

GOGOLE CRYPT



Spécialité ISN 2016-2017 : Informatique et Sciences du Numérique

Groupe : ABELANSKI Violette ; BUARD Elise ; VOISIN Pauline

Lien : <https://github.com/ecolelasource92/Gogole-crypt>



★ Sommaire :

- 1) Introduction p. 1
- 2) Présentation du projet p. 3
- 3) Mise en place et fonctionnement du projet p. 6
- 4) Démarche Collaborative et difficultés rencontrées p. 16
- 5) Bilan personnel et conclusion p. 18
- 6) Annexe p. 22

◆ I/ INTRODUCTION

Cette année de Terminale Scientifique 2016-2017, nous avons décidé de former un groupe de 3 personnes pour réaliser notre projet d'Informatique et Sciences du Numérique. Il se compose de Violette Abelanski, Elise Buard et de Pauline Voisin. Après concertation, nous avons eu l'idée de réaliser une plateforme de cryptage et de décryptage de messages.

- Contexte : Quelques temps après avoir entendu parler du C.E.S (Consumer Electronic Show) qui s'est tenu à Las Vegas pour sa session de 2017, nous nous sommes rendues compte que nous vivions dans un monde hyperconnecté dans lequel l'innovation, la créativité et les nouvelles technologies sont parties intégrantes. Malheureusement, à l'heure actuelle, les nouvelles technologies sont aussi à l'origine de nombreuses dérives : la surveillance internet en est un exemple frappant. Après un stage dans la police scientifique et des ateliers MPS, nous nous sommes interrogées aux modes de cryptage qui permettent de minimiser la grande surveillance accrue notamment par Google et les réseaux sociaux.
- Enjeu du Projet : À partir de ce constat, il s'agit de mettre en place une interface graphique à la façon de GOOGLE Traduction pour pouvoir traduire le langage courant en langage crypté et indéchiffrable pour une personne lambda. À cela s'ajoute une fonction décryptage qui permet l'action inverse. L'utilisateur rentre le texte qu'il souhaite coder, celui-ci est codé. Même chose pour l'utilisateur qui souhaite déchiffrer un message reçu sous forme de code. Cet interface ne comporte qu'une seule langue de traduction : le français. Notre code ne code que l'alphabet latin et les caractères spéciaux présent sur un clavier azerty et qwerty.
- But : Le but du projet est de créer un programme capable de crypter et de décrypter un message grâce à différentes méthodes de cryptage. Nous voulions utiliser une seule méthode de cryptage, se référant à une seule clé de code.. Nous voulions créer une interface simple et efficace.
- Problématique : Inspirées par les travaux d'Alan Turing, nous voulions nous confronter au monde complexe du codage. Nous voulions nous même expérimenter la création d'un code, si possible indéchiffrable, et voir ses possibles limites. Nous voulions comparer notre programme de codage aux méthodes déjà existantes. Nous nous sommes inspirées du code de César, du code de Vigenère, du chiffage par alphabet décalé.

❖ II/ Presentation du projet

Notre projet se nomme "GOOGLE CRYPT" en clin d'oeil à Google Traduction.

- Les Moyens mis en place :

- nos 3 ordinateurs portables
- l'espace de travail collaboratif GITHUB pour mettre en commun nos codes
- les modules Python Tutor, TextWrangler, Sublime Text et NotePad pour coder et créer l'interface graphique
- Google Drive pour échanger nos idées et avancer plus rapidement dans le projet
- Nous codons en PYTHON exclusivement, après avoir testé puis abandonné le code HTML

- Objectif initial :

- Pour que le message soit indécodable, l'idée est de changer de méthode de cryptage toutes les 5 lettres (puis cela sera 8 lettres finalement).
- chaque méthode de cryptage code pour deux caractères à la suite
- au début de notre projet nous avions des ébauches d'idées mais nous ne voulions codes que sur la base d'une seule méthode de cryptage en changeant toutes les lettres de conditions :
 - remplacer une lettre par la suivante dans l'alphabet
 - remplacer une lettre par 5 lettres qui la suivent dans l'alphabet
 - remplacer une lettre par son numéro $x2 + 3$
 - si le numéro de la lettre est inférieure ou égale à 13, il remplacer par 3 lettres avant, sinon par 7 lettres après dans l'alphabet.
 - si le numéro de la lettre est paire, remplacer la lettre par son numéro divisé par 2. Sinon la remplacer par son numéro $x3$
 - Chaque caractère espace sera codée par le caractère suivant l'espace dans le texte +1

Nous voulions nous inspirer de cette interface graphique de codage-décodage trouvée sur GITHUB :

Codes secrets (github, fiche signalétique)

Chiffre par subsitution avec alphabet décalé

Options

Que dois-je faire?

Texte avant transformation :

Clef : 3

Alphabet original : abcdefghijklmnopqrstuvwxyz

Alphabet décalé : xyzabcdefghijklmnopqrstuvw

Chiffrer

Déchiffrer

Echanger les textes

Texte après transformation :

Attaque par force brute :

Lancer

Arrêter

<

>

Attaque à texte clair connu :

Chercher

Chiffre par subsitution avec alphabet permuté

Options

Que dois-je faire?

Texte avant transformation :

Clef : 314

Alphabet original : abcdefghijklmnopqrstuvwxyz

Alphabet permuté : upqgshvlkmrtifyowxceznjabd

Chiffrer

Déchiffrer

Echanger les textes

◆ III/ Fonctionnement du Projet :

Nous voulions que notre programme de cryptage et décryptage soit à la fois simple et technique, en utilisant le plus possible les techniques et fonctions que nous avons appris depuis ce début d'année.

Après plusieurs séances, pour rendre notre travail plus professionnel et plus clair, nous avons décidé de mettre nos différentes méthodes de codage sous la forme de fonctions comme sur l'image ci-dessous :

```
def asci_lettres() :
```

Ainsi, il devenait plus simple de les “appeler” directement dans notre programme principal.

On commence tout d’abord à rentrer nos différentes variables et données de base :

```
texte = input()

nbtexte = len(texte)

alphabet = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U",
            "V", "W", "X", "Y", "Z", " ", ".", ",", ":", "!", "?", ";", "#", "(", ")", "'", "\"", "\\", "-", "1", "2", "3",
            "4", "5", "6", "7", "8", "9", "0", "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N",
            "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z"]

n = 0
cle = "CODE" # modifiable
nbcle = len(cle)
quotient = 2 // nbcle # penser a changer le 2 quand on veut faire prendre + de 2 lettres en vigenere = pour le reste
reste = 2 % nbcle
```

Le texte est à remplir par l'utilisateur. Cela est marqué par le terme “**input ()**”.

On rentre également un tableau qui est un alphabet un peu spécifique puisqu'il contient des lettres, des chiffres et des caractères spéciaux.

On rentre déjà les variables “**quotient**” et “**reste**” qui seront utiles pour la méthode de cryptage utilisant le code de Vigenère.

La variable **n**, représente une sorte de curseur pour déterminer quel caractère doit être codé pour quelle méthode, et va de deux lettres en deux lettres.

On incrémente **n** à 0, ce qui sera utile dans notre boucle “**While**”.

On rentre également les variables **nbtexte**, **clé** et **nbcle**.

On peut alors passer au traitement, c’est à dire aux fonctions et programmes de codage.

● Les Méthodes de cryptage :

1) ASCII lettre + 1 :

```
def asci_lettres() :  
    for loop in range(2) :  
        newCaractere = chr(ord(lettres[loop]) + 1)  
        print(newCaractere, end="")
```

Par cette fonction, on souhaite transformer les deux premiers caractères du texte en nouvelles lettres ou caractères en passant par le tableau ASCII.

À-travers la boucle non conditionnelle “**for loop in range**”, on transforme le caractère. On emploie les fonctions “**ord**” et “**chr**” pour transformer le caractère en son chiffre complémentaire dans le tableau ASCII on fait + 1 puis on le convertit en lettre en se reportant toujours au tableau ASCII. Enfin, on choisit d’afficher le nouveau caractère sans retourner à la ligne avec la fonction “**end = “”**”.

2) ASCII lettre + 8 :

```
def asci_diminu() :  
    for loop in range(2):  
        newCaractere = chr(ord(lettres[loop]) + 8)  
        print(newCaractere, end="")
```

Même chose pour ce programme qui code 2 nouveaux caractères avec un décalage de 8 rangs en plus (+8) dans le tableau ASCII.

3) Notre Alphabet :

```
def notre_alphabet() :  
    for loop in range(2):  
        for eachPos in range(50):  
            if lettres[loop].upper() == alphabet[eachPos]:  
                print(alphabet[eachPos + 5], end="")
```

Ici, on a employé deux boucles “**for loop in range**” : une pour coder 2 caractères et une pour parcourir notre tableau alphabétique spécial que nous avons définie au début de notre programme sous forme de tableau.

La fonction “**.upper**” sert à transformer un caractère minuscule en majuscule.

Cela à condition que le caractère soit similaire à un des caractères de notre tableau grâce à la fonction si “if”, alors on peut afficher le nouveau caractère, crypté par une lettre de 5 rang supérieur à celle initiale.

4) Le code de Vigenère

```
def vige_nere() :  
  
    for loopi in range(reste):  
  
        for eatchpos in range(50):  
  
            if cle[loopi] == alphabet[eatchpos]:  
                nvlcle = eatchpos  
  
                for eachPos in range(50):  
  
                    if lettres[loopi + nbcle * quotient].upper() == alphabet[eachPos]:  
                        print(alphabet[eachPos + nvlcle], end="")
```

Nous avons employé une méthode de cryptage déjà connue, celle du Code de Vigenère. La clé de notre méthode de cryptage de vigenère est le mot “CODE”.

Cette clé va venir additionner sa place dans notre alphabet à celle de chaque lettre de notre texte que l’on transforme. Pour cela nous utiliserons successivement deux boucles la première pour faire tourner la clé et la faire reconnaître dans notre alphabet définit sous forme de tableau au début de notre programme. Puis la deuxième boucle pour chercher dans notre alphabet la lettre que l’on souhaite transformer + la lettre de la clé fois le quotient. On additionne donc après le lettre de notre texte que l’on souhaite coder à celle de la clé lui correspondant.

5) Pour un texte impair :

```
def texte_impair() :  
    dernierelettre = chr(ord(texte[nbtexte - 1]) + 6)  
    print(dernierelettre, end="")
```

Pour régler un des problèmes développé dans la partie personnelle d’Elise , nous avons créé cette méthode de cryptage en plus. Elle intervient lorsque le texte à coder est impair. Ainsi le dernier caractère par cette méthode spécifique va être codé et empêche que l’une des méthodes tourne dans le vide. Même démarche que précédemment. Le caractère codé sort en un nouveau caractère de 6 rangs supérieur. Quand le texte est pair il fait les méthodes de cryptage vu au dessus sans passer par cette fonction de décryptage.

6) Le Programme principal :

```
while n < (nbtexte - 1):  
  
    lettres = texte[n:n + 2]  
  
    if n % 8 == 0 or n % 8 == 1:  
        |   asci_lettres()  
  
    elif n % 8 == 2 or n % 8 == 3:  
        |   asci_diminu()  
  
    elif n % 8 == 4 or n % 8 == 5:  
        |   notre_alphabet()  
  
    elif n % 8 == 6 or n % 8 == 7:  
        |   vige_nere()  
  
    n = n + 2  
  
if nbtexte % 2 == 1:  
    texte_impaire()
```

Nous avons décidé de mettre notre programme principal sous forme de boucle conditionnelle “while” et de boucles “if”. Ainsi, tant que la variable *n* qui nous permet de crypter un à un les caractères de notre texte est inférieure à la longueur du texte (- 1) car le premier caractère est compté en tant que 0), chaque méthode de cryptage va coder deux caractères côte à côte ; grâce à “ *lettres = texte[n:n+2]* ” qui lui commande d’aller de deux en deux à chaque tour. De plus, en faisant la division euclidienne de *n* par 8, nous pouvons appeler les différentes méthodes de cryptage suivant la valeur de *n*. En effet *n* prend + 2 a chaque fin de méthode de cryptage, de cette façon le code passe aux deux lettres suivantes du texte à coder et ne code pas avec chaque méthode de cryptage les deux mêmes lettres.

De même, si le reste de la division euclidienne de la longueur du texte par 2 vaut 1, cela signifie que le texte est impair. On appelle donc la fonction “texte impair”.

● Les méthodes de décryptage :

1) ASCII lettres - 1 et ASCII lettres - 8 :

```
def ascii_lettres() :  
  
    for loop in range(2) :  
  
        newCaractere = chr(ord(lettres[loop]) - 1)  
        print(newCaractere, end="")
```

```
def ascii_diminue() :  
  
    for loop in range(2):  
  
        newCaractere = chr(ord(lettres[loop]) - 8)  
        print(newCaractere, end="")
```

Les deux premières méthodes de décryptages sont sur le même principe que le cryptage sauf que cette fois-ci nous allons chercher à retrouver le texte original à partir du texte crypté. On fait donc l'inverse des opérations du cryptage ; en l'occurrence quand nous avons un + 1 en cryptage nous faisons un -1 en décryptage. De même pour la seconde méthode de cryptage utilisant un - 8 car nous avons fait un + 8 en cryptage.

2) Notre Alphabet et Vigenère :

```
def notre_alphabet() :  
  
    for loop in range(2):  
  
        for eachPos in range(50):  
  
            if lettres[loop].upper() == alphabet[eachPos]:  
                print(alphabet[eachPos - 5].lower(), end="")
```

```
def vige_nere() :  
  
    for loopi in range(reste):  
        for eatchpos in range(50):  
            if cle[loopi] == alphabet[eatchpos]:  
                nvlcle = eatchpos  
                for eachPos in range(50):  
                    if lettres[loopi - nbcle * quotient] == alphabet[eachPos]:  
                        print(alphabet[eachPos - nvlcle].lower(), end="")
```

Même chose pour notre alphabet. Quand nous arrivions à identifier le caractère à coder grâce à notre alphabet situé en haut du code, nous faisons + 5 donc maintenant pour le décrypter il faut faire - 5. Il en va de même avec Vigenère ou il fallait faire [eachPos + nvlcle], il faut maintenant faire [eachPos - nvlcle].

3) Pour un texte impair :

```
def texte_impair() :  
  
    dernierelettre = chr(ord(texte[nbtexte - 1]) - 6)  
    print(dernierelettre.lower(), end="")
```

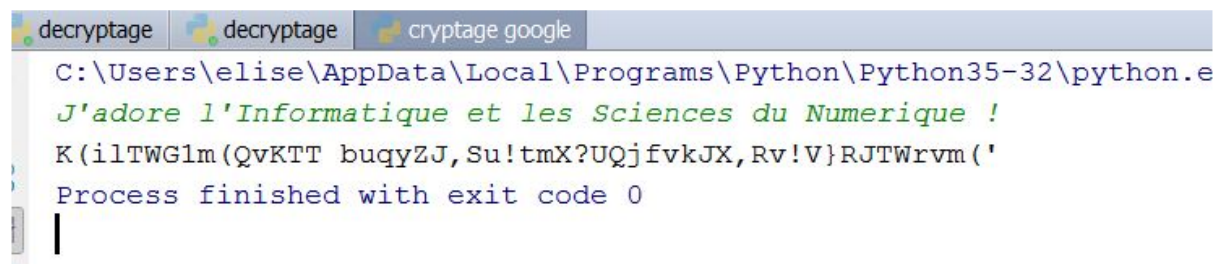
Au lieu d'ajouter 6 rangs au caractère, il faut maintenant lui enlever 6 rangs.

4) Programme principal :

Le programme principal lui ne change pas il va donc appeler les différentes fonctions correspondants aux méthodes de décryptage.

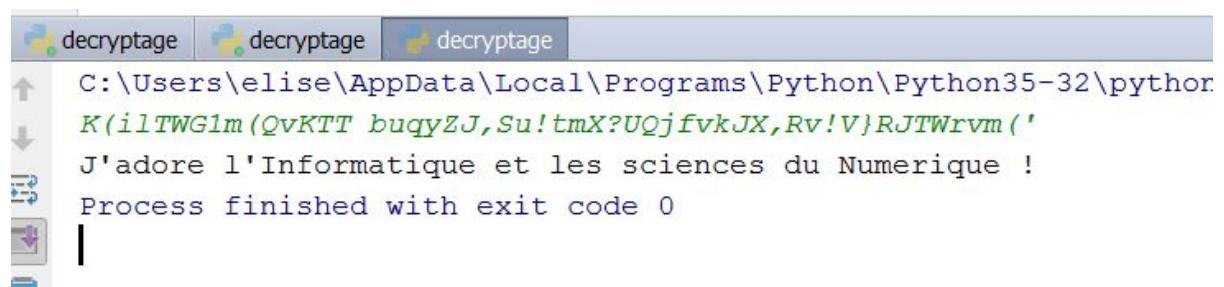
5) Rendu final sous forme de code brut :

Voici le cryptage :



```
decryptage decryptage cryptage google  
C:\Users\elise\AppData\Local\Programs\Python\Python35-32\python.e  
J'adore l'Informatique et les Sciences du Numerique !  
K(ilTWGlm(QvKTT buqyZJ,Su!tmX?UQjfvkJX,Rv!V}RJTWrvm(''  
Process finished with exit code 0
```

Et voici le décryptage :



```
decryptage decryptage decryptage  
C:\Users\elise\AppData\Local\Programs\Python\Python35-32\pythor  
K(ilTWGlm(QvKTT buqyZJ,Su!tmX?UQjfvkJX,Rv!V}RJTWrvm(''  
J'adore l'Informatique et les sciences du Numerique !  
Process finished with exit code 0
```

Interface graphique

Notre interface graphique, codée en python grâce à la bibliothèque **TKinter**, se divise en deux parties : la parties physique (la forme), et la partie fonctionnelle de codage (le fond).

- La forme

Tout commence par la mise en place d'une fenêtre, qui s'affiche après exécution du code de l'interface.

```
from tkinter import *
fenetre = Tk()
fenetre.geometry("1000x700+0+0")
fenetre.title("Gogole crypt")
fenetre["bg"] = "turquoise3"
```

On débute par importer tkinter. Par la suite, on crée une fenêtre, en choisissant ses dimensions (1000x700) ainsi que l'endroit où elle s'ouvrira sur l'écran (0+0). Le titre de cette fenêtre sera "Gogole crypt", et l'arrière plan sera turquoise.

Par la suite, nous créons les différents textes, entrées et boutons qui apparaîtront sur cette fenêtre.

```
texte1 = Label(fenetre, text="Gogole crypt").pack()
texte2 = Label(fenetre, text="Testez notre nouvelle methode de cryptage revolutionnaire et presque indecryptable...").pack()
texte3 = Label(fenetre, text="Texte à crypter :").place(x="400", y="100")
entree1 = Entry(fenetre, textvariable=texteacrypter).place(x="400", y="150")
label = Label(fenetre)
label["bg"] = "turquoise3"
label["fg"] = "black"
label.place(x="400", y="250")
bouton1 = Button(fenetre, command=crypte)
bouton1["text"] = "Crypter"
bouton1.place(x="400", y="200")
texte4 = Label(fenetre, text="Texte à décrypter :").place(x="400", y="400")
entree2 = Entry(fenetre, textvariable=texteadecrypter).place(x="400", y="450")
bouton2 = Button(fenetre, command=decrypte)
bouton2["text"] = "Décrypter"
bouton2.place(x="400", y="500")
lebel = Label(fenetre)
lebel["bg"] = "turquoise3"
lebel["fg"] = "black"
lebel.place(x="400", y="550")
texte5 = Label(fenetre, text="Attention, les accents ne sont pas lus par ce programme").pack()
```

Nous créons d'abord un premier label (texte1) pour le titre "Gogole crypt". Avec le mot "pack", ce titre sera centré et disposé en haut de la fenêtre.

On crée un deuxième label (texte2) présentant notre méthode de cryptage.

Par la suite, nous avons créé l'emplacement d'entrée du texte à crypter. Les indications x et y donnent l'emplacement de ce label sur la fenêtre.

En dessous du texte "texte à crypter" apparaît une zone d'entrée appelée entrée1. Le texte que l'utilisateur entrera ici s'appellera alors "texteacrypter". On définit "texteacrypter" grâce à "stringvar" :

```
texteacrypter = StringVar()
```

Le but sera alors de saisir cette entrée "texteacrypter" afin de l'utiliser dans notre code en tant qu' input pour le crypter. Pour cela, nous créons un label à fond turquoise et à écriture blanche qui n'apparaîtra qu'après exécution du code.

Le code s'exécute lorsqu'on appuie sur le bouton 1 qui a pour fonction "crypt", c'est à dire la fonction de cryptage, et sur lequel apparaît un texte : "crypter".

Après cela, nous créons l'emplacement d'entrée du texte à décrypter. Nous commençons par une zone de texte (texte4) "texte à décrypter", puis une entrée (entrée2). L'entrée 2 aura pour nom "texteadecrypter". On définit "texteadecrypter" grâce à "stringvar":

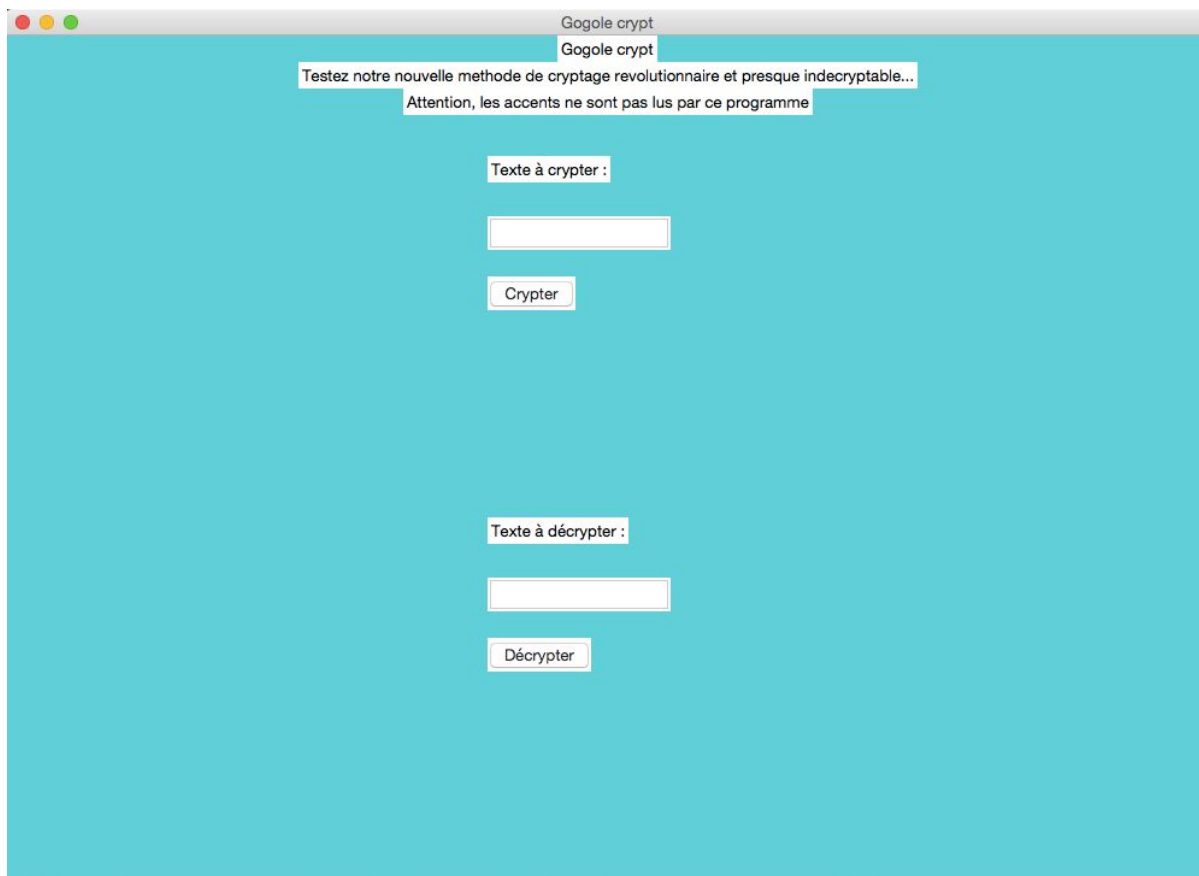
```
texteadecrypter = StringVar()
```

On crée alors un second bouton (bouton2) ayant pour fonction "decrypte", c'est à dire la fonction de décryptage.

Un second label est alors créé, appelé "lebel", qui lui aussi ne s'affichera qu'après exécution du code.

La dernière zone de texte (texte5) est simplement une indication faisant remarquer que les accents ne sont pas lus par le code. Celle ci sera centrée et placée en haut de la fenêtre.

Voici le rendu final de la forme de l'interface :



L'étape suivante de l'interface est donc d'y ajouter le fond.

- Le fond

La partie fonctionnelle de l'interface est en fait notre code, expliqué plus haut, avec cependant quelques modifications. Le code se trouve dans une fonction appelée "crypt" pour le cryptage et "decrypte" pour le décryptage.

```
def crypte() : def decrypte():
```

Le code en lui-même nécessite des modifications. Il faut d'abord modifier le nom de l'entrée faite par l'utilisateur. On ne parlera plus de "input()" mais de "texteacrypter.get()" ou de "texteadecrypter.get()" afin que le code utilise les entrées présentes sur la fenêtre. Afin que la sortie de notre code s'affiche dans les différents label nommés "label" et "lebel", nous avons créé un tableau, que l'on a nommé "tableau" :

```
tableau = [0]*nbtexte
for i in range(nbtexte) :
    tableau[i] = texte[i]
```

Ce tableau contiendra le mot ou la phrase entrée par l'utilisateur, avec un caractère par case. Par la suite, à la fin de chaque fonction de cryptage ou de décryptage, on modifie la lettre initiale par la nouvelle lettre codée, puis on avance d'une case pour que la prochaine lettre modifiée soit la suivante. Pour cela on crée une variable nommée "a", qui prend a+1 à chaque fois qu'une lettre a été cryptée.

```
tableau[a] = newCaractere
if a < (nbtexte-1) :
    a = a + 1
```

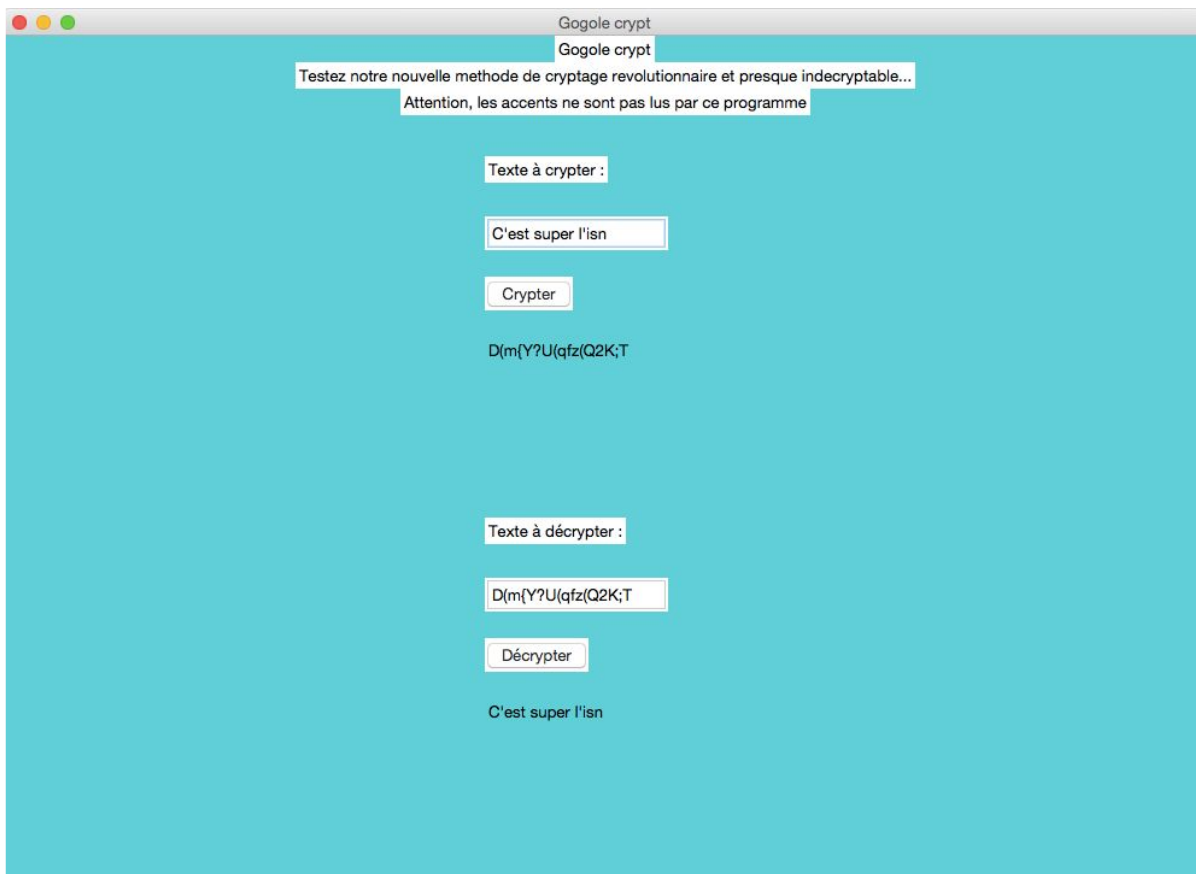
De manière à ce que l'addition a+1 ne continue pas lorsque le mot est terminé, nous avons créé une condition : cette addition se fait seulement si "a" est plus petit que le nombre de caractère entré - 1 (car le premier caractère vaut 0).

À la fin de code, il nous suffira alors de regrouper les éléments du tableau et de les faire apparaître à l'emplacement "label" pour le cryptage et "lebel" pour le décryptage.

```
mot = "".join(tableau)
label["text"] = mot

mot = "".join(tableau)
lebel["text"] = mot
```

Voici la preuve du bon fonctionnement de l'interface graphique : Nous avons testé la phrase "C'est super l'isn".



❖ IV/ Repartition des Taches

Tâche	Nom
Création d'un cahier des charges sur Github : repository. Y ajouter notre cahier des charges déjà rédigé ultérieurement.	Tout le monde
Faire des recherches développées sur les différents types de cryptage.	Pauline
S'informer sur la création d'un site internet et ses difficultés.	Elise
Recherche sur le tableau ASCII	Violette
Création du code ASCII + 1 lettre	Elise
Création du code ASCII + 8 lettres	Pauline
Création du code Notre Alphabet + 5	Violette
Création du Code de Vigenère	Elise
Lier les codes en un seul	Elise
Decryptage	Elise et Pauline
Interface Graphique en Python	Violette
Légender le cryptage / décryptage sur Github	Pauline

- Difficultés et changements dans notre programme de cryptage et décryptage :

Nous avons choisi d'évoquer dans notre partie personnelle chacune les problèmes que nous avons rencontrés au fur et à mesure du code. Dans chaque parties personnelles chacune dira donc à quelles ont été les problèmes qu'elle a rencontrer.

- **Aller plus loin dans la démarche : ce que nous aurions pu améliorer**

Si nous avions eu plus de temps, c'est à dire si l'échéance de début juin avait été repoussée, nous aurions pu améliorer certains problèmes et aller encore plus loin dans notre démarche de cryptage et de décryptage :

- améliorer les méthodes de cryptage : aujourd'hui, bien que nous avons utilisé différentes méthodes de cryptages pour rendre le décryptage plus complexe, nous sommes arrêtées à des décalages de caractères. Nous aurions pu faire des permutations / substitutions de caractères par exemples.
- Aujourd'hui il est de plus en plus facile de "craquer" un code. Avec différents essais par ordinateur et par la méthode de la fréquence d'apparition des caractères, il est possible de décrypter notre code. Il ne semble malheureusement pas infaillible.
- Nous aurions pu vaincre les difficultés des caractères spéciaux (accents) pour que notre programmes soit encore meilleur.
- Nous aurions pu tenter de coder d'autres méthodes de cryptage déjà existantes : le chiffre de César, Enigma, le chiffre du CHE...etc
- Nous aurions pu ajouter d'autres caractères spéciaux pour crypter d'autres langues comme l'arabe ou le chinois.

Conclusion : Notre travail de groupe a été une réussite, nous nous sommes réparties équitablement les tâches, et ce travail a été effectué avec efficacité et dans la bonne humeur. Chacune de nous trois a su trouver ce dans quoi elle était meilleure et ce qu'elle a pu apporter au projet. Le fait de travailler en équipe nous a motivé et nous a fait découvrir de nouvelle façon de travailler. En effet il n'est pas facile de travailler en groupe chacun chez soi quand il s'agit d'informatique mais grâce au drive de google cela a été plus facile.

Lien contenant notre projet, celui-ci comporte une branche contenant le cryptage brut, une branche contenant le décryptage brut, et une branche contenant le code de l'interface.

→ <https://github.com/ecoledasource92/Gogole-crypt>

Synthèse Personnelle ISN : Pauline Voisin

★ Réalisation Personnelle :

Dans un groupe de 3, il est important de bien se répartir le travail. Cependant, comme nous débutions toutes les 3, il était donc impératif que nous nous entraisions à chaque difficulté, pour ne pas rester bloqué très longtemps et avancer rapidement.

Mon rôle a alors été dans un premier temps de faire des recherches sur les différentes méthodes de cryptages déjà existantes. Par exemple, une des plus vieilles méthodes de cryptage, la technique assyrienne, transmet des messages confidentiels à l'aide de bâton de même diamètre ou des bandes étaient enroulées pour inscrire un message spécifique.

Plus récemment, il existe 4 types de cryptage :

- **substitution mono-alphabétique** : par exemple le chiffre de César consiste à faire une permutation circulaire des lettres de l'alphabet en remplaçant chacune par celle placée en un même rang à déterminer soit en avant soit en arrière.
- **substitution polyalphabétique** : utilise une suite de chiffres monoalphabétique de manière répétitive (une clé, un peu comme le code de Vigenère)
- **substitution homophonique** : consiste à faire correspondre à chaque lettre du message un ensemble d'autres caractères.
- **substitution de polygrammes** : substituer un groupe de caractères dans le message par un même groupe de caractères.

Ce rôle a permis au groupe de creuser notre idée de base et d'en sortir avec des propositions de différentes méthodes de cryptage, celles qui nous semblaient les plus pertinentes et les plus faisables à coder.

Réalisation du Code Ascii + 8 caractères :

À-travers cette fonction, nous voulions crypter les deux caractères suivant les deux premiers du texte venant d'être cryptés par une première fonction. Pour ce faire, nous avons encore une fois utiliser une méthode de substitution additive. En effet, 2 nouveaux caractères doivent devenir deux nouvelles lettres ou caractères propres au tableau ASCII. Pour se faire, j'ai choisi d'utiliser dans ma fonction une boucle définie "*for loop in range (2)*" qui permet de répéter la méthode deux fois pour que deux caractères soient cryptés.

Nous ne savions pas comment faire reconnaître à notre fonction le caractère dans le tableau ASCII. Après quelques recherches sur le web, nous avons vu que les fonctions "*ord*" et "*chr*" permettent de reconnaître le caractère codé pour le transformer en son chiffre ASCII complémentaire puis en la lettre ou caractère complémentaire (donc celui tapé à l'origine qui est compris et reconnu). Puis, dès que le caractère est reconnu dans le tableau, nous faisons "+8" pour sortir un nouveau caractère de 8 rang supérieur toujours dans le tableau ASCII. Enfin, la fonction "print" ainsi que "*end =*" permettent de sortir le caractère crypté,

sans revenir à la ligne pour que cela forme un mot ou une phrase naturellement. Nous avons changé plusieurs fois le rang car finalement les soustractions rendaient le cryptage de l'interface non fonctionnelle.

```
def asci_diminu() :  
  
    for loop in range(2):  
  
        newCaractere = chr(ord(lettres[loop]) + 8)  
        print(newCaractere, end="")
```

Le Décryptage :

Pour faire le décryptage, nous avons pendant les vacances, avec Elise, et pendant que Violette était occupée sur la plateforme, coder le décryptage. Pour ce faire, il fallait faire l'inverse du cryptage. À chaque addition de rang, il fallait maintenant faire une soustraction.

```
def asci_lettres() :  
  
    for loop in range(2):  
  
        newCaractere = chr(ord(lettres[loop]) - 8)  
  
        print(newCaractere, end="")  
  
def vige_nere() :  
  
    for loopi in range(reste):  
        for eatchpos in range(50):  
            if cle[loopi] == alphabet[eatchpos]:  
                nvlc = eatchpos  
                for eachPos in range(50):  
                    if lettres[loopi - nbcle * quotient] == alphabet[eachPos]:  
                        print(alphabet[eachPos - nvlc].lower(), end="")
```

Légender le code sur Github :

Pendant que Violette finissaient les quelques imprécisions de l'interface et Elise s'occupaient de lier tous les codes et les réarranger pour que ce dernier fonctionne, je me suis occupée de légender notre code entier sur la plateforme GitHub pour le rendre plus lisible. Il fallait que les explications soient les plus claires possibles. Cela m'a permis d'employer et de me familiariser avec un lexique informatique spécifique.

pour le cryptage :

```
#Cryptage
texte = input() #utilisateur doit rentrer le texte qu'il souhaite coder
nbtexte = len(texte) # fonction qui permet de retourner la longueur de la chaine de caracteres

n = 0 # variable-curseur qui selon son reste par la division euclidienne permet de definir la place caractere a coder dans texte
cle = "CODE" # utile pour le programme de Vigenere
nbcle = len(cle) # retourner la longueur de la chaine de caracteres qui compose la cle
quotient = 2 // nbcle # penser a changer le 2 quand on veut faire prendre + de 2 lettres en vigenere = pour le reste
reste = 2 % nbcle

def asci_diminu() : #fonction qui transforme deux caractères côte à côte en deux lettres contenues dans le tableau ASCII

    for loop in range(2) : #boucle qui tourne deux fois pour coder les deux caractères côte à côte

        newCaractere = chr(ord(lettres[loop]) + 1)
        # ord = fonction qui transforme la lettre associée au caractère à son chiffre complémentaire dans ASCII
        # chr = fonction qui transforme chaque caractère en associant le chiffre du tableau ASCII en sa lettre associée
        # +1 = avancer de 1 rang pour retrouver le rang intial et donc pour sortir le caractère intial

        print(newCaractere, end="") # sortir les 2 nouveaux caractères codés sur une même ligne
```

pour le décryptage :

```
for loopi in range(reste):
    for eachpos in range(50): #chercher la lettre de la clé dans notre alphabet spécifique
        if cle[loopi] == alphabet[eachpos]: # attribuer la lettre de la clé demandé a son positionnement dans le tableau
            nvlcle = eachpos
            for eachPos in range(50):
                if lettres[loopi - nbcle * quotient] == alphabet[eachPos]:
                    # faire la position de la lettre a crypter - le nombre de lettre de la clé * quotient
                    # pour être sur que ce caractere existe dans notre alphabet spécifique
                    print(alphabet[eachPos - nvlcle].lower(), end="") #afficher les 2 nouveaux caractères

def notre_alphabet() : # fontion qui se réfère au tableau que nous avons crée, qui prend en compte les caractères spéciaux

    for loop in range(2): #boucle qui tourne deux fois pour coder les deux caractères côte à côte

        for eachPos in range(50): # boucle qui tourne entierement dans notre tableau unique de 50 caractères

            if lettres[loop].upper() == alphabet[eachPos]:
                # "if" est une condition
                # si le caractère dans le texte qui est sous forme majuscule
                # est identique a un des 50 caracteres repertoriés dans notre tableau

                print(alphabet[eachPos - 5].lower(), end="")
                # afficher 2 nouveaux caractères de 5 rangs inférieurs
                # -5 = reculer de 5 rangs pour retrouver le rang intial et donc pour sortir le caractère intial
```

★ Intégration et Validation :

la réalisation finale :

Nous avons eu quelques difficultés à faire fonctionner notre programme final. Le programme de l'interface, non identique au code de cryptage / décryptage, ne fonctionnait pas entièrement. On avait des problèmes avec les espaces, les mots trop longs, les chiffres... Même chose avec les modules bibliothèque comme Django, qui était trop compliqué d'utilisation. Nous avons préféré utiliser Tkinter.

lien avec le travail de l'équipe :

J'ai pu insérer mon travail de cryptage et de décryptage au code commun. Comme notre travail restait beaucoup en groupe, je pouvais aider à débloquer, ou à résoudre dans certains problèmes que ce soit avec notre code ou parfois avec l'interface graphique. Comme ça nous pouvions échanger plus facilement et avancer à un bon rythme.

Problème des caractères spéciaux : En créant notre propre alphabet nous avons pu y rentrer l'alphabet et les caractères spéciaux que nous voulions, ainsi nous pouvons coder des tirets, les virgules, les slashes, etc... Mais nous avons décidé de supprimer les lettres ayant des accents de notre alphabet car pour une raison que nous n'expliquions pas le code refuse de les voir et de les prendre en compte. Les méthodes "notre alphabet" et "Vigenère" ne marchaient pas. Il est donc affiché sur notre interface graphique qu'il est interdit de rentrer des mots comportant des accents, et qu'il faut les enlever au préalable. C'est un problème que nous aurions pu résoudre si nous avions plus de connaissances et plus de temps. Nous avons également éprouvé quelques difficultés de dernière minute. L'interface graphique ne lisait pas correctement les soustractions de rangs dans notre méthode de cryptage. Nous avons donc remplacé les soustractions par des additions.

★ Bilan et Perspective :

comment s'est déroulé le travail en groupe :

Le travail de groupe s'est très bien passé. C'était un exercice qui rappelait le TPE. Rien que pour le sujet en lui-même nous avons mis très peu de temps à nous décider et être motivées pour le créer. Comme nous débutions l'informatique, l'exercice de chercher les réponses directement par soi-même sur internet était très instructif. C'était une pédagogie d'autonomie. Nous avons beaucoup fonctionné en groupe. Dès qu'une avait des difficultés, nous nous aidions mutuellement. J'ai parfois eu du mal à choisir quelle boucle choisir, ou quels paramètres modifier. Mais je savais que je pouvais demander à Violette ou Elise. Nous avançons bien pendant les vacances et en cours car nous savons ce que nous voulions avoir. Très peu de choses se sont faites seules, puisque la plupart du temps notre code avançait pendant les deux heures de cours. Les quelques obstacles que nous avons rencontrés nous ont permis de rebondir, que ce soit en résolvant le problème ou en modifiant certains paramètres. Si le résultat n'est pas parfois, je suis très fière de ce que nous avons réussi à accomplir en moins d'une année. Nous étions toutes les 3 très proches, amies dans la vraie vie et sur une même longueur d'onde.

Résumé année de ISN :

À-travers cette année d'ISN, j'ai découvert un langage qui m'intriguait et que je ne connaissais absolument pas : le langage informatique. Je voulais découvrir cette matière car je savais que dans notre monde mondialisé et numérique en évolution constante, mes connaissances en informatique me seraient précieuses dans mes études, mes futures professions et pour ma culture générale également. Je savais que l'informatique et le digital seraient utiles à appliquer aux sciences sociales. Il y a eu une très bonne ambiance dans la classe. Les niveaux étaient très disparates, entre débutants et experts, mais chacun avait la volonté, dont moi, de progresser au maximum. De plus, il y avait une entraide constante entre nous, en cours et chez nous. Être en autonomie et chercher les réponses par nous même étaient un exercice particulier, parfois déroutant mais généralement très enrichissant. Cela nécessitait une grande maturité. Cela a également aiguisé ma curiosité, mon esprit de découverte et ma motivation pour cette matière. J'ai eu du plaisir à partager mon projet avec Elise et Violette car nous nous sommes très bien entendues toutes les trois.

Je ressors de cette année avec beaucoup de connaissances qui me serviront plus tard j'en suis sûre. Mais je pense que l'informatique n'est pas pour moi, cette trop grande autonomie n'était pas en accord avec ma manière de travailler.

Annexe :

Lien : <https://github.com/ecolelasource92/Gogole-crypt>