wc final report for mbaxsca5

Overview

1.The requirements met

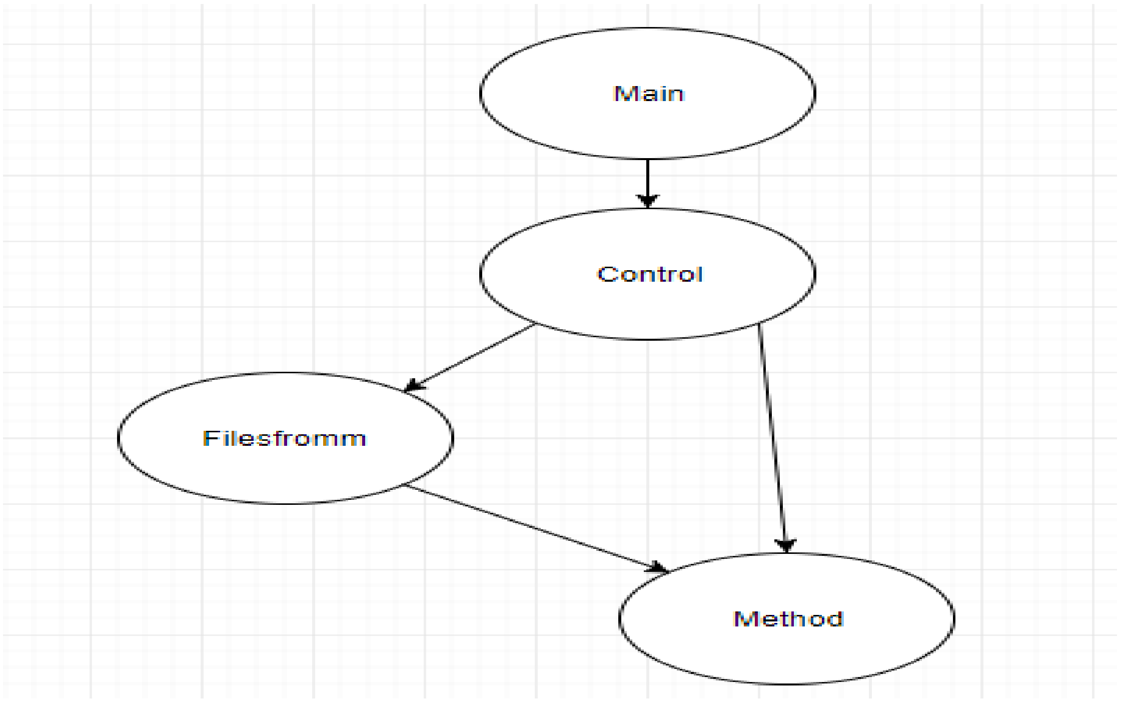
Although some minor functionalities may be missed, the tasks mentioned in coursework have been coded properly in order to make the wc.py work like wc.All the flags(-l, -w, -c, -m, -L, –files0-from, –help and –version) working for wc have been added to wc.py for illustrating the number of lines, bytes, words etc.In addition, combinations as well as the long forms of flags can be used with multiple filenames.If '-' is written on the command line, users should type their own inputs to get the proper output.Furthermore, if there is no flag given on command line, the program allows users to give input.Mostly all error messages occur in various situations like in wc whereas some of the messages might be missed.The libraries apart from argparse and sys have not been included.Finally, the output is tab delimited.

2.The requirements missed

Reading pdf file has been missed in spite of the fact that wc.py supports various file types such as; bin, py, xml, txt, html etc.In addition to reading pdf files, the reason of the message "File name too long" which occurs while using the flag --files0-from has not been found.Therefore,it has not been fixed.

Product:Design and implementation

1.Architecture



1.Architecture of wc.py

The architecture of wc.py can be reckoned like MVC.However, in some circumstances printing the output happens in different functions than the main method.Therefore, the architecture can not be described as MVC.It can be considered that, there is a hierarchical relationship between functions because the main function calls the control and the control calls either filesfromm or method.

2.Source code metrics

The total number of lines in wc.py is 761 although the proper number could be around 400 which has been discussed during the class.

There are totally 15 functions and 1 class in the program.Talking about the functions,10 of them are about calculating the longest line, lines, charachters, words and bytes for the standard input or file given.

Function	Task
Methodline	Returns the number of lines in a file
Methodword	Returns the number of words in a file
Methodbyte	Returns the number of bytes in a file
Methodcharcount	Returns the number of characters in a file
Methodlongestline	Returns the length of longest line in a file

2. Table of functions

The functions mentioned in table get files as an argument instead of user inputs.Apart from those 5 functions,the other 5 functions about calculating the longestline,lines,charachters, words and bytes take the user inputs as an argument.Therefore,I described the functions names like usermethodline, usermethodword, usermethodbyte,usermethodlongestline and usermethodcharcount.

A function named as main preprocess the input list in order to make the list ready for controlling in argparse.Once preprocessing is finished, input list is sent to a function called control.In control function,argparse is used for validating the flags used.Although argparse has its own error messages for invalid flags, overriding the error function in argparse was necesarry in order to show the same error messages with wc's.Therefore,argparse class has been defined and the messages have been modified in error function.Following this, if the flag --files0-from is seen in input list, method described as filesfromm will be called.Filesfromm reads input from the files specified by NUL-terminated and sends the file names to a function called method.

Method is the longest function in wc.py which is around 272 lines.It can call 10 functions depending on the input given.The fundamental task of method is checking the flags in input list and calculating the total which is relevant to flags as well as filenames typed.Global variables have been used in method in order to save the previous information.

Almost all the units in wc.py interact with method because printing the results of calculations occurs in method.

3.Correctness testing

Using doctest has helped me to control my program even after some minor modifications.Approximately 83 system tests have been done in order to further suitably without missing possible bugs which arise from configurations.

I have done 67 tests by unittest.The number of tests can be less than it is supposed because standart input was tough for assessing the output .In addition,some functions in wc.py possess an option which is sys.exit().For instance,if argparse detects any unvalid flag,the errorr message will be shown and program will be closed immediately without any calculation.

The percentage of wc.py’s line coverage is 80.The coverage tool has been run on unit tests and the result was not 100 percent.Not only using standard inputs in some functions but also the option system exit can be considered as the two primary reasons explaining why the proportion is less than 100.

4.Performance

	Avarage	Min	Max	Median
wc	0m0.001s	0m0.000s	0m0.001s	0m0.001s
wc.py	0m0.151s	0m0.000s	0m0.168s	0m0.154s

3.Efficiency test report

It can be noticed that, there is a remarkable difference between wc and wc.py about the time spent for illustrating the output.The CPU time of wc is mostly stable although the time for wc.py depends on the type as well as the combination of flags used.Therefore,wc.py does not have a clear time definition for showing the output.

Process

1.The first version of wc.py

The first version has three functionalities which are about showing the number of lines, bytes and words for a single file given.There are no error detections or usage of multiple flags.Therefore, the first version can be considered as a simple mechanism illustrating a few information.

2.The second version of wc.py

The second version allows users to use flags in any combinations.In addition, multiple files can be given as input and the output is appropriate about showing the total lines, bytes or words.Error checking has been done for invalid filenames as well as flags.One function called as method has been defined in order to print and save the results of calculations by using global variables.However, some error messages which is about treating ‘-‘ may be missed because there was not a suitable input list controller in wc.py.

3.The third version of wc.py

Refactoring the code has been done in order to use argparse, meaning that wc.py has been transformed to possess a proper mechanism for validating the flags used.Despite the fact that overriding the argparse class was necessary to demonstrate the proper messages,argparse has led to showing error messages confidently because correct flags have been described in it.Moreover, the code has split into many functions for testing each unit in wc.py.In other words, the testability and the readability of wc.py have increased owing to the fact that the whole code is divided into functions according to tasks.

Although in the previous versions printing the results of flags was shown with spaces as well as tabs, basic corrections about strings fixed that bug.

4.The last version of wc.py

Some functionalities have been added to wc.py such as counting characters and showing the longest line’s length.In addition, standard input is able to be taken if ‘-‘ is seen in the input list.Therefore, more functions have been described in order to calculate the standard input’s information about lines, words, bytes etc.The function called filesfromm reading input from the files specified by NUL-terminated or taking standard input if the argument is ‘-‘ has been created.Furthermore, writing new messages related to filesfromm was necessary in order to improve the variety of error messages.

The 3rd version had some bugs about the flag ‘- -‘ because the flags as well as filenames written after ‘- -‘ are considered always as files in wc.Therefore, conditional processing has been done for checking whether flag is used after or before ‘ - - ‘.Moreover, the error message “unrecognised option” is created if there is an unknown flag given.

Version	Lines of code
1th	30
2nd	155
3rd	324
4th	761

4. Table of the number of lines for each version

It can be reckoned that, adding more functionalities or constraining the program in order to manage the bugs results in a sharp increase in the number of lines written in wc.py.

Lessons Learned

1.Major challenges and felicities

In my opinion, the most challenging part of programming wc.py was overriding the error function of argparse because I have tried several techniques but after a moment I realized that without creating an argparse class showing a proper error message is tough.Moreover, pre-processing the input list to make the array ready for argparse was hard to think.In addition to argparse, researching information about the usage of python has been a difficult process due to the fact that I was not familiar with the syntax before programming wc.py.

On the other hand, optimising the program should be done by using different algorithms or by removing useless variables in order to decrease the time spent on CPU.Furthermore, new functionalities such as showing the longest word or illustrating the most repeated character can be added for improving the usefulness of the program.

2.Lesson Learned

I have the impression that, not only improving my coding skills about python but also perceiving the mechanism of a single program called wc can be listed as the two primary utilities which have developed my knowledge.Moreover, learning how to test a program with tools is one the most beneficial phenomena contributing my software engineering skills.Therefore, my approach would be more effective for recent software systems because I have learned the fundamental of reverse engineering.