Homework 2

PS 398 - Computational Frameworks for Social Science

November 21, 2011

This homework is meant to get you started creating object oriented programs in Python.

Problem Description

Our goal is to build a piece of software for a financial institution to model one of their clients' portfolios. A portfolio can consist of 3 types of items:

- **Cash** can be added to a portfolio, removed from a portfolio or used to buy stocks/mutual funds.
- **Stock** can be purchased with existing cash in the portfolio, or sold (adding cash to the portfolio). Note that stocks can only be purchased or sold as whole units. Stocks have a price and ticker symbol. For simplicity's sake, Stocks can be purchased for $X/share and when sold are sold for a price that is uniformly drawn from [0.5X-1.5X]
- **Mutual Funds** can be purchased with existing cash in the portfolio, or sold (adding cash to the portfolio). Note that mutual funds can only be purchased as fractional shares. Mutual funds have a price and ticker symbol. For simplicity's sake, Mutual funds can be purchased for $1/share and when sold are sold for a price that is uniformly drawn from [0.9-1.2]

Your program must facilitate managing the correct balance of cash, stocks and mutual funds as the user buys and sells items. Assume that the person using your library will specify the correct buy price so you can trust it and just need to maintain a proper internal state given the specified buy price (and then compute some sell price using the above formulas). Finally, in order to help with customer service your portfolio software needs to keep an audit log of all transactions and make them available to users of your program.

You can implement this software however you wish, but a consumer of the application must at a minimum be able to do the following:

```
portfolio = Portfolio()                 #Creates a new portfolio
portfolio.addCash(300.50)               #Adds cash to the portfolio
s = Stock(20, "HFH")                    #Create Stock with price 20 and symbol "HFH"
portfolio.buyStock(5, s)                #Buys 5 shares of stock s
mf1 = MutualFund(5, "BRT")              #Create MF with price 5 and symbol "BRT"
mf2 = MutualFund(2, "GHT")              #Create MF with price 2 and symbol "GHT"
portfolio.buyMutualFund(10.3, mf1)      #Buys 30.3 shares of "BRT"
```

```
portfolio.buyMutualFund(2, mf2)        #Buys 2 shares of "GHT"
print(portfolio)                       #Prints portfolio
                                       #cash: $140.50
                                       #stock: 5 HFH
                                       #mutual funds: 10.33 BRT
                                       #              2     GHT
portfolio.sellMutualFund("BRT", 3)     #Sells 3 shares of BRT
portfolio.sellStock("HFH", 1)          #Sells 1 share of HFH
portfolio.withdrawCash(50)             #Removes $50
portfolio.history()                    #Prints a list of all transactions
                                       #ordered by time
```

Recommendations

Oftentimes the easiest way to work through a problem such as this is to think about what you are trying to model and assign "ownership" of data. What are the real world objects that we are dealing with? What information does each of those objects need to keep track of so that they can answer the questions required by our spec? How are these objects related and how might we make use of inheritance/polymorphism to stay DRY? What types of errors could occur?

It might be a good idea to take what we learned about TDD and see how you can apply it to this problem.

BONUS: Using inheritance show how it would be easy to add a third type of investments—Bonds—to the mix.