Activité 7 : Le code de César

Depuis la nuit des temps, que ce soit en temps de guerre, lors de tractations politiques ou dans le monde des affaires, des hommes et des femmes ont tenté de garder pour eux certaines informations ayant le potentiel de leur donner un avantage.
Plus récemment, les révélations d'Edward Snowden ont aussi rappelé l'importance pour tous de protéger ses données personnelles.
Dans cette activité, nous allons introduire les principes de base de cryptographie, à partir d'exemples concrets.

1. Décoder les secrets de Jules César

La cryptographie ne date pas d'hier. Jules César, le célèbre empereur romain, rendait illisibles ses messages au cas où ils tomberaient entre les mains de ses ennemis. Sa technique était très simple : il utilisait le chiffrement par décalage. Et, vous allez le voir, cette technique est particulièrement facile à pirater.

Imaginons que Jules César souhaite envoyer le message suivant à un de ses généraux :

« Ce soir, on attaque les Gaulois! »

Puisque apparemment il aimait utiliser le chiffre 3 comme clé de chiffrement, il procédait alors au décalage suivant :

La première lettre du message est « C ». Il s'agit de la troisième lettre de l'alphabet et la clé est 3 donc 3 + 3 = 6. « F » est la sixième lettre de l'alphabet. « C » est donc remplacé par « F ».
Ce code consiste donc en une permutation circulaire des caractères. L'utilisation d'un simple disque à chiffrer (https://inventwithpython.com/cipherwheel/) facilite le codage d'un texte.

➔ A partir du disque à chiffrer (https://inventwithpython.com/cipherwheel/) **coder** le texte secret *« ce soir on attaque les gaulois »*
➔ A partir de cet exemple **décrire** les principales limites de ce codage.

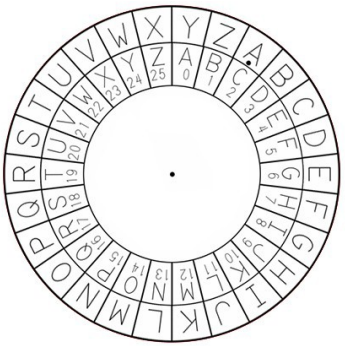

Schéma 1: Disque à chiffrer

2.Cryptage informatique du code de César

La table de codage du code de César peut facilement être construite à partir de la table ASCII. Cette table normalisée attribue à chaque caractère une valeur numérique (voir fiche savoir : Représentation de l'information)

➔ **Compléter** sur la première ligne du schéma suivant les 8 premiers caractères du texte crypté puis le codage ASCII en décimal correspondant. **Analyser** l'écart de valeur entre chaque caractère en clair et son équivalent crypté.

Texte en clair									
C	E		S	O	I	R		O	N
67	69		83	79	73	82		79	78

Texte crypté									

Algorithme de cryptage du code de César

L'algorithme 1 ci dessous permet de crypter un texte avec le codage de César en suivant un décalage *d*. Cet algorithme est incomplet. Si l'on ne tient pas compte de la zone rouge repérée, il effectue un décalage simple ce qui engendrera un débordement pour les caractères en fin d'alphabet. En effet si l'on décale de 3 la lettre « Y », nous sommes sensés obtenir la lettre « B » alors que le décalage simple effectué renverra le caractère « \ » (voir table ASCII).

```
Début
d=décalage
text=texte en clair

pour chaque caractère c de text
    convertir en hexa c
    c=c+d

    si c>____                    Zone à compléter
        alors c=____

    convertir c an ASCII
```

Algorithme 1: Cryptage avec le code de César

➔ **Compléter** avec des expressions mathématiques la zone rouge de l'algorithme 1 afin de gérer les débordements

Codage sous Python de l'algorithme de cryptage

Le code suivant est l'implémentation sous le langage Python de l'algorithme 1 de cryptage.

```
1 # -*- coding: utf-8 -*-
2 """
3 Éditeur de Spyder
4 """
5
6 d=3
7 crypte="" #Texte crypté
8 text="GAULOIS" #Texte en clair
9
10 for c in text:
11     c=ord(c)
12     c=c+d
13     if c>     :
14         c=
15
16     c=chr(c)
17     crypte=crypte+c  #Ajout des caracteres pour construire texte crypte
18
19 print("texte crypté : "+crypte)
```

Fonctions Python

ord() : renvoie le code décimal ASCII d'un caractère.
ord('a') renvoie la valeur 97

chr() : renvoie le caractère ASCII d'une valeur décimale. Cette fonction est l'inverse de ord().
chr(97) renvoie 'a'

Ce code est présent sur le réseau sur Moodle,

➔ **Compléter** le programme avec les expressions mathématiques définies précédemment. **Tester** le programme avec le mot en clair « GAULOIS » puis avec la phrase secrète « CE SOIR ON ATTAQUE LES GAULOIS ». **Vérifier** le résultat obtenu et **justifier** les écarts constatés. **Modifier** le code afin de rendre opérationnel programme.

3.Décryptage du code de César

➔ A partir de la traduction en langage Python de l'algorithme 1, **écrire** une fonction decryptage(message) de décryptage d'un message codé suivant la méthode du code de César avec un décalage de 5. **Définir** un test possible puis **valider** le fonctionnement ce programme.

4.Conclusion

➔ **Conclure** sur cette méthode de cryptage en répondant par écrit aux questions suivantes :

Combien y a-t-il de codages possibles ? Est-il sûr ? Est-il possible de trouver directement la clé en analysant la fréquence des lettres et en tenant compte de la langue utilisée ? Pourquoi le Rot 13 (cas particulier où le décalage est de 13) est il intéressant ?

5.Pour aller plus loin :

Amélioration 1 du programme de décryptage

➔ **Modifier** le programme de décryptage afin de décrypter un message défini avec un décalage quelconque. Il faudra pour cela tester et afficher les 26 combinaisons possibles.

➔ **Décrypter** le message envoyé par César à Cléopatre :

VQ BQZEQ CGQ HAGE MHQL OAYBDUE BMEEQL M XM EGUFQ

Amélioration 2 du programme de décryptage

Afin d'augmenter facilement le nombre de combinaisons et rendre insensible aux attaques fréquentielles il est possible de procéder à un décalage affine à la place d'un décalage fixe. La fonction de codage est ici du type f(x) = (ax + b) où x est le code ASCII correspondant à la lettre que l'on souhaite crypter. La clé est donc cette fois du type (a,b).

➔ **Coder** sous Python ce nouvel algorithme de cryptage.

Remarque : La gestion des débordements devra tenir compte du fait que la valeur de décalage pourra atteindre des valeurs importantes. Cette valeur étant modulo 26 (26 caractères dans l'alphabet) il sera nécessaire de ramener la valeur dans l'espace [0 ; 26]. En effet si le décalage est égale 30 ou à 56 ou à 82... le résultat de cryptage obtenu sera identique à un décalage de 4 (30 =1*26+4 ; 56 = 2*26+4 ; 82 = 3*26+4).