FILE_PROGRAM_VHA

January 17, 2024

1 1.file Filtering. Display all lines of a file, except those that start with a pound sign (#), the comment character for Python, Perl, Tcl, and most other scripting languages.

Extra credit: Also strip out comments that begin after the first character.

[3]:
```
f=open("Friends.txt")
d=open("vishal.txt","w")

while(True):
    buff=f.readline()
    if len(buff)!=0:
        if buff[0]=="#":
            continue
        else:

            if "#" in buff:
                for index in range (1,len(buff)):
                    if buff[index]=="#":
                        d.write(buff[0:index])
            else:
                d.write(buff)
    else:
        break

f.close()
d.close()
c=open("vishal.txt")
print(c.read())
```

```
Friends are crazy, Friends are naughty !
Friends  6 are honest, Friends are  best !
Friends are like keygen, friends are like license key !
new We are nothing without friends, Life is not possible without friends !
```

2 2.Write a program to compare two text files. If they are different, give the line and column numbers in the files where the first difference occurs.

[4]:
```
f1 = open("friends.txt", "r")
f2 = open("dost.txt", "r")

fileOne = f1.readlines()
fileTwo = f2.readlines()

f1.close()
f2.close()
outFile = open("file3.txt", "w")
x = 0

for i in fileOne:
   if i != fileTwo[x]:
      outFile.write(i+" <> "+fileTwo[x])
   x += 1
outFile.close()
d=open("file3.txt")
print(d.read())
```

```
Friends  6 are honest, Friends are  best !
 <> Friends are honest, Friends are  best !
new We are nothing without friends, Life is not possible without friends ! <> We
are nothing without friends, Life is not possible without friends !
```

[7]:
```
f1 = open("friends.txt", "r")
f2 = open("dost.txt", "r")

fileOne = f1.readlines()
fileTwo = f2.readlines()

f1.close()
f2.close()


for index,line in enumerate(fileOne):
   if line!= fileTwo[index]:
    for ind,char in enumerate(line):
        if char!=fileTwo[index][ind]:
            print("line number",index+1,"colum",ind+1)
            break
    break
```

```
line number 2 colum 9
```

3 3.Write a “pager” program. Your solution should prompt for a filename, and display the text file 25 lines at a time, pausing each time to ask the user to “press a key to continue.”

[ ]:
```
def pagetext(text_lined, num_lines=25):
   for index,line in enumerate(text_lined):
       if index % num_lines == 0 and index:
           X=input("Hit any key to continue press q to quit")
           if X.lower() != 'q':
                print(line)
           if X.lower() == 'q':
                break

       else:
               print (line)
text_lined=open(input("enter file name: "))
pagetext(text_lined,25)
```

```
enter file name: bhai.txt
.
```

File Filtering. Display all lines of a file, except those that start with a pound sign (#), the comment character for Python, Perl, Tcl, and most other scripting languages.

Extra credit: Also strip out comments that begin after the first character.

9-2.

File Access. Prompt for a number N and file F, and display the first N lines of F.

9-3.

File Information. Prompt for a filename and display the number of lines in that text file.

9-4.

File Access. Write a "pager" program. Your solution should prompt for a filename, and display the text file 25 lines at a time, pausing each time to ask the user to "press a key to continue."

9-5.

Test Scores. Update your solution to the test scores problems (Exercises 5-3 and 6-4) by allowing a set of test scores be loaded from a file. We leave the file format to your discretion.

9-6.

File Comparison. Write a program to compare two text files. If they are different, give the line and column numbers in the files where the first difference occurs.

9-7.

Parsing Files. Win32 users: Create a program that parses a Windows .ini file. POSIX users: Create a program that parses the /etc/services file. All other platforms: Create a program that parses a system file with some kind of structure to it.

9-8.

Module Introspection. Extract module attribute information. Prompt the user for a module name (or accept it from the command line). Then, using dir() and other built-in functions, extract all its attributes, and display their names, types, and values.

Hit any key to continue press q to quitd

9-9.

"PythonDoc." Go to the directory where your Python standard library modules are located. Examine each .py file and determine whether a __doc__ string is available for that module. If so, format it properly and catalog it. When your program has completed, it should present a nice list of those modules that have documentation strings and what they are. There should be a trailing list showing

which modules do not have documentation strings (the shame list). Extra credit: Extract documentation for all classes and functions within the standard library modules.

9-10.

Home Finances. Create a home finance manager. Your solution should be able to manage savings, checking, money market, certificate of deposit (CD), and similar accounts. Provide a menu-based interface to each account as well as operations such as deposits, withdrawals, debits, and credits. An option should be given to a user to remove transactions as well. The data should be stored to file when the user quits the application (but randomly during execution for backup purposes).

9-11.

Web site Addresses.

Write a URL bookmark manager. Create a text-driven menu-based application that allows the user to add, update, or delete entries. Entries include a site name, Web site URL address, and perhaps a one-line description (optional). Allow search functionality so that a search "word" looks through both names and URLs for possible matches. Store the data to a disk file when the user quits the application, and load up the data when the user restarts.

(b) Upgrade your solution to part (a) by providing output of the bookmarks to a legible and syntactically correct HTML file (.htm or .html) so that users can then point their browsers to this output file and be presented with a list of their bookmarks. Another feature to implement is allowing the creation of "folders" to allow grouping of related bookmarks. Extra credit: Read the literature on regular expressions and the Python re module. Add regular expression validation of URLs that users enter into their databases.

9-12.

Users and Passwords.

Do Exercise 7-5, which keeps track of usernames and passwords. Update your code to support a "last login time" (7-5a). See the documentation for the time module to obtain timestamps for when users "log in" to the system.

Also, create the concept of an "administrative" user that can dump a list of all the users, their passwords (you can add encryption on top of the passwords if you wish [7-5c]), and their last login times (7-5b).

The data should be stored to disk, one line at a time, with fields delimited by colons (:), e.g., "joe:boohoo:953176591.145", for each user. The number of lines in the file will be the number of users that are part of your system.

Further update your example such that instead of writing out one line at a time, you pickle the entire data object and write that out instead. Read the documentation on the pickle module to find out how to flatten or serialize your object, as well as how to perform I/O using picked objects. With the addition of this new code, your solution should take up fewer lines than your solution in part (a).

4 4.Text Processing. You are tired of seeing lines on your e-mail wrap because people type lines that are too long for your mail reader application. Create a program to scan a text file for all lines longer than 80 characters. For each of the offending lines, find the closest word before 80 characters and break the line there, inserting the remaining text to the next line (and pushing the previous next line down one). When you are done, there should be no lines longer than 80 characters.

[10]:
```
f=open('bhai.txt','r')

lis=[]
def ding(a):

    if len(a)<=80:
```

```
        lis.append(a)
        return
    else:
        if a[79]==' ':

            lis.append(a[:80])
            ding(a[80:])

        elif a[79]!=' ':
            ind=a.rfind(' ',0,79)

            lis.append(a[:ind+1])
            ding(a[ind+1:])

for x in f:
    if len(x)>80:
        ding(x)
    else:
        lis.append(x)

ty=open('9-16o.txt','w')
for x in lis:
    if x[-1]==' ':
        x=x[:-1]+'\n'
    else :
        x+='\n'
    ty.write(x)
f.close()
ty.close()
f=open("9-16o.txt")
print(f.read())
```

.

File Filtering. Display all lines of a file, except those that start with a pound sign (#), the comment character for Python, Perl, Tcl, and most other scripting languages.

Extra credit: Also strip out comments that begin after the first character.

9-2.

File Access. Prompt for a number N and file F, and display the first N lines of F.

9-3.

File Information. Prompt for a filename and display the number of lines in that text file.

9-4.

File Access. Write a "pager" program. Your solution should prompt for a filename, and display the text file 25 lines at a time, pausing each time to ask the user to "press a key to continue."

9-5.

Test Scores. Update your solution to the test scores problems (Exercises 5-3 and 6-4) by allowing a set of test scores be loaded from a file. We leave the file format to your discretion.

9-6.

File Comparison. Write a program to compare two text files. If they are different, give the line and column numbers in the files where the first difference occurs.

9-7.

Parsing Files. Win32 users: Create a program that parses a Windows .ini file. POSIX users: Create a program that parses the /etc/services file. All other platforms: Create a program that parses a system file with some kind of structure to it.

9-8.

Module Introspection. Extract module attribute information. Prompt the user for a module name (or accept it from the command line). Then, using dir() and other built-in functions, extract all its attributes, and display their names, types,

and values.

9-9.

"PythonDoc." Go to the directory where your Python standard library modules are located. Examine each .py file and determine whether a __doc__ string is available for that module. If so, format it properly and catalog it. When your program has completed, it should present a nice list of those modules that have documentation strings and what they are. There should be a trailing list showing which modules do not have documentation strings (the shame list). Extra credit: Extract documentation for all classes and functions within the standard library modules.

9-10.

Home Finances. Create a home finance manager. Your solution should be able to manage savings, checking, money market, certificate of deposit (CD), and similar accounts. Provide a menu-based interface to each account as well as operations such as deposits, withdrawals, debits, and credits. An option should be given to a user to remove transactions as well. The data should be stored to file when the user quits the application (but randomly during execution for backup purposes).

9-11.

Web site Addresses.

Write a URL bookmark manager. Create a text-driven menu-based application that allows the user to add, update, or delete entries. Entries include a site name, Web site URL address, and perhaps a one-line description (optional). Allow search functionality so that a search "word" looks through both names and URLs for possible matches. Store the data to a disk file when the user quits the application, and load up the data when the user restarts.

(b) Upgrade your solution to part (a) by providing output of the bookmarks to a legible and syntactically correct HTML file (.htm or .html) so that users can then point their browsers to this output file and be presented with a list of their bookmarks. Another feature to implement is allowing the creation of

"folders" to allow grouping of related bookmarks. Extra credit: Read the literature on regular expressions and the Python re module. Add regular expression validation of URLs that users enter into their databases.

9-12.

Users and Passwords.

Do Exercise 7-5, which keeps track of usernames and passwords. Update your code to support a "last login time" (7-5a). See the documentation for the time module to obtain timestamps for when users "log in" to the system.

Also, create the concept of an "administrative" user that can dump a list of all the users, their passwords (you can add encryption on top of the passwords if you wish [7-5c]), and their last login times (7-5b).

The data should be stored to disk, one line at a time, with fields delimited by colons (:), e.g., "joe:boohoo:953176591.145", for each user. The number of lines in the file will be the number of users that are part of your system.

Further update your example such that instead of writing out one line at a time, you pickle the entire data object and write that out instead. Read the documentation on the pickle module to find out how to flatten or serialize your object, as well as how to perform I/O using picked objects. With the addition of this new code, your solution should take up fewer lines than your solution in part (a).

Replace your login database and explicit use of pickle by converting your code to use shelve files. Your resulting source file should actually take up fewer lines than your solution to part (b) because some of the maintenance work is gone.

9-13.

Command-Line Arguments.

What are they, and why might they be useful?

Write code to display the command-line arguments which were entered.

9-14.

Logging Results. Convert your calculator program

(Exercise 5-6) to take input from the command line, i.e.,

$ calc.py 1 + 2

Output the result only. Also, write each expression and result to a disk file. Issuing a command of…

$ calc.py print

… will cause the entire contents of the "register tape" to be dumped to the screen and file reset/truncated. Here is an example session:

$ calc.py 1 + 2 3 $ calc.py 3 ^ 3 27 $ calc.py print 1 + 2 3 3 ^ 3 27 $ calc.py print $

Extra credit: Also strip out comments that begin after the first character.

9-15.

Copying Files. Prompt for two filenames (or better yet, use command-line arguments). The contents of the first file should be copied to the second file.

9-16.

Text Processing. You are tired of seeing lines on your e-mail wrap because people type lines that are too long for your mail reader application. Create a program to scan a text file for all lines longer than 80 characters. For each of the offending lines, find the closest word before 80 characters and break the line there, inserting the remaining text to the next line (and pushing the previous next line down one). When you are done, there should be no lines longer than 80 characters.

9-17.

Text Processing. Create a crude and elementary text file editor. Your solution is menu-driven, with the following options:

create file [prompt for filename and any number of lines of input],

display file [dump its contents to the screen],

edit file (prompt for line to edit and allow user to make changes),

save file, and

quit.

9-18.

Searching Files. Obtain a byte value (0-255) and a filename. Display the number of times that byte appears in the file.

9-19.

Generating Files. Create a sister program to the previous problem. Create a binary data file with random bytes, but one particular byte will appear in that file a set number of times. Obtain the following three values:

a byte value (0-255),

the number of times that byte should appear in the data file, and

the total number of bytes that make up the data file.

Your job is to create that file, randomly scatter the requested byte across the file, ensure that there are no duplicates, the file contains exactly the number of occurrences that byte was requested for, and that the resulting data file is exactly the size requested.

9-20.

Compressed Files. Write a short piece of code that will compress and decompress gzipped or bzipped files. Confirm your solution works by using the command-line gzip or bzip2 programs or a GUI program like PowerArchiver, StuffIt, and/or WinZip.

9-21.

ZIP Archive Files. Create a program that can extract files from or add files to, and perhaps creating, a ZIP archive file.

9-22.

ZIP Archive Files. The unzip -l command to dump the contents of ZIP archive is boring. Create a Python script called lszip.py that gives additional information such as: the compressed file size, the compressed percentage of each file (by comparing the original and compressed file sizes), and a full time.ctime() timestamp instead of the unzip output (of just the date and HH:MM). Hint: The date_time attribute of an archived file does not contain enough information to feed to time.mktime()… it is up to you!

5 5. Write a python program to create and read the city.txt file in one go and print the contents on the output screen.

[1]:
```
f=open("city.txt","w")
f.write("ahmedabad \n")
f.write("new york \n")
f.close()
f=open("city.txt","r")
h=f.read()
print(h)
```

```
ahmedabad
new york
```

6 6.Write a function count_lines() to count and display the total number of lines from the file. Consider the following lines for the file – friends.txt.

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

We are nothing without friends, Life is not possible without friends !

[6]:
```
def countline(f):
    #file_read=f.readlines()
    count=0
    for lines in f:
        count+=1
    print(count)

f=open("1.txt")
```

```
countline(f)
```

```
4
```

7 7.Write a function display_oddLines() to display odd number lines from the text file. Consider the following lines for the file – friends.txt.

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

We are nothing without friends, Life is not possible without friends !

[7]:
```
def oddline(f):
    #file_read=f.readlines()
    count=0
    for lines in f:
        if(count%2!=0):
            print(lines)
        count+=1


f=open("1.txt")
oddline(f)
```

```
Friends are honest, Friends are best !

We are nothing without friends, Life is not possible without friends !
```

8 8. Write a Python program to read a text file and do following: 1. Print no. statements 2. Print no. of words

[10]:
```
def oddline(f):
    #file_read=f.readlines()
    countLine=0
    words=0
    for lines in f:
        countLine+=1
        for word in lines.split():
            if word.isalpha()==True or word.isalnum()==True:
                words+=1
    print(countLine,words)

f=open("1.txt")
oddline(f)
```

```
4 28
```

9 9. Write a Python program to count words, characters and spaces from text file.

Friends are crazy , Friends are naughty !

Friends are honest , Friends are best !

[11]:
```
import string
def oddline(f):
    file_read=f.read()
    count=1
    word=1
    spaces=0
    special=0
    tab=0
    line=0
    char=0
    for lines in file_read:
        char+=1
        if lines==' ':
                spaces+=1
        if lines in string.punctuation:
                special+=1
        if lines=='\t':
                tab+=1
        if lines=="\n":
                count+=1
        #word=spaces-line-tab
        if lines==" " or lines=="\n" :
                 word+=1

    print(count,char-spaces-special-1,word-special,spaces,special,tab)

f=open("2.txt")
oddline(f)
```

```
2 62 12 14 4 0
```

10 10.Write a function cust_data() to ask user to enter their names and age to store data in customer.txt file.

[13]:
```
def cust_data():
    name=input("Enter customer name:")
    age=int(input("Enter customer age:"))
    data=str([name,age])
```

```
    f=open("customer.txt","a")
    f.write(data)
    f.close()
cust_data()
f=open("customer.txt")
r=f.read()
print(r)
```

```
Enter customer name:"kavit"
Enter customer age:78
['"vishal"', 87]['"kavit"', 78]
```

11 11. Write a python program to create and read the city.txt file in one go and print the contents on the output screen.

[14]:
```
f=open("city.txt","w")
f.write("My city is very clean city.")
f.close()
f=open("city.txt","r")
dt=f.read()
print(dt)
f.close()
```

```
My city is very clean city.
```

12 12. Write a python program that reads a text file and changes the file by capitalizing each character of file.

gg lk mn

lkj

nm

[17]:
```
f=open("11.txt")
data=f.read()
data=data.upper()
f.close()
f=open("11.txt","w")
for i in  data:
    f.write(i)
f.close()
f=open("11.txt")
print(f.read())
```

```
GG LK MN
LKJ
NM
```

13 13.Write a Python program to read a file line by line store it into a variable

[18]:
```
def file_read(fname):
    f=open(fname,"r")
    data=f.readlines()
    print(data)
file_read('friends.txt')
```

```
['Friends are crazy, Friends are naughty !\n', 'Friends  6 are honest, Friends
are  best !\n', 'Friends are like keygen, friends are like license key !\n',
'new We are nothing without friends, Life is not possible without friends !']
```

[19]:
```
file=open('friends.txt','r')
lines=file.readlines()
for index,line in enumerate(lines):
    print("Line {}: {}".format(index,line.strip()))
file.close()
```

```
Line 0: Friends are crazy, Friends are naughty !
Line 1: Friends  6 are honest, Friends are  best !
Line 2: Friends are like keygen, friends are like license key !
Line 3: new We are nothing without friends, Life is not possible without friends
!
```

14 14.Write a Python program to copy the contents of a file to another file.

[20]:
```
file1=open("1.txt")
file2=open("2.txt")
file3=open("3.txt","w")
for lines in file1:
    file3.write(lines)
for lines in file2:
    file3.write(lines)
file1.close()
file2.close()
file3.close()
f=open("3.txt")
print(f.read())
```

```
Friends are crazy, Friends are naughty !
Friends are honest, Friends are best !
Friends are like keygen, friends are like license key !
We are nothing without friends, Life is not possible without friends !Friends
are crazy , Friends are naughty !
Friends are honest , Friends are best !
```

15 15.Write a python program to read line by line from a given files file1 & file2 and write into file3.

File 1

GG JH LK

LKJ

NM

File 2

kk jj mm

nj lk

nj

nh

kk

[3]:
```
file1=open("121.txt")
file2=open("221.txt")
file3=open("321.txt","w")
f1=file1.readlines()
f2=file2.readlines()
lf1=len(f1)
lf2=len(f2)
if lf1>lf2:
    maxi=lf1
else:
    maxi=lf2
for i in range (0,maxi+1):
    if i<lf1:
        file3.write(f1[i])
    if i<lf2:
        file3.write(f2[i])
file1.close()
file2.close()
file3.close()
f=open("321.txt")
print(f.read())
```

```
GG JH LK
kk jj mm
LKJ
nj lk
NMnj
nh
kk
```

16 16.Write a python program to find the longest words in a read file.

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

We are nothing without friends, Life is not possible without friends !

[5]:
```
d={}
f=open("friends.txt")
for lines in f:
    for word in lines.split():
        if word not in d:
            d[word]=len(word)

a=max(d.values())
for i,j in d.items():
    if j==a:
        print(i,j)
```

```
friends, 8
possible 8
```

17 17.Write a python program to search for a string in text files.

[7]:
```
f=open('friends.txt')
lines=f.readlines()
for row in lines:
    word='Lif'
    # print(row.find(word))
    if row.find(word)!=-1:
        print('string exists in file')
        print('line Number:',lines.index(row),row.index(word),row)
f.close()
```

```
string exists in file
line Number: 3 36 new We are nothing without friends, Life is not possible
without friends !
```

18 18.Write a Python program to remove newline characters from a file.

[9]:
```
f=open("221.txt")
a=f.readlines()
b=""
for i in a:
    b+=" "+i.rstrip("\n")
f.close()
print(b)
f=open("221.txt",'w')
f.write(b)
f.close()
```

```
 kk jj mm nj lk nj nh kk
```

19 19.Write a Python program to assess if a file is closed or not.

[15]:
```
f=open('1.txt','r')
fata=f.read()
print(f.closed)
f.close()
print(f.closed)
```

```
False
True
```

[14]:
```
with open("1.txt","r") as f:
    data=f.read()
print(f.closed)
```

```
True
```

20 20.Write a Python program to count the frequency of words in a file

[2]:
```
from collections import Counter
def word_count(fname):
        with open(fname) as f:
                return Counter(f.read().split())

print("Number of words in the file :",word_count("example.txt"))
```

```
Number of words in the file : Counter({'and': 6, 'is': 3, 'Python': 3, 'a': 3,
'programming': 3, 'dynamic': 2, 'code': 2, 'to': 2, 'in': 2, 'or': 2, 'what': 1,
'language?': 1, 'widely': 1, 'used': 1, 'high-level,': 1, 'general-purpose,': 1,
'interpreted,': 1, 'language.': 1, 'Its': 1, 'design': 1, 'philosophy': 1,
```

```
'emphasizes': 1, 'readability,': 1, 'its': 1, 'syntax': 1, 'allows': 1,
'programmers': 1, 'express': 1, 'concepts': 1, 'fewer': 1, 'lines': 1, 'of': 1,
'than': 1, 'possible': 1, 'languages': 1, 'such': 1, 'as': 1, 'C++': 1, 'Java.':
1, 'supports': 1, 'multiple': 1, 'paradigms,': 1, 'including': 1, 'object-
oriented,': 1, 'imperative': 1, 'functional': 1, 'procedural': 1, 'styles.It':
1, 'features': 1, 'type': 1, 'system': 1, 'automatic': 1, 'memory': 1,
'management': 1, 'has': 1, 'large': 1, 'comprehensive': 1, 'standard': 1,
'library.': 1, 'The': 1, 'best': 1, 'way': 1, 'we': 1, 'learn': 1, 'anything':
1, 'by': 1, 'practice': 1, 'exercise': 1, 'questions.': 1, 'We': 1, 'have': 1,
'started': 1, 'this': 1, 'section': 1, 'for': 1, 'those': 1, '(beginner': 1,
'intermediate)': 1, 'who': 1, 'are': 1, 'familiar': 1, 'with': 1, 'Python.': 1})
```

[5]:
```
f=open("example.txt")
d={}
for line in f:
    for words in line.split():
        d[words]=d.get(words,0)+1
print(d)
```

```
{'what': 1, 'is': 3, 'Python': 3, 'language?': 1, 'a': 3, 'widely': 1, 'used':
1, 'high-level,': 1, 'general-purpose,': 1, 'interpreted,': 1, 'dynamic': 2,
'programming': 3, 'language.': 1, 'Its': 1, 'design': 1, 'philosophy': 1,
'emphasizes': 1, 'code': 2, 'readability,': 1, 'and': 6, 'its': 1, 'syntax': 1,
'allows': 1, 'programmers': 1, 'to': 2, 'express': 1, 'concepts': 1, 'in': 2,
'fewer': 1, 'lines': 1, 'of': 1, 'than': 1, 'possible': 1, 'languages': 1,
'such': 1, 'as': 1, 'C++': 1, 'or': 2, 'Java.': 1, 'supports': 1, 'multiple': 1,
'paradigms,': 1, 'including': 1, 'object-oriented,': 1, 'imperative': 1,
'functional': 1, 'procedural': 1, 'styles.It': 1, 'features': 1, 'type': 1,
'system': 1, 'automatic': 1, 'memory': 1, 'management': 1, 'has': 1, 'large': 1,
'comprehensive': 1, 'standard': 1, 'library.': 1, 'The': 1, 'best': 1, 'way': 1,
'we': 1, 'learn': 1, 'anything': 1, 'by': 1, 'practice': 1, 'exercise': 1,
'questions.': 1, 'We': 1, 'have': 1, 'started': 1, 'this': 1, 'section': 1,
'for': 1, 'those': 1, '(beginner': 1, 'intermediate)': 1, 'who': 1, 'are': 1,
'familiar': 1, 'with': 1, 'Python.': 1}
```

21 21.write file from list

[6]:
```
color = ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']
with open('abc.txt', "w") as myfile:
        for c in color:
                myfile.write(c+"\n")

content = open('abc.txt')
print(content.read())
```

```
Red
Green
White
```

```
Black
Pink
Yellow
```

22 22.Write a Python program to combine each line from first file with the corresponding line in second file

[7]:
```
with open('example.txt') as fh1, open('friends.txt') as fh2:
    for line1, line2 in zip(fh1, fh2):
        print(line1+line2)
```

```
what is Python language?
Friends are crazy, Friends are naughty !

Python is a widely used high-level, general-purpose, interpreted, dynamic
programming language. Its design philosophy emphasizes code readability, and its
syntax allows programmers to express concepts in fewer lines of code than
possible in
Friends  6 are honest, Friends are  best !

languages such as C++ or Java.
Friends are like keygen, friends are like license key !

Python supports multiple programming paradigms, including object-oriented,
imperative and functional programming or procedural styles.It features a dynamic
type system and automatic memory management and has a large and comprehensive
standard library. The best way we learn anything is by practice and exercise
questions. We  have started this section for those (beginner to intermediate)
who are familiar with Python.new We are nothing without friends, Life is not
possible without friends !
```

23 23.write a Python program that takes a text file as input and returns the number of words of a given text file.

Note: Some words can be separated by a comma with no space.

[9]:
```
def count_words(filepath):
   with open(filepath) as f:
       data = f.read()
       data.replace(",", " ")
       return len(data.split(" "))
print(count_words("example.txt"))
```

```
189
```

24 24.Write a function in python to count the number of lines from a text file "example2.txt" which is not starting with an alphabet "T".

A boy is playing there.

There is a playground.

An aeroplane is in the sky.

The sky is pink.

Alphabets and numbers are allowed in the password.

[12]:
```
def line_count():
    file = open("example2.txt","r")
    count=0
    for line in file:
        if line[0] not in 'T':
            count+= 1
    file.close()
    print("No of lines not starting with 'T'=",count)

line_count()
```

```
No of lines not starting with 'T'= 3
```

25 25. Write a function in Python to read lines from a text file "example3.txt". Your function should find and display the occurrence of the word "the".

India is the fastest-growing economy. India is looking for more investments around the globe. The whole world is looking at India as a great market. Most of the Indians can foresee the heights that India is capable of reaching.

[13]:
```
def count_words():
    file = open("example3.txt","r")
    count = 0
    data = file.read()
    words = data.split()
    for word in words:
        if word =="the" or word =="The":
            count += 1
    print(count)
    file.close()

count_words()
```

```
5
```

26 26.Write a function display_words() in python to read lines from a text file “example2.txt”, and display those words, which are less than 4 characters.

[15]:
```
def display_words():
    file = open("example2.txt","r")
    data = file.read()
    words = data.split()
    for word in words:
        if len(word) < 4:
            print(word, end=" ")
    file.close()

display_words()
```

```
A boy is is a An is in the The sky is and are in the
```

27 27.Write a function in Python to count words in a text file those are ending with alphabet “e”.

[18]:
```
def count_words():
    file = open("example2.txt","r")
    count = 0
    data = file.read()
    words = data.split()
    for word in words:
        if word[-1] == 'e':
            print(word)
            count+=1
    print(count)
    file.close()

count_words()
```

```
There
aeroplane
the
The
are
the
6
```

28 28.student has used a text editing software to type some text. After saving the article as example4.TXT, she realised that she has wrongly typed alphabet J in place of alphabet I everywhere in the article.

Write a function definition for JTOI() in Python that would display the corrected version of entire content of the file WORDS.TXT with all the alphabets "J" to be displayed as an alphabet "I" on screen.

Note: Assuming that example4.TXT does not contain any J alphabet otherwise.

input WELL, THJS JS A WORD BY JTSELF. YOU COULD STRETCH THJS TO BE A SENTENCE

The function JTOI() should display the following content:

WELL, THIS IS A WORD BY ITSELF. YOU COULD STRETCH THIS TO BE A SENTENCE

[20]:
```
def JTOI():
    file = open("example4.txt","r")
    data = file.read()
    for letter in data:
        if letter == 'J':
            print("I",end="")
        else:
            print(letter,end="")

    file.close()

JTOI()
```

```
WELL, THIS IS A WORD BY ITSELF. YOU COULD STRETCH THIS TO BE A SENTENCE
```

29 29.Read text file into a variable and replace all newlines with space

[21]:
```
with open('example2.txt', 'r') as file:
    data = file.read().replace('\n', ' ')
    print(data)
```

```
A boy is playing there. There is a playground. An aeroplane is in the sky. The
sky is pink. Alphabets and numbers are allowed in the password.
```

30 30.Take input from user and store in .txt file in Python

[23]:
```
temp = input("Please enter your information!!   ")
try:
    with open('gfg.txt', 'w') as gfg:
        gfg.write(temp)
except Exception as e:
    print("There is a Problem", str(e))
f=open("gfg.txt")
print(f.read())
```

```
Please enter your information!!   mnjh
mnjh
```

31 31. Change case of all characters in .txt file using Python

[26]:
```
with open('example2.txt', 'r') as data_file:
    print(data_file.read())
    with open('output.txt', 'a') as output_file:
         output_file.write(data_file.read().swapcase())
f=open("output.txt")
print(f.read())
```

```
A boy is playing there.
There is a playground.
An aeroplane is in the sky.
The sky is pink.
Alphabets and numbers are allowed in the password.
a BOY IS PLAYING THERE.
tHERE IS A PLAYGROUND.
aN AEROPLANE IS IN THE SKY.
tHE SKY IS PINK.
aLPHABETS AND NUMBERS ARE ALLOWED IN THE PASSWORD.
```

32 32.Python Program to Replace Text in a File

Replacing Text could be either erasing the entire content of the file and replacing it with new text or it could mean modifying only specific words or sentences within the existing text.

[3]:
```
s = input("Enter text to replace the existing contents:")
f = open("example2.txt", "r+")
f.truncate(0)
f.write(s)
f.close()
print("Text successfully replaced")
f=open("example2.txt")
```

```
print(f.read())
```

```
Enter text to replace the existing contents:a BOY IS PLAYING THERE. tHERE IS A
PLAYGROUND. aN AEROPLANE IS IN THE SKY. tHE SKY IS PINK. aLPHABETS AND NUMBERS
ARE ALLOWED IN THE PASSWORD
Text successfully replaced
a BOY IS PLAYING THERE. tHERE IS A PLAYGROUND. aN AEROPLANE IS IN THE SKY. tHE
SKY IS PINK. aLPHABETS AND NUMBERS ARE ALLOWED IN THE PASSWORD
```

[5]:
```
# Python program to replace text in a file
x = input("enter text to be replaced:")
y = input("enter text that will replace:")
f = open("example2.txt", "r")
l = f.readlines()
k=[]
c = 0
for i in l:
    if x in i:
        Replacement = i.replace(x, y)
        k.append(Replacement)
    else:
        k.append(i)
    c += 1
f.close()
f=open("example2.txt","w")
f.writelines(k)
f.close()
print("Text successfully replaced")
f=open("example2.txt","r")
print(f.read())
```

```
enter text to be replaced:is
enter text that will replace:i#s
Text successfully replaced
A boy i#s playing there.
There i#s a playground.
An aeroplane i#s in the sky.
The sky i#s pink.
```

33 33.Python Program to Delete Specific Line from File

1

2

3

4

5

6

7

8

9

10

11

12

[6]:
```
line_delete=int(input("enter line number which you want to delete"))
with open('number.txt', 'r') as fr:

        lines = fr.readlines()
        ptr = 1
with open('number.txt', 'w') as fw:
    for line in lines:

        if ptr != line_delete:
            fw.write(line)
        ptr += 1
print("Deleted")
f=open("number.txt")
print(f.read())
```

```
enter line number which you want to delete5
Deleted
1
2
3
4
6
7
8
9
10
11
12
```

34 34.Python Program to Print Lines Containing Given String in File

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

We are nothing without friends, Life is not possible without friends !

[7]:
```
file_name = input("Enter The File's Name: ")
try:
    file_read = open(file_name, "r")
    text = input("Enter the String: ")
    lines = file_read.readlines()

    new_list = []
    idx = 0
    for line in lines:
        if text in line:
            new_list.insert(idx, line)
            idx += 1

    file_read.close()


    if len(new_list)==0:
        print("\n\"" +text+ "\" is not found in \"" +file_name+ "\"!")
    else:

        lineLen = len(new_list)
        print("\n**** Lines containing \"" +text+ "\" ****\n")
        for i in range(lineLen):
            print(end=new_list[i])
        print()
except :
  print("\nThe file doesn't exist!")
```

```
Enter The File's Name: dost.txt
Enter the String: friends

**** Lines containing "friends" ****

Friends are like keygen, friends are like license key !
We are nothing without friends, Life is not possible without friends !
```

35 35.How to remove lines starting with any prefix using Python?

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

We are nothing without friends, Life is not possible without friends !

[8]:
```
file1 = open('dost.txt','r')
file2 = open('dost1.txt','w')
for line in file1.readlines():
    if not (line.startswith('We')):
        print(line)
        file2.write(line)
file2.close()
file1.close()
print("*"*50)
file=open("dost1.txt")
print(file.read())
```

```
Friends are crazy, Friends are naughty !

Friends are honest, Friends are  best !

Friends are like keygen, friends are like license key !

**************************************************
Friends are crazy, Friends are naughty !
Friends are honest, Friends are  best !
Friends are like keygen, friends are like license key !
```


36 36.Eliminating repeated lines from a file using Python

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

Friends are honest, Friends are best !

We are nothing without friends, Life is not possible without friends !

[9]:
```
outputFile = open('pre.txt', "w")
inputFile = open('dost.txt', "r")
lines_seen_so_far = set()
for line in inputFile:
    if line not in lines_seen_so_far:
        outputFile.write(line)
        lines_seen_so_far.add(line)
inputFile.close()
outputFile.close()
f=open("pre.txt")
print(f.read())
```

```
Friends are crazy, Friends are naughty !
Friends are honest, Friends are  best !
```

```
Friends are like keygen, friends are like license key !
We are nothing without friends, Life is not possible without friends !
```

[10]:
```
file = open('dost.txt')
read = file.read()
file.seek(0)
line = 1
for word in read:
    if word == '\n':
        line += 1
print("Number of lines in file is: ", line)
array = []
for i in range(line):
    array.append(file.readline())
print(array)
```

```
Number of lines in file is:  5
['Friends are crazy, Friends are naughty !\n', 'Friends are honest, Friends are
best !\n', 'Friends are like keygen, friends are like license key !\n', 'Friends
are honest, Friends are  best !\n', 'We are nothing without friends, Life is not
possible without friends !']
```


37 37.Python – Append content of one text file to another

Friends are crazy, Friends are naughty !

Friends are honest, Friends are best !

Friends are like keygen, friends are like license key !

Friends are honest, Friends are best !

We are nothing without friends, Life is not possible without friends !

[11]:
```
firstfile = input("Enter the name of first file ")
secondfile = input("Enter the name of second file ")
f1 = open(firstfile, 'r')
f2 = open(secondfile, 'r')
print('content of first file before appending -', f1.read())
print('content of second file before appending -', f2.read())
f2.close()
f1 = open(firstfile, 'a+')
f2 = open(secondfile, 'r')
f1.write(f2.read())
f1.seek(0)
f2.seek(0)
print('content of first file after appending -', f1.read())
print('content of second file after appending -', f2.read())
f1.close()
f2.close()
```

```
Enter the name of first file dost.txt
Enter the name of second file number.txt
content of first file before appending - Friends are crazy, Friends are naughty
!
Friends are honest, Friends are  best !
Friends are like keygen, friends are like license key !
Friends are honest, Friends are  best !
We are nothing without friends, Life is not possible without friends !
content of second file before appending - 1
2
3
4
5
6
7
8
9
10
11
12
content of first file after appending - Friends are crazy, Friends are naughty !
Friends are honest, Friends are  best !
Friends are like keygen, friends are like license key !
Friends are honest, Friends are  best !
We are nothing without friends, Life is not possible without friends !1
2
3
4
5
6
7
8
9
10
11
12
content of second file after appending - 1
2
3
4
5
6
7
8
9
10
11
12
```

38 38.Python Program to Reverse the Content of a File using Stack

[14]:
```
class Stack:

    def __init__(self):
        self._arr = []
    def push(self, val):
        self._arr.append(val)

    def is_empty(self):
        return len(self._arr) == 0
    def pop(self):

        if self.is_empty():
            print("Stack is empty")
            return

        return self._arr.pop()
def reverse_file(filename):

    S = Stack()
    original = open(filename)

    for line in original:
        S.push(line.rstrip("\n"))

    original.close()


    output = open(filename, 'w')

    while not S.is_empty():
        output.write(S.pop()+"\n")

    output.close()
filename = "number.txt"

# Calling the reverse_file function
reverse_file(filename)

# Now reading the content of the file
with open(filename) as file:
        for f in file.readlines():
            print(f, end ="")
```

```
12
11
```

```
10
9
8
7
6
5
4
3
2
1
```

39 39.Python program to Reverse a single line of a text file.line number given by user

[30]:
```
f = open('dost.txt', 'r')
lines = f.readlines()
f.close()
choice = int(input("enter choice: "))
line = lines[choice].split()
Reversed = " ".join(line[::-1])
lines.pop(choice)
Reversed=Reversed+"\n"
lines.insert(choice, Reversed)
u = open('dost.txt', 'w')
u.writelines(lines)
u.close()
k=open("dost.txt","r")
print(k.read())
```

```
enter choice: 0
! naughty are Friends crazy, are Friends
Friends are honest, Friends are  best !
Friends are like keygen, friends are like license key?
We are nothing without friends, Life is not possible without friends !
```

40 40Python program to reverse the content of a file and store it in another file

41 full reverse

[31]:
```
f1 = open("output1.txt", "w")
with open("dost.txt", "r") as myfile:
    data = myfile.read()
data_1 = data[::-1]
f1.write(data_1)
f1.close()
```

```
f= open("output1.txt", "r")
print(f.read())
```

```
! sdneirf tuohtiw elbissop ton si efiL ,sdneirf tuohtiw gnihton era eW
?yek esnecil ekil era sdneirf ,negyek ekil era sdneirF
! tseb  era sdneirF ,tsenoh era sdneirF
! ythguan era sdneirF ,yzarc era sdneirF
```

[32]:
```
# Open the file in write mode
f2 = open("output2.txt", "w")
with open("dost.txt", "r") as myfile:
    data = myfile.readlines()
data_2 = data[::-1]
f2.writelines(data_2)
f2.close()
f= open("output2.txt", "r")
print(f.read())
```

```
We are nothing without friends, Life is not possible without friends !Friends
are like keygen, friends are like license key?
Friends are honest, Friends are  best !
Friends are crazy, Friends are naughty !
```

42 41.Python program to Count the Number of occurrences of a key-value pair in a text file

[33]:
```
f = open("dost.txt", "r")
d = dict()

for res in f:

    res = res.strip()
    res = res.lower()
    lines = res.split()

    for line in lines:

        if line in d:
            d[line] = d[line]+1
        else:
            d[line] = 1

f.close()
for key in list(d.keys()):
    print("The count of {} is {}".format(key,d[key]))
```

```
The count of friends is 7
```

```
The count of are is 7
The count of crazy, is 1
The count of naughty is 1
The count of ! is 3
The count of honest, is 1
The count of best is 1
The count of like is 2
The count of keygen, is 1
The count of license is 1
The count of key? is 1
The count of we is 1
The count of nothing is 1
The count of without is 2
The count of friends, is 1
The count of life is 1
The count of is is 1
The count of not is 1
The count of possible is 1
```

43 42.Python Program to merge two files into a third file

[24]:
```
data = data2 = ""
with open('dost.txt') as fp:
    data = fp.read()
with open('number.txt') as fp:
    data2 = fp.read()
data += "\n"
data += data2

with open ('file3.txt', 'w') as fp:
    fp.write(data)
f= open ('file3.txt', 'r')
print(f.read())
```

```
Friends are crazy, Friends are naughty !
Friends are honest, Friends are  best !
Friends are like keygen, friends are like license key?
We are nothing without friends, Life is not possible without friends !
12
11
10
9
8
7
6
5
4
```

```
3
2
1
```

[36]:
```
class String:
    def __init__(self):
        self.vowels=0
        self.spaces=0
        self.consonants=0
        self.uppercase=0
        self.lowercase=0
        self.string=str(input("Enter string: "))
    def count_uppercase(self):
        for letter in self.string:
            if letter.isupper():
                self.uppercase+=1
    def count_lowercase(self):
        for letter in self.string:
            if letter.islower():
                self.lowercase+=1
    def count_vowels(self):
        for letter in self.string:
            if (letter in ("a","e","i","o","u","A","E","I","O","U")):
                self.vowels+=1
    def count_spaces(self):
        for letter in self.string:
            if letter==" ":
                self.spaces+=1
    def count_consonants(self):
        for letter in self.string:
            if (letter not in ("a","e","i","o","u","A","E","I","O","U") and␣
 ↪letter!=" "):
                self.consonants+=1
    def compute_stat(self):
        self.count_uppercase()
        self.count_lowercase()
        self.count_vowels()
        self.count_spaces()
        self.count_consonants()
    def display(self):
        print("vowels:",self.vowels)
        print("consonants:",self.consonants)
        print("spaces:",self.spaces)
        print("uppercase:",self.uppercase)
        print("lowercase:",self.lowercase)
s=String()
```

```
s.compute_stat()
s.display()
```

```
Enter string: lkj bgh  vgnb#%$%  bgSEDSF cc
vowels: 1
consonants: 22
spaces: 6
uppercase: 5
lowercase: 14
```

44 43.Write a program that prompts for a file name, then opens that file and reads through the file, looking for lines of the form:

X-DSPAM-Confidence: 0.8475

Count these lines and extract the floating point values from each of the lines and compute the average of those values and produce an output as shown below. Do not use the sum() function or a variable named sum in your solution. You can download the sample data at http://www.py4e.com/code3/mbox-short.txt when you are testing below enter mbox-short.txt as the file name.

[37]:
```
# Use the file name mbox-short.txt as the file name
fname = input("Enter file name: ")
fh = open(fname)
count=0
total=0
for line in fh:
    if not line.startswith("X-DSPAM-Confidence:"):
        continue
    t=line.find("0")
    number=float(line[t: ])
    total=total+number
    count=count+1
Average=total/count
print("Average spam confidence:",Average)
```

```
Enter file name: mbox-short.txt
Average spam confidence: 0.7507185185185187
```

45 44.Open the file romeo.txt and read it line by line. For each line, split the line into a list of words using the split() method. The program should build a list of words. For each word on each line check to see if the word is already in the list and if not append it to the list. When the program completes, sort and print the resulting words in python sort() order as shown in the desired output.

[38]:
```
fname = input("Enter file name: ")
fh = open(fname)
lst = list()
for line in fh:
    words=line.split()
    for i in words:
        if i in lst:
            continue
        lst.append(i)
ist=sorted(lst)
print(ist)
```

```
Enter file name: romeo.txt
['Arise', 'But', 'It', 'Juliet', 'Who', 'already', 'and', 'breaks', 'east',
'envious', 'fair', 'grief', 'is', 'kill', 'light', 'moon', 'pale', 'sick',
'soft', 'sun', 'the', 'through', 'what', 'window', 'with', 'yonder']
```

46 45.Open the file mbox-short.txt and read it line by line. When you find a line that starts with 'From' like the following line:

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008 You will parse the From line using split() and print out the second word in the line (i.e. the entire address of the person who sent the message). Then print out a count at the end. Hint: make sure not to include the lines that start with 'From:'. Also look at the last line of the sample output to see how to print the count.

[39]:
```
fname = "mbox-short.txt"
count=0
fh = open(fname)
for line in fh :
    line = line.rstrip()
    if not line.startswith('From '): continue
    count+=1
    words = line.split()
    print(words[1])


print("There were", count, "lines in the file with From as the first word")
```

```
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
zqian@umich.edu
rjlowe@iupui.edu
zqian@umich.edu
rjlowe@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
gsilver@umich.edu
gsilver@umich.edu
zqian@umich.edu
gsilver@umich.edu
wagnermr@iupui.edu
zqian@umich.edu
antranig@caret.cam.ac.uk
gopal.ramasammycook@gmail.com
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
louis@media.berkeley.edu
ray@media.berkeley.edu
cwen@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
There were 27 lines in the file with From as the first word
```


47 46. Write a program to read through the mbox-short.txt and figure out who has sent the greatest number of mail messages. The program looks for 'From' lines and takes the second word of those lines as the person who sent the mail. The program creates a Python dictionary that maps the sender's mail address to a count of the number of times they appear in the file. After the dictionary is produced, the program reads through the dictionary using a maximum loop to find the most prolific committer.

[42]:
```
lst=list()
d=dict()
name = input("Enter file:")
if len(name) < 1:
    name = "mbox-short.txt"
```

```
handle = open(name)
for line in handle:
    line = line.rstrip()
    if not line.startswith('From '): continue
    words = line.split()
    lst.append(words[1])
for i in lst:
    d[i]=d.get(i,0)+1

b=max(d.values())
for k,v in d.items():
    if b==v:
        print(k,v)
```

```
Enter file:mbox-short.txt
cwen@iupui.edu 5
```

48 47.Write a program to read through the mbox-short.txt and figure out the distribution by hour of the day for each of the messages. You can pull the hour out from the 'From' line by finding the time and then splitting the string a second time using a colon.From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008Once you have accumulated the counts for each hour, print out the counts, sorted by hour as shown below.

[41]:
```
lst=list()
gst=list()
d=dict()
name = input("Enter file:")
if len(name) < 1:
    name = "mbox-short.txt"
handle = open(name)
for line in handle:
    line = line.rstrip()
    if not line.startswith('From '): continue
    words = line.split()
    lst.append(words[5])

for i in lst:
    g=i.split(":")
    gst.append(g[0])
gst=sorted(gst)
for j in gst:
    d[j]=d.get(j,0)+1
for k,v in d.items():
```

```
    print(k,v)
```

```
Enter file:mbox-short.txt
04 3
06 1
07 1
09 2
10 3
11 6
14 1
15 2
16 4
17 2
18 1
19 1
```

[9]:
```
f = open("last.txt", "r")
d = dict()
d={}
for s in f:
    for i in s.split():
        if i[0] not in d.keys():
            d[i[0]]=[]
            d[i[0]].append(i)
        else:
            d[i[0]].append(i)
f.close()
print(d)
```

```
{'a': ['are', 'aeroplanr', 'asd'], 'b': ['boy', 'banana', 'bghy', 'bghy'], 'v':
['vfg', 'vfg'], 'c': ['car', 'cat', 'cdf', 'cdf', 'caw'], 'r': ['rty', 'ret'],
'x': ['xsd'], 'z': ['zsd', 'zsx'], 'f': ['fgv'], 'd': ['dcf']}
{'a': ['are', 'aeroplanr', 'asd'], 'b': ['boy', 'banana', 'bghy', 'bghy'], 'v':
['vfg', 'vfg'], 'c': ['car', 'cat', 'cdf', 'cdf', 'caw'], 'r': ['rty', 'ret'],
'x': ['xsd'], 'z': ['zsd', 'zsx'], 'f': ['fgv'], 'd': ['dcf']}
```

[1]:
```
%%writefile word.txt
Cse student
I am student
We are engineers
Apple
1234apple
Juikmnhh
Vishal Acharya
```

```
Writing word.txt
```

49 48 print all three letter word

[2]:
```
wordlist = [line.strip() for line in open('word.txt')]
for word in wordlist:
    #print(word)
    for i in word.split():
        if len(i)==3:
            print(i)
```

```
Cse
are
```

50 49 print word with start st or Ju

[3]:
```
wordlist = [line.strip() for line in open('word.txt')]
for word in wordlist:
    #print(word)
    for i in word.split():
        if i[0:2]=="st" or i[0:2]=="Ju":
            print(i)
```

```
student
student
Juikmnhh
```

51 50 Determine what percentage of words start with a vowel.

[5]:
```
f=open("word.txt")
word_count=0
count=0
for line in f:
    line=line.strip()
    for word in line.split():
        word_count=word_count+1
        if ((word[0] in "aeiou") or (word[0] in "AEIOU")):
            count=count+1
print(count/word_count,"%","start with a vowel")
print(word_count,count)
f.close()
```

```
0.46153846153846156 % start with a vowel
13 6
```

52 51 Find the longest word that can be made using the letters “s” and”t” in word

[6]:
```
f=open("word.txt")
largest = 0
largest_word=""
for line in f:
    line=line.strip()
    for i in line.split():
        if i.find("s")==-1 and i.find("t")==-1:
            break
        else:
            if len(i)>largest:
                largest=len(i)
                largest_word=i
print(largest_word)
f.close()
```

```
student
```

[7]:
```
%%writefile words.txt
Cse staudebntc
I am student
We are engineers
Apple
1234apple
Juikmnhh
Vishal Acharya
```

```
Writing words.txt
```

53 52 Use word.txt to find all the words that can be made from a user’s string

[10]:
```
user=input("enter string: ")
d={}
for i in user:
    d[i]=d.get(i,0)+1
f=open("words.txt")
for line in f:
    line=line.strip()
    for i in line.split():
        e={}
        for x in i:
            e[x]=e.get(x,0)+1
            flag=True
        for j in d:
```

```
            if j not in e or d[j]>e[j]:
                flag=False
        if flag:
                print(i)
f.close()
```

```
enter string: abc
staudebntc
```

54 53 You are given a file called class_scores.txt, where each line of thefile contains a oneword

username and a test score separated by spaces, like below:.

GWashington 83

JAdams 86

- Write code that scans through the file, adds 5 points to each test score, and outputs the usernames and new test scores to a new file,scores2.txt.

[16]:
```
lines = [line.strip() for line in open("class_scores.txt")]
inside_lines = [line.split(' ') for line in lines]
#print("inside lines",inside_lines)
for num in inside_lines:
    num[1]=str(int(num[1])+5)

lines=[" ".join(line) for line in inside_lines]

f=open('scores2.txt','w')
for line in lines:
    f.write(line+"\n")

f.close()
f=open('scores2.txt')
print(f.read())
f.close()
```

```
GWashington 88
JAdams 91
TJefferson 84
JMadison 102
JMonroe 72
JQAdams 97
AJackson 82
MVanBuren 66
JKPolk 96
```

55 54 You are given a file called grades.txt, where each line of the filecontains a one-word student

username and three test scores separated by spaces, like below:.

GWashington 83 77 54 JAdams 86 69 90

Write code that scans through the file and determines how many students passed all three tests

[17]:
```
passed_all3 =[]
failed_atleast1 =[]
with open("grades.txt") as f:
    s = f.readlines()
lines = [line.strip() for line in s]
structure_content = [line.split(" ") for line in lines]
for score in structure_content:
    c = 0
    for i in range(1, len(score)):
        if int(score[i]) > 50 :
            c +=1
        if c == 3:
            passed_all3.append(score[0])
    if c !=3:
        failed_atleast1.append(score[0])
print("Student who passed all three Subjects:",passed_all3,"Their number␣
 ↪is",len(passed_all3))
print("Student who Failed Atleast 1 Subject:",failed_atleast1,"Their number␣
 ↪is",len(failed_atleast1))
```

```
Student who passed all three Subjects: ['GWashington', 'JAdams', 'TJefferson',
'JMadison', 'AJackson', 'MVanBuren'] Their number is 6
Student who Failed Atleast 1 Subject: ['JMonroe', 'JQAdams', 'JKPolk'] Their
number is 3
```

56 55 You are given a file called logfile.txt that lists log-on and log-offtimes for users of a system.

A typical line of the file looks like this:

Van Rossum, 14:22, 14:37

Each line has three entries separated by commas: a username, a log-on time, and a log-off time. Times are given in 24-hour format. You mayassume that all log-ons and log-offs occur within a single workday.

Write a program that scans through the file and prints out all users who were online for at least an hour

[19]:
```
file = open("logfile.txt")
lines = [line.strip() for line in file ]
structure_data = [line.split(" ") for line in lines]
for line in structure_data:
    for time in range(1,len(line)):
        log_of = ((int(line[2][:2])) + (int(line[2][-2:])) / 60)
        log_in = ((int(line[1][:2])) + (int(line[1][3:-1])) / 60)
        time_onine = log_of - log_in
        #print(log_of)
        #print("log in",log_in)
        if time_onine >= 1:
            print(line[0]," Have Been Active For Over An Hour.",time_onine)
```

```
Stroustrup  Have Been Active For Over An Hour. 1.1333333333333329
Stroustrup  Have Been Active For Over An Hour. 1.1333333333333329
Richie  Have Been Active For Over An Hour. 2.0500000000000007
Richie  Have Been Active For Over An Hour. 2.0500000000000007
Armstrong  Have Been Active For Over An Hour. 1.3000000000000007
Armstrong  Have Been Active For Over An Hour. 1.3000000000000007
Wirth  Have Been Active For Over An Hour. 3.4833333333333343
Wirth  Have Been Active For Over An Hour. 3.4833333333333343
Bachus  Have Been Active For Over An Hour. 1.2666666666666675
Bachus  Have Been Active For Over An Hour. 1.2666666666666675
Matz  Have Been Active For Over An Hour. 2.383333333333333
Matz  Have Been Active For Over An Hour. 2.383333333333333
Eich  Have Been Active For Over An Hour. 1.0500000000000007
Eich  Have Been Active For Over An Hour. 1.0500000000000007
Gosling  Have Been Active For Over An Hour. 1.5500000000000007
Gosling  Have Been Active For Over An Hour. 1.5500000000000007
```

57 56 You are given a file called students.txt. A typical line in the filelooks like:

walter melon melon@email.msmary.edu (mailto:melon@email.msmary.edu) 555-3141 There is a name, an email address, and a phone number, each separated by tabs. Write a program that reads through the file line-by-line,and for each line, capitalizes the first letter of the first and last name and adds the area code 301 to the phone number. Your program shouldwrite this to a new file called students2.txt. Here is what the first line of the new

file should look like:

Walter Melon melon@email.msmary.edu (mailto:melon@email.msmary.edu) 301-555-3141

[21]:
```
with open("students.txt") as student:
    s =student.readlines()
lines= [line.strip() for line in s]
structure_content = [line.split("\t") for line in lines]
#print("structured Content",structure_content)
```

```
for data in structure_content:
    data[0] = data[0].title()
    data[1] = data[1].title()
    data[2] = "305-" + data[2]
#print("new list",structure_content)
f=open("student2.txt","w")
for i in structure_content:
        f.writelines(i)
        f.write("\n")
f.close()
f=open("student2.txt","r")
print(f.read())
```

```
Walter MelonMelon@Email.Msmary.Edu305-447-3141
Warren PiecePiece@Email.Msmary.Edu305-447-5926
Winnie BagoBago@Email.Msmary.Edu305-447-5358
Telly VisionVision@Email.Msmary.Edu305-447-9793
Shirley KnotKnot@Email.Msmary.Edu305-447-2384
```


58 57 You are given a file namelist.txt that contains a bunch of names.Some of the names are a first name and a last name separated byspaces, like George Washington, while others have a middle name,like John Quincy Adams. There are no names consisting of just oneword or more than three words. Write a program that asks the user toenter initials, like GW or JQA, and prints all the names that matchthose initials. Note that initials like JA should match both John Adamsand John Quincy Adams.

[22]:
```
with open("namelist.txt") as n:
    s = n.readlines()
lines =[line.strip() for line in s]
str_data = [line.split(" ") for line in lines]

Init =input("Enter User Initial: ")
initial=[]
for i in Init:
    initial.append(i)

for j in str_data:
        if len(initial)==2:
            if len(j)==3:
                if initial[0] == j[0][0] and initial[1]==j[2][0]:
```

```
                    print(" ".join(j))
            else:
                if initial[0] == j[0][0] and initial[1] == j[1][0]:
                    print(" ".join(j))
        elif len(initial)==3:
            if initial[0] == j[0][0] and initial[1] == j[1][0] and initial[2]␣
↪== j[2][0]:
                print(" ".join(j))
        else:
                print("Enter Two Initials Atleast")
                break
```

```
Enter User Initial: MS
Mavis Paola Shindler
Myrtie Seibold
Max T. Swayze
Maryann Schultze
Meg Sherman
```

59 58 You are given a file called baseball.txt. A typical line of the filestarts like below.

Ichiro Suzuki SEA 162 680 74 ...[more stats]

Each entry is separated by a tab, ⌢ The first entry is the player's name and the second is their team. Following that are 16 statistics. Homeruns are the seventh stat and stolen bases are the eleventh. Print out all the players who have at least 20 home runs and at least 20 stolenbases

[23]:
```
file1 = open("baseball.txt","r")
#print(file1.read())
str_data =[lines.split() for lines in file1]
for player in str_data:
    if len(player) > 13:
        if int(player[9]) > 20 and int(player[13] )>= 20:
            print("Player Names: ", player[0:2])
```

```
Player Names:  ['Carlos', 'Gonzalez']
Player Names:  ['Chris', 'Young']
Player Names:  ['Alex', 'Rios']
Player Names:  ['Shin-Soo', 'Choo']
Player Names:  ['Hanley', 'Ramirez']
Player Names:  ['Drew', 'Stubbs']
```

60 59 For this problem, use the file of NCAA basketball scores .

(a) Find the average of the points scored over all the games in the file.

(b) Pick your favorite team and scan through the file to determine how many games they won and how many games they lost.

(c) Find the team(s) that lost by 30 or more points the most times

(d) Find all the teams that averaged at least 70 points a game.

(e) Find all the teams that had winning records but were collectively outscored by their opponents.

A team is collectively outscored by their opponents if the total number of points the team scored over all their games is less than the totalnumber of points their opponents scored in their games against the team

[28]:
```
file1= open("scores.txt","r")
str_data = [line.split() for line in file1]
#print(str_data)
#a)
scorelist=[]
for scores in str_data:
    scorelist.extend([scores[2],scores[4]])
for i in range(0,len(scorelist)): ##convert string to INT
        scorelist[i] = int(scorelist[i])
average = sum(scorelist)/len(scorelist)
print("a")
print("Average",average)
#b)
myteam ="AlcornSt."
team1=[]
team2 =[]
for teams in str_data:
    if teams[1] == myteam:
        team1.append(teams[1:])
    if teams[3] == myteam:
        team2.append(teams[1:])
wins= 0
loss =0
for matches in team1:
    if int(matches[1]) > int(matches[3]):
        wins = wins + 1
        print("wins")
    else:
        loss = loss +1
for matches in team2:
    if int(matches[3]) > int(matches[1]):
        wins = wins + 1
    else:
        loss = loss +1
print("b")
```

```
print(myteam,"Won: ", wins,"Matches and Lost: ",loss,"Matches")
#c
looser_team =[]
for teams in str_data:
    if int(teams[2]) - int(teams[4]) >= 30:
        if teams[3] not in looser_team:
            looser_team.append(teams[3])
    if int(teams[4]) - int(teams[2]) >= 30:
        if teams[1] not in looser_team:
            looser_team.append(teams[1])
print("c")
print("Teams who lost Match with Atleast 30 or more points",looser_team)
#d)
winner_teams=[]
for teams in str_data:
    if int(teams[4]) - int(teams[2]) >= 70:
    #if teams[3] not in looser_team:
        winner_teams.append(teams[3])
print("d")
print(winner_teams)
```

```
a
Average 68.8329283110571
b
AlcornSt. Won:  2 Matches and Lost:  29 Matches
c
Teams who lost Match with Atleast 30 or more points ['AlcornSt.', 'Albany',
'RobertMorris', 'NorthCarolinaCentral', 'Detroit', 'WesternIllinois',
'Dartmouth', 'VirginiaIntermont', 'Presbyterian', 'NCGreensboro', 'LeesMcRae',
'Reinhardt', 'NebraskaKearney', 'Hofstra', 'LoyolaChicago',
'CharlestonSouthern', 'FloridaGulfCoast', 'TennesseeTech', 'SouthwesternAZ',
'WinstonSalemSt.', 'Bryant', 'SacramentoSt.', 'NewMexicoSt.', 'AlabamaA&M',
'ScienceandArtsOK', 'HowardPayne', 'SouthDakota', 'HopeInternational',
'WestVirginiaWesleyan', 'Covenant', 'FloridaA&M', 'NCPembroke',
'SouthernVirginia', 'NorthernMichigan', 'MissouriSt.Louis', 'Suffolk',
'UCDavis', 'Schreiner', 'LeTourneau', 'Piedmont', 'TrumanSt.', 'PittsburgSt.',
'BethuneCookman', 'UCIrvine', 'FloridaInternational', 'Hiram', 'PortlandSt.',
'EastCarolina', 'VirginiaMilitaryInst', 'MountSaintMary', 'Brescia',
'Pennsylvania', 'NovaSoutheastern', 'AlabamaSt.', 'MayvilleSt.',
'StephenF.Austin', 'TexasCollege', 'NWOklahomaSt.', "SaintJoseph'sNY",
'MacMurray', 'Liberty', 'Longwood', 'NCAsheville', 'BowlingGreen', 'Arkansas',
'NichollsSt.', 'IdahoSt.', 'Charlotte', 'Fisk', 'Marist', 'Stetson',
'SIUEdwardsville', 'Southern', 'WesternCarolina', 'PeruSt.', 'St.Gregory',
'Rider', 'ChampionBaptist', 'CentralArkansas', 'Southwest', 'SantaClara',
'SanFrancisco', 'IUPUFortWayne', 'FarmingdaleSt.', 'AtlantaChristian',
'FDUFlorham', 'MDBaltimoreCounty', 'Voorhees', 'Radford', 'LouisianaCollege',
'Mercer', 'RochesterMI', 'SacredHeart', 'Grambling', 'Valparaiso',
'TexasPanAmerican', 'MississippiValleySt.', 'HardinSimmons', 'TrinityFL',
```

```
'Winthrop', 'Vermont', 'Delaware', 'Geneva', 'TennesseeSt.', 'Chattanooga',
'HoustonBaptist', 'Jacksonville', 'IndianaEast', 'FredoniaSt.', 'Oakland',
'CalMaritime', 'JarvisChristian', 'CoppinSt.', 'ArkansasFortSmith',
'CumberlandTN', 'VirginiaWise', 'Allen', 'EasternIllinois', 'Lafayette',
'SoutheastMissouriSt.', 'St.Ambrose', 'Tenn.Wesleyan', 'St.FrancisPA',
'NorthernArizona', 'Dana', 'Millsaps', 'SpringHill', 'SouthernUtah',
'Massachusetts', 'SanDiego', 'Greenville', 'Elon', 'MidlandLthrn', 'Toledo',
'NJInstofTechnology', 'Colgate', 'FreedHardeman', 'MidAmericaChristian',
'ToccoaFalls', 'ChicagoSt.', 'TexasSouthern', 'PaulQuinn', 'Cameron',
'DakotaSt.', 'CalSt.Bakersfield', 'Hartford', 'SUNYCobleskill',
'SalemInternational', 'EasternMichigan', 'Samford', 'St.Bonaventure', 'Brown',
'Oregon', 'Buffalo', 'Maine', 'MoreheadSt.', 'LongBeachSt.', 'MorganSt.',
'PrairieViewA&M', 'AugustanaIL', 'Linfield', 'CulverStockton', 'DePaul',
'MainePresqueIsle', 'Ecclesia', 'KentuckyChristian', 'IndianaSouthBend',
'Wilberforce', 'BridgewaterVA', 'DelawareSt.', 'TexasSt.', 'SavannahSt.',
'Malone', 'CSUMontereyBay', 'SouthCarolinaSt.', 'Marian', 'FairleighDickinson',
'OaklandCity', 'WilliamsBaptist', 'UtahValley', 'NorthernNewMexico',
'Occidental', 'WesternMichigan', 'Roanoke', 'Montreat', 'CarverBible',
'JacksonSt.', 'GardnerWebb', 'NorthernIllinois', 'OralRoberts', 'NorthFlorida',
'CalPoly', 'Berry', 'CollegeofNewJersey', 'Howard', 'Vassar', 'Fordham',
'Centenary', 'Gonzaga', 'MarylandEasternShore', 'Fla.Christian',
'CollegeofCharleston', 'EasternWashington', 'Portland', 'SouthCarolinaUpstate',
'Bryan', 'HighPoint', 'TexasArlington', 'Drexel', 'TennesseeTemple',
'WilliamJessup', 'Bucknell', 'Marshall', 'Navy', 'CalSt.Northridge',
'NorthDakota', 'EastCentral', 'Franklin', 'Idaho', 'SouthDakotaSt.',
'GreatFalls', 'Tulsa', 'IllinoisChicago', 'CornellIA', 'PanhandleSt.',
'SamHoustonSt.', 'PurdueN.Central', 'Alma', 'Arizona', 'NorfolkSt.', 'Belmont',
'WesternOregon', 'CentralBaptist', 'Polytechnic', 'Manhattan',
'MissouriKansasCity', 'Dillard', 'UnionCollege', 'North.NewMexico', 'Whittier',
'PomonaPitzer', 'NCWilmington', 'SouthernMississippi', 'AdamsSt.', 'HolyNames',
'Temple', 'EasternNewMexico', 'TexasA&MCorpusChris', 'KennesawSt.',
'ColumbiaUnion', 'Berea', 'MichiganDearborn', 'Rutgers', 'OregonSt.',
'Oklahoma', 'Clarkson', 'Hawaii', 'CentralConnecticut', 'Stanford',
'YoungstownSt.', 'ColoradoSt.', 'Providence', 'SWAssembliesofGod',
'LoyolaMarymount', 'CentralFlorida', 'Goucher', 'Seattle', 'Towson', 'Duquesne',
'KentuckySt.', 'TennesseeMartin', 'Harvard', 'RioGrande', 'BallSt.',
'LouisianaSt.', 'Wyoming', 'Pepperdine', 'Nebraska', 'AirForce', 'Binghamton',
"SaintJoseph's", 'SouthAlabama', 'Indiana', 'FresnoSt.', 'JacksonvilleSt.',
'TexasChristian', 'NorthCarolina', 'Iowa', 'LouisianaTech', 'WakeForest']
d
['Tennessee', 'Marshall', 'Maine', 'WesternCarolina']
```

61 60 Benford's law states that in real data where the values are spreadacross several orders ofmagnitude, about 30% of the values will startwith the number 1, where as only about 4.6%of the values will startwith the number 9. This is contrary to what we might expect, namelythat values starting with 1 and 9 would be equally likely. Using the fileexpenses.txt which consists of a number of costs from an expenseaccount, determine what percentage start with each of the digits 1through 9. This technique is used by accountants to detect fraud

¶

[33]:
```
file1 = open("expenses.txt","r")
str_data =[line.strip("\n") for line in file1]
total = len(str_data)
percentage_occurence =[]
count = 0
for benford_law in range(1,10):
    for number in range(0,len(str_data)):
        value = str_data[number][:1]
        if int(str_data[number][:1]) == benford_law:
            count = count + 1
    percentage = count/total * 100
    percentage_occurence.append([benford_law,"Occurence Percentage is␣
 ↪",percentage])
    count = 0
#print(percentage_occurence)
for i in percentage_occurence:

    print(i[0],i[1],i[2])
```

```
1 Occurence Percentage is  23.0
2 Occurence Percentage is  15.0
3 Occurence Percentage is  9.0
4 Occurence Percentage is  16.0
5 Occurence Percentage is  15.0
6 Occurence Percentage is  8.0
7 Occurence Percentage is  4.0
8 Occurence Percentage is  5.0
9 Occurence Percentage is  5.0
```

[ ]: