Programmation Fonctionnelle
Introduction

Gilles Enée – LS4
Université des Antilles Guyane

Pourquoi Scheme ? (skime)

- Vous avez étudié jusqu'à présent la programmation impérative :
  - Vient de la conception d'un programme comme une suite d'instructions.
  - L'ordinateur doit impérativement suivre cette suite d'instructions pour que le programme s'exécute correctement.
  - Cette conception date des débuts de l'informatique et semble au premier abord assez naturelle.

Pourquoi Scheme ?

- Il existe en programmation comme dans la plupart des autres disciplines scientifiques, différentes manières de « voir le monde ».
- La programmation fonctionnelle propose un autre style, plus proche de la pensée mathématique pure.

Pourquoi Scheme ?

- Pour le mathématicien, le concept d'instruction n'existe pas, celui d'affectation encore moins :
  - Avez-vous déjà vu une démonstration où des objets changeraient de valeur entre le début de la preuve et le CQFD ?
  - Quand à celui de boucle, peu de démonstrations s'en servent (en réalité, ce concept existe depuis Euclide mais le mathématicien ne le théorise pas)

Pourquoi Scheme ?

- Qu'emprunte-t-on au mathématicien ?
  - Deux concepts fondamentaux :
    - Les **fonctions** et la possibilité de les composer, la loi o jouant le rôle du début…fin impératif.
    - Le **principe de récurrence**, dont l'itération (boucle tant_que) formera un cas particulier.

Pourquoi Scheme ?

- Vous devrez donc avoir la même rigueur de pensée qu'en algèbre, ni plus ni moins.
- Nous n'allons pas pour autant manipuler systématiquement des objets mathématiques !
- Mais nos schémas de pensée seront au fond les mêmes qu'en mathématiques.

Pourquoi Scheme ?

- AUCUNE CONNAISSANCE PREALABLE DE PROGRAMMATION N'EST NECESSAIRE A CE COURS !
- Ne raisonnez pas en C ou en algorithmique : nous abordons une autre façon de pensée (panser ?)
- Ayez un esprit de débutant.

Une syntaxe étrange venue d'ailleurs !

- Des premières années de l'informatique avec le langage LISP : 1958.
- Scheme date lui de 1980.
- LISP fut conçu à l'origine pour les besoins des théoriciens de la programmation et surtout pour des programmeurs qui souhaitaient un langage suffisamment malléable pour les besoins de l'Intelligence Artificielle naissante (1956).

Une syntaxe étrange venue d'ailleurs !

- La syntaxe de Scheme et LISP est donc très proche.
- Basée sur l'idée de représenter une expression arithmétique sous une ***forme préfixée totalement parenthésée***.

Une syntaxe étrange venue d'ailleurs !

- En C, on écrit sqrt(x+2) pour exprimer la racine carrée de x+2
- En Scheme, nous écrirons de manière « préfixée » : ***l'opérateur sera toujours en tête*** de l'expression, et pour éviter toute ambigüité, nous placerons des parenthèses comme ceci :

(sqrt (+ x 2))

Une syntaxe étrange venue d'ailleurs !

- Donc au lieu d'écrire f(x,y,z), nous écrirons (f x y z) sans virgules
- Le même processus de traduction s'opérant à tous les niveaux, c'est-à-dire dans les sous-expressions x, y et z.
- Dans une expression bien formée, il y aura donc autant de parenthèses ouvrantes que de parenthèses fermantes.

Une syntaxe étrange venue d'ailleurs !

- Maths
  - 1+2+3
  - 1-2+3
  - sin(ωt+φ)
  - x+2y=0
  - 2.71828
  - 3/2
  - 3-2i
  - x -> x+2
  - f o g o h
  - si x=2 alors 3 sinon y+1
- Scheme
  - (+ 1 2 3)
  - (+ 1 -2 3) ou (+ (- 1 2) 3)
  - (sin (+ (* omega t) phi)
  - (= (+ x (* 2 y)) 0) ou (zero? (+ x (* 2 y))
  - 2.71828 réel approché
  - 3/2 pas une opération mais un rationnel
  - 3-2i pas une opération mais un complexe
  - (lambda (x) (+ x 2)) la fonction « x a pour image x+2 »
  - (compose f g h) la composition de fonction
  - (if (= x 2) 3 (+ y 1))

Une syntaxe étrange venue d'ailleurs !

- Voilà ! Vous savez tout sur la syntaxe de SCHEME : l'art du bon balancement des parenthèses…
- Notation compliquée pour des problèmes simples mais qui simplifie la vie des programmeurs pour les problèmes compliqués.

Mise en jambe

- Traduire en notation Scheme l'expression

$$\sin(x+y/3) \geq a$$

- Traduire en notation mathématique usuelle l'expression Scheme suivante, dans laquelle log désigne la fonction « logarithme népérien » et abs… vous devinez quoi !

```
(/ (log (- x 2)) (+ 1 (abs (- x 4))))
```

Mise en jambe

- Trouvez une erreur de syntaxe dans l'expression Scheme suivante :

  (* 3 sqrt(y + 2))

Dr Scheme

- Ce logiciel à l'origine d'universitaires des Etats-Unis (Brown University, Providence, RI, Northeastern University, Boston, MA, University of Chicago, Chicago, IL, University of Utah, Salt Lake City, UT) est disponible gratuitement pour toutes les plateformes dans sa version 360.

http://www.plt-scheme.org/

Dr Scheme

- Dr Scheme est un gros système dont vous n'exploiterez pas toutes les possibilités.
- Il a été conçu à la fois pour l'enseignement, la recherche et le développement de gros logiciels en Scheme.
- Il est lui-même écrit en Scheme !

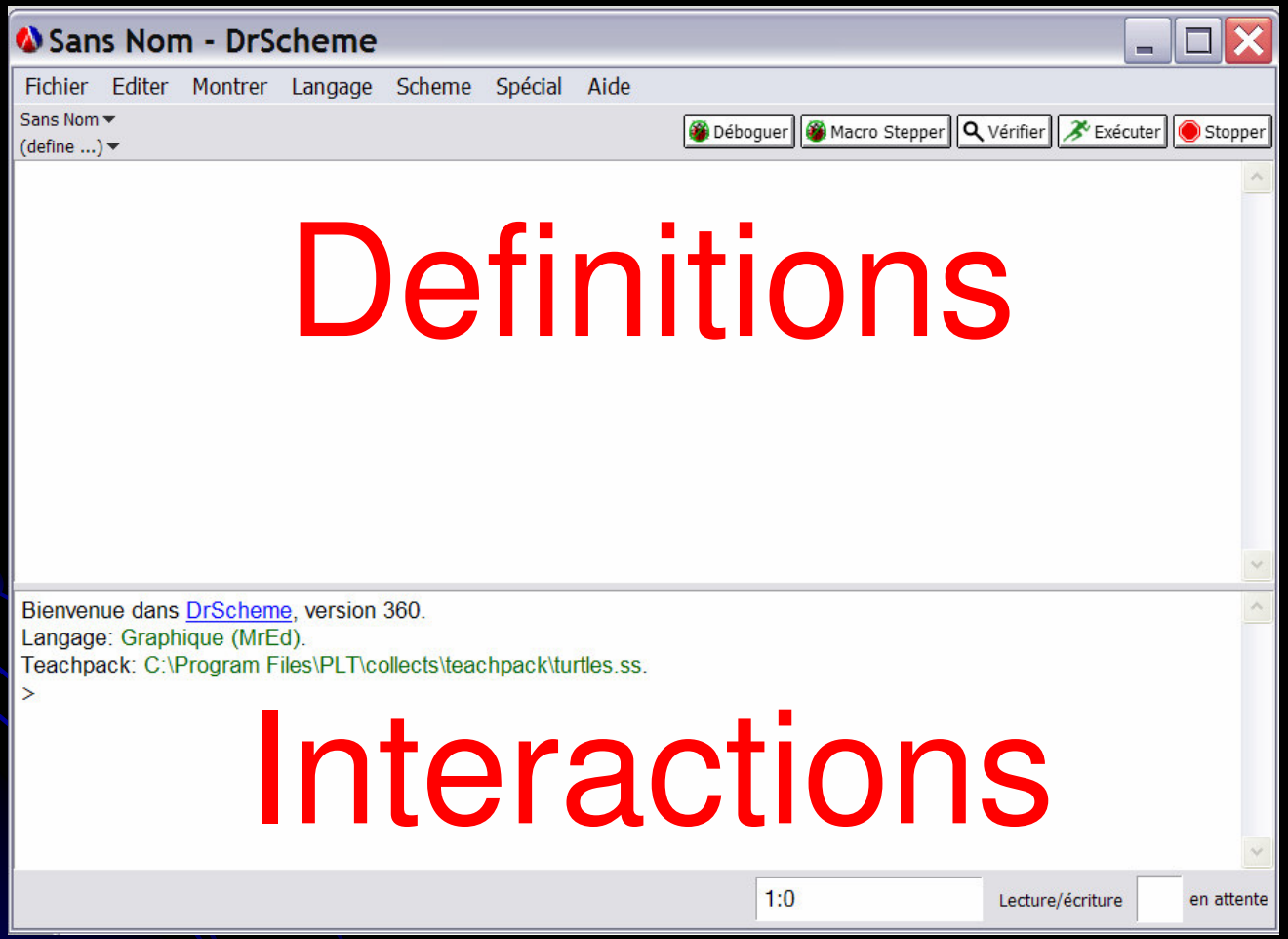
Dr Scheme

- Il dispose d'un compilateur, d'une couche objet de type Java, de modules pour la compilation séparée et de bibliothèques.
- Le compilateur est omniprésent mais de manière transparente (on parle d'interpréteur)

Dr Scheme

- Le logiciel est composé de deux sous-fenêtres :
  - En haut les définitions de votre programme (donc un programme Scheme est un ensemble de définitions) avec un éditeur coloré
  - En bas les interactions avec Dr Scheme. Vous pourrez y faire vos calculs et tester vos programmes au « toplevel »

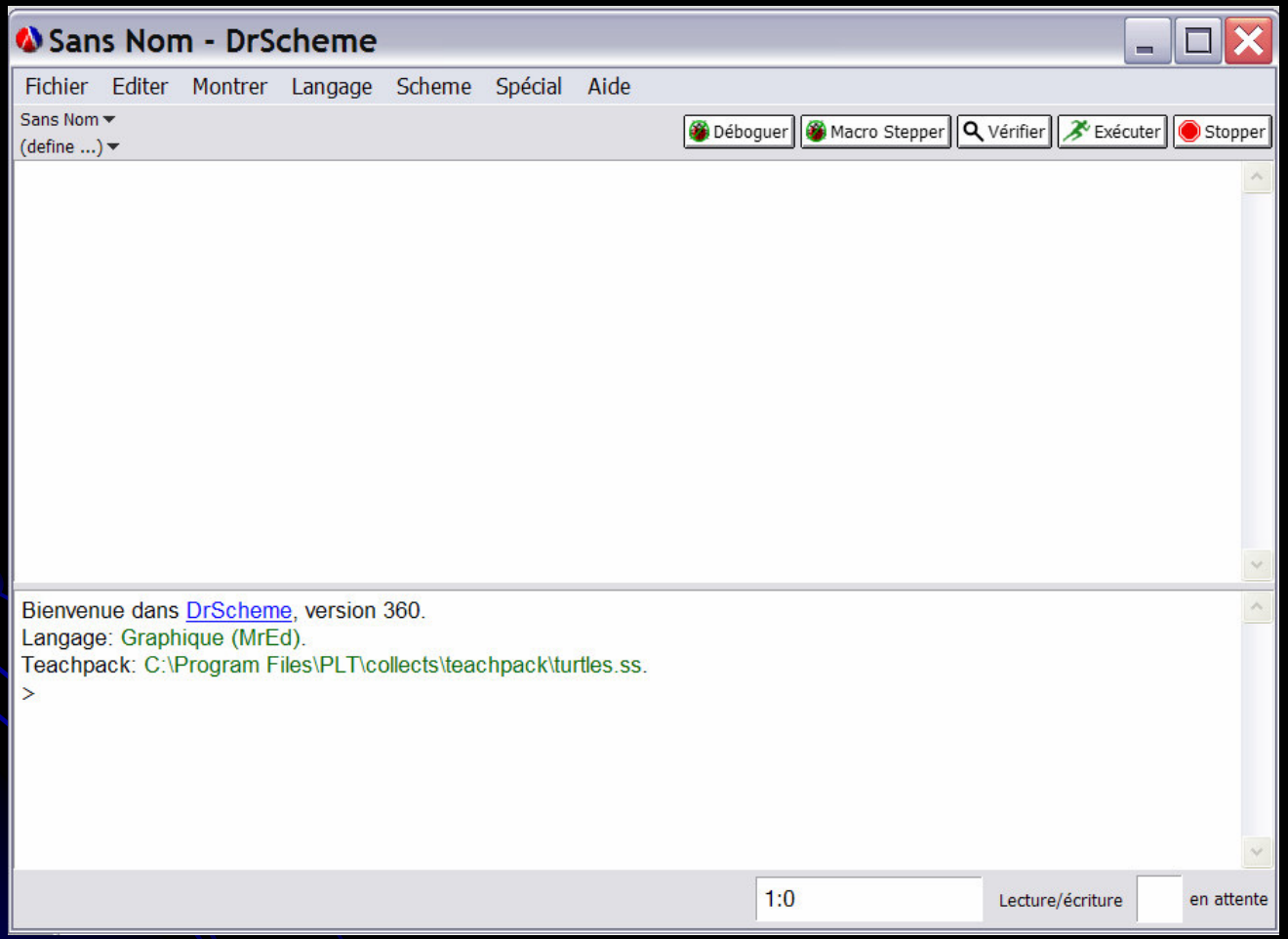
Dr Scheme


Sans Nom - DrScheme
Fichier Editer Montrer Langage Scheme Spécial Aide
Sans Nom
(define ...)
Déboguer
Macro Stepper
Vérifier
Exécuter
Stopper
Definitions
Bienvenue dans DrScheme, version 360.
Langage: Graphique (MrEd).
Teachpack: C:\Program Files\PLT\collects\teachpack\turtles.ss.
>
Interactions
1:0
Lecture/écriture
en attente

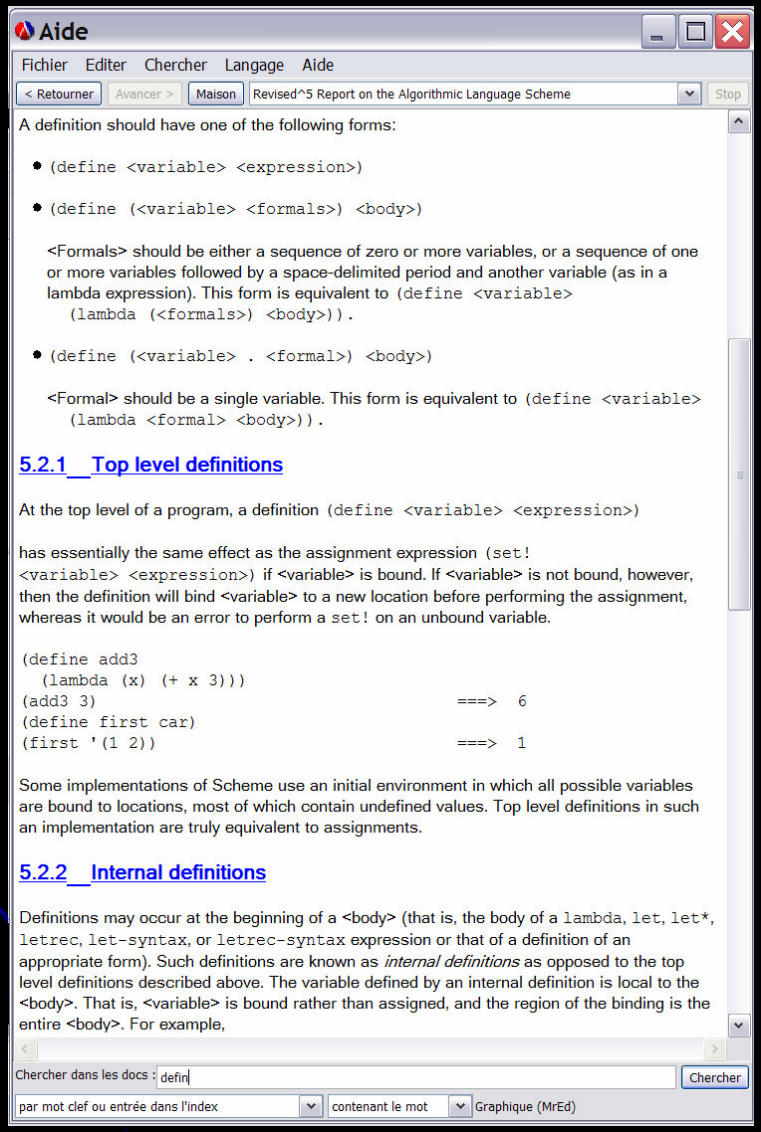
Dr Scheme

- Vous trouverez ensuite :
  - Un menu classique de gestion de fichiers.
  - Un menu d'édition un peu plus étoffé.
  - Un bouton « vérifier » pour contrôler la syntaxe de vos programmes.
  - Un bouton « exécuter » pour compiler et exécuter toutes vos définitions.
  - Un bouton « déboguer » pour faire une trace de l'exécution de vos programmes.

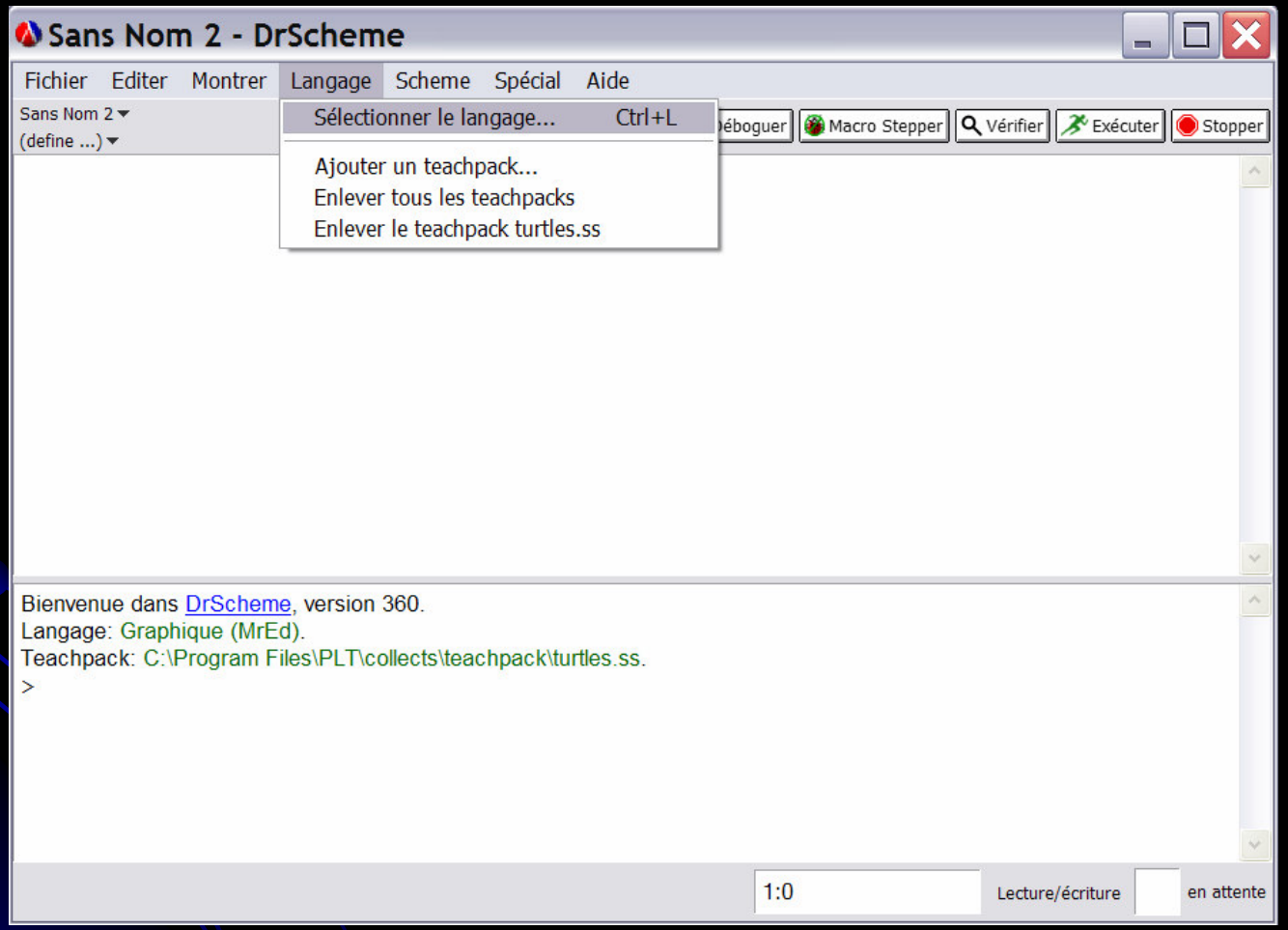
Dr Scheme


Sans Nom - DrScheme
Fichier
Editer
Montrer
Langage
Scheme
Spécial
Aide
Sans Nom
(define ...)
Déboguer
Macro Stepper
Vérifier
Exécuter
Stopper
Bienvenue dans DrScheme, version 360.
Langage: Graphique (MrEd).
Teachpack: C:\Program Files\PLT\collects\teachpack\turtles.ss.
>
1:0
Lecture/écriture
en attente

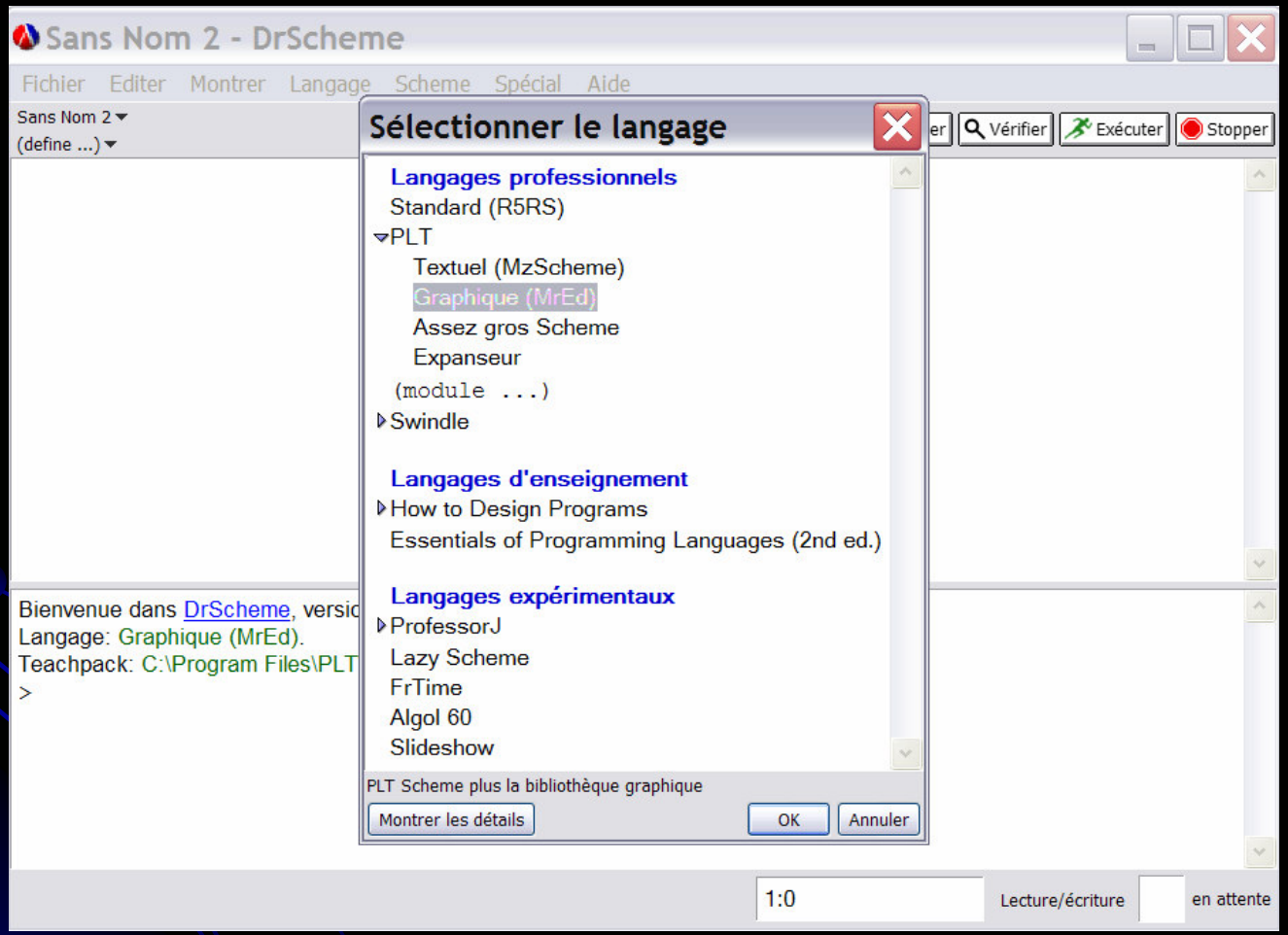
Dr Scheme : Un outil indispensable



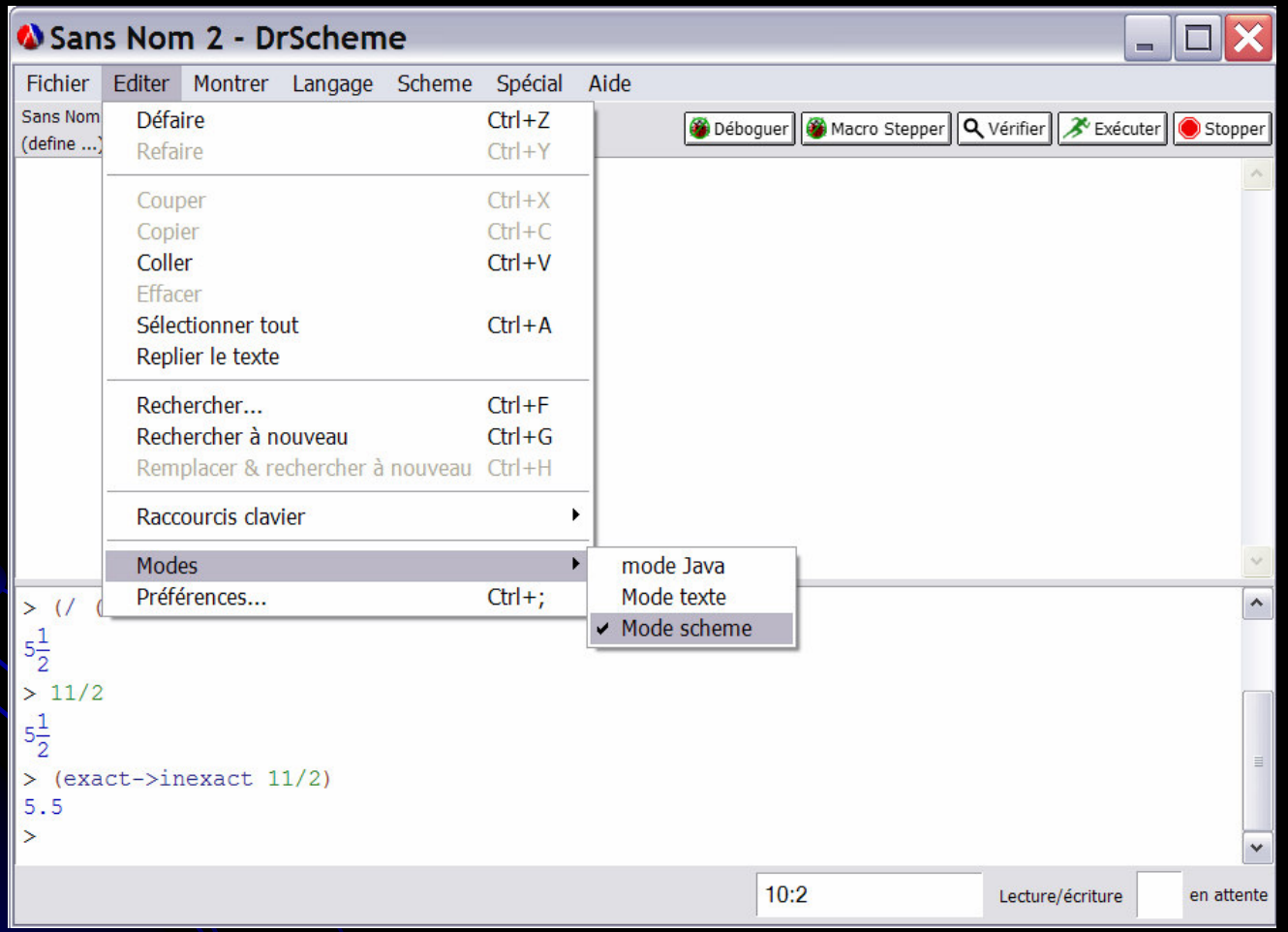
Dr Scheme : A faire



Dr Scheme : A faire



Dr Scheme : A faire



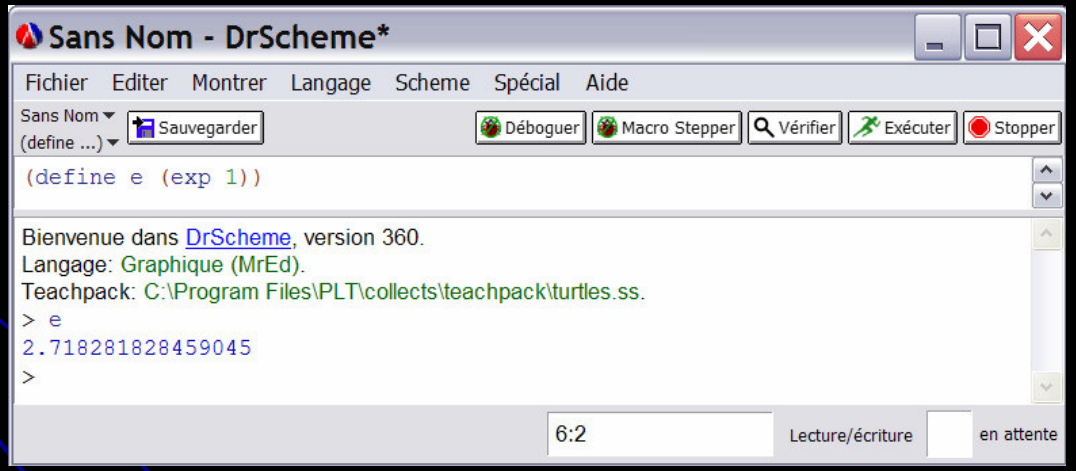
Dr Scheme

- Premières définitions
  - Programme = ensemble de définitions.
  - Une définition a la forme : (define symb expr)
  - Où symb est un « symbole », c'est-à-dire un mot qui ne soit pas une primitive (un identificateur !), et expr, une expression par exemple numérique

Dr Scheme

- Par exemple

(define e (exp 1))



Dr Scheme

- Pour exprimer en Scheme la fonction
  - x -> x+2
  - On écrira

```
(lambda (x) (+ x 2))
```

  - (x,y) -> x+2y
  - On écrira

```
(lambda (x y) (+ x (* 2 y)))
```

Dr Scheme

- De manière générale :

  (lambda (x1 … xn) expr)

  - Où x1 … xn sont les paramètres (n>=0)
  - Et expr l'expression formant le corps de la fonction.

Dr Scheme

- Il existe deux styles de définitions
  - Le style Indiana que nous venons de voir :

```
(define f
    (lambda (x) (+ x 2)))
```

  - Le style MIT qui vous plaira peut-être plus :

```
(define (f x)
    (+ x 2))
```

Dr Scheme

- Ecrivez en style Indiana et MIT, les fonctions suivantes :
  - x -> 3
  - x -> x*x
  - (x, y) -> a*x + b*y + c
  - La fonction « rond » qui compose deux fonctions à un paramètre