CS 1110, LAB 5: OBJECTS AND CONDITIONALS

http://www.cs.cornell.edu/courses/cs1110/2016fa/labs/lab05/

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

You have now had an extensive assignment that made you an expert of string slicing. However, none of the functions in that assignment required conditionals. This lab builds upon those skills, and gives you experience writing more complex functions involving conditionals.

This lab also have some questions reguarding objects. You are going to get a lot more experience with them on Assignment 3. To prepare you for that assignment, we want you to write some functions for a new class that you have never seen before.

The coding exercises in this lab are prime exam questions. In fact, both the Pig Latin question and the Time object question come from Prelim 1 in a previous semester of CS 1110.

Lab Materials. We have created several Python files for this lab. You can download all of the from the Labs section of the course web page.

http://www.cs.cornell.edu/courses/cs1110/2016fa/labs

For today's lab you will notice two files.

- `lab05.py` (a module with functions to implement)
- `timeclass.py` (a module with the `Time` class)

Once again you should create a *new* directory on your hard drive and download all of the files into that directory. Alternatively, you can get all of the files bundled in a single ZIP file called `lab05.zip` from the Labs section of the course web page.

Getting Credit for the Lab. Once again, you have a choice between getting credit through the online system or your instructor. The online lab is available at the web page

http://www.cs.cornell.edu/courses/cs1110/2016fa/labs/lab05/

The advantage of the online system is that you can do it on your own time and verify that you got credit. The disadvantage is that your answers must be correct. If you want more guided help on this lab, you should use the worksheet instead. Despite the demands of the online system, labs are graded on effort, not correctness.

If you use this worksheet, your answer will include both a sheet of paper (or the sheet provided to you in lab) and the file `lab05.py`. When you are finished you should show both to your lab instructor, who will record that you did it.

As with all previous labs, if you do not finish during the lab, you have **until the beginning of lab next week to finish it**. You should always do your best to finish during lab hours.

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

1. Pig Latin

Pig Latin is childish encoding of English that adheres to the following rules:

(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 English 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) If the English 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) If the English 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 `'ssssh'` becomes `'sssshay'`.

Your goal is to write a function `pigify` that take a single English word (e.g. a string with only letters and no spaces), and converts it into Pig Latin.

1.1. **The Function `first_vowel()`.** To aid with our Pig Latin conversion, we have provided a helper function `first_vowel(w)`, with the following specification:

```
def first_vowel(w):
    """Returns: position of the first vowel; -1 if no vowels.

    Parameter w: the word to check
    Precondition: w is a nonempty string with only lowercase letters"""
```

We hope that this helper function is correct. To verify this, write down at least 8 key test cases. **We do not want you to write a test script or try to fix any bugs**. Just write down the test cases.

1.2. **The Function `pigify`.** The function `pigify()` has a short-and-simple specification:

```
def pigify(w):
    """Returns: copy of w converted to Pig Latin.

    Parameter w: the word to convert to Pig Latin
    Precondition: w is a nonempty string with only lowercase letters"""
```

This specification assumes that you have read the definition of Pig Latin on the previous page. **Implement this function** in `lab05.py`

When you are done, you will want to test your answer. Instead of creating a unit test, we only want you to write down a list of test cases to verify that your implementation is correct.

2. The Time Class

The objects of class `Time` have exactly two attributes, `minutes` and hours. Both of these attributes are integers, though there is an important restriction on `minutes`: it must be between 0 and 59. If you go over 59 `minutes`, you are supposed to increase the `hours` attribute instead.

Open up the Python interactive shell and type in the following two statements:

```
>>> from timeclass import Time
>>> t = Time(2,30)
```

You should fill out the tables below, just as you did in the second lab on assignment statements. In the first table you are to determine the value of the expression or a command. If it is a command, you should just write either "None" or "error" (if the command causes an error). In the second table, you should guess the variable or literal that makes the second column true.

Statement; Expression	Expected Value	Calculated Value
`t.minutes`		
`t.hours`		
`t.days`		
`s = t`		
`s.minutes`		
`t.minutes = 20`		
`s.minutes`		
`s.minutes = 60`		

Statement	True Expression	Contents in the Box
`x = Time(2, [ ])`	`x.minutes==35`	
`y = Time([ ],10)`	`y.hours == 5`	
`y.[ ] = 25`	`y.hours == 5`	
`z = [ ]`	`id(z) == id(x)`	
`z.minutes = [ ]`	`x.minutes==15`	
`z = [ ]`	`id(z) == id(y)`	
`z.hours = [ ]`	`y.hours == 3`	
`w = str([ ])`	`w == '2:15'`	

3. The Function `add_time(time1,time2)`

At the end of the file `lab05.py`, you will see a stub for a function called `add_time`. This function has the following specification:

```
def add_time(time1, time2):
    """Returns: The sum of time1 and time2 as a new Time object

    Example:  Sum of 1 hr 59 min and 1 hr 2 min is 3 hr 1 min
    DO NOT ALTER time1 or time2, even though they are mutable

    Parameter time1: The starting time
    Precondition: time1 is a Time object

    Parameter time2: The time to add
    Precondition: time2 is a Time object"""
```

Because you are creating a new Time object, this function will need to call the constructor for Time. Implement this function inside `lab05.py`. This time we do not need you to list any test cases. However, it might be a good idea to test the function before showing it to your instructor. Once you have implemented this function, you are done with the lab.