EMAT10007 – Introduction to Computer Programming Assignment 2021 – Encrypted Information

Louis Oskar Swanepoel

mailto:ah21269@bristol.ac.ukah21269@bristol.ac.uk

1 Introduction

My code is separated into three main modules:

1. Main.py
2. CeasarCipher.py
3. MetricFunctions.py

Module 1 is used to allow the ask the user to input the arguments used by Module 2 and 3 to run functions that allow decryption, encryption, auto decryption and metric collection/data plotting. Each question in module 1 is designed to allow the user to input one of the options in the question and then if statements and a while loop to produce relevant error messages and to guide the user to a new question until eventually the conclusion is reached and all the arguments provided by the user are processed by the functions in Module 2 and 3 to write metrics to metrics.txt and return the encrypted, decrypted or auto-decrypted message.

- The first question is whether the user would like to use message entry mode 'm' or 'f'.
- If mode m is picked the user will input their own message to be ciphered
- If mode f is picked the user will be asked to input a file path and the file (if recognised) will subsequently be read and loaded onto the variable cipher, encrypted message or decrypted message depending on which mode is picked.
- After that the process for both mode 'm' and 'f' is the same.
- The user is then faced with option 'E', 'D' or 'A' for the corresponding Ceasar cipher mode
- For option E, the metric and graph functions are called immediately to be written to the metrics.txt file as the metrics needed to be taken are what the user has first put into the programme.
- For option D however, metrics and graph information are taken after the message has been decrypted
- For option A the metrics and graph information are called to be written to the metrics.txt file within the auto-decrypt function in Module 2 once a correct answer has been verified.

- For Ceasar cipher options E and D, the user will be asked whether they would like to choose option M to manually input their rotation or option R to generate random number to be their rotation value.
- Whatever the rotation value selected, the message/cipher and the rotation (decrypted message rotation) will be the arguments of the functions encrypt and decrypt, while on auto-decrypt mode there is only one argument and that is the message to be decrypted.
- The programme then analyses whatever information provided and returns a response from the Ceasar Cipher module and then also a rundown of the metrics.
- Metrics are mainly worked out by converting the data to a dictionary and using counters in for loops.

2 Analysis

From the beginning of part 1 I decided that it would make sense to if and elif statements to provided set ranges in the ASCII characters that the programme accepts in the decryption or encryption shift. I did this rather than importing a faster and more efficient module that would simply turn everything into uppercase letters because I feel giving the options of ranges in the ASCII values means that if the code where to be changed or reused to make it possible for any other characters that are not just letters to be shifted, it would be easier the way I have done it. Following on from this, during my development of part 1, I decided that the since my Ceasar Cipher code is longer than it could be, I should make a module that can be called from within the directory which has three functions to Encrypt, Decrypt and Auto-Decrypt. Later in the Assignment this made my code much neater and more user friendly. Initially I had been printing each character (ch) directly from the cipher functions when they were called, however I ran into trouble doing this and 'none' messages started showing up in the console as I believe the programme was double printing. Instead of doing this, I changed the programme so that the string data put in by the user is immediately converted into a for loop and then as each character is shifted and reverted by the function chr() back to letters from the ASCII characters they are added to a new empty list created at the top of the function which essentially, depending on the Ceasar Cipher function, is a list of all the new characters of the string to be returned by the programme. Therefore, the function returns the joined list as the new message. The module where I utilised certain data types to the best of my advantage most was in the MetricFunctions.py module. Before I performed any of the functions required by Parts 2.2, 2.3 and 2.4, I first got rid of all the numbers and punctuation in the string using a string import function to remove punctuation in a more concise way than with pure python and the isdigit() function to remove numbers. Subsequently I then converted the decrypted message from a string into a list of the words to do list comprehension, a set, in order to find the number of unique words and two dictionaries which turned out to be very useful at arranging each word as the key to the dictionary, followed by either the world length or the word frequency as the value in the dictionary. Processing the data into these data types meant that there was no repeated information in the metrics functions and that I could write all the data to the metrics.txt file in one function by calling all the functions that produce the metrics using each data type as an argument for each. In part 5 I polished my use of the two metrics and cipher file modules in the main.py file. I also created a small edit to the auto-decryption programme that prints the value of the rotation that was used to gain the correct message. The main thing that I did in part 5 was create a bar chart with the top 5 most common words against their frequency and I did this using dictionaries, list comprehension and sorting once again. Placing the graph in a function also means that I can decide very easily in the main.py file when I want the graph to be created and from what data. Formatting the graph also allowed me to do some more research into how to change the margins in a graph

and how to use hexadecimal colour codes to change colours on the graph.

3 Conclusion

I believe in the future I will attempt to make a class within the MetricFunctions.py Module to allow for more organised further development of the programme if any further developments were to ever get made. I also think in the future that when I am starting out on an assignment, I will plan my programme much more thoroughly rather than starting straight away with coding. I planned Part 4 on paper beforehand and I believe that it is my most concise and effective piece of code in this programme. I would like to find a way to replace the common word comparison using words.txt in part 4 with a method using distribution of word frequency and distribution of letter frequency so that the programme could be applied to different languages.