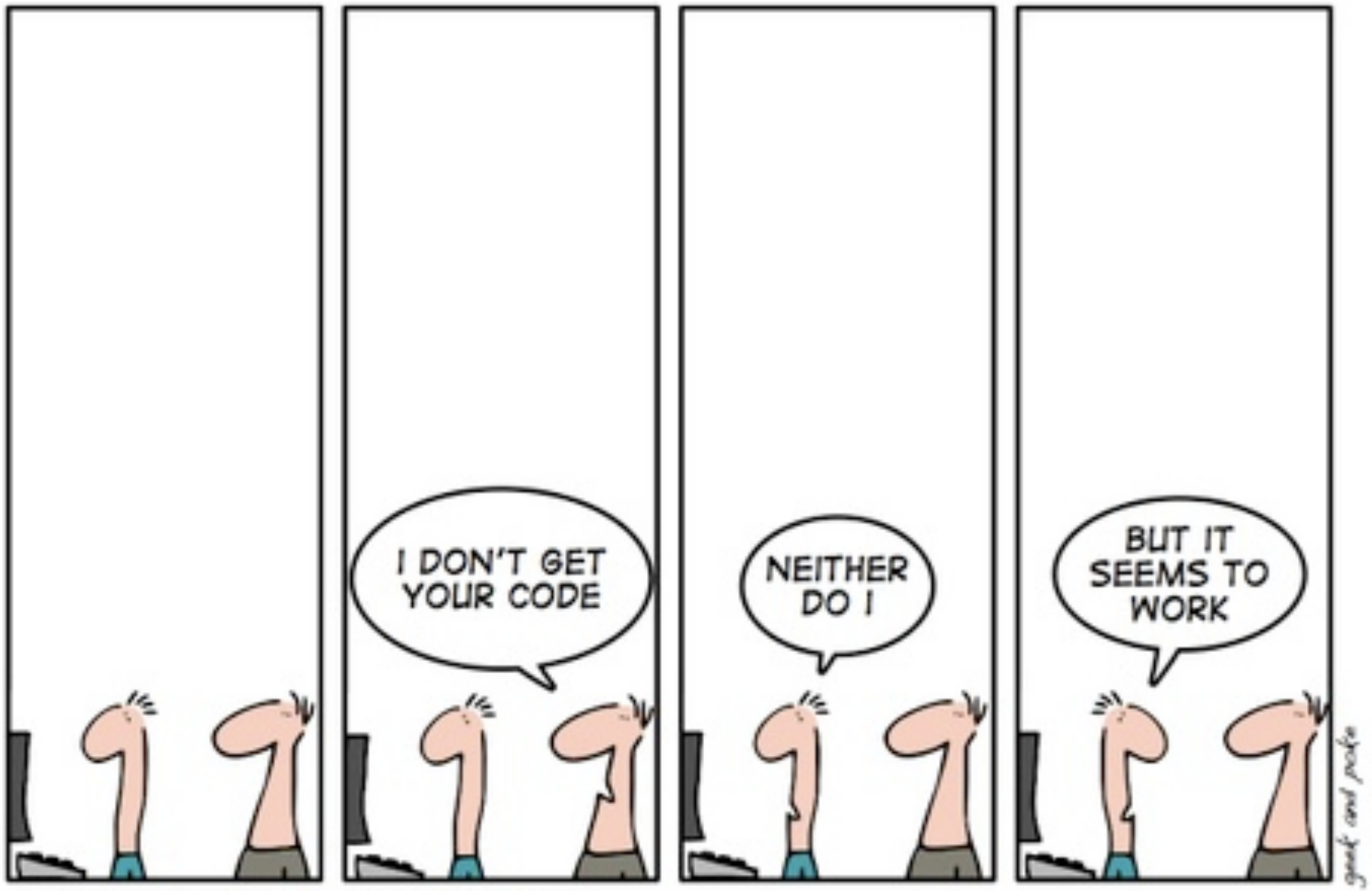
Python
- plotting using matplotlib



Python 2 print

In [1]:
```
a=0.13456789
b="hello"
```

In [2]:
```
print a
```

```
0.13456789
```

In [3]:
```
print (a) # do not use this because in a python program it prints "(a)"
```

```
0.13456789
```

In [4]:
```
print "%.2f" % a
```

```
0.13
```

In [5]:
```
print "{} {}".format(a,b)
```

```
0.13456789 hello
```

In [7]:
```
print "{1} {0}".format(a,b)
```

```
hello 0.13456789
```

In [13]:
```
print "|{}|".format(a)
print "|{:.2}|".format(a)
print "|{:>10}|".format(b)
print "|{:<10}|".format(b)
print "|{:^10}|".format(b)
```

```
|0.13456789|
|0.13|
|     hello|
|hello     |
|  hello   |
```

In [15]:
```
from __future__ import print_function #now with python3 syntax
```

In [16]:
```
print a
```

```
  File "<ipython-input-16-da1608c9d425>", line 1
    print a
          ^
SyntaxError: invalid syntax
```

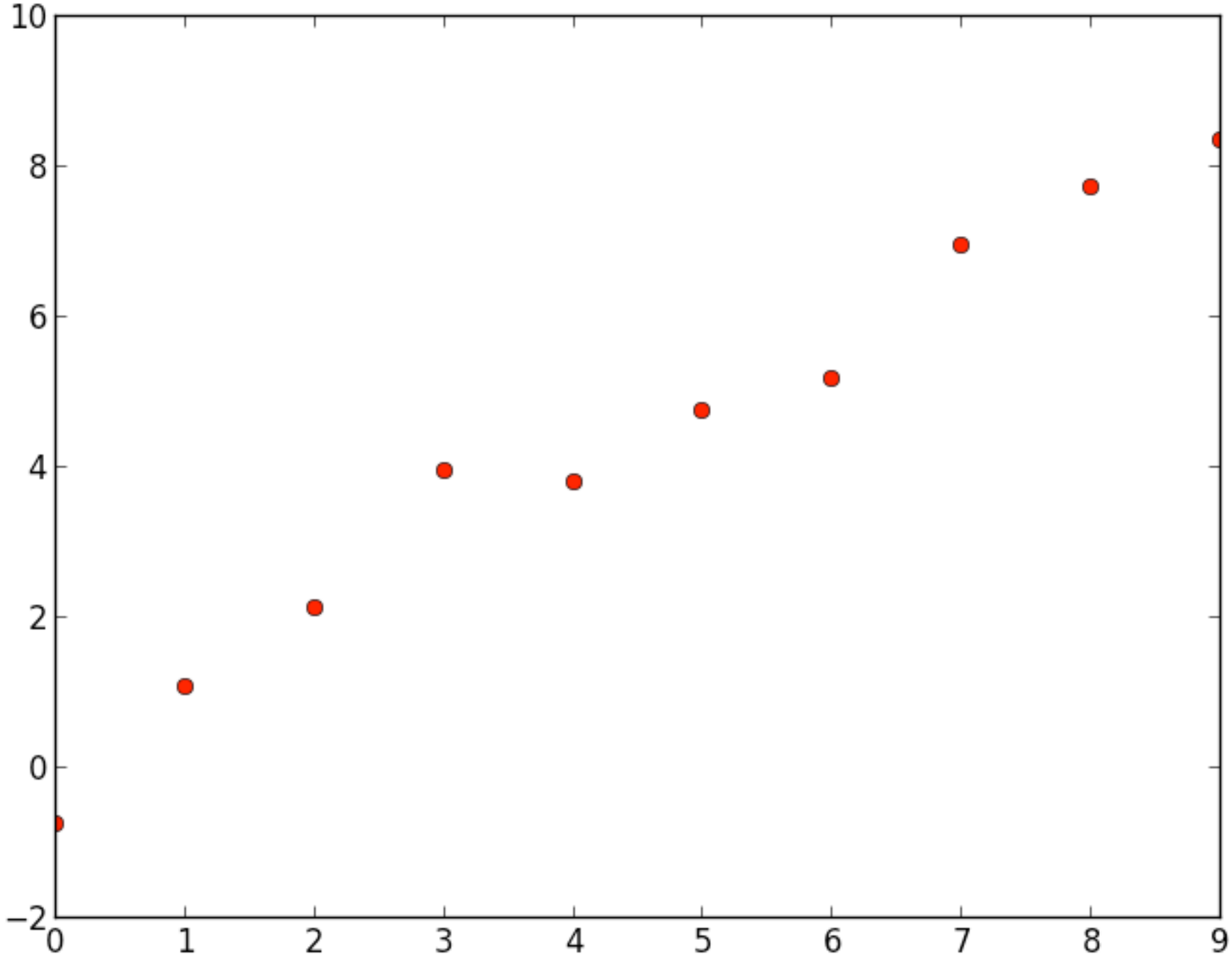
In [17]:
```
print(a)
```

```
0.13456789
```

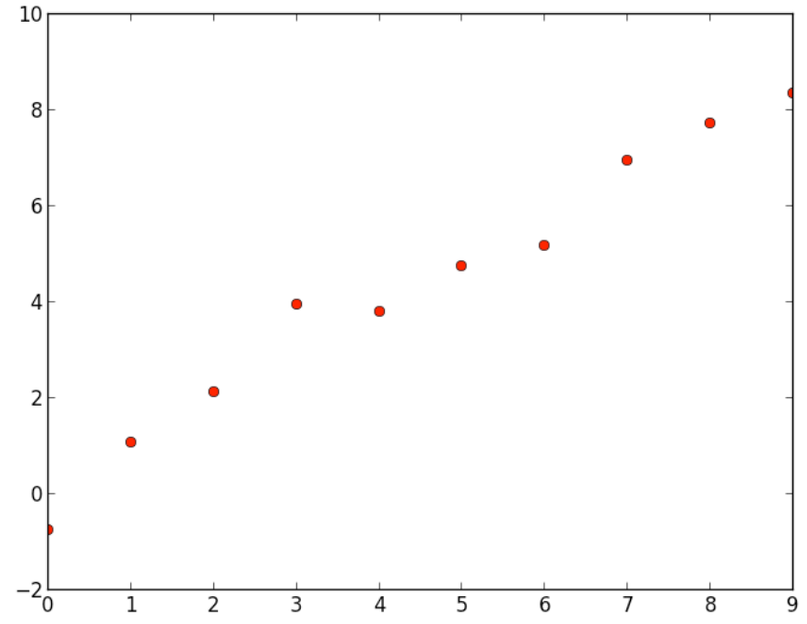
In [20]:
```
print("|{}|".format(a),end=' ')
print ("|{:.2}|".format(a),end='\n')
print ("|{:>10}|".format(b),end='@')
print ("|{:<10}|".format(b),end='@')
print ("|{:^10}|".format(b))
```

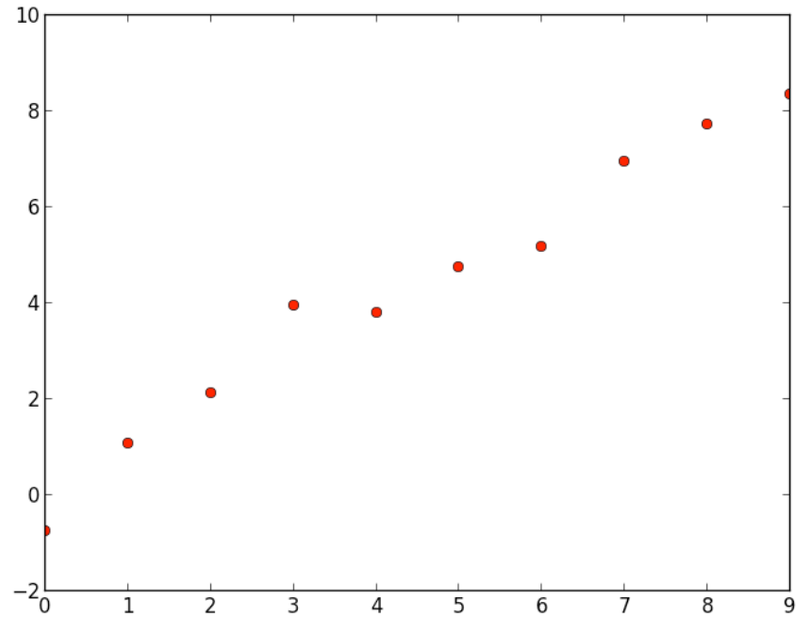
```
|0.13456789| |0.13|
|     hello|@|hello     |@|  hello    |
```

https://pyformat.info

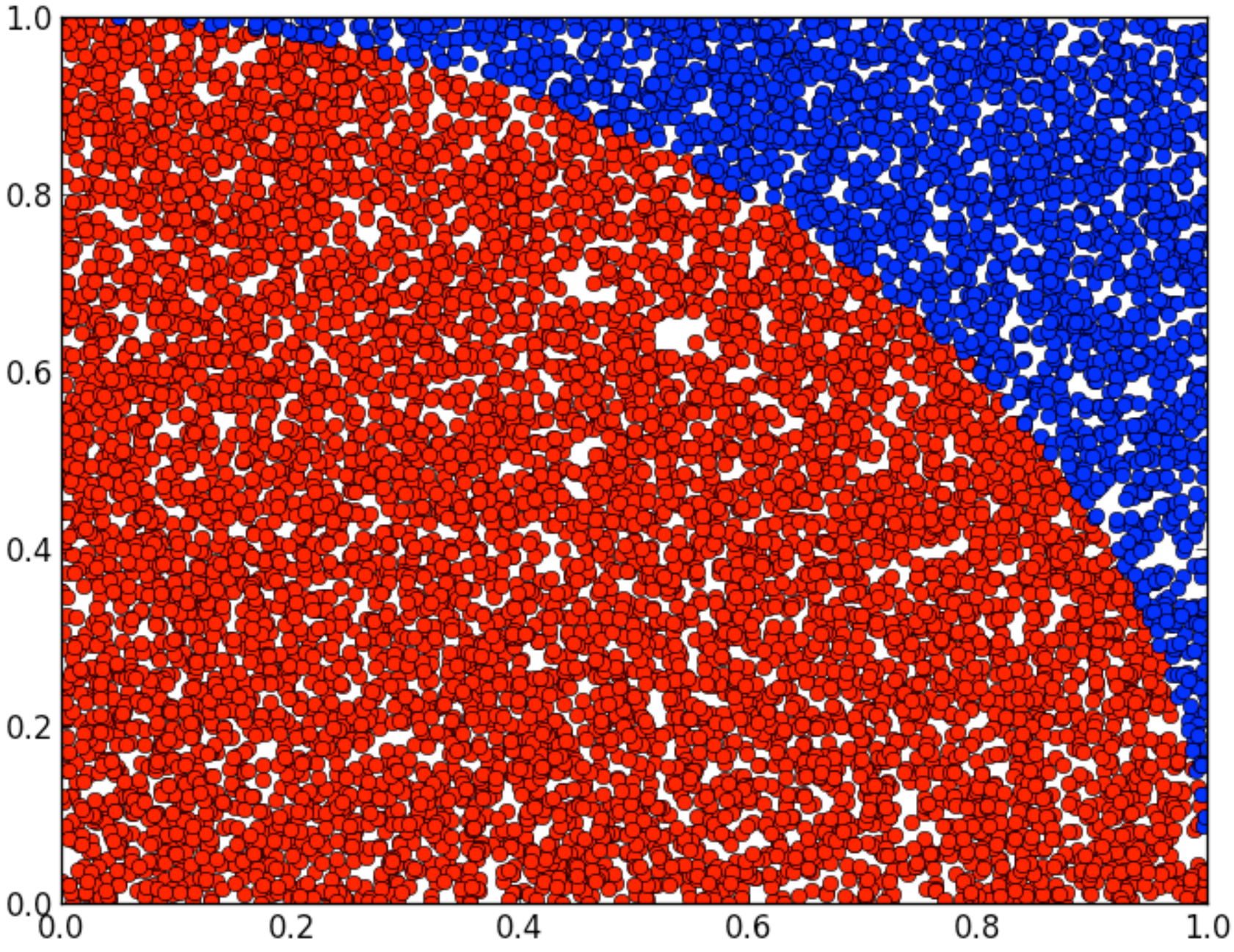

10
8
6
4
2
0
−2
0
1
2
3
4
5
6
7
8
9

```
#!/usr/bin/env python
import sys
import matplotlib
import matplotlib.pyplot as plt
import random
x = range(10)
y = [xi + random.uniform(-1.0,1.0) for xi in x]
plt.figure()
plt.plot(x,y,'ro')
plt.show()
```



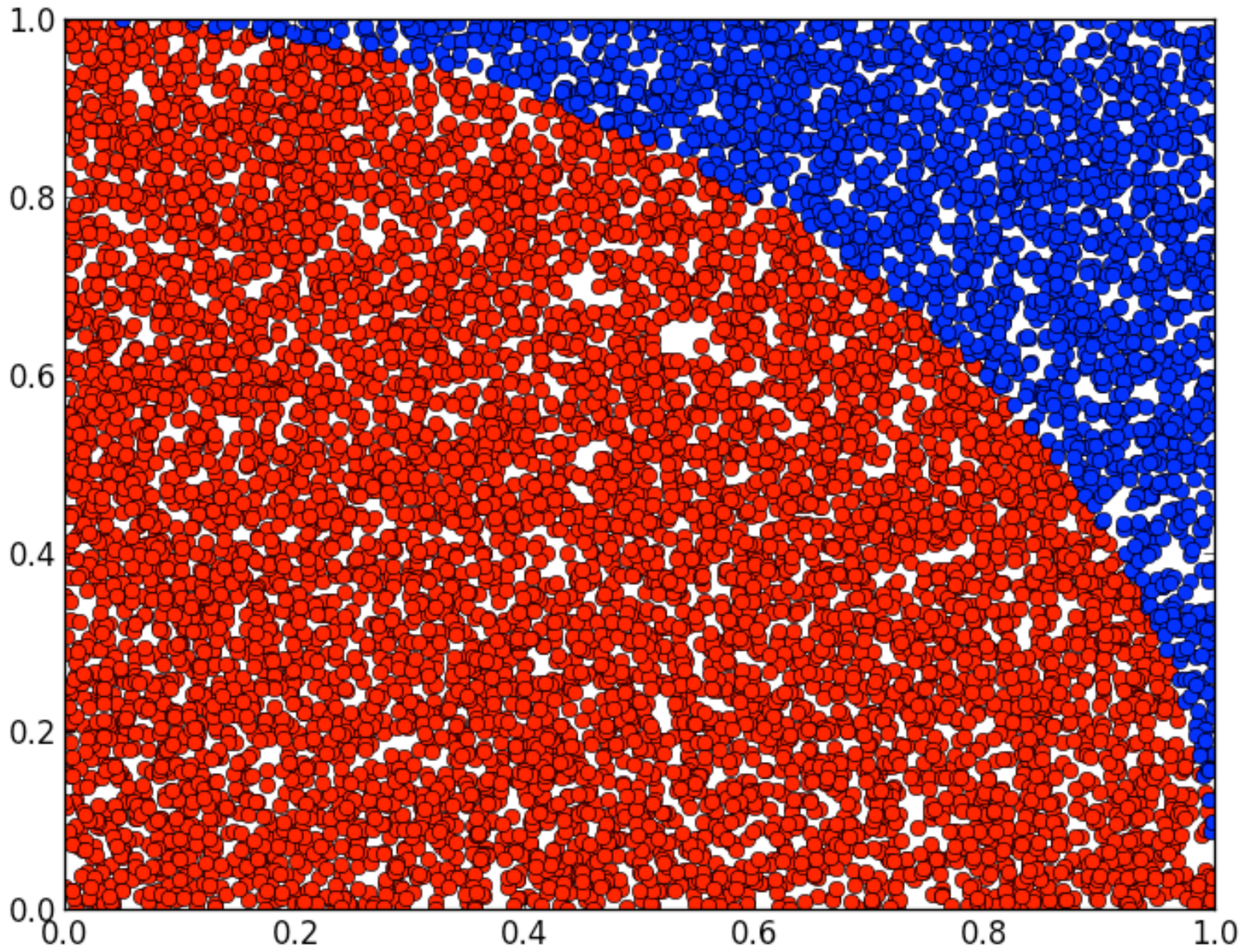



```
#!/usr/bin/env python
import sys
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfFile
import random
x = range(10)
y = [xi + random.uniform(-1.0,1.0) for xi in x]
plt.figure()
plt.plot(x,y,'ro')
plt.savefig('plot.pdf', format='pdf')
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

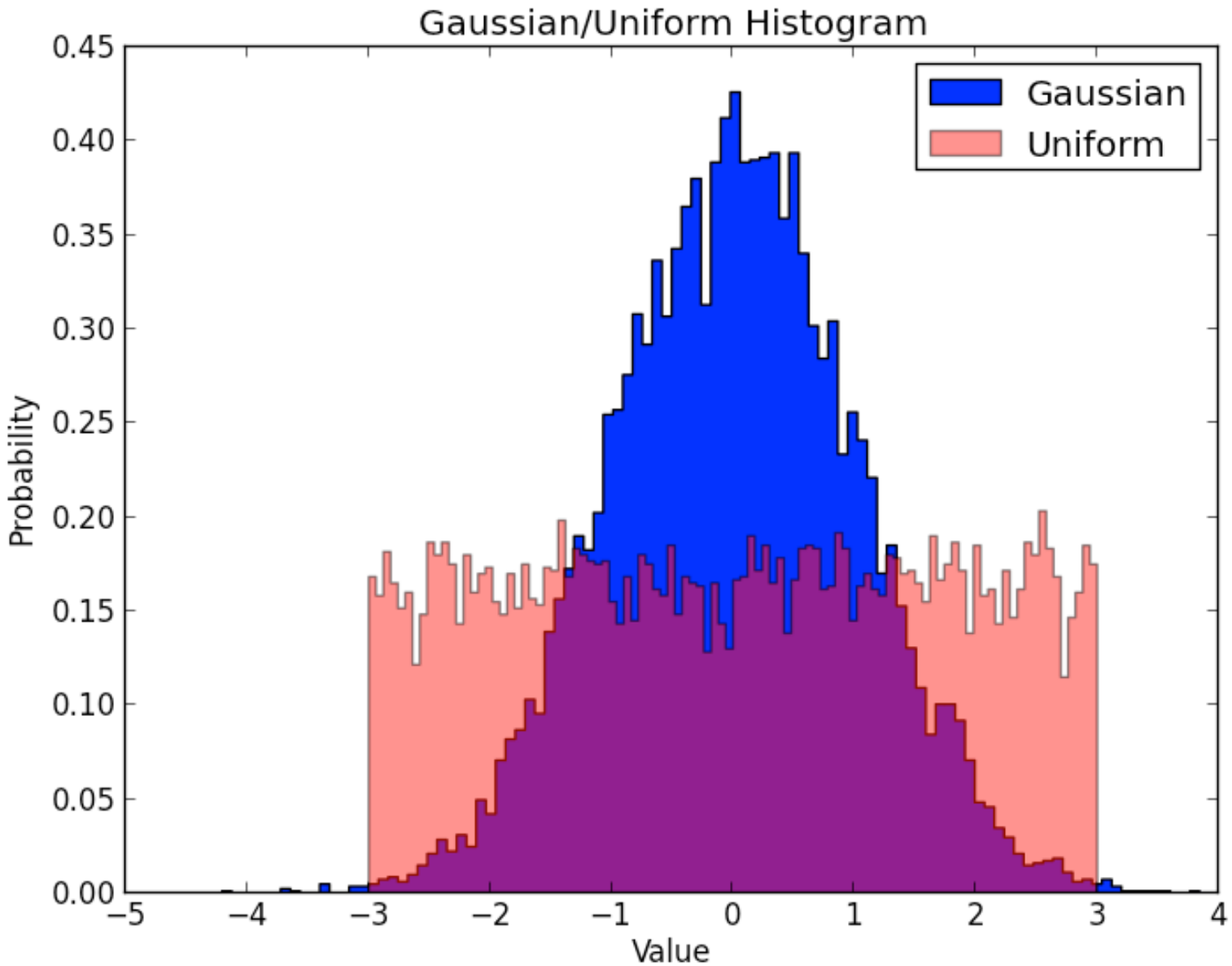
Programming example

```
import random
import math
import sys
import matplotlib
import matplotlib.pyplot as plt
i = 0
n = 10000
r = 1.0
circle = 0.0
square = 0.0
xc=[],yc=[],xr=[],yr=[]
while i < n:
    i += 1
    x = random.uniform(0.,r)
    y = random.uniform(0.,r)
    d = math.sqrt(x*x + y*y)
    if d < r:
        xc.append(x)
        yc.append(y)
        circle += 1
    else:
        xr.append(x)
        yr.append(y)
    square += 1
print "pi=", circle/square * 4.0
plt.figure()
plt.plot(xc,yc,'ro')
plt.plot(xr,yr,'bo')
plt.show()
```



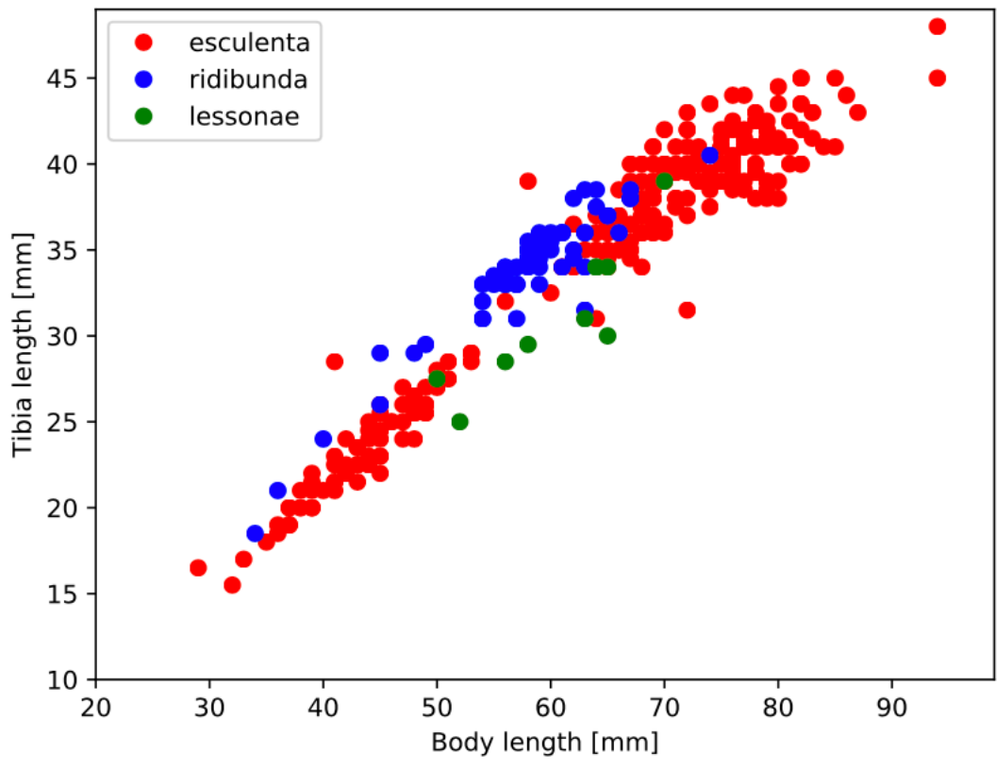
Examples of plots

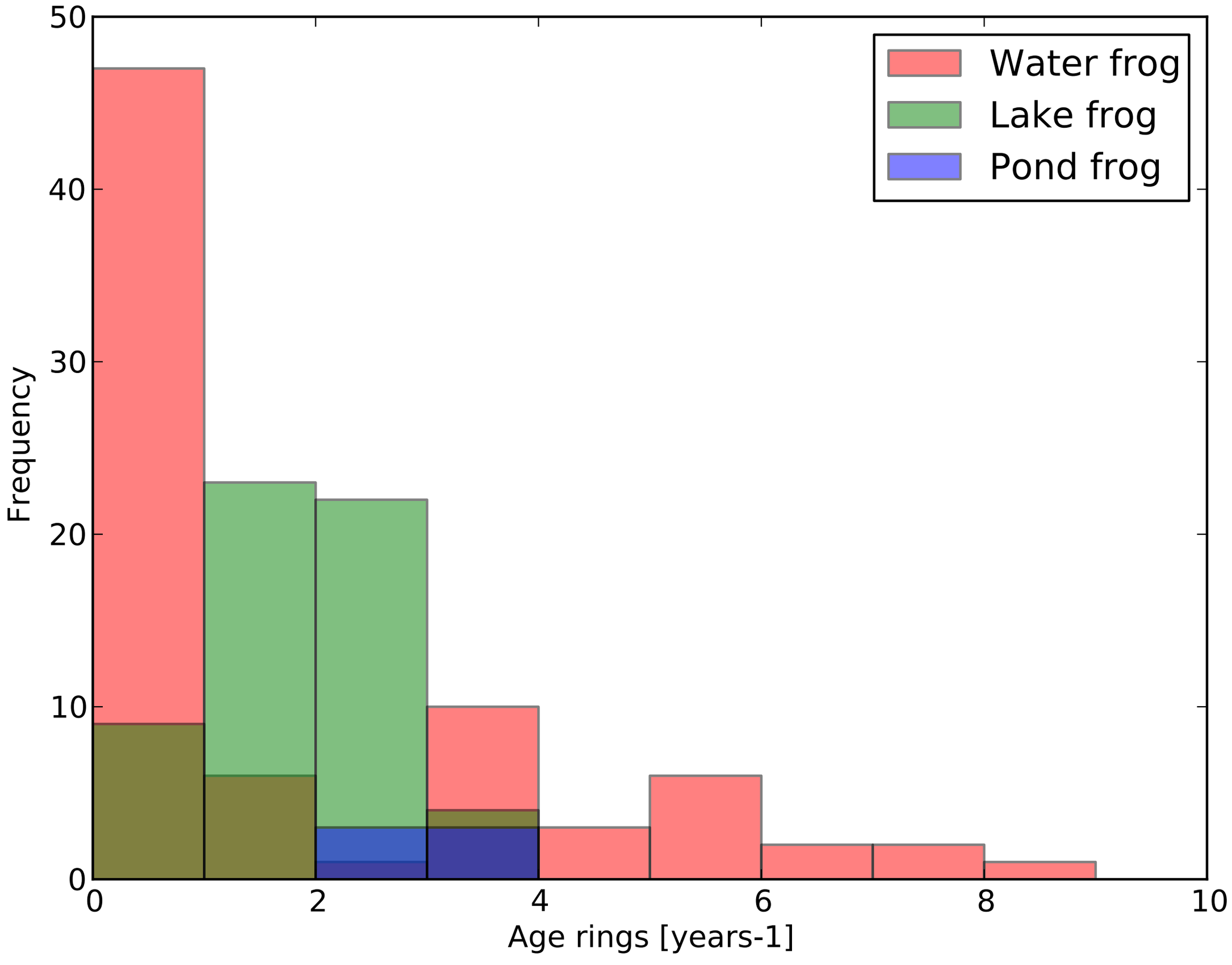
```
import matplotlib.pyplot as plt
from numpy.random import normal, uniform
gaussian_numbers = normal(size=1000)
uniform_numbers = uniform(low=-3, high=3, size=1000)
plt.hist(gaussian_numbers, bins=20, histtype='stepfilled', normed=True,
color='b', label='Gaussian')
plt.hist(uniform_numbers, bins=20, histtype='stepfilled', normed=True,
color='r', alpha=0.5, label='Uniform')
plt.title("Gaussian/Uniform Histogram")
plt.xlabel("Value")
plt.ylabel("Probability")
plt.legend()
plt.show()
```



Programming example

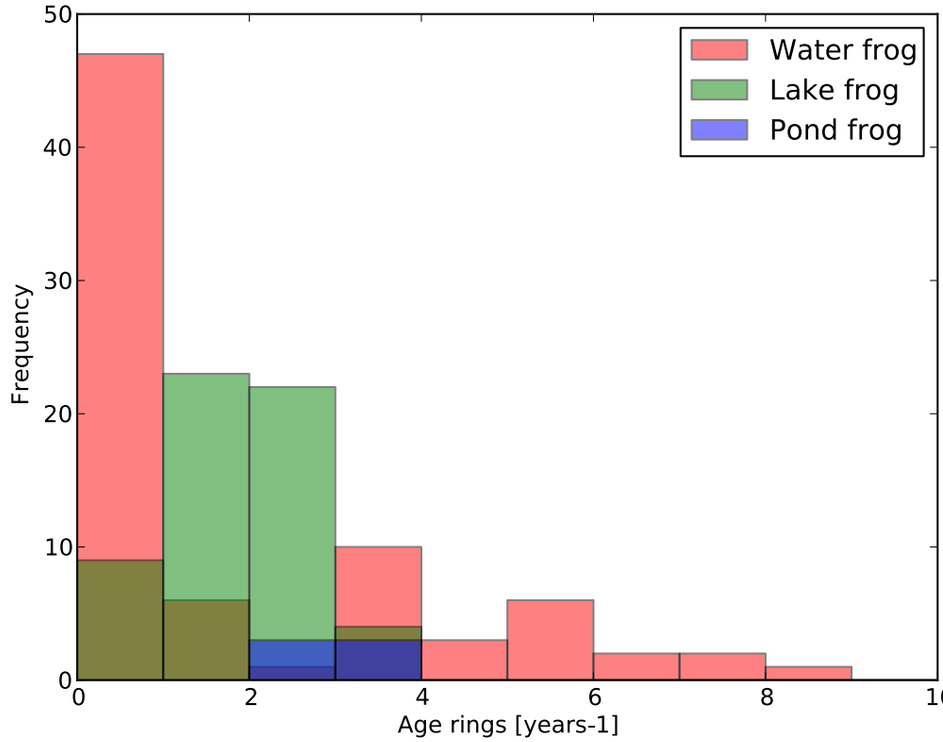
```
#!/usr/bin/env python
import sys
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfFile
import frogs

def column(matrix,i):
    return [float(row[i]) for row in matrix]

data = frogs.read_data('swissfrogs.txt')
newdata = frogs.extract(data,[0,3,4,5])
sl = frogs.unique(newdata,0)
finaldata = frogs.partition_species(sl, newdata)
#print finaldata
xe = column(finaldata[-1],1)
ye = column(finaldata[-1],2)
xl = column(finaldata[1],1)
yl = column(finaldata[1],2)
xr = column(finaldata[0],1)
yr = column(finaldata[0],2)
xheader="Body length [mm]"
yheader="Tibia length [mm]"
xmin = 20
ymin = 10
xmax = max(xe)+5
ymax = max(ye)+1
plt.figure()
plt.plot(xe,ye,'ro')
plt.plot(xr,yr,'bo')
plt.plot(xl,yl,'go')
plt.axis([xmin,xmax,ymin,ymax])
plt.xlabel(xheader)
plt.ylabel(yheader)
plt.savefig('plot.pdf', format='pdf')
```

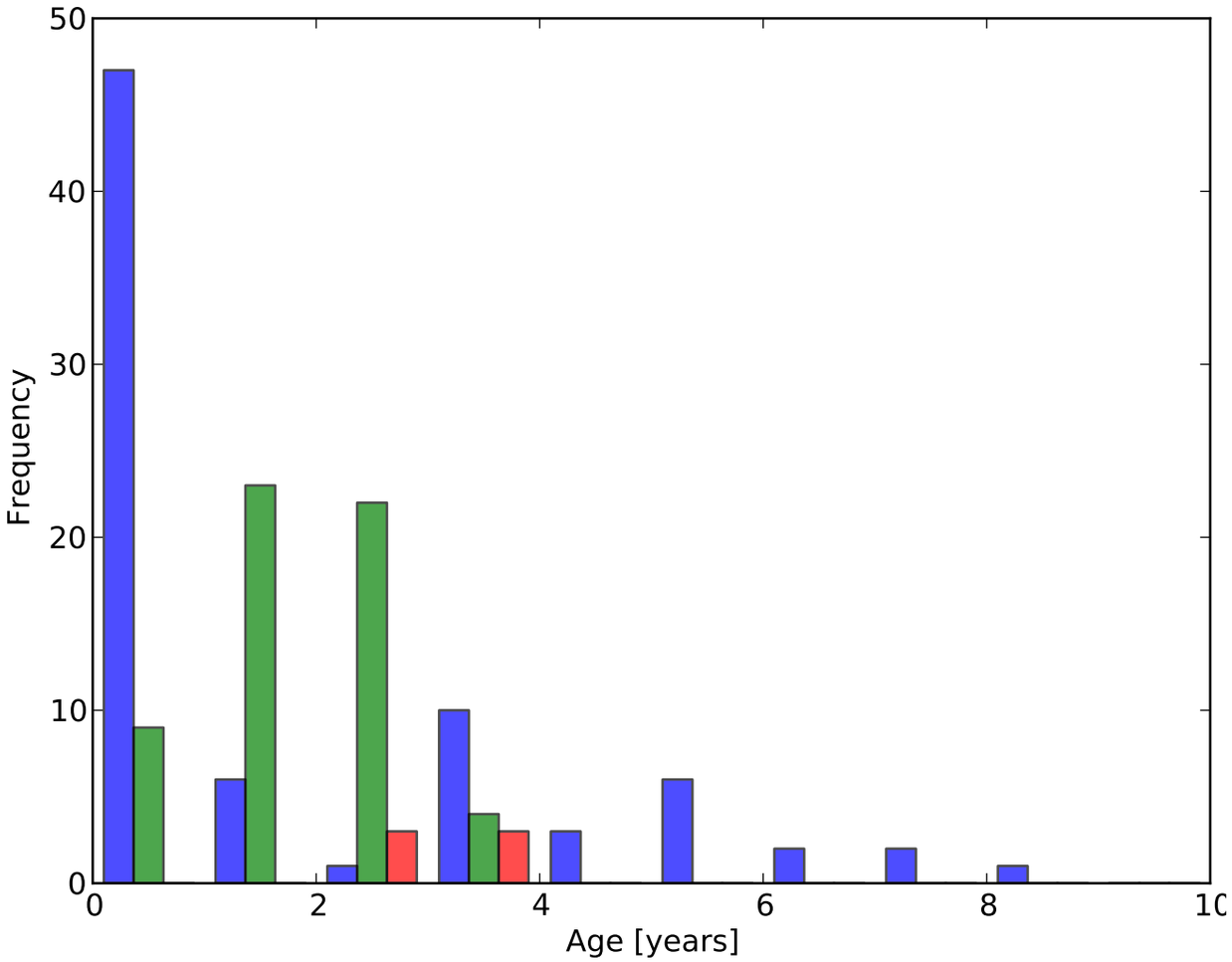


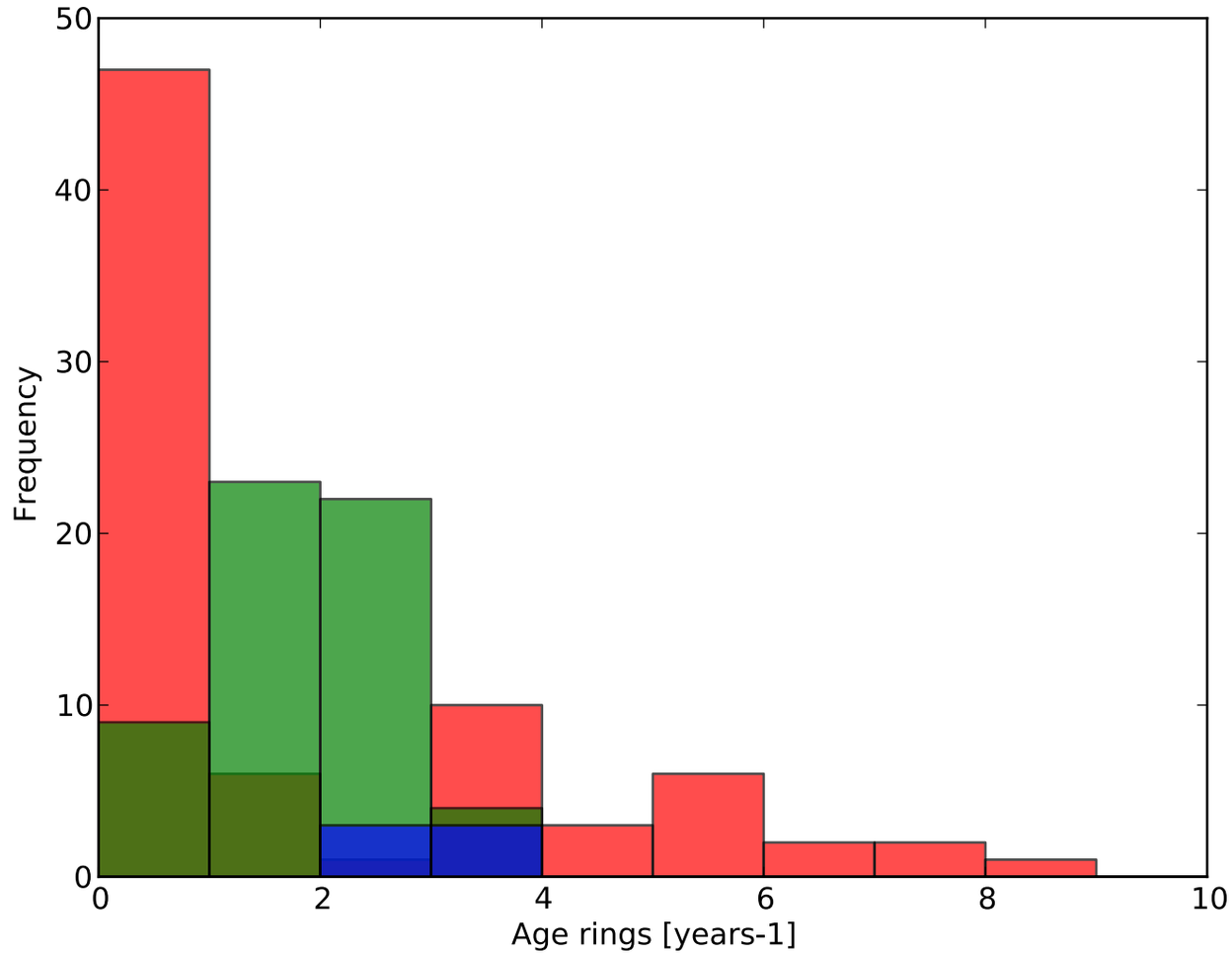

Water frog
Lake frog
Pond frog
Frequency
Age rings [years-1]
0
10
20
30
40
50
0
2
4
6
8
10

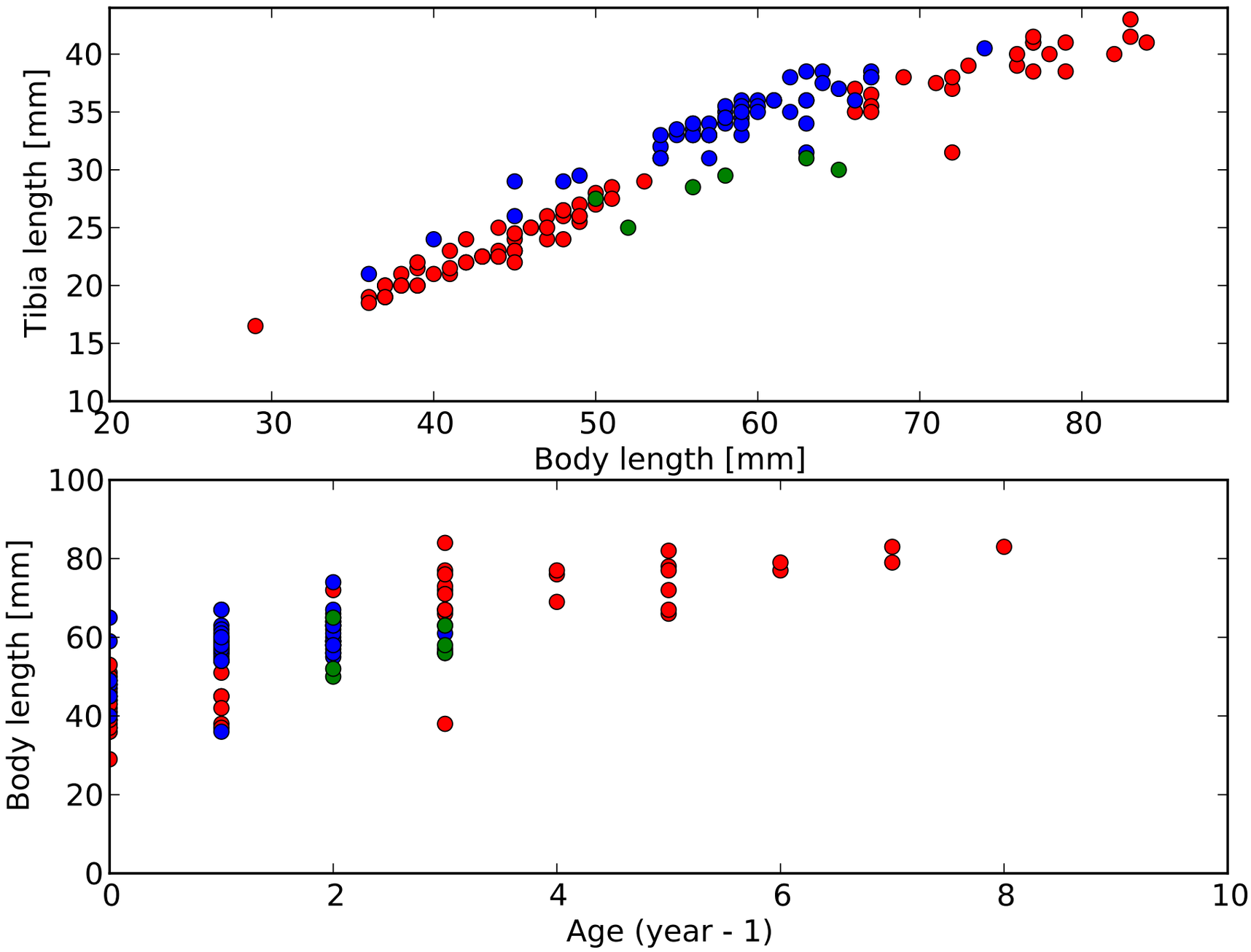
11

```
#!/usr/bin/env python
import sys
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfFile
import frogs

def column(matrix,i):
    return [float(row[i]) for row in matrix]

data = frogs.read_data('swissfrogs.txt')
newdata = frogs.extract(data,[0,12])
specieslist = frogs.unique(newdata,0)
finaldata = frogs.partition_species(specieslist, newdata
xe = column(finaldata[-1],1)
xl = column(finaldata[1],1)
xr = column(finaldata[0],1)
xheader="Age rings [years-1]"
yheader="Frequency"
plt.figure()
#plt.hist([xe,xr,xl],range=[0, 10], alpha=0.7,align='mid')
plt.hist(xe,range=[0, 10], facecolor='red',
alpha=0.5,align='mid',label='Water frog')
plt.hist(xr,range=[0, 10], facecolor='green', alpha=0.5,
align='mid',label='Lake frog')
plt.hist(xl,range=[0, 10], facecolor='blue', alpha=0.5, align='mid',
label='Pond frog')
#plt.axis([0,xmax,ymin,ymax])
plt.xlabel(xheader)
plt.ylabel(yheader)
plt.legend()
plt.savefig('hist.pdf', format='pdf')
```



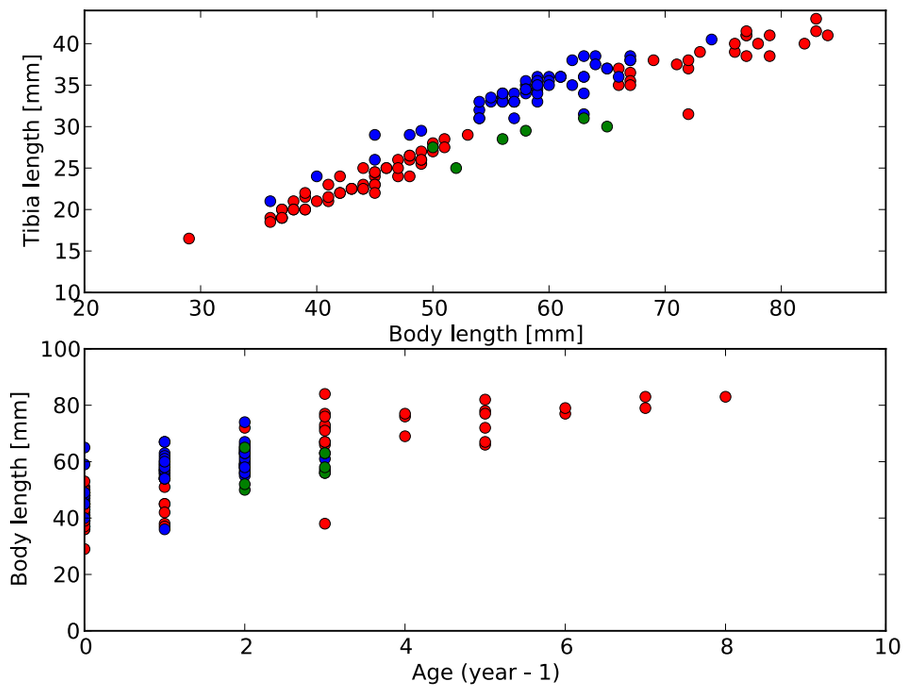

Frequency
Age [years]
0
10
20
30
40
50
0
2
4
6
8
10


Frequency
Age rings [years-1]
0
10
20
30
40
50
0
2
4
6
8
10


Tibia length [mm]
10
15
20
25
30
35
40
20
30
40
50
60
70
80
Body length [mm]
Body length [mm]
0
20
40
60
80
100
0
2
4
6
8
10
Age (year - 1)

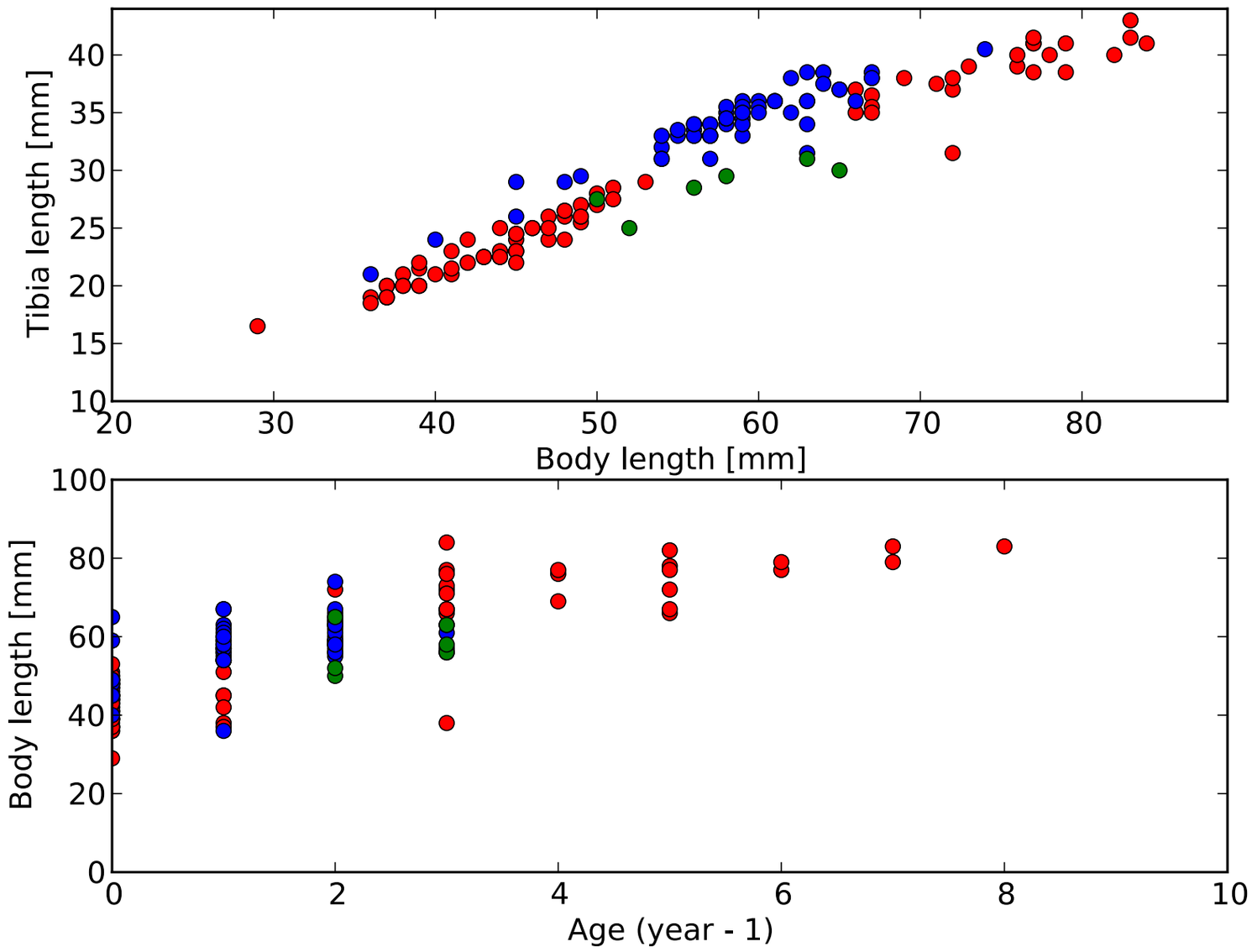
Programming example

```
#!/usr/bin/env python
import sys
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from matplotlib.backends.backend_pdf import PdfFile
import frogs
def column(matrix,i):
    return [float(row[i]) for row in matrix]
data = frogs.read_data('swissfrogs.txt')
newdata = frogs.extract(data,[0,3,4,5,12])
specieslist = frogs.unique(newdata,0)
finaldata = frogs.partition_species(specieslist, newdata,
#print finaldata
xe = column(finaldata[-1],1)
ye = column(finaldata[-1],2)
xl = column(finaldata[1],1)
yl = column(finaldata[1],2)
xr = column(finaldata[0],1)
yr = column(finaldata[0],2)
xheader="Body length [mm]"
yheader="Tibia length [mm]"
xmin = 20
ymin = 10
xmax = max(xe)+5
ymax = max(ye)+1
# plotting instructions
```



Programming example

```
#plotting instructions
plt.figure()
gs = gridspec.GridSpec(2,1)
# set up subplots
ax1 = plt.subplot(gs[0])
ax2 = plt.subplot(gs[1])
ax1.plot(xe,ye,'ro')
ax1.plot(xr,yr,'bo')
ax1.plot(xl,yl,'go')
ax1.axis([xmin,xmax,ymin,ymax])
ax1.set_xlabel(xheader)
ax1.set_ylabel(yheader)
# second plot
xe2 = column(finaldata[-1],4)
xl2 = column(finaldata[1],4)
xr2 = column(finaldata[0],4)
print xe2
ax2.plot(xe2,xe,'ro')
ax2.plot(xr2,xr,'bo')
ax2.plot(xl2,xl,'go')
ax2.axis([0,10,0,100])
ax2.set_xlabel(xheader)
ax2.set_ylabel(yheader)
ax2.set_xlabel("Age (year - 1)")
ax2.set_ylabel(xheader)
plt.savefig('combined2.pdf', format='pdf')
```



Easily creating subplots

In early versions of matplotlib, if you wanted to use the pythonic API and create a figure instance and from that create a grid of subplots, possibly with shared axes, it involved a fair amount of boilerplate code. e.g.

```
# old style
fig = plt.figure()
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222, sharex=ax1, sharey=ax1)
ax3 = fig.add_subplot(223, sharex=ax1, sharey=ax1)
ax3 = fig.add_subplot(224, sharex=ax1, sharey=ax1)
```

Fernando Perez has provided a nice top level method to create in `subplots()` (note the "s" at the end) everything at once, and turn on x and y sharing for the whole bunch. You can either unpack the axes individually:

```
# new style method 1; unpack the axes
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex=True, sharey=True)
ax1.plot(x)
```

or get them back as a numrows x numcolumns object array which supports numpy indexing:

```
# new style method 2; use an axes array
fig, axs = plt.subplots(2, 2, sharex=True, sharey=True)
axs[0,0].plot(x)
```

```
# make up some data on a regular lat/lon grid.
nlats = 73; nlons = 145; delta = 2.*np.pi/(nlons-1)
lats = (0.5*np.pi-delta*np.indices((nlats,nlons))[0,:,:])
lons = (delta*np.indices((nlats,nlons))[1,:,:])
wave = 0.75*(np.sin(2.*lats)**8*np.cos(4.*lons))
mean = 0.5*np.cos(2.*lats)*((np.sin(2.*lats))**2 + 2.)
# compute native map projection coordinates of lat/lon grid.
x, y = map(lons*180./np.pi, lats*180./np.pi)
# contour data over the map.
CS = map.contour(x,y,wave+mean,15,linewidths=1.5)
```