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

http://www.cs.cornell.edu/courses/cs1110/2018sp/labs/lab05/lab05.pdf

First Name: _____ **Last Name**: _____ **NetID**: _____

Getting Credit: **As always, strive to finish during the lab session** — it's the best way to stay on track in this course.[1]

1. Setup

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 from `http://www.cs.cornell.edu/courses/cs1110/2018sp/labs` .

2. Working with objects: the Timer class

The class `Timer` is defined in `timer.py`, but don't open that file — it uses Python we haven't covered in class yet. Rather, here is all the information you need to know about `Timer` objects.

First, `Timer` objects represent time durations, and have exactly two attributes, `hours` and `minutes`. Both of these attributes must be non-negative integers, and `minutes` must furthermore be between 0 and 59, inclusive.

Second, to create a new Timer object, one does a call like `Timer(119, 55)` (= five days minus 5 minutes), which returns the ID of a newly created `Timer` object.

2.1. First task.

On the next page there is some code.[2]

(1) Below it, draw all variables and objects that are created by the code, using the "folder" notation we've introduced in class. Lightly cross out any values that change and put their new values next to the cross-outs.
(2) Next to each print statement, write down what you think its output is.

We've started off the drawings for you.

Lab authors: D. Gries, L. Lee, S. Marschner, W. White

[1] But if you don't manage finish during lab, here are the alternate checkoff opportunities: (a) at ACCEL Green room consulting hours, listed at `http://www.cs.cornell.edu/courses/cs1110/2018sp/about/staff.php` , **from today until Tue Mar 6 inclusive**, (b) at non-professorial TA office hours **from today to Wed Mar 7 3:45pm inclusive**, although at TA office hours, questions about course material or assignments take precedence over lab check-offs; or (c) during the **first 10 minutes of your next scheduled lab (Tue Mar 6 or Wed Mar 7)**. 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.

[2] This happens to be the contents of file `timer_script.py`, but you do not need to open that file if you don't want to. We have provided it in case you want to enter it into Python Tutor, should you have trouble with this part of the lab.

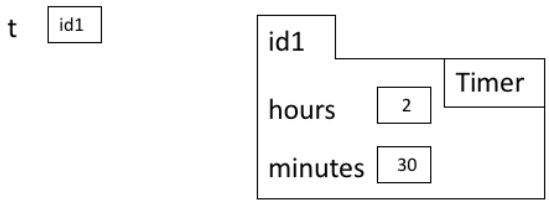
Think of `t` and `s` as the time two students have left until a deadline.

```
from timer import Timer

t = Timer(2,30)
# This is like grouping two students on CMS, so that their deadlines are linked
s = t
t.minutes = 20
print("t.minutes is: " + str(t.minutes))
print("s.minutes is: " + str(s.minutes) + '\n')

# This is like un-grouping the two students on CMS, so their deadlines are
# now no longer linked, even if they have the same values.
s = Timer(t.hours, t.minutes)
print("t.minutes is: " + str(t.minutes))
print("s.minutes is: " + str(s.minutes) + '\n')

t.minutes = 10
print("t.minutes is: " + str(t.minutes))
print("s.minutes is: " + str(s.minutes))
```



2.2. **Second task.** Complete function `add_timers` in file `lab05.py`.

```
def add_timers(timer1, timer2):
    """Returns: A new Timer object representing the sum of the times in
    timer1 and timer2.  Does not alter timer1 or timer2.

    Example: if timer1 represents 1 hr 59 min and timer2 represents 1 hr 2 min,
    then the **new** returned Timer represents 3 hr 1 min.

    Preconditions:
        timer1 is a Timer object
        timer2 is a Timer object"""
    pass  # PLACEHOLDER: REPLACE WITH YOUR IMPLEMENTATION.
```

You can test your function by running Python on `lab05_test.py` on the command line – we've provided some test cases in that file for you.

3. Pig Latin

Pig Latin is a simple[3] 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'`, `'qu'` becomes `'quay'`.

(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.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_test.py` on the command line — we've provided some test cases for you.

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