SAE Crypto

Droucheau Quentin , Gnaneswaran Roshan

Contenu du ZIP :

- rapport.md

Fichiers txt :

- indice1_chiffre.txt
- indice1_dechiffre.txt
- indice2_chiffre.txt
- indice2_dechiffre.txt

Fichiers pythons :

- cesar.py
- vigenere.py
- substitution.py

Message numéro 1 :

```
BDQE PG OTQYUZ EQ OMOTQ GZ FDQEAD
MOODAOTQ M GZ MDNDQ FAGF DQOAGHQDF P'AD
ZQ ZQSXUSQ BME XM VQGZQ BAGOQ RQGUXXG
SDMZP QEF EAZ EQODQF YMXSDQ EM FMUXXQ YQZGQ
DAZPQE QF OAXADQQE EAZF XQE NMUQE CG'UX BADFQ
MZUEQQE QF EGODQQE, XQGDE EMHQGDE EAZF RADFQE.
YMUE MFFQZFUAZ M ZQ BME XQE ODACGQD,
YQYQ EU XM RMUY FUDMUXXQ FQE QZFDMUXXQE,
QZ MGOGZ OME FG ZQ PAUE EGOOAYNQD
```

Tout d'abord nous avons essayé en premier le chiffrement de césar.Nous avions commencés par créer un dictionnaire de l'alphabet avec en clef la lettre et en valeur un indice commençant à partir de 0.

```
alphabet = dict()
i = 0
for lettre in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
    alphabet[lettre] = i
    i += 1
```

Par la suite nous avions donc utilisé le dictionnaire pour créer la méthode de déchiffrement de césar.

```
def dechiffrement_cesar(cle, texte):
    """
    Déchiffre un chiffrement de César avec une clé donnée.

    Args:
        cle (int): La clé utilisée pour déchiffrer le texte.
        texte (str): Le texte à déchiffrer.

    Returns:
        str: Le texte déchiffré.
    """
    global alphabet
    texte_decode = ""
    for lettre in texte:
        if lettre in alphabet.keys():
            nb = (alphabet.get(lettre) + cle) % 26

            for k in alphabet.keys():
                if alphabet[k] == nb:
                    texte_decode += k
        else:
            texte_decode += lettre
    return texte_decode
```

Afin de déterminer quel était le décalage nous avions fais une fonction permettant de tester les 26 décalages possibles.

```
def test_tous_les_decalage(texte):
    """
    Cette fonction prend en entrée un texte et affiche le texte chiffré
avec tous les décalages possibles.

    Args:
        texte (str): Le texte à chiffrer.
    """
    for decalage in range(26):
        print(f"Texte chiffré avec un décalage de {decalage} :
{dechiffrement_cesar(decalage, texte)}")
```

Pour plus de visibilité dans le terminal nous avions testé sur un mini texte :

```
Texte chiffré avec un décalage de 0 : BDQE PG OTQYUZ EQ OMOTQ GZ FDQEAD
MOODAOTQ
Texte chiffré avec un décalage de 1 : CERF QH PURZVA FR PNPUR HA GERFBE
NPPEBPUR
Texte chiffré avec un décalage de 2 : DFSG RI QVSAWB GS QOQVS IB HFSGCF
OQQFCQVS
Texte chiffré avec un décalage de 3 : EGTH SJ RWTBXC HT RPRWT JC IGTHDG
```

```
PRRGDRWT
Texte chiffré avec un décalage de 4 : FHUI TK SXUCYD IU SQSXU KD JHUIEH
QSSHESXU
Texte chiffré avec un décalage de 5 : GIVJ UL TYVDZE JV TRTYV LE KIVJFI
RTTIFTYV
Texte chiffré avec un décalage de 6 : HJWK VM UZWEAF KW USUZW MF LJWKGJ
SUUJGUZW
Texte chiffré avec un décalage de 7 : IKXL WN VAXFBG LX VTVAX NG MKXLHK
TVVKHVAX
Texte chiffré avec un décalage de 8 : JLYM XO WBYGCH MY WUWBY OH NLYMIL
UWWLIWBY
Texte chiffré avec un décalage de 9 : KMZN YP XCZHDI NZ XVXCZ PI OMZNJM
VXXMJXCZ
Texte chiffré avec un décalage de 10 : LNAO ZQ YDAIEJ OA YWYDA QJ PNAOKN
WYYNKYDA
Texte chiffré avec un décalage de 11 : MOBP AR ZEBJFK PB ZXZEB RK QOBPLO
XZZOLZEB
Texte chiffré avec un décalage de 12 : NPCQ BS AFCKGL QC AYAFC SL RPCQMP
YAAPMAFC
Texte chiffré avec un décalage de 13 : OQDR CT BGDLHM RD BZBGD TM SQDRNQ
ZBBQNBGD
Texte chiffré avec un décalage de 14 : PRES DU CHEMIN SE CACHE UN TRESOR
ACCROCHE
Texte chiffré avec un décalage de 15 : QSFT EV DIFNJO TF DBDIF VO USFTPS
BDDSPDIF
Texte chiffré avec un décalage de 16 : RTGU FW EJGOKP UG ECEJG WP VTGUQT
CEETQEJG
Texte chiffré avec un décalage de 17 : SUHV GX FKHPLQ VH FDFKH XQ WUHVRU
DFFURFKH
Texte chiffré avec un décalage de 18 : TVIW HY GLIQMR WI GEGLI YR XVIWSV
EGGVSGLI
Texte chiffré avec un décalage de 19 : UWJX IZ HMJRNS XJ HFHMJ ZS YWJXTW
FHHWTHMJ
Texte chiffré avec un décalage de 20 : VXKY JA INKSOT YK IGINK AT ZXKYUX
GIIXUINK
Texte chiffré avec un décalage de 21 : WYLZ KB JOLTPU ZL JHJOL BU AYLZVY
HJJYVJOL
Texte chiffré avec un décalage de 22 : XZMA LC KPMUQV AM KIKPM CV BZMAWZ
IKKZWKPM
Texte chiffré avec un décalage de 23 : YANB MD LQNVRW BN LJLQN DW CANBXA
JLLAXLQN
Texte chiffré avec un décalage de 24 : ZBOC NE MROWSX CO MKMRO EX DBOCYB
KMMBYMRO
Texte chiffré avec un décalage de 25 : ACPD OF NSPXTY DP NLNSP FY ECPDZC
LNNCZNSP
```

C'est ainsi qu'on a vu que le décalage était de 14 car elle formait une phrase compréhensible.

```
texte = "BDQE PG OTQYUZ EQ OMOTQ GZ FDQEAD MOODAOTQ M GZ MDNDQ FAGF
DQOAGHQDF P AD ZQ ZQSXUSQ BME XM VQGZQ BAGOQ RQGUXXG SDMZP QEF EAZ EQODQF
YMXSDQ EM FMUXXQ YQZGQ DAZPQE QF OAXADQQE EAZF XQE NMUQE CG'UX BADFQ
MZUEQQE QF EGODQQE, XQGDE EMHQGDE EAZF RADFQE YMUE MFFQZFUAZ M ZQ BME XQE
```

```
ODACGQD, YQYQ EU XM RMUY FUDMUXXQ FQE QZFDMUXXQE, QZ MGOGZ OME FG ZQ PAUE
EGOOAYNQD"
mini_texte = "BDQE PG OTQYUZ EQ OMOTQ GZ FDQEAD MOODAOTQ M"
test_tous_les_decalage(mini_texte)

print(dechiffrement_cesar(14, texte))
```

Afin d'exploiter le fichier txt donnée dans le sujet nous avons une fonction pour passer du fichier chiffré en un fichier déchiffré.

```
def dechiffrer_fichier_cesar(fichier, cle, fichier_sortie):
    """
    Cette fonction prend en entrée un fichier et affiche le texte
déchiffré avec tous les décalages possibles.

    Args:
        fichier (str): Le nom du fichier à déchiffrer.
        cle (int): La clé utilisée pour déchiffrer le texte.
        fichier_sortie (str): Le nom du fichier de sortie.
    """
    with open(fichier, "r") as f:
        texte = f.read()
    texte_decode = dechiffrement_cesar(cle, texte)
    with open(fichier_sortie, "w") as f:
        f.write(texte_decode)


dechiffrer_fichier_cesar("indice1_chiffre.txt", 14,
"indice1_dechiffre.txt")
```

Message trouvé :

```
PRES DU CHEMIN SE CACHE UN TRESOR
ACCROCHE A UN ARBRE TOUT RECOUVERT D'OR
NE NEGLIGE PAS LA JEUNE POUCE FEUILLU
GRAND EST SON SECRET MALGRE SA TAILLE MENUE
RONDES ET COLOREES SONT LES BAIES QU'IL PORTE
ANISEES ET SUCREES, LEURS SAVEURS SONT FORTES.
MAIS ATTENTION A NE PAS LES CROQUER,
MEME SI LA FAIM TIRAILLE TES ENTRAILLES,
EN AUCUN CAS TU NE DOIS SUCCOMBER
```

Le code caché dans le message est le suivant en prennat les premières lettres de chaque ligne :

```
PANGRAMME
```

Message numéro 2 :

Le message chiffré est le suivant :

```
AE IOW ZQBLNR WASIXQ WJR YKJ KGYUJAGY UU OXSLN TXRCUQYM
IY IRCTQ HPNF RR RQBIIIGOFN XQ WTCEKK DQ OIH MHXDUDQW BAYNVUDQYM
NR MRRPQD SU CXVMUQV HOHLWLQ CYT LRY GRQYMTRRY RPBMVXTVUES
QF EXNFO UEHAMAEM RV MQEWPGR IRCTQ HTREOVRQ XE HUOYKIFGXXOA
```

Nous avons donc utilisé la méthode de cryptage de vigenère. La méthode de déchiffrement de vigenère consiste à décaler chaque lettre du message par la lettre correspondante de la clé. Pour cela nous avons utilisé un dictionnaire avec en clef la lettre et en valeur un indice commençant à partir de 0.

Pour cela nous avons utilisé le code suivant :

```python
def dechiffrement_vigenere(cle, texte):
    """
    Déchiffre un chiffrement de Vigenère avec une clé donnée.

    Args:
        cle (str): La clé utilisée pour déchiffrer le texte.
        texte (str): Le texte à déchiffrer.

    Returns:
        str: Le texte déchiffré.
    """
    global alphabet
    texte_decode = ""
    i = 0
    for lettre in texte:
        if lettre in alphabet.keys():
            nb = (alphabet.get(lettre) - alphabet.get(cle[i])) % 26
            for k in alphabet.keys():
                if alphabet[k] == nb:
                    texte_decode += k
            i += 1
            if i == len(cle):
                i = 0
        else:
            texte_decode += lettre
    return texte_decode
```

Concernant la clé nous avions déduis que c'était "PANGRAMME" du message numéro 1. Nous avons donc utilisé la fonction suivante pour déchiffrer le message. Afin d'en être convaincu nous l'avions testé sur la première ligne du message.

```
premiere_ligne = "AE IOW ZQBLNR WASIXQ WJR YKJ KGYUJAGY UU OXSLN TXRCUQYM"
print(dechiffrement_vigenere("PANGRAMME", premiere_ligne))
```

Ce qui nous a permis d'obtenir le message suivant :

```
LE VIF ZEPHYR JUBILE SUR LES KUMQUATS DU CLOWN GRACIEUX
```

En utilisant le même principe que le message numéro 1 nous avons mis le résulat du fichier texte en entrée dans un autre. Ce qui nous permet d'obtenir le message suivant :

```
LE VIF ZEPHYR JUBILE SUR LES KUMQUATS DU CLOWN GRACIEUX
IL CACHE DANS LA REPETITION LE SECRET DE CES MURMURES MALHEUREUX
NE GARDEZ DU PREMIER SOUFFLE QUE LES PREMIERES APPARITIONS
ET AINSI DEVOILEZ LE MESSAGE CACHE DERRIERE LA SUBSTITUTION
```

Message numéro 3 :

```
EALOK, OKCT LOFX PLPSF! UF VKIF L ZKCASYA FTD: FUYXFEFDH
```

En lisant le message numéro 2 nous avions comme indice visible au premier abord la substitution. Le principe de substitution consite à remplacer chaque lettre du message par une autre lettre. C'est à dire qu'il nous faut un alphabet de substitution.

Avec l'indice précédant : PANGRAMME , dont la définition est une phrase contenant toutes les lettres de l'alphabet. En relisant le message nous avions compris que la première phrase était bien un pangramme. Cependant des lettres sont en double alors avec la suite de l'indice nous pouvons déduire qu'il suffit de garder les premieres apparitions de chaque lettre.

C'est alors qu'avec le pangramme afin d'obtenir un alphabet de substitution nous avons utilisé le code suivant :

```
pangramme = "LE VIF ZEPHYR JUBILE SUR LES KUMQUATS DU CLOWN GRACIEUX"

def garde_premiere_appariton(texte):
    """
    Retourne une chaîne de caractères contenant les caractères uniques de
la chaîne d'entrée, dans l'ordre de leur première apparition.

    Args:
    - texte (str): La chaîne d'entrée à partir de laquelle extraire les
caractères uniques.
```

```
    Returns:
    - alphabet_substitution (str): Une chaîne de caractères contenant les
caractères uniques de la chaîne d'entrée, dans l'ordre de leur première
apparition.
    """

    deja_vu = set()

    alphabet_substitution = ""

    for i in range(len(texte)):
        if texte[i] != " " and texte[i] not in deja_vu:
            alphabet_substitution += texte[i]
            deja_vu.add(texte[i])

    return alphabet_substitution
```

Ce qui nous donne l'alphabet de substitution suivant :

```
LEVIFZPHYRJUBSKMQATDCOWNGX
```

Afin de le rendre manipulable nous l'avions mis dans un dictionnaire avec en clef la lettre de l'alphabet de substitution et en valeur la lettre de l'alphabet.

```
def dico_alphabet_substitution(texte):
    """
    Cette fonction prend une chaîne de caractères en entrée et
    renvoie un dictionnaire qui associe chaque caractère de la chaîne
d'entrée
    à un caractère correspondant dans l'alphabet.

    Args:
    - texte (str): La chaîne de caractères à associer à l'alphabet.

    Returns:
    - dico_alphabet (dict): Un dictionnaire qui associe chaque caractère
    de la chaîne d'entrée à un caractère correspondant dans l'alphabet.
    """

    alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    dico_alphabet = dict()

    for i in range(len(texte)):
        dico_alphabet[texte[i]] = alphabet[i]

    return dico_alphabet
```

Par la suite nous avions une fonction substitution qui prend en entrée le message et le dictionnaire de l'alphabet de substitution et qui renvoie le message déchiffré.

```
def substitution(texte, alphabet_substitution):
    """
    Déchiffre un texte chiffré par substitution avec un alphabet de
substitution donné.

    Args:
        texte (str): Le texte à déchiffrer.
        alphabet_substitution (str): L'alphabet de substitution utilisé
pour déchiffrer le texte.

    Returns:
        str: Le texte déchiffré.
    """

    resultat = ""

    for lettre in texte:
        if lettre.isalpha():
            resultat += alphabet_substitution[lettre]
        else:
            resultat += lettre

    return resultat

substitution(message_dechiffre, dico_substitution)
```

Ainsi nous obtenons le message suivant :

```
BRAVO, VOUS AVEZ GAGNE! LE CODE A FOURNIR EST: ELIZEBETH
```