Topic 2a - Elements of Good Code
Part 3, Application of your Primer

Pig Latin

- **Variable naming** - We did not like the variable naming in program 1b and 1c because the names were not meaningful and were hard to follow.
- **Comments** - Program 1c did a nice job using comments to help follow what was happening throughout the program. Do you actually need all of those comments? If they would have used more meaningful variable names, they could have cut down on the comments that were used. They did a nice job of explaining what the functions were above their code.
- **Structure** - At first glance, program 1a looks nice and clean.

Reading Level

- **Consistency** - We prefered 2b over 2c in this category as they were more specific in the names that they used throughout their program.
- **Comments** - We liked the way they used comments in 2b over 2a because the comments were above instead of of off to the side.
- **Variable naming** - 2a and 2b used meaningful/specific names, while 2c was more abbreviated which made it more difficult to figure out what was going on.
Purpose: Read a text file and display the most commonly used words.
Author: Ben Schafer

```python
def countWords(filename):
    remove='.,?()!"'
    wordcount = {}
    fin = open(filename,"r")
    for line in fin:
        for rem in remove:
            line=line.replace(rem,"")
        line=line.strip()
        words=line.split()
        if len(words)>0:
            for w in words:
                w=w.lower()
                if w in wordcount.keys():
                    wordcount[w]+=1
                else:
                    wordcount[w]=1
    fin.close()
    return wordcount

#Remove stop words
def deleteStopWords(wordcount):
    fin = open("stopWords.txt","r")
    for line in fin:
        line=line.strip()
        if line in wordcount.keys():
            wordcount.pop(line)
    return wordcount

#Find the top 40 words
#clean this to an alphabetical list
#also mark the sizes needed
def makeTagCloud(filename):
    wordcount = countWords(filename)
    deleteStopWords(wordcount)
    lyst = []
    for key in wordcount.keys():
        tup = [wordcount[key],key]
        lyst.append(tup)
    lyst.sort()
    lyst.reverse()
    end=min(40,len(lyst))
    highCount=lyst[0][0]
    lowCount=lyst[end-1][0]
    alpha=[]
    print("Most common words were")
    print(lyst[:50])
    for x in range(0,end):
        alpha.append((lyst[x][1],lyst[x][0]))
    alpha.sort()
    body="""for word,cnt in alpha:
        body = body + makeHTMLword(word,cnt,highCount,lowCount)
    box = makeHTMLbox(body)
    printHTMLfile(box,filename[:-4])"

import string
```
# Reorganized so there were no extra blank lines between code than necessary, but kept two blank lines in between the different function codes.

def printHTMLfile(body, title):
    ""
    """ + title + """
    <html> <head> <title>" + title + """ + title + """ + title + """ + title + """
    </head> <body> <html> """ + title + """
    """
    fd.write(theStr)
    fd.close()

def makeHTMLbox(body):
    boxStr = ""
    width: 300px;
    background-color: rgb(250, 250, 250);
    text-align: center;"""
    return boxStr % (body)

# make a word with a font size to be placed in the box.
# Font size is scaled between htmlBig and htmllittle (to be user set).
# high and low represent the high and low counts in the document.
# cnt is the cnt of the word
def makeHTMLword(word, cnt, high, low):
    htmlBig = 96
    htmlLittle = 14
    ratio = (cnt-low)/float(high-low)
    fontsize = htmlBig*ratio + (1-ratio)*htmlLittle
    fontsize = int(fontsize)
    wordStr = '<span style="font-size:%spx;">%s</span>' % (str(fontsize), word)
    return wordStr % (str(fontsize), word)

# example usage
def demo():
    pairs = [('hi', 5), ('there', 6), ('mom', 10), ('fred', 2), ('bill', 20)]
    highCount=20
    lowCount=2
    body ='
    for word, cnt in pairs:
        body = body + makeHTMLword(word, cnt, highCount, lowCount)
    box = makeHTMLbox(body)
    printHTMLfile(box, 'testFile')