Projet Janvier 2020: Convertisseur Binaire

15 Novembre 2019 Nathan Sakkriou

Structure du programme

Nous avons structuré notre code de la façon suivante :

- La première partie correspond a notre menu. L'utilisateur doit chosir la fonction qu'il va appeler, (Décimal vers Binaire ou Binaire vers Décimal). Tout en gérant d'éventuelles erreurs de frappes, sachant que la sélection de la fonction se fait en entrant la valeur 1 ou 0.

- La seconde partie, sert à définir nos deux fonctions qui exécuteront tous les calculs nécessaires.

- Et pour finir la dernière partie est celle qui en fonction du choix utilisateur exécutera la fonction liée à la valeur saisie

---

Fonctionnement géneral

1° La Gestion des erreurs et des exceptions

Avant tout nous avons, pour chaque saisie utilisateur, fait attention qu'il rentre l'une des possibilités que nous lui proposions, en faisant passer à la valeur saisie une suite de tests.

projet.py

```
while True:
    try:
        menu = int(input("Binaire: 0, Décimal: 1 --> : "))
        if menu != 0 and menu != 1:
            print("Il faut choisir entre 0 et 1, et pas autre chose =)")
        else:
            break
    except ValueError:
        print("Vous avez entré une (des) lettre(s), il faut choisir entre 0, ou 1")
```

Par exemple pour celui-ci, étant donné qu'il faut donner soit 1 ou 0, on verifie :

- Que la valeur entrée n'est pas différente de 1 ou 0, si la la valeur est différente de 1 ou 0, alors on retourne à l'utilisateur un message (d'erreur) et on recommence la boucle à la saisie de la valeur initiale

- Dans un second temps on vérifie également que la valeur entrée est bien de type int (donc un nombre). Si ce n'est pas le cas, python retourne une erreur, nommé ValueError, et donc si le programme détecte cette erreur il retourne un message à l'utilisateur et recommence au niveau de la saisie.

Pour sortir de cette boucle, il faut simplement que la valeur saisie soit de type int et qu'elle soit 1 ou 0.

Afin d'effectuer ces tests, nous avons d'abord mis en place un boucle infinie (While True) pour gérer toutes les erreurs possibles, ensuite nous avons dans cette boucle l'instruction Try: qui permet de gérer les erreurs/exception de notre code. Et après la après la saisie et la vérification de la valeur on trouve à la suite du else:, l'instruction break: Celle ci permet de couper la boucle While true et donc nous permettre de continuer le programme. Et pour finir on y retrouve la ligne "except ValueError", etant donné que nous avons déjà explicité cette erreur, il reste le except, celui ci permet de gérer l'erreur ValueError et c'est grâce à lui que la saisie peut recommencer quand un non int est rentré dans la demande de saisie utilisateur.

On va retrouver ce mécanisme à chaque saisie utilisateur du programme, et donc le principe serra le même à chaque fois

2° Les Fonctions

Dans notre programme on retrouve a deux reprise ce genre de ligne :

```
projet.py
31  #---------------------------------
32  def binairetodeci():
```

Celle-ci permet de créer une fonction. Une fonction est une suite d'instructions que l'on peut appeler n'importe quand dans le programme et qui ne s'exécutera jamais sans cela.

Def ....(): Permet cela

3° Les Listes

Une liste permet de stocker plusieurs valeurs indépendamment (sans devoir les additionners par exemple), ce qui est parfait lorsque l'on veut stocker un binaire.

```
projet.py
40
41      for loop in range(nbBinaire):
42          print("Entrez votre binaire bit par bit")
43          while True:
44              try:
45                  binaire = int(input())
46                  if binaire != 0 and binaire != 1:
47                      print("Il faut choisir entre 0 et 1")
48                  if len(str(binaire)) != 1:  # Pas sur que ça fonctionne parfaitement
49                      print("Il ne faut qu'un bit")
50                  else:
51                      break
52              except ValueError:
53                  print("Vous avez entré une/des lettre/s, il faut choisir entre 0, ou 1")
54
55          lstbinaire.append(binaire)
56
57      lstbinaire.reverse()
58      print(lstbinaire)
```

On voit ici mise à part la verification certaine commande utilisant des listes:

- lstbinaire.append(binaire)

- lstbinaire.reverse()

La première permet d'ajouter la valeur que possède la variable binaire dans la liste nommée lstbinaire, et la seconde permet d'inverse l'ordre de toutes les valeurs.

Dans cette fonction étant donnée que nous travaillons sur un binaire, les calculs se feront tout le temps dans des listes

4° Les autres

Il y a certaines instructions qu'on ne pouvait pas classer dans les autres catégories :

- len() : permet de retourner la longueur du paramètre, le nombre de caractères qu'il possède

- str() permet de transformer le paramètre en string (chaine de caractères)

- % : (modulo) permets de récupérer le reste dans une division euclidienne