Un peu de binaire

Exemple de conversion de nombres simples décimal vers binaire et vice versa

Méthode systématique de conversion d'un entier N :

On divise N euclidiennement par 2, on obtient un quotient et un reste. Le reste est le premier chiffre de notre écriture binaire, puis on divise le quotient par 2, on a un nouveau quotient et comme reste le deuxième chiffre de l'écriture binaire. Etc.

A l'aide de la console de Pyscripter Se servir de python comme d'une calculatrice avec les opérateurs % et //

Algorithme

Traduction sous forme d'un petit programme (pour N=53) :

```
x=53
while x !=0 :
    print(x%2)
    x//2
```

Ça nous oblige à tourner la tête … ce n'est pas vraiment terrible… on pourrait utiliser une liste, mais ça vous verrez ça en détail avec M. Espinosa

Comment le changer pour qu'il demande le nombre à convertir ? (attention la fonction input renvois une chaine… il faudra donc utiliser la fonction int() pour convertir en entier.

Création d'une fonction

```
def monbinaire(nombredeci):
    x=nombredeci
    nombrebinaire=''
    while x !=0 :
        nombrebinaire=str(x%2)+nombrebinaire
        x=x//2
    return int(nombrebinaire)
```

Quelques remarques sur les variables et leur typage (fonctions int et str)

Notion de librairie

Quand on programme, pour se faciliter la vie on évite de réécrire 50 fois la même routine, c'est pour cela qu'on crée souvent des fonctions. Même si le langage python contient bien des instructions et autres fonctions intégrées, il est loin de contenir tout ce qu'il nous faut, donc on va devoir fabriquer nous-même nos propres fonctions ou exploiter celles qui ont déjà été écrite par d'autres programmeur. Ceux-ci très généreux ont créé ce qu'on appelle des librairies/ bibliothèques qui en contiennes chacune un sacré paquet.

Les librairies sont thématiques : il y en a pour faire des jeux vidéos, il y en a pour faire des mathématiques de haut niveau, d'autres pour faire de la création de sites web (Django) Quand on programme on vise l'efficacité et la rapidité donc même réinventer le fil à couper le beurre, c'est bien sympa mais on préfèrera éviter (à moins que vous ayez trouvé une manière de faire plus efficace).

Vous avez rencontré ici deux librairies turtle et random, ces bibliothèques contiennent moult fonctions qui peuvent être utile ou pas pour le programme en cours.

Niveau 1

Voyons voir comment est-ce qu'on fait pour les invoquer :

From turtle import * nous permet de rendre disponibles toutes les commandes du module turtle

From random import randint nous permet de rendre disponible une fonction (randint) du module random.

importer plusieurs librairies c'est prendre le risque de créer des doublons, ce qui serait perturbant pour la machine et le programmeur.

Pour circonvenir ce problème : on peut importer de manière un peu différente : *Import turtle*

Mais dans ce cas, à chaque fois qu'on utilisera une fonction de turtle il faudra le préciser

A la place de *forward(50)* il faudra utiliser *Turtle.forward(50)*

Bonus : On est d'accord, c'est vraiment lourd, mais ça pourrait être pire, imaginez que le nom de la librairie soit très long par exemple Masuperlibrairiedontjesuissuperfier, si on doit recopier à chaque fois le nom de la librairie ça va nous prendre du temps et notre code sera bien moins lisible, je pourrais racourcir ce nom en « Msl » et dans ce cas il me faudra écrire : *import Masuperlibrairiedontjesuissuperfier as Msl*

Niveau 2 (pour un autre jour)
Créer un package
Pour créer votre propre package, commencez par créer dans le même dossier que votre programme un dossier portant le nom de votre package. Dans notre exemple, nous le nommerons "meslibrairies".

Dans ce dossier, créons le fichier suivant: __init__.py, cela indique à python qu'il s'agit d'un package. Ce fichier peut être vide, seule sa présence est importante.

Ensuite créons un fichier toujours dans ce réportoire "meslibrairies" que nous nommerons par exemple "conversion.py"

Ainsi dans le dossier de votre projet: vous aurez vos fichier de programme avec une extension en .py et un répertoire "meslibrairies" dans lequel vous aurez un fichier __init__.py et "mesconversions.py"

Maintenant éditons le fichier monbinaire.py et créons une nouvelle fonction :

```
def monbinaire(nombredeci):
    x=nombredeci
    nombrebinaire=''
    while x !=0 :
        nombrebinaire=str(x%2)+nombrebinaire
        x=x//2
    return int(nombrebinaire)
```