Rapport TP°1

Membres de groupe:

- Eldis YMERAJ
- Zied JERBI

Le code Inverse

```
1. message = 'Hello'
```

La ligne 1 stocke la chaîne que nous voulons crypter dans une variable nommée message. Chaque fois que nous voulons crypter ou décrypter une nouvelle chaîne, nous allons simplement taper la chaîne directement dans le code de la ligne 1.

```
2. traduction = ' '
```

La variable traduction est l'endroit où notre programme stockera la chaîne inversée. Au démarrage du programme, il contiendra la chaîne vide.

```
3. i= len(message)-1
```

Ce code n'est qu'une instruction d'affectation qui stocke une valeur dans une variable nommée i. L'expression qui est évaluée et stockée dans la variable est len(message) - 1. La fonction len() accepte un argument de valeur de chaîne et renvoie une valeur entière du nombre de caractères contenus dans la chaîne (c'est-à-dire la longueur de la chaîne). Dans ce cas, nous passons la variable message à len(), donc len(message) nous dira combien de caractères se trouvent dans la valeur de chaîne stockée dans message.

```
4.  while i>=0 :
5.          traaduction=traduction + message[i]
6.           i=i-1
7.
8.  print(traduction)
```

Ce qu'une boucle while indique de faire est d'abord de vérifier à quoi la condition (qui à la ligne 4 est i >= 0) est évaluée. Si la condition est évaluée à Vrai, l'exécution du programme entre dans le bloc suivant l'instruction while. En regardant l'indentation, ce bloc est composé des lignes 5 et 6.

Si la condition de boucle while est évaluée à False, l'exécution du programme ignorera le code à l'intérieur du bloc suivant et passera à la première ligne après le bloc (qui est la ligne 8).

La ligne 5 est une instruction d'affectation qui stocke une valeur dans la variable traduction. La valeur qui est stockée est la valeur actuelle de traduction concaténée avec le caractère à l'index i dans le message. De cette façon, la valeur de chaîne stockée dans la traduction augmente » jusqu'à ce qu'elle devienne la chaîne entièrement cryptée.

La ligne 6 une instruction d'affectation. Il prend la valeur entière actuelle dans i et en soustrait un, puis stocke cette valeur en tant que nouvelle valeur de i.

La ligne 12, on a imprimé le contenu de la variable traduction. Cela montrera à l'utilisateur à quoi ressemble la chaîne inversée.

Le code César simple

```
1.  message = 'hello'
```

La ligne 1 stocke la chaîne que nous voulons crypter dans une variable nommée message.

```
2.  cle=6
```

Dans la ligne 2 on a declaré la clé de cryptage/décryptage.

```
3.  mode='cryptage'
```

La ligne 3 indique au programme de crypter ou de décrypter.

```
4.  alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

On a declaré une variable alphabet qui content tous les symboles possibles qui peuvent être cryptés.

```
5.  traduction=' '
```

La variable traduction stocke la forme chiffrée/déchiffrée du message.

```
6.  lettre=message[1]
```

```
7.  message = message.upper()
```

Fonction .upper() met la chaîne en majuscule dans le message.

```
8.  for lettre in message:
9.      num=alphabet.find(lettre)
10.     if mode=='cryptage':
11.         num=num+cle
12.         lettre=alphabet[num%26]
13.         traduction=traduction+lettre
14.     else:
15.         if num-cle<0:
16.             lettre=alphabet[num-cle+26]
17.     lettre=alphabet[num-cle]
18.     traduction=traduction+lettre
19. print(traduction)
```

La ligne 8: Chaque fois que l'exécution du programme parcourt la boucle (c'est-à-dire à chaque itération de la boucle), la variable lettre prend la valeur du caractère suivant de la chaîne(message).

La ligne 9: Nous voulons vérifier si la lettre est une lettre majuscule car notre programme ne chiffrera (ou déchiffrera) que les lettres majuscules. Tout autre caractère sera ajouté à la chaîne traduite sans être chiffré (ou déchiffré). La variable num prend le nombre de lettres.

La ligne 10: Manitement si le condition est vrai, ca veut dire si on est dans le mode cryptage alors on passe dans la ligne 11, 12, 13.

Lignes 11, 12: la variable num prend la valeur num+ clé, la lettre est stockée dans indexe num%26 d'alphabet.

La ligne 13: Jusqu'a ce moment la variable traduction c'est vide, quand on est entré dans le boucle for et si le condition mode== 'cryptage' est vrai alors traduction prends les valeurs de tous lettres cryptées a pertir de l'indice 0 jusqu'a n-1, et donc dans traduction c'est le message crypté.

La ligne 14: Si le condition if à la ligne 10 n'est pas vrai alors on est dans le mode decryptage.

La ligne 15, 16 : Si la valeur de num-cle est negative alors il faut changer et mettre dans un modulo26 positive, alors la lettre est stocké dans l'indice de (num-cle)+26 de l'alphabet.

La ligne 17, 18: Si num-cle n'est pas une valeur negative alors on continue normalement dans la ligne 17. Dans la tableau lettre enverrai tous les lettres decrypées.

La ligne 18, 19: Notre variebale traduction qui est vide est égal à la somme de lettres décryptés et après on va imprimer le mot crypté/decrypté.

Le code César affine

Rappel: Le code de César affine n'est pas fini.

```
1.  import pyperclip
2.
3.  alphabet='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
    0123456789:,.?!@#$%&*=+-;\/><é^€[]{}~'
4.  message = 'Hello'
5.  a =
6.  b =
7.  a1
8.  a2=
9.  mode ='cryptage'
10. cle =6
11. traduction = ' '
```

Ligne 1-8: On a stocker dans un variable alphabet les chaines des caracteres qu'on va utiliser, notre message est Hello, on a 2 clés dans le mode cryptage: a que est premier et le clé b(y=ax+b). Dans le mode decrypytage y=a1x +b1 avec aa1Ξ1[26] ou a1 est l'inverse de a mod26 et b1Ξ(-a1b)+26. On a declaré une variable vide traduction qu'on va utiliser a la fin de code pour imprimer le message crypté/decrypté.

```
13. def pgcd(a, b):
14.  while a != 0:
15.      a, b = b % a, a
16. return b
```

Lignes 13-16: On a écrit le fonction pgcd pour détécter si les nombres sont premiers entre eux.

```
17. def inverse(a, m):
18.  #Ca va retourner l'inverse de a%m (a*x %m=1)
19.  if pgcd(a, m) !=1:
20.      return None #il n'existe pas mod inverse si a & m ne sont pas premiers entre
eux.
21.
22.  #On calcule en utilisant algorithme d'Euclide etendu
23.  u1, u2, u3 = 1, 0, a
24.  v1, v2, v3 = 0, 1, m
25.  while v3 != 0:
26.      q = u3 // v3 # // c'est l'operateur de division d'entiers
27.      v1, v2, v3, u1, u2, u3 = (u1 - q * v1), (u2 - q * v2), (u3 - q * v3), v1, v2, v3
28.      return u1 % m
```

Lignes 17-28: Ici on a érit l'inverse de deux nombres en utilisant pgcd et algo d'Euclide étendu.