

Thème : Types de bases

Cryptage et décryptage

18 d'un message

1. Approfondissement : application au cryptage d'un message



Cryptage

Transformation d'un message en clair en un message codé compréhensible seulement par qui dispose du code

L'informatique permet d'automatiser le procédé de cryptage et décryptage d'un message, dans ce notebook vous allez simuler le fonctionnement d'un logiciel de mail qui crypte/décrypte les messages.

Deux personnes 'Alice' et 'Bob' vont utiliser ce procédé pour communiquer.

Cher agent 'EVE', votre mission sera de découvrir le message envoyé par Alice à Bob qui a été intercepté en écoutant le réseau. Nos cryptographes n'ont pas réussi, nous comptons sur vous ! Voici le message :

```
0100110101110100110100101110010111101001110101001011010110001011010100111000001  
101110001000001110000000100000011001110010011101100110100101100010011001110111  
101000100000011100101110100010000001100111011001000000110110101110000011110  
100010000001101101011010001110110011001110111010001000000111101001101111  
0010000001110010111010001000000110011101100010000001101101011100000111101000  
1000000111001011101000100000010101110110100111000001101101011100001001101010011
```

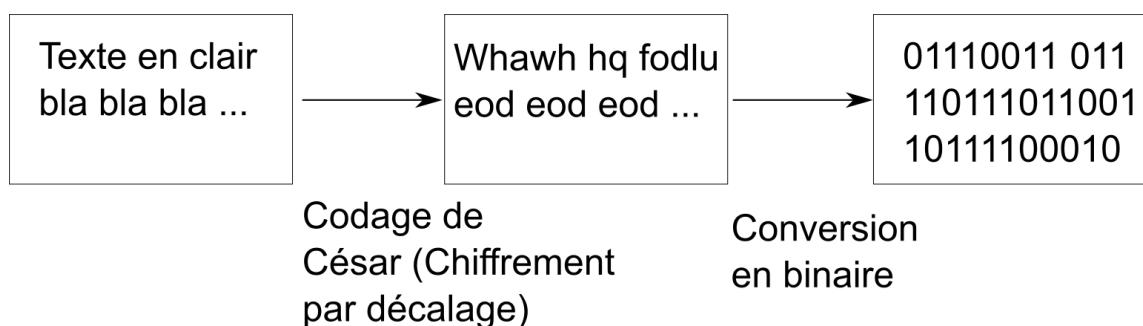
```
0010011010010111101000100000111000000010000001101000011001000111100101100100001000
0001101011011001000110011101110100010000000100001
```

Nos équipes ont mis la main sur la documentation de leur logiciel de messagerie, à vous d'en comprendre le fonctionnement pour déchiffrer leur message.

1.1 Principe de fonctionnement.

Le texte en clair sera : - chiffré par Alice en utilisant le code de césar, c'est un chiffrement par décalage des lettres. Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres (dans le cas d'un décalage à droite), on reprend au début. Par exemple avec un décalage de 3 vers la droite, A est remplacé par D, B devient E, et ainsi jusqu'à W qui devient Z, puis X devient A etc... - chaque caractère sera ensuite converti en binaire sur un octet (8 bits).

C'est ce message binaire qui sera envoyé à Bob qui devra le décoder, Bob connaît la clé qu'Alice a utilisée pour son chiffrement.



1.2 Chiffrement de césar.

[Wikipédia : Chiffrement de César](#)

Fonction `ord` :

Renvoie le nombre entier représentant le code Unicode du caractère représenté par la chaîne donnée. Par exemple, `ord('a')` renvoie le nombre entier 97 et `ord('€')` (symbole Euro) renvoie 8364. Il s'agit de l'inverse de `chr()`.

Fonction `chr()`

Renvoie la chaîne représentant un caractère dont le code de caractère Unicode est le nombre entier `i`. Par exemple, `chr(97)` renvoie la chaîne de caractères 'a', tandis que `chr(8364)` renvoie '€'. Il s'agit de l'inverse de `ord()`.

L'intervalle valide pour cet argument est de 0 à 1114111 (0x10FFFF en base 16). Une exception `ValueError` sera levée si `i` est en dehors de l'intervalle.

Fonction % 26

Renvoie le reste de la division entière par 26. Par exemple : 67 % 26 donne 15

Fonction .split()

La méthode split() divise une chaîne selon le séparateur spécifié et renvoie une liste de chaînes.

```
str = "Bonjour a tous"
res = str.split()
print(res)
['Bonjour', 'a', 'tous']
```

Les codes UTF-8 et ASCII sont identiques, ainsi pour vous aider voici un [lien vers une table ASCII](#)

```
def chiffrement_caractere(carac,cle):
    """
    Chiffre un caractère par la méthode de césar , les lettres a..z et A..Z
    sont décalées par la méthode de césar
    les autres caractères ne sont pas modifiés (accents, tiret ...)
    Entrées :
        carac (str) un caractère
        cle (int) la clé de codage (classiquement entre 0 et 25)
    Sortie :
        (str) Le caractère décalé par la méthode de césar
    """
    if 65<= ord(carac) <=90:
        # Votre code ici
    elif 97 <= ord(carac) <= 122:
        # Votre code ici
    else :
        # Votre code ici
    return # Votre code ici
```

```
# Tests, si pas d'erreurs continuez.
assert chiffrement_caractere('A',3) == 'D'
assert chiffrement_caractere('T',5) == 'Y'
assert chiffrement_caractere('w',7) == 'd'
# Gestion accent ...
assert chiffrement_caractere('é',10) == 'é'
assert chiffrement_caractere('ç',15) == 'ç'
```

```
def chiffrement_mot(mot,cle):
    """
    Chiffre un mot (chaîne de caractères sans espace) par la méthode de césar
    Entrées :
        mot (str)
        cle (int) la clé de codage
    Sortie :
        (str) Le mot décalé par la méthode de césar
    """
```

```

# Votre code ici
return # Votre code ici

# Tests, si pas d'erreurs continuez.

assert chiffrement_mot('Bonjour', 7) == 'Ivuqvby'
assert chiffrement_mot('Décodage', 16) == 'Tésetqwu'
assert chiffrement_mot('Bonjour', 33) == 'Ivuqvby'
assert chiffrement_mot('Anticonstitutionnellement', 4) ==
'Erxmgsrwxmxymxsrrippiqirx'

def decoupage(texte):
    """
    Découpe un texte suivant ses espaces
    Entrée : texte (str)
    Sortie : (lst) une liste de str constitué des mots du texte
    """
    # Votre code ici
    return # Votre code ici

# Tests, si pas d'erreurs continuez.
assert decoupage('Ceci est un test.') == ['Ceci', 'est', 'un', 'test.']
assert decoupage(' Un texte avec des espaces !!!') == ['Un', 'texte',
'avec', 'des', 'espaces', '!!!']

def cesar(texte, cle):
    """
    Chiffre un texte par la méthode de césar. Le texte ne commence, ni ne se
    termine par des espaces.
    Entrées :
        texte (str)
        cle (int) la clé de codage
    Sortie :
        (str) Le texte décalé par la méthode de césar
    """
    # Votre code ici
    return # Votre code ici

# Tests, si pas d'erreurs continuez.
assert cesar('Bonjour', 7) == 'Ivuqvby'
assert cesar("WIKIPEDIA ENCYCLOPEDIE LIBRE", 9) == 'FRTRYNMRJ NWLHLUXYNMRN
URKAN'
assert cesar("Le texte chiffré s'obtient en remplaçant chaque lettre du texte
clair original par une lettre à distance fixe.", 11) == "Wp epiet nstqqc
d'zmetpye py cpxawlçlye nslbfp wpeecp of epiet nwltc zctrtylw alc fyp wpeecp à
otdelynp qtip."

```

1.3 Conversion en binaire

Compléter les fonction de conversion entre base 2 et base 10.

- Soit vous essayez de faire le programme sans utiliser de fonction intégrée à python (A privilégier pour travailler la programmation et la compréhension du processus)
- Soit vous utiliser une méthode existante dans python :

```
>>> format(138, '08b')
'10001010'
>>> format(103, '08b')
'01100111'
```

Dans ce cas ne pas tester avec le assert

```
def dec-vers-bin(nombre):
    """
    Convertir un nombre entier naturel vers la base 2 sans passer par les
    fonctions pré-existantes.

    Entrées :
        nombre (int) : entier naturel
    Sortie :
        (str) Le nombre écrit en base 2
    """
    # Votre code ici
    return # Votre code ici
```

```
from random import randint
# Tests, si pas d'erreurs continuez.
assert dec-vers-bin(1) == '1'
assert dec-vers-bin(2) == '10'
assert dec-vers-bin(1) == '1'
assert dec-vers-bin(17) == '10001'
# test aléatoire
nbe = randint(3,1000)
assert dec-vers-bin(nbe) == bin(nbe)[2:]
```

```
def bin-vers-dec(binaire):
    """
    Convertir un nombre binaire vers la base 10 sans passer par les fonctions
    pré-existantes.

    Entrées :
        binaire (str) : un nombre binaire
    Sortie :
        (int) Le nombre écrit en base 10
    """
    # Votre code ici
    return # Votre code ici
```

```
# Tests, si pas d'erreurs continuez.
assert (bin-vers-dec('1111011') == 123 )
assert (bin-vers-dec('101111011') == int('0b101111011',2) )
assert (bin-vers-dec('1001101') == int('0b1001101',2) )
# test aléatoire
nbe = randint(30,1000)
```

```
binaire = bin(nbe)[2:]
assert bin_vers_dec(binaire) == nbe
```

1.4 Codage et décodage du message

Rappels :

Fonction `ord` :

Renvoie le nombre entier représentant le code Unicode du caractère représenté par la chaîne donnée. Par exemple, `ord('a')` renvoie le nombre entier 97 et `ord('€')` (symbole Euro) renvoie 8364. Il s'agit de l'inverse de `chr()`.

Fonction `chr()`

Renvoie la chaîne représentant un caractère dont le code de caractère Unicode est le nombre entier `i`. Par exemple, `chr(97)` renvoie la chaîne de caractères '`a`', tandis que `chr(8364)` renvoie '`€`'. Il s'agit de l'inverse de `ord()`.

L'intervalle valide pour cet argument est de 0 à 1114111 (0x10FFFF en base 16). Une exception `ValueError` sera levée si `i` est en dehors de l'intervalle.

Pour chaque caractère codé par la méthode de César, on trouve son code UTF-8 puis on le converti en binaire sur 8 bits.

```
def codage_message(texte,cle):
    """
    Chiffre un texte par la méthode de césar puis code les caractères en
    binaire en utilisant leur code UTF-8
    Le texte ne commence, ni ne se termine par des espaces.
    Entrées :
        texte (str)
        cle (int) la clé de codage
    Sortie :
        (str) Le texte codé
    """
    # Votre code ici
    return # Votre code ici
```

```
# Tests, si pas d'erreurs continuez.
assert codage_message('Coucou le monde',5) ==
'0100100001110100011110100110100001110100011110100010000001110001011010101000100000

assert codage_message('César',10) ==
'0100110111101001011000110110101101100010'
```

```
def decodage_message(binaire,cle):
    """
    Déchiffre un message binaire codé avec la méthode du notebook
```

```
Le texte ne commence, ni ne se termine par des espaces.  
Entrées :  
    binaire (str)  
    cle (int) la clé de décodage  
Sortie :  
    (str) Le texte décodé  
"""  
# Votre code ici  
return # Votre code ici
```

```
# Tests, si pas d'erreurs continuez.  
assert  
decodage_message('010010000111010001111010011010000111010001111010001000000111000  
== 'Coucou le monde'  
assert decodage_message('0100110111101001011000110110101101100010', 10) ==  
'César'
```

```
# A vous de décoder le message d'Alice, à vous d'être malin pour décoder sans  
connaître la clé.  
message_code =  
'01010111011001000111101001101001001000000110010101101010011100001110100100101100  
# Votre code ici
```