Master CSI 2 2013-2014

Cryptanalyse — M1MA9W06

Responsable : G. Castagnos

Initiation au système Sage

1 Présentation

Sage est un logiciel de calcul formel multiplate-forme sous licence GPL disponible librement sur `http://www.sagemath.org` développé depuis 2005. Sage combine les fonctions de nombreux logiciels libres (GAP, Maxima, Singular, PARI/GP, NTL...) avec une interface commune basée sur Python. Sage peut être utilisé de différentes façons : via une interface graphique (le *Notebook*), avec une ligne de commande interactive, ou par des scripts Python en utilisant la bibliothèque Sage. Nous utiliserons principalement la ligne de commande interactive et des fichiers interprétés.

Des versions électroniques de cette fiche, des futures feuilles de TP et des corrections sont disponibles sur

`http://www.math.u-bordeaux1.fr/~gcastagn/?rubrique=Enseignement`

2 Pour démarrer

- Lancer la ligne de commande interactive de sage : `sage`
- Fermer une session : `Ctrl d` ou `exit`
- Interrompre un calcul sans fermer la session : `Ctrl c`
- Dernier résultat : `_`
- Lire un fichier par `load`, par exemple `load("toto.sage")`
- Plus pratique, on peut également « attacher » un fichier : par exemple `attach("toto.sage")`. En pressant la touche *return* dans Sage le fichier sera automatiquement ré-interprété après une modification
- La syntaxe suit celle de Python
- La complétion par la touche Tab permet d'accéder aux fonctions : essayer par exemple de taper `fac` puis `Tab`. Il y a également une complétion pour les méthodes applicables à un objet, essayer par exemple `a=5` puis `a.` et `Tab`
- Commentaires : `#` pour une ligne, `'''` et `'''` pour un bloc
- Renseignements sur une fonction : `?`. Par exemple : `factor?`. Taper `q` pour sortir de l'aide.
- Beaucoup de documentation en ligne sur `http://www.sagemath.org/doc/` pour Sage et sur `http://docs.python.org/` pour Python.

Pour la suite il est conseillé, pour garder une trace des commandes et fonctions tapées de les enregistrer dans un fichier avec un éditeur de texte, puis de lire ce fichier dans Sage avec la commande `attach`. Pour emacs, vous pouvez ajouter la ligne

```
(setq auto-mode-alist (cons '("\\.sage$" . python-mode) auto-mode-alist))
```

dans le fichier .emacs pour passer automatiquement en mode python lors de l'édition d'un fichier avec l'extension .sage.

3 Calculs de base

— Expérimenter les commandes suivantes sur les nombres entiers :

```
a = 5
a
2 == 2
2 < 3
a == 5
2^3
10%3
10/3
10//3
```

```
gcd(24,15)
a = 10 ; b=2/3
parent(a)
b.parent()
a = a/3
a
parent(a)
```

4 Listes

Expérimenter les commandes suivantes pour stocker une liste d'objets.

— Premiers exemples :

```
S = [1,2,3,12]
len(S)
S[0]
S[3]
S[2] = 5
S.append(18)
S
S[2:4]
S[2:]
S[:-1]
S = [1..10]
S = [1,5,..33]
10 in S
S.extend([4,3])
```

— Attention à la copie de listes !

```
S = [1..10]
M = S
M[2] = 78
M
S
S = [1..10]
M = copy(S)
M[3] = 24
M
S
```

— Constructions plus complexes :

```
S = [i^2 for i in [1..10]]
S = [i^2 + i + 1 for i in [3..5] if is_prime(i^2 + i + 1)]
```

— Conversions en bits, somme :

```
ZZ
a = ZZ.random_element(3,2^10)
L = a.digits(2, padto=16)
```

```
b = ZZ(L,2)
a == b
sum(L[i]*2^i for i in range(len(L)))
```

— Autres structures :
  — Sage fournit les séquences : similaire aux listes mais tous les éléments partagent un « univers » commun : tout élément rajouté à la séquence sera, si possible, transformé dans cet univers. Par exemple :
  `S=Sequence([3,4]); print S.universe(); print parent(S[1]); S.append(4/3)`
  — On peut aussi créer des ensembles : `Set([3,4,4])` il n'y aura pas de répétitions.
  — Python fournit aussi des dictionnaires : ce sont des collections non ordonnées d'objets. On accède aux valeurs d'un dictionnaire par des clés : `dico = {}; dico["une clef"] = 10; dico["une autre clef"] = 100; dico; dico["une clef"]`

5 Structures

Sage privilégie les structures. Tout objet mathématique (polynôme,vecteur, matrice, etc.) est créé comme un élément d'une structure. Celle-ci doit être créée au préalable (à l'exception des ensembles usuels : naturels, rationels ou réels).

Expérimenter les exemples suivants :

— Polynômes :

```
QQ
PR.<x> = PolynomialRing(QQ); PR
p = x^2-2 # Variante :
p = PR([-2,0,1])
p.is_irreducible()
p.roots()
factor(p)
p.roots(ring=RealField())
p(0)
p.coeffs()
p[2]
p.degree()
```

— Corps finis :

```
k1 = GF(7); k1
print [el for el in k1]
print k1.list()
a = k1(3); a
a^2
a.multiplicative_order()
a^6
k2.<b> = GF(2^3); k2
print k2.list()
b.minimal_polynomial()
p = k2.modulus(); p
p.is_irreducible()
PR.<y> = PolynomialRing(GF(2))
k3.<c> = GF(2^3, modulus=y^3+y^2+1)
k3; k3.list(); k3.modulus()
```

— $\mathbf{Z}/n\mathbf{Z}$:

```
A = IntegerModRing(8); A
A.list()
A.is_field()
A.list_of_elements_of_multiplicative_group()
euler_phi(8)
a = A(5)
a.multiplicative_order()
a.additive_order()
```

— Matrices :

```
F = GF(2)
MA = MatrixSpace(F,2); MA
M = MA([1,2,3,4]); M
# Variante pour ces 3 lignes :
N = matrix(GF(2),2,2,[1,2,3,4])
M == N
M[1,1] = 5; M
M.nrows(); M.ncols()
M.determinant()
MA(0)
MA(1)
T = matrix(2,3,[1,2,3,4,5,GF(3)(1)])
T;parent(T)
```

— Vecteurs, espaces vectoriels :

```
V = VectorSpace(GF(3), 3); V
u = V([1,2,0]); u
w = vector(GF(3), [4,2,10]); w
T*u
T[0] == u
T.stack(u)
T.augment(vector(GF(3), [1,1]))
```

6 Boucles if, for, while

La syntaxe est celle de Python : les débuts et fin de blocs ne sont pas marqués par des accolades ou des mots de types *begin* ou *end* mais uniquement par indentation.

— if :

```
if condition:
   instruction 1
   instruction 2
   ⋮
elif autre condition:
   instruction 1
   instruction 2
   ⋮
else:
   instruction 1
   instruction 2
   ⋮
```

— while :

```
while condition:
   instruction 1
   instruction 2
   ⋮
```

— for :

```
for enumeration:
   instruction 1
   instruction 2
   ⋮
```

L'énumération dans la boucle for peut être de la forme `i in [1..100]`, ou `i in S` avec `S` une séquence ou une liste ou bien une structure admettant une énumération dans Sage comme par exemple un anneau $\mathbf{Z}/n\mathbf{Z}$, un corps fini, un espace vectoriel sur un corps fini :

```
for i in VectorSpace(GF(2), 4):
  print i
```

Pour parcourir un ensemble d'indice d'une liste on pourra utiliser `range` :

```
L = [0,2,..10]
for i in range(len(L)):
  print L[i]+1
```

Mais attention le type des éléments de la liste `range` est le type entier de python et non pas un élément du `ZZ` de Sage !

```
L = range(10)
L[0].parent()
```

Énumération multidimensionnelle :

```
for i in mrange([2,5,3]):
  print i
```

7 Créer ses propres fonctions

— Création de fonctions (à mettre dans un fichier texte) :

```
def f(a,b,c,d):
   instruction 1
   instruction 2
   ⋮
   return u,v
```

— Appel de fonctions :

```
x,y = f(0,1,5,r)
```