Cryptographie post quantique : TP Introduction à sage

Contents

0.1 Installation de SageMath

Option 1 : Installation de sage en local (recommandé)

- Installer sage via : https://doc.sagemath.org/html/en/installation/index.html,
- Lancer un Jupyter Notebook: `sage -n jupyter`.
- Ecrire vos codes dans des `.sage` et les exécutez depuis le terminal.

Option 2 : Installation de sage en ligne

- Connectez-vous, et créer un projet sur Cocalc: https://cocalc.com/,
- Vous pourrez alors y importer vos propres Jupyter Notebook.

0.2 Documentation de SageMath

Pour accéder à l'aide d'une méthode dans SageMath, vous pouvez utiliser la fonction ? :

- divisors? affichera la documentation de la fonction divisors.
- La fonction **help**(.) peut également être utiliser (pour avoir un affichage différent) avec comme argument la fonction à documenter : **help**(divisors).

Vous pouvez également lire la documentation (lien) ainsi que les exemples directement sur le site officiel de sage (recommandé).

1 Opérations de base, tests, affectations et typages

(PreliminairesFR) **Exercice 1.** *Opérations de base*

1. Expérimenter les commandes de bases et comprendre leur utilité:
   - 1+1
   - x==8 **and** x != 5
   - **print**("texte",x,"encore␣du␣texte")
   - parent(x)
   - **type**(x)

(TypesFR) **Exercice 2.** *Types*

Les différences entre parent et type sont importantes dans la gestion de structures par SageMath: lien et leur égalité ne veut pas toujours dire que ce sont des éléments identiques. La différente se traduit notamment par leur définition et leur affectation dans le cadre de structures algébriques comme nous allons le voir lors de la section 3. Anneau, Corps Finis et Polynômes.

1.
   - 10/3
   - 10//3
   - 10%3
2. Dans quel structures appartiennent chacuns des résultats des 3 résultats précédents ?

2 Listes et ranges

(ListeFR) **Exercice 3.** *Listes et ranges*

1. Expérimenter les commandes de bases sur une liste S et comprendre leur utilité.
   - S =[1,2,3,4]
   - S[0]
   - S[−1]
   - S[0:3]
   - S[1:]
   - S[:2]
   - T1 = copy(S)
   - T2 = S
   - S[0] = 2
   - T1 == T2
   - T3 = **range**(1,5)
   - T1 == T3
2. A l'aide de la documentation de **range**, construire les "ranges":
   - [1,2,..,9,10]
   - [2,4,6,8,10,..,120]
   - [33,29,..,5,1]
3. A l'aide de la documentation de **list**, construire les mêmes "listes" ci-dessus.
4. Avec une boucle 'for'imbriquée, construire la liste contenant les 10 premiers carrés non nuls.

Nous n'utiliserons ici que les listes mais il existe d'autres classes de collections comme par exemple les ensembles **set** ou encore les dictionnaires **dict**.

3 Anneaux, Corps Finis et Polynômes

(AnneauxFR) **Exercice 4.** *Anneaux*

1. Distinguer les 3 "ensembles" de nombres suivants, à quoi correspondent-ils ?
   - ZZ (représentant $\mathbb{Z}$)
   - QQ (représentant $\mathbb{Q}$)
   - RR (représentant $\mathbb{R}$)

SageMath permet surtout de manipuler des éléments appartenant à des ensembles et structures algébriques différentes que simplement les entiers, rationnels et réels.

2. Expérimenter ces commandes de bases et comprendre leur utilité.

- Z11 = IntegerModRing(11)
- Z11.**list** ()
- Z16 = IntegerModRing(16)
- Z16.list_of_elements_of_multiplicative_group()
- euler_phi(16)
- z1 = Z11(13); z2 = Z16(13)
- z1.multiplicative_order()
- z2.multiplicative_order()

Voici un exemple de différence entre parent et **type**.

- parent(z1) == parent(z2),
- **type**(z1) == **type**(z2).

(CorpsFinisFR) **Exercice 5.** *Corps Finis*

Tout corps fini $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ pour tout nombre premier p. Mais il existe également un moyen d'instancier tout corps finis directement graĉe à $\texttt{GF}(n)$, pour tout entier $n = p^k$.

- F11 = GF(11)
- F11.**list** ()

Malgré que les objets soient théoriquement les mêmes, les classes définissant les 2 objects sont différents et induisent des fonctions et méthodes différentes.

- F11 == Z11
- **type**(F11)
-
- **type**(Z11)

Dans le cas de $\mathbb{F}_{p^k}$, on voit la liste des éléments comme des polynômes en x.

- F4.<x> = GF(4)
- F4.**list** ()
- F4.modulus()

Nous introduirons la structure de polynôme dans la section qui suit. Mais il est possible de choisir le polynôme générateur, unitaire et irréductible dans $\texttt{GF}(p)[X]$, de degré k, utilisé afin d'instancier $\texttt{GF}(p^k)$:

- PR.<y> = PolynomialRing(GF(5))
- f = y^2 + 2
- assert f.is_irreducible()
- F25.<x> = GF(25,modulus=f)
- F25.modulus() != GF(25).modulus()

(PolynomeFR) **Exercice 6.** *Structures polynomiales*

1. Expérimenter ces commandes de bases et comprendre leur utilité.
   - K1.<x> = PolynomialRing(QQ)
   - K2.<x> = QQ[]
   - K3 = QQ['x']
   - K4 = PolynomialRing(QQ,'x')
   - K1 == K2 == K3 == K4

2. Expérimenter ces commandes d'affectations de polynômes.
   - p1 = x^3 + 2*x − 2
   - p2 = K1([−2,2,0,1])

3. Expérimenter ces commandes liées aux objets polynomiaux et comprendre leur utilité.
   - p1.degree()
   - p1.is_irreducible()
   - factor(p1)
   - p1.roots()
   - p1.roots(ring=CC)
   - p1(2)
   - p1[2]

(IdeauxFR) **Exercice 7.** *Idéaux et quotients*

1. Expérimenter les commandes sur les idéaux et comprendre leur utilité:
   - ZZ.ideal(2)
   - Z6 = IntegerModRing(6); I23 = Z6.ideal(ZZ(2),ZZ(3))
   - ZZ.ideal(2,3)
   - parent(x)
   - **type**(x)

Pareil pour, le cas des anneaux de polynômes:

2. Expérimenter les commandes sur les idéaux et quotient et comprendre leur utilité:
   - K5.<x> = PolynomialQuotientRing(K1,p1)
   - I = K1.ideal(p1)
   - K1.quotient(I) == K5

4 Vecteurs et Matrices

(VecMatFR) **Exercice 8.** *Vecteurs et Matrices*

1. Expérimenter les commandes de bases et comprendre leur utilité.

   - v = vector ([6,0,6])
   - v2 = vector(ZZ ,[6,0,6])
   - v3 = vector(Z11 ,[−1,12,0])
   - Z = zero_matrix(4,3)
   - z = zero_vector(3)
   - M = Matrix ([[1,2,1],[2,1,2],[3,−3,3]])
   - I3 = identity_matrix(3,3)
   - M.det()
   - M.T
   - y1 = kernel(M)
   - y2 = kernel(M.T)
   - M.nrows()
   - M.ncols()
   - M[0,2]
   - assert M.solve_left(v) ∗ M == v
   - assert M ∗ M.solve_right(z) == z

On peut également définir des vecteur/matrices pour tout type de structure algébriques. Soit en instanciant la structure tout entière, soit en spécifiant lors de l'affectation.

2. Expérimenter les commandes de bases et comprendre leur utilité.

   - F4.<x> = GF(4)
   - MS = MatrixSpace(F4,3)
   - M = MS([1,1+x,x] **for** i **in range**(3))
   - MS.random_element()
   - assert MS(0) == zero_matrix(F4,3,3)
   - assert MS(1) == identity_matrix(F4,3,3)

5 Exercice d'entrainements

(TrainPGCDFR) **Exercice 9.** *PGCD*

1. Calculer le pgcd, le reste et le quotient de la division euclidienne de 4092 par 1017.

2. Que fait la fonction xgcd ? Utilisez-la sur des entiers, puis des polynômes

(TrainAlgebreFR) **Exercice 10.** *Structures algébriques*

On rappelle que les inversible de $\mathbb{Z}/n\mathbb{Z}$ sont les entiers premiers avec n.

1. Écrire une fonction inverse(n) qui calcule $(\mathbb{Z}/n\mathbb{Z})^\times$ pour un entier positif non nul n en entrée.
2. Rien qu'en regardant la taille de $(\mathbb{Z}/391\mathbb{Z})^\times$, 391 est-il premier ? Pourquoi ?
3. Rechercher l'ordre multiplicatif de 12 modulo 391, par une boucle, puis par l'anneau $(\mathbb{Z}/391\mathbb{Z})^\times$.
4. Construire deux listes L1 et L2, telle que L2[i] est l'inverse de L1[i] dans $(\mathbb{Z}/391\mathbb{Z})^\times$.
5. Combien d'éléments de $\mathbb{Z}/391\mathbb{Z}$ sont les inverses de eux mêmes?

(TrainIdeauxFR) **Exercice 11.** *Idéaux*

1. Calculer tous les idéaux I de $\mathbb{Z}/391\mathbb{Z}$ engendrés par un élément, et l'anneau quotient $R = (\mathbb{Z}/391\mathbb{Z})/I$.
2. Expliquer mathématiquement la structure de R.

(TrainCorpsFinisFR) **Exercice 12.** *Corps Finis et Polynômes*

1. Créer le corps finis à 8 éléments, donner en une base puis lister ces éléments en fonction de la base.
2. Vérifier l'identité $(x+y)^2 = x^2 + y^2$ dans $\mathbb{F}_8$ et puis prouvez-le.
3. Quel est le polynôme minimal utilisé pour sa construction ? Prouvez qu'il est irréductible (avec la fonction Sage puis par le calcul)?

(TrainReseaux1FR) **Exercice 13.** *Réseaux euclidiens 1*

Importer une librairie de Réseaux euclidiens: **from** sage.modules.free_module_integer **import** IntegerLattice
Voici un exemple d"instance de l'objet: IntegerLattice ([[3,0], [2,1]], False).

- Le premier paramètre est la base de notre réseaux euclidiens sous forme de **listes de listes**, **listes de vecteurs**, mais ça peut également être un objet **matrix**.
- La seconde option permet de réduire la base indiquée en 1er paramètre, avec l'option False la base d'entrée ne sera pas réduite (par défaut, elle le sera). Chaque élément de la base sera alors une combinaison linéaire des lignes indiquées.
- IntegerLattice? pour plus d'informations.

1. Ecrire la fonction `gauss(u,v)` ressortant une base réduite par Lagrange-Gauss en dimension 2. Tester pour une matrice aléatoire de dimension 2 avec random_matrix.

Soit p un nombre premier tel que $p \equiv 1 \mod 4$ On rappelle que dans ce cas -1 est un carré modulo p, on note alors s un tel entier i.e. $s^2 \equiv -1 \mod p$.
On considère le réseau $\mathcal{L}$ de $\mathbb{R}^2$ de base $\begin{bmatrix} 1 & 0 \\ s & p \end{bmatrix}$ i.e. $\ell \in \mathcal{L} \iff \exists x_1, x_2$ tel que $\ell = x_1 \begin{bmatrix} 1 \\ s \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ p \end{bmatrix}$
On sait qu'il existe un vecteur $u \in \mathcal{L}$ tel que $\|u\| \leqslant \frac{3}{4}\sqrt{\det(\mathcal{L})}$.

2. En déduire et implémenter un algorithme polynomial decomposition_somme_carree(u,v) prenant en entrée un nombre p et retournant deux entiers a et b tel que $a^2 + b^2 = p$.

3. Décomposer 123794003928538027489912435 7 en somme de deux carrés.

(TrainReseaux2FR) **Exercice 14.** *Réseaux euclidiens 2*

1. Générer des réseaux aléatoires en utilisant random_matrix. A partir de dimension 10, puis de dimension de plus en plus grande.

2. Essayer de résoudre le problème du plus court vecteur dans vos réseaux en utilisant sur un objet **IntegerLattice** la méthode .shortest_vector(algorithm="pari"). A partir de quelle dimension le problème devient-il trop long à résoudre ? (ça peut dépendre de vos machines).

3. Générer de nouveaux réseaux de mêmes dimensions mais cette fois avec la fonction gen_lattice, issu du paquetage sage.crypto.

4. Essayer de résoudre le problème du plus court vecteur dans vos réseaux. A partir de quelle dimension le problème devient-il trop long à résoudre ? Expliquer les différences avec la résolution affectuée lors de la question 2.