Informatique tronc commun TP 7

16 décembre 2013

Note : après la séance, vous devez rédiger un compte-rendu de TP (fichier python + compte-rendu) et le faire parvenir à votre enseignant, au plus tard une semaine après la séance de TP.

Le nom de votre fichier python sera **impérativement** forme `dupont_jean_tp07.py`, où `dupont` est à remplacer par votre nom et `jean` par votre prénom, les deux étant en **minuscules** (même la première lettre) et **sans caractère accentué**.

1 Représentation des flottants

Les exemples suivants peuvent surprendre :

```
a = 1e20
b = -1e20
c = 1.0
(a + b) + c
a + (b + c)

rad = 2 ** .5
rad * rad - 2

.1 + .2
.1 + .2 == .3
.5 + .25
.5 + .25 == .75
```

1. Rappeler la norme de représentation des flottants, et expliquer notamment les différences de résultats des deux dernières comparaisons.
2. Confirmer ces explications en utilisant la fonction `Decimal` du module `decimal`, qui permet de donner la valeur exacte du représentant du nombre représenté, et la fonction `repr` permettant d'afficher ce représentant.
3. La méthode `hex()` permet quant à elle de proposer une représentation hexadécimal des nombres, c'est-à-dire une représentation binaire où les bits sont regroupés quatre par quatre en des chiffres de `0` à `e`.

1.1 Taille de la mantisse

Selon la norme de réprésentation des flottants, la mantisse est composée de 53 bits, le premier étant toujours `1`. On souhaite vérifier que la représentation des flottants en Python respecte cette norme. On note m le nombre 0.5.

1. Écrire la représentation de $1 + m$, de $1 + m/2$, de $1 + m/4$.
2. Pour connaître le nombre de bits de la mantisse, il suffit donc de compter le nombre de division par 2 qu'il est possible de faire avant que le bit représentant $m/2^\alpha$ ne soit hors de la mantisse. Écrire cet algorithme et le mettre en œuvre en Python.

1.2 Le rendu de monnaie

Dans le cadre de la programmation d'une caisse automatique, on souhaite concevoir un algorithme de rendu de monnaie. La caisse dispose d'un nombre supposé illimité de billets de 20€, 10€, 5€ et de pièces de 2€, 1€, 0.50€, 0.20€, 0.10€, 0.05€, 0.02€ et 0.01€.

1. Pour un montant d'achat et une somme donnée par le client, proposer un algorithme permettant de rendre la monnaie au client. Cet algorithme détaillera les coupures utilisées pour rendre la monnaie et cherchera à minimiser le nombre de coupures utilisées.
2. Implanter cet algorithme en Python.
3. Tester l'algorithme sur plusieurs valeurs. Par exemple un achat de 50€ payé 100€. Ou un achat de 75.25€ payé 100€. Est-on prêt à commercialiser la caisse automatique ?

2 Représentation des entiers

2.1 Écriture d'un entier en base 2

1. Écrire une fonction `décimal2binaire` qui transforme un entier positif n en une chaîne de caractère représentant n en base 2.
2. Écrire une fonction `binaire2décimal` qui transforme une chaîne de caractères s composée de `'0'` et de `'1'` en l'entier correspondant.
3. Tester les fonctions précédentes, et comparer avec les fonctions :

```
bin(2013)
# '0b11111011101'
int('11111011101',2)
# 2013
```

2.2 Un grand entier

Le 48-ième nombre premier de Mersenne, récemment découvert, est $2^{57885161} - 1$.

1. Comment s'écrit ce chiffre en base 2 ?
2. Est-il représentable comme flottant ? L'expérimenter.
3. Quel est l'ordre de grandeur du nombre de chiffres de son écriture en base 10 ?
4. Comment déterminer le nombre de chiffres de son écriture en base 10 ? Sachant qu'une division en base 10 a un « coût » proportionnel au nombre de chiffres, estimer le temps de calcul nécessaire pour connaître le nombre exact de chiffre dans l'écriture en base 10 du 48-ième nombre de Mersenne.

2.3 Écriture bibi-binaire

Parce que seize peut s'écrire 2^{2^2}, Boby Lapointe estimait qu'on pourrait parler de « Bi-Binaire » pour la base 4, et de « BiBi-Binaire » pour la base 16, terme qu'il abrège en « bibi ».

Il inventa la notation et la prononciation de seize chiffres. À l'aide de quatre consonnes et de quatre voyelles, on obtient les seize combinaisons nécessaires :

décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
bibi	HO	HA	HE	HI	BO	BA	BE	BI	KO	KA	KE	KI	DO	DA	DE	DI

Pour définir un nombre, il suffit d'énumérer les chiffres (hexadécimaux) qui le composent.

Exemple : en Bibi, le nombre 2013 (en base décimale), qui se traduit, en hexadécimal, par 7dd, est appelé BIDADA.

1. Écrire une fonction `décimal2bibi` qui donne l'écriture en bibi d'un entier.
2. Proposer la fonction inverse `bibi2décimal`, en supposant que le nombre écrit en bibi est bien formé.