Aide-mémoire Sage

Gabriel Lepetit

Ce document vise à recenser les principales commandes utilisées dans le logiciel Sage pour l'oral de modélisation de l'agrégation, option C.

NB : Le manuel de documentation est peu synthétique et manque de hiérarchisation, c'est pourquoi il est nécessaire de connaître ces choses essentielles.

1 Les erreurs fréquentes à déboguer

- Vérifier d'abord l'indentation et la présence de : après def, for, while, if, else...
- Les fonctions s'appliquent seulement à certains type d'objets, vérifier que l'erreur ne vient pas de là. Par exemple, une fonction polynômiale n'est pas un polynôme pour Sage.
- Faire des fonctions courtes, utiliser des fonctions auxiliaires et tester régulièrement sur des exemples
- Quand on n'a besoin de faire les calculs que sur un exemple, il n'est pas nécessaire de programmer une fonction générale.

2 Objets courants

2.1 Divers

- Ensembles de travail courants :
  `ZZ, QQ, RR, CC, QQ[sqrt(5)]`
- Demander le type d'un élément :
  `type(x)`
- Demander dans quel espace se trouve un élément :
  `x.parent()`
- Afficher un élement :
  `print x`
- Vérifier une égalité/une inégalité :
  `x == y (renvoie un booléen), x<>y, x<= y,x >= y`
- Boucles :
  `for, while, if, else, elif`
- Connecteurs logiques :
  `and, or, not`
- obtenir de l'aide sur une fonction :
  `fonction?`
- Faire un commentaire dans le code :
  `#mon commentaire`
- Fonctions courantes : partie entière, valeur absolue, factorielle, coefficients binomiaux
  `floor(x), abs(x), factorial(n), binomial(n,k)`

- Résoudre une équation :
  ```
  var("x,y")
  eq=x^2-y==0
  solve(eq,z)
  ```

2.2 Listes

Avertissement : les listes sont numérotées à partir de 0, c'est une source fréquente de confusion. Dans Sage les listes sont à la fois des tableaux et des listes de CamL : leur longueur n'est pas fixée par exemple.

- Définir une liste. Vide : v=[] ; $[0,\dots,n-1]$: v=range(n). $[n,\dots,m-1]$: v=range(n,m)
- Utilisation des listes dans les boucles : for x in l
- Longueur d'une liste : len(v)
- Concaténer deux listes : v+w
- Accéder au coefficient de rang $i \leqslant \text{len}(v)-1$ de la liste v : v[i]
- Enlever toutes les occurrences de x dans une liste : v.remove(x)
- Appliquer la fonction f à la liste l :
  ```
  map(f,l)
  ```
- Copier une liste :
  ```
  m=copy(l)
  ```
- Trier une liste
  ```
  l.sort()
  ```

2.3 Tracer des graphes

Un graphe est un objet graphique de Sage. Comme pour les listes, c'est un objet pouvant être concaténé à un autre graphe, à l'aide de la commande + ou +=; cela est utile pour tracer plusieurs graphes sur une même figure.

Voici une syntaxe typique :

```
p=Graphics()
for i in range(10):
    p=p+plot(x^i,(x,0,1))
p
```

- Graphique vide :
  ```
  Graphics()
  ```
- Fonction plot standard :
  ```
  plot(sin,(0, pi))
  plot(x^2,(x,0,pi))
  plot([sin(n*x) for n in [1..4]], (0, pi)) (plusieurs graphes)
  ```
  Paramètres possibles (disponibles sur toutes les formes de plot)
  ```
  color='red' (ou blue, green, purple, code hexadécimal...),
  legend_label='titre', legend_color='couleur', linestyle='-'
  ```
  (cela devient cosmétique !).
- Graphe avec une équation implicite (c'est à dire défini par une équation F(x,y)=0) : commencer par déclarer les variables puis utiliser la fonction de graphe implicite.
  ```
  var("x,y")
  implicit_plot(x^2+y^2-1,(x,-3,3),(y,-3,3))
  implicit_plot(x^2+y^2==3*x, (x,-3,3), (y,-3,3))
  ```
  Version 3D selon le même principe :

```
var("x,y,z")
implicit_plot3d(x^2+y^2+z^2-1,(x,-2,2),(y,-2,2),(z,-2,2))
```

- Courbe définie par une équation paramétrique (là encore, il faut déclarer les variables)
```
var("t")
parametric_plot((cos(t),sin(t),(t,0,2*pi))
parametric_plot3d((cos(t)-exp(-t), sin(t), cosh(t)), (t,0,2*pi))
```
- Graphe d'une liste (nuage de points)
```
list_plot([i^2 for i in range(5)])
list_plot([exp(I*theta) for theta in [0, .2..pi]])
```
- Autres fonctions *a priori* moins utiles :
```
complex_plot (graphe d'une fonction de la variable complexe avec des couleurs)
polar_plot (pour les courbes définies en polaire)
```

2.4 Polynômes

Ajouter les manipulations sur les anneaux de polynômes (idéaux, réduction modulo un idéal...)

- Préciser l'anneau de polynômes dans lequel on se trouve (indispensable pour éviter les bogues) :
```
R.<X,Y>=PolynomialRing(QQ)
```
- Tirer un polynôme au hasard :
```
P=R.random_element(degree=4)
```
- Degré :
```
P.degree()
```
- Pour savoir si P est unitaire, constant :
```
P.is_monic() ; P.is_constant()
```
- Accéder à la liste des coefficients de P, au coefficient de X^k, aux variables de P :
```
P.coeffs() ; P[k] ; P.variables()
```
- Appliquer la transformation f aux coefficients de P :
```
P.map_coefficients(f)
```
- Changer d'anneau de base
```
P.change_ring(B)
```
- Division euclidienne : obtenir Q,R tels que $P = QD + R$:
```
Q,R=P.quo_rem(D)
```
- PGCD, PPCM :
```
P.gcd(Q) ; gcd([P1,P2,P3]) ; P.lcm(Q) ; lcm([P1,P2,P3])
```
- Factorisation en produit d'irréductibles, savoir si P est irréductible :
```
P.factor() ; P.is_irreducible()
```
- Polynôme dérivé en la variable X :
```
P.derivate(X) ou P.diff()
```
- Racines dans $\mathbb{Q}$ (on peut choisir aussi une extension de $\mathbb{Q}$)
```
P.roots(QQ)
```
- Résultant en X de deux polynômes :
```
P.resultant(Q,X)
```
- Interpolation de Lagrange :
```
P=R.lagrange_polynomial([(x1,y1),...])
```

2.5 Algèbre linéaire

2.5.1 Créer et manipuler des matrices

Il y a deux manières de construire des matrices : ou bien on définit en premier lieu l'*espace des matrices* avec la commande MatrixSpace puis on dispose de fonctions pour construire des matrices dans cet espace. Ou bien on ne le spécifie pas et on travaille avec les commandes comme matrix et ses dérivés. Dans ce deuxième cas, Sage considère qu'on travaille avec des entiers ou des rationnels.

Attention, comme pour les listes, la numérotation des lignes et des colonnes commence à 0.

- Espace de matrice $\mathscr{M}_{2,3}(\mathbb{Z})$: MS = MatrixSpace(ZZ,2,3)
- Obtenir l'anneau de base à partir de l'espace MS :
  `MS.base_ring()`
- Changer d'anneau :
  `MS.change_ring(B)`
- Définir une matrice : matrice nulle
  `MS() ou MS.zero()`
  matrice identité :
  `MS.one() ou MS.identity_matrix()`
  Si MS n'est pas spécifié :
  `A=matrix([[1,2],[3,4]]) ; matrix(ZZ,4,4,range(16)) (numérotée de 0 à 15); random_matrix(ZZ,4,4) ; identity_matrix(K,n)`
- Produit, produit par un scalaire, inversion, transposition, conjugaison :
  `A*B, a*A, A^{-1}, A.transpose(), A.conjugate(),`
- Matrice par blocs : si A, B, C, D sont des matrices, on peut définir
  `M=block_matrix([[A,B],[C,D]]) ou M=block_diagonal_matrix(A,A.transpose())`
  Options : enlever les marqueurs de subdivision : subdivide=false
- Accéder à un coefficient : M[i,j] ;) une ligne M[i, :]
- Extraire une matrice
  `A[3:4,2:5] (lignes 3 à 4, colonnes 2 à 5), A[:0:8:2] (colonnes paires)`
  `matrix_from_rows(A,[0,2,3]), matrix_from_columns(A,[1,2])`
  `matrix_from_rows_and_columns(A, [0,2,3], [1,2])`
  (on extrait les lignes 0,2,3 et les colonnes 1,2)
- Concaténer deux matrices par les lignes, par les colonnes :
  `A.stack(B), A.augment(B)`

2.5.2 Systèmes linéaires, manipulation sur les espaces vectoriels

- Définir un espace vectoriel :
  `E=VectorSpace(QQ, 3)`
- Base d'un espace :
  `E.basis()`
- Définir un vecteur colonne : vector([1,1,1])
- Résoudre $Ax = b$:
  `A\b ou A.solve_right(b)`
- Image, noyau d'une matrice : attention, kernel pour Sage est le noyau à **gauche**, c'est à dire le noyau de ${}^tA$! ! Ces fonctions donnent une base de l'image et du noyau.
  `A.image(), A.right_kernel()`
- Forme échelonnée d'une matrice (réduction de Gauss) :
  `A.echelon_form()`
- Déterminant : A.det()

2.5.3 Réduction

- Polynôme caractéristique, polynôme minimal : A.charpoly(), A.minpoly()
- Eléments propres d'une matrice : valeurs propres avec multiplicité sous forme d'une liste rangée par ordre croissant, couples (λ, [base de vecteurs propres de E_λ], dim E_λ) pour la deuxième.
  `A.eigenvalues(), A.eigenvectors_right()`
- Forme de Jordan : ne fonctionne que si le polynôme caractéristique est scindé. La deuxième expression renvoie J matrice de Jordan et U inversible tel que $A = UJU^{-1}$.
  `A.jordan_form() ; J,U=A.jordan_form(transformation=True)`
- Forme de Frobenius :
  `A.frobenius() (matrice de Frobenius) ; A.frobenius(1) (facteurs invariants) ; A.frobenius(2) (matrice de passage)`

2.6 Arithmétique

Remarque sur les corps finis : il est connu qu'un corps fini $\mathbb{F}_{p^n}$ pour p premier se construit comme un quotient $\mathbb{Z}/p\mathbb{Z}[X]/(f)$ où f est un polynôme irréductible de degré n. Quand on demande "le" corps fini de cardinal p^n à Sage, il choisit donc un tel polynôme pour nous, mais on peut le spécifier.

- L'arithmétique dans $\mathbb{Z}$ utilise les memes fonctions PGCD, PPCM, factorisation en facteurs irréductibles, division euclidienne, etc que dans les anneaux de polynomes. On peut tester si un entier est premier avec :
  `p.is_prime()`
- Se placer dans $\mathbb{Z}/n\mathbb{Z}$:
  `R=IntegerModRing(15) ; x=R(3) ou bien x=mod(3,15)`
- Définir un corps fini engendré sur $\mathbb{F}_p$ par x / quotient de $\mathbb{Z}/p\mathbb{Z}[X]/(f)$:
  `R.<x>=GF(9) ;`
  `Q.<x> = PolynomialRing(GF(3))`
  `R.<x> = GF(9, modulus=x^2+1)`
- Obtenir le polynôme définissant un corps fini :
  `R.polynomial()`
- Ordre d'un élément dans le groupe multiplicatif :
  `x.multiplicative_order()`