# Programmation applicative – L2
# TP n° 0 : Prise de contact avec l'interpréteur Scheme

H. Chahdi {hatim.chahdi@umontpellier.fr} T. Lorieul {titouan.lorieul@umontpellier.fr}
J. Thiebaut {josselyn.thiebaut@umontpellier.fr}

## 1 Préambule

Cet enoncé concerne la première séance de TP, consacrée à la prise en main de l'interpréteur Scheme que vous utiliserez tout au long du semestre. Le but n'étant pas de vous noter mais de vous accompagner dans la découverte du langage Scheme et de l'interpréteur associé, ce TP ne sera pas évalué. Vous n'avez donc rien à rendre à la fin de la séance.

Pour les séances suivantes, vos chargés de TP pourront à tout moment relever votre travail afin de l'évaluer. **En conséquence, il faudra sauvegarder précieusement votre travail au fur et à mesure des séances.**

## 2 L'environnement de développement

L'interpréteur Scheme que nous allons utiliser dans les TPs est DrRacket, accessible via le terminal avec la commande `drracket`. Pour ceux qui souhaiteraient l'installer sur leur machine personnelle, il est en libre téléchargement sur le site *https ://racket-lang.org/download/*.

### 2.1 Introduction

DrRacket est un environnement interactif de programmation en langage Scheme. La fenêtre qui apparaît est divisée en deux zones principales (elle comporte aussi un certain nombre de menus et de boutons en haut sous la ligne de menu) :

- la zone blanche supérieure est la **zone de définitions** : c'est là que vous allez entrer les nouvelles fonctions que vous voulez définir (c'est-à-dire des expressions Scheme (parenthésées et préfixées) commençant par le mot clef *define*).
- la zone blanche inférieure est la **zone d'interactions** : c'est là que vous allez demander à Scheme d'évaluer des expressions. La présence de l'invite (un signe '>') vous indique que Scheme est déjà prêt à recevoir vos ordres.

Le curseur clignotant ('|') indique où vont être retransmis les prochains caractères que vous taperez (dans une zone ou dans l'autre). Pour écrire des commentaires dans votre code, il suffit de faire précéder la ligne de commentaire d'un point-virgule " ;".

### 2.2 Premiers pas

#### 2.2.1 Interactions

Dans la zone d'interactions, positionnez-vous après le signe '>' en cliquant à cet endroit avec le bouton gauche de la souris. Entrez au clavier :

```
(+ 3 4)
```

en séparant les arguments par des espaces puis tapez sur la touche "entrée". Après la phase **lecture**, Scheme passe dans la phase d'**évaluation** (calcule rapidement le résultat) puis dans la phase **résultat** (où le résultat de l'évaluation, 7, s'affiche à l'écran). Puis, suivant le principe de la *boucle interactive*, il se repositionne dans la phase lecture (affichage de l'invite, attente de votre prochain ordre à lire au clavier).

**Petite astuce** (qui sauve la vie) : si vous voulez éditer une commande que vous avez tapée précédemment, il est possible d'accéder à l'historique des commandes avec (CTRL + flêche du haut) ...

**Exercice 1** *Tapez maintenant les expressions suivantes en essayant de prévoir leur résultat :*

```
(< 10 100)
(and (< 55 100) (> 55 0))
(* 1 2 3)
(if (10 < 20) #t #f)
```

**Exercice 2** *Corrigez la dernière expression pour que son évaluation ne produise pas d'erreur.*
*Simplifiez maintenant cette expression (trouvez une expression équivalente mais plus courte).*

**Exercice 3** *Mettez les expressions suivantes sous forme parenthésée et préfixée, puis vérifiez que vous avez bon en les tapant dans la zone d'interactions, puis notez leur résultat :*

| *expressions initiales* | *expressions Scheme* | *résultats de l'évaluation* |
|---|---|---|
| `2 - 6 + 10` | | |
| `2 + 5 * ( -3 + 12)` | | |
| `(9/ 2) / (2 / 4)` | | |

**Exercice 4** *Évaluez les expressions logiques ci-dessous, pouvez-vous prévoir leur résultat ?*

| *expressions logiques* | *résultats* |
|---|---|
| `(not (and (< 55 100) (> 55 0)))` | |
| `(or (not (< 55 100)) (not (> 55 0)))` | |

**Exercice 5** *Pouvez-vous expliquer pourquoi elles donnent toutes les deux le même résultat (aussi dans le cas où tous les 55 sont remplacés par une autre valeur numérique dans les deux expressions) ?*

*Indice* : considérez les deux sous-expressions `(< 55 100)` et `(> 55 0)` comme étant deux expressions logiques A et B et établissez la table de vérité des expressions du tableau ci-dessus en fonction des valeurs de A et de B (soit #t, soit #f).

### 2.2.2 Définition de nouvelles fonctions

### 2.2.3 Entrée des fonctions

**Exercice 6** *Positionnez maintenant le curseur dans la zone de définitions (zone supérieure). Tapez la suite de lignes suivante en respectant les espaces, les passages à la ligne et l'indentation (espaces insérés en début de certaines lignes) :*

```
(define (puis2 x)
   (* x x))
```

### 2.2.4 Vérification de la syntaxe des fonctions

Pour vérifier la syntaxe dans la zone de définitions, cliquez ensuite sur le bouton "*Vérifier*" ("*Check Syntax*" en anglais).

Amenez maintenant le pointeur de la souris (sans cliquer) au dessus du caractère '`x`' de la première ligne : l'environnement vous indique par des flèches dans quelle(s) sous-expression(s) de la définition courante est utilisé cet argument. Si vous positionnez la souris au dessus d'un des '`x`' d'une sous-expression de la fonction (ici de "`(* x x)`"), vous remarquez qu'une flèche vous indique où cet argument que vous utilisez a été défini.

### 2.2.5 Évaluation et appel des fonctions

Cliquez maintenant sur le bouton "*Exécuter*" : la zone d'interactions est vidée de son ancien contenu, et les expressions de la fenêtre supérieure sont évaluées.

Si aucune erreur ne survient lors de l'évaluation (c'est-a-dire si votre fonction est syntaxiquement correcte), le curseur est positionné après l'invite et vous engage à taper un ordre.

**Exercice 7** *Vérifiez que la fonction* `puis2` *est maintenant une fonction que connaît Scheme en tapant les expressions suivantes dans la zone d'interactions :*

```
(puis2 3)
(puis2 -4)
(puis2 1)
(puis2 (puis2 2))
(puis2 (puis2 1))
```

Dans la suite, nous vous recommandons de suivre toujours cette démarche en trois temps : entrée des fonctions, vérification de leur syntaxe (jusqu'à obtenir une définition sans erreurs), essais d'appels de ces fonctions pour vérifier qu'elles fonctionnent correctement.

**Exercice 8** *Repositionnez-vous maintenant dans la zone de définitions (après la définition de puis2) et entrez texto (sans modification) les lignes suivantes :*

```
(define (puis4 x)
   (puis2 (puis2))
)
```

Vérifiez la syntaxe, et positionnez la souris sur un "`puis2`" de la ligne centrale de cette définition. Automatiquement l'environnement vous indique où est définie cette fonction que vous appelez.

**Exercice 9** *Après avoir cliqué sur le bouton* Exécuter*, passez maintenant dans la zone d'interactions et essayez d'appeler la nouvelle fonction, par exemple en tapant :*

```
(puis4 10)
```

*Quel est le résultat ? Corrigez la partie définition suivant ce que vous indique Scheme, puis vérifiez que vous avez bon (dans l'ordre, syntaxe, exécute, essai d'appel de la fonction). Recommencez jusqu'à obtenir des résultats corrects.*

### 2.2.6 Sauvegarde du travail réalisé

Cliquez sur l'icône "**Sauvegarder**" (ou "**Save**"), dans la fenêtre apparaissant, dans la zone où le curseur est positionné, indiquez un nom de fichier (par exemple *tp0.scm*) dans lequel l'environnement va sauver les fonctions que vous avez actuellement définies (contenu de la zone *définition de fonctions*). Ceci permettra de ne pas avoir à retaper ces fonctions, si vous en avez encore besoin une prochaine fois.

## 3 Obtenir de l'aide

Cliquez sur le menu d'aide (*Help*). Recherchez la description complète du langage Scheme telle qu'elle est décrite dans le document officiel "Revised 5 Report".

**Exercice 10** *Recherchez dans l'aide la réponse aux questions suivantes :*

- *Comment calculer une expression comme* $12345^{12345}$*, que l'on calculera (chercher comment calculer les exponentielles).*
- *Quelle est la fonction Scheme qui permet de calculer la partie entière d'un nombre ?*
- *Quelles sont les deux fonctions Scheme qui permettent de calculer la division entière d'un entier par un autre, et le reste de cette division ? En déduire une fonction* `divisible?` *qui teste si son premier argument est divisible par le second.*
- *Comment écrire une conditionnelle (*`cond`*) ? Écrivez une expression dans la* zone de définitions *qui dépend de* $x$ *et qui rend 3 si* $x = 5$*,* $x^2$ *si* $x \geq 8$*, "toto" si* $x = 6$*,* `#t` *si* $x = 7$ *et* `#f` *dans tous les autres cas.*