Dealing with data with plain Python – Workbook

|  |  |
| --- | --- |
| http://www.clker.com/cliparts/M/Q/8/m/N/Z/warning-sign-hi.png | **NOTE**: Code in this document has been tested with Python 2.7.7. However, it may work also with previous and newer versions. |

Tabla de contenido

Dealing with data with plain Python – Workbook 1

1. Proposed exercises 2

1.1 Back to the P/E ratios 2

0.2. Deciding the cases in which you should buy 2

1.3 Doing it random 3

2. Exercises for extra credit 3

Return for IBM stocks 3

Moving average for IBM stocks (advanced) 4

# 1. Proposed exercises

## 1.1 Back to the P/E ratios

Consider the following definition of the PE function:

def PE(price, earnings):

assert price>=0, "Prices cannot be negative"

if earnings <=0:

return float("NaN")

return round(float(price)/float(earnings),1)

Now we have these sentences that create two lists of the same size, the first with prices and the second with earnings.

prices = [20, 15, 12, 75, 40, 5]

earnings = [3, 50, 68, 5, 5, 0]

The lists are paired, i.e. prices[2] and earnings[2] for example define the case of a single company.

Using a for loop, apply the PE function to each of the pairs in the two lists and output the results as follows:

6.7

0.3

0.2

15.0

8.0

nan

*Hint*: maybe you can use the range function. Remember that you can index lists inside the for loop.

## Deciding the cases in which you should buy

Now we will use the PE function to decide to invest only in companies with P/E <=10. Define a function that takes two paired lists (like the ones above), one with prices and one with earnings, and that prints a message only in the investment cases.

def advice(prices, earnings):

# Your code here.

Hint: You should use the PE function inside the advice function.

This is an example usage:

prices = [20, 15, 12, 75, 40, 5]

earnings = [3, 50, 68, 5, 5, 0]

advice(prices, earnings)

And corresponding output:

Invest in company with index 0

Invest in company with index 1

Invest in company with index 2

Invest in company with index 4

## 1.3 Doing it random

Finally, let’s change the code of the previous exercise so that you generate a large number (i.e. several hundreds) of random prices and earnings. For doing so, you will need to investigate the use of some random number generator in Python, there is a module in the Python standard library.

Be sure to generate random numbers that are floating point, and that prices are in the range of zero to 500, and earnings can be negative or positive in an interval from -50 to 200.

# 2. Exercises for extra credit

## Return for IBM stocks

In the class notes, you have an example to get stock data from a symbol from Yahoo Finance. You had also some example code to get adjusted close price from the file downloaded.

Now your task is to define a function that computes the return (rate of return) of a single IBM share given two values, with the simple formula:

*Return = (final price – initial price)/initial price*

So for example, if the value of a share is today 10$, and we sell it at price 20$, the return will be (20-10)/10 = 1, i.e. a 100% of return. Note return can be negative.

Then apply it to all the adjusted close data gathered from Yahoo Finance, and tell the maximum historical return of IBM if we would have bought one day and sell the next trading day.

## Moving average for IBM stocks (advanced)

Now it is time to compute some metrics on IBM stock values. Concretely, in this exercise we ask to code a function that takes a list of prices of a stock over a time period, and returns a list with the simple moving average.

To keep things simple, use the simple numerical example in this Web page to test your program:

<http://www.investopedia.com/terms/m/movingaverage.asp>