

# STSCI 4060

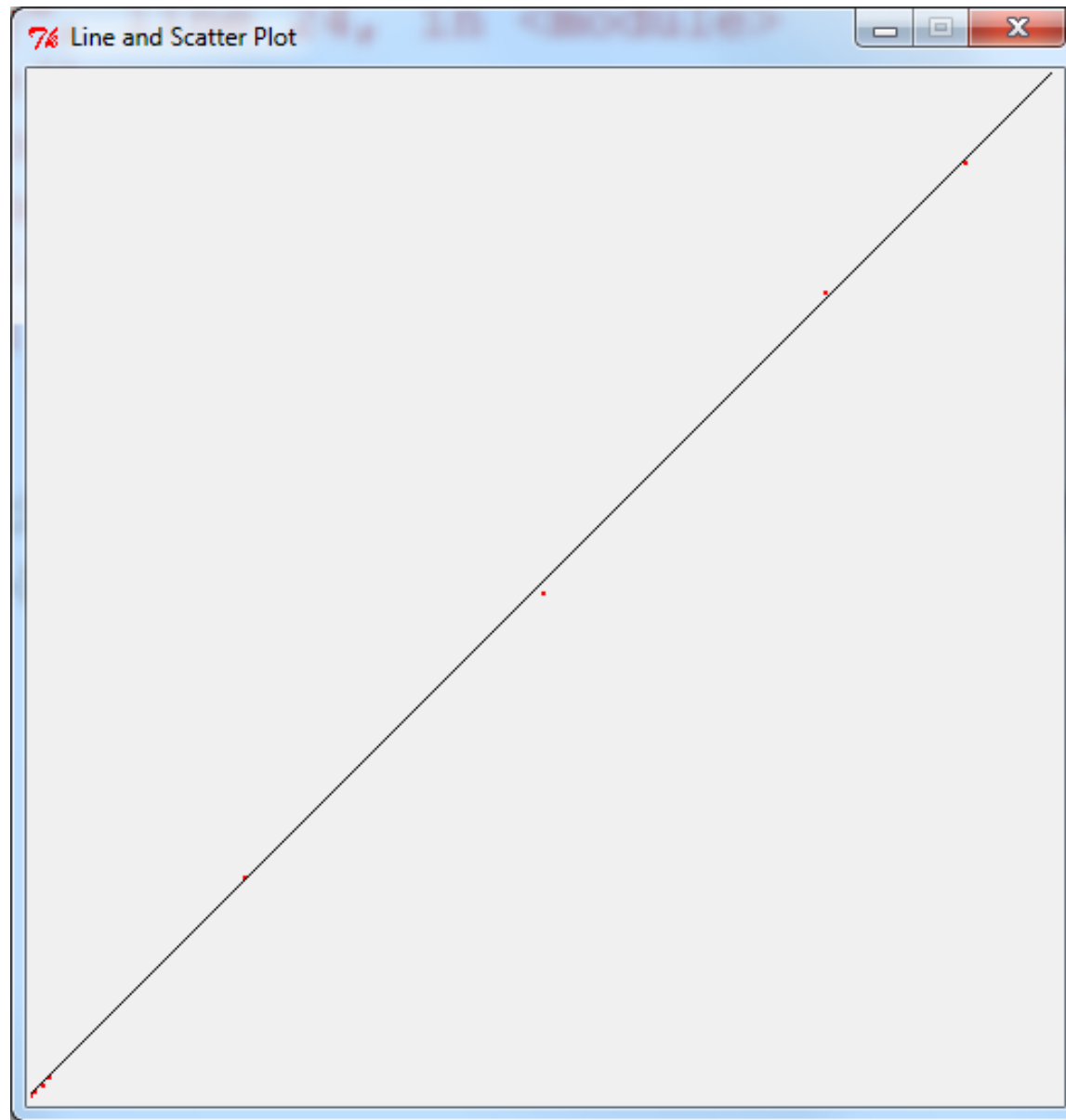
## Lecture File 6

**Xiaolong Yang**  
**(xy44@cornell.edu)**

# More on Python Graphics

- Plotting
- Bar Chart
- Graphical user interface (GUI)
- Linear regression plot
- 3-D plotting with plotting modules

# Draw a Plot



# Draw a Plot

```
#!/usr/bin/env python
#from graphics import *
import graphics
# define a matrix lt holding some coordinate values for plotting
lt=[[0,0], [1,1], [3.5,3.4],[6.6,6.2],[9.9,9.8], [105,107],[250,245],[388,392],[456,455]]

# define a graphic window of size 500x500 pixels
win=graphics.GraphWin('Line and Scatter Plot', 500,500)

# set the coordinates to conventional formats
win.setCoords(0,0,500,500)

num=len(lt) #determine the number of points

# draw the scatter plot
for i in range(num):
    p=graphics.Point(lt[i][0],lt[i][1])
    p.setFill('red')
    p.draw(win)

# draw a diagonal
aline= graphics.Line(graphics.Point(1,1), graphics.Point(499,499))
aline.draw(win)
win.getMouse()
win.close()
```

# Draw a Plot

Read the points from a file

```
from graphics import *
```

```
# read the points from an external file
```

```
infile=file('/Users/xy44/data/points_scatterplot1.txt', 'r')
```

```
win=GraphWin('Scatter Plot', 500,500)
```

```
win.setCoords(0,0,500,500)
```

```
for line in infile:
```

```
    twoNum=line.split(',')
```

```
    p=Point(float(twoNum[0]),float(twoNum[1]))
```

```
    p.setFill('red')
```

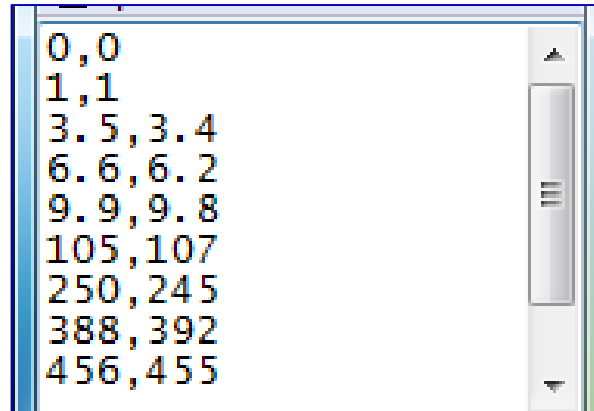
```
    p.draw(win)
```

```
aline=Line(Point(1,1), Point(499,499))
```

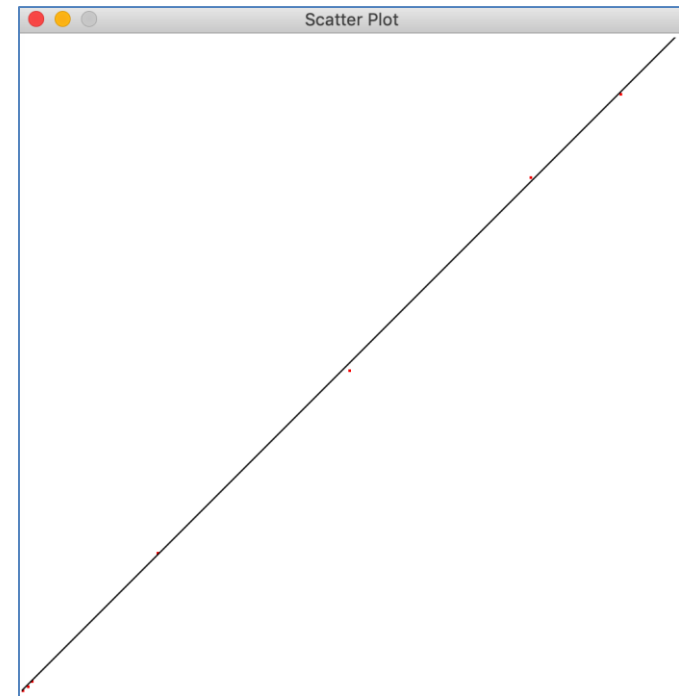
```
aline.draw(win)
```

```
win.getMouse()
```

```
win.close()
```



```
0,0  
1,1  
3.5,3.4  
6.6,6.2  
9.9,9.8  
105,107  
250,245  
388,392  
456,455
```



# Draw a Plot

Read the points from an external file interactively through keyboard

```
from graphics import *
```

```
# read the points from an external file interactively through keyboard
```

```
fileLoc=raw_input('Please input a file name and its location.')
```

```
infile=file(fileLoc)
```

```
win=GraphWin('Scatter Plot', 500,500)
```

```
win.setCoords(0,0,500,500)
```

```
for line in infile:
```

```
    twoNum=line.split(',')
    p=Point(float(twoNum[0]),float(twoNum[1]))
    p.setFill('red')
    p.draw(win)
```

```
    aline=Line(Point(1,1), Point(499,499))
```

```
    aline.draw(win)
```

```
    win.getMouse()
```

```
    win.close()
```

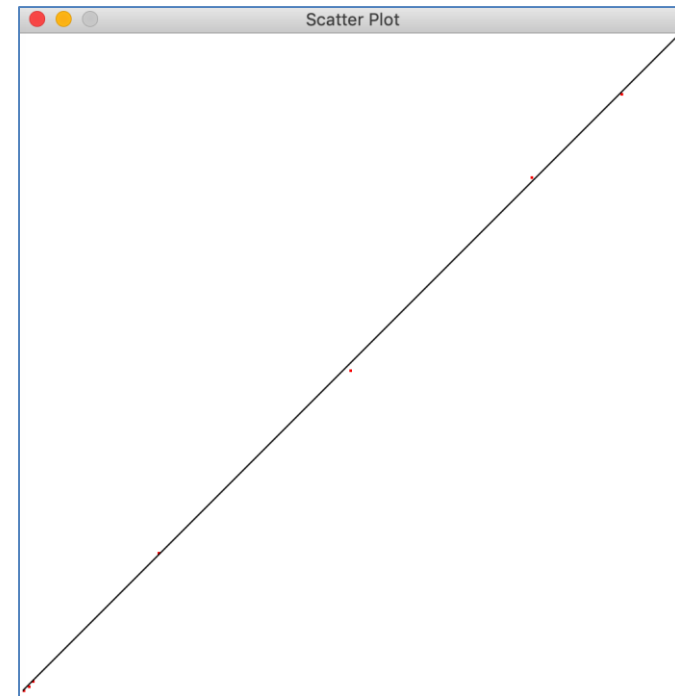
```
aline=Line(Point(1,1), Point(499,499))
```

```
aline.draw(win)
```

```
win.getMouse()
```

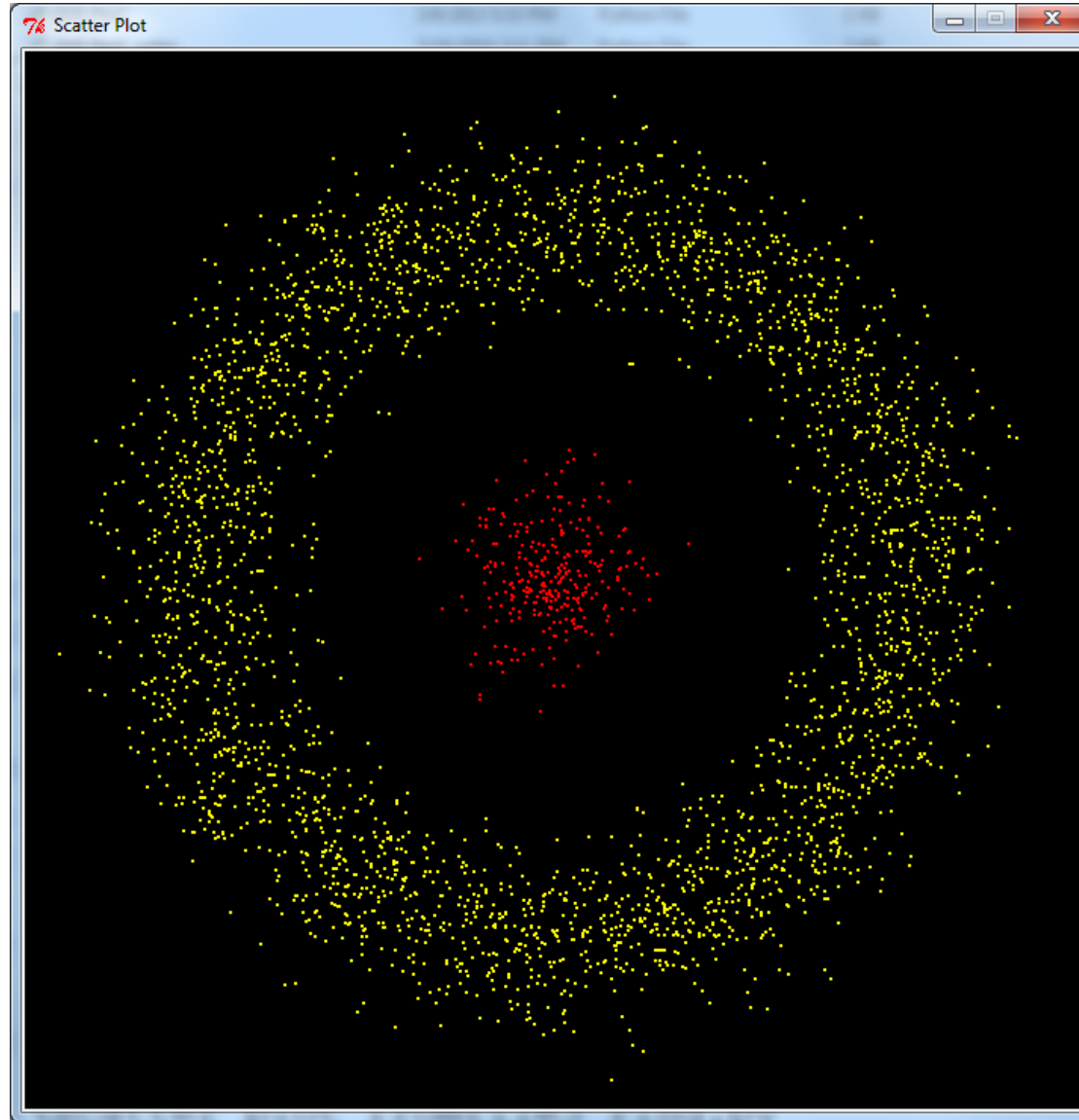
```
win.close()
```

```
0,0
1,1
3.5,3.4
6.6,6.2
9.9,9.8
105,107
250,245
388,392
456,455
```



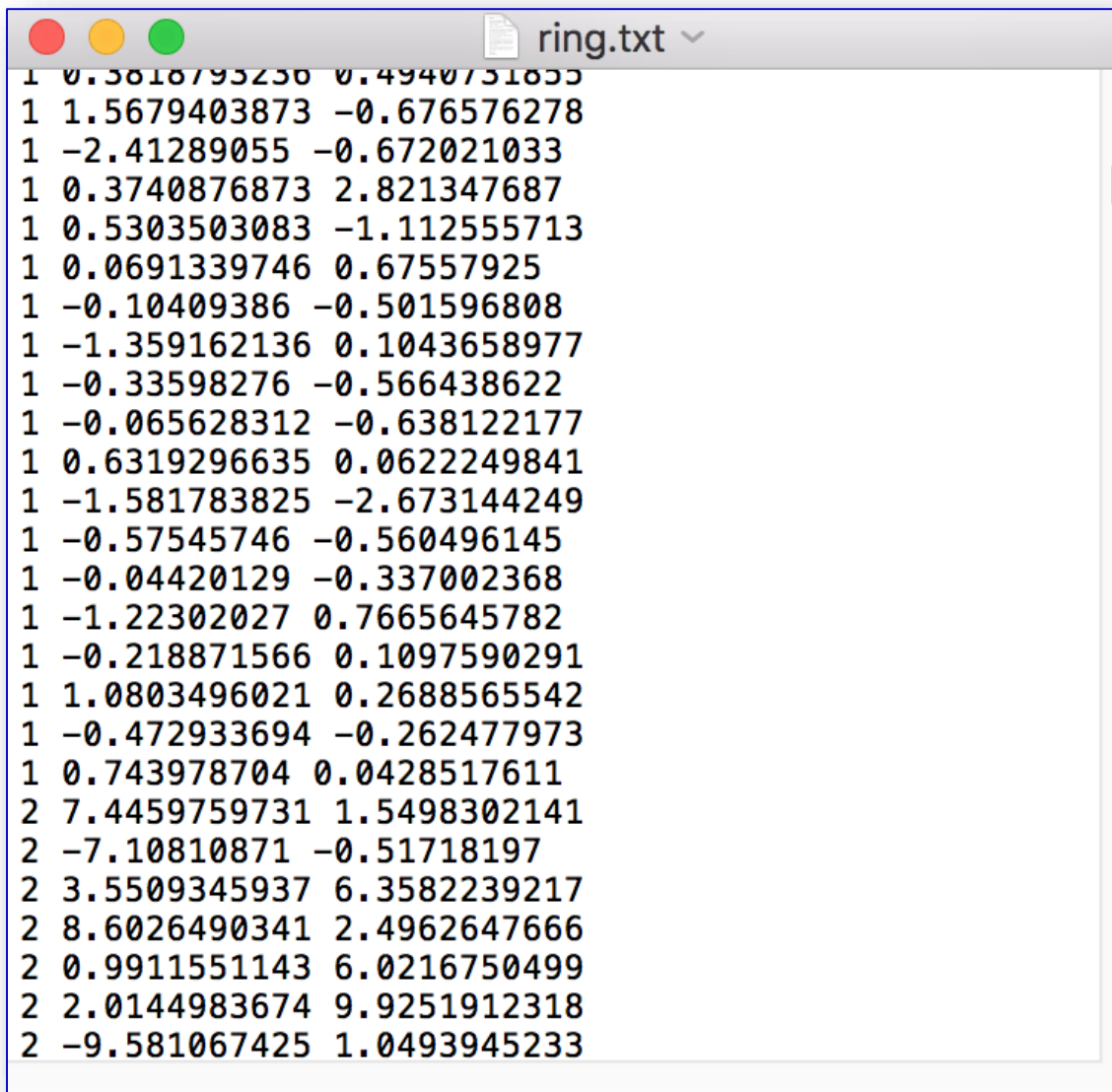
# Draw a Scatter Plot

A scatter plot of two clusters of very unequal variances: the output



# Draw a Scatter Plot

A dataset of two clusters of very unequal variances



```
ring.txt
1 0.5818795238 0.4940731855
1 1.5679403873 -0.676576278
1 -2.41289055 -0.672021033
1 0.3740876873 2.821347687
1 0.5303503083 -1.112555713
1 0.0691339746 0.67557925
1 -0.10409386 -0.501596808
1 -1.359162136 0.1043658977
1 -0.33598276 -0.566438622
1 -0.065628312 -0.638122177
1 0.6319296635 0.0622249841
1 -1.581783825 -2.673144249
1 -0.57545746 -0.560496145
1 -0.04420129 -0.337002368
1 -1.22302027 0.7665645782
1 -0.218871566 0.1097590291
1 1.0803496021 0.2688565542
1 -0.472933694 -0.262477973
1 0.743978704 0.0428517611
2 7.4459759731 1.5498302141
2 -7.10810871 -0.51718197
2 3.5509345937 6.3582239217
2 8.6026490341 2.4962647666
2 0.9911551143 6.0216750499
2 2.0144983674 9.9251912318
2 -9.581067425 1.0493945233
```



# Draw a Scatter Plot

A scatter plot of two clusters of very unequal variances: the code

```
'''
This ptogram draws a scatter plot of two different data sets.
'''

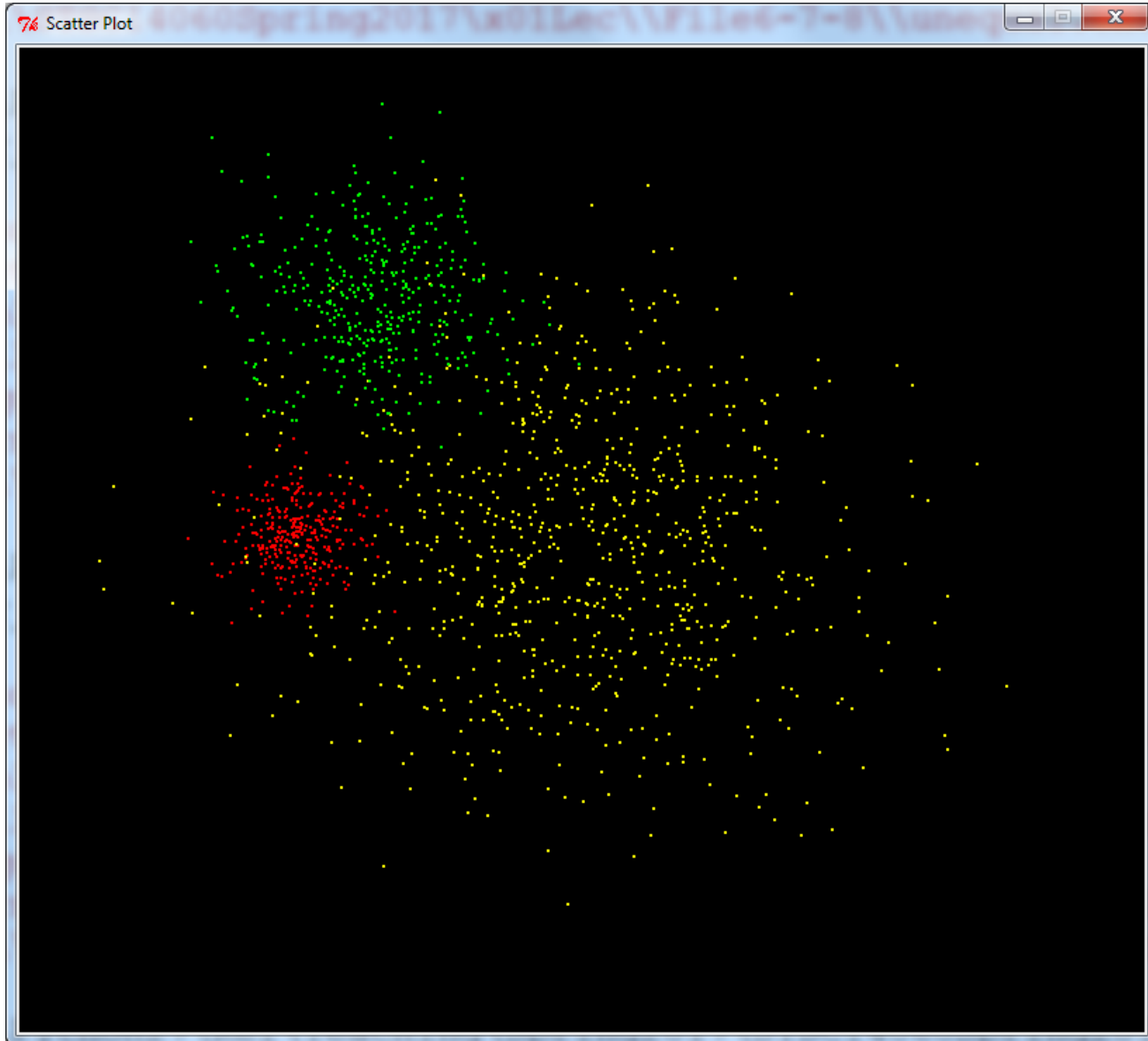
from graphics import *

# read the data points from an external file
infile=file("C:\Python27\Ref\Teaching\Spring2018\\\
STSCI4060Spring2018\ring.txt", 'r')
win=GraphWin('Scatter Plot', 700,700)
win.setCoords(-350,-350,350,350)
win.setBackground('black')
for line in infile:
    twoNum=line.split()
    p=Point(float(twoNum[1])*30,float(twoNum[2])*30)
    if float(twoNum[0])==1:
        p.setFill('red')
    elif float(twoNum[0])==2:
        p.setFill('yellow')
    p.draw(win)

win.getMouse()
win.close()
```

# Draw a Scatter Plot

Another example: a scatter plot of three clusters of unequal variances

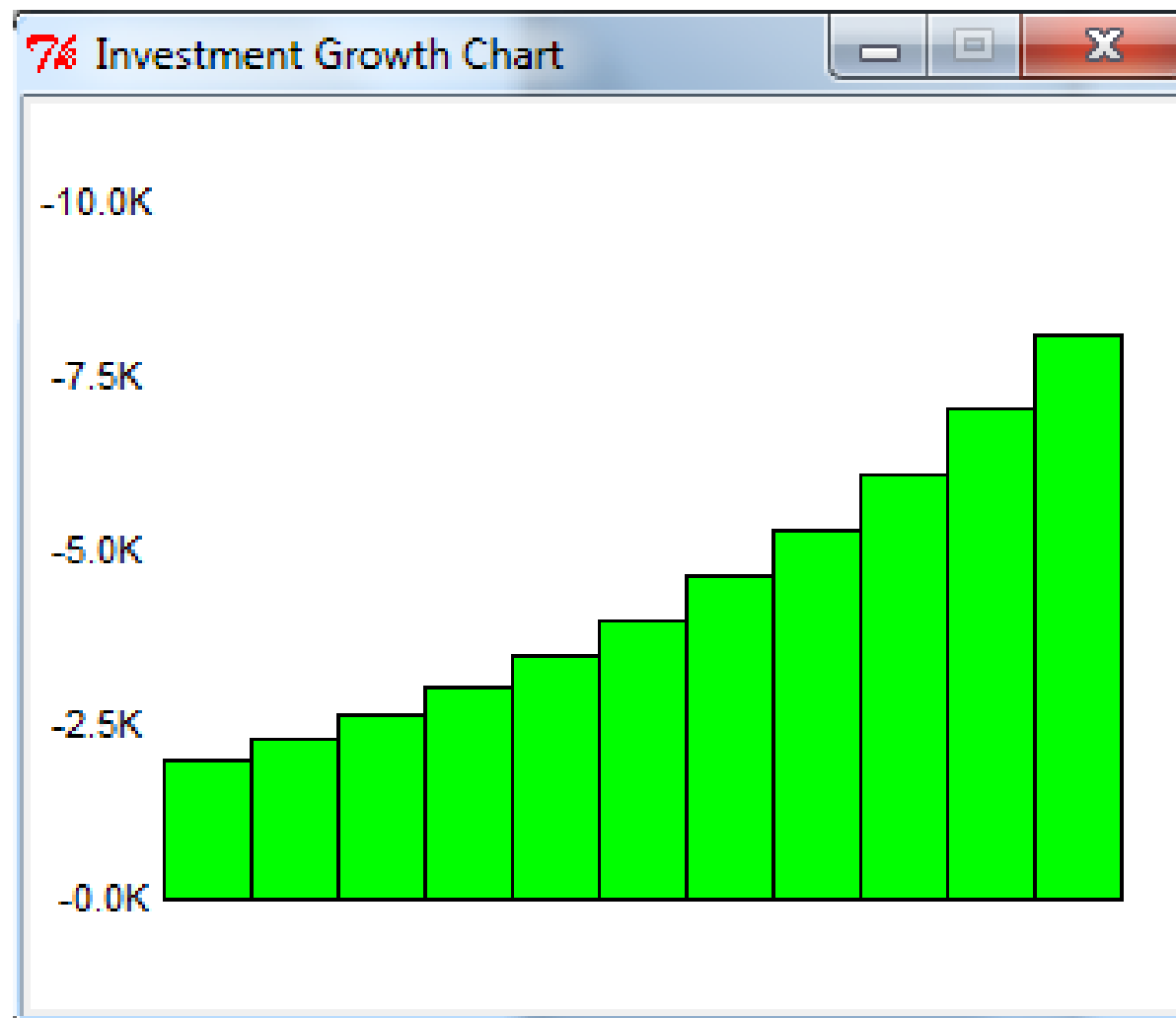


# Draw a Bar Chart of Interest Growth

Plot a Bar Chart of a 10-year investment with the initial principle and the interest rate, annual percentage rate (APR), taken from the keyboard.

# Draw a Bar Chart of Interest Growth

Interest growth chart -- a simple version: the output



# Draw a Bar Chart

Interest growth chart -- a rough version: the code, part 1

```
'''Draw a histogram of investment.'''
from graphics import *

def main():
    print 'Ploting the growth of a 10-year investment.'

    #get principal and interest rate
    principal = input('Enter initial principal: ')
    apr = input('Enter annualized interest rate: ')

    #create a graphic window with labels on the left edge
    win = GraphWin('Investment Growth Chart', 330, 260)
    win.setBackground('white')
    lbl=Text(Point(21,230), ' -0.0K')
    lbl.setSize(8)
    lbl.draw(win)
    lbl1=Text(Point(21,180), ' -2.5K')
    lbl1.setSize(8)
    lbl1.draw(win)
    lbl2=Text(Point(21,130), ' -5.0K')
    lbl2.setSize(8)
    lbl2.draw(win)
    lbl3=Text(Point(21,80), ' -7.5K')
    lbl3.setSize(8)
    lbl3.draw(win)
    lbl4=Text(Point(21,30), ' -10.0K')
    lbl4.setSize(8)
    lbl4.draw(win)
```

# Draw a Bar Chart

Interest growth chart -- a rough version: the code, part 2

```
lbl4=Text(Point(21,30), '-10.0K')
lbl4.setSize(8)
lbl4.draw(win)

#draw bar for initial principal
height = principal*0.02
bar=Rectangle(Point(40, 230), Point(65,230-height))
bar.setFill('green')
bar.draw(win)

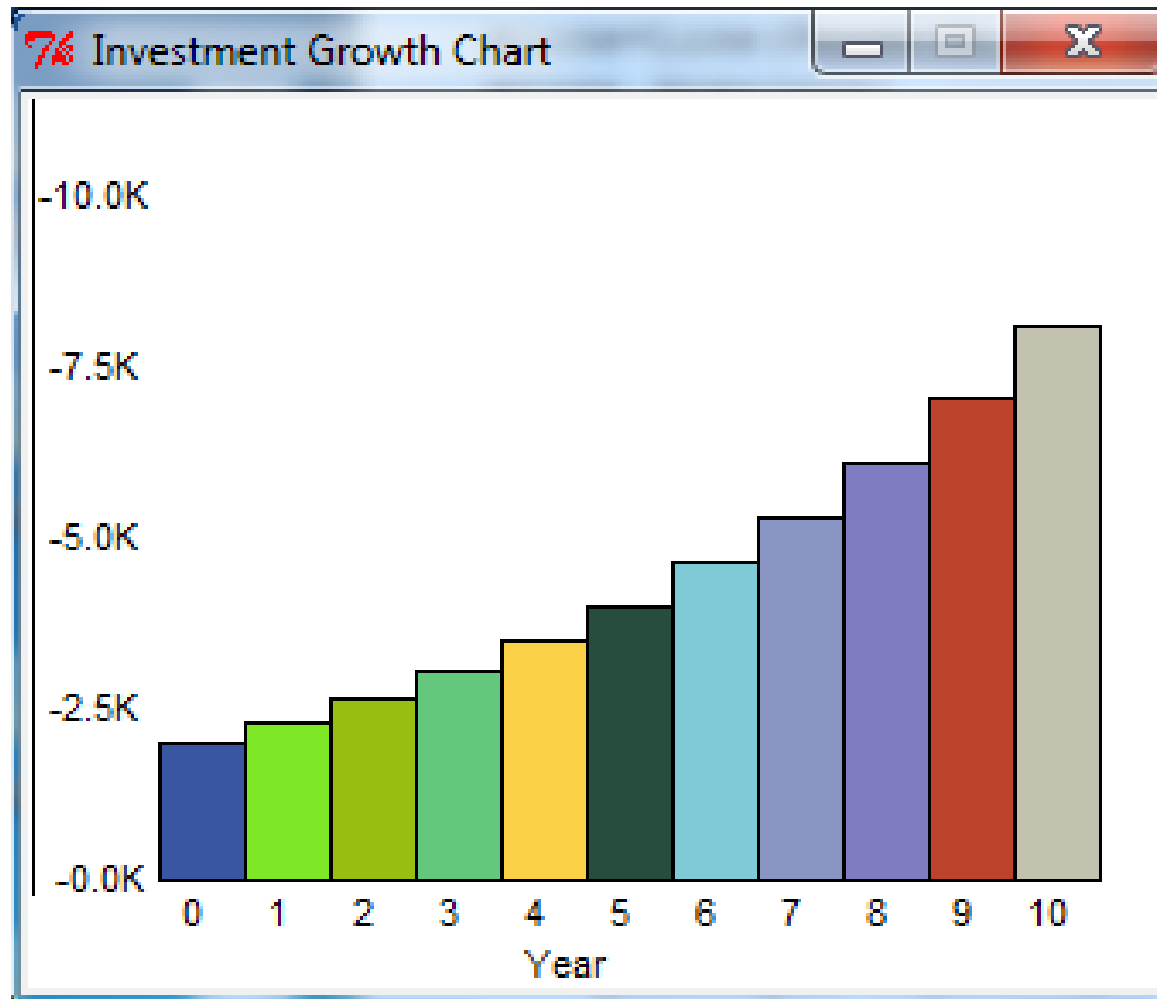
#draw bars for successive years
for year in range(1,11):
    #calculate value for the next year
    principal = principal*(1+apr)
    #draw for this value
    x1=year*25+40
    height=principal*0.02
    bar=Rectangle(Point(x1, 230),Point(x1+25, 230-height))
    bar.setFill('green')
    bar.draw(win)

raw_input('Press <Enter> to quit')
win.close()

main()
```

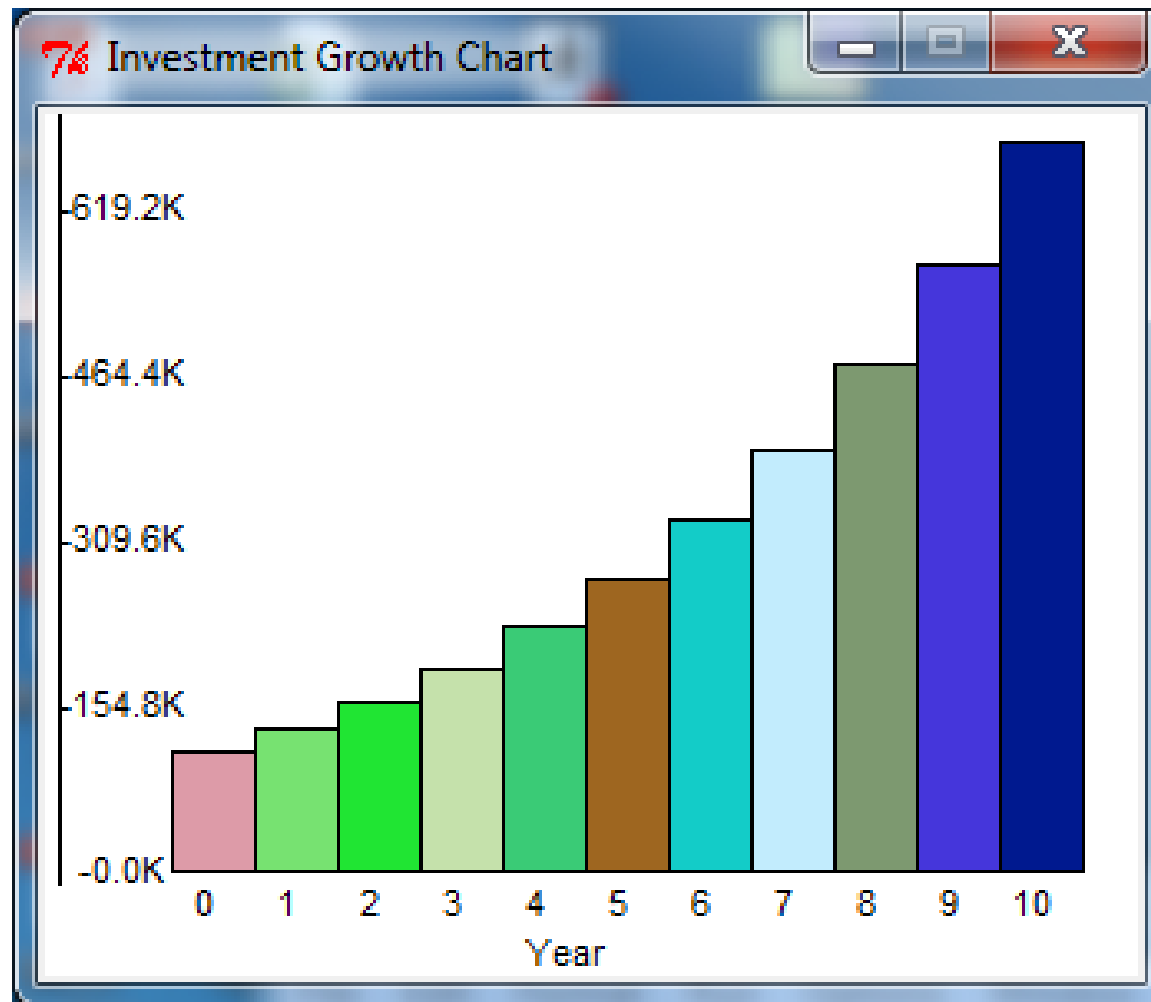
# Draw a Bar Chart

Interest growth chart -- an improved version: the output



# Draw a Bar Chart

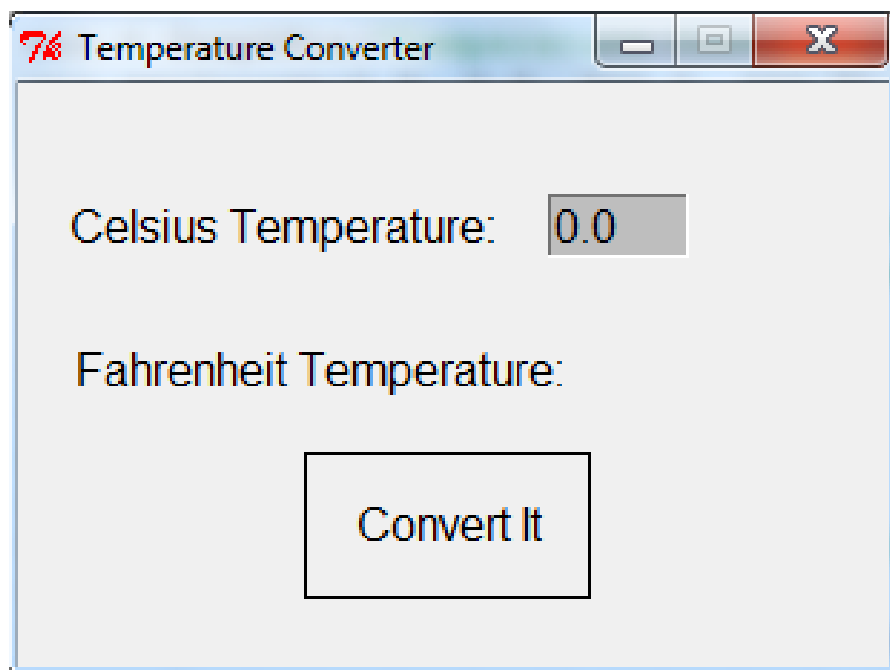
Interest growth chart – a more advanced version: the output





# Graphical User Interface (GUI)

Result window 1



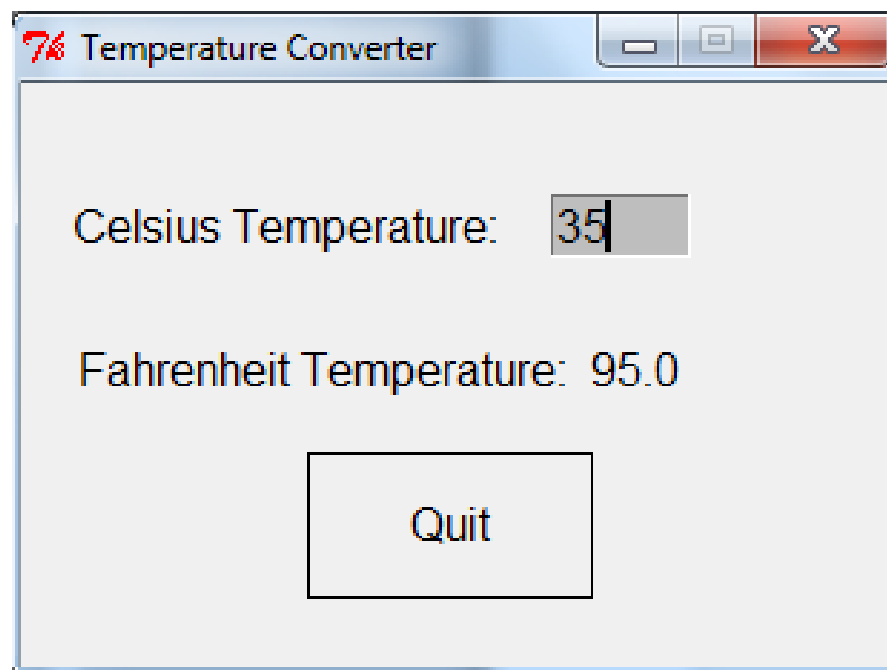
7% Temperature Converter

Celsius Temperature: 0.0

Fahrenheit Temperature:

Convert It

Result window 2



7% Temperature Converter

Celsius Temperature: 35

Fahrenheit Temperature: 95.0

Quit

# Graphical User Interface (GUI)

Get user input from GUI, process the input and display the result

```
''' This program converts Celsius to Fahrenheit using a simple
graphical interface.'''

from graphics import *
def main():
    win = GraphWin("Temperature Converter", 300, 200)
    win.setCoords(0.0, 0.0, 300.0, 200.0)

    # Draw the interface
    Text(Point(90,150), " Celsius Temperature:").draw(win)
    Text(Point(105,100), "Fahrenheit Temperature:").draw(win)
    input = Entry(Point(210,150), 5)
    input.setText("0.0")
    input.draw(win)
    output = Text(Point(215,100), "")
    output.draw(win)
    button = Text(Point(150,45), "Convert It")
    button.draw(win)
    Rectangle(Point(100,20), Point(200,70)).draw(win)

    # wait for a mouse click
    win.getMouse()

    # convert input
    celsius = eval(input.getText())
    fahrenheit = 9.0/5.0 * celsius + 32

    # display output and change button
    output.setText("%0.1f" % fahrenheit)
    button.setText("Quit")

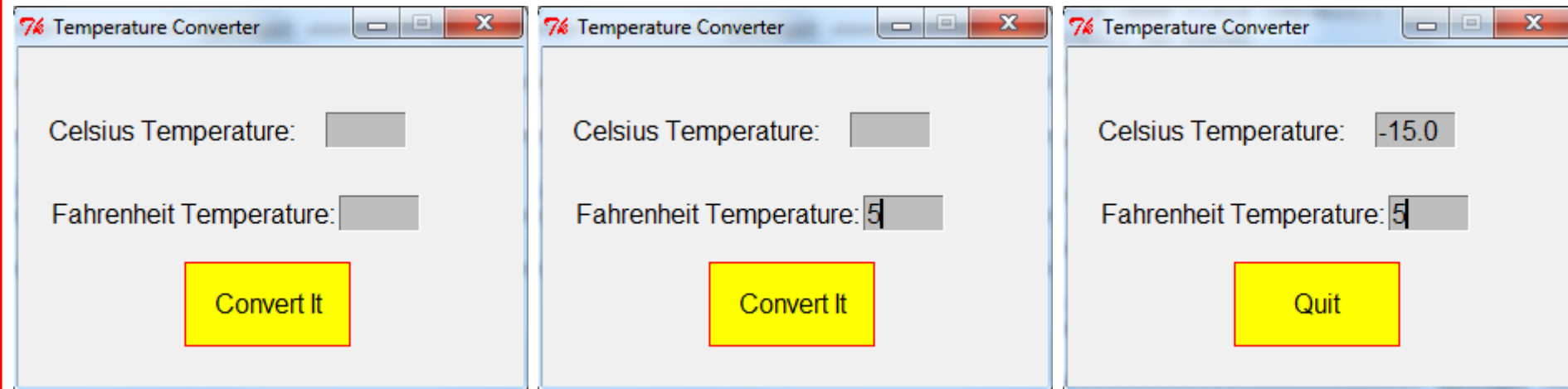
    # wait for click and then quit
    win.getMouse()
    win.close()

main()
```

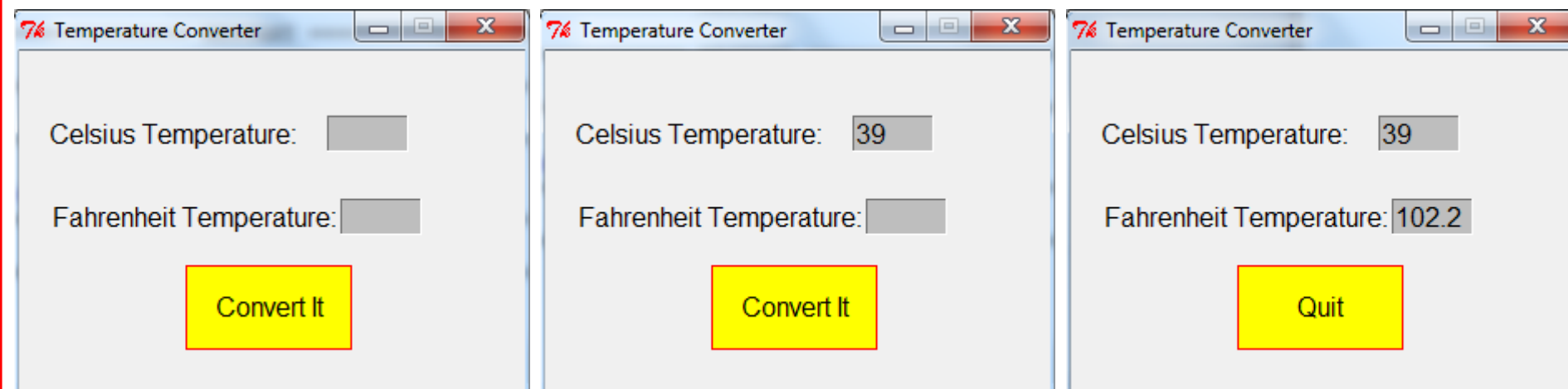
# Graphical User Interface (GUI)

## An improved version

### From Fahrenheit to Celsius



### From Celsius to Fahrenheit

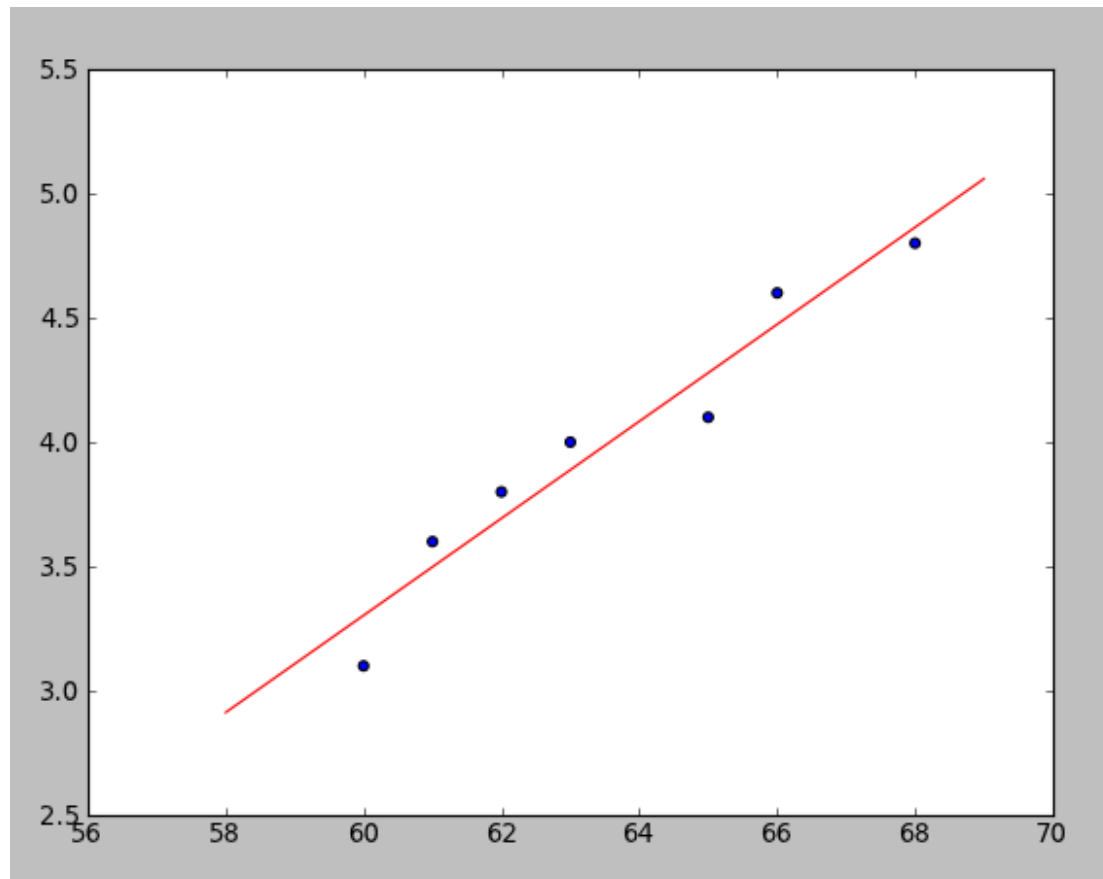


# Code a Simple Linear Regression Analysis

- Input: a list of X values and a list of Y values
- Output:
  - number of observations
  - equation,  $y = a + bx$
- The regression plot

# Simple Linear Regression Program: the result

```
*Python Shell*
File Edit Shell Debug Options Windows Help
>>>
The number of observations: 7
Slope = 0.195114942529
Intercept = -8.40373563218
The equation is:  $y = -8.40373563218 + 0.195114942529 x$ 
Ln: 5 Col: 0
```



# Code a Simple Linear Regression Program I

## Part 1

```
''' This program codes a simple linear regression example.
The equation is  $y = a + bx$ , for which we need to find slope  $b$  and intercept  $a$ .'''

from graphics import *

# data for the regression analysis
X = [55, 98, 150, 200, 256, 308, 480]
Y = [67, 112, 180, 230, 280, 267, 349]

# Step 1: Count the number of values.
N = len(X)
n = len(Y)
if (N <> n):
    raise RuntimeError, "Arrays X and Y have a different number of elements"
print "The number of observations: ", N

# Step 2. Find XY, XX, SX, SY, SXY, and SXX
XY = [0]*N
XX = [0]*N
SX=SY=SXY=SXX=0
for i in range(N):
    XY[i] = X[i] * Y[i]
    XX[i] = X[i] * X[i]
for i in range(N):
    SX = SX + X[i]
    SY = SY + Y[i]
    SXY = SXY + XY[i]
    SXX = SXX + XX[i]
```

## Part 2

```

# Step 3. Calculate the slope: Slope(b) = (N*SXY - (SX)(SY)) / (N*SXX - (SX)2)
Slope = float((N*SXY - (SX * SY))/float((N*SXX - (SX * SX)))
print "Slope = ", Slope

# Step 4. Calcualte the intercept: Intercept(a) = (SY - b(SX)) / N
Intercept = (SY - (Slope * SX)) / N
print "Intercept = ", Intercept
print "The equation is: y = ", Intercept, "+", Slope, "x"

# Step 5. Use the equation to find out the plotting range of x and y
LX = [0]*2
LY = [0]*2
i = 0
for x in (min(X),max(X)):
    LX[i] = float(x)
    LY[i] = Intercept + Slope * x
    i = i + 1

# Step 6. Plot the data points and regression equation
# define a graphic window of size 500x500 pixels and set the coordinates

win=GraphWin('Scat Plot', 500,500)
win.setCoords(0,0,500,500)

# draw the scatter plot
for i in range(N):
    p=Point(X[i],Y[i])
    p.setFill('red')
    p.draw(win)

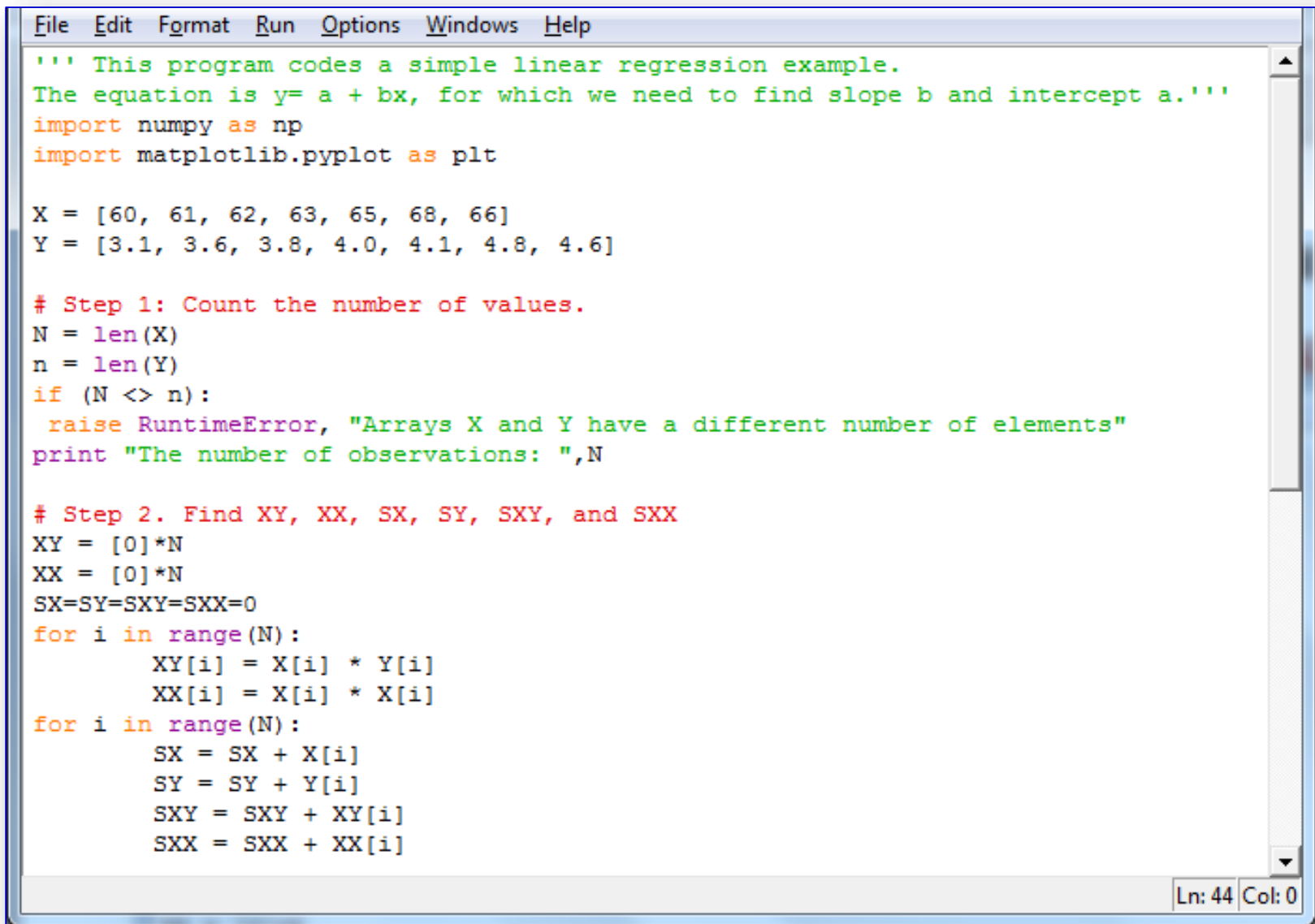
# draw the line
aline= Line(Point(LX[0],LY[0]), Point(LX[1],LY[1]))
aline.draw(win)

raw_input('Press <Enter> to quit')
win.close()

```

# Code a Simple Linear Regression Program II

## Part 1



```
File Edit Format Run Options Windows Help

''' This program codes a simple linear regression example.
The equation is  $y = a + bx$ , for which we need to find slope  $b$  and intercept  $a$ .'''
import numpy as np
import matplotlib.pyplot as plt

X = [60, 61, 62, 63, 65, 68, 66]
Y = [3.1, 3.6, 3.8, 4.0, 4.1, 4.8, 4.6]

# Step 1: Count the number of values.
N = len(X)
n = len(Y)
if (N <> n):
    raise RuntimeError, "Arrays X and Y have a different number of elements"
print "The number of observations: ",N

# Step 2. Find XY, XX, SX, SY, SXY, and SXX
XY = [0]*N
XX = [0]*N
SX=SY=SXY=SXX=0
for i in range(N):
    XY[i] = X[i] * Y[i]
    XX[i] = X[i] * X[i]
for i in range(N):
    SX = SX + X[i]
    SY = SY + Y[i]
    SXY = SXY + XY[i]
    SXX = SXX + XX[i]
```

Ln: 44 Col: 0



# Code a Simple Linear Regression Program II

## Part 2

```
File Edit Format Run Options Windows Help

    SY = SY + Y[i]
    SXY = SXY + XY[i]
    SXX = SXX + XX[i]

# Step 3. Calculate the slope: Slope(b) = (N*SXY - (SX)(SY)) / (N*SXX - (SX)2)
Slope = (N*SXY - (SX * SY)) / (N*SXX - (SX * SX))
print "Slope = ", Slope

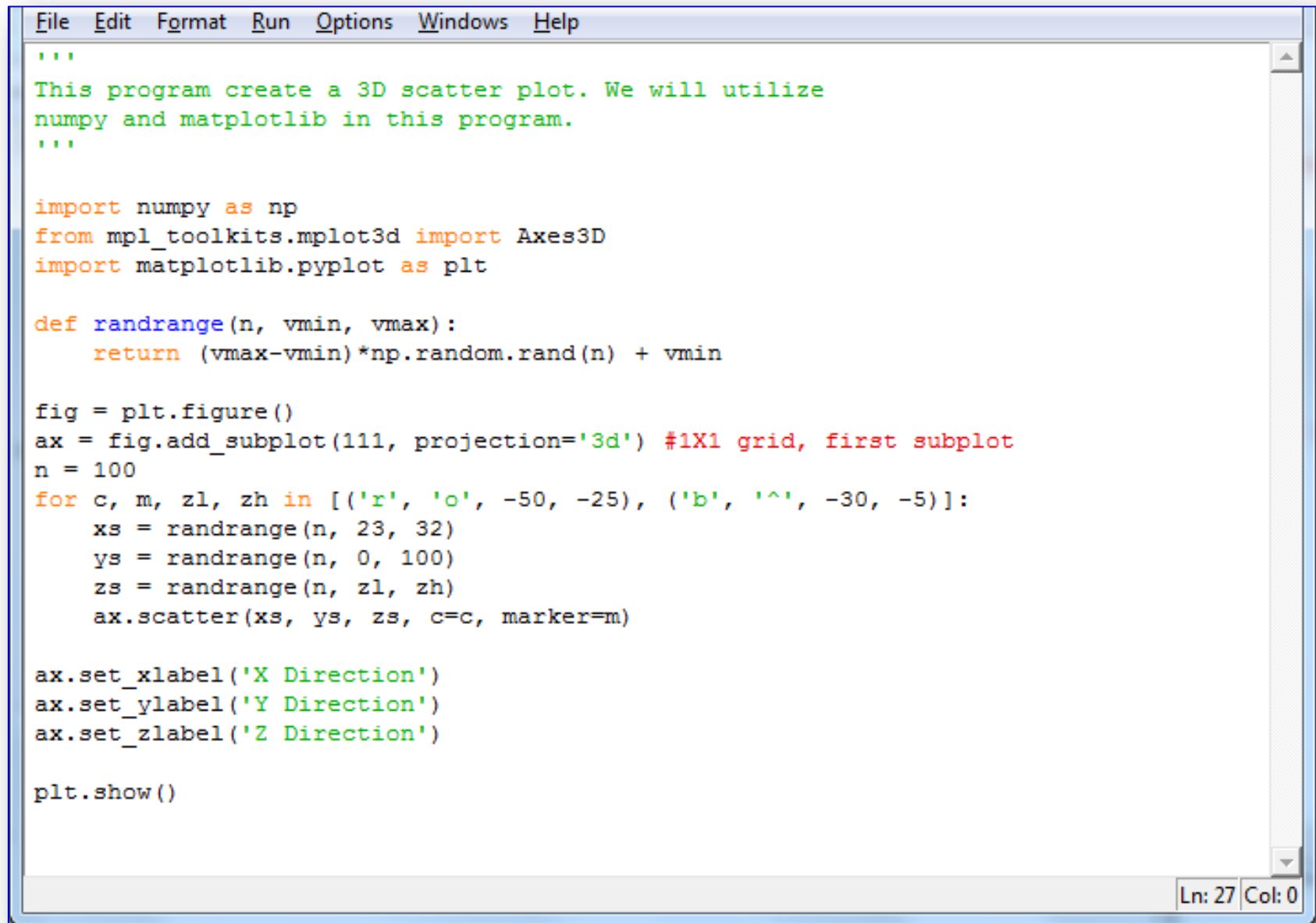
# Step 4. Calcualte the intercept: Intercept(a) = (SY - b(SX)) / N
Intercept = (SY - (Slope * SX)) / N
print "Intercept = ", Intercept
print "The equation is: y = ", Intercept, "+", Slope, "x"

# Step 5. Use the equation to find out the plotting range of x and y
LX = [0]*12
LY = [0]*12
i = 0
for x in range(58,70,1):
    LX[i] = float(x)
    LY[i] = Intercept + Slope * x
    i = i + 1

# Step 6. Plot the data points and regression equation
plt.plot(LX, LY, 'r')
plt.scatter(X, Y)
plt.show()
```

Ln: 38 Col: 0

# A 3D-Scatter Plot Program

A screenshot of a Python IDE window with a menu bar (File, Edit, Format, Run, Options, Windows, Help) and a code editor. The code is a Python script to create a 3D scatter plot using numpy and matplotlib. It includes a docstring, imports for numpy, mpl\_toolkits.mplot3d, and matplotlib.pyplot. A custom function 'randrange' is defined to generate random values. The main code creates a 3D subplot, generates 100 data points with two different colors and markers, and displays the plot. The status bar at the bottom right shows 'Ln: 27 Col: 0'.

```
File Edit Format Run Options Windows Help

'''
This program create a 3D scatter plot. We will utilize
numpy and matplotlib in this program.
'''

import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

def randrange(n, vmin, vmax):
    return (vmax-vmin)*np.random.rand(n) + vmin

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d') #1X1 grid, first subplot
n = 100
for c, m, zl, zh in [('r', 'o', -50, -25), ('b', '^', -30, -5)]:
    xs = randrange(n, 23, 32)
    ys = randrange(n, 0, 100)
    zs = randrange(n, zl, zh)
    ax.scatter(xs, ys, zs, c=c, marker=m)

ax.set_xlabel('X Direction')
ax.set_ylabel('Y Direction')
ax.set_zlabel('Z Direction')

plt.show()

Ln: 27 Col: 0
```

# A 3D-Scatter Plot Program: the result

