Programmation python partie : vers un véritable convertisseur

1) Se rendre dans le répertoire / commun / Travail / isn / et copier l'archive convertisseur.tar.gz
Aller ensuite dans l'espace personnel, répertoire isn et copier l'archive, puis la décompresser

2) Rentrer avec le navigateur de fichiers dans le répertoire nouvellement créé (convertisseur) , et ouvrir le fichier convertisseur.py avec geany, interpréter le programme avec F5.
Appeler si certaines lignes ne sont pas comprises.

3) Pour créer un programme plus évolué, il faut commencer par décomposer celui-ci en fonctions.
Une fonction est une suite d'instructions programmée, qui reçoit éventuellement un/plusieurs paramètre(e) en entrée et retourne une valeur en sortie. Ainsi, nous pourrons associer chaque conversion à une fonction.
Modifier le programme ainsi :
L'exécuter avec F5. Il ne se passe rien , si ce n'est une sortie correct avec le code 0 (pas d'erreurs)
Il faut donc utiliser la fonction, sinon, il ne se passera rien !
Ajouter en fin de programme :

```
17  toto = convBinDec("1101")
18  print(toto)
```

convertisseur.py

```
1   #/usr/bin/python3
2   # -*- coding: utf-8 -*-
3
4
5   def convBinDec(e) :
6       l=len(e)-1
7       n = 0
8       sortie = 0
9       i = l
10      while i != -1 :
11          if e[i]== "1":
12              sortie=sortie+2**n
13          n=n+1
14          i=i-1
15      return sortie
16
```

4) Prêt à continuer ? Avant de passer à la suite, on apporte encore une amélioration, on va déporter notre fonction dans un fichier en dehors du programme, de façon à pouvoir l'utiliser dans d'autres applications .
Avec Geany, créer un fichier nommé mesfonctions.py, dans lequel on va copier les lignes 5 à 15 (il faudra les couper du fichier convertisseur.py) . Sauvegarder.

Notre programme principal est un peut court maintenant :

```
1   #/usr/bin/python3
2   # -*- coding: utf-8 -*-
3
4
5
6   toto = convBinDec("1101")
7   print(toto)
8
```

D'ailleurs, quand on l'exécute , on se heurte à un flots d'insultes :

```
Traceback (most recent call last):
  File "convertisseur.py", line 6, in <module>
    toto = convBinDec("1101")
NameError: name 'convBinDec' is not defined
```

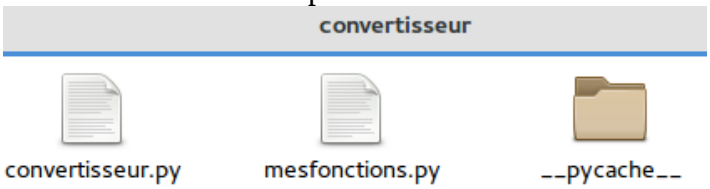
Et en plus, il a raison il ne peut pas trouver la fonction convBinDec, car elle est dans un autre fichier !

Pour corriger notre oubli, il suffit d'ajouter une ligne indiquant ou se trouve le fichier contenant nos fonctions .

```
1   #/usr/bin/python3
2   # -*- coding: utf-8 -*-
3
4   from mesfonctions import *
5
6   toto = convBinDec("1101")
7   print(toto)
8
```

YES !!!!!!!!!!

Le contenu du répertoire est maintenant :



5) Enfin, un début . On s'occupe maintenant de notre programme principal.
On commence par demander quelle conversion désire l'utilisateur .

Méthode proposée

initialiser une variable booléenne q à false
faire une boucle de type while sur q
demander une saisie (variable commande)
si commande égal q ou Q n sort du programme avec une message cool
si commande = bd ,on traite binaire décimal
si commande = db , on traite décimal binaire
sinon on donne la doc d'utilisation des commandes

la structure python :
if toto :
    action….
elif :
    action autre …

else :
    dernier choix

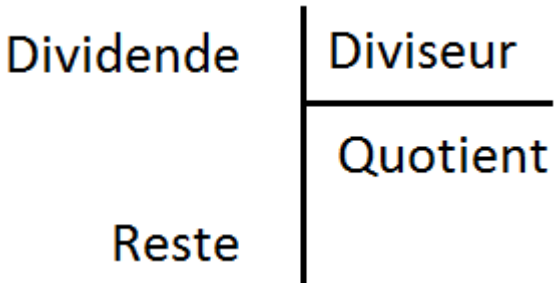
Une solution ici : http://bellevue-albi.entmip.fr/lectureFichiergw.do?ID_FICHIER=1506014934952

6) On modifie le programme pour :
pour qu'il utilise la fonction convBinDec et qu'il indique le message encours de développement pour l'autre conversion .

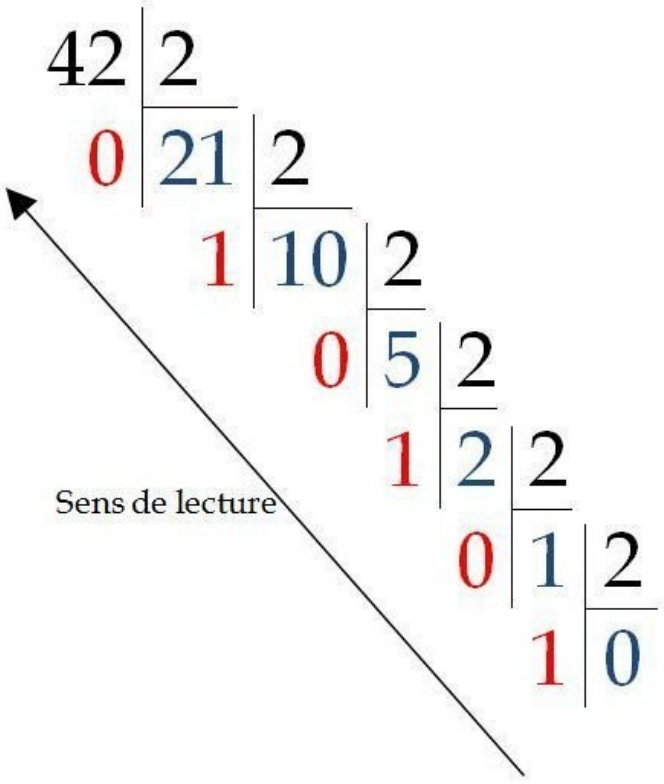
6) Ajoutons une fonction qui nous sorte un joli binaire à partir d'un décimal .

Donc , on prend le décimal :

On le divise par deux , on stocke le reste dans le binaire et on continue avec le quotient.



On stockera le binaire dans une **liste**
Une **liste** est un outil permettant de stocker plusieurs nombres, chaînes de caractères, etc.. dans un même objet.
En python, une liste est écrite de la manière suivante :
L=[1,2]
On peut tout aussi bien initialiser une liste vide :
L=[ ]
Le premier élément d'une liste sera d'indice 0, le second d'indice 1, etc. Pour sélectionner un élément on procède de la manière suivante :
L=[1,2,3,4]
print(L[0]) // Affiche 1
print(L[3]) // Affiche 4
La commande « L.append(40) » permet de rajouter l'élément 40 à la fin de la liste L.



Pour le **reste**, on utilisera le modulo , par exemple
reste = 13%2   // reste = 1
pour le quotient , quelque chose comme
quotient = (quotient -reste)/2 // quotient = 6

Annexes :

Solution au 5)

```python
#/usr/bin/python3
# -*- coding: utf-8 -*-

from mesfonctions import *

q = False

while not(q):

    commande=input("Entrer bd ou db , q pour quitter ")
    if commande == 'q' or commande == 'Q':
        q = True
    elif commande == 'bd':
        print ("conversion binaire -> decimal")
    elif commande == 'db':
        print ("conversion decimal -> binaire")
    else :
        print("les commandes sont : ")
```

Solution au 6)

```python
#/usr/bin/python3
# -*- coding: utf-8 -*-

from mesfonctions import *

q = False

while not(q):

    commande=input("Entrer bd ou db , q pour quitter ")
    if commande == 'q' or commande == 'Q':
        q = True
    elif commande == 'bd':
        a= input("Entrer un nombre binaire")
        toto = convBinDec(a)
        print(toto)
    elif commande == 'db':
        print ("Fonction en cours de développement")
    else :
        print("les commandes sont : ")
```