This assignment is worth 30 points (3.0% of the course grade) and must be completed and turned in before 11:59 on Monday, September 24.

Assignment Overview
In this assignment you will practice with strings, conditionals and loops (for, while).

Background
In many languages, there are games that people play to make normal speech sound incomprehensible, except for a few people who are part of the game. Instead of creating a completely new language, certain sounds are added to words following rules only known to those who are playing, so that anyone else listening will hear only “gibberish” or nonsense words. For some basic examples, see http://en.wikipedia.org/wiki/Gibberish_(language_game).

Project Specification
We are going to create some simple rules for translating normal English into Gibberish. A common rule is to add sounds to each syllable, but since syllables are difficult to detect in a simple program, we’ll use a rule of thumb: every vowel denotes a new syllable. Since we are adding a Gibberish syllable to each syllable in the original words, we must look for the vowels.

To make things more unique, we will have two different Gibberish syllables to add. The first Gibberish syllable will be added to the first syllable in every word, and a second Gibberish syllable will be added to each additional syllable. For example, if our two Gibberish syllables were “ib” and “ag”, the word “program” would translate to “pribogradam.”

In some versions of Gibberish, the added syllable depends on the vowels in a word. For example, if we specify “*b” that means we use the vowel in the word as part of the syllable: e.g. “dog” would become “dobog” (inserting “ob” where the “*” is replaced by the vowel “o”) and “cat” would become “cabat” (inserting “ab” where “a” is used). Note that the “*” can only appear at the beginning of the syllable (to make your programming easier).

After the Gibberish syllables are specified, prompt the user for the word to translate. As you process the word, make sure you keep track of two things. First, if the current letter is a vowel, add a Gibberish syllable only if the previous letter was not also a vowel. This rule allows us to approximate syllables: translating “weird” with the Gibberish syllable “ib” should become “wibeird”, not “wibeibird”. Second, if we’ve already added a Gibberish syllable to the current word, add the secondary syllable to the remaining vowels. How can you use Booleans to handle these rules?

Finally, print the Gibberish word. Afterwards, ask the user if they want to play again, and make sure their response is an acceptable answer (“yes”/“no”, “y”/“n”)

Your program will:
1. Print a message explaining the game.
2. Prompt for two Gibberish syllables (indicate the allowed wildcard character “*”).
3. Prompt for a word to translate.
4. Process the word and add the syllables where appropriate.
5. Print the final word, and ask if the user wants to play again.

Deliverables
You must use Handin to turn in the file proj03.py – this is your source code solution; be sure to include your section, the date, the project number and comments describing your code. Please be sure to use the specified file name, and save a copy of your proj03.py file to your H: drive as a backup.

Notes and Hints:
You should start with this program, as with all programs, by breaking the program down into parts. Each of the four steps will be worth 6 points so get each step working before moving on to the next (and hand in after each step).
1. Getting started: Simplify!
   a) First solve the program using a single Gibberish syllable, without checking for multiple vowels in a row or using the wildcard (“*”). This means that “weird” with the “ib” syllable will become “wibeibird” (that will not be correct in the final version, but good enough for starting your program: simplify!).
   b) You will need to decide how you will check for vowels. Good possibilities are string indexing or using a Boolean, but use a method that works best with your own program. When you check for vowels it may be handy to create a string vowels = “aeiouAEIOU” and use in vowels to check if a character is a vowel (is the character in the string named vowels).
   c) In this simplified version assume that the Gibberish syllable is exactly two characters long.
   d) For your resulting word start with an empty string and add characters onto it as you process characters from the original word.
2. After that, add the second Gibberish syllable and allow Gibberish syllables longer than two characters.
   a) A Boolean to keep track of whether you have already made a substitution for the first Gibberish syllable will be useful, e.g. done_with_first_vowel = False
   b) I found slicing useful for longer Gibberish syllables.
3. Add the wildcard ability after you’ve completed the above steps.
4. Finally, add handling of the special case of consecutive vowels. For most people this is the hardest part of the program. It doesn’t add much Python code, but until you see it the logic can be elusive. I used a variable named previous_character that held the previous character so when I was looking at a new character I could check whether the previous character had been a vowel.

The string library has a couple of useful tools. If you add import string at the beginning of your program, string.digits and string.ascii_letters are strings that contain all the digits (0 through 9) and all the letters (uppercase and lowercase).

Example Output:

(Make sure to explain rules!)
Enter your first Gibberish syllable (add * for the vowel substitute): i3
Syllable must only contain letters or a wildcard ('*'): ip
Enter the second Gibberish syllable (* for vowel substitute): *zz
Please enter a word you want to translate:
--> Gibberish
Your final word: Gipibbezzerizzish

Play again? (y/n) m
Please enter y to continue or n to quit: n

Thanks for playing!