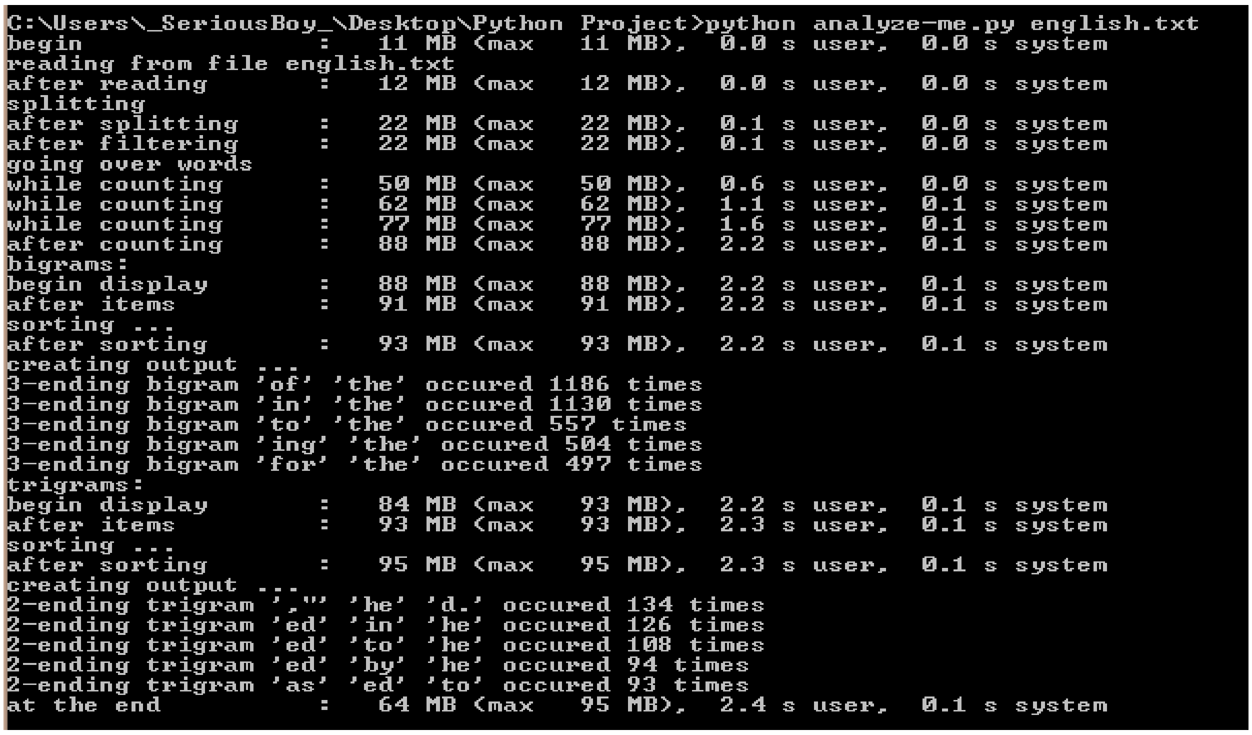
# OBJECT ORIENTED SOFTWARE DESIGN – CSE363

## PYTHON PROJECT- analyze.py



```
C:\Users\_SeriousBoy_\Desktop\Python Project>python analyze-me.py english.txt
begin                  :    11 MB (max    11 MB),  0.0 s user,  0.0 s system
reading from file english.txt
after reading          :    12 MB (max    12 MB),  0.0 s user,  0.0 s system
splitting
after splitting        :    22 MB (max    22 MB),  0.1 s user,  0.0 s system
after filtering        :    22 MB (max    22 MB),  0.1 s user,  0.0 s system
going over words
while counting         :    50 MB (max    50 MB),  0.6 s user,  0.0 s system
while counting         :    62 MB (max    62 MB),  1.1 s user,  0.1 s system
while counting         :    77 MB (max    77 MB),  1.6 s user,  0.1 s system
after counting         :    88 MB (max    88 MB),  2.2 s user,  0.1 s system
bigrams:
begin display          :    88 MB (max    88 MB),  2.2 s user,  0.1 s system
after items            :    91 MB (max    91 MB),  2.2 s user,  0.1 s system
sorting ...
after sorting          :    93 MB (max    93 MB),  2.2 s user,  0.1 s system
creating output ...
3-ending bigram 'of' 'the' occured 1186 times
3-ending bigram 'in' 'the' occured 1130 times
3-ending bigram 'to' 'the' occured 557 times
3-ending bigram 'ing' 'the' occured 504 times
3-ending bigram 'for' 'the' occured 497 times
trigrams:
begin display          :    84 MB (max    93 MB),  2.2 s user,  0.1 s system
after items            :    93 MB (max    93 MB),  2.3 s user,  0.1 s system
sorting ...
after sorting          :    95 MB (max    95 MB),  2.3 s user,  0.1 s system
creating output ...
2-ending trigram ',"' 'he' 'd.' occured 134 times
2-ending trigram 'ed' 'in' 'he' occured 126 times
2-ending trigram 'ed' 'to' 'he' occured 108 times
2-ending trigram 'ed' 'by' 'he' occured 94 times
2-ending trigram 'as' 'ed' 'to' occured 93 times
at the end             :    64 MB (max    95 MB),  2.4 s user,  0.1 s system
```

UFUK GÜRBÜZ – 150113058

## PROCEDURES

### STEP 3

I created a class which name is "NgramCounter(n,m)" that can count n-grams and can take final 3 characters of word. I replaced BigramCounter and TrigramCounter by NgramCounter. My program is wanting a input(number) for ngram counter. Then I checked my class for bigram and trigram and compared my results with teacher's results. The everything is ok. The results are true all. I saved my program which name is "analyzeRefactorNGram".

### STEP 4

I ran analyzeRefactor1NGram.py for bigram of 3 final characters (2,3). I observed, maximum memory is 45 MB, user time is 0.7s and system time is 0.1 s for english.txt. Then I ran german.txt and I observed, maximum memory is 68 MB, user time is 1.1s and system time is 0.1 s. Finally, I observed, maximum memory is 1545 MB, user time is 33.7s and system time is 0.8 s for turkish.txt. I didn't see so much differrence between analyze.py and analyzeRefactor1Ngram.py for bigram which endings of length 3 (2,3). Then I saved it which name is "*analyzeRefactor1NGram.py*".

### STEP 5

In this step, I deleted unused variables and objects. As a result, the memory space of program is decreased. When I ran my program for english.txt and bigram of 3 final characters (2,3), I observed, maximum memory is 44 MB, user time is 0.5s and system time is 0.1 s. Then I ran german.txt and I observed, maximum memory is 65 MB, user time is 1.2s and system time is 0.1 s. Finally, I observed, maximum memory is 1437 MB, user time is 36.5s and system time is 1.2 s for turkish.txt. Then I saved it which name is "*analyzeRefactor2Delete.py*".

### STEP 6

In this step, I implemented iterator operation. The program was fast so much and began using less memory space. Because, it is reading file line by line and is deleting immediately previous line. So, there isn't unnecessary memory space. When I ran my program for english.txt and bigram which endings of length 3 (2,3), I observed, maximum memory is 30 MB, user time is 1.9s and system time is 1.2 s. Then I ran german.txt and I observed, maximum memory is 42 MB, user time is 4.2s and system time is 2.9 s. Finally, I observed, maximum memory is 440 MB, user time is 145.4s and system time is 116.9 s for turkish.txt. Then I saved it which name is "*analyzeRefactor3Iterator.py*".

## STEP 7

I created a function which name is displayKMostFrequentNGramsInFile(k,n,m,filename). The function  opens which file, prints k most frequent n-grams of m final characters, counts n-grams which word endings of length m. Then it creates NgramCounter() class object. In my main function, I called that function 7 times: show the 30 most frequent 2-grams of 2/3/4 final characters, show the 20 most frequent 3-grams of 2/3 final characters, show the 15 most frequent 4-grams of 2/3 final characters. I ran my program and observed the results of function. Then I saved it which name is “*analyzeRefactor4More.py*”.

## STEP 8

| **Program** | **Lines of code** | english.txt | | | german.txt | | | turkish.txt | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **Memory Max (MB)** | **User Time (s)** | **System Time (s)** | **Memory Max (MB)** | **User Time (s)** | **System Time (s)** | **Memory Max (MB)** | **User Time (s)** | **System Time (s)** |
| ***analyze.py*** | 137 | 95 | 2.5 | 0.1 | 135 | 5.5 | 0.1 | 2893 | 132 | 1.7 |
| ***analyze1NGram.py(2,3)*** | 103 | 45 | 0.7 | 0.1 | 68 | 1.1 | 0.1 | 1545 | 33.7 | 0.8 |
| ***analyze2Delete.py(2,3)*** | 124 | 44 | 0.5 | 0.1 | 65 | 1.2 | 0.1 | 1437 | 36.5 | 1.2 |
| ***analyze3Iterator.py(2,3)*** | 129 | 30 | 1.9 | 1.2 | 42 | 4.2 | 2.9 | 440 | 145.4 | 116.9 |
| ***analyze4More.py***_(30,2,2)_ | 159 | 20 | 2.5 | 1.3 | 22 | 3.8 | 3.3 | 114 | 146.8 | 121.6 |
| ***analyze4More.py***_(30,2,3)_ | 159 | 30 | 1.6 | 1.1 | 41 | 3.7 | 2.2 | 439 | 153.6 | 135.9 |
| ***analyze4More.py***_(30,2,4)_ | 159 | 38 | 2.1 | 1.4 | 62 | 10.2 | 7.4 | 855 | 154.8 | 128.8 |
| ***analyze4More.py***_(20,3,2)_ | 159 | 46 | 1.9 | 1.3 | 59 | 5.5 | 3.7 | 804 | 158.3 | 120.7 |
| ***analyze4More.py***_(20,3,3)_ | 159 | 52 | 4.1 | 3.1 | 93 | 4.4 | 2.7 | 2055 | 195.3 | 130.6 |
| ***analyze4More.py***_(15,4,2)_ | 159 | 62 | 3.2 | 2.4 | 106 | 4.2 | 2.6 | 2473 | 166.8 | 106.8 |
| ***analyze4More.py***_(15,4,3)_ | 159 | 58 | 1.6 | 1.2 | 114 | 4.3 | 2.1 | 2913 | 172.5 | 110.6 |

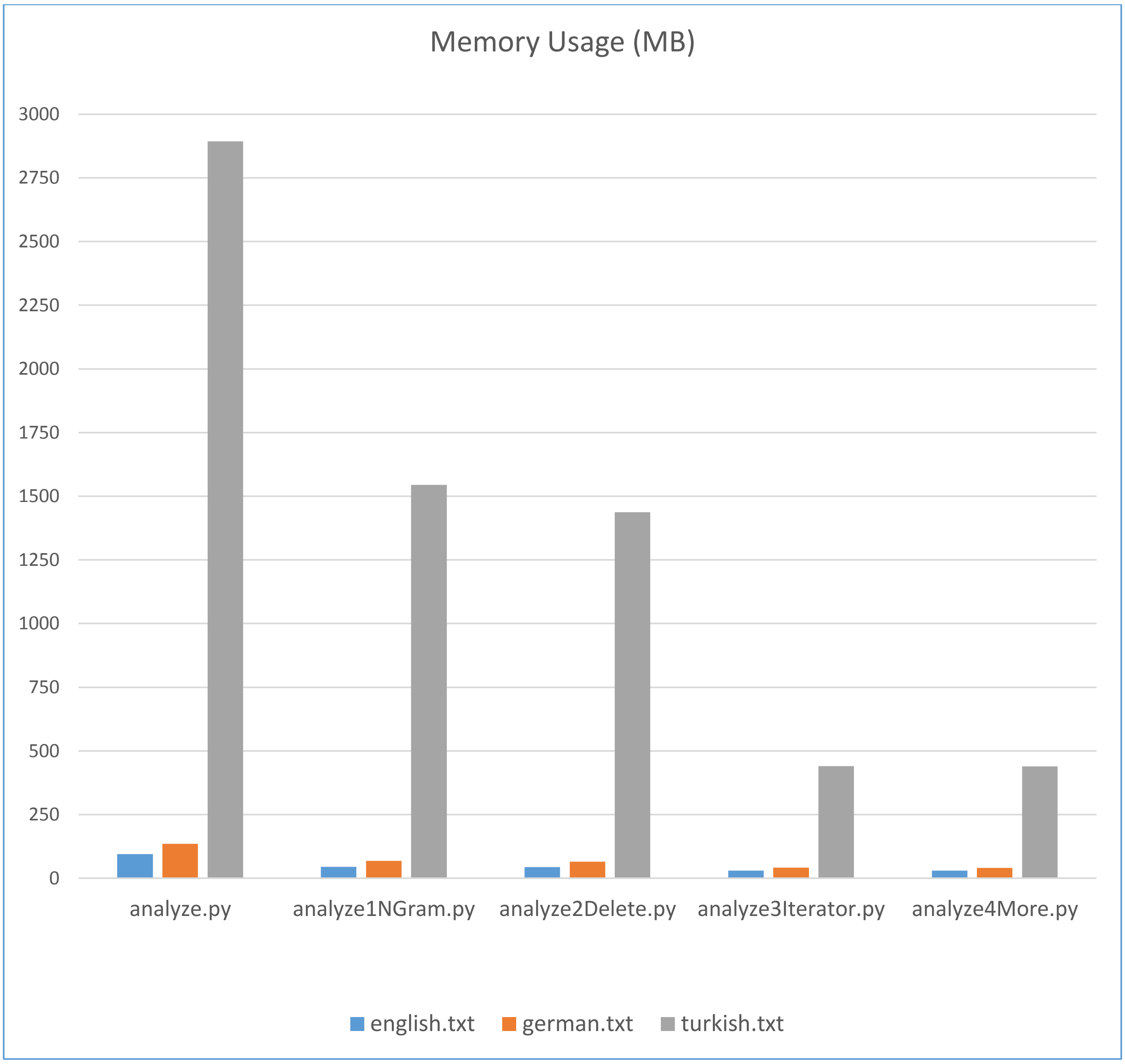
## CONCLUSION

After run  these programs, I get conclusion and more analysis. They are following:

- As I proceed our code get more understandable/readable except Iterator part.When I came into Iterator our code started to get less understandable.And the most readable our code is when I arrived analyzeRefactor4More.py code
- After testing analyzeRefactor1Ngram we see that there is not much difference between analyze.py and analyzerefactor1ngram.py.It is logical because I only converted bigram and trigram into ngram and did not save any memory or did not change our style to read files.
- When I ran analyzeRefactor2Delete.py program for all text file, I saw that memory usage less than analyzeRefactor1Ngram.  Because I deleted unused and unnecessarry variable from my program.

- When I run analyzeRefactor3Iterator.py program, I saw that memory usage so much less than analyzeRefactor2Delete.py. Reading line by line affects time negatively. However, the reading line by line opened so much space in memory.
- When I ran analyzeRefactor4More.py program, I saw the time is longer than analyzeRefactor3Iterator.py program.

## MEMORY USAGE GRAPHIC


Memory Usage (MB)
analyze.py
analyze1NGram.py
analyze2Delete.py
analyze3Iterator.py
analyze4More.py
english.txt
german.txt
turkish.txt