Project “I am my own Shakespeare” Overview
This program is a Markov text synthesizer that takes learns all of Shakespeare’s sonnets and generates a sonnet based on the frequency that Shakespeare uses those words. I used a website (http://www.shakespeares-sonnets.com/all.php) as the data source and made a histogram that analyzes the frequency each word is used. I hoped to be able to create a sonnet that would imitate Shakespeare’s sonnets, using the same words etc.

Implementation
I used the function plaintext from pattern.web module in order to convert the data from my data source into just readable text. After that, I heavily utilized .strip and .split in order to process my data into text that can be analyzed easily. After that I easily made a histogram that analyzed the frequency of each word and put that data in a dictionary.

A large component of this program is the syllable function that counts the number of syllables in a word. I had a list of vowels and a list of dipthongs that were used to check the components of a word to count syllables in accordance to certain rules I implemented. With the English language, however, there are many exceptions to any language rule. Thus, the rules were roughly constructed to count the number of syllables of most words correctly. I could have refined the rules more or allowed for more exceptions in my code; however, due to time constraints, I left it as is.

Another integral function in this program is the final function called Shakespeare that basically defines the syntax of a Shakespearean sonnet, such as 10 syllables per line and 14 lines in a sonnet. Shakespearean sonnets are also known for iambic pentameter. Because of the difficulty level and amount of time needed in order to find stressed and unstressed syllables accurately to a significant degree, I did not incorporate iambic pentameter in my generated sonnet.

Results*]*
After sorting the histogram, I found that the more frequent the word, the less significant of a word it is. For example, the word “and” appeared 490 times in 154 sonnets. Likewise, “the” appeared 441 times, “to” appeared 415 times, “thou” appeared 235 times, “thy” appeared 280 times, etc.

Examples of the sonnets generated:

“wherein and owners better all bids
me and mark dear i and and light to thy
and from clouds it glass me compile muse
hold chose my from pursuit of the in
as sell bad gone only that do grace a
the have it the world which the that am thine
in acquaintance look thou die i
time do cry of outbraves shade winds do
foot invention would be thy air or
guilty from tender me flattered
dost my thoughts help child thy he better my
all grieved but and as his will for
as with turns return fairer the do
pine that placed for the of a where is”
“doth drooping you canst self purging his
of world nor thee up thou afar a the night
vices shape for high men fever of to
hath ill if begins the kill what the proud

```
injury this art what mock true your
stand although seal when poets see tender poor
find thy is pent heavenly show still when
receives say how worth having
how the living so totter'd against
must give of gainst liker from mine so
been your winter golden forgot want
which and times not grief all that to
year thou his our thou so this broke to
beloved is after do dote thus do”

“will make in are thee like mine greeing and that
thee forgot dost stone seconds my that unkind
you to it your that as of burn
thou or thy desired have the be no
my maladies mad to legacy
and blind clears do prophecies use but
when in how in yet love's true art
beauty the i hearts and thou o mind
the loves or heart not the need thee sunken
great light heart old chopp'd coming thy why which
day becoming doth to put'st like what
muse heart hungry pilgrimage so those sweet
loves thou beauteous mine be so
west absent can to bereft a found stick'st”
```

Reflection
Of the two significant functions in this program, the last function worked very well in which it shaped the entire final product. Had I more time, though, I would have wanted to further refine the function that counts syllables in a word. I might have incorporated more exceptions to more accurately count syllables. Beyond that, I’d like to have been able to find stressed and unstressed syllables. From the generated sonnets, we can see that the words, while they fit within the syntax for a Shakespearean sonnet, they don’t make sense together as a sonnet. I wish I knew that Markov text synthesizer was a lot harder than I had initially thought. I might have spent more time on the project. Overall, I am pleased with this program.