NSI 1ère - Données

Complément à deux - Travaux dirigés

qkzk

1. Tailles minimales et maximales

1. On encode des entiers sur 2 octets.
   a. Les entiers sont tous positifs. Quels sont les plus petits et plus grands entiers qu'on puisse encoder ?
   b. Les entiers sont encodés en complément à 2. Même question.
2. Dans un programme les entiers sont encodés en complément à deux. La taille utilisée a été perdue.
   On sait que `111010101` est un entier négatif.

   a. Quelle est la taille employée ?

   b. Quels sont les plus petits et plus grands entiers qu'on puisse encoder ?

2. Complément à deux sur un octet.

On encode les entiers en complément à 2 sur un octet.
Compléter le tableau suivant :

Lorsque c'est impossible, écrire une croix

Entier	Complément à 2 sur un octet
1	
127	
	`1111 1111`
-12	
-93	
101	
-139	
	`0101 1100`
	`1101 0011`

Réaliser les additions des éléments du tableau en complément à 2, deux lignes à la fois :

premier + second, second + troisième, troisième + quatrième.

Vérifier les résultats obtenus.

3. Complément à deux, un programme

Dans cet exercice nous allons écrire une fonction qui réalise le complément à deux en Python.

- La foncion `int` permet d'obtenir la représentation décimale d'un entier depuis n'importe quelle base $b \leq 36$

```
>>> int('10101', 2) # en binaire
21
>>> int("azerjsdkfjlqdkfj", 36) # les "chiffres" 0-9 et a-z
2428190859766979995068079
```

- La fonction `bin` donne la représentation binaire d'un entier naturel

```
>>> bin(120)
'0b1111000'
```

1. Écrire une fonction qui inverse un bit :

```
>>> inversion('0')
'1'
>>> inversion('1')
'0'
```

2. Python permet de transformer une chaîne de caractère en tableau avec la fonction `list`

```
>>> list("bonjour")
['b', 'o', 'n', 'j', 'o', 'u', 'r']
```

Écrire une fonction Python :

1. qui prend un entier positif (`int`),
2. le converti en binaire,
3. enlève `0b` au début,
4. Le converti en un tableau de chaînes de caractères.

Exemple :

```
>>> vers_tableau(123)
['1', '1', '1', '1', '0', '1', '1']
```

3. La méthode `join` des chaînes de caractères permet de convertir une liste de chaîne en une chaîne :

```
>>> '|'.join(['1', '3', '5'])
'1|3|5'
>>> ''.join(["b", "o", "n"])
'bon'
```

Utiliser les fonctions précédentes et la méthode `join` pour écrire le complément à 2 sur une taille donnée.