CS 1110, LAB 5: OBJECTS; CONDITIONALS; OINK!

http://www.cs.cornell.edu/courses/cs1110/2017sp/labs/lab05.pdf

First Name: ____________ **Last Name**: ____________ **NetID**: ____________

Getting Credit: Deadline: **in the first 10 minutes of (your) lab next week (Tue Mar 7 or Wed 8)**. The checking-off procedure is the same as before.[1]

Lab Materials. Create a new directory on your hard drive for this lab's files. Then, download into that new directory the files you need for lab 05; get them packaged in a single zip file from the Labs section of the course web page, http://www.cs.cornell.edu/courses/cs1110/2017sp/labs .

1. The Time Class

The class `Time` is defined in `lab05_accessories.py`. Objects of class `Time` represent time durations, and have exactly two attributes, `minutes` and `hours`. Both of these attributes are non-negative integers, though there is an important condition you must maintain on `minutes`: it must be between 0 and 59, inclusive.

Open up the Python interactive shell and use it to fill in the following table. We've done the first two entries for you, to set things up.

Course authors: E. Andersen, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White

[1] In case you've forgotten, here's a reminder: Show this handout and/or your code to a staff member either (a) during your lab 05 session, (b) in consulting hours listed at http://www.cs.cornell.edu/courses/cs1110/2017sp/about/staff.php up to the day **before** your next scheduled lab section, or (c) in the first 10 minutes of (your) next scheduled lab (Tue Mar 7 or Wed 8). Beyond that time, the staff have been instructed not to give you credit.
Labs are graded on effort, not correctness. We just want to see that you tried all the exercises, and to clarify any misunderstandings or questions you have.

Statement/Expression	**Expected Value**	**Calculated Value**
`from lab05_accessories import Time`	(none)	(none)
`t = Time(2,30)`	(none)	(none)
`t.minutes`		
`t.hours`		
`t.days`		
`s = t`		
`s.minutes`		
`t.minutes = 20`		
`s.minutes`		
`u = Time(t.hours, t.minutes)`		
`u.minutes`		
`t.minutes = 10`		
`u.minutes`		
`s.minutes`		
`s = Time(2,10)`		
`s.minutes = 2`		
`t.minutes`		

It might help to think of `t`, `s`, and `u` as representing time that different students have until a deadline. The assignment `s = t` is like grouping two students on CMS: their time-left variables refer to the same object, so that changing one person's changes the other. The assignment `u = Time(t.hours, t.minutes)` gives a third student a different time-remaining that coincidentally has the same hours and minutes left as `t`, but `u` and `t` refer to *different* objects. So, a change to the time-left in `t` does not affect the time-left for `u`. It *does* affect the time-left for `s` ... until the assignment `s = Time(2,10)`, which "ungroups" `s` and `t` by making the time-left for `s` a new, *different* object than the one for `t`.

2. The Function add_time(time1,time2)

Complete function `add_time` in file `lab05.py`.

```
def add_time(time1, time2):
    """Returns: A new Time object representing the sum of time1 and time2.
    Does not alter time1 or time2.

    Example: if time1 is 1 hr 59 min and time2 is 1 hr 2 min,
    then the returned object is 3 hr 1 min.

    Preconditions:
        time1 is a Time object
        time2 is a Time object"""
```

Remember that according to the constraints on Time attributes, if a Time object's `minutes` attribute would go over 59, the `hours` attribute should be incremented, so that `minutes` remains below 60. Also, because you are creating a new Time object, this function will need to call the constructor for Time; and since the Time class is housed in an imported file, your constructor call will take the form `lab05_accessories.Time(...)`.

You can test your function by running `Python lab05_accessories.py` on the command line – we've provided some test cases for you.

3. Pig Latin

Pig Latin is a simple[2] encoding of strings that adheres to the following rules, where here "word" means a non-empty string consisting only of lowercase letters:

(1) The vowels are `'a'`, `'e'`, `'i'`, `'o'`, `'u'`, as well as any `'y'` that is *not* the first letter of a word. All other letters are consonants. For example, `'yearly'` has three vowels (`'e'`, `'a'`, and the last `'y'`) and three consonants (the first `'y'`, `'r'`, and `'l'`).

(2) If the input word begins with a vowel, append `'hay'` to the end of the word to get the Pig Latin equivalent. For example, `'ask'` becomes `'askhay'`, `'use'` becomes `'usehay'`.

(3) Otherwise, if the word starts with `'q'`, assume it is followed by `'u'`; move `'qu'` to the end of the word, and append `'ay'`. Hence `'quiet'` becomes `'ietquay'`, `'quay'` becomes `'ayquay'`.

(4) Otherwise, if the word begins with a consonant, move all the consonants up to the first vowel (if any) to the end and add `'ay'`. For example, `'tomato'` becomes `'omatotay'`, `'school'` becomes `'oolschay'`, `'you'` becomes `'ouyay'`, and `'pssst'` becomes `'pssstay'`.

Your goal is to write a function `pigify` that takes a word (defined above), and converts it into Pig Latin.

Rule (1) is slightly tricky to implement, so we've provided a helper function `first_vowel(w)` in `lab05_accessories.py` that implements it for you. Using this helper, complete the function `pigify` in `lab05.py`. You can test your function by running `Python lab05_accessories.py` on the command line — we've provided some test cases for you.

[2]So ixnay for ationalnay ecuritysay, if you get our iftdray.