Défi

Le texte suivant a été encodé en ISO-8859-1, décodez-le.

```
01000001011101000111010001100101011011100111010001101001011011110110111
00010110000100000011000110110010100100000011011010110010101110011011100
11011000010110011101100101001000000110000100100000011001010110111000100
00001100110011000010110100101110100001000001100001110101001011101001100
00111010100100100000011001010110111001100011011011110110010011000011101
01001001000000110010101101110001000000101010101010100010001100010110100
11100000100000001000010000101000001010010101010110111000100000011100000
11000010110111001100111011100100110000101101101011011010110010100100000
00111010001000001100001010101011110000101010000001110110011011110110100
10111100000100000011000010110110101100010011010010110011101110101110000
11101010110010000001100100001001110111010101101110001000000110001111000
10110010011011101010111001000100000011100010111010101101001001011000010
00000110000101110101001000000111101011000011101010010111000001101000011
11001011100100010110000100000011100000111001011000011101010010110011011
00001110101000011100100110010100100000011011000110010101110011001000000
11010100110000101110100011101000110010101110011001000000110010001100101
00100000011010110110100101110111011010010111001111000010101000001100001
01011101100101110000010100000101001000101011011100110001101101111011001
00011001010111101000100000011011000110000100100000011100000110100001110
01001100001011100110110010100100000011001000010011101101001011011100111
01000111001001101111011001000111010101100011011101000110100101101111011
01110001000000110010001100101001000000110001101100101011101000010000001
10010101111000011001010111001001100011011010010110001101100101001000000
11001010110111000100000010101010101010001000110001011010011100000101110
```

Pour un décodage automatisé, peut-être préfèrerez-vous le message sous cette forme:

```
0100000101110100011101000110010101101110011101000110100101101111011011100010110000100000011
```

Corrigé

In [3]:
```
def decoupage(chaine):
    """
    Découpe la longue chaîne de caractères chaine en des petites chaînes de 8 ca
ractères.
    Les petites chaines sont stockees dans la liste retournee.

    Parametres nommes
    -----------------
        chaine : de type str
        La chaine de caratères 0 et 1 constituant le message a decoder.

    Retourne
    --------
        liste : de type list
        La liste des petites chaines de 8 caracteres 0 et 1.

    """
    liste = []
    for i in range(0, len(chaine), 8):
        octet_str = chaine[i : i+8]  #  Extraction successive de petites chaine
s.
        liste.append(octet_str)

    return liste
```

In [4]:
```
ma_chaine = '0100000101110100011101000110010101101110011101000110100101101111011
0111000101100001000000110001101100101001000000110110101100101011100110111001101
0000101100111011001010010000001100001001000000110010101101110001000000110011001
0000101101001011101000010000011000011101010010111010011000011101010010010000001
0010101101110011000110110111101100100110000111010100100100000011001010110111000
0000001010101010101000100011000101101001110000010000000100001000010100000101001
0101011011100010000001110000011000010110111001100111011100100110000101101101011
0110101100101001000000011101000100000110000101010101111000010101000000111011001
0111101101001011110000010000001100001011011010110001001101001011001110111010111
0001110101011001000000110010000100111011101010110111000100000011000111100010110
0100110111010101110010001000000111000101110101011010010010110000100000011000010
1010100100000011110101100001110101001011100000110100001111001011100100010110000
0000001110000011100101100001110101001011001101100001110101000011100100110010100
0000001101100011001010111001100100000011010100110000101110100011101000110010101
1001100100000011001000110010100100000011010110110100101110111011010010111001111
0001010100000110000101011101100101110000010100000101001000101011011100110011011
0111101100100011001010111101000100000011011000110000100100000011100000110100001
1001001100001011100110110010100100000011001000010011101101001011011100111010001
1001001101111011001000111010101100011011101000110100101101111011011100010000001
0010001100101001000000110001101100101011101000010000001100101011110000110010101
1001001100011011010010110001101100101001000000110010101101110001000000101010101
0101000100011000101101001110000010111 0'
ma_liste = decoupage(ma_chaine)

print(ma_liste)
```

```
['01000001', '01110100', '01110100', '01100101', '01101110', '01110100', '0110
1001', '01101111', '01101110', '00101100', '00100000', '01100011', '01100101',
'00100000', '01101101', '01100101', '01110011', '01110011', '01100001', '01100
111', '01100101', '00100000', '01100001', '00100000', '01100101', '01101110',
'00100000', '01100110', '01100001', '01101001', '01110100', '00100000', '11000
011', '10101001', '01110100', '11000011', '10101001', '00100000', '01100101',
'01101110', '01100011', '01101111', '01100100', '11000011', '10101001', '00100
000', '01100101', '01101110', '00100000', '01010101', '01010100', '01000110',
'00101101', '00111000', '00100000', '00100001', '00001010', '00001010', '01010
101', '01101110', '00100000', '01110000', '01100001', '01101110', '01100111',
'01110010', '01100001', '01101101', '01101101', '01100101', '00100000', '00111
010', '00100000', '11000010', '10101011', '11000010', '10100000', '01110110',
'01101111', '01101001', '01111000', '00100000', '01100001', '01101101', '01100
010', '01101001', '01100111', '01110101', '11000011', '10101011', '00100000',
'01100100', '00100111', '01110101', '01101110', '00100000', '01100011', '11000
101', '10010011', '01110101', '01110010', '00100000', '01110001', '01110101',
'01101001', '00101100', '00100000', '01100001', '01110101', '00100000', '01111
010', '11000011', '10101001', '01110000', '01101000', '01111001', '01110010',
'00101100', '00100000', '01110000', '01110010', '11000011', '10101001', '01100
110', '11000011', '10101000', '01110010', '01100101', '00100000', '01101100',
'01100101', '01110011', '00100000', '01101010', '01100001', '01110100', '01110
100', '01100101', '01110011', '00100000', '01100100', '01100101', '00100000',
'01101011', '01101001', '01110111', '01101001', '01110011', '11000010', '10100
000', '11000010', '10111011', '00101110', '00001010', '00001010', '01000101',
'01101110', '01100011', '01101111', '01100100', '01100101', '01111010', '00100
000', '01101100', '01100001', '00100000', '01110000', '01101000', '01110010',
'01100001', '01110011', '01100101', '00100000', '01100100', '00100111', '01101
001', '01101110', '01110100', '01110010', '01101111', '01100100', '01110101',
'01100011', '01110100', '01101001', '01101111', '01101110', '00100000', '01100
100', '01100101', '00100000', '01100011', '01100101', '01110100', '00100000',
'01100101', '01111000', '01100101', '01110010', '01100011', '01101001', '01100
011', '01100101', '00100000', '01100101', '01101110', '00100000', '01010101',
'01010100', '01000110', '00101101', '00111000', '00101110']
```

In [5]:
```
def conversion_int(liste):
    """
    Convertit les petites chaines de 8 caractères 0 et 1 présentes dans
    la liste donnee en argument en des entiers. Les entiers sont stockes dans la
liste retournee.

    Parametres nommes
    -----------------
        liste : de type list
        Une liste de petites chaines de 8 caracteres 0 et 1

    Retourne
    --------
        liste_int : de type list
        La liste contenant les petites chaines converties en entiers d'après leu
r valeur en binaire.

    """
    liste_int = []
    for i in range(len(liste)):
        element = int(liste[i], 2)  # Convertit en entier une chaine de 0 et de
1
        liste_int.append(element)
    return liste_int
```

In [6]:
```
ma_liste_int = conversion_int(ma_liste)
print(ma_liste_int)
```

```
[65, 116, 116, 101, 110, 116, 105, 111, 110, 44, 32, 99, 101, 32, 109, 101, 11
5, 115, 97, 103, 101, 32, 97, 32, 101, 110, 32, 102, 97, 105, 116, 32, 195, 16
9, 116, 195, 169, 32, 101, 110, 99, 111, 100, 195, 169, 32, 101, 110, 32, 85,
84, 70, 45, 56, 32, 33, 10, 10, 85, 110, 32, 112, 97, 110, 103, 114, 97, 109,
109, 101, 32, 58, 32, 194, 171, 194, 160, 118, 111, 105, 120, 32, 97, 109, 98,
105, 103, 117, 195, 171, 32, 100, 39, 117, 110, 32, 99, 197, 147, 117, 114, 3
2, 113, 117, 105, 44, 32, 97, 117, 32, 122, 195, 169, 112, 104, 121, 114, 44,
32, 112, 114, 195, 169, 102, 195, 168, 114, 101, 32, 108, 101, 115, 32, 106, 9
7, 116, 116, 101, 115, 32, 100, 101, 32, 107, 105, 119, 105, 115, 194, 160, 19
4, 187, 46, 10, 10, 69, 110, 99, 111, 100, 101, 122, 32, 108, 97, 32, 112, 10
4, 114, 97, 115, 101, 32, 100, 39, 105, 110, 116, 114, 111, 100, 117, 99, 116,
105, 111, 110, 32, 100, 101, 32, 99, 101, 116, 32, 101, 120, 101, 114, 99, 10
5, 99, 101, 32, 101, 110, 32, 85, 84, 70, 45, 56, 46]
```

In [7]:
```
def decode_latin1(liste_int):
    """
    Retourne la liste des caracteres presents dans la liste d'entiers donnee en argument,
    selon les points de code de la table de caracteres iso-8859-1 (ou latin-1).

    Parametres nommes
    -----------------
        liste_int : de type list
        Une liste d'entiers representant les caracteres du message a decoder.

    Retourne
    --------
        liste_str : de type list
        La liste des caracteres du message en suivant la charset 'latin-1'


    """

    liste_str = []
    for i in range(len(liste_int)):
        caractere_str = chr(ma_liste_int[i])  #  chr donne les points de code 'latin-1'
        liste_str.append(caractere_str)
    return liste_str
```

In [8]:
```
ma_liste_str = decode_latin1(ma_liste_int)
print(ma_liste_str)
```

```
['A', 't', 't', 'e', 'n', 't', 'i', 'o', 'n', ',', ' ', 'c', 'e', ' ', 'm', 'e
', 's', 's', 'a', 'g', 'e', ' ', 'a', ' ', 'e', 'n', ' ', 'f', 'a', 'i', 't',
' ', 'Ã', '©', 't', 'Ã', '©', ' ', 'e', 'n', 'c', 'o', 'd', 'Ã', '©', ' ', 'e
', 'n', ' ', 'U', 'T', 'F', '-', '8', ' ', '!', '\n', '\n', 'U', 'n', ' ', 'p
', 'a', 'n', 'g', 'r', 'a', 'm', 'm', 'e', ' ', ':', ' ', 'Â', '«', 'Â', '\xa0
', 'v', 'o', 'i', 'x', ' ', 'a', 'm', 'b', 'i', 'g', 'u', 'Ã', '«', ' ', 'd',
"'", 'u', 'n', ' ', 'c', 'Å', '\x93', 'u', 'r', ' ', 'q', 'u', 'i', ',', ' ',
'a', 'u', ' ', 'z', 'Ã', '©', 'p', 'h', 'y', 'r', ',', ' ', 'p', 'r', 'Ã', '©
', 'f', 'Ã', '¨', 'r', 'e', ' ', 'l', 'e', 's', ' ', 'j', 'a', 't', 't', 'e',
's', ' ', 'd', 'e', ' ', 'k', 'i', 'w', 'i', 's', 'Â', '\xa0', 'Â', '»', '.',
'\n', '\n', 'E', 'n', 'c', 'o', 'd', 'e', 'z', ' ', 'l', 'a', ' ', 'p', 'h', '
r', 'a', 's', 'e', ' ', 'd', "'", 'i', 'n', 't', 'r', 'o', 'd', 'u', 'c', 't',
'i', 'o', 'n', ' ', 'd', 'e', ' ', 'c', 'e', 't', ' ', 'e', 'x', 'e', 'r', 'c
', 'i', 'c', 'e', ' ', 'e', 'n', ' ', 'U', 'T', 'F', '-', '8', '.']
```

- Ci-dessus on découvre que ce message est en fait encodé en utf-8.
- On va donc utiliser la méthode .decode pour decoder :

```
mon_message.decode('utf-8')
```

- mon_message doit être du type bytes.

In [17]:
```
def decode_utf8(liste_int):
    """
    Retourne la liste des caracteres presents dans la liste d'entiers donnee en
argument,
    selon les points de code de la table de caracteres utf-8.

    Parametres nommes
    -----------------
        liste_int : de type list
        Une liste d'entiers representant les caracteres du message a decoder.

    Retourne
    --------
        liste_str : de type list
        La liste des caracteres du message en suivant la charset 'utf-8'


    """

    message_bytes = bytes(liste_int)  #  Convertit en bytes la liste d'entiers
    message = message_bytes.decode('utf-8')
    return message
```

In [18]:
```
ma_liste_str = decode_utf8(ma_liste_int)
print(ma_liste_str)
```

```
Attention, ce message a en fait été encodé en UTF-8 !

Un pangramme : « voix ambiguë d'un cœur qui, au zéphyr, préfère les jattes de
kiwis ».

Encodez la phrase d'introduction de cet exercice en UTF-8.
```

- Ci-dessus on découvre la consigne " Encodez la phrase d'introduction de cet exercice en UTF-8" . La phrase est :

```
Le texte suivant a été encodé en ISO-8859-1, décodez-le.
```

In [ ]:
```
def encode_utf8(chaine):
    """
    Retourne en bytes les caracteres presents dans la chaine donnee en argument,
    selon les points de code de la table de caracteres utf-8.

    Parametres nommes
    -----------------
        chaine : de type str
        Une chaine de caracteres a encoder.

    Retourne
    --------
        octets : de type bytes
        La chaine de bytes de la chaine encodee en suivant la charset 'utf-8'


    """

    octets = chaine.encode('utf-8')
    return octets
```

In [21]:
```
ma_chaine = 'Le texte suivant a été encodé en ISO-8859-1, décodez-le.'
mes_octets = encode_utf8(ma_chaine)
print(mes_octets)
```

```
b'Le texte suivant a \xc3\xa9t\xc3\xa9 encod\xc3\xa9 en ISO-8859-1, d\xc3\xa9c
odez-le.'
```

- Option 1 : La même fonction, mais qui retourne la liste des octets en hexadécimal

In [29]:
```
def encode2_utf8(chaine):
    """
    Retourne en bytes les caracteres presents dans la chaine donnee en argument,
    selon les points de code de la table de caracteres utf-8.

    Parametres nommes
    -----------------
        chaine : de type str
        Une chaine de caracteres a encoder.

    Retourne
    --------
        liste_octets : de type list
        La listes d'octets de la chaine encodee en suivant la charset 'utf-8'


    """

    octets = chaine.encode('utf-8')
    liste_octets = []
    for element in octets:
        hexadecimal_str = hex(element)  # Transforme l'ecriture de l'entier en c
haine de caracteres hexadecimale
        liste_octets.append(hexadecimal_str)

    return liste_octets
```

In [31]:
```
ma_chaine = 'Le texte suivant a été encodé en ISO-8859-1, décodez-le.'
ma_liste = encode2_utf8(ma_chaine)
print(ma_liste)
```

```
['0x4c', '0x65', '0x20', '0x74', '0x65', '0x78', '0x74', '0x65', '0x20', '0x73
', '0x75', '0x69', '0x76', '0x61', '0x6e', '0x74', '0x20', '0x61', '0x20', '0x
c3', '0xa9', '0x74', '0xc3', '0xa9', '0x20', '0x65', '0x6e', '0x63', '0x6f', '
0x64', '0xc3', '0xa9', '0x20', '0x65', '0x6e', '0x20', '0x49', '0x53', '0x4f',
'0x2d', '0x38', '0x38', '0x35', '0x39', '0x2d', '0x31', '0x2c', '0x20', '0x64
', '0xc3', '0xa9', '0x63', '0x6f', '0x64', '0x65', '0x7a', '0x2d', '0x6c', '0x
65', '0x2e']
```

- Option 2 : La même fonction mais retourne la liste des octets en binaire

In [30]:
```
def encode3_utf8(chaine):
    """
    Retourne en bytes les caracteres presents dans la chaine donnee en argument,
    selon les points de code de la table de caracteres utf-8.

    Parametres nommes
    -----------------
        chaine : de type str
        Une chaine de caracteres a encoder.

    Retourne
    --------
        liste_bits : de type list
        La listes de bits de la chaine encodee en suivant la charset 'utf-8'


    """

    octets = chaine.encode('utf-8')
    liste_bits = []
    for element in octets:
        binaire_str = bin(element)  # Transforme l'ecriture de l'entier en chain
e de caracteres binaire
        liste_bits.append(binaire_str)

    return liste_bits
```

In [32]:
```
ma_chaine = 'Le texte suivant a été encodé en ISO-8859-1, décodez-le.'
ma_liste = encode3_utf8(ma_chaine)
print(ma_liste)
```

```
['0b1001100', '0b1100101', '0b100000', '0b1110100', '0b1100101', '0b1111000',
'0b1110100', '0b1100101', '0b100000', '0b1110011', '0b1110101', '0b1101001', '
0b1110110', '0b1100001', '0b1101110', '0b1110100', '0b100000', '0b1100001', '0
b100000', '0b11000011', '0b10101001', '0b1110100', '0b11000011', '0b10101001',
'0b100000', '0b1100101', '0b1101110', '0b1100011', '0b1101111', '0b1100100', '
0b11000011', '0b10101001', '0b100000', '0b1100101', '0b1101110', '0b100000', '
0b1001001', '0b1010011', '0b1001111', '0b101101', '0b111000', '0b111000', '0b1
10101', '0b111001', '0b101101', '0b110001', '0b101100', '0b100000', '0b1100100
', '0b11000011', '0b10101001', '0b1100011', '0b1101111', '0b1100100', '0b11001
01', '0b1111010', '0b101101', '0b1101100', '0b1100101', '0b101110']
```

- Option 3 : La même fonction que encode2, mais qui retourne uniquement les chiffres hexadécimaux sous forme d'une chaîne de caractères chaque octet étant séparé par un espace.

In [43]:
```
def encode4_utf8(chaine):
    """
    Retourne en bytes les caracteres presents dans la chaine donnee en argument,
    selon les points de code de la table de caracteres utf-8.

    Parametres nommes
    -----------------
        chaine : de type str
        Une chaine de caracteres a encoder.

    Retourne
    --------
        chaine_octets : de type str
        La chaine de caracteres hexadecimaux de la chaine encodee en suivant la
charset 'utf-8'


    """

    octets = chaine.encode('utf-8')
    chaine_octets = ''
    for element in octets:
        hexadecimal_str = hex(element)  # Transforme l'ecriture de l'entier en c
haine de caracteres hexadecimale
        hexadecimal_str_nettoyee = hexadecimal_str[2:] # Elimine les caracteres
0x situes au debut.
        chaine_octets = chaine_octets + hexadecimal_str_nettoyee + ' '  #  Ajout
e un bout de chaine et un espace.

    return chaine_octets
```

In [44]:
```
ma_chaine = 'Le texte suivant a été encodé en ISO-8859-1, décodez-le.'
ma_chaine_octets = encode4_utf8(ma_chaine)
print(ma_chaine_octets)
```

```
4c 65 20 74 65 78 74 65 20 73 75 69 76 61 6e 74 20 61 20 c3 a9 74 c3 a9 20 65
6e 63 6f 64 c3 a9 20 65 6e 20 49 53 4f 2d 38 38 35 39 2d 31 2c 20 64 c3 a9 63
6f 64 65 7a 2d 6c 65 2e
```