Introduction

L'objectif de ce programme est d'implémenter diverses méthodes de cryptographie afin de chiffrer et déchiffrer des messages.

Librairies utilisées

> librairie math permettant d'utiliser la fonction gcd(a, b) renvoyant le pgcd de a et b.

Rôle des fonctions

Afin d'implémenter les diverses méthodes, nous avons définis deux constantes, `ALPHABET` qui vaut toutes les lettres minuscules et majuscules de l'alphabet latin, ainsi que l'espace. En réalité on peut y ajouter n'importe quels caractères supplémentaires sans que ça ne pose soucis. à partir d' `ALPHABET` on définit `LEN_ALPHA` qui n'est autre que la longueur de l'alphabet.

- `find(char: str) -> int`

-> Prend en paramètre un caractère et renvoie la position de celui-ci dans l'alphabet.

- `reverse(msg: str) -> str:`

-> Prend en paramètre un message et renvoie son chiffrement par son écriture à l'envers. Nous avons pris l'initiative d'utiliser le slicing en python nous permettant de ne pas utiliser de boucle.

- `cesar_simple() -> str`

-> Renvoie le message chiffré selon la méthode de César simple. Le programme demande à l'utilisateur en console de saisir un mode, soit le chiffrement soit le déchiffrement. Puis, il va demander une clé de chiffrement respectivement de déchiffrement selon le mode choisi. Ensuite il est demandé de saisir le message sur lequel appliquer le mode. Enfin, pour chaque caractère dans le message, on applique la formule :

1. Pour chiffrer : caractère_chiffré = (position_caractère + clé) mod LEN_ALPHA.
2. Pour déchiffrer : caractère_déchiffré = (position_caractère - clé) mod LEN_ALPHA. On va concaténer le résultat à une chaine à l'origine vide.

- `vigenere() -> str`

-> Renvoie le message chiffré selon la méthode de Vigenère. Similairement à césar simple, on demande le mode, un message ainsi qu'une clé, seulement, la clé n'est pas un entier mais une chaîne de caractères. Ainsi le calcul du message diffère, à chaque caractère dans le message on détermine le décalage en parcourant la clé, on commence avec le premier dont on saisi la position et on chiffre / déchiffre identiquement à césar simple, au caractère suivant dans le message, on décale avec le caractère suivant de la clé. Jusqu'à ce que tous les caractères du message soient traités, si on atteint la fin de la clé, on recommence du début.

- `inverse_modulaire(a: int) -> int`

-> Prend en paramètre un entier a dont est renvoyé son inverse modulaire. On utilise l'algorithme d'Euclide étendu afin de déterminer u où $u \equiv a^{-1} \mod LEN\_ALPHA$.

- `cesar_affine() -> str`

-> Renvoie le message chiffré selon la méthode de César affine. Comme pour les autres chiffrements, on demande le mode et un message. On demande egalement deux clés, une clé a et b, la première a pour condition d'être première avec LEN_ALPHA soit pgcd(a, LEN_ALPHA = 1) qui ici est calculé par la fonction gcd() de la bibliothèque math.

1. Afin de chiffrer : on parcourt le message, à chaque caractère on fait (clé_a × position_caractère + clé_b) mod LEN_ALPHA.
2. Afin de déchiffrer on va notamment utiliser la fonction de calcul d'inverse modulaire : à chaque caractère on fait ($clé\_a^{-1}$ × position_caractère + clé_b) mod LEN_ALPHA.

Programme principal

Le programme principal met en exergue les différentes fonctions et les présente.