

TD 7 : Codage de l'information

Conversion hexadécimal - binaire

1. Écrire un programme *hex2dec* qui convertisse un nombre hexadécimal en décimal non signé.

NB : on pourra utiliser une liste de symboles éponyme qui contiendra les chiffres [0..F]. Ainsi la lettre F pourra prendre la valeur de son index soit 15.

Pour rappel : $F12_{16} = 15 \times 16^2 + 1 \times 16^1 + 2 \times 16^0 = 3\,858$

2. Donner les conversions de 1, A, a et A5 en décimal.
3. Donner la conversion de AZ et conclure.
4. Modifier votre programme comme précédemment pour gérer des erreurs éventuelles (bugs).

NB : on affichera « valeur hexadécimale incorrecte »

On souhaite réutiliser le code de programme *dec2bin* pour écrire un programme *hex2bin* qui convertisse un nombre hexadécimal en binaire. Pour cela, il « suffit » de récupérer la valeur obtenue par *hex2dec* et la passer à *dec2bin*.

Lorsqu'une tâche doit utiliser un bout de programme déjà existant ou doit être réalisées plusieurs fois par un programme, on peut l'isoler au sein d'une fonction. Une fonction est une « boîte noire » (ou blanche dans notre cas) qui peut prendre des valeurs (paramètres) en entrée et donne un résultat en sortie.

5. Dans le programme *hex2bin*, faire un copier/coller du code du programme *dec2bin*.
6. Remodeler (refactor) le code pour le transformer en fonction dont le prototype sera le suivant :

```
dec2bin(nombre : int, nbits : int) -> list
```

- nombre est le nombre décimal à convertir
- nbits est le nombre de bits
- la fonction renvoie la conversion sous forme d'une liste binaire (au lieu de l'afficher)

7. Tester la fonction en l'appelant depuis le corps principal (`## main ##`) du programme.
8. À la suite de cette fonction, faire un copier/coller du code du programme *hex2dec*.
9. Remodeler (refactor) le code pour le transformer en fonction dont le prototype sera le suivant :

```
hex2dec(hexa : str) -> int
```

- hexa est le nombre hexadécimal à convertir
- la fonction renvoie la conversion sous forme d'une liste binaire (au lieu de l'afficher)

10. Tester la fonction en l'appelant dans le corps principal (`## main ##`) du programme.
11. Enchaîner les appels successifs pour afficher la valeur binaire des nombres suivants : 1, A, a et A5.
12. Donner la conversion de AZ et conclure.

Traiter l'erreur en local a ses limites : il faut faire remonter l'erreur vers les couches supérieures. La dernière couche traitera en dernier l'erreur selon des spécifications bien précises.

13. Modifier les fonctions *hex2dec* et *dec2bin* pour qu'elles renvoient « None » au lieu d'afficher une erreur.

NB : L'objet Python None, exprime l'absence de valeur. Cet objet n'a aucune méthode.

14. Modifier le corps principal du programme pour tester la valeur de retour sur chaque fonction appelée. En cas d'erreur (valeur None), afficher le message d'erreur ad-hoc.

Le langage Python est auto documenté. Lorsque l'on tape une fonction dans l'IDE, ou que l'on demande explicitement de l'aide sur une fonction, sa documentation associée apparaît. Une fonction correctement écrite doit faire de même.

Pour cela, on utilise un docstring. Le docstring se place immédiatement après le prototype de la fonction et se déclare par """. Exemple :

```
def carre(nombre : int) -> int:
    """
        calcul le carré d'un nombre

        arguments:
            nombre - nombre à calculer

        retour :
            carré du nombre
    """
```

15. Créer un docstring pour chacune des fonctions du programme *hex2bin*.

Challenge #1

Écrivez en Python un programme *conv.py* permettant de convertir un nombre d'une base de départ vers une base d'arrivée (les bases allant de 2 à 16).

Le prototype de la fonction sera le suivant :

```
conversion(nombre : str, depart : int, arrivee : int) -> str
```

- nombre : nombre en base de départ à convertir
- depart : base de départ
- arrivee : base d'arrivée
- la fonction renvoie la valeur convertie sous forme de chaîne (str)

Les tests unitaires suivants devront être passés avec succès (ie : renvoyer les résultats indiqués en face de chaque appel) :

- | | |
|-------------------------------------|------------------------------------|
| • Conversion("10", 10, 10) == "10" | • conversion("A", 18, 2) == None |
| • conversion("10", 10, 2) == "1010" | • conversion("F", 12, 2) == None |
| • conversion("10", 10, 16) == "A" | • conversion("Z", 16, 2) == None |
| • conversion("A0", 16, 10) == "160" | • conversion("1.1", 10, 2) == None |
| • conversion("A", 16, 2) == "1010" | • conversion("a", 16, 2) == "1010" |