**Appendix:**

**Programs Written:**

**Soundex.py**

# Two implementations of Soundex algorithm

def soundexNaive(name, len=4):

'''A naive implementation of the soundex algorithm.

By Deepak Kumar, 2/12/2008. dkumar@brynmawr.edu

Expects name to be in uppercase.

'''

sndx = name[0] # Keep first letter

for i in name[1:]:

if i in "BFPV":

d = "1"

elif i in "CGJKQSXZ":

d = "2"

elif i in "DT":

d = "3"

elif i == "L":

d = "4"

elif i in "MN":

d = "5"

elif i == "R":

d = "6"

else:

d = ''

# Do not add if repeated/duplicated

if d != sndx[-1]:

sndx += d

return (sndx + (len \* '0'))[:len] # first 4 letters or padd with 0 to make 4

def soundex(name, len=4):

""" soundex module conforming to Donald Knuth's algorithm

implementation 2000-12-24 by Gregory Jorgensen

public domain.

Expects name to be in uppercase.

Modified by Deepak Kumar, 2/11/2008. dkumar@cs.brynmawr.edu

Modified by Deepak Kumar, 3/21/2017. dkumar@cs.brynmawr.edu

"""

# Define soundex values for each letter, 0 means omitted

# ABCDEFGHIJKLMNOPQRSTUVWXYZ

digits = '01230120022455012623010202'

# retain first letter of string

sndx = name[0]

# translate each successive letter in name

for c in name[1:]:

d = digits[ord(c)-ord('A')]

# If 2 or more letters with same # are adjacent then just keep one

if d != '0' and d != sndx[-1]:

sndx += d

# remove all 0s from the soundex code

sndx = sndx.replace('0','')

# return soundex code padded to len characters

return (sndx + (len \* '0'))[:len]

def main():

person = input("Enter your name: ")

print("Algorithm 1 result for", person, soundexNaive(person.upper(), len(person)))

print("Algorithm 2 result for", person, soundex(person.upper(), len(person)))

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Metaphone.py**

import re

def metaphone(name):

'''Finds the Metaphone value for a word. Note that only the letters A-Z are

supported, so any language-specific processing should be done beforehand.

BAsed on algorithm described by Lawrence Phillips in Computer Language

Volume 7, No. 12, December 1990, pp 39-43.

This version is based on one originally written by

Paul Battley (version 0.4, 2005-04-18, Alpha). pbattley@gmail.com

Adapted for Python by Deepak Kumar, 2/11/2008. dkumar@cs.brynmawr.edu

'''

RULES = [

# Regexp, replacement

[ r'([bcdfhjklmnpqrstvwxyz])\1+',r'\1' ],# Remove doubled consonants except g.

[ '^ae', 'E' ], # ae -> E

[ '^[gkp]n', 'N' ], # initial kn-, gn-, pn- -> N

[ '^wr', 'R' ], # initial wr- -> R

[ '^x', 'S' ], # x- -> S

[ '^wh', 'W' ], # initial wh- -> W

[ 'mb$', 'M' ], # -mb (as in dumb) -> M

[ '(?!^)sch', 'SK' ], # sch -> SK

[ 'th', '0' ], # 0 represents the th sound

[ 't?ch|sh', 'X' ], # tch, tsh, ch, sh -> X

[ 'c(?=ia)', 'X' ], # cia -> X

[ '[st](?=i[ao])', 'X' ], # stia, stio -> X

[ 's?c(?=[iey])', 'S' ], # ci, ce, cy, sci, sce, scy -> S

[ '[cq]', 'K' ], # c, q -> K

[ 'dg(?=[iey])', 'J' ], # dgi, dge, dgy -> J (as in ledger, edgy)

[ 'd', 'T' ], # d -> T

[ 'g(?=h[^aeiou])', '' ], # gh -> silent (gh- not at end or before vowel)

[ 'gn(ed)?', 'N' ], # gne, gnd -> N

[ '([^g]|^)g(?=[iey])',r'\1J' ], # gi, ge, gy, but not gg -> J

[ 'g+', 'K' ], # g, gg -> K (as in egg)

[ 'ph', 'F' ], # ph -> F

[ r'([aeiou])h(?=\b|[^aeiou])',r'\1' ], # silent h if after vowel and no following vowels

[ '[wy](?![aeiou])', '' ], # wy is silent if not followed by vowel

[ 'z', 'S' ], # z -> S

[ 'v', 'F' ], # v -> F

[ '(?!^)[aeiou]+', '' ], # vowels silent unless first letter

]

# Normalise case and remove non-ASCII

name = name.lower()

s = re.sub('[^a-z]', '', name)

# Apply the Metaphone rules

for (rx, rep) in RULES:

s = re.sub(rx, rep, s)

return s.upper()

def main():

person = input("Enter your name: ")

print("Metaphone result for", person, metaphone(person))

if \_\_name\_\_ == '\_\_main\_\_':

main()

**modifiedurl.py**

import urllib.request

f = open('FemaleNames.txt', 'w')

local\_filename, headers = urllib.request.urlretrieve('https://cs.brynmawr.edu/Courses/cs330/spring2018/FemaleNames.\

txt')

html = open(local\_filename)

line = html.readline()

while line:

#print(line)

f.write(line)

line = html.readline()

f.close()

**SoundexOnFile.py**

import Soundex

f = open("FemaleNamesSortedUnique.txt", "r")

f2 = open("FemaleNamesMatched.txt", "w")

line = f.readline()

while line:

lineSoundex = Soundex.soundexNaive(line.upper(), len(line))

f2.write(lineSoundex + ' ' + line)

line = f.readline()

f.close()

f2.close()

**squash.py**

f = open("FemaleNamesSortedMatchedSoundex.txt", "r")

f2 = open("FemaleNamesSortedMatchedSoundexCompressed.txt", "w")

line1 = f.readline()

line2 = f.readline()

list1 = line1.split()

list2 = line2.split()

f2.write(list1[1] + ' ')

while line1:

if len(list1) > 0 and len(list2) > 0 and line1 and list2:

while len(list1) > 0 and len(list2) > 0 and list1[0] == list2[0]:

f2.write(list2[1] + ' ')

line2 = f.readline()

list2 = line2.split()

# get a new line, since the list value is no longer the same

line1 = line2

list1 = list2

f2.write('\n')

else:

break

f.close()

f2.close()

**analyze.py**

f = open("soundexClasses.txt", "r")

f2 = open("analysisSoundex.txt", "w")

line1 = f.readline()

count = 0

while line1:

list1 = line1.split()

count += 1

length = len(list1)

f2.write(str(length) + " " + line1)

line1 = f.readline()

print("Lines in File: " + str(count))

f.close()

f2.close()

**MetaphoneOnFile.py**

import Metaphone

f = open("FemaleNamesSortedUnique.txt", "r")

f2 = open("FemaleNamesMatchedMetaphone.txt", "w")

line = f.readline()

while line:

lineMetaphone = Metaphone.metaphone(line)

f2.write(lineMetaphone + ' ' + line)

line = f.readline()

f.close()

f2.close()

I used the same squash and analyze programs for metaphone’s analysis, but changed the title of the files read and written to as described, below.

**How Programs Were Used in the Lab Exercises:**

1. Got FemaleNames.txt by using modifiedurl.py to read website contents into a .txt file
2. Ran a tr -d [:punct:] < FemaleNames.txt |tr A-Z a-z|sort|uniq >FemaleNamesSortedUnique.txt command to get FemaleNamesSortedUnique.txt which had all names as unique and as lower case
3. Ran SoundexOnFile.py and MetaphoneOnFile.py, respectively, to get the signature name pairings of the encoding to the name. This was output into FemaleNamesMatchedSoundex.txt and FemaleNamesMatchedMetaphone.txt, respectively.
4. I then sorted the resulting .txt file. sort FemaleNamesMatchedSoundex.txt > FemaleNamesSortedMatchedSoundex.txt. The equivalent for Metaphone, too.
5. Then I ran squash.py for each text file to create space-separated lines of names which sound alike for each algorithm.
6. Finally, I ran the awk -f filter.awk outputFromStep5 > soundexClasses.txt and MetaphoneClasses.txt, respectively. This is what gave me the information.
7. Finally, I wrote a short analyze.py file that looked at the contents of a file and printed out the sought information. The output was a .txt file that had the number of line entries to the left and the words to the right.
8. I then sorted the output file to get the top ten classes.