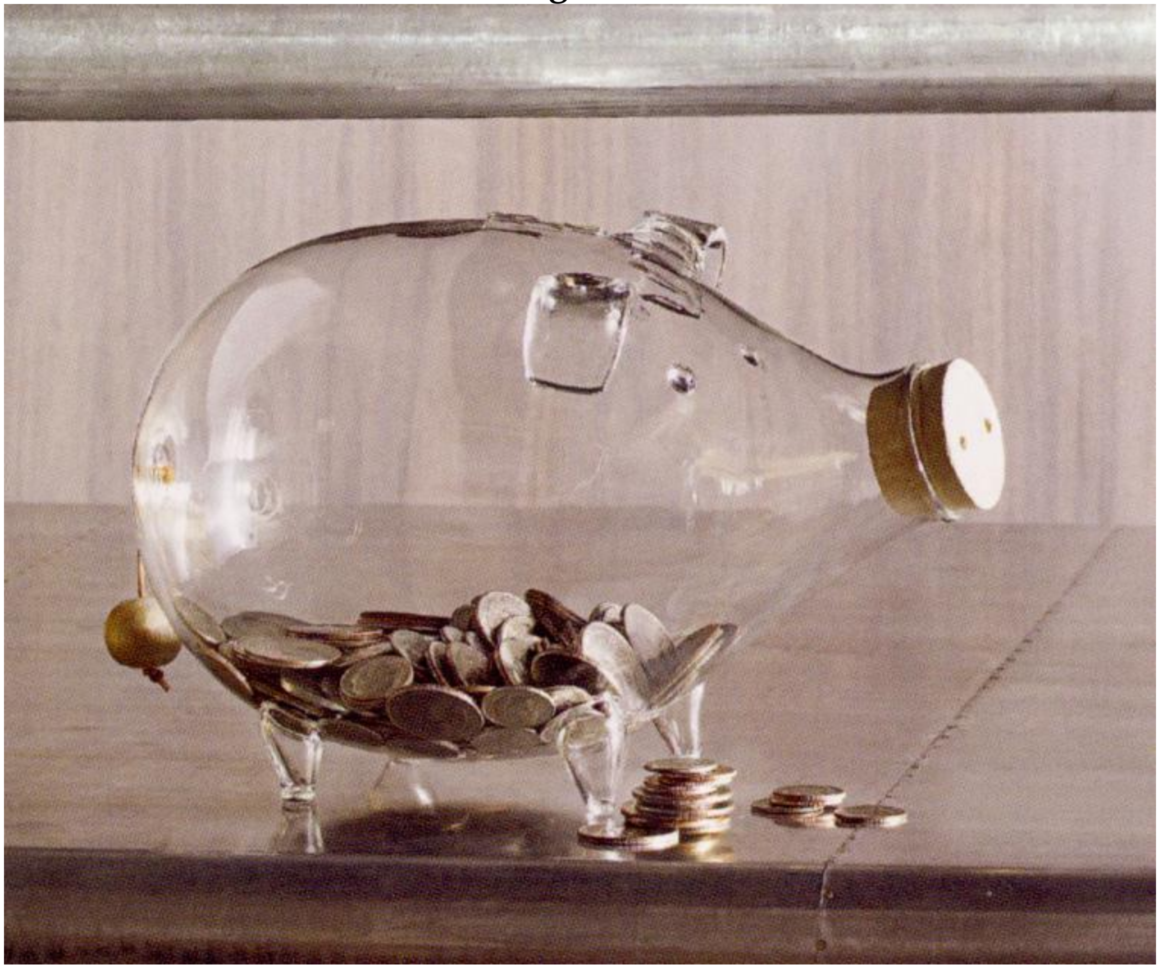
Banking Accounts


picture from http://ep.yimg.com/ca/I/yhst-53777107109672_2258_44115549.jpg

Objectives:

- To practice writing classes
- To learn how to use inheritance

Background:

This week we'll be writing a program simulating financial accounts for a company. This company would like to manage their accounts using a full electronic system. Specifically, they want to manage two types of accounts: a banking account and a college bond. For the banking account, they would like users to be able to deposit and withdraw money, calculate the interest for their account, and see their account status. For the college bond, the company would like for users to be able to deposit money, calculate the interest for their account, see their account status, and see the number of years remaining until the bond matures. To complete this lab, we'll use inheritance to solve both problems and do so without excessive copying of code.

Assignment:

To complete this assignment, write the following three classes as instructed and store them in a file named Banking.py. To test your program, run driver.py in the same directory that Banking.py is stored in.

For demo 1, write an **Account** class with the following functions

__init__():	This method needs to accept the name of the account holder and the name of the banking company (also a string). Use the string "unknown" for the default argument. It should also initialize the amount of money deposited in the bank to 0 and the interest rate to 2.
deposit():	This method needs to accept an integer value as input and increase the variable representing the amount of money deposited in the bank by that value.
calculateInterest():	This method needs to calculate the (simple) interest and return it. (To calculate the interest, simply multiply the amount of money deposited by the interest rate and divide by 100.)
getStatus():	This method should return a string that says "[ownerName] has a net deposit of [moneyDeposited] dollars in [companyName]."

For demo 2, write a **BankAccount** class and a **CollegeBond** class. Both of these classes should inherit from the account class.

BankAccount should have the following methods:

withdraw():	This method needs to accept an integer value as input and decrease the variable representing the amount of money deposited in the bank by that value. Note that you will have to consider what happens if you end up with negative money.

And, finally, **CollegeBond** should have the following methods:

__init__():	See above. In addition to the above parameters, this method needs to accept an additional argument to set the length, in years, of the savings bond. The default value for the length

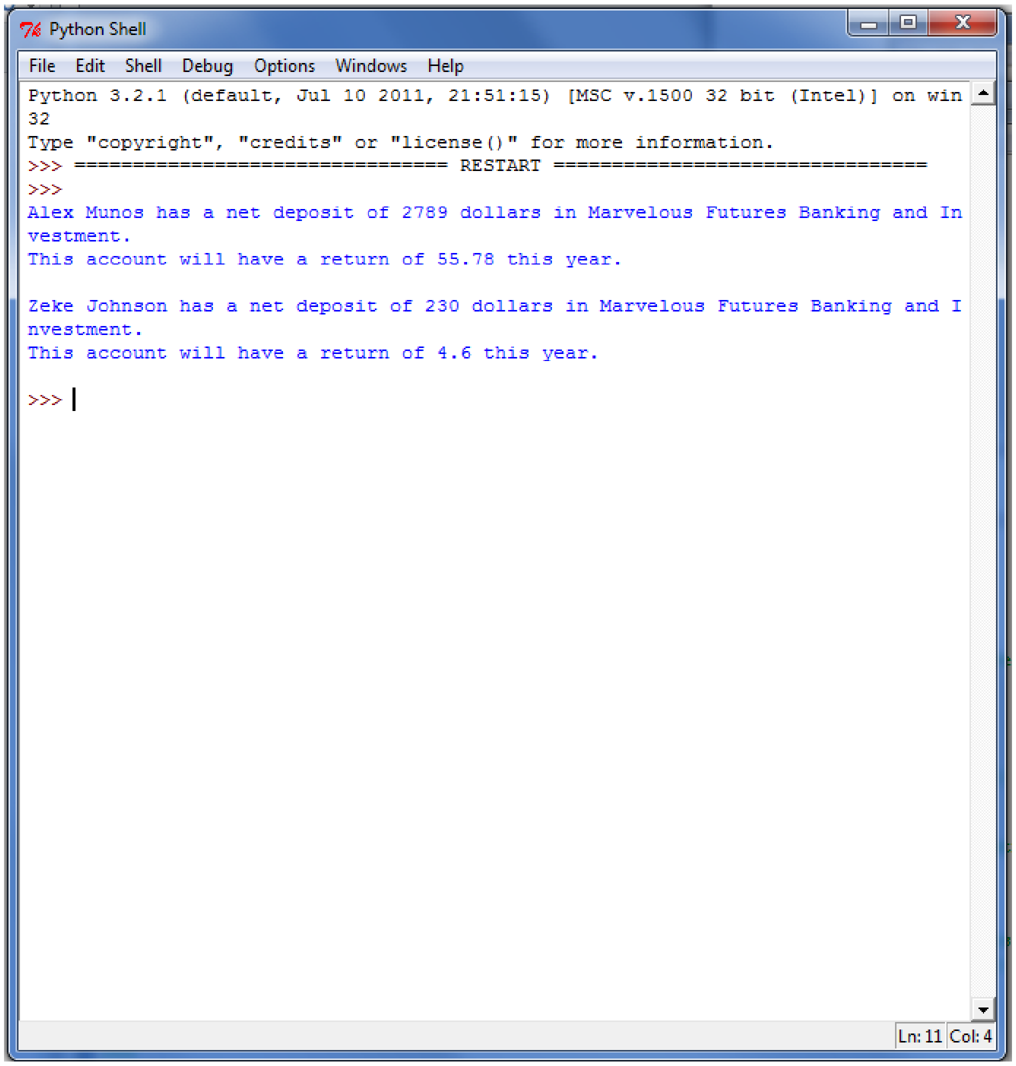
	should be 8 years. Also, the interest rate should be initialized to 10.
getBondLength():	This method needs to return the number of years until the savings bond reaches maturity.

Demo 1: Write the Account class and its corresponding functions. Run the driver program as it is given. **Do this first!**

Demo 2: Write the other two classes and their functions. Run the given program with the part that is originally commented out. (Remove the “””).

Sample Execution:

Demo 1:

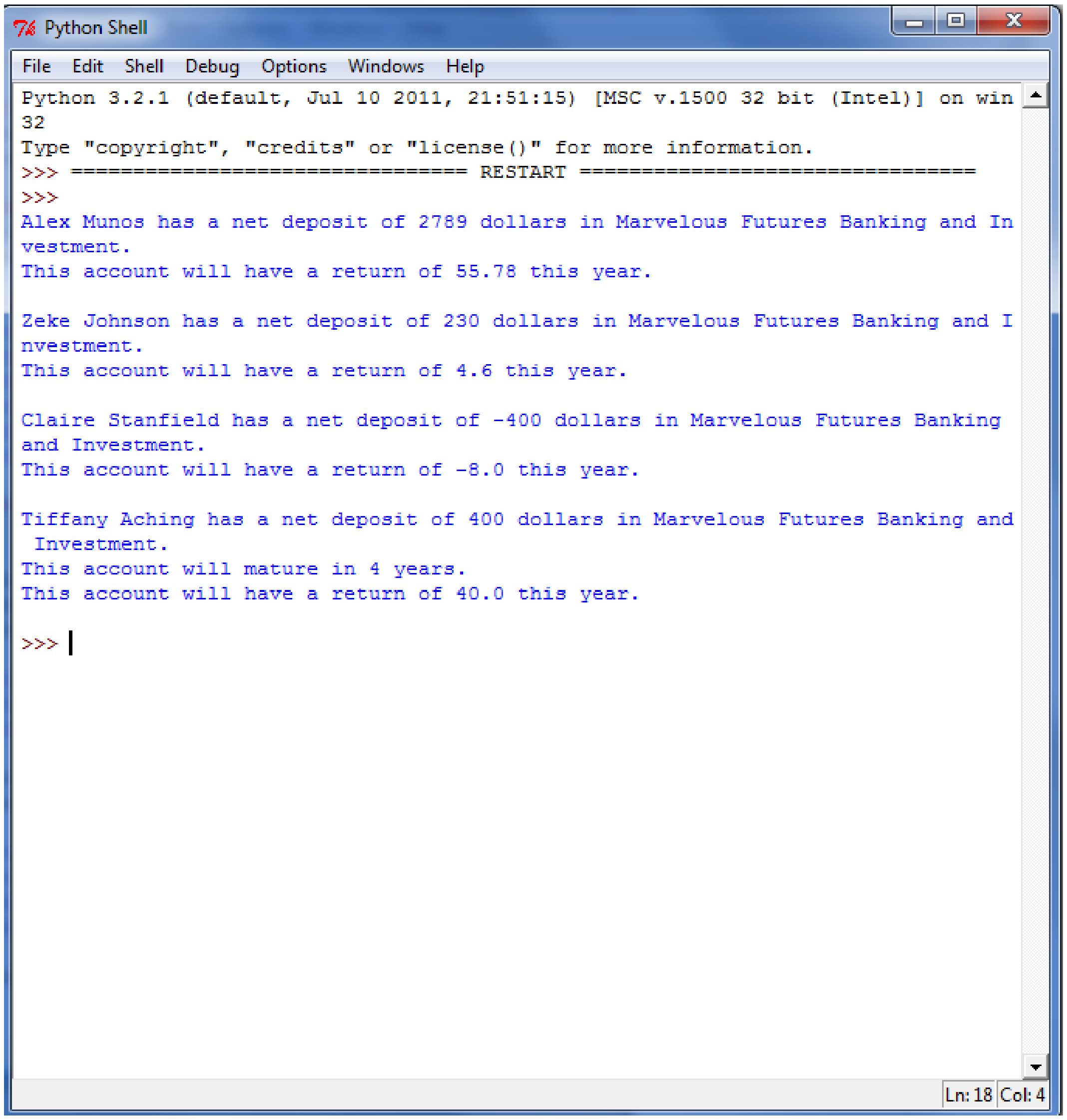


```
Python 3.2.1 (default, Jul 10 2011, 21:51:15) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Alex Munos has a net deposit of 2789 dollars in Marvelous Futures Banking and In
vestment.
This account will have a return of 55.78 this year.

Zeke Johnson has a net deposit of 230 dollars in Marvelous Futures Banking and I
nvestment.
This account will have a return of 4.6 this year.

>>> 
```

Demo 2:


Python Shell

File Edit Shell Debug Options Windows Help

```
Python 3.2.1 (default, Jul 10 2011, 21:51:15) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>> 
Alex Munos has a net deposit of 2789 dollars in Marvelous Futures Banking and In
vestment.
This account will have a return of 55.78 this year.

Zeke Johnson has a net deposit of 230 dollars in Marvelous Futures Banking and I
nvestment.
This account will have a return of 4.6 this year.

Claire Stanfield has a net deposit of -400 dollars in Marvelous Futures Banking
and Investment.
This account will have a return of -8.0 this year.

Tiffany Aching has a net deposit of 400 dollars in Marvelous Futures Banking and
 Investment.
This account will mature in 4 years.
This account will have a return of 40.0 this year.

>>> 
```

Ln: 18 Col: 4

Program Requirements:

For the code:

- Since this is a paired assignment, your program should have the following comment block at the top.

```
#
#Names:        Name of one partner         Name of other partner
#Date Assigned:  Date of lab                Date Submitted:  Date submitted
#Course:       CSE 1384                     Lab Section:  Your lab section
#
#File name:  lastName.py  (of the person submitting file in myCourses)
#
#Program Description:  A short description of what the program is
#supposed to accomplish.
#
```

- Use appropriate comments throughout the code. Be sure to include comments before each class definition and before each method (member function) explaining the interface (what must be sent, what is returned) and the purpose of the function. Also, don't forget to make appropriate comments within the method definition.

- Make good use of whitespace. Be sure to include a blank line of space before each comment. Two lines of space between functions is nice, but one will suffice – whichever you choose, be consistent. Follow the Python programming style guide: https://www.python.org/dev/peps/pep-0008/#introduction

- Don't forget to use good **descriptive** variable names

For the lab report, follow the Lab Report Format Guide and complete the following sections:

- Title Page
- Design – Create the UML diagrams for all 3 classes.
- Analysis and Conclusions
- Extra Analysis Questions:
  - What were the advantages, if any, of using inheritance to solve this problem?
  - After this lab, do you think you will find it easier to use inheritance in your own code? What are some of the reasons why you would or would not use inheritance?

- Appendix B - code (Be sure that you copy and paste the code into your word document, don't take a screen shot of it and paste that – it ends up being too hard to read.) **Be sure that both partners have a copy of the code before leaving lab.**

Deliverables:

Electronic submission in myCourses:

- Code (due by the end of the class period) submitted by **one** of the partners, not both
- Lab Report – to be done individually (due by the start of the next time that your lab meets) -- either a Microsoft Word document or a pdf

Paper submission in class:

- Lab Report (due at the start of lab the next time that it meets)

Grading:

Task	**Points**
Lab Report	40 points
Title Page	5 points
Design (UML diagrams)	10 points
Analysis and Conclusions	10 points
Extra Analysis Questions	5 points
Appendix B – coding style points (see requirements section)	10 points
Program	60 points
Account class	20 points
BankAccount class (using inheritance)	20 points
CollegeBond class (using inheritance)	20 points