Module decimath

Version 1.0.5

pour python™

Python pédagogique pour les mathématiques

Sommaire

© Franck CHEVRIER, 2019-2020

Me contacter :

franck.jean.nic.chevrier@ac-strasbourg.fr

1. Pourquoi ce module ?

Ce module **n'est pas** un module destiné à un développement expert, ni un module de calcul formel, ni un module optimisant les temps de calcul.

Ce module **est** destiné à l'enseignement de l'algorithmique en langage Python dans un cours de mathématiques en lycée. C'est un outil se voulant pédagogique, permettant de contourner certains problèmes de calculs ou de tests sur les float, qui apparaissent lors de la mise en œuvre d'algorithmes mathématiques au Lycée. Il a pour but de permettre au professeur de concentrer son enseignement sur la compréhension des algorithmes (calculs de discriminants, de distances entre deux points, recherches dichotomiques, balayages…) sans s'attarder sur les erreurs de calculs et de tests liées au codage du type float en langage Python.

```
>>> a=0.1

>>> a*3
0.30000000000000004
```

```
>>> from decimath import*

>>> a=decimal(0.1)

>>> a*3
0.3
```

Comparaison de calculs effectués en type **float** et en type **decimal**

Par exemple, les raisons pour lesquelles un calcul tel que 0.1+0.1+0.1 n'a pas pour résultat 0.3 en type float serait à développer en cours d'informatique, mais éloignerait de la visée pédagogique d'un cours de mathématique.

Ce module définit un nouveau type d'objet, nommé decimal, qui stocke :

- une fraction décimale du type $\frac{a}{10^p}$ où a et p sont de type int avec p positif ou nul.
- une information booléenne qui indique si la valeur est exacte ou non. (1)
- une éventuelle propriété sur une puissance exacte connue. (2)
- une éventuelle propriété sur un produit connu. (3)

Il existe d'autres bibliothèques générant des types d'objet approchant, beaucoup plus puissantes et complètes, mais les affichages console qu'ils génèrent ne sont pas satisfaisant pédagogiquement (voir encadré ci-contre), or c'est bien cet aspect qui est visé ici.

```
>>> from decimal import*

>>> a=Decimal(0.4)

>>> a
Decimal('0.400000000000000222044604925031308084726333618164062
5')
```

```
>>> from decimath import*

>>> a=decimal(0.4)

>>> a
0.4
```

Comparaison de saisies avec le module **decimal** et avec le module **decimath**

(1) Cette information est obtenue, lors des calculs, à l'aide de tests sur les chiffres de la décomposition décimale du nombre.
(2) Si d a été généré comme racine $n^{\text{ème}}$ de a, alors le calcul de sa puissance $n^{\text{ème}}$ fournira bien la valeur a, avec ses éventuelles propriétés.
(3) Si d a été généré comme le quotient de a par b, alors le calcul du produit $d \times b$ fournira bien la valeur a, avec ses propriétés.

2. Exemples d'utilisation

L'import du module se fait à l'aide des syntaxes usuelles, et en particulier « from decimath import* » permet de s'affranchir du suffixe « decimath » pour les appels. Il convient au préalable de s'assurer que le fichier « decimath.py » est à un emplacement accessible pour la console d'exécution (shell).
Attention : Si l'import du module math avec la syntaxe « from math import* » est nécessaire, celui-ci doit être effectué avant l'import du module decimal, car le module reconstruit certaines fonctions.

2.1 Racines d'un polynôme du 2nd degré

On donne ci-contre des fonctions de calculs de discriminant et de la liste des racines d'un polynôme du 2nd degré.

Avec le type float, le test de l'égalité $\Delta = 0$ peut ne pas être mathématiquement correct. Ci-dessous figure un exemple où le discriminant est nul, mais où les fonctions utilisées en type float renvoient respectivement un discriminant strictement positif et deux racines pour le polynôme correspondant. Le passage au type decimal permet de rétablir les résultats mathématiques corrects attendus.

```
from math import*
from decimath import*

def delta(a,b,c):
    return b**2-4*a*c

def racines(a,b,c):
    d=delta(a,b,c)
    if d==0:
        return [-b/(2*a)]
    if d>0:
        return [(-b-sqrt(d))/(2*a),(-b+sqrt(d))/(2*a)]
    return []
```

Fonctions de calcul du discriminant et des racines d'un polynôme du 2nd degré

```
>>> a=0.005 ; b=0.1 ; c=0.5

>>> delta(a,b,c)
1.734723475976807e-18

>>> racines(a,b,c)
[-10.000000131708903, -9.999999868291098]
```

```
>>> a=decimal(0.005) ; b=decimal(0.1) ; c=decimal(0.5)

>>> delta(a,b,c)
0

>>> racines(a,b,c)
[-10]
```

Comparaison d'appels des fonctions effectués en type **float** et en type **decimal**

2.2 Egalité de Pythagore

Le test de l'égalité de Pythagore sur des variables de type float peut ne pas s'avérer concluant.
Le problème est résolu avec le type décimal, pour des saisies de nombres décimaux voire même pour des racines carrées.

```
from decimath import*

def Pythagore(a,b,c):
    return c**2==a**2+b**2
```

Fonction de vérification de l'égalité de Pythagore

```
>>> a=1.5 ; b=0.8 ; c=1.7

>>> Pythagore(a,b,c)
False
```

```
>>> a=decimal(1.5) ; b=decimal(0.8) ; c=decimal(1.7)

>>> Pythagore(a,b,c)
True
```

```
>>> a=sqrt(3) ; b=sqrt(4) ; c=sqrt(7)

>>> Pythagore(a,b,c)
False
```

```
>>> a=sqrt(decimal(3)) ; b=sqrt(decimal(4)) ; c=sqrt(decimal(7))

>>> Pythagore(a,b,c)
True
```

Comparaisons d'appels de la fonction effectués en type **float** et en type **decimal**

2.3 Balayage

Le type decimal permet également de résoudre les problèmes d'affichages rencontrés pour la recherche d'encadrement de solutions d'équations par balayage. A noter également que la puissance cinquième de $\sqrt[5]{2}$ ne donne pas 2 en type float, alors que les propriétés mémorisées en type decimal permettent de rétablir la valeur exacte.

```
from decimath import*

def f(x):
    return x**5

def balayage(f,x,pas):
    while f(x)<2:
        x=x+pas
    return x-pas,x
```

Fonctions pour la recherche par balayage d'un encadrement de $\sqrt[5]{2}$

```
>>> balayage(f,0,0.1)
(1.0999999999999999, 1.2)

>>> balayage(f,0,0.01)
(1.1400000000000008, 1.1500000000000008)
```

```
>>> balayage(f,0,decimal(0.1))
(1.1, 1.2)

>>> balayage(f,0,decimal(0.01))
(1.14, 1.15)
```

Comparaisons d'appels de la fonction effectués en type **float** et en type **decimal**

```
>>> a=2**(1/5)

>>> a
1.148698354997035

>>> a**5
2.000000000000001
```

```
>>> a=sqrt(decimal(2),5)

>>> a
1.148698354997035

>>> a**5
2
```

Comparaisons de saisies effectuées en type **float** et en type **decimal**

3. Accès aux propriétés d'une variable de type decimal

Les appels à la méthode prop() ne sont pas nécessaires lorsqu'on utilise le module decimath, mais permettent de comprendre la méthode de mémorisation et d'hérédité des propriétés des nombres de type decimal.

```
>>> a=decimal(80.035)
>>> a
80.035

>>> a.fdec()

80035
_____
  3
10

>>> a.exact
True

>>> a.prop()
a = 80.035
Cette valeur de a est exacte

>>> b=sqrt(a) ; c=a*b ; d=a/3 ; e=sqrt(b,3) ; f=sqrt(d)

>>> b.prop()
b = 8.946228255527578
Cette valeur de b est approchée.
La valeur exacte de b a pour caractéristique(s):
b**2 = 80.035 (exact)

>>> c.prop()
c = 716.0113784311497
Cette valeur de c est approchée.
La valeur exacte de c a pour caractéristique(s):
c**2 = 512672.294042875 (exact)

>>> d.prop()
d = 26.67833333333333
Cette valeur de d est approchée.
La valeur exacte de d a pour caractéristique(s):
d * 3 = 80.035  (exact)

>>> e.prop()
e = 2.0759329626055702
Cette valeur de e est approchée.
La valeur exacte de e a pour caractéristique(s):
e**3 = 8.946228255527578 (approché)
e**6 = 80.035 (exact)
e**2 = 4.3094976652323446 (approché)

>>> f.prop()
f = 5.165107291560683
Cette valeur de f est approchée.
La valeur exacte de f a pour caractéristique(s):
f**2 = 26.67833333333333 (approché)
f * 1.7320508075688772 = 8.946228255527578  (approché)
>>> v=sqrt(decimal(3)) ; g=(f*v)

>>> g==b
True

>>> e.formel()
((80.035)**(1/2))**(1/3)
(80.035)**(1/6)
((80.035)**(1/3))**(1/2)
```

Génération du nombre $a = 80{,}035$

Ecriture sous forme de fraction décimale $a = \frac{80\,035}{10^3}$

Accès aux propriétés de a :
La valeur décimale affichée est une valeur exacte.

Génération des nombres :
$b = \sqrt{a}$ $\quad e = \sqrt[3]{b}$
$c = ab = a\sqrt{a}$ $\quad f = \sqrt{d}$
$d = \frac{a}{3}$

Propriétés de b :
La valeur décimale affichée est une valeur approchée.
La propriété $b^2 = a$ est mémorisée

Propriétés de c :
La valeur décimale affichée est une valeur approchée.
La propriété $c^2 = \left(a\sqrt{a}\right)^2 = a^3$ est mémorisée.

Propriétés de d :
La valeur décimale affichée est une valeur approchée.
La propriété $d \times 3 = a$ est mémorisée.

Propriétés de e :
La valeur décimale affichée est une valeur approchée.
La propriété $e^3 = \left(\sqrt[3]{b}\right)^3 = b$ est mémorisée.
Les propriétés du nombre b ont généré des propriétés pour e :
$e^6 = b^2 = a$
$e^2 = \left(\sqrt[3]{b}\right)^2 = \sqrt[3]{b^2} = \sqrt[3]{a}$

Propriétés de f :
La valeur décimale affichée est une valeur approchée.
La propriété $f^2 = d$ est mémorisée.
Les propriétés du nombre d ont généré une propriété pour f :
$f \times \sqrt{3} = \sqrt{d} \times \sqrt{3} = \sqrt{\frac{a}{3} \times 3} = \sqrt{a} = b$

Les propriétés mémorisées sont utilisées si les calculs saisis le permettent : $f \times \sqrt{3} = b$

On peut accéder à des écritures « formelles » des variables :
$e = \left((a)^{1/2}\right)^{1/3} = (a)^{1/6} = \left((a)^{1/3}\right)^{1/2}$

4. Syntaxes

	Syntaxe	type des variables appelées	Résultat		Remarques
			type de retour	Description	
Conversions	**decimal(v)**	v : **int** ou **float**	**decimal**	Renvoie la valeur décimale correspondant à v.	
	str(a)	a : **decimal**	**str**	Renvoie une chaîne de caractère de la valeur (exacte ou approchée) de a sous sa forme décimale.	
	int(a)	a : **decimal**	**int**	Renvoie une valeur correspondant à la partie entière de a.	
	float(a)	a : **decimal**	**Float**	Renvoie une valeur float correspondant à a.	
Propriétés	**a** (saisie console)	a : **decimal**	*aucun*	chaine de caractère de la valeur (exacte ou approchée) de a sous sa forme décimale.	
	a.exact	a : **decimal**	**bool**	Indique si la valeur affichée est la valeur exacte de a	a peut posséder des propriétés particulières (racine ou quotient particulier) et sa valeur décimale affichée peut être approchée.
	a.prop()	a : **decimal**	*aucun*	Affichage en console : • valeur décimale de a • indique si cette valeur est exacte • indique les puissances exactes connues de a (cas où a est généré comme racine n-ème d'une valeur) • indique les produits exacts de a connus (cas où a est généré comme quotient de valeurs)	Les propriétés sont héréditaires si on compose les opérations puissances, racines, produits ou quotients.
	~a ou **a.formel()**	a : **decimal**	*aucun*	Affichage en console de notations formelles de a.	
	a.fdec()	a : **decimal**	*aucun*(*)	Affiche en console : fraction décimale (exacte ou approchée) correspondant à a	Arguments facultatifs : • affiche(=True) pour affichage en console • (*) renvoie(=False) pour renvoi de la chaine de caractères
Tests	**a == b**	a ou b : **decimal** (4)	**bool**	Teste si a=b	
	a < b **a > b** **a <= b** **a >= b**	a ou b : **decimal** (4)	**bool**	Teste si a<b ; a>b ; a≤b ; a≥b	
Opérations élémentaires	**a + b**	a ou b : **decimal** (4)	**decimal**	Renvoie la somme de a et b	
	a * b	a ou b : **decimal** (4)	**decimal**	Renvoie le produit de a par b	
	a - b	a ou b : **decimal** (4)	**decimal**	Renvoie la différence entre a et b	
	- a	a : **decimal**	**decimal**	Renvoie l'opposé de a	
	+ a	a : **decimal**	**decimal**	Renvoie a	
	a / b	a ou b : **decimal** (4)	**decimal**	Renvoie le quotient de a par b	
	a ** n	a : **decimal** n : **int** (positif)	**decimal**	Renvoie la puissance $n^{ème}$ de a	
	sqrt(a)	a : **decimal**	**decimal**	Renvoie la racine carrée de a	
	sqrt(a,n)	a : **decimal** n : **int** (positif)	**decimal**	Renvoie la racine $n^{ème}$ de a	

(4) si une des variables est de type decimal et pas l'autre, cette dernière sera (si possible) automatiquement convertie en type decimal pour le calcul