NWYH — Computer Science — Python — HW #11

Assigned on Mon, 2017-03-13. Complete by Tue, 2017-03-21.

1. Jump forward a bit and read chapters 15 and 16 of Think Python, 2nd ed. (classes, objects, and functions) this week.
2. **[Turn in]** This assignment deals with function composition, which works just like it does in mathematics.

Recall that in math, if f(x) and g(x) are both functions, then their composition is defined as f(g(x)).

This can be illustrated in a diagram like so:



Temperature scales are related to each other through linear functions, of the form temperature(x) = m\*x + b. A linear function like this can be built as a composition of two simpler functions, which we'll call multx and plusb.



**Note:** Be careful about the ordering of these functions! Although g(f(x)) is written with the function g to the left of function f, the function f is applied to the input before the function g. This makes a difference, since the order of application matters: usually g(f(x)) is not equal to f(g(x)).

Download the file **composed\_c2f.py.zip** from the course website, and unzip it to get **composed\_c2f.py**.

The script was written to demonstrate that functions can take lambdas as function arguments, and can also return lambdas as their return values. In this script, the function **composed\_c2f** composes two functions, as illustrated in the diagrams above. I added assert statements to ensure that the script worked as intended, but one of the assert statements fails. Please fix the script so that all of the assert statements succeed, and then submit the fixed script as an attachment. *If you're unsure of how a piece of software works, one common way to better understand it is to add print statements to print the values that different variables have at specific (hard-to-understand) places in the script.*

1. **[Turn in]** Download the file **bank\_account\_transfer.py.zip** from the course website, and unzip it to get **bank\_account\_transfer.py**. Make the two modifications described in the file:
2. Define a function transfer, that transfers from the first argument (of type BankAccount) to the second argument (also of type BankAccount) the amount specified by the third argument (of type float).
3. Add a few lines to the end of the file to transfer $75.00 from the savings account to the checking account, then printing the values of both accounts.

Lastly, submit the resulting file as an attachment.

1. **[Turn in]**
   1. Browse through the draft of the Unit #3 slides that are available on the course website, in case they give you any possible course project ideas.
   2. Submit three final "elevator pitch" ideas for a course project. (This makes nine total.) These should present complete ideas using complete sentences.
2. Download the file **named\_bank\_account.py.zip**, and unzip it to get **named\_bank\_account.py**. This script is based on the class **BankAccount** that we saw in class earlier, with a few modifications:
3. Each account stores the name of its owner as an instance-level attribute.
4. A class-level dictionary called **accounts** is maintained that stores account owner names as dict keys, and the corresponding accounts themselves as dict values.
5. A function called **print\_balances** is added that uses this dictionary to print all the account balances.

Figure out what you think the output of the script would be, and then run it to confirm.