

CHAPITRE 6 : CALCUL AVEC DES NOMBRES DÉCIMAUX ET FRACTIONNAIRES

I. Calculs avec des nombres décimaux avec le module decimal.

1.1 Mise en situation.

```
>>> 0.5*0.5
0.25
>>> 0.7*0.7
0.48999999999999994
>>> 0.1+0.2
0.30000000000000004
>>> 0.7*0.7==0.49
False
>>> [0.5**n for n in range(1,8)]
[0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078125]
>>> [0.1**n for n in range(1,5)]
[0.1, 0.010000000000000002, 0.0010000000000000002, 0.00010000000000000002]
>>>
```

- Les "erreurs" ci-dessus ne sont que des conséquences de la représentation en base 2 des flottants : même un nombre aussi inoffensif que $0.1=1/(2*5)$ n'est pas représenté de façon exacte en mémoire (contrairement à $0.5=2^{-1}$).
- Le module decimal permet d'effectuer des calculs exacts sur les nombres décimaux, dans les limites d'une précision fixée par l'utilisateur (mais par défaut égale à 28 chiffres significatifs).
- Les "nombres décimaux" (au sens du module decimal) sont obtenus par application du constructeur Decimal (noter le D majuscule) appliqué à un entier ou à une chaîne de caractères (elle-même une représentation d'un flottant, ce qui permet à l'utilisateur de former des valeurs décimales avec une précision donnée).

```
>>> from decimal import *
>>> [Decimal('0.5')**n for n in range(1,5)]
[Decimal('0.5'), Decimal('0.25'), Decimal('0.125'), Decimal('0.0625')]
>>> [Decimal('0.1')**n for n in range(1,5)]
[Decimal('0.1'), Decimal('0.01'), Decimal('0.001'), Decimal('0.0001')]
>>> Decimal('0.1')+Decimal('0.2')
Decimal('0.3')
>>>
```

1.2 Précision des calculs.

Tout cela n'est peut-être pas très impressionnant, alors on va augmenter la précision. L'instruction `getcontext().prec = n` fixe la précision à n chiffres significatifs.

```
>>> getcontext().prec=50 #augmente la précision à 50 chiffres
>>> Decimal(1)/Decimal(19)
Decimal('0.052631578947368421052631578947368421052631578947368')
>>> Decimal(3).sqrt() #racine de 3 avec 50 chiffres significatifs
Decimal('1.7320508075688772935274463415058723669428052538104')
>>> Decimal(1).exp() # le nombre e avec 50 chiffres significatifs
```

```
Decimal('2.7182818284590452353602874713526624977572470937000')
>>> Decimal(2).ln() # ln 2 avec 50 chiffres significatifs.
Decimal('0.69314718055994530941723212145817656807550013436026')
>>>
```

Le nombre de fonctions mathématiques qui sont compatibles avec ce module est essentiellement limité au logarithme, à l'exponentielle et à la racine carrée.

1.3 Affichage d'un nombre décimal :

```
>>> from decimal import *
>>> a=Decimal('0.125')
>>> a
Decimal('0.125')
>>> print(a)
0.125
>>>
```

```
>>> a=[print(Decimal('0.1')**n) for n in range(5)]
1
0.1
0.01
0.001
0.0001
>>>
```

1.4 Utilisation du module *decimal* dans un programme.

```
from decimal import *

a=input('Entrer un nombre décimal : ') #a est un str
b=input('Entrer un nombre décimal : ') #b est un str

print('La somme est =',Decimal(a)+Decimal(b)) # Decimal(a+b) génère une erreur
```

```
>>>
Entrer un nombre décimal : 0.1
Entrer un nombre décimal : 0.2
La somme est = 0.3
>>>
```

II. Calculs avec des nombres fractionnaires avec le module *fractions*.

2.1 Exemples.

Le module *fractions* permet d'effectuer des calculs exacts sur les nombres rationnels. Un nombre rationnel s'obtient par le constructeur *Fraction* qui prend en argument deux entiers (le numérateur, puis le dénominateur qui par défaut vaut 1) ou une chaîne (par exemple '12/17') ou un flottant (sous forme décimale).

```
>>> from fractions import * # importe le module fractions
>>> Fraction(1,2) + Fraction(1,3) #Fraction(1,2) peut s'écrire aussi Fraction('0.5')
```

```
Fraction(5, 6)
>>> float(_)
0.8333333333333334 #convertit ce résultat en flottant
>>> from math import pi # importe la valeur _ depuis le module math
>>> Fraction(pi) # approximation rationnelle de pi
Fraction(884279719003555, 281474976710656)
>>> Fraction(pi).numerator #extrait le numérateur
884279719003555
>>> Fraction(pi).denominator #extrait le dénominateur
281474976710656
```

2.2 Affichage d'un nombre rationnel :

```
>>> from fractions import *
>>> a=Fraction('5/49')
>>> a
Fraction(5,49)
>>> print(a)
5/49
>>>
```

2.3 Utilisation du module *fractions* dans un programme.

```
from fractions import *

a=input('Entrer un nombre rationnel : ') #a est un str
b=input('Entrer un nombre rationnel : ') #b est un str

print(a,'+',b,'=',Fraction(a)+Fraction(b)) #Fraction(a+b) génère une erreur
```

```
>>>
Entrer un nombre rationnel : 1/2
Entrer un nombre rationnel : 2/3
1/2
2/3
7/6
>>>
```