Compte rendu

Choix de programmations :

Mon choix principal à était d'utiliser une fenêtre graphique. Pour ne pas avoir à refaire tous les éléments nécessaires dans un fenêtre graphique avec pygame (gestion des évènements, titre…), j'ai décide de créer une librairie ou je mettrais tous mes codes supplémentaires (pour de multiples programmes) : MLib. Je vous laisse aller voir la documentation présente pour en savoir plus. Pour commencer, je crée tout les éléments graphiques de bases.

Je commence par créer une fenêtre graphique de base « app » ligne 11 avec la taille défini ligne 9. Ensuite, je défini une fenêtre MGui ligne 12 qui permettra de prendre en charge l'application. J'applique à la fenêtre un fond qui est une image gif jolie, qui défile à 48 images par secondes et qui est de même ration que la fenêtre pour qu'elle puisse la couvrir en intégralité. Elle a pour titre « Chiffrage ». Ensuite, je défini un titre ligne 14 pour l'aspect esthétique. Ce titre est de type MTexte et porte l'inscription « Chiffrage ». Elle est configuré pour avoir une petit bordure avec un fond blanc transparent. Je défini ensuite une MEntreeTexte qui correspond au texte a chiffrer ligne 15. Elle est configuré pour recevoir du texte dans une interface. Une interface similaire est créée ligne 22 pour le texte à déchiffrer. Deux boutons sont crées ligne 16 et 17. Ils permettent soit de chiffrer, soit de déchiffrer le texte. Le bouton « chiffrer » transforme le texte de l'entrée de texte à chiffrer en texte chiffrer et le place dans l'entrée de texte à déchiffrer. Le bouton « déchiffrer » fait la même chose mais de manière inversé. À la ligne 18, un MTexte « bordureChoixBouton » est crée. Il devra contenir le nom de l'algorithme de chiffrage qui sera utilisé pour le chiffrage. Pour changer, il faut clické sur les boutons défini à la ligne 19 et 20, qui sont enfants du MTexte, qui sont des flèches. Lors du lancement, l'algorithme est « Rot13 ». À la ligne 21, un MEntreeTexte est défini. Il est placé au dessous du MTexte défini précédemment, et permet de rentrer une clé si l'algorithme choisi en demande une.

Je commence ensuite une variable infini, et actualise ma fenêtre « app » dedans ligne 24 et 25. À la ligne 27, on vérifie si un des boutons pour changer d'algorithme est clické. Si oui, alors l'autre algorithme sera choisis. Pour cela, le texte de « bordureChoixBouton » est changé selon sa valeur actuelle (si elle est de rot13, alors elle sera à Vigenère). Si l'algorithme nécessite une clé, alors elle sera affiché ligne 35 à 38. Dans ce cas, seul Vigenère en nécessite une. À la ligne 40, on test si le bouton de chiffrage est clické. Si oui, si « bordureChoixBouton » à pour texte « Rot13 », alors l'algorithme Rot13 est appliqué au texte ligne 43. Il est gérer dans le module cryptage_ROT13.py fait par Alexis Bergère. Le texte est ensuite appliqué à l'entrée de texte du texte à déchiffrer. Sinon, si l'algorithme choisi est Vigenère, alors le texte va d'abord être converti en majuscule (avec un système pour conserver les places des majuscules originelles). Ensuite, le chiffrage est fait. Il est gérer dans le module cryptage_VIGENERE fait par Illiam Pont Layus. Le résultat est ensuite reconverti en minuscule, sauf pour les endroits qui étaient originalement en majuscule, et appliqués au texte à déchiffrer. Si le bouton de chiffrage n'est pas clické mais que le bouton de déchiffrage est clické, comme dit à la ligne 67, alors le déchiffrage à lieu. Si l'algorithme choisis est Rot13, alors l'algorithme Rot13 à l'inverse est appliqué au texte ligne 70. Il est gérer dans le module

cryptage_ROT13.py fait par Alexis Bergère. Sinon, si l'algorithme choisi est Vigenère, alors le texte va d'abord être converti en majuscule (avec un système pour conserver les places des majuscules originelles). Ensuite, le déchiffrage est fait. Il est gérer dans le module cryptage_VIGENERE fait par Illiam Pont Layus. Le résultat est ensuite reconverti en minuscule, sauf pour les endroits qui étaient originalement en majuscule, et appliqués au texte à déchiffrer. L'écran de l'application est ensuite actualisé ligne 95. J'ai décidé de m'arranger pour que mon code soit adapté au codes de Alexis Bergère et Illiam Pont Layus, et non pas à leur demander de s'adapter à moi. Ce choix est dût à nos différences de niveaux qui me permet de très facilement faire ce genre de choses.

Difficultés rencontrés :

J'ai rencontrés quelques difficultés lors de la création de MLib. Le fait de pouvoir bouger le curseur a été assez difficile à mettre en place. La gestion de la taille maximal des lignes des MTexte était aussi un défi. Appart cela, je n'ai rencontré aucunes difficultés notables.

Améliorations possibles :

Je ne parlerai pas des améliorations que je peut apporter à MLib, car elle se compte en centaines. Cependant, je parlerais de celle applicables à cryptographie.py. L'ajout d'autre système de chiffrage (arborescence de Huffman[même si il ne s'agit pas de chiffrement mais de codage, plus précisément de compression], LZ77[idem pour lui]…). De plus , on peut rajouter une option de chiffrement d'un fichier entier.