TP 3 - Changement de signe d’un nombre binaire.

D’abord nous avons cherché à changer le signe d’un nombre binaire, mais un problème s’est posé à nous : Comment changer les bits un à un puis additionner 1 pour changer de signe ?
Nous avons créé un algorithme qui demande a l’utilisateur s’il souhaite additionner deux nombres binaires, ou s’il souhaite changer le signe d’un nombre.

Changement de signe.
Nous avons donc créé une fonction qui change le bit (première partie du changement de signe).

Ex : 11001101 --> 00110010

Le nombre est codé sous 1 octet (8bits). On demande le signe du nombre (1 pour le négatif et 0 pour le positif), puis le nombre passe dans la fonction qui additionne. Ici, le nombre inversé se voit additionner 00000001 pour compléter le changement de signe.

Ex : 11001101 --> 00110010 + 00000001 = 00110011 ce nombre est l’opposé de 11001101

On a rencontré un problème, si l’utilisateur entrait un nombre binaire contenant autre chose que des 1 ou des 0, le programme rendait un résultat faux. Nous avons alors créé une fonction qui permet de vérifier si l’utilisateur n’a pas entré de valeur inadéquate. Pour ce faire, on analyse chaque valeur de la liste entrée, et si une valeur est différente de 1 ou de 0, oubien si le nombre est suppérieur à 8 ou inférieur à 2 bits la fonction renvoie un booléen False.

Ex : Ici, l’utilisateur a entré un 2, qui n’est pas une valeur acceptable lorsqu’on code en binaire, alors le programme lui demande de recommencer.

```
Entrez le signe (0:positif / 1:négatif): 1
Entrez un chiffre en binaire pour avoir son inverse (7 char max): 211100
Vous n'avez pas rentré de données compatibles avec la fonction!

Voulez vous recommencer (y/n):
```

Nous avons été confronté à un autre problème, en effet si le nombre de bit était trop petit, le programme buguait. Nous avons donc réagi en créant une fonction qui adapte le nombre de bit au rangement dans un octet. si un nombre est par exemple codé sous 6 bits, il rajoute des 0 pour le coder dans un octet.

Ex : L’utilisateur rentre son chiffre, s’il est plus petit que 8 bits, le programme rajoute des 0.

```
def ranger(a,b):          #cette
    lenb=len(b)-1
    lena=len(a)-1
    if len(a)>len(b):
        la=list(a)
        lb=['0']*len(a)
        longueur=len(lb)-1
        while lenb>=0:       #per
            lb[longueur]=b[lenb]
            lenb-=1
            longueur-=1
    else :                   #même p
        lb=list(b)
        la=['0']*len(b)
        longueur=len(la)-1
        while lena>=0:
            la[longueur]=a[lena]
            lena-=1
            longueur-=1
    return la, lb
```