Explication du Code Python pour l'Encodeur/Décodeur Base64

Introduction

Ce document explique le fonctionnement du script Python conçu pour encoder et décoder des chaînes en utilisant différentes transformations de la base64. Le code permet de choisir entre mélanger, incrémenter, ou appliquer un décalage fixe à la base64 pour encoder ou décoder des textes.

Fonction b64

La fonction 'b64' retourne la chaîne des caractères utilisés en Base64. Elle utilise un cache pour ne calculer la valeur qu'une seule fois, améliorant ainsi les performances lors des appels répétés. Cette chaîne est utilisée comme référence pour encoder et décoder des données.

```
@functools.lru_cache()
def b64():
    return 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
```

Fonction shuffle_b64

La fonction 'shuffle_b64' mélange aléatoirement les caractères de la base64. Elle prend en paramètre la base et une graine pour le générateur de nombres aléatoires, assurant que le mélange peut être reproduit si la même graine est utilisée.

```
def shuffle_b64(base, seed):
    random.seed(seed)
    lst = list(base)
    random.shuffle(lst)
    return ''.join(lst)
```

Fonctions increment_b64 et fixed_shift_b64

Ces fonctions modifient la séquence des caractères de la base64 par rotation. 'increment_b64' effectue une rotation basée sur un décalage qui peut varier, tandis que 'fixed_shift_b64' utilise un décalage fixe, donnant un contrôle plus prévisible.

```
def increment_b64(base, shift):
    return base[-shift:] + base[:-shift]

def fixed_shift_b64(base, shift):
    return base[-shift:] + base[:-shift]
```

Fonctions encode et decode

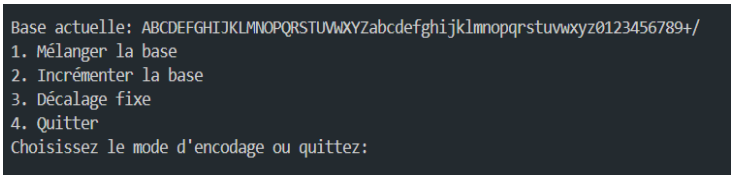
'encode' convertit une chaîne en sa représentation binaire, puis en une chaîne Base64 en utilisant la base fournie. Elle gère également le padding pour s'assurer que la sortie est valide selon les spécifications Base64.

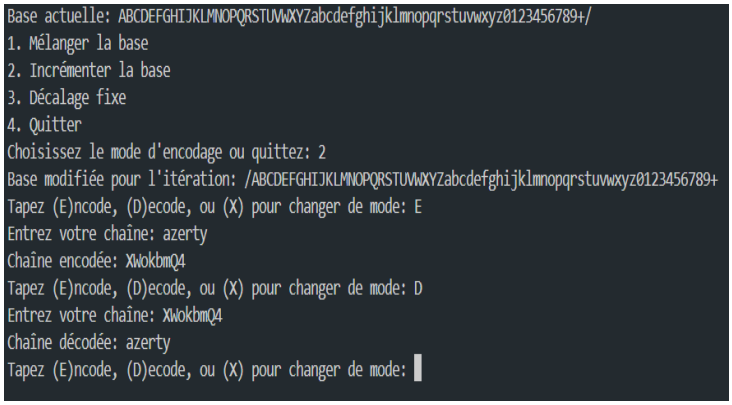
```
def encode(s_to_encode, base):
    binary = ''.join(format(char, '0>8b') for char in s_to_encode.encode())
    padded_binary = binary + '0' * ((6 - len(binary) % 6) % 6)
    indices = [int(padded_binary[i:i+6], 2) for i in range(0, len(padded_binary), 6)]
    return ''.join(base[index] for index in indices) + '=' * ((4 - len(s_to_encode) % 3) % 4)
```

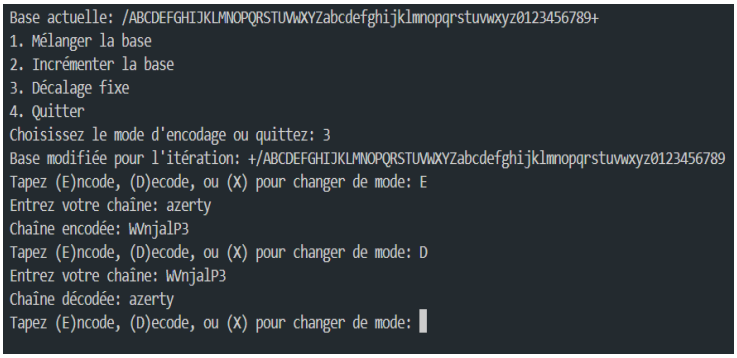
'decode' convertit une chaîne Base64 en sa représentation binaire utilisant un dictionnaire qui mappe chaque caractère à son indice dans la base, et reconstruit la chaîne originale à partir des octets décodés.

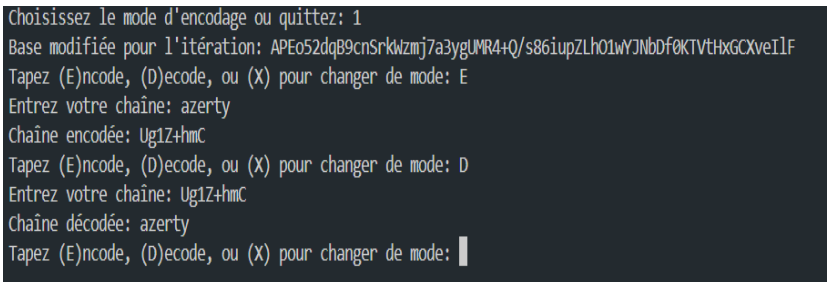
```
def decode(s_to_decode, base):
    base_dict = {char: i for i, char in enumerate(base)}
    encoded = s_to_decode.rstrip('=')
    binary = ''.join(format(base_dict[char], '06b') for char in encoded)
    byte_length = len(binary) - len(binary) % 8
    return bytes(int(binary[i:i+8], 2) for i in range(0, byte_length, 8)).decode('utf-8', errors='ignore')
```

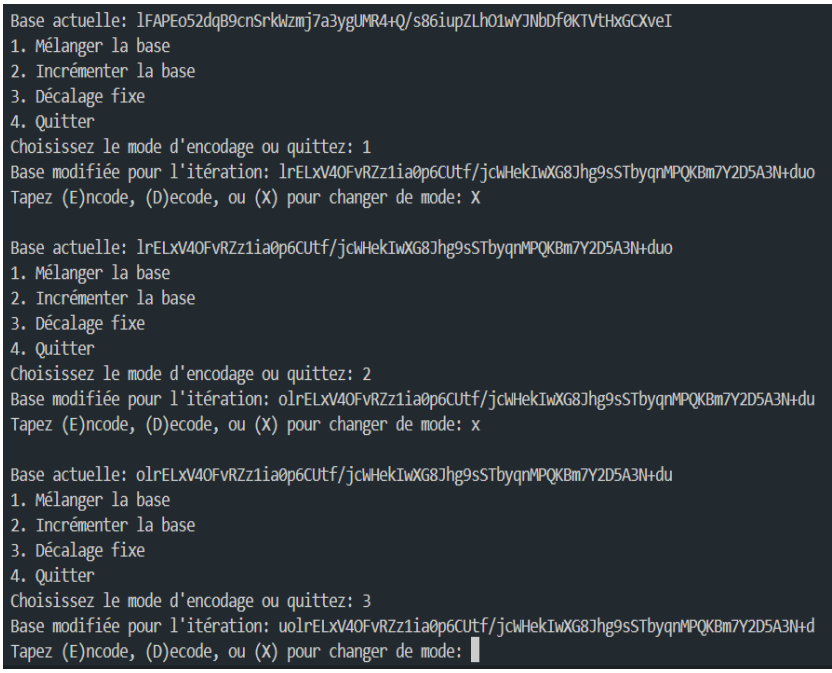
Utilisation











Conclusion

Le script offre une flexibilité permettant de manipuler la base64 de différentes manières avant l'encodage ou le décodage, offrant ainsi diverses méthodes de sécurisation des données par obfuscation simple.