $(x - 2)^2 + (y - 3)^2 = 4$

9. Write a program that asks the user to input the center and radius of a circle. Output the equation of the circle in standard form.

```
center = input("center? ")
h, k = [int(i) for i in center.split(',')]
r = int(input("radius? "))
circle = '(x - '+str(h)+')**2 + (y - '+str(k)+')**2 = '+str(r)
print(circle)
```

10. Write a program that asks the user to input the endpoints of the diameter (as in #7 above); output the equation of the circle in standard form.

```
from math import sqrt

point1 = input("Point 1? ")
point2 = input("Point 2? ")

x1,y1 = [int(i) for i in point1.split(',') ]
x2,y2 =[int(i) for i in point2.split(',')]

d = sqrt((x2 - x1)**2 + (y2 - y1)**2)
r = d/2


h = (x1+x2)/2
k = (y1+y2)/2
r_squared = round(r**2,1)

print('(x-',h,')^2 + (y-',k,')^2 =',r_squared)
```

11. Write a program that asks user to input general form of a circle. Find the x-intercepts.

```
from sympy import *
x = Symbol('x')
circ = input('General form A,B,C,D,E: ')
A,B,C,D,E = [n for n in circ.split(',')]
gen = A+'*x**2+'+C+'*x+' + E
gen = sympify(gen)
zeros= solve(gen)
real = True;
for n in zeros:
   if n.is_complex:
      real = False;
if real == True:
   print(zeros)
else:
   print('no x-intercepts')
```

Exercise Set 6.5.2

1. A. $y = \pm\sqrt{81-(x+7)^2}$

   B. $y = \pm\sqrt{9-(x-5)^2}-1$

2. Fill in yellow highlighted code to draw lower semi-circle.

```
y_list2 = [-sqrt(r**2-(x-h)**2)+k for x in x_list]
plt.plot(x_list,y_list2)
```

Exercise Set 6.5.3

1. Write a program that draws a circle inscribed inside a square.

2. Write a program that draws a square inside a circle.

3. Write a program that generates and draws randomly placed circles in different colors on the screen.

```
from matplotlib import pyplot as plt
from random import randint
color = ['b','g','r','c','m','y']
for i in range(1,20):
   h = randint(1,10)
```

```
    k = randint(1,10)
    r = randint(1,10)
    c = randint(0, 5)
    circle=plt.Circle((h,k), r,ec=color[c],fc='white')
    ax = plt.gca()
    ax.add_patch(circle)
plt.axis('scaled')
plt.show()
```

4.

Exercise Set 6.6.1

1. A. The 3 vertices are at $(0, \frac{\sqrt{3}}{2} \cdot 7)$, (-7/2, 0) and (7/2, 0).

   B. The area is $\frac{\sqrt{3}}{2} \cdot 49$.

2. A. The 3 vertices are at $(0,10)$, $\left(-\frac{7}{2}, 10 - \frac{\sqrt{3}}{2} \cdot 10\right)$ and $\left(\frac{7}{2}, 10 - \frac{\sqrt{3}}{2} \cdot 10\right)$

   B. The area is $\frac{\sqrt{3}}{2} \cdot 49$.

3. The 3 vertices are at (6, 9), $\left(4,\ 9 - \frac{\sqrt{3}}{2} \cdot 9\right)$ and $\left(8,\ 9 - \frac{\sqrt{3}}{2} \cdot 9\right)$

Exercise Set 6.6.2

1. Fill in the yellow highlighted code with:

```
plt.plot([0,x1,x2,0],[y,0,0,y])
```

2. Append the following two lines:

```
area = round((d**2)*sqrt(3)/2,2)
plt.annotate(area,xy=(0, y/2))
```

3. Modify program so that y-coordinate of vertex is at (0, f).

```
from matplotlib import pyplot as plt
from math import sqrt

d = float(input("Side length? "))
f = float(input("y-coordinate of top vertex? "))

# top vertex is (0,f)
```

```
# y-coordinate of base
y = f - d* sqrt(3)/2

# base vertices are at (x1, y) and (x2, y)
x1 = -d/2
x2 = d/2


plt.plot([0,x1,x2,0],[f, y, y, f])

area = round((d**2)*sqrt(3)/2,2)
# print area at midpoint
plt.annotate(area,xy=(0, (y+f)/2))
plt.show()
```

4. Modify the program so that the top vertex can be at any ordered pair (e, f).

```
from matplotlib import pyplot as plt
from math import sqrt

d = float(input("Side length? "))
vertex = input("Coordinates of top vertex? ")
e,f = [float(i) for i in vertex.split(',')]

# top vertex is (e,f)
# y-coordinate of base
y = f - d* sqrt(3)/2

# base vertices are at (x1, y) and (x2, y)
x1 = e-d/2
x2 = e+d/2

plt.plot([e,x1,x2,e],[f, y,y, f])

area = round((d**2)*sqrt(3)/2,2)
plt.annotate(area,xy=(e, (y+f)/2))
plt.show()
```

Chapter Project X: Linear System

```
from matplotlib import pyplot as plt
from sympy import *

def draw_axes(x_list,y_list):
```

```
    x_min = min(x_list)
    x_max = max(x_list)
    y_min = min(y_list)
    y_max = max(y_list)
    plt.plot([x_min,x_max],[0,0],c='k')
    plt.plot([0,0],[y_min,y_max],c='k')
    return

# main program
x_list = range(-10,11)

x = Symbol('x')
func1 = sympify(input("Line 1? "))
func2 = sympify(input("Line 2? "))

y1_list = [ ]
y2_list = [ ]

for a in x_list:
    y1 = func1.subs({x:a})     # calculate y1(a)
    y1_list.append(y1)
    y2 = func2.subs({x:a})     # calculate y2(a)
    y2_list.append(y2)

plt.plot(x_list,y1_list)
plt.plot(x_list,y2_list)
draw_axes(x_list,y1_list)

m1,b1 = poly(func1).all_coeffs()
m2,b2 = poly(func2).all_coeffs()

if m1==m2:
    plt.title('Inconsistent')
    if b1==b2:
        plt.title('Consistent Dependent')
else:
    if m1*m2==-1:
        plt.title('Consistent Independent - Perpendicular')
    else:
        plt.title('Consistent Independent')
    equ = str(func1)+'-('+str(func2)+')'
    s = solve(equ)   # returns a list of solutions of func1 = func2
    a = s[0]
```

```
    b = func1.subs({x:a})
    op = '('+str(a)+','+str(b)+')'
    plt.text(a,b,op)
```

Chapter Project XI: Nonlinear System

```
from matplotlib import pyplot as plt
from sympy import *

def draw_axes(x_list,y1_list,y2_list):
    x_min = min(x_list)
    x_max = max(x_list)
    y_min = min(y1_list+y2_list)
    y_max = max(y1_list+y2_list)
    plt.plot([x_min,x_max],[0,0],c='k')
    plt.plot([0,0],[y_min,y_max],c='k')
    return

# main program
x_list = range(-10,11)

x = Symbol('x')
func1 = sympify(input("Function 1? "))
func2 = sympify(input("Function 2? "))

y1_list = [ ]
domain1 = [ ]
for a in x_list:
    y1 = func1.subs({x:a})
    if (not y1.is_imaginary):
        domain1.append(a)
        y1_list.append(y1)
plt.plot(domain1,y1_list)

y2_list = [ ]
domain2 = [ ]
for a in x_list:
    y2 = func2.subs({x:a})
    if (not y2.is_imaginary):
        domain2.append(a)
        y2_list.append(y2)
plt.plot(domain2,y2_list)
```

```
draw_axes(x_list,y1_list,y2_list)


equ = str(func1)+'-('+str(func2)+')'
s = solve(equ)   # returns a list of solutions of func1 = func2
for a in s:
   if (not a.is_imaginary):
      b = func1.subs({x:a})
      if (not b.is_imaginary):
         a = round(float(a),2)
         b = round(float(b),2)
         op = '('+str(a)+','+str(b)+')'
         plt.text(a,b,op)
```