jupyter Assignment3 Last Checkpoint: Last Friday at 5:29 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

Anon Suphatphon 62011090

Exercise 1

Write a program that reads words.txt and prints only the words with more than 20 characters(not counting whitespaces).

In [3]:
```
file = open("words.txt")
for line in file:
    words = line.strip()
    if len(words) > 19:
        print(words)
```

```
counterdemonstration
counterdemonstrations
counterdemonstrators
hyperaggressivenesses
hypersensitivenesses
microminiaturization
microminiaturizations
representativenesses
```

jupyter Assignment3 Last Checkpoint: Last Friday at 5:29 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

Exercise 2

In 1939 Ernest Vincent Wright published a 50,000 word novel called Gadsby that does not contain the letter “e”. Since “e” is the most common letter in English, that’s not easy to do.In fact, it is difficult to construct a solitary thought without using that most common symbol. It is slow going at first, but with caution and hours of training you can gradually gain facility.Write a functioncalled has_no_e that returns True if the given word doesn’t have the letter “e” in it.Write a program that reads words.txt and prints only the words that have no “e”. Compute the percentage of words in the list that have no “e”.

In [3]:
```
def has_no_e(word):
    for letter in word:
        if letter == 'e':
            return False
    return True


fin = open('words.txt')
count = 0
num_of_words = 113809

for line in fin:
    word = line.strip()
    if has_no_e(word) == True:
        count += 1
        print(word)

print(count / num_of_words * 100, '%')
```

```
for line in fin:
    word = line.strip()
    if has_no_e(word) == True:
        count += 1
        print(word)

print(count / num_of_words * 100, '%')
```

```
zorillas
zorillo
zorillos
zorils
zounds
zoysia
zoysias
zucchini
zucchinis
zygoma
zygomas
zygomata
zygosis
zygosity
zygotic
zymology
zymosis
zymotic
zymurgy
33.05626092839758 %
```

Exercise 3

Write a function named avoidsthat takes a word and a string of forbidden letters, and that returns Trueif the word doesn't use any of the forbidden letters. Write a program that prompts the user to enter a string of forbidden letters and then prints the number of words that don't contain any of them.

In [10]:
```
def avoids(strg, word):
    for letter in strg:
        if letter in word:
            return False;
    return True;

fin = open('words.txt')
def No_Contain(strg):
    count_word_no_contain = 0;
    for line in fin:
        result = avoids(strg, line.strip());
        if(result == True):
            count_word_no_contain += 1;
            print(line.strip());
    return count_word_no_contain;

print(avoids("HUY STRAUSS", "LION"));
strg = "lion king";
print("The number of words doesn't have the letters of the string", strg, "is: ", No_Contain(strg));
```

```
zebrasses
```

```
        if(result == True):
            count_word_no_contain += 1;
            print(line.strip());
    return count_word_no_contain;

print(avoids("HUY STRAUSS", "LION"));
strg = "lion king";
print("The number of words doesn't have the letters of the string", strg, "is: ", No_Contain(strg));
```

```
zebrasses
zebu
zebus
zed
zeds
zee
zees
zephyr
zephyrs
zest
zested
zests
zesty
zeta
zetas
zymase
zymases
zyme
zymes
The number of words doesn't have the letters of the string lion king is:  9969
```

Exercise 4

Write a function named uses_only that takes a word and a string of letters, and that returns Trueif the word contains only letters in the list. Can you make a sentence using only the letters acefhlo? Other than "Hoe alfalfa"?

In [5]:

```
def use_only(text,seek):
    for letter in text:
        if letter not in seek:
            return False
    return True
emlist = []
output = []
seek = 'acefhlo'
with open('words.txt','r') as data:
    emlist = data.read().split()
for word in emlist:
    if use_only(word,seek) == True:
        output.append(word)
print("Seeked words are: {}".format(seek))
print("Number of words found: {}".format(len(output)))
print("Founded words include: {}".format(output))
```

```
Seeked words are: acefhlo
Number of words found: 188
Founded words include: ['aa', 'aah', 'aal', 'ace', 'ache', 'achoo', 'ae', 'aff', 'ah', 'aha', 'ahchoo', 'ala', 'alae', 'alcohol', 'ale', 'alec',
'alee', 'alef', 'alfa', 'alfalfa', 'all', 'allele', 'allheal', 'aloe', 'aloha', 'aloof', 'cacao', 'cache', 'caeca', 'caecal', 'cafe', 'caleche', 'calf',
```

```
    for letter in text:
        if letter not in seek:
            return False
    return True
emlist = []
output = []
seek = 'acefhlo'
with open('words.txt','r') as data:
    emlist = data.read().split()
for word in emlist:
    if use_only(word,seek) == True:
        output.append(word)
print("Seeked words are: {}".format(seek))
print("Number of words found: {}".format(len(output)))
print("Founded words include: {}".format(output))
```

```
Seeked words are: acefhlo
Number of words found: 188
Founded words include: ['aa', 'aah', 'aal', 'ace', 'ache', 'achoo', 'ae', 'aff', 'ah', 'aha', 'ahchoo', 'ala', 'alae', 'alcohol', 'ale', 'alec',
'alee', 'alef', 'alfa', 'alfalfa', 'all', 'allele', 'allheal', 'aloe', 'aloha', 'aloof', 'cacao', 'cache', 'caeca', 'caecal', 'cafe', 'caleche', 'calf',
'call', 'calla', 'ceca', 'cecal', 'cee', 'cell', 'cella', 'cellae', 'cello', 'chafe', 'chaff', 'chalah', 'chaleh', 'challah', 'chef', 'chela', 'chelae',
'cholla', 'clach', 'clef', 'cloaca', 'cloacae', 'cloacal', 'cloche', 'coach', 'coal', 'coala', 'coalhole', 'coca', 'coccal', 'cochlea', 'cochlea
e', 'coco', 'cocoa', 'coff', 'coffee', 'coffle', 'coho', 'col', 'cola', 'cole', 'coo', 'cooch', 'cooee', 'coof', 'cool', 'each', 'eche', 'echo', 'ec
ole', 'eel', 'ef', 'eff', 'efface', 'eh', 'el', 'elf', 'ell', 'fa', 'face', 'faecal', 'fall', 'fallal', 'falloff', 'feal', 'fecal', 'fee', 'feel', 'fell', 'fella', 'fell
ah', 'felloe', 'feoff', 'feoffee', 'flea', 'fleche', 'flee', 'fleece', 'fleech', 'floc', 'floe', 'foal', 'focal', 'foe', 'foh', 'fool', 'ha', 'haaf', 'hae',
'hah', 'halala', 'halalah', 'hale', 'half', 'hall', 'hallah', 'hallel', 'hallo', 'halloa', 'halloo', 'halo', 'haole', 'he', 'heal', 'heel', 'hell', 'hellh
ole', 'hello', 'ho', 'hoe', 'hole', 'holla', 'hollo', 'holloa', 'holloo', 'hooch', 'hoof', 'la', 'lac', 'lace', 'lall', 'lea', 'leach', 'leaf', 'leal', 'lec
h', 'lee', 'leech', 'lo', 'loach', 'loaf', 'loca', 'local', 'locale', 'loch', 'loco', 'locofoco', 'loll', 'loo', 'loof', 'loofa', 'loofah', 'oaf', 'oca', 'o
e', 'of', 'off', 'offal', 'oh', 'oho', 'ole', 'olea', 'oleo', 'olla', 'ooh']
```

Exercise 5

Write a functionnamed uses_all that takes a word and a string of required letters,and that returns Trueif the word uses all the required letters at least once. How many words arethere that use all the vowels aeiou? How about aeiouy?

In [7]:
```
def use_all(text,seek):
    for letter in text:
        if letter not in seek:
            return False
    return True
emlist = []
output = []
seek = 'aeiou'
with open('words.txt','r') as data:
    emlist = data.read().split()
for word in emlist:
    if use_all(word,seek) == True:
        output.append(word)
print("Seeked words are: {}".format(seek))
print("Number of words found: {}".format(len(output)))
print("Founded words include: {}".format(output))
```

```
Seeked words are: aeiou
Number of words found: 5
Founded words include: ['aa', 'ae', 'ai', 'eau', 'oe']
```

Exercise 6

Write a function called is_abecedarian that returns True if the letters in a wordappear in alphabetical order (double letters are ok). How many abecedarian words are there?

In [6]:
```
def is_abcdian(text):
    pin = 0
    while pin < len(text)-1:
        if text[pin] > text[pin+1]:
            return False
        else:
            pin += 1
    return True
emlist = []
output = []
with open('words.txt','r') as data:
    emlist = data.read().split()

for word in emlist:
    if is_abcdian(word) == True:
        output.append(word)
print("Seeked words are: {}".format(seek))
print("Number of words found: {}".format(len(output)))
print("Founded words include: {}".format(output))
```


```
    emlist = data.read().split()

for word in emlist:
    if is_abcdian(word) == True:
        output.append(word)
print("Seeked words are: {}".format(seek))
print("Number of words found: {}".format(len(output)))
print("Founded words include: {}".format(output))
```


```
Seeked words are: acefhlo
Number of words found: 596
Founded words include: ['aa', 'aah', 'aahs', 'aal', 'aals', 'aas', 'abbe', 'abbes', 'abbess', 'abbey', 'abbot', 'abet', 'abhor', 'abhors',
'ably', 'abo', 'abort', 'abos', 'abuzz', 'aby', 'accent', 'accept', 'access', 'accost', 'ace', 'acers', 'aces', 'achoo', 'achy', 'act', 'ad', 'ad
d', 'adder', 'adders', 'adds', 'adeem', 'adeems', 'adept', 'adios', 'adit', 'ado', 'adopt', 'ados', 'ads', 'adz', 'ae', 'aegis', 'aery', 'aff',
'affix', 'afflux', 'afoot', 'aft', 'agin', 'agio', 'agios', 'agist', 'aglow', 'agly', 'ago', 'ah', 'ahoy', 'ai', 'ail', 'ails', 'aim', 'aims', 'ain', 'ain
s', 'air', 'airs', 'airt', 'airy', 'ais', 'ait', 'all', 'allot', 'allow', 'alloy', 'alls', 'ally', 'almost', 'alms', 'alow', 'alp', 'alps', 'alt', 'am', 'ammo',
'ammos', 'amort', 'amp', 'amps', 'amu', 'an', 'annoy', 'ant', 'any', 'apt', 'ar', 'ars', 'art', 'arty', 'as', 'ass', 'at', 'aw', 'ay', 'be', 'bee',
'beef', 'beefily', 'beefs', 'beefy', 'been', 'beep', 'beeps', 'beer', 'beers', 'beery', 'bees', 'beet', 'befit', 'beg', 'begin', 'begins', 'begi
rt', 'begot', 'begs', 'beknot', 'bel', 'bell', 'bellow', 'bells', 'belly', 'below', 'bels', 'belt', 'ben', 'benny', 'bens', 'bent', 'berry', 'best',
'bet', 'bevy', 'bey', 'bhoot', 'bi', 'bijou', 'bijoux', 'bill', 'billow', 'billowy', 'bills', 'billy', 'bin', 'bins', 'bint', 'bio', 'biopsy', 'bios', 'birr',
'birrs', 'bis', 'bit', 'bitt', 'bitty', 'bloop', 'bloops', 'blot', 'blotty', 'blow', 'blowy', 'bo', 'boo', 'boor', 'boors', 'boos', 'boost', 'boot', 'b
ooty', 'bop', 'bops', 'bort', 'borty', 'bortz', 'bos', 'boss', 'bossy', 'bot', 'bott', 'bow', 'box', 'boxy', 'boy', 'buy', 'buzz', 'by', 'cee', 'ce
es', 'ceil', 'ceils', 'cell', 'cello', 'cellos', 'cells', 'celt', 'cent', 'cess', 'chi', 'chill', 'chills', 'chilly', 'chimp', 'chimps', 'chin', 'chino', 'chin
os', 'chins', 'chintz', 'chip', 'chippy', 'chips', 'chirr', 'chirrs', 'chis', 'chit', 'chitty', 'chivvy', 'chivy', 'choosy', 'chop', 'choppy', 'chop
s', 'chott', 'chow', 'cist', 'city', 'civvy', 'cloot', 'clop', 'clops', 'clot', 'clotty', 'cloy', 'coo', 'coop', 'coops', 'coopt', 'coos', 'coot', 'co
p', 'cops', 'copy', 'cos', 'coss', 'cost', 'cosy', 'cot', 'cow', 'cowy', 'cox', 'coy', 'coz', 'crux', 'cry', 'de', 'dee', 'deem', 'deems', 'deep',
'deeps', 'deer', 'deers', 'dees', 'defi', 'defis', 'deft', 'defy', 'dehort', 'dei', 'deil', 'deils', 'deist', 'deity', 'del', 'dell', 'dells', 'delly', 'de
ls', 'demo', 'demos', 'demy', 'den', 'dens', 'dent', 'deny', 'derry', 'des', 'dev', 'dew', 'dewy', 'dex', 'dey', 'dhow', 'dill', 'dills', 'dill
y', 'dim', 'dims', 'din', 'dins', 'dint', 'dip', 'dippy', 'dips', 'dipt', 'dirt', 'dirty', 'dit', 'ditty', 'divvy', 'do', 'doer', 'doers', 'dopy', 'dor'
```