# Pig Latin

## *Part I: PigLatinize single words*

## Basic Rules

Pig Latin is a pretend language. The basic rule is simple: move all the letters that appear before the first vowel to the end of the word and append “ay”.

Write your algorithm to return the basic-rules PigLatinized string:

Take first letter, check if it is a vowel. If it is, append “way”. If it isn’t, find the index of the first vowel, then create a substring of consonants from the beginning to that index. Remove from beginning and append to end. Add “ay”. If vowel is not found, return “\*\*\*\* NO VOWEL \*\*\*\*”.

pig 🡪 igpay

latin 🡪 atinlay

this 🡪 isthay

strange 🡪 angestray

However, if the first letter is a vowel, append “way”.

apple 🡪 appleway

a 🡪 away

upper 🡪 upperway

If the word has no vowel, return “\*\*\*\* NO VOWEL \*\*\*\*”

sfghjkl 🡪 \*\*\*\* NO VOWEL \*\*\*\*

Consider using charAt, indexOf, contains, and substring, and others. The shell PigLatin.java

defines these helpful constants:

String punct = ",./;:'\"?<>[]{}|`~!@#$%^&\*()";  
 String letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";  
 String vowels = "AEIOUaeiou";

## Special Cases

If “qu” is before a vowel, “qu” is treated like a consonant. There are two ways to think about this. Whichever way you do it, make sure that a first-letter “u” still works.

Write your algorithm for “qu”:

If starts with “q”. If it does, check if the next character is a “u”. Append “qu” and add “ay”

Write your algorithm for “y”:

If it starts with a “y”, remove from beginning, add to end, and append “ay”.

Write your test to see if the first letter is capitalized:

If it is uppercase, append to the end and convert to lowercase. Append “ay”. Convert the first letter to uppercase.

|  |
| --- |
| question 🡪 estionquay |
| squeeze 🡪 eezesquay  esquire 🡪 esquireway |

If the first letter is a “y” then treat it as a consonant; if the first vowel is a “y”, treat it as a vowel.

yellow 🡪 ellowyay

rhyme 🡪 ymerhay

eye 🡪 eyeway

## Capitalization

If the first letter is capitalized, then the PigLatin word’s first letter is capitalized and the original first letter is in lower case. Ignore capital letters in the middle of the original word; they should remain capitalized in the PigLatin word.

Thomas 🡪 Omasthay

Question 🡪 Estionquay

McDonald 🡪 OnaldmcDay

I 🡪 Iway

BUBBLES 🡪 UBBLESbay

## Punctuation

Punctuation occurring before the word should be removed, stored, and prepended to the piglatinized word. Punctuation occurring after the word should be removed, stored, and appended to the piglatinized word. Any punctuation that is in the middle of the word is to be treated as a regular letter.

Write your algorithm to remove and store punctuation occurring before the word:

Store all punctuation up to the first letter in a String. Then starting from the end store all the punctuation up to the last error in a String. Append the first punctuation to the word before the word, then append the end punctuation in reverse.

what? 🡪 atwhay?

Oh!!! 🡪 Ohway!!!

“hello” 🡪 “ellohay”

([Hello]) 🡪 ([Ellohay])

don’t 🡪 on’tday

“pell-mell” 🡪 “ell-mellpay”

## Make more test cases similar to the ones above. Please don’t waste your time making up crazy test cases!

## *Part II: PigLatinize each word in a Text File*

Open PigLatin.java. In the main, comment out part\_1\_using\_pig(); and comment in part\_2\_using\_piglatenizeFile();

The shell prompts the user to enter the name of an input text file and the name of the output text file. The input file will contain English words and the output file will contain PigLatin words.

A Scanner object reads in the filename. Another Scanner object and a File object read the text file. A PrintWriter object and a FileWriter object work together to output the file. We use try-catch blocks to catch the exceptions.

You are to process the input file into PigLatin, preserving the line and paragraph structure of the input file. There are several ways to preserve the line structure, but they all use nested loops. You may want to consider using nextLine and split, Scanner, or StringTokenizer. Use these files for testing:

PigLatin.txt

LittleFrog.txt

declaration.txt

**Sample** (using LittleFrog.txt)

Input Output

|  |  |
| --- | --- |
| I have a little frog, His name is Tiny Tim.  I put him in the bathtub, to see if he could swim. | Iway avehay away ittlelay ogfray, Ishay amenay isway Inytay Imtay. Iway utpay imhay inway ethay athtubbay, otay eesay ifway ehay ouldcay imsway. |

## Extension

Write a pigReverse method that returns an already PigLatinized word in reverse, preserving before-and-after punctuation and adjusting an initial upper case letter. For example,

|  |
| --- |
| pigReverse("Atwhay!?") returns the String Yahwta!? |
| pigReverse("{(Ellohay!)}") returns the String {(Yaholle!)}!? |
| pigReverse("\"OnaldmcDay???\"" returns the String "YaDcmdlano???" |