# Prise en main rapide de Sage

Connectez-vous à l'adresse `http://sage-math.univ-lyon1.fr`. Rentrez votre `login` puis si votre navigateur le demande ajoutez une exception pour un certificat de sécurité. Rentrez à nouveau votre `login` puis votre `mot de passe`. Cliquez sur le lien "New Worksheet" : une feuille de calcul vierge apparaît. Cliquez sur son nom "Untitled" et renommez-la par exemple "Prise_en_main".

## 1 Feuille de calcul

Elle se présente sous la forme de cellules dans lesquelles vous taperez vos commandes. Entrez par exemple l'expression :

```
factor(12780)
```

puis validez par les touches `<MAJ><ENTREE>`.

La décomposition en facteur premier de 12780 a été retournée. Vous pouvez l'afficher sous une forme plus lisible en utilisant la commande `view` ou la commande `show`. Entrez pour cela dans la cellule suivante :

```
view(factor(12780))
```

puis validez.

A tout moment, on peut :
- *insérer une nouvelle cellule* : placer la souris entre deux cellules et cliquer sur la ligne bleue qui apparaît,
- *supprimer une cellule* : effacer d'abord son contenu puis presser la touche `<DELETE>`,
- *insérer un texte au-dessus d'une cellule* : placer la souris juste au-dessus de la cellule et cliquer sur la ligne bleue qui apparaît en tenant la touche `<MAJ>` enfoncée.

Insérez un commentaire au-dessus d'une des cellules. Pour cliquez sur "Saves changes". Pour modifier ce texte il suffit de cliquer deux fois dessus.

## 2 Assignation

Pour des calculs un peu longs, il est pratique de nommer les expressions intermédiaires par l'instruction d'*assignation*, de la forme : `nom = expression`

Dans une nouvelle cellule, tapez (puis validez) :

```
a = 62735572 ; b = 49362
r = a.mod(b)
r
```

Notez au passage que dans une cellule :
- on peut saisir plusieurs lignes,
- sur chaque ligne, on peut saisir plusieurs instructions séparées par des points-virgules,
- le résultat affiché est la valeur de la dernière expression rencontrée, ici `r`.

Il est possible d'assigner plusieurs noms à la fois. Dans la cellule précédente, on peut remplacer la première ligne par :

```
a,b = 62735572, 49362
```

Pour Sage, les expressions mathématiques sont des *objets* au sens informatique du terme. Tout objet a un *type* bien défini (par exemple Integer, Rational etc.), des *attributs* (i.e. données internes) et des *méthodes* (i.e. fonctions qui peuvent lui être appliquées). Pour faire appel à une méthode d'un objet, la syntaxe est de la forme : `objet.methode(`*paramètres éventuels*`)`

Par exemple, ci-dessus on a fait appel à la méthode `mod()` de l'objet `a`.

Remarque.— Certaines méthodes peuvent également être appelées sous une forme plus classique : `methode(objet,`*paramètres éventuels*`)`.

Pour afficher le type de `r`, vous pouvez entrer la commande : `type(r)`

Dans une nouvelle cellule, tapez (puis validez) :

```
c = a/b ; type(c)
```

# 3 Aide en ligne

Sage contient de nombreuses fonctions et méthodes prédéfinies.

Cherchons si l'entier `r` est premier ou non. Tapez dans une cellule

```
r.
```

suivi de la touche `<TAB>` ce qui fait apparaître la liste des méthodes applicables à l'objet `r`. Cliquez sur la méthode `is_prime`, complétez la cellule par un couple de parenthèses `()` et validez :

```
r.is_prime()
```

*Auto-complétion* des mots : dans une nouvelle cellule, tapez le début d'une commande, par exemple :

```
r.is
```

suivi de la touche `<TAB>` : seules les méthodes commençant par `is` vous sont proposées. Vous pouvez alors compléter successivement par quelques lettres suivies de `<TAB>`. Par exemple tapez ensuite `_s` suivi de `<TAB>`, puis `f` suivi de `<TAB>`, enfin complétez par `()` et validez :

```
r.is_squarefree()
```

*Aide sur une méthode* : dans la cellule précédente, placez votre curseur juste après la première parenthèse de la méthode, par exemple

```
r.is_squarefree(
```

puis tapez sur la touche `<TAB>` pour faire apparaître la documentation sur `is_squarefree()`. avec des exemples d'utilisation.

Vous pouvez également tapez la méthode suivi d'un point d'interrogation, par exemple

```
factor?
```

suivi de `<TAB>`.

# 4 Expressions symboliques

Sage permet de faire du calcul formel, c'est-à-dire manipuler des expressions mathématiques contenant des symboles.

Essayez cet exemple (ce qui est écrit après le symbole # est un commentaire, inutile de le recopier)

```
var('x,y') # déclare deux variables symboliques x et y
f(x) = (x-2)*(x-3)
f
```

puis

```
f(y^2)
```

et enfin

```
solve_mod([f(x) == y^2],6)
```

# 5 Types composés

Sage utilise le langage de programmation Python, qui lui-même possède trois types de données composés qui vous seront utiles :

1. *n-uplet* : plusieurs expressions séparées par des virgules, et entourées d'un couple de parenthèses (en général facultatives), par exemple :

   ```
   v = (x,y,z) ; v
   ```

2. *liste* : se note avec des crochets, par exemple :

   ```
   L = [1,4,2,8,5,7] ; L
   ```

   Faites afficher le type de `v` et de `L`.

   L'accès à un élément d'un $n$-uplet ou d'une liste se fait en précisant son indice (numéroté à partir de 0). Evaluez par exemple `L[0]`. Le nombre d'éléments de L s'obtient avec `L.len()` (ou `len(L)`). On peut modifier les éléments d'une liste, mais pas ceux d'un $n$-uplet. Essayez par exemple :

   ```
   L[0] = -1 ; L
   ```

3. *dictionnaire* : c'est un ensemble de couples de la forme `clef:valeur`, par exemple :

   ```
   d = {4 : 8, 1 : 25}
   ```

   L'accès à un élément se fait en précisant la clef entre crochet. Evaluez par exemple `d[4]`.

On peut mettre bout-à-bout des listes (ou des $n$-uplets) avec l'opération `+`. Essayez :

```
M = [1,3] + [5,15]
M == Integer(15).divisors()
```

Une liste d'entiers successifs s'obtient avec la fonction `range`. Testez les commandes `range(10)`, `range(3,25)` et `range(3,25,2)`

Il y a aussi la construction `[n..m]` qui est équivalente à `range(n,m+1)`.

Enfin, on peut construire des listes par *compréhension* :

```
[n^3 for n in [1..10]]
```

# 6 Programmation

Vous serez amené à programmer vos propres fonctions et utiliser des tests ou des boucles. La syntaxe utilisée est celle du langage Python, avec ses structures de contrôle `if, for, while`.

L'exemple suivant définit une fonction `phi` :

```
def phi(n) :
    res = 0
    for i in [1..n] :
        if (gcd(i,n) == 1) :
            res = res + 1
    return(res)
```

Notez bien le symbole " :" à la fin des lignes `def ...`, `for ...` et `if ...`, ce symbole est indispensable. Vous remarquerez également que le texte est indenté. Ces indentations sont également indispensables, elles définissent les *blocs d'instructions* pour les structures `def`, `for` et `if` respectivement (qui sont ici imbriquées). Toutes les instructions d'un même bloc doivent impérativement commencer sur une même colonne.

Vous trouverez de la documentation pour Python sur le site officiel `http://www.python.org/`

# 7 Interface Notebook

Vous utilisez l'interface *Notebook* de Sage. En haut de la page vous trouverez de nombreuses fonctionalités et divers liens.

En particulier, en haut à gauche dans la rubrique *File* :

– *Download to file* pour enregistrer une copie de votre feuile de calcul sous forme d'un fichier `.sws`.

– *Upload worksheet from file* pour charger une feuille de calcul à partir d'un fichier `.sws`.

Enregistrez dans un fichier votre feuille de calcul.

Dans la rubrique *Action* :

– *Interrupt* pour interrompre le calcul en cours,

– *Restart worksheet* pour effacer de la mémoire tous les calculs effectués,

– *Evaluate all* pour effectuer l'ensemble des calculs de la feuille à partir du haut.

En haut à droite, vous trouvez un lien *Help* vers la documentation sage et en dessous des boutons *Save*, *Save & quit* et *Discard & quit* qui permettent respectivement d'enregistrer votre feuille de calcul sur le serveur, d'enregistrer et fermer cette feuille, de fermer sans enregistrer.

Enregistrez et fermez cette première feuille. Puis récupérer la feuille pour le premier TD se trouvant sur `http://math.univ-lyon1.fr/~blossier/ATN/arith.sws`

A la fin du TD vous quitterez le serveur Sage en utilisant le bouton *Sign out* en haut à droite.