MIT AITI
Python Software Development

Lab 5: Functions
Due: Tuesday 21, 2011

In this lab, we will practice using python functions in the context of managing a simple mobile banking system. Functions allow us to gain access to and manipulate data in a wide variety of ways.

We will use a dictionary to store information, with each key being the account holder's name and the value being the amount of money they currently have in the bank.

1. Implement the dictionary data structure, naming it `myBank`. Initialize it with three names: John, Isaac, and Tony, each having an initial account balance of 0. Refer to the previous lab if you have trouble. (You do not need to use a function for this).

2. Next, create each of the following functions.

- `def createAcct(name, initBal, dict=myBank):`

  #adds a member with name and initial balance to a dictionary.

- `def deleteAcct(name, dict=myBank):`

  #removes a member from this dictionary. Returns true if the name has been
  #successfully removed, and false if the name did not exist.

- `def deposit(name, amount, dict=myBank):`

  #deposits a certain amount of money to a member's account. If “name” does not
  #exist in “dict”, create it with “amount” as the initial deposit.

- `def withdraw(name, amount, dict=myBank):`

  #withdraws a certain amount of money from the member's account. However, if this
  #withdrawal would take the person's balance to below 0, do not make the
  #transaction, and instead return a message saying that the transaction could not be
  #completed.

- `def transfer(nameFrom, nameTo, amount, dict=myBank):`

  #transfers an amount of money from one person to another. *Do not repeat code*
  #(use the functions defined above).

- `def bankAssets(dict=myBank):`

  #returns the total amount of money in your bank.

- ```
  def reverseLookup(amount, dict=myBank):
  ```

  #returns the name of the person with “amount” in the bank, or “None” if not found.

3. Add documentation to at least two of the previous functions, so that, for example, createAcct.__doc__() returns a brief description of what createAcct does. Refer to http://docs.python.org/tutorial/controlflow.html#documentation-strings for more explanation.

4. We want to be able to investigate what happens to our lenders' bank accounts over time. Create the following function:

- ```
  def interestGrowth(days, iRate=.05, compounded='yearly',
                     dict=myBank):
  ```

  #simulates the passage of some amount of days, applying the given interest rate to #each bank member. “compounded” should be able to take values 'yearly', #'monthly', and 'daily'. Use the equation P = C(1+r/n)^nt, where P is the future value, #C is the initial value, r is the interest rate, t is the number of years passed, and n is #the number of times per year compounded. Note that the argument in the function #is in days, while t is in years.