Anagrams & Programs

By Kevin Ramirez, Marla Anderson, and Christina Tetrick

Introduction

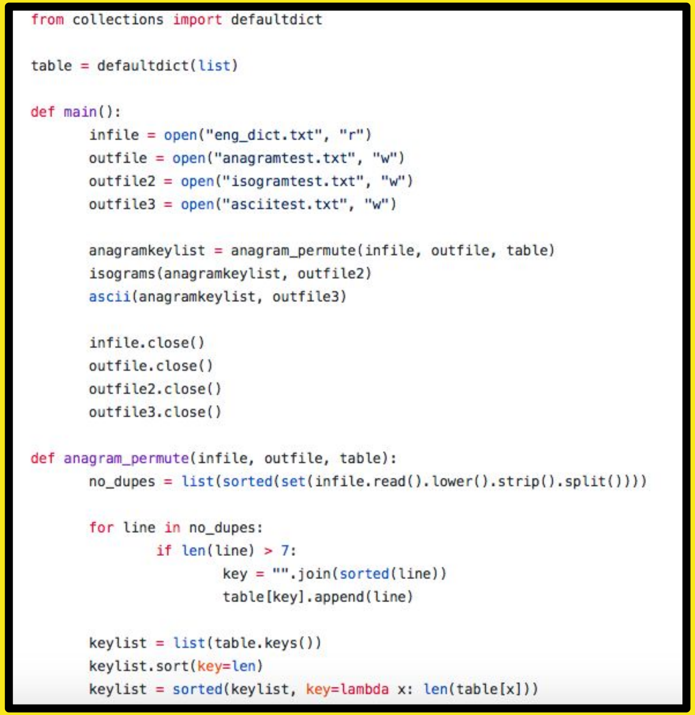
The purpose of this project was to explore the potential anagrams a program could generate when given a dictionary with a set of words.

The Algorithm

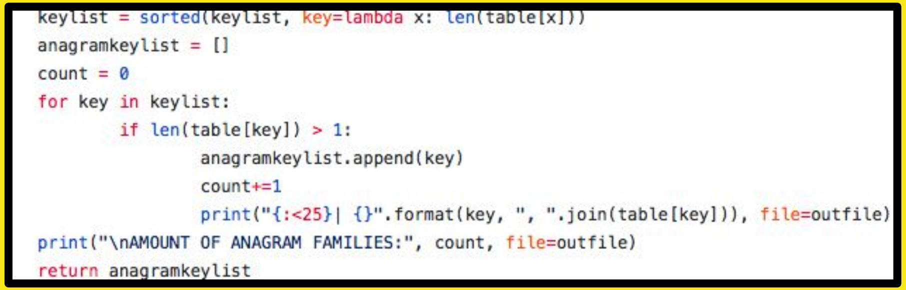
Our algorithm hinges upon the use of a default dictionary. A default dictionary, unlike a typical dictionary, is able to assign a value to a nonexistent key. This makes it ideal for our anagrams because several values are being attached to one key.

Our Code

- Reads through the eng_dict file
    - Generates Anagrams and eliminates words shorter than 8
- Table[key].append places those anagrams in a keylist
    - Orders anagrams based on length
- Prints anagrams and number of anagrams generated



```
from collections import defaultdict

table = defaultdict(list)

def main():
        infile = open("eng_dict.txt", "r")
        outfile = open("anagramtest.txt", "w")
        outfile2 = open("isogramtest.txt", "w")
        outfile3 = open("asciitest.txt", "w")

        anagramkeylist = anagram_permute(infile, outfile, table)
        isograms(anagramkeylist, outfile2)
        ascii(anagramkeylist, outfile3)

        infile.close()
        outfile.close()
        outfile2.close()
        outfile3.close()

def anagram_permute(infile, outfile, table):
        no_dupes = list(sorted(set(infile.read().lower().strip().split())))

        for line in no_dupes:
                if len(line) > 7:
                        key = "".join(sorted(line))
                        table[key].append(line)

        keylist = list(table.keys())
        keylist.sort(key=len)
        keylist = sorted(keylist, key=lambda x: len(table[x]))
```



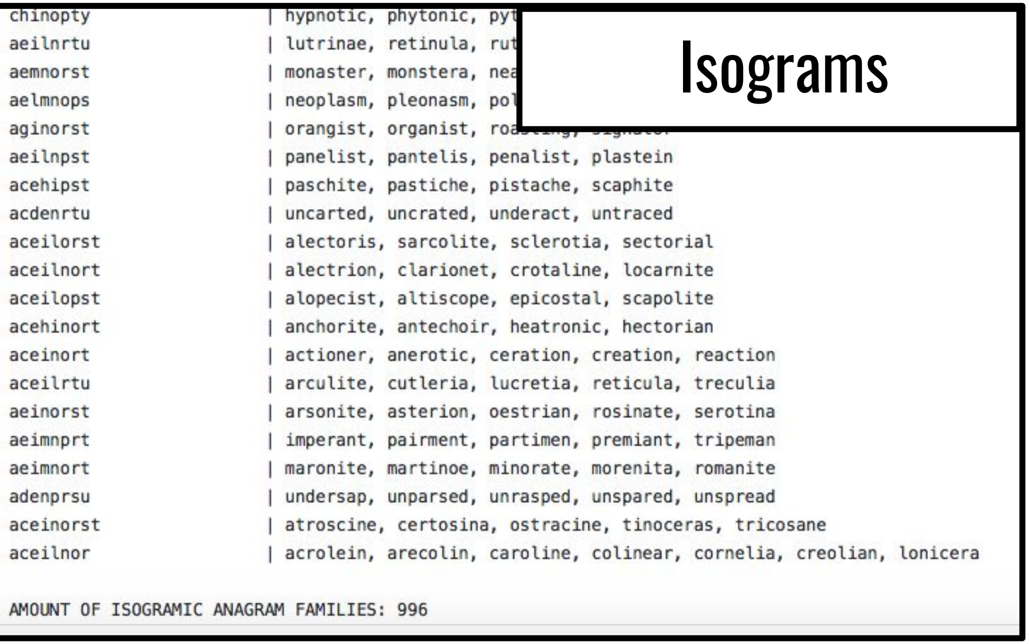
```
keylist = sorted(keylist, key=lambda x: len(table[x]))
anagramkeylist = []
count = 0
for key in keylist:
        if len(table[key]) > 1:
                anagramkeylist.append(key)
                count+=1
                print("{:<25}| {}".format(key, ", ".join(table[key])), file=outfile)
print("\nAMOUNT OF ANAGRAM FAMILIES:", count, file=outfile)
return anagramkeylist
```

Cool Findings

Several cool things were discovered during this project. First, the generated anagrams does not contain any palindromes, and there is no symmetry. Second, there are 996 isograms contained within the list. Third, the anagrams can be translated into ASCII values.

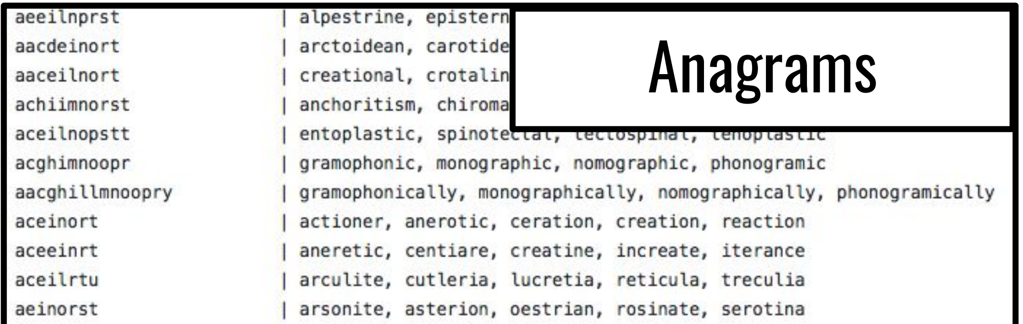
Examples of Cool Findings

Isograms



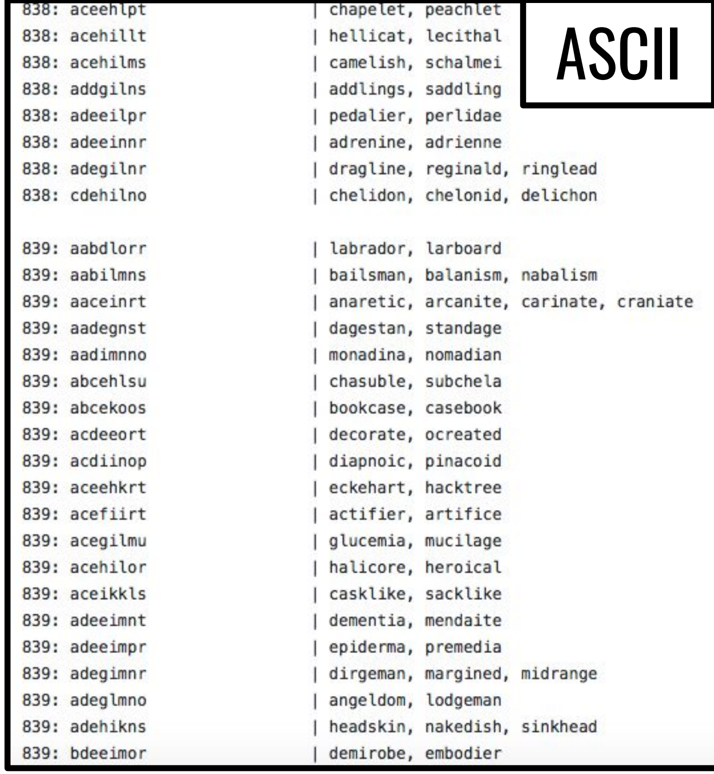
```
chinopty                | hypnotic, phytonic, pyt
aeilnrtu                | lutrinae, retinula, rut
aemnorst                | monaster, monstera, nea
aelmnops                | neoplasm, pleonasm, pol
aginorst                | orangist, organist, roa
aeilnpst                | panelist, pantelis, penalist, plastein
acehipst                | paschite, pastiche, pistache, scaphite
acdenrtu                | uncarted, uncrated, underact, untraced
aceilorst               | alectoris, sarcolite, sclerotia, sectorial
aceilnort               | alectrion, clarionet, crotaline, locarnite
aceilopst               | alopecist, altiscope, epicostal, scapolite
acehinort               | anchorite, antechoir, heatronic, hectorian
aceinort                | actioner, anerotic, ceration, creation, reaction
aceilrtu                | arculite, cutleria, lucretia, reticula, treculia
aeinorst                | arsonite, asterion, oestrian, rosinate, serotina
aeimnprt                | imperant, pairment, partimen, premiant, tripeman
aeimnort                | maronite, martinoe, minorate, morenita, romanite
adenprsu                | undersap, unparsed, unrasped, unspared, unspread
aceinorst               | atroscine, certosina, ostracine, tinoceras, tricosane
aceilnor                | acrolein, arecolin, caroline, colinear, cornelia, creolian, lonicera

AMOUNT OF ISOGRAMIC ANAGRAM FAMILIES: 996
```

Anagrams



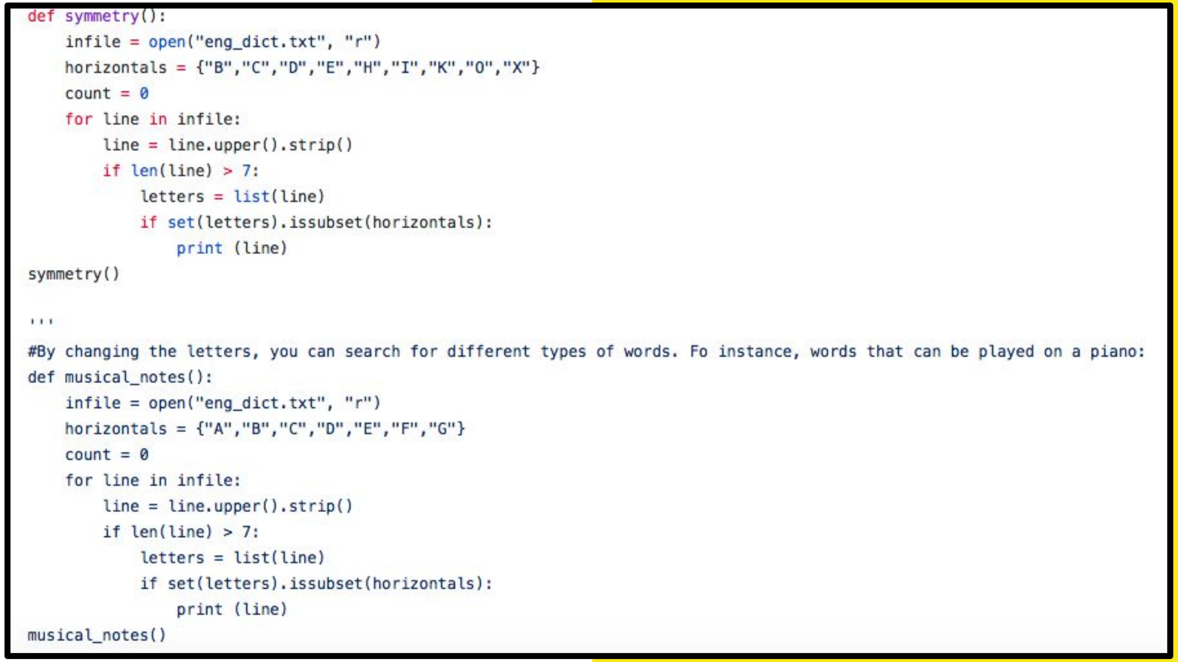
```
aeeilnprst              | alpestrine, epistern
aacdeinort              | arctoidean, carotide
aaceilnort              | creational, crotalin
achiimnorst             | anchoritism, chiroma
aceilnopstt             | entoplastic, spinotectal, tectospinal, tenoplastic
acghimnoopr             | gramophonic, monographic, nomographic, phonogramic
aacghillmnoopry         | gramophonically, monographically, nomographically, phonogramically
aceinort                | actioner, anerotic, ceration, creation, reaction
aceeinrt                | aneretic, centiare, creatine, increate, iterance
aceilrtu                | arculite, cutleria, lucretia, reticula, treculia
aeinorst                | arsonite, asterion, oestrian, rosinate, serotina
```

ASCII

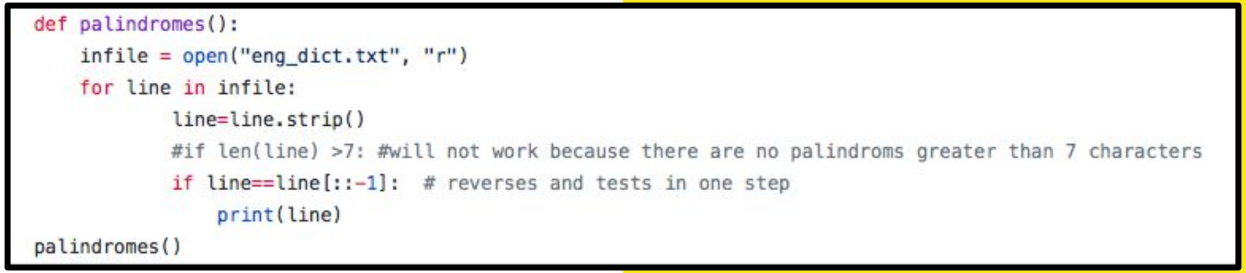


```
838: aceehlpt           | chapelet, peachlet
838: acehillt           | hellicat, lecithal
838: acehilms           | camelish, schalmei
838: addgilns           | addlings, saddling
838: adeeilpr           | pedalier, perlidae
838: adeeinnr           | adrenine, adrienne
838: adegilnr           | dragline, reginald, ringlead
838: cdehilno           | chelidon, chelonid, delichon

839: aabdlorr           | labrador, larboard
839: aabilmns           | bailsman, balanism, nabalism
839: aaceinrt           | anaretic, arcanite, carinate, craniate
839: aadegnst           | dagestan, standage
839: aadimnno           | monadina, nomadian
839: abcehlsu           | chasuble, subchela
839: abcekoos           | bookcase, casebook
839: acdeeort           | decorate, ocreated
839: acdiinop           | diapnoic, pinacoid
839: aceehkrt           | eckehart, hacktree
839: acefiirt           | actifier, artifice
839: acegilmu           | glucemia, mucilage
839: acehilor           | halicore, heroical
839: aceikkls           | casklike, sacklike
839: adeeimnt           | dementia, mendaite
839: adeeimpr           | epiderma, premedia
839: adegimnr           | dirgeman, margined, midrange
839: adeglmno           | angeldom, lodgeman
839: adehikns           | headskin, nakedish, sinkhead
839: bdeeimor           | demirobe, embodier
```

Symmetry & Palindromes

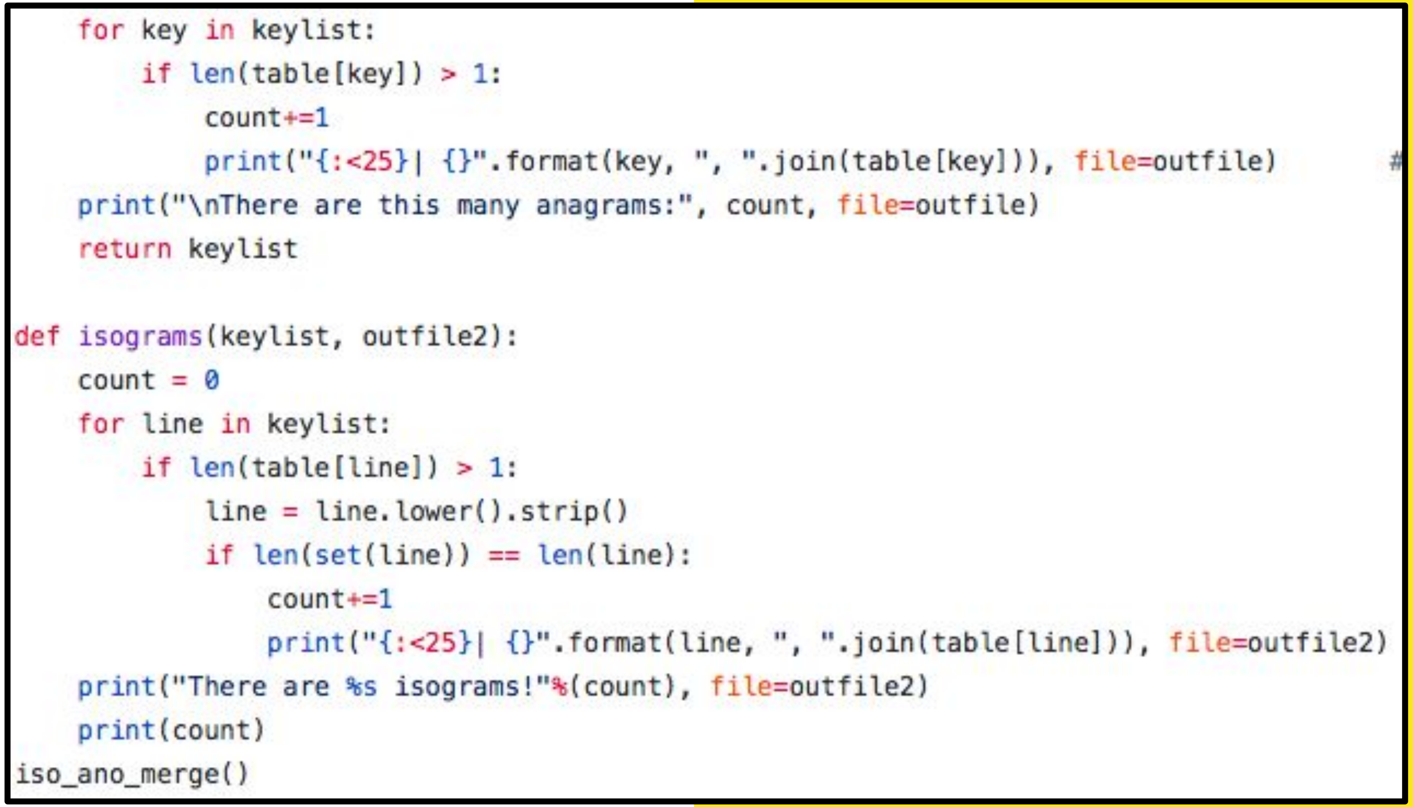


```
def symmetry():
    infile = open("eng_dict.txt", "r")
    horizontals = {"B","C","D","E","H","I","K","O","X"}
    count = 0
    for line in infile:
        line = line.upper().strip()
        if len(line) > 7:
            letters = list(line)
            if set(letters).issubset(horizontals):
                print (line)
symmetry()

'''
#By changing the letters, you can search for different types of words. Fo instance, words that can be played on a piano:
def musical_notes():
    infile = open("eng_dict.txt", "r")
    horizontals = {"A","B","C","D","E","F","G"}
    count = 0
    for line in infile:
        line = line.upper().strip()
        if len(line) > 7:
            letters = list(line)
            if set(letters).issubset(horizontals):
                print (line)
musical_notes()
```



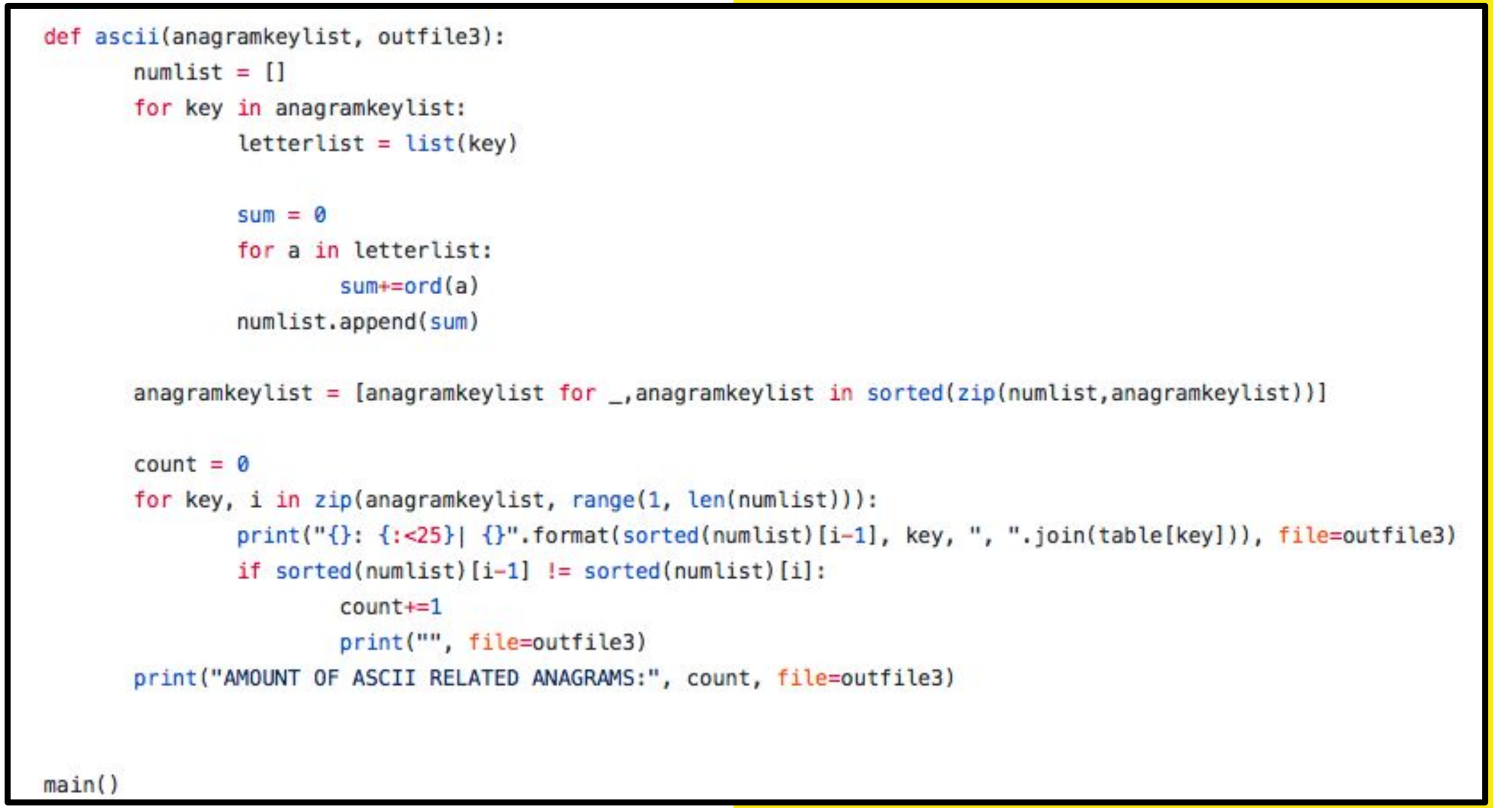
```
def palindromes():
    infile = open("eng_dict.txt", "r")
    for line in infile:
            line=line.strip()
            #if len(line) >7: #will not work because there are no palindroms greater than 7 characters
            if line==line[::-1]:  # reverses and tests in one step
                print(line)
palindromes()
```

Isograms



```
    for key in keylist:
        if len(table[key]) > 1:
            count+=1
            print("{:<25}| {}".format(key, ", ".join(table[key])), file=outfile)        #
    print("\nThere are this many anagrams:", count, file=outfile)
    return keylist

def isograms(keylist, outfile2):
    count = 0
    for line in keylist:
        if len(table[line]) > 1:
            line = line.lower().strip()
            if len(set(line)) == len(line):
                count+=1
                print("{:<25}| {}".format(line, ", ".join(table[line])), file=outfile2)
    print("There are %s isograms!"%(count), file=outfile2)
    print(count)
iso_ano_merge()
```

ASCII Values



```python
def ascii(anagramkeylist, outfile3):
        numlist = []
        for key in anagramkeylist:
                letterlist = list(key)

                sum = 0
                for a in letterlist:
                        sum+=ord(a)
                numlist.append(sum)

        anagramkeylist = [anagramkeylist for _,anagramkeylist in sorted(zip(numlist,anagramkeylist))]

        count = 0
        for key, i in zip(anagramkeylist, range(1, len(numlist))):
                print("{}: {:<25}| {}".format(sorted(numlist)[i-1], key, ", ".join(table[key])), file=outfile3)
                if sorted(numlist)[i-1] != sorted(numlist)[i]:
                        count+=1
                        print("", file=outfile3)
        print("AMOUNT OF ASCII RELATED ANAGRAMS:", count, file=outfile3)


main()
```

Conclusions and Future Directions

- Linguistics
  - Other potential letter or word combinations
- Computer Science
  - Behavior of anagrams when translated into ASCII code