Introduction à Sage-Python

Aurélien Greuet
Université de Versailles
`http://lmv.math.cnrs.fr/annuaire/aurelien-greuet/`

1 Python ? Sage ? ! Calcul formel ? ! ?

1.1 Python

Python est un langage de programmation *interprété* (contrairement aux langages *compilés*) multi-plateforme et libre. Il sert à faire plein de choses. Par exemple la NASA l'utilise et il sert aussi à coder Sage. On va plutôt s'intéresser à Sage donc on ne s'étend pas trop sur Python, le lecteur intéressé pourra trouver plein d'informations sur le web, en particulier sur le site officiel `http://www.python.org/`.

1.2 Sage

Sage est un logiciel *libre* de calcul formel initialement développé par William Stein, un enseignant-chercheur américain, en 2005. Il est disponible pour Windows, Mac, Linux, en live CD ou même utilisable directement en ligne sur `http://www.sagemath.org/`.

Sage est écrit en Python et s'utilise en écrivant des commandes qui sont interprétées par Python. Concrètement, cela signifie que tout ce qu'on peut faire en Python peut être fait en Sage. Inversement, la plupart des commandes de base de Sage restent valable en Python. On peut donc voir Sage comme Python à qui on a rajouté des fonctionnalités de *calcul formel*.

Sage utilise parfois d'autres logiciels libres de calcul formel, mais de manière complètement transparente pour l'utilisateur. Cela lui permet d'être relativement efficace dans plusieurs domaines.

1.3 Calcul formel

Mais qu'est-ce que le calcul formel ? En général on parle de calcul *formel* par opposition au calcul *numérique*. En calcul *numérique*, seuls les nombres entiers d'une douzaine de chiffres peuvent être manipulés de manière exacte. Au delà ou pour des nombres qui ne sont pas entiers, des approximations sont faites, ce qui peut entraîner des erreurs. Au contraire en calcul formel, les calculs sont effectués sur des formules, de façon algébrique, en manipulant formellement des symboles et en faisant des transformations exactes.

2 Prise en main

Nous allons donner une liste de fonctions ou opérations avec un descriptif très court ou simplement des commandes tapées dans Sage, suivies de quelques questions qu'on peut être amené à se poser (ce ne sont pas les seules, n'hésitez pas à les partager si vous en avez d'autres). L'objectif est que vous manipuliez ces fonctions pour comprendre par vous-même leur fonctionnement et que vous soyez capable de vous débrouiller seul face à un problème en consultant l'aide, une documentation ou simplement en « bidouillant ».

Avant de commencer, quelques petites choses à savoir :

– lorsqu'on tape `<Entrée>`, on passe à la ligne. Pour évaluer l'expression courante, on peut faire `<Maj>+<Entrée>` ;
– complétion automatique : si on tape le début d'un mot suivi de le touche `<Tab>`, Sage affiche toutes les commandes, fonctions ou constantes commençant par ce début de mot ;
– aide en ligne : on peut accéder au manuel de toute commande, fonction ou constante en la faisant suivre d'un point d'interrogation ;
– si l'aide en ligne ne suffit pas, on peut consulter l'excellent « Calcul mathématique avec Sage », dit « le Sagebook » ;
– si le Sagebook ne suffit pas, il reste internet et ses moteurs de recherche ;
– en cas d'erreur, on peut cliquer à gauche de l'erreur (le curseur de la souris change de forme) pour obtenir plus de détails.

2.1 Calculs élémentaires

2.1.1 Liste de fonctions

– addition, soustraction, etc : `a+b, a-b, a/b, a*b`
– puissance : `a^b` ou `a**b`
– division entière : `a//b`
– reste de la division (calcul de modulo) : `a % b`
– approximation : `numerical_approx`
– racine carrée : `sqrt(a)`
– racine n-ième : `a^(1/n)`
– valeur absolue : `abs(a)`
– fonctions usuelles : `sin, cos, tan, log, exp,...`
– π : `pi`
– affectation de variable : `y = 1 + 3`
– nombre de chiffres d'un entier : `ndigits(a)`
– écriture binaire : `bits(a)`, `bits(a, padto = i)`

2.1.2 Exemples

```
2+2
```

```
4
```

```
2*(2+1)
```

```
6
```

```
2**3
```

```
8
```

```
2^3
```

```
8
```

```
40/28
```

```
10/7
```

Pas de limite sur la taille des nombres :

```
2^1000
```

```
1071508607186267320948425049060001810561404811705533607443 7\
5038837035105112493612249319837881569585812759467291755314 6\
8251871452856923140435984577574698574803934567774824230985 4\
2107460506237114187795418215304647498358194126739876755916 5\
5439460770629145711964776865421676604298316526243868372056 6\
8069376
```

```
20 // 6
```

```
3
```

```
20 % 6
```

```
2
```

```
y = sqrt(2) + sqrt(2); y
```

```
2*sqrt(2)
```

```
numerical_approx(y/2)
```

```
1.41421356237310
```

```
numerical_approx(sqrt(2), digits = 100)
```

```
1.4142135623730950488016887242096980785696718753769480731766 7\
9737990732478462107038850387534327641573
```

```
40.0/28
```

```
1.42857142857143
```

2.1.3 Autres questions intéressantes (ou pas)

Si on fait `y = 3` suivi de `y = y + 2`, que vaut `y` ? Comment définir une fonction (par exemple $\cos(x) + 2\sin(x)$), que vaut $\cos(\pi)$, que vaut $\cos^2(x) + \sin^2(x)$, que vaut $\left(\sqrt{2}\right)^2$, que vaut $\sqrt{x^2}$?

2.2 Manipulations symboliques

2.2.1 Simplifications

- Pour ne pas avoir à déclarer chaque variable symbolique : `automatic_name(True)`
- `factor`, `expand`, `simplify`
- `simplify_trig`, `reduce_trig`, `expand_trig`
- `simplify_radical`

2.2.2 Questions intéressantes (ou pas)

Trouver une expression simple de $x^3 - y^3$, retrouver les identités remarquables, développer l'expression $(x + y + z)^{10}$, factoriser $x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5$, factoriser $x^2 - 2\sqrt{2}x + 2$, retrouver les formules de trigo, etc.

2.3 Résolution d'équations

2.3.1 Les outils

- résolution symbolique : `solve`
- racines de polynômes : `roots`

2.3.2 Questions intéressantes (ou pas)

Peut-on résoudre des systèmes ? Des inéquations ? Les solutions sont-elles toujours exactes, approchées ? Sait-on résoudre avec des coefficients qui ne sont pas des nombres ?

2.4 Outils pour l'analyse

2.4.1 Les outils

- limites : `limit( f(x), x=a )`
- fonction symbolique : `function('f',x)`
- dérivation : `diff( f(x),x )`
- tracé d'une courbe : `plot( f(x), x, a, b)`

2.4.2 Questions intéressantes (ou pas)

Peut-on retrouver les formules de la dérivée d'une somme, d'un produit, d'une composée ? De la racine carrée d'une fonction ? L'équation de la tangente à une courbe ? La tracer sur le même graphique que la courbe ?

Peut-on calculer la limite de la suite $u_n = \dfrac{n^{100}}{100^n}$? Peut-on deviner cette limite en calculant les premiers termes ? En traçant une fonction ?

Peut-on tracer des courbes de couleurs différentes ?

2.5 Programmation

2.5.1 Boucles, fonctions et tests

On donne quelques exemples. Essayez de les comprendre et d'en faire d'autres. Pour les tests et comparaisons, on peut essayer `==, !=, <, >, <=, >=`.

```
for i in range(4) :
    print i
```

On fera très attention à l'indentation (et on demandera ce que c'est si on ne sait pas), primordiale en Python et Sage :

```
i = 1
while i < 10 :
  i = i+1
  print(i)
```

```
i = 1
while i < 10 :
   i = i+1
print(i)
```

```
i = 1
while i < 10 :
   i = i+1
 print(i)
```

```
j = 0
i = 1523489753
while i > 1000 :
    i = i/2
    j = j+1
j
```

```
def vaut_quatre (x) :
    if x == 4 :
        print x,'vaut bien 4'
        return True
    else :
        print x, 'ne vaut pas 4'
        return False
```

```
toto = vaut_quatre( 2 + 2 ); toto
```
```
4 vaut bien 4
True
```

```
titi = vaut_quatre( 42 ); titi
```
```
42 ne vaut pas 4
False
```

```
for i in range(100) :
    if i%10 == 0 :
        print i
```

Que fait la fonction (récursive) suivante (attention, n doit être un nombre entier) ?

```
def factorielle (n) :
    if n < 1 :
        raise ValueError('n doit etre >= 1')
    if n == 1 :
        return n
    else :
        return n*factorielle(n-1)
```

Peut-on obtenir la même fonction avec une boucle ?

Peut-on utiliser des boucles pour calculer la somme des termes d'une suite ? Pour calculer des approximations de π en utilisant l'égalité $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots$? Pour calculer une approximation de $\sqrt{2}$ par dichotomie ? Par la méthode de Babylone (grâce à la suite $u_{n+1} = \frac{1}{2}(u_n + 2/u_n)$ avec n'importe quel premier terme) ?

2.5.2 Listes

```
L = [2,3,5,7,11,13,42,0];L
```
```
[2,3,5,7,11,13,42,0]
```

```
print L[0], L[1], L[-1], L[-2], len(L), len([])
```
```
2 3 0 42 8 0
```

```
L[2:5]
```
```
[5,7,11]
```

```
[1..3, 7, 10..13]
```
```
[1,2,3,7,10,11,12,13]
```

On dispose des outils suivants pour manipuler les listes :
- concaténation : `L1 + L2`
- itération de la concaténation : `3*[1,2]`
- application d'une fonction aux éléments : `map`
- tri, renversement (change la valeur de L) : `reverse(L), sort(L), L.sort(), L.reverse()`
- suppression, indice, ajout d'éléments : `L.append(x), append(L,x), insert(L,i,x), L.insert(i,x), pop(L,i)`

2.5.3 Chaînes de caractères

Pour faire de la cryptographie ou des codes correcteurs d'erreur, on va avoir besoin de manipuler du texte. On donne d'abord quelques commandes.

```
phrase = 'hello world'
```

```
len(phrase)
```

```
11
```

```
print phrase[0], phrase[1], phrase[2], phrase[3]
```

```
h e l l
```

```
ord('u')
```

```
117
```

```
chr(117)
```

```
'u'
```

2.5.4 Questions intéressantes (ou pas)

Peut-on afficher tous les caractères manipulables avec Sage ? Convertir une chaîne de caractères en une liste de nombre ? Une chaîne de majuscules en minuscules ? Une liste de nombres en caractères ?

2.6 Aléatoire

Pour obtenir un élément aléatoire, il faut préciser à Sage où doit vivre cet élément. Par exemple, pour obtenir un entier relatif compris entre -3 e t3 :

```
ZZ.random_element(-3,3)
```

Pour les ensembles, on pourra essayer `RR, QQ, GF(7), MatrixSpace(QQ, GF(2))`, etc. Pour les polynômes, `QQ[x], QQ[x,y]`. On peut préciser un degré : `QQ[x,y].random_element(degree = 12)`.

2.6.1 Questions intéressantes (ou pas)

Comment jouer à pile ou face ? À un lancer de dé ? Écrire une fonction qui simule 20 lancers de pile ou face et qui affiche le résultat ? Compter le nombre moyen de tirages à faire pour obtenir 42 « Piles » au pile ou face ?

2.7 Si le temps le permet

On peut faire du crible d'Ératosthène (efficace ou pas), du *square and multiply*, des matrices, des divisions euclidiennes, des corps finis, des matrices, des matrices à coefficients dans des corps finis, du chiffrement par décalage, du chiffrement par permutations, etc.