#-------------------------------------------------------------------------------  
# Name:        ball  
# Purpose:  
#  
# Author:      Clarissa Joyce  
#  
# Created:     17/09/2014  
#-------------------------------------------------------------------------------  
#!/usr/bin/env python  
  
from numpy import \*  
import matplotlib.pyplot as plt  
import numpy as np  
  
v0 = 20.0  
g = 9.81  
  
t=np.linspace(-50,50)  
y=(v0\*t)-(g\*(t\*\*2)\*0.5)  
  
max\_y = max(y)  # Find the maximum y value  
max\_x = t[y.argmax()]  # Find the x value corresponding to the maximum y value  
print "Maximum height: ",max\_y, " m"  
  
plt.xlabel("time (s)")  
plt.ylabel("position (m)")  
plt.plot(t, y, 'ro')  
plt.show()

#-------------------------------------------------------------------------------

# Name: ball2

# Purpose:

#

# Author: Clarissa Joyce

#

# Created: 17/09/2014

#-------------------------------------------------------------------------------

#!/usr/bin/env python

from numpy import \*

import matplotlib.pyplot as plt

import numpy as np

def y(t):

return v0\*t-0.5\*g\*t\*\*2

v0 = [20.0,5.0,15.0,25.0]

g = 9.81

t=np.linspace(-50,50)

y=(v0[0]\*t)-(g\*(t\*\*2)\*0.5)

max\_y = max(y)

max\_x = t[y.argmax()]

print "Maximum height, V0 = 20.0: ",max\_y, " m"

plt.xlabel("time (s)")

plt.ylabel("position (m)")

plt.plot(t, y, 'ro')

# V0 = 5.0

y1=(v0[1]\*t)-(g\*(t\*\*2)\*0.5)

max\_y1 = max(y) # Find the maximum y value

max\_x1 = t[y.argmax()] # Find the x value corresponding to the maximum y value

print "Maximum height, V0 = 5.0: ",max\_y1, " m"

plt.plot(t, y1, 'b--')

# V0 = 15.0

y2=(v0[2]\*t)-(g\*(t\*\*2)\*0.5)

max\_y2 = max(y) # Find the maximum y value

max\_x2 = t[y.argmax()] # Find the x value corresponding to the maximum y value

print "Maximum height, V0 = 15.0: ",max\_y2, " m"

plt.plot(t, y2, 'yo')

# V0 = 25.0

y3=(v0[3]\*t)-(g\*(t\*\*2)\*0.5)

max\_y3 = max(y) # Find the maximum y value

max\_x3 = t[y.argmax()] # Find the x value corresponding to the maximum y value

print "Maximum height, V0 = 25.0: ",max\_y3, " m"

plt.plot(t, y3, 'g--')

plt.show()

#-------------------------------------------------------------------------------

# Name: normal

# Purpose:

#

# Author: Clarissa Joyce

#

# Created: 17/09/2014

#-------------------------------------------------------------------------------

#!/usr/bin/env python

from numpy import \*

import matplotlib.pyplot as plt

import numpy as np

#sigma describes the width of the distribution

mean=0

x=range(-5,6,1)

sigma = [0.5,1.0,2.0]

p1=[]

p2=[]

p3=[]

for j in x:

G=(1.0/(sigma[0]\*((2.\*pi)\*\*(0.5))))\*(np.e\*\*(-(x[j]-mean)\*\*2/(2.0\*sigma[0]\*\*2.)))

G=-math.log10(G)

p1.append(G)

for j in x:

G=(1.0/(sigma[1]\*((2.\*pi)\*\*(0.5))))\*(np.e\*\*(-(x[j]-mean)\*\*2/(2.0\*sigma[1]\*\*2.)))

G=-math.log10(G)

p2.append(G)

for j in x:

G=(1.0/(sigma[2]\*((2.\*pi)\*\*(0.5))))\*(np.e\*\*(-(x[j]-mean)\*\*2/(2.0\*sigma[2]\*\*2.)))

G=-math.log10(G)

p3.append(G)

plt.xlabel("x/sigma")

plt.ylabel("G(X) (log10 scaled)")

plt.plot(x,p1,"r--")

plt.plot(x,p2,"b--")

plt.plot(x,p3,"y--")

plt.show()

#-------------------------------------------------------------------------------

# Name: geiger

# Purpose:

#

# Author: Clarissa Joyce

#

# Created: 17/09/2014

#-------------------------------------------------------------------------------

#!/usr/bin/env python

import numpy as np

import matplotlib.pyplot as plt

data = [10,12,15,8,13,14,19,18,11,13,7,8,11,8,12,6,13,8,6]

mean = np.mean(data)

sigma=np.std(data)

difference = abs(sigma - mean\*\*0.5)

if (difference<1):

print "The values of sigma and the square root of the mean are close"

else:

print "The values of sigma and the square root of the mean are not close"

num\_bins=100

plt.hist(data,num\_bins,facecolor='green')

plt.show()

print "Mean=",mean, ", Sigma=",sigma, ", Difference= ",difference

#-------------------------------------------------------------------------------

# Name: spring

# Purpose:

#

# Author: Clarissa Joyce

#

# Created: 17/09/2014

#-------------------------------------------------------------------------------

#!/usr/bin/env python

from numpy import \*

from math import \*

import matplotlib.pyplot as plt

import numpy as np

m = np.arange(0.2,1.0,0.1)

y = [4.9,5.3,5.7,6.7,7.2,7.5,8.4,9.2]

g=9.8

#m[0]\*g=k\*(y-y0)

# initialize sums

sum\_m = 0

sum\_y = 0

sum\_mm = 0

sum\_my = 0

mmin = 0

#perform our calculations

for i in range(0,len(m)):

sum\_m = sum\_m+m[i-1]

sum\_y = sum\_y+y[i-1]

mm = m[i-1]\*m[i-1]

sum\_mm = sum\_mm +mm

my = m[i-1]\*y[i-1]

sum\_my = sum\_my+my

#calculate the coefficients

D = len(m)\*sum\_mm - sum\_m\*sum\_m

A= (sum\_mm\*sum\_y - sum\_m\*sum\_my)/D

B= (len(m)\*sum\_my-sum\_m\*sum\_y)/D

#plot data points

plt.plot(m,y,"bo")

#plot least squares fit line

xc= linspace(mmin,m[len(m)-1],10)

yc= A+B\*xc

y0 = A+B

k= m\*g/(y-y0)

print "y0 = ",y0,", k = ",k

plt.plot(xc,yc,"r-")

plt.show()

#-------------------------------------------------------------------------------

# Name: karate

# Purpose:

#

# Author: Clarissa Joyce

#

# Created: 17/09/2014

#-------------------------------------------------------------------------------

#!/usr/bin/env python

import numpy as np

from math import \*

import matplotlib.pyplot as plt

data = np.loadtxt("Karate.txt")

t=data[:,0]

p=data[:,1]

print len(t)

v=[]

a=[]

d=[]

for i in range(0,len(p)-1):

n=p[i+1]-p[i]

d.append(n)

v.append(0)

for i in range(0,len(t)-1):

n=d[i]/t[i]

v.append(n)

#print len(v)

a.append(0)

for i in range(0,len(v)-1):

n=(v[i+1]-v[i])/2

a.append(n)

#print len(a)

m=max(v)

print "The max velocity of the hand is: ",m

mvp = p[v.index(max(v))]

print "The hand is at ",mvp

ma = max(a)

g=ma/9.8

print "The max acceleration of the hand is: ",ma,"or ",g,"g's"

mxp = p[a.index(max(a))]

print "The hand is at ",mxp

np.savetxt("karate2.txt", np.column\_stack((v,a)),delimiter="," )#, header='Velocity, Acceleration')

plt.xlabel("time (s)")

plt.ylabel("position(m), velocity(m/s), acceleration(m/s^2")

plt.plot(p,t,"bo")

plt.plot(v,t,"ro")

plt.plot(a,t,"g\*")

plt.show()