Homework 10:

Final Project, Phase 3: Coding and Documentation

Out Tuesday, 4/15/14
Due Tuesday, 4/22/14 @ midnight
10 points

Programming Logic and Design
ITEC 1150
Spring Semester, 2014
Eric V. Level, Instruct

In this Phase 3 of your Final Project, you'll begin your coding and documentation of your proposed application. You translate your pseudocode from Phase 2 into Python functions and the modules that contain them, then add descriptive documentation to it.

One approach is to code "top-down"; that is, create a **`main()`** function that contains the Python that implements your top-level pseudocode. Code your **`main()`** function incrementally: write beginning code that ends calling your first function, "mocking" its definition like this:

```
def functionX(args):
          # a "mock" function:
          #  just print a message and return
     print ("functionX(args) called.")

def main():
          # do set up in main, then call first function:
     functionX(args)

main()
```

Get the above running first - then implement your actual **`function1()`** code in Python. Add return arguments that are needed in **`main()`** like this:

```
def functionX(args):
          # now this function is "real"
     print ("functionX(args) called and stuff computed….")
          # more code follows…
          # final, return results as list; you can return
          #  individual data or dictionaries or strings or
          #  tuples as needed
     return [result1,result2]

def main():
          # do set up in main, then call first function:
     (calc1,calc2) = functionX(args)
          # continue with top-level coding…
```

Again, work incrementally: add another function definition, mock it, call it from your **`main()`**, and so forth.

Another approach is to begin by using your list of candidate classes, then implementing the most important first:

```
class CentralClass(object):
     def __init__(self, initArg1, initArg2,…):
          self.attribute1 = …
          self.attribute2 = …

     def __str__(self):
          return "string rep of CentralClass instance"
```

Initially just code the constructor (`__init__(self,…)`) and the "convert to string" method (`__str__(self)`). Then test your work by writing a **`main()`** that creates an instance and prints its string representation:

```
def main():
     obj = CentralClass(initData1, initData2)
     print (obj)
```

Then start coding the main behavior of your application, using these objects. Add methods as you need them - but be aware that figuring out exactly which methods you need in a class is tricky. It's best to add methods only when you need them; otherwise, you might be writing that is never used. (More on this in class.)

You can use your procedural design's pseudocode from the first part of your Phase 2 design document as a guide for coding your main, adding code so it manipulates instances of the classes you identified and coded.

Finally, don't worry if you discover flaws in your original design. Add any overlooked classes or pseudocode functionality while writing your code as needed. If you do, don't worry about changing your original design document (both procedural and object-oriented).

When you have a rough working version of your application (remember it's fine for it to be incomplete at first, since ideally you are building it out, bit by bit), then add docstring documentation (`""" ... """`) for each function, class, and method. You can also add internal comments (`# ...`) to clarify your code - but don't overcomment.

Submit your application as a collection of .py files (modules) to the **Homework 10 Dropbox** on our D2L site. Name them descriptively - but you don't need to include your name. You should have a set of comments at the top of each submitted file that identifies you and your team partner, if any.

And remember that this is not your final version of your Project code. It's just a "first draft" which you can continue improving until your final presentation.