Day Objectives

24th June-2019

In [11]:
```
# Function to find the unique word count and the frequency of the data in a

# This can be do with using the  values and keys (dictionaries) as well as t
def Unique(filePath):
    unique=[]
    freq=[]
    with open(filePath, 'r') as f:
        filedata=f.read().split()
    print(filedata)
    for element in filedata:
        if element not in unique:
            unique.append(element)
    print(unique)
    for i in unique:
        print(i,':',filedata.count(i))

filePath = 'DataFiles/data.txt'
Unique(filePath)
```

```
['Line1', 'Line2', 'line2', 'Anyone', 'who', 'reads', 'Old', 'and', 'Middle',
'English', 'literary', 'texts', 'will', 'be', 'familiar', 'with', 'the', 'mid-b
rown', 'volumes', 'of', 'the', 'EETS,', 'with', 'the', 'symbol', 'of', "Alfre
d's", 'jewel', 'embossed', 'on', 'the', 'front', 'cover.', 'Most', 'of', 'the',
'works', 'attributed', 'to', 'King', 'Alfred', 'or', 'to', 'Aelfric,', 'along',
'with', 'some', 'of', 'those', 'by', 'bishop', 'Wulfstan', 'and', 'much', 'anon
ymous', 'prose', 'and', 'verse', 'from', 'the', 'pre-Conquest', 'period,', 'ar
e', 'to', 'be', 'found', 'within', 'the', "Society's", 'three', 'series;', 'al
l', 'of', 'the', 'surviving', 'medieval', 'drama,', 'most', 'of', 'the', 'Middl
e', 'English', 'romances,', 'much', 'religious', 'and', 'secular', 'prose', 'an
d', 'verse', 'including', 'the', 'English', 'works', 'of', 'John', 'Gower,', 'T
homas', 'Hoccleve', 'and', 'most', 'of', "Caxton's", 'prints', 'all', 'find',
'their', 'place', 'in', 'the', 'publications.', 'Without', 'EETS', 'editions,',
'study', 'of', 'medieval', 'English', 'texts', 'would', 'hardly', 'be', 'possib
le.']
['Line1', 'Line2', 'line2', 'Anyone', 'who', 'reads', 'Old', 'and', 'Middle',
'English', 'literary', 'texts', 'will', 'be', 'familiar', 'with', 'the', 'mid-b
rown', 'volumes', 'of', 'EETS,', 'symbol', "Alfred's", 'jewel', 'embossed', 'o
n', 'front', 'cover.', 'Most', 'works', 'attributed', 'to', 'King', 'Alfred',
'or', 'Aelfric,', 'along', 'some', 'those', 'by', 'bishop', 'Wulfstan', 'much',
'anonymous', 'prose', 'verse', 'from', 'pre-Conquest', 'period,', 'are', 'foun
d', 'within', "Society's", 'three', 'series;', 'all', 'surviving', 'medieval',
'drama,', 'most', 'romances,', 'religious', 'secular', 'including', 'John', 'Go
wer,', 'Thomas', 'Hoccleve', "Caxton's", 'prints', 'find', 'their', 'place', 'i
n', 'publications.', 'Without', 'EETS', 'editions,', 'study', 'would', 'hardl
y', 'possible.']
Line1 : 1
Line2 : 1
line2 : 1
Anyone : 1
who : 1
reads : 1
Old : 1
and : 6
Middle : 2
English : 4
literary : 1
texts : 2
```

```
will : 1
be : 3
familiar : 1
with : 3
the : 11
mid-brown : 1
volumes : 1
of : 9
EETS, : 1
symbol : 1
Alfred's : 1
jewel : 1
embossed : 1
on : 1
front : 1
cover. : 1
Most : 1
works : 2
attributed : 1
to : 3
King : 1
Alfred : 1
or : 1
Aelfric, : 1
along : 1
some : 1
those : 1
by : 1
bishop : 1
Wulfstan : 1
much : 2
anonymous : 1
prose : 2
verse : 2
from : 1
pre-Conquest : 1
period, : 1
are : 1
found : 1
within : 1
Society's : 1
three : 1
series; : 1
all : 2
surviving : 1
medieval : 2
drama, : 1
most : 2
romances, : 1
religious : 1
secular : 1
including : 1
John : 1
Gower, : 1
Thomas : 1
Hoccleve : 1
Caxton's : 1
```

```
prints : 1
find : 1
their : 1
place : 1
in : 1
publications. : 1
Without : 1
EETS : 1
editions, : 1
study : 1
would : 1
hardly : 1
possible. : 1
```

In [24]:
```
def uniqueData(filePath):
    # create an empty unique list
    unique=[]
    # for every element in the main list,
        # Checck if it exists in the unique list
        # If it does not exist,add it to unique list
        # else if it already exists ,move  on to the main list and add it to
    with open (filePath,'r') as f:
        filedata=f.read().split()
        for i in filedata:
            if i not in unique:
                unique.append(i)
    return unique
filePath='DataFiles\data.txt'
uniqueData(filePath)


```

Out[24]:
```
['Line1',
 'Line2',
 'line2',
 'Anyone',
 'who',
 'reads',
 'Old',
 'and',
 'Middle',
 'English',
 'literary',
 'texts',
 'will',
 'be',
 'familiar',
 'with',
 'the',
 'mid-brown',
 'volumes',
 'of',
 'EETS,',
 'symbol',
 "Alfred's",
 'jewel',
 'embossed',
 'on',
 'front',
 'cover.',
 'Most',
 'works',
 'attributed',
 'to',
 'King',
 'Alfred',
 'or',
 'Aelfric,',
 'along',
 'some',
 'those',
```

```
 'by',
 'bishop',
 'Wulfstan',
 'much',
 'anonymous',
 'prose',
 'verse',
 'from',
 'pre-Conquest',
 'period,',
 'are',
 'found',
 'within',
 "Society's",
 'three',
 'series;',
 'all',
 'surviving',
 'medieval',
 'drama,',
 'most',
 'romances,',
 'religious',
 'secular',
 'including',
 'John',
 'Gower,',
 'Thomas',
 'Hoccleve',
 "Caxton's",
 'prints',
 'find',
 'their',
 'place',
 'in',
 'publications.',
 'Without',
 'EETS',
 'editions,',
 'study',
 'would',
 'hardly',
 'possible.']
```

In [7]:
```
# Read a File - File should Exist in some part of the sotorage (Read Mode)

# Write to a File - Existing( append mode)  or New File(Write mode)
def reading(filePath):
    with open(filePath, 'r') as f:
        filedata = f.read()
        lines=filedata.split(',')
    return filedata

filePath = 'DataFiles/data.txt'
print(reading(filePath))
```

```
Line1
Line2
line2
Anyone who reads Old and Middle English literary
texts will be familiar with the mid-brown volumes of the EETS,
with the symbol of Alfred's jewel embossed on the front cover.
Most of the works attributed to King Alfred or to Aelfric, along with some of t
hose by bishop
Wulfstan and much anonymous prose and verse from the pre-Conquest period, are t
o be found within
the Society's three series;
all of the surviving medieval drama, most of the Middle
English romances, much religious and secular prose and verse including the Engl
ish
works of John Gower, Thomas Hoccleve and most of Caxton's prints all find their
place in the publications.
Without EETS editions, study of medieval English texts would hardly be possibl
e.
```

In [25]:
```
import re
def wordcount(filePath):
    pattern='[, \n]'
    #filepath='Data Files\data.txt'
    filedata =reading(filePath)
    count=len(re.split(pattern,filedata))
    return count


def wordsFromFile(filePath):
    pattern='[ ,\n.]'
    filedata=reading(filePath)
    allWordsList=re.split(pattern,filedata)
    return allWordsList
wordsFromFile(filePath)
```

```
"Caxton's",
'prints',
'all',
'find',
'their',
'place',
'in',
'the',
'publications',
'',
'',
'Without',
'EETS',
'editions',
'',
'study',
'of',
'medieval',
'English',
```

In [ ]:
```
1
```

In [26]:
```
def FreqDistribution1(filePath):
    allwords=wordsFromFile(filePath)
    uniquewords=uniqueData(filePath)

    for word in uniquewords:
        count=allwords.count(word)
        print(word,':',count)
    return
FreqDistribution1(filePath)
```

```
Line1 : 1
Line2 : 1
line2 : 1
Anyone : 1
who : 1
reads : 1
Old : 1
and : 6
Middle : 2
English : 4
literary : 1
texts : 2
will : 1
be : 3
familiar : 1
with : 3
the : 11
mid-brown : 1
volumes : 1
of : 9
EETS, : 0
symbol : 1
Alfred's : 1
jewel : 1
embossed : 1
on : 1
front : 1
cover. : 0
Most : 1
works : 2
attributed : 1
to : 3
King : 1
Alfred : 1
or : 1
Aelfric, : 0
along : 1
some : 1
those : 1
by : 1
bishop : 1
Wulfstan : 1
much : 2
anonymous : 1
prose : 2
verse : 2
from : 1
```

```
pre-Conquest : 1
period, : 0
are : 1
found : 1
within : 1
Society's : 1
three : 1
series; : 1
all : 2
surviving : 1
medieval : 2
drama, : 0
most : 2
romances, : 0
religious : 1
secular : 1
including : 1
John : 1
Gower, : 0
Thomas : 1
Hoccleve : 1
Caxton's : 1
prints : 1
find : 1
their : 1
place : 1
in : 1
publications. : 0
Without : 1
EETS : 2
editions, : 0
study : 1
would : 1
hardly : 1
possible. : 0
```

In [43]:
```
def FreDistribution2(filePath):
    allwords=wordsFromFile(filePath)
    wordfrequency={}
    for word in allwords:
        if word not in wordfrequency.keys():
            wordfrequency[word] =1
        else:
            wordfrequency[word]=-1
    return wordfrequency
FreDistribution2(filePath)
```

Out[43]:
```
{'Line1': 1,
 'Line2': 1,
 'line2': 1,
 'Anyone': 1,
 'who': 1,
 'reads': 1,
 'Old': 1,
 'and': -1,
 'Middle': -1,
 'English': -1,
 'literary': 1,
 '': -1,
 'texts': -1,
 'will': 1,
 'be': -1,
 'familiar': 1,
 'with': -1,
 'the': -1,
 'mid-brown': 1,
 'volumes': 1,
 'of': -1,
 'EETS': -1,
 'symbol': 1,
 "Alfred's": 1,
 'jewel': 1,
 'embossed': 1,
 'on': 1,
 'front': 1,
 'cover': 1,
 'Most': 1,
 'works': -1,
 'attributed': 1,
 'to': -1,
 'King': 1,
 'Alfred': 1,
 'or': 1,
 'Aelfric': 1,
 'along': 1,
 'some': 1,
 'those': 1,
 'by': 1,
 'bishop': 1,
 'Wulfstan': 1,
 'much': -1,
 'anonymous': 1,
```

```
'prose': -1,
'verse': -1,
'from': 1,
'pre-Conquest': 1,
'period': 1,
'are': 1,
'found': 1,
'within': 1,
"Society's": 1,
'three': 1,
'series;': 1,
'all': -1,
'surviving': 1,
'medieval': -1,
'drama': 1,
'most': -1,
'romances': 1,
'religious': 1,
'secular': 1,
'including': 1,
'John': 1,
'Gower': 1,
'Thomas': 1,
'Hoccleve': 1,
"Caxton's": 1,
'prints': 1,
'find': 1,
'their': 1,
'place': 1,
'in': 1,
'publications': 1,
'Without': 1,
'editions': 1,
'study': 1,
'would': 1,
'hardly': 1,
'possible': 1}
```

In [ ]:

```
#contacts Application
    # Add,Search,List,Modify Delete Contacts

# Find and Replace Application
    # Count the total number of occurances of a word
    # If word is existing
    # Replace all occurances of word with another word
# Marks Analysis Application
    # Generate marks file-
    # Input: Marks text file - each line contacts marks of students
    # Generates a report with with the following information
        # Class Average
        # % Of Students passed
        # % of Students failed
        # % of students with Distinction
```

In [53]:
```
# Function  to generate marks data for n students

from random import randint


def generateMarks(n, lb, ub):
    with open('Data Files\marks.txt','w') as f:
        for i in range(0, n):
            r=randint(lb, ub)
            f.write(str(r) + '\n')
    return
generateMarks(10, 1, 100)

```

In [27]:
```
def appendDataToFile(filename,filedata):
    with open(filename,'a') as f:
        f.writelines(filedata)
    return
filedata=["Line2,Line3"]
filename='Data Files\data.txt'
appendDataToFile(filename,filedata)

```

In [ ]:
```

```