Create a function remove\_punct that takes arguments line and punct\_list

Assign 0 to idx

While idx < length of line

For char in line

If char is in punct\_list

Line = line.replace(char,””)

Else

Add one to idx

Return line.lower()

Create a function next\_word that takes arguments line and begin\_idx

Word = “”

While idx is less than length of line

If line at begin\_idx does not equal “ “ and begin\_idx is less than length of line

Add char to word

Add one to begin\_idx

Else

Add one to beg\_idx

Return word,beg\_idx

Create a function isStop that takes arguments word and stop\_list

If word is in stop\_list

Return True

Else

Return False

Create a function called convert\_to\_dict that takes arguments file\_name and fileName\_dict

assign the opening of filename for reading to inFile + “.txt”

for line in inFile:

idx = 0

newLine = remove\_punt(line)

while idx is less than the length of newline

nextWord,idx = next\_word(newline, idx)

if the opposite of isStop:

if word in fileName\_dict

add one to fileName\_dict at word

else

set fileName\_dict at word to one

create a function called total\_words that takes argument speech\_dict:

totalWords = 0

for word in speech\_dict

add word[-1] to totalWords

return totalWords

create a function called shared\_words that takes arguments comp\_speech and speech\_dict:

commonWords is 0

for word in speech\_dict

if word[0] in comp\_speech

add one to commonWords

return commonWords

create a function called find\_rms\_freq that takes arguments speech\_dict and comp\_speech and sharedWords

freq\_sum is 0

total1 = total\_words(speech\_dict)

total2 = total\_words(comp\_speech)

for word in speech dict

if word[0] in comp\_speech

add square root of the absolute value of word[-1] divided by total1 – comp\_speech[word[0]][-1] divided by total2 to freq\_sum

return the square root of freq\_sum divided by sharedWords

Make dictionarys for all of the texts called fileName\_dict and add them to a list called dict\_list

file\_list = [“mystery1”,”mystery2”,’mystery3’,”mystery4”,”Romney”,”Obama”,”Clinton”,”Trump”]

for dict,index in enumerate dict\_list

convert\_to\_dict(file\_list at index, dict)

outerCounter is 0

innerCounter is 4

while outerCounter is less than or equal to 3

word\_commonality is []

rel\_freq is []

speech\_dict is dict\_list at outerCounter

while innerCounter is less than or equal to 8

comp\_speech is dict\_list at innerCounter

sharedWords is shared\_words(comp\_speech,speech\_dict)

distinctWords is length of comp\_speech + length of speech\_dict – sharedWords

add sharedWords/distinctWords to word\_commonality

add find\_rms\_freq(speech\_dict,comp\_speech,sharedWords) to rel\_freq

assign max of word\_commonality to mostCommon

assign the index of mostCommon to commIndex

assign the min of rel\_freq to bestFreq

assign the index of bestFreq to freqIndex

print(“The text {} has the highest word commonality with {} {}” formatted with file\_list at outerCounter, file\_list at commIndex + 4, mostCommon)

print(“The text {} has the highest frequency similarity with {} {}” formatted with file\_list at outerCounter, file\_list at freqIndex + 4, bestFreq)