How to do a proper Spearman Rank correlation in python manually ?

This little example shows how to make a proper Sparman Rank correlation calculation in python. For definition of Spearman Rank correlation and the example used, please look here: http://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

The actual implementation is basically taken from http://stackoverflow.com/questions/5284646/rank-items-in-an-array-using-python-numpy

In [47]:
```
#imports
from scipy import stats

#generate sample data from Wikipedia example
X=np.asarray([106,7,86,0,100,27,101,50,99,28,103,29,97,20,113,12,112,6,110,17]).astype('float')
X.shape = (10,2)
```

Now we caluculate the Spearman correlation coefficient either manually or by using the stats.spearmanr function.

The reference solution should be: ρ = −0.175757575... with a P-value = 0.6864058 (using the t distribution)

In [48]:
```
#assign data to x/y
x = X[:,0]; y=X[:,1]

#sort the x-data
ox = x.argsort() #order
rx = np.arange(len(ox)) #assign ranks 0 ... n-1 to x-data

#now sort the y-data in accordance to the x-data sorting which gives a new y-data array
yn = y[ox]

#... and now (this is the trick) we identify the rank of the y-data by first estimating the indices of the order using argsort() anc
oy = yn.argsort() #order THIS IS THE KEY!!!!
ry = oy.argsort()

print '*** Ranked ordered data ***'
print x[ox]
print yn[oy]

print '*** RANKS ***'
print rx
print ry
```

```
*** Ranked ordered data ***
[  86.   97.   99.  100.  101.  103.  106.  110.  112.  113.]
[  0.   6.   7.  12.  17.  20.  27.  28.  29.  50.]
*** RANKS ***
[0 1 2 3 4 5 6 7 8 9]
[0 5 7 6 9 8 2 4 1 3]
```

This result looks now similar to the the result in the reference solution and we can now go ahead with calculating the correlation between the ranks and validate the accuracy of the calculations.

In [49]:
```
print 'Solution by manually calculating Spearman Ranke correlation: ', np.corrcoef(rx,ry)[0,1]

print 'Results from stats.spearmanr                              : ', stats.spearmanr(X[:,0],X[:,1])
print 'Results from stats.mstats.spearmanr                       : ', stats.mstats.spearmanr(X[:,0],X[:,1])
```

```
Solution by manually calculating Spearman Ranke correlation:  -0.175757575758
Results from stats.spearmanr                              :  (-0.17575757575757578, 0.62718834477648444)
Results from stats.mstats.spearmanr                       :  (-0.17575757575757578, masked_array(data = 0.627188344776,
             mask = False,
       fill_value = 1e+20)
)
```

Thus results of all three methods are consistent. *q.e.d*