Background:

­Objective: To develop an encryption software that takes a text file as an input, and produce a new file that encrypts the data of the input file. Create a decryption algorithm that works alongside the encryption.

Encryption:

Encrypting data is the process of scrambling words and letters into a different form that can be translated back to the original format for the purposes of security. For this software, I aim to use a Caesar Cipher which shifts the index of a character in the alphabet by a certain amount. The new obtained character becomes the encrypted character.

File/Input Data:

User will be able to enter both a text input and a file input, which will only encrypt admissible characters (see Design Elements). Text input via text box and file input via choosing a file from directory.

Output Data:

Data will be outputted either via text box inside the GUI or a file will be created inside a directory. Output data could be either encrypted or decrypted depending on choice of the user.

Design Elements:

Technical Specifications:

Using Python 3.7.7 in Thonny IDE. Using Py2app to create the .exe file (see Deployment). Using Tkinter (Python Library) for Graphical User Interface.

Encryption:

To create something that is harder to break into, I previously used a randomizer to shuffle all processed characters so for example if we have the letter ‘a’, this will not shift to ‘b’ if the shift is 1, it could shift to ‘@’ instead. This makes the encryption more secure but also means if the unique representation of the character list is lost, a brute force algorithm will be needed to find it again – an unfortunate problem but a better encryption.

Decryption:

The Decryption process works the exact same as the encryption but backwards. Will calculate the length and the key to decrypt and then shift the characters back to the original position.

Inputting Data:

The user can input data in 2 ways, text input or text file input. For Text input, a text box appears in front of the user, where the user interacts with this and enters their text and consequently presses a button to submit the text to be encrypted or decrypted. For a text file input, the user can choose from their computer a text file, and this will be processed line by line and encrypted or decrypted. File handling is also present to avoid errors and include exceptions.

Outputting Data:

After the user presses the submit button, two possible events can occur. If text is via input box, an output box will appear that will output the text in a textbox very similar to the text inputted. This text cannot be modified (read only), but can be copied and pasted, for example into the decoder and the text will be decoded. This happens for both the decoder and encoder. For a text file, after a submission and a new file name is entered, a message is given to the user to notify them that a file has been made in the directory specified. This file will contain the encoded/decoded information.

GUI:

User is presented with a graphical user interface when the .exe file is executed. The interface will look as described: The window will be divided into vertical sections. The small left-most section will be a side bar where the user can choose between functionalities of the program and will show the name of the application. The remaining vertical sections will be the interface for the application, this is where the user will be able to interact with the buttons and textboxes to encrypt/decrypt their data.

2 GUIs will be made because of an issue that prevents colors in MacOS, so a new library is imported so the buttons and colors can render properly.

Deployment:

Exporting to a .exe file:

I was able to achieve this relatively easy, however this short-lived because the application would only work locally. Recognized by other computers and systems as a virus, this meant that the file could not be properly deployed in other systems. However, it had worked up to a good enough standard and this could be something that, if worked around, could mean the application can be deployed to other system without being detected as a virus.

Preliminary Testing:

My preliminary testing overviews small bugs I found during development. The nature of the project means I won’t include any post-development editing. Majority of the bugs found were mostly due to GUI elements appearing in faulty places, but this was too specific and frequent to note all of the issues that have been since resolved.

Found a bug where the double layer of Caesar cipher would cross over meaning the encoding was the exact same as the input, but this problem is now fixed by adding a condition to stop this from happening. It happened because of the rare situation where both shifts add to 91 meaning the shift would loop around instead. This means this must be updated in the decoding where the shifts add up to 91, shift2 gets a +1 to essentially shift the characters by 1.

Vulnerabilities and Potential Expansions:

By using an Object-Oriented paradigm, instances of the different modes are created, every time the button is clicked. This is a potential problem the program running for a long time this can consume a lot of memory as the program can build up lots of instances of this object. I found this particularly difficult to resolve because of the way the buttons work in Tkinter. I believe this can be solved using global variables or rather global objects, but I can run into many complications very quickly and very easily so ideally, I want to minimize this at all costs. However, at higher performance programs this might be the better choice to save memory and improve performance at the risk of logical errors.

In terms of deployment, it would have to translated to a native or cross-platform development environment to achieve deployment as the current .exe doesn’t run.

Overview of the Project

Overall, I achieved my background objective and to a good standard with working GUI elements. The scope of the project did not include a full analysis into user feedback, so I was unable to obtain a wide scope of the feel of the project to add small modifications. Nevertheless, the encryption and decryption worked well, the file handling was sufficient so there aren’t many issues with the logic and process of the program. I was disappointed because of the deployment issues of the project, which I believe holds this project back and gives reasonable cause for a constant stream of updates and maintenance. However, looking forward, I learn python is more experimental and better for smaller niche areas of a program rather than using it to build projects from the ground up.