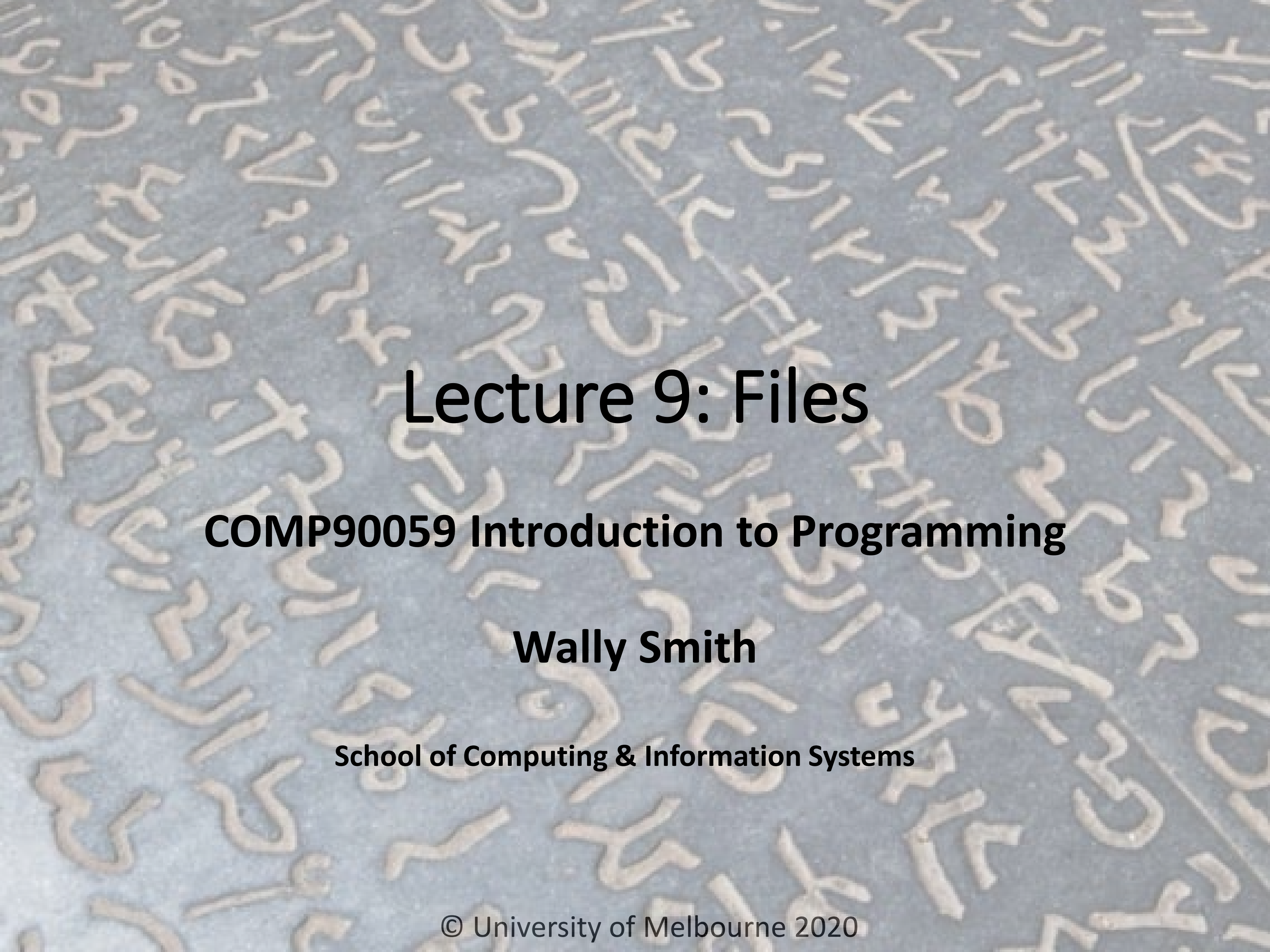


Lecture 9: Files

COMP90059 Introduction to Programming

Wally Smith

School of Computing & Information Systems

ASSESSMENT - REVISED on 9th April 2020

% contributions to overall grade for the subject are shown for each piece of assessment

•**Assignment 1** (10%) is ongoing and due on Friday 24th April.

•**Mid-Semester Test** (0%) will take place on Week 8, at 2 - 3 pm on Monday 4th May. This will be a one-hour test online via Grok in place of the usual Monday lecture. Note that your score on the test will NOT contribute to your overall grade for the subject, but it will provide you with feedback on your learning so far, and it will also give you a good indication of what the final examination will be like.

•**Assignment 2** (20%) is a second set of programming tasks on Grok. It will be released on Wednesday 6th May (during Week 8, after the Mid- Semester Test) , and is due on Friday 29th May (Week 11).

•**Final Examination** (70%) will be held in the examination period after semester (date to be advised). It will be delivered online in a way to be advised. You will carry out the exam during a specified 3 hours.

•**Hurdle requirement: 35/70 or greater** on the Final Examination is needed to pass the subject.

Please rest assured that the Final Examination is not intended to present fiendishly complex problems. Instead, it will present a mixture of questions that are designed to allow you demonstrate the knowledge and skills you have developed through the lectures, tutes and assignments. The nature of the Final Examination will be discussed in lectures in the remainder of semester, and we will look at many examples of the kinds of questions that will be included.

Lecture Overview

Files

We will look at how to read and write data to files. This opens up the true power of the techniques we have learned so far.

Multidimensional Lists

A further technique with lists that let us hold and manipulate larger data sets that are organized into dimensions.

Coding patterns

This is a theme we will have in every lecture from now on. We will identify the recurring patterns in the way we solve coding problems. This will help you to focus on how to approach Assignment 2 and the Final Exam.

Files

Python treats files of data stored on disk as **sequences** of data ... like lists and strings.

When dealing with a file, we need to know what separates the items in the sequence.

A text-data file is likely to be separated by **line-**returns (\n):

```
Anais Nin “risk”\n
\n
And then the day came,\n
when the risk\n
to remain tight\n
in a bud\n
was more painful\n
than the risk\n
it took\n
to blossom. \n
```

A comma-separated values (csv) is another common structure for data as a series of values:

```
name, age, speed\n
'Wilson', 34, 120\n
'Wang', 21, 101\n
'Amarit', 45, 117\n
'Jones', 18, 165\n
'Zhu', 23, 175\n
```

Reading from files

Suppose we want to read from a file called 'poem.txt' ...

```
infile = open('poem.txt' , 'r')

allText = infile.read()
print(allText)

infile.close()
```

creates a file object

applies a method of the file object

Alternative methods for reading in the data from the file:

- read() - reads everything in the file as one long string
- readline() - reads the next single line in the file (up to \n)
- readlines() - reads all of the (remaining) lines in the file

'poem.txt'

```
Anais Nin “risk”\n\nAnd then the day came,\nwhen the risk\nto remain tight\nin a bud\nwas more painful\nthan the risk\nit took\nto blossom. \n
```

```
infile = open('poem.txt' , 'r')

allText = infile.read()
print(allText)

infile.close()
```

```
ANAIS NIN “RISK”
And then the day came,
when the risk
to remain tight
in a bud
was more painful
than the risk
it took
to blossom.
```

```
infile = open('poem.txt' , 'r')

lines = infile.readlines()
for line in lines:
    print(line)

infile.close()
```

```
infile = open('poem.txt' , 'r')

for line in infile:
    print(line)

infile.close()
```

```
ANAIS NIN “RISK”

And then the day came,

when the risk

to remain tight

in a bud

was more painful

than the risk

it took

to blossom.
```

Exercise 1: Write a program to read from a file called 'poem.txt' and print the lines of poem out in reverse order. Assume the file holds the data as lines of the poem, separated by line breaks.

Tips:
* use the readlines() method to read the lines of the poem into a **list**
* use a **for loop** to work through the list of lines in reverse order
* use **range()** to count in steps forwards or backwards:

  range(start position, stop position + 1, step)
* give it negative step like -1, and it will count backwards

```
infile = open('poem.txt' , 'r')

lines = infile.readlines()

for i in range(len(lines),0,-1):
    print(lines[i - 1])

infile.close()
```

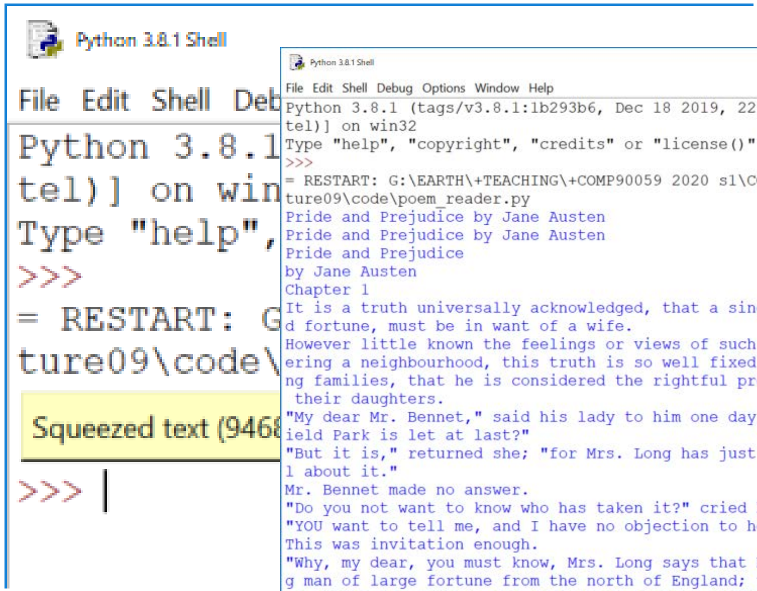
```
to blossom.
it took

than the risk

was more painful

in a bud

to remain tight

when the risk

And then the day came,

ANAIS NIN “RISK”
```

Reading from LARGE files

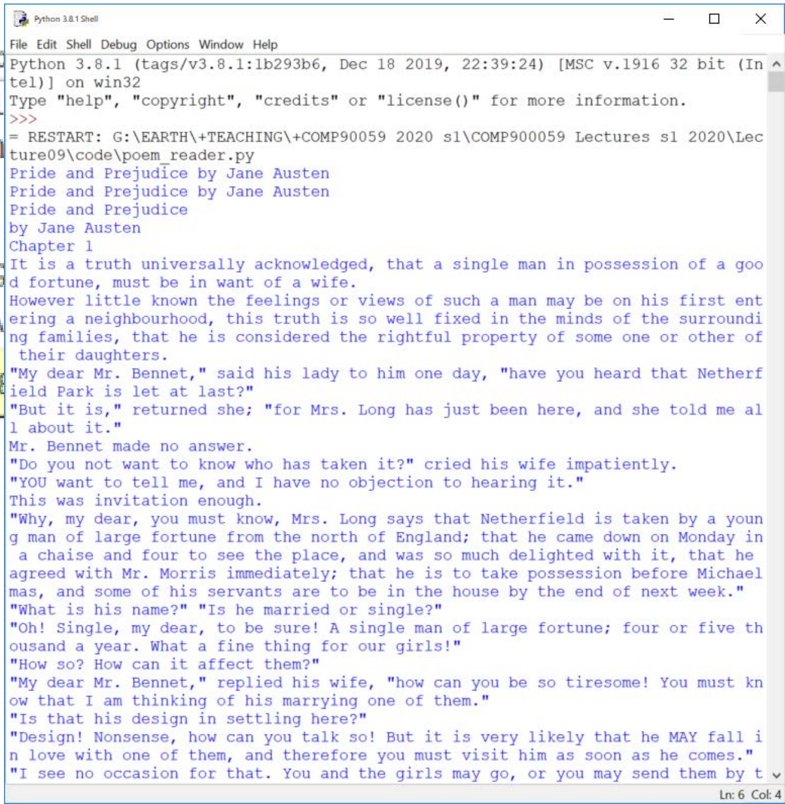
```
infile = open('poem.txt' , 'r')

allText = infile.read()
print(allText)

infile.close()
```

ANAIS NIN “RISK”
And then the day came,
when the risk
to remain tight
in a bud
was more painful
than the risk
it took
to blossom.

```
infile = open('Pride-and-Prejudice.txt' , 'r')

allText = infile.read()
print(allText)

infile.close()
```



```
Python 3.8.1
tel)] on win
Type "help",
>>>
= RESTART: G
ture09\code\
Squeezed text (9468
>>> |
```



```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: G:\EARTH\+TEACHING\+COMP90059 2020 s1\COMP900059 Lectures s1 2020\Lec
ture09\code\poem_reader.py
Pride and Prejudice by Jane Austen
Pride and Prejudice by Jane Austen
Pride and Prejudice
by Jane Austen
Chapter 1
It is a truth universally acknowledged, that a single man in possession of a goo
d fortune, must be in want of a wife.
However little known the feelings or views of such a man may be on his first ent
ering a neighbourhood, this truth is so well fixed in the minds of the surroundi
ng families, that he is considered the rightful property of some one or other of
 their daughters.
"My dear Mr. Bennet," said his lady to him one day, "have you heard that Netherf
ield Park is let at last?"
"But it is," returned she; "for Mrs. Long has just been here, and she told me al
l about it."
Mr. Bennet made no answer.
"Do you not want to know who has taken it?" cried his wife impatiently.
"YOU want to tell me, and I have no objection to hearing it."
This was invitation enough.
"Why, my dear, you must know, Mrs. Long says that Netherfield is taken by a youn
g man of large fortune from the north of England; that he came down on Monday in
 a chaise and four to see the place, and was so much delighted with it, that he
agreed with Mr. Morris immediately; that he is to take possession before Michael
mas, and some of his servants are to be in the house by the end of next week."
"What is his name?" "Is he married or single?"
"Oh! Single, my dear, to be sure! A single man of large fortune; four or five th
ousand a year. What a fine thing for our girls!"
"How so? How can it affect them?"
"My dear Mr. Bennet," replied his wife, "how can you be so tiresome! You must kn
ow that I am thinking of his marrying one of them."
"Is that his design in settling here?"
"Design! Nonsense, how can you talk so! But it is very likely that he MAY fall i
n love with one of them, and therefore you must visit him as soon as he comes."
"I see no occasion for that. You and the girls may go, or you may send them by t
```

Exercise 2: Write a program to read from an eBook (of the users choosing) and print the book out line by line under the user's control. Assume the file holds the data as lines of the poem, separated by line breaks.
Tip: use readline() to read one line, print it out, then let the user decide if they want to continue ... and use a loop to keep going right through the file.

```
#program to read a given eBook(txt) line by line

#get name of eBook from user
eBook = input('Enter name of eBook: ')

#create a file object that reads from eBook
infile = open(eBook, 'r')

#read from the file line by line under users control
keyPress = ' '
while keyPress != 's':
    line = infile.readline()
    print(line)
    keyPress = input()

#close the file object
infile.close()
```

Try this program out for two books on Canvas:

- Pride and Prejudice
- Harry Potter and the Sorcerer's Stone

You will see the lines of text are very spaced out. Why do you think this is? How can you make them less spaced out?

Pattern: Sentinal while loop

Processing large data sets

We can now apply the techniques we have learned so far in this subject to find the answers to interesting questions about large data sets.

For example:

- What are the most frequent words in "Pride and Prejudice", and "Harry Potter and the Sorcerer's Stone"?
- What are the new words created by "Harry Potter and the Sorcerer's Stone" that are not common in the English language?
- What is the average age of all the athletes taking part in the Olympic Games between 1896 - 2016?

Multidimensional lists

We have already talked about how lists can contain a mixture of data types, including lists.

Lists within lists is a very powerful technique for handling large data sets.

```
customers = [
        ['James', 'Johnson', 'jd@gmail.com', '23A Cedar Avenue, VIC 3078', 25],
        ['Xinyu', 'Wang', 'xw@gmail.com', '17B Fountain Court, VIC 3000', 37],
        ['Aesha', 'Acharya', 'aa@gmail.com', '22 Franklin Rd, VIC 3043', 23],
        ['John', 'Dang', 'jd@gmail.com', '102 Princess Lane, VIC 3002', 28]
                    ]
```

```
print(customers[2])      ['Aesha', 'Acharya', 'aa@gmail.com', '22 Franklin Rd, VIC 3043', 23]

print(customer[2][0])     Aesha
```

Exercise 3: Write a program to produce the messages below to customers in the multidimensional list.

```
customers = [
        ['James', 'Johnson', 'jd@gmail.com', '23A Cedar Avenue, VIC 3078', 25],
        ['Xinyu', 'Wang', 'xw@gmail.com', '17B Fountain Court, VIC 3000', 37],
        ['Aesha', 'Acharya', 'aa@gmail.com', '22 Franklin Rd, VIC 3043', 23],
        ['John', 'Dang', 'jd@gmail.com', '102 Princess Lane, VIC 3002', 28]
        ]

for customer in customers:
        print('To:', customer[2])
        print()
        print('Dear', customer[0]+',')
        print('I believe that you are living at', customer[3]+'.')
        print('Please let me know if this is incorrect.')
        print()


for c in range(len(customers)):
        print('To:', customers[c][2])
        print()
        print('Dear', customers[c][0]+',')
        print('I believe that you are living at', customers[c][3]+'.')
        print('Please let me know if this is incorrect.')
        print()
```

```
To: jd@gmail.com

Dear James,
I believe that you are living at 23A Cedar
Avenue, VIC 3078.
Please let me know if this is incorrect.

To: xw@gmail.com

Dear Xinyu,
I believe that you are living at 17B Fountain
Court, VIC 3000.
Please let me know if this is incorrect.

To: aa@gmail.com

Dear Aesha,
I believe that you are living at 22 Franklin
Rd, VIC 3043.
Please let me know if this is incorrect.

To: jd@gmail.com

Dear John,
I believe that you are living at 102 Princess
Lane, VIC 3002.
Please let me know if this is incorrect.
```

- What are the most frequent words in "Harry Potter and the Sorcerer's Stone?"

```
# program to report on top most frequently occurring words in eBook
def main():
    words = read_words('Harry_Potter_and_the_Sorcerer.txt')
    words = clean(words)                    #remove non alpha characters
    uniques = unique(words)                 #get all the unique words of the text
    uniqueCounts = count(uniques, words)    #count how often each unique word occurs
    report(words, uniqueCounts, 20)
```

'the cat the cat the cat the sat'

```
the   4
cat   3
sat   1
```

'The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs. Potter was Mrs. Dursley's sister, but they hadn't met for several years; in fact, Mrs. Dursley pretended she didn't have a sister, because her sister and her good-for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbors would say if the Potters arrived in the street. The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that. ... '

- What are the most common words in Harry Potter and the Sorcerer's Stone?

```
# program to report on top most frequently occuring words in eBook
def main():
    words = read_words('Harry_Potter_and_the_Sorcerer.txt')
    words = clean(words)                        #remove non alpha characters
    uniques = unique(words)                     #get all the unique words of the text
    uniqueCounts = count(uniques, words)  #count how often each unique word occurs
    report(words, uniqueCounts, 20)


def read_words(file):
     inFile = open(file, 'r')
     textOfFile = inFile.read()
     inFile.close()
     return textOfFile.split(' ')
```

Pattern: converting a string into a list of values for processing

```
['Harry', 'Potter', 'and', 'the', "Sorcerer's", 'Stone', '\n\nCHAPTER', 'ONE', '\n\nTHE', 'BOY',
'WHO', 'LIVED', '\n\nMr.', 'and', 'Mrs.', 'Dursley,', 'of', 'number', 'four,', 'Privet', ... ]
```

about 78,000 words

- What are the most common words in Harry Potter and the Sorcerer's Stone?

```
# program to report on top most frequently occuring words in eBook
def main():
    words = read_words('Harry_Potter_and_the_Sorcerer.txt')
    words = clean(words)                    #remove non alpha characters
    uniques = unique(words)                 #get all the unique words of the text
    uniqueCounts = count(uniques, words)    #count how often each unique word occurs
    report(words, uniqueCounts, 20)
```

```
['Harry', 'Potter', 'and', 'the', "Sorcerer's", 'Stone', '\n\nCHAPTER', 'ONE', '\n\nTHE', 'BOY', 'WHO',
'LIVED', '\n\nMr.', 'and', 'Mrs.', 'Dursley,', 'of', 'number', 'four,', 'Privet', ... ]
```

```
def clean(words):
    cleanWords = [ ]
    for w in range(len(words)):
        cleanWord = ''
        for ch in words[w]:
            if ch.isalpha():
                cleanWord += ch
        if cleanWord != '':
            cleanWords.append(cleanWord.lower())
    return cleanWords
```

Pattern: rebuilding a list

Pattern: rebuilding a string

```
['harry', 'potter', 'and', 'the', 'sorcerers', 'stone', 'chapter', 'one', 'the', 'boy', 'who', 'lived', 'mr', 'and',
'mrs', 'dursley', 'of', 'number', 'four', 'privet' ... ]
```

- What are the most common words in Harry Potter and the Sorcerer's Stone?

```
# program to report on top most frequently occuring words in eBook
def main():
    words = read_words('Harry_Potter_and_the_Sorcerer.txt')
    words = clean(words)                        #remove non alpha characters
    uniques = unique(words)                     #get all the unique words of the text
    uniqueCounts = count(uniques, words)        #count how often each unique word occurs
    report(words, uniqueCounts, 20)
```

['harry', 'potter', 'and', 'the', 'sorcerers', 'stone', 'chapter', 'one', 'the', 'boy', 'who', 'lived', 'mr', 'and', 'mrs', 'dursley', 'of', 'number', 'four', 'privet', ...]

```
def unique(words):
    uniques = [ ]
    for word in words:
        if word not in uniques:
            uniques.append(word)
    return uniques
```

Pattern: rebuilding a list

['harry', 'potter', 'and', 'the', 'sorcerers', 'stone', 'chapter', 'one', 'boy', 'who', 'lived', 'mr', 'mrs', 'dursley', 'of', 'number', 'four', 'privet', ...]

- What are the most common words in Harry Potter and the Sorcerer's Stone?

```
# program to report on top most frequently occuring words in eBook
def main():
    words = read_words('Harry_Potter_and_the_Sorcerer.txt')
    words = clean(words)                        #remove non alpha characters
    uniques = unique(words)                     #get all the unique words of the text
    uniqueCounts = count(uniques, words)   #count how often each unique word occurs
    report(words, uniqueCounts, 20)
```

```
['harry', 'potter', 'and', 'the', 'sorcerers', 'stone', 'chapter', 'one', 'boy', 'who', 'lived', 'mr', 'mrs',
'dursley', 'of', 'number', 'four', 'privet', ...]
```

```
def count(uniques, words):
    wordCounts = [ ]
    for w in range(len(uniques)):
        wordCounts.append([words.count(uniques[w]),uniques[w]])
    return wordCounts
```

Pattern: building a list

Method: count()

```
[  [1214, 'harry'], [95, 'potter'], [1919, 'and'], [3628, 'the'], [16, 'sorcerers'], [75, 'stone'], [17, 'chapter'],
[255, 'one'], [83, 'boy'], [160, 'who'], [9, 'lived'], [81, 'mr'], [46, 'mrs'], [54, 'dursley'], [1259, 'of'], [17,
'number'], [32, 'four'], [16, 'privet'],  ...      ]
```

- What are the most common words in Harry Potter and the Sorcerer's Stone?

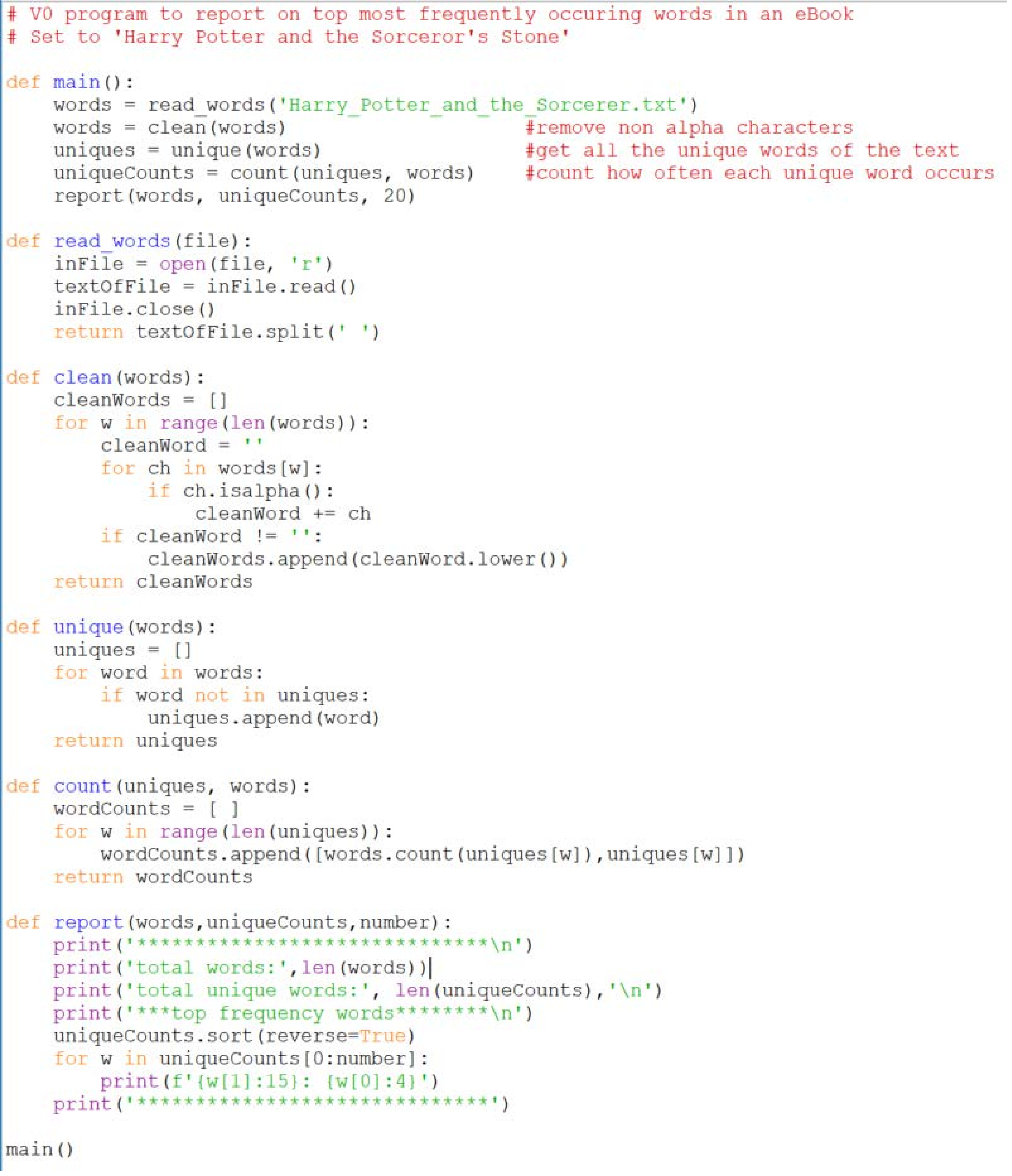
```
# program to report on top most frequently occuring words in eBook
def main():
    words = read_words('Harry_Potter_and_the_Sorcerer.txt')
    words = clean(words)                     #remove non alpha characters
    uniques = unique(words)                  #get all the unique words of the text
    uniqueCounts = count(uniques, words)     #count how often each unique word occurs
    report(words, uniqueCounts, 20)
```

```
[  [1214,  'harry'], [95, 'potter'], [1919, 'and'], [3628, 'the'], [16, 'sorcerers'], [75, 'stone'], [17,
'chapter'], [255, 'one'], [83, 'boy'], [160, 'who'], [9, 'lived'], [81, 'mr'], [46, 'mrs'], [54, 'dursley'], [1259,
'of'], [17, 'number'], [32, 'four'], [16, 'privet'],  ...      ]
```

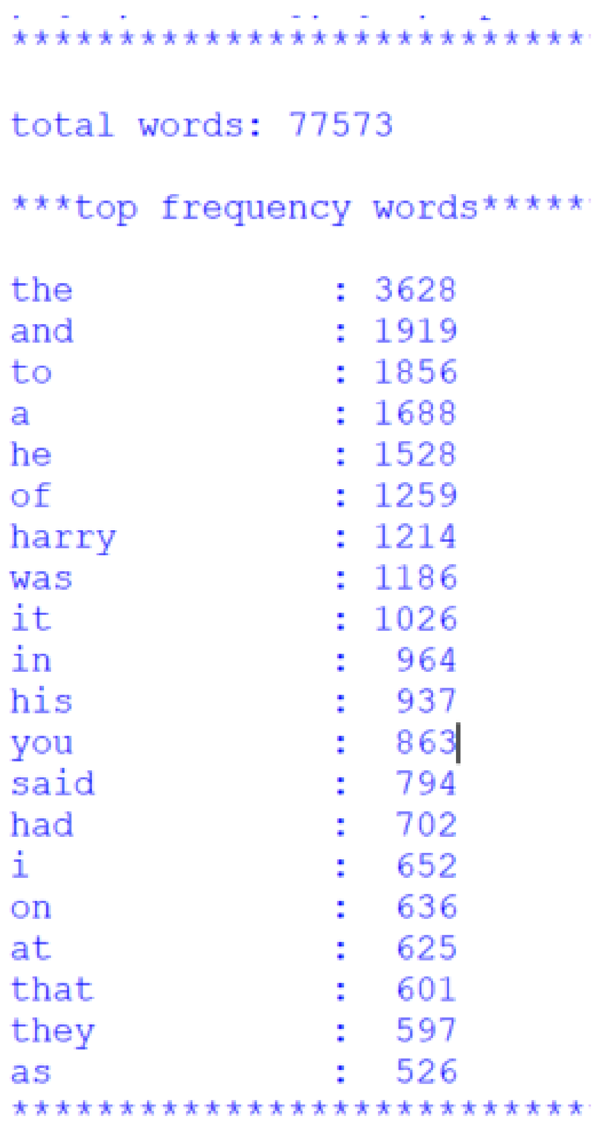
```
def report(words,uniqueCounts,number):
    print('******************************\n')
    print('total words:', len(words),'\n')
    print('***top frequency words********\n')
    uniqueCounts.sort(reverse=True)
    for w in uniqueCounts[0:number]:
        print(f'{w[1]:15}: {w[0]:4}')
    print('******************************')
```

Method: sort()

```
[  [3628, 'the'], [1919, 'and'], [1856, 'to'], [1688, 'a'], [1528, 'he'], [1259, 'of'], [1214, 'harry'], [1186, 'was'] ... ]
```



```python
# V0 program to report on top most frequently occuring words in an eBook
# Set to 'Harry Potter and the Sorceror's Stone'

def main():
    words = read_words('Harry_Potter_and_the_Sorcerer.txt')
    words = clean(words)                    #remove non alpha characters
    uniques = unique(words)                 #get all the unique words of the text
    uniqueCounts = count(uniques, words)    #count how often each unique word occurs
    report(words, uniqueCounts, 20)

def read_words(file):
    inFile = open(file, 'r')
    textOfFile = inFile.read()
    inFile.close()
    return textOfFile.split(' ')

def clean(words):
    cleanWords = []
    for w in range(len(words)):
        cleanWord = ''
        for ch in words[w]:
            if ch.isalpha():
                cleanWord += ch
        if cleanWord != '':
            cleanWords.append(cleanWord.lower())
    return cleanWords

def unique(words):
    uniques = []
    for word in words:
        if word not in uniques:
            uniques.append(word)
    return uniques

def count(uniques, words):
    wordCounts = [ ]
    for w in range(len(uniques)):
        wordCounts.append([words.count(uniques[w]),uniques[w]])
    return wordCounts

def report(words,uniqueCounts,number):
    print('****************************\n')
    print('total words:',len(words))
    print('total unique words:', len(uniqueCounts),'\n')
    print('***top frequency words********\n')
    uniqueCounts.sort(reverse=True)
    for w in uniqueCounts[0:number]:
        print(f'{w[1]:15}: {w[0]:4}')
    print('******************************')

main()
```



```
**************************

total words: 77573

***top frequency words*****

the            : 3628
and            : 1919
to             : 1856
a              : 1688
he             : 1528
of             : 1259
harry          : 1214
was            : 1186
it             : 1026
in             :  964
his            :  937
you            :  863
said           :  794
had            :  702
i              :  652
on             :  636
at             :  625
that           :  601
they           :  597
as             :  526
**************************
```

- What are the new words created by Harry Potter that are not part of the common English language?

```
# program to report on top most frequently occuring words in eBook
def main():
    words = read_words('Harry_Potter_and_the_Sorcerer.txt')
    words = clean(words)                     #remove non alpha characters
    uniques = unique(words)                  #get all the unique words of the text
    uniqueCounts = count(uniques, words)     #count how often each unique word occurs
    report(words, uniqueCounts, 20)
```

- What are the new words created by Harry Potter that are not part of the common English language?

```
# program to report on top most frequently occurring words in eBook
def main():
    words = read_words('Harry_Potter_and_the_Sorcerer.txt')
    words = clean(words)                       #remove non alpha characters
    specials = special(words)                  #remove common English words
    uniques = unique(specials)                 #get all the unique words of the text
    uniqueCounts = count(uniques, words)       #count how often each unique word occurs
    report(words, uniqueCounts, 20)
```

'english_top500.txt

the of to and a in is it you that he was for on are with as I his they be at one have this from or had by hot but some what there we can out other were all your when up use word

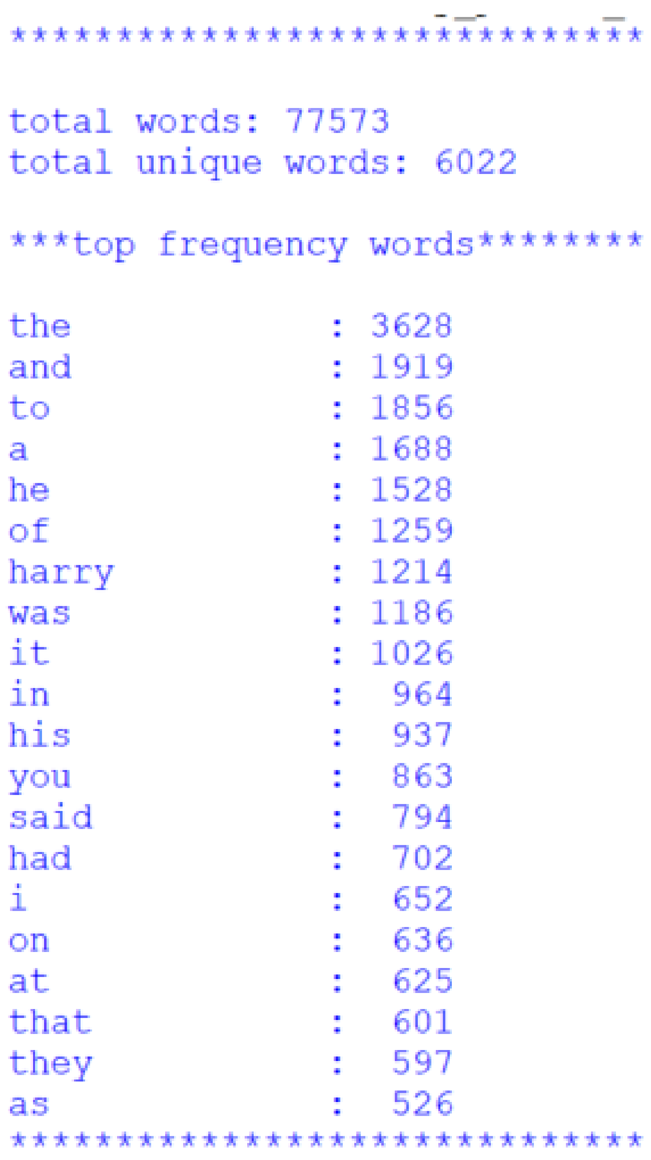
```
def special(words):
    inFile = open('english_top500.txt', 'r')
    english = inFile.read()
    inFile.close()
    special = [ ]
    for word in words:
        if word not in english:
            special.append(word)
    return special
```

Pattern: building a list

specials

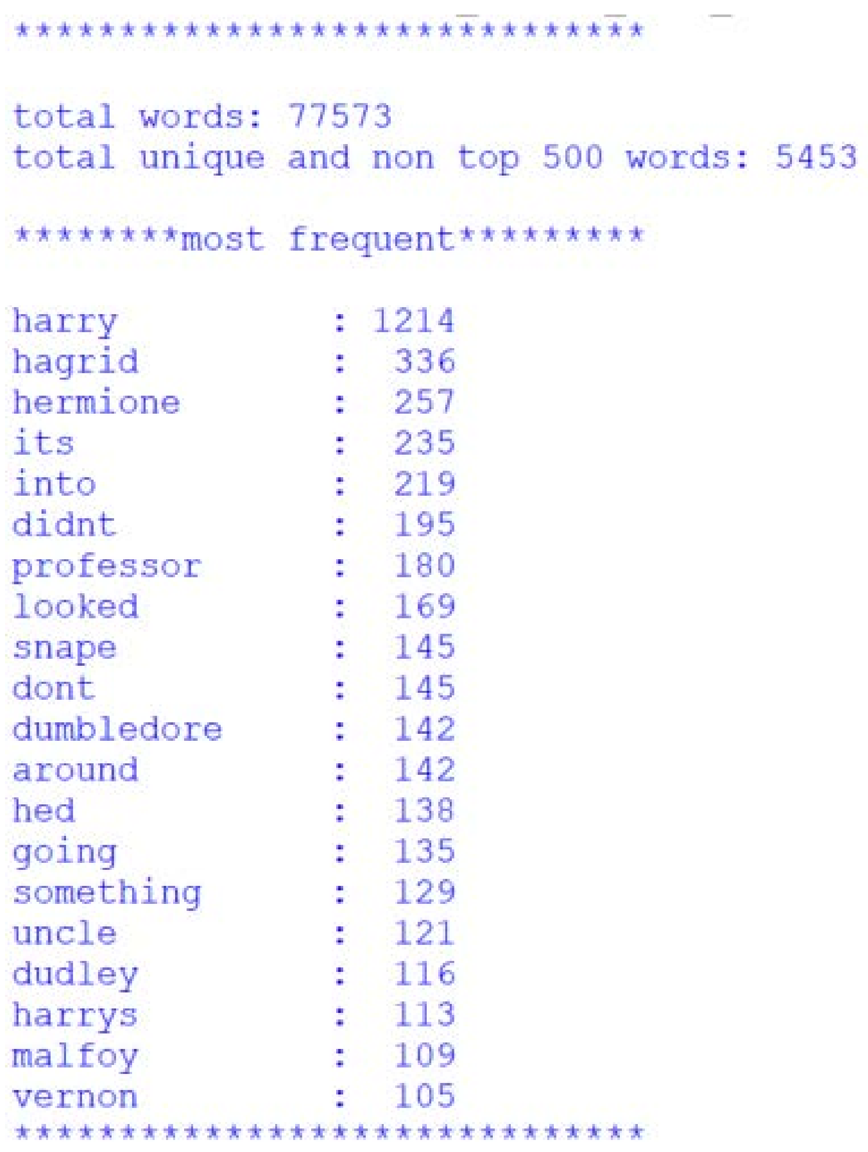
['sorcerers', 'dursley', 'privet', 'youd', 'didnt', 'nonsense', 'dursley', 'grunnings', 'drills', 'beefy', 'mustache', 'dursley', 'craning', 'fences', 'spying', 'dursleys', 'dudley', 'finer', 'dursleys', 'didnt', ...]

harry_potter_word_counter_v0



```
******************************

total words: 77573
total unique words: 6022

***top frequency words********

the                 : 3628
and                 : 1919
to                  : 1856
a                   : 1688
he                  : 1528
of                  : 1259
harry               : 1214
was                 : 1186
it                  : 1026
in                  :  964
his                 :  937
you                 :  863
said                :  794
had                 :  702
i                   :  652
on                  :  636
at                  :  625
that                :  601
they                :  597
as                  :  526
******************************
```

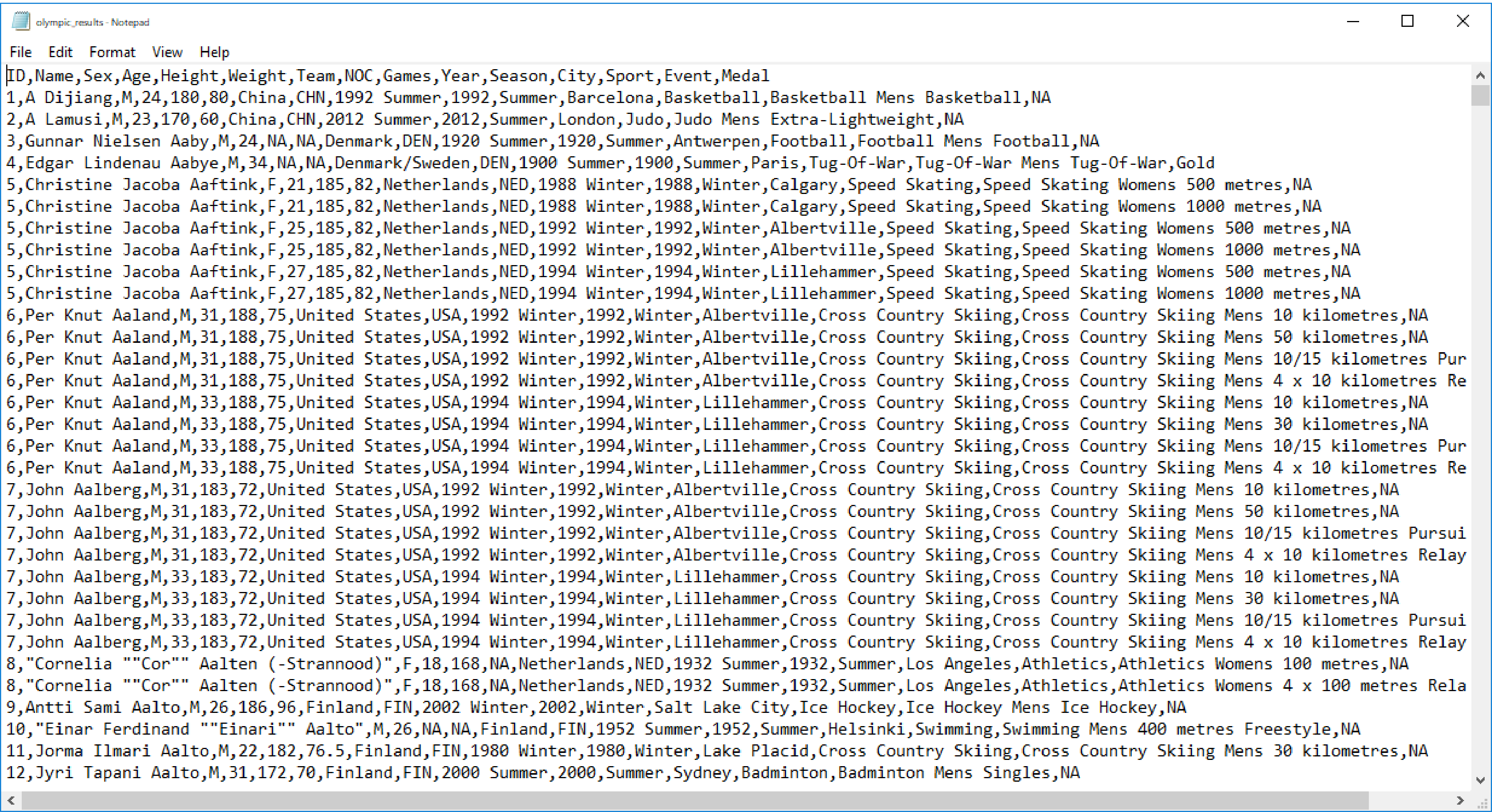
harry_potter_word_counter_v1



```
******************************

total words: 77573
total unique and non top 500 words: 5453

********most frequent*********

harry               : 1214
hagrid              :  336
hermione            :  257
its                 :  235
into                :  219
didnt               :  195
professor           :  180
looked              :  169
snape               :  145
dont                :  145
dumbledore          :  142
around              :  142
hed                 :  138
going               :  135
something           :  129
uncle               :  121
dudley              :  116
harrys              :  113
malfoy              :  109
vernon              :  105
******************************
```

- What is the average age of all the athletes taking part in the Olympic Games between 1896 - 2016?

'olympic_results.csv' 271116 lines of data



```
ID,Name,Sex,Age,Height,Weight,Team,NOC,Games,Year,Season,City,Sport,Event,Medal
1,A Dijiang,M,24,180,80,China,CHN,1992 Summer,1992,Summer,Barcelona,Basketball,Basketball Mens Basketball,NA
2,A Lamusi,M,23,170,60,China,CHN,2012 Summer,2012,Summer,London,Judo,Judo Mens Extra-Lightweight,NA
3,Gunnar Nielsen Aaby,M,24,NA,NA,Denmark,DEN,1920 Summer,1920,Summer,Antwerpen,Football,Football Mens Football,NA
4,Edgar Lindenau Aabye,M,34,NA,NA,Denmark/Sweden,DEN,1900 Summer,1900,Summer,Paris,Tug-Of-War,Tug-Of-War Mens Tug-Of-War,Gold
5,Christine Jacoba Aaftink,F,21,185,82,Netherlands,NED,1988 Winter,1988,Winter,Calgary,Speed Skating,Speed Skating Womens 500 metres,NA
5,Christine Jacoba Aaftink,F,21,185,82,Netherlands,NED,1988 Winter,1988,Winter,Calgary,Speed Skating,Speed Skating Womens 1000 metres,NA
5,Christine Jacoba Aaftink,F,25,185,82,Netherlands,NED,1992 Winter,1992,Winter,Albertville,Speed Skating,Speed Skating Womens 500 metres,NA
5,Christine Jacoba Aaftink,F,25,185,82,Netherlands,NED,1992 Winter,1992,Winter,Albertville,Speed Skating,Speed Skating Womens 1000 metres,NA
5,Christine Jacoba Aaftink,F,27,185,82,Netherlands,NED,1994 Winter,1994,Winter,Lillehammer,Speed Skating,Speed Skating Womens 500 metres,NA
5,Christine Jacoba Aaftink,F,27,185,82,Netherlands,NED,1994 Winter,1994,Winter,Lillehammer,Speed Skating,Speed Skating Womens 1000 metres,NA
6,Per Knut Aaland,M,31,188,75,United States,USA,1992 Winter,1992,Winter,Albertville,Cross Country Skiing,Cross Country Skiing Mens 10 kilometres,NA
6,Per Knut Aaland,M,31,188,75,United States,USA,1992 Winter,1992,Winter,Albertville,Cross Country Skiing,Cross Country Skiing Mens 50 kilometres,NA
6,Per Knut Aaland,M,31,188,75,United States,USA,1992 Winter,1992,Winter,Albertville,Cross Country Skiing,Cross Country Skiing Mens 10/15 kilometres Pur
6,Per Knut Aaland,M,31,188,75,United States,USA,1992 Winter,1992,Winter,Albertville,Cross Country Skiing,Cross Country Skiing Mens 4 x 10 kilometres Re
6,Per Knut Aaland,M,33,188,75,United States,USA,1994 Winter,1994,Winter,Lillehammer,Cross Country Skiing,Cross Country Skiing Mens 10 kilometres,NA
6,Per Knut Aaland,M,33,188,75,United States,USA,1994 Winter,1994,Winter,Lillehammer,Cross Country Skiing,Cross Country Skiing Mens 30 kilometres,NA
6,Per Knut Aaland,M,33,188,75,United States,USA,1994 Winter,1994,Winter,Lillehammer,Cross Country Skiing,Cross Country Skiing Mens 10/15 kilometres Pur
6,Per Knut Aaland,M,33,188,75,United States,USA,1994 Winter,1994,Winter,Lillehammer,Cross Country Skiing,Cross Country Skiing Mens 4 x 10 kilometres Re
7,John Aalberg,M,31,183,72,United States,USA,1992 Winter,1992,Winter,Albertville,Cross Country Skiing,Cross Country Skiing Mens 10 kilometres,NA
7,John Aalberg,M,31,183,72,United States,USA,1992 Winter,1992,Winter,Albertville,Cross Country Skiing,Cross Country Skiing Mens 50 kilometres,NA
7,John Aalberg,M,31,183,72,United States,USA,1992 Winter,1992,Winter,Albertville,Cross Country Skiing,Cross Country Skiing Mens 10/15 kilometres Pursui
7,John Aalberg,M,31,183,72,United States,USA,1992 Winter,1992,Winter,Albertville,Cross Country Skiing,Cross Country Skiing Mens 4 x 10 kilometres Relay
7,John Aalberg,M,33,183,72,United States,USA,1994 Winter,1994,Winter,Lillehammer,Cross Country Skiing,Cross Country Skiing Mens 10 kilometres,NA
7,John Aalberg,M,33,183,72,United States,USA,1994 Winter,1994,Winter,Lillehammer,Cross Country Skiing,Cross Country Skiing Mens 30 kilometres,NA
7,John Aalberg,M,33,183,72,United States,USA,1994 Winter,1994,Winter,Lillehammer,Cross Country Skiing,Cross Country Skiing Mens 10/15 kilometres Pursui
7,John Aalberg,M,33,183,72,United States,USA,1994 Winter,1994,Winter,Lillehammer,Cross Country Skiing,Cross Country Skiing Mens 4 x 10 kilometres Relay
8,"Cornelia ""Cor"" Aalten (-Strannood)",F,18,168,NA,Netherlands,NED,1932 Summer,1932,Summer,Los Angeles,Athletics,Athletics Womens 100 metres,NA
8,"Cornelia ""Cor"" Aalten (-Strannood)",F,18,168,NA,Netherlands,NED,1932 Summer,1932,Summer,Los Angeles,Athletics,Athletics Womens 4 x 100 metres Rela
9,Antti Sami Aalto,M,26,186,96,Finland,FIN,2002 Winter,2002,Winter,Salt Lake City,Ice Hockey,Ice Hockey Mens Ice Hockey,NA
10,"Einar Ferdinand ""Einari"" Aalto",M,26,NA,NA,Finland,FIN,1952 Summer,1952,Summer,Helsinki,Swimming,Swimming Mens 400 metres Freestyle,NA
11,Jorma Ilmari Aalto,M,22,182,76.5,Finland,FIN,1980 Winter,1980,Winter,Lake Placid,Cross Country Skiing,Cross Country Skiing Mens 30 kilometres,NA
12,Jyri Tapani Aalto,M,31,172,70,Finland,FIN,2000 Summer,2000,Summer,Sydney,Badminton,Badminton Mens Singles,NA
```

```
def main():
    data = read_lines('olympic_results.csv')
    results = itemize(data)
    ages = extract_ages(results)
    report(ages)


def read_lines(file):
    inFile = open(file, 'r')
    alldata = inFile.readlines()
    inFile.close()
    return alldata
```

```
['ID,Name,Sex,Age,Height,Weight,Team,NOC,Games,Year,Season,City,Sport,Event,Medal\n', '1,A Dijiang,M,24,180,80,China,CHN,1992 Summer,1992,Summer,Barcelona,Basketball,Basketball Mens Basketball,NA\n', '2,A Lamusi,M,23,170,60,China,CHN,2012 Summer,2012,Summer,London,Judo,Judo Mens Extra-Lightweight,NA\n', '3,Gunnar Nielsen Aaby,M,24,NA,NA,Denmark,DEN,1920 Summer,1920,Summer,Antwerpen,Football,Football Mens Football,NA\n', '4,Edgar Lindenau Aabye,M,34,NA,NA,Denmark/Sweden,DEN,1900 Summer,1900,Summer,Paris,Tug-Of-War,Tug-Of-War Mens Tug-Of-War,Gold\n',  ... ]
```

data

```
['ID,Name,Sex,Age,Height,Weight,Team,NOC,Games,Year,Season,City,Sport,Event,Medal\n', '1,A Dijiang,M,24,180,80,China,CHN,1992 Summer,1992,Summer,Barcelona,Basketball,Basketball Mens Basketball,NA\n', '2,A Lamusi,M,23,170,60,China,CHN,2012 Summer,2012,Summer,London,Judo,Judo Mens Extra-Lightweight,NA\n', '3,Gunnar Nielsen Aaby,M,24,NA,NA,Denmark,DEN,1920 Summer,1920,Summer,Antwerpen,Football,Football Mens Football,NA\n', '4,Edgar Lindenau Aabye,M,34,NA,NA,Denmark/Sweden,DEN,1900 Summer,1900,Summer,Paris,Tug-Of-War,Tug-Of-War Mens Tug-Of-War,Gold\n',  ... ]
```

Method: pop()

```
def itemize(data):
    data.pop(0)        #removes header line
    results = [ ]
    for line in range(len(data)):
        results.append(data[line].split(','))    #creates a list of lists
    return results
```

Pattern: building a list

Method: split()

results

```
[['1', 'A Dijiang', 'M', '24', '180', '80', 'China', 'CHN', '1992 Summer', '1992', 'Summer', 'Barcelona', 'Basketball', 'Basketball Mens Basketball', 'NA\n'], ['2', 'A Lamusi', 'M', '23', '170', '60', 'China', 'CHN', '2012 Summer', '2012', 'Summer', 'London', 'Judo', 'Judo Mens Extra-Lightweight', 'NA\n'], ['3', 'Gunnar Nielsen Aaby', 'M', '24', 'NA', 'NA', 'Denmark', 'DEN', '1920 Summer', '1920', 'Summer', 'Antwerpen', 'Football', 'Football Mens Football', 'NA\n'], ['4', 'Edgar Lindenau Aabye', 'M', '34', 'NA', 'NA', 'Denmark/Sweden', 'DEN', '1900 Summer', '1900', 'Summer', 'Paris', 'Tug-Of-War', 'Tug-Of-War Mens Tug-Of-War', 'Gold\n'], ['5', 'Christine Jacoba Aaftink', 'F', '21', '185', '82', 'Netherlands', 'NED', '1988 Winter', '1988', 'Winter', 'Calgary', 'Speed Skating', 'Speed Skating Womens 500 metres', 'NA\n'] ...]
```

results

```
[['1', 'A Dijiang', 'M', '24', '180', '80', 'China', 'CHN', '1992 Summer', '1992', 'Summer', 'Barcelona', 'Basketball', 'Basketball Mens Basketball', 'NA\n'], ['2', 'A Lamusi', 'M', '23', '170', '60', 'China', 'CHN', '2012 Summer', '2012', 'Summer', 'London', 'Judo', 'Judo Mens Extra-Lightweight', 'NA\n'], ['3', 'Gunnar Nielsen Aaby', 'M', '24', 'NA', 'NA', 'Denmark', 'DEN', '1920 Summer', '1920', 'Summer', 'Antwerpen', 'Football', 'Football Mens Football', 'NA\n'], ['4', 'Edgar Lindenau Aabye', 'M', '34', 'NA', 'NA', 'Denmark/Sweden', 'DEN', '1900 Summer', '1900', 'Summer', 'Paris', 'Tug-Of-War', 'Tug-Of-War Mens Tug-Of-War', 'Gold\n'], ['5', 'Christine Jacoba Aaftink', 'F', '21', '185', '82', 'Netherlands', 'NED', '1988 Winter', '1988', 'Winter', 'Calgary', 'Speed Skating', 'Speed Skating Womens 500 metres', 'NA\n'] ...]
```

```
def extract_ages(results):
    ages = [ ]
    for participation in results:
        age = participation[3]
        if not(age in 'FMNA'):
            ages.append(int(age))
    return ages
```

Pattern: building a list

ages

```
[24, 23, 24, 34, 21, 21, 25, 25, 27, 27, 31, 31, 31, 31, 33, 33, 33, 33, 31, 31, 31, 31, 33, 33, 33, 33, 18, 18, 26, 26 ... ]
```

```
def main():
    data = read_lines('olympic_results.csv')
    results = itemize(data)
    ages = extract_ages(results)
    report(ages)
```

```
[24, 23, 24, 34, 21, 21, 25, 25, 27, 27, 31, 31, 31, 31, 33, 33, 33, 33, 31, 31, 31, 31, 33,
33, 33, 33, 18, 18, 26, 26 ... ]
```

```
def report(ages):
    print('Ages of athletes')
    print('*****************')
    print(f'Range:   {min(ages)} - {max(ages)}')
    print(f'Mean:    {mean(ages):<10.2f}')
    print(f'Median: {median(ages):<10.2f}')
    print(f'Mode:    {mode(ages):<10.0f}')
```

```
Ages of athletes
*****************
Range:  10 - 97
Mean:   25.56
Median: 24.00
Mode:   23
```

Exercise 4: Write a program to report the **mean** value of a list of integers.
Remember that the mean = total score / number of scores.

Exercise 5: Write a program to report the **median** value of a list of integers.
Remember that the median is the middle score, if we put all the scores in order.

3, 5, 12, 345, 567 .... median = 12

If there are an even number of scores, then we take the mean of the middle two.

3, 5, 12, 345, 567, 599 .... median = (12 + 345)/2 = 178.5

Exercise 6: Write a program to report the **mode** value of a list of integers.
Remember that the mode is the most frequent value among all the scores.

```
def mean(x):

    ...
    ...

    return mean
```

```
def median(x):

    ...
    ...

    return mean
```

```
def mode(x):

    ...
    ...

    return mean
```

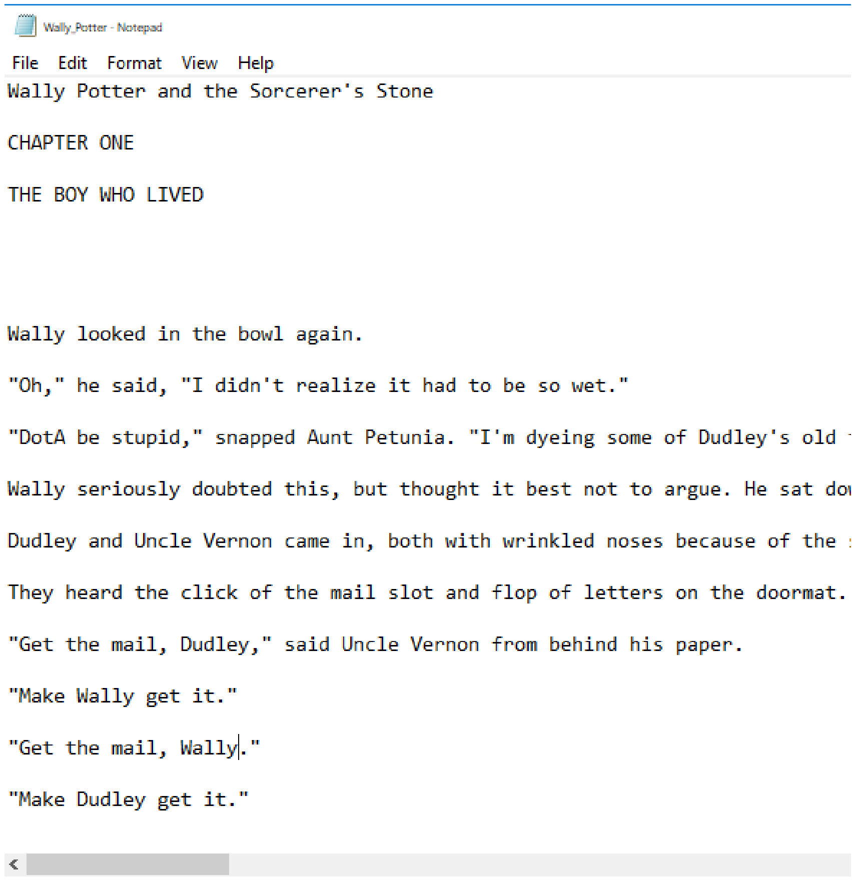
Writing to files

We can also write data back to create new files ...

creates a file object

```
outfile = open('test_data.txt' , 'w')

for i in range(1000):
        print(i, file=outfile)

outfile.close()
```

```
#program to swap user for Harry Potter in eBook

def main():
    yourName = input('Your given name: ')
    text = read_eBook()
    newtext = change(text, yourName)
    write_eBook(newtext, yourName)

def read_eBook():
    infile = open('Harry_Potter_and_the_Sorcerer.txt', 'r')
    lines = infile.readlines()
    infile.close()
    return lines

def change(text, yourName):
    newtext = [ ]
    for line in text:
        if 'Harry' in line:
            newline = swap(line, 'Harry', yourName)
            newtext.append(newline)
        else:
            newtext.append(line)
    return newtext

def swap(line, name1, name2):
    words = line.split()
    newline = ''
    for word in words:
        if word == name1:
            newline += name2 +' '
        else:
            newline += word + ' '
    return newline

def write_eBook(newtext, yourName):
    fileName = yourName + '_Potter.txt'
    outfile = open(fileName, 'w')
    for line in newtext:
        print(line[:-1], file=outfile)
    outfile.close()
    print('New eBook', fileName, 'created.')

main()
```



```
Wally Potter and the Sorcerer's Stone

CHAPTER ONE

THE BOY WHO LIVED




Wally looked in the bowl again.

"Oh," he said, "I didn't realize it had to be so wet."

"DotA be stupid," snapped Aunt Petunia. "I'm dyeing some of Dudley's old

Wally seriously doubted this, but thought it best not to argue. He sat do

Dudley and Uncle Vernon came in, both with wrinkled noses because of the

They heard the click of the mail slot and flop of letters on the doormat.

"Get the mail, Dudley," said Uncle Vernon from behind his paper.

"Make Wally get it."

"Get the mail, Wally."

"Make Dudley get it."
```

Lecture 9: Files

COMP90059 Introduction to Programming

Wally Smith

School of Computing & Information Systems