sp-5. リスト，シンボル，文字列

（Scheme プログラミング）

URL: https://www.kkaneko.jp/pro/scheme/index.html

金子邦彦

アウトライン

5-1 リストとは

5-2 Scheme プログラムでのリストの記法

　　list 句

5-3 リストに関する演算子

　　first, rest, empty?, length, list-ref, append

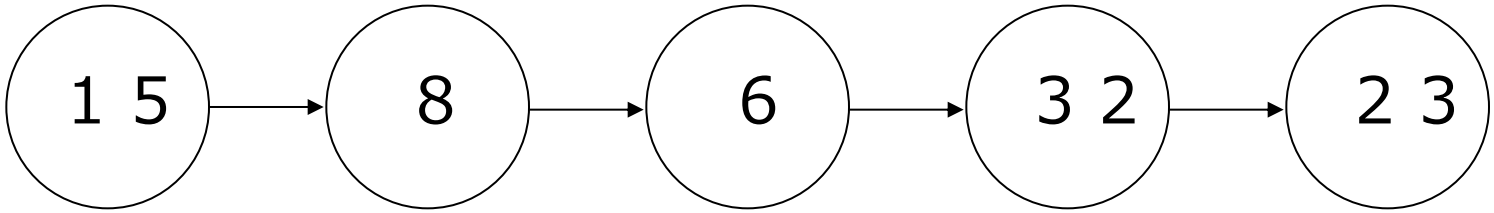
5-4 数字，シンボル，文字列を含むリスト

5-5 パソコン演習

5-6 課題

5-1 リストとは

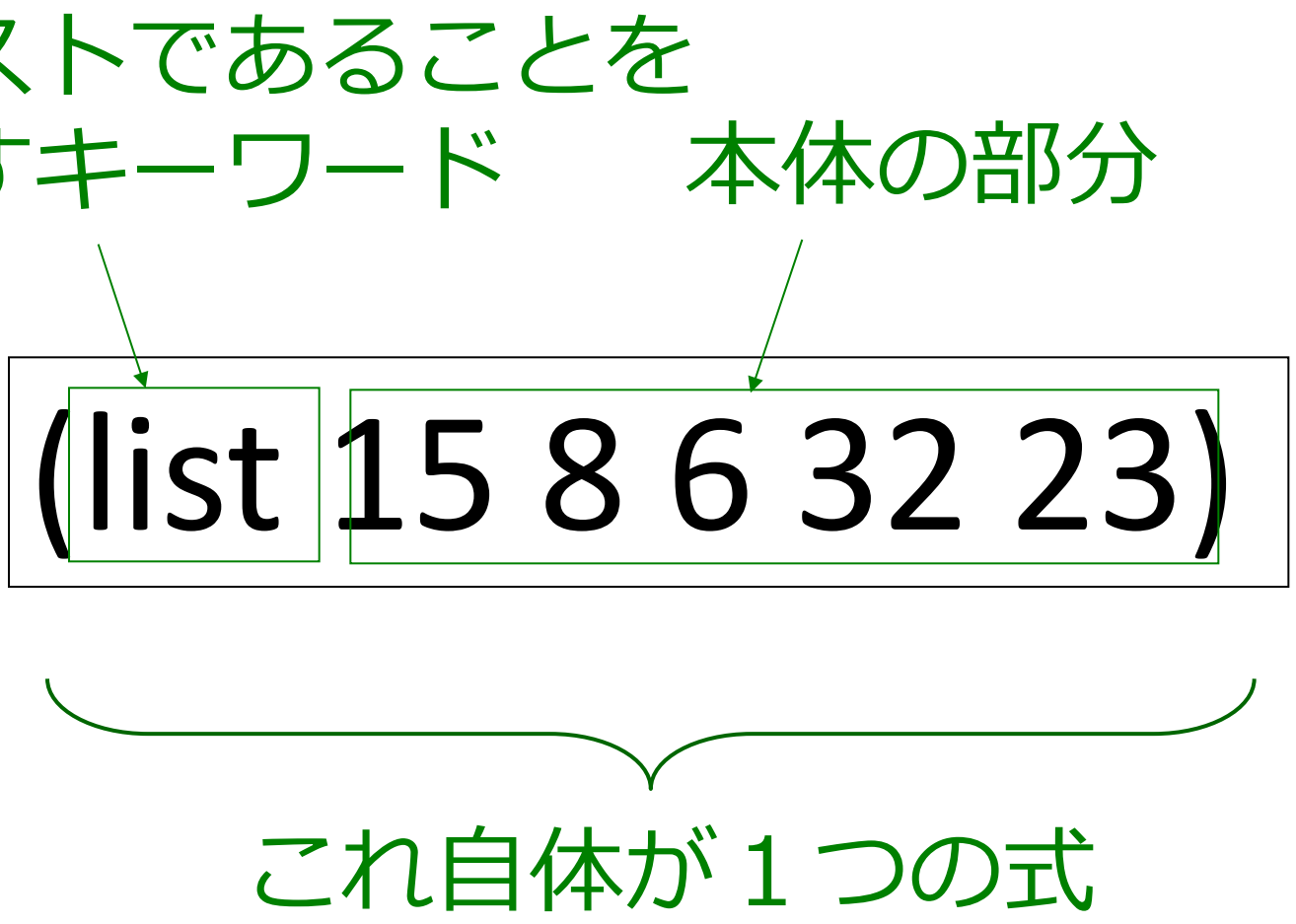
リストとは



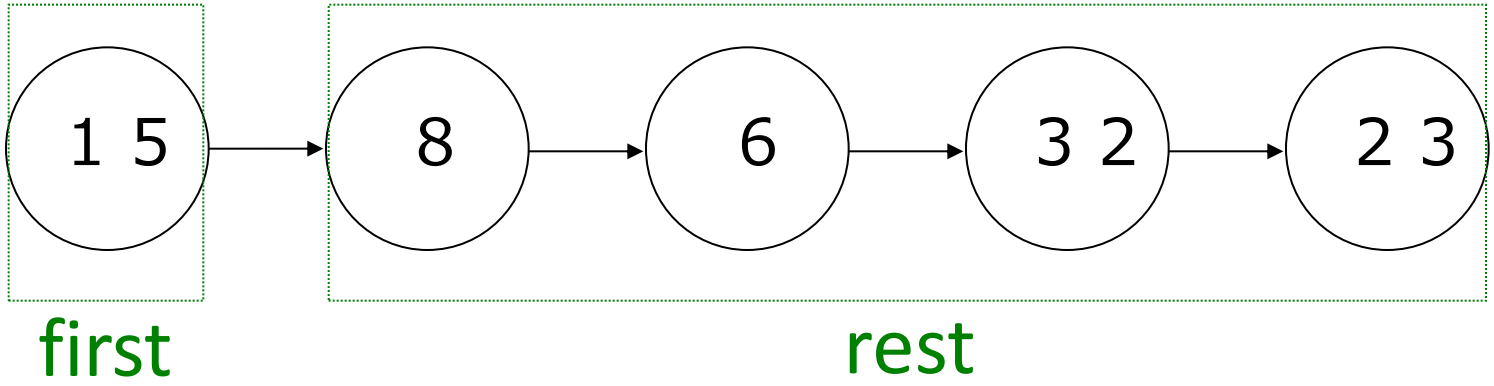
データの並び
データに順序がある

5-2 Scheme でのリストの表記

Scheme でのリストの記法



first と rest



- first
  - 先頭のこと
- rest
  - 先頭を取り除いた残り
  - rest もまたリスト

例）　上のリストの rest の rest の rest の first は：　3
　2

empty

- empty は「空リスト」を表す特別な記号
- rest との関係
  - リストの長さが 1 の時には，empty

例)

(rest (list 15))

= empty

first, rest に関する実行エラー

- 要するに，「空でないリスト」だけ，first, rest が実行できる

	空で無いリスト	空リスト	数値
first	OK	実行エラー	実行エラー
rest	OK	実行エラー	実行エラー

first, rest に関する実行エラーの例

```
> (first (list 10))
10
> (first empty)
first: expects argument of type <non-empty list>; given empty
> (first 10)
first: expects argument of type <non-empty list>; given 10
> (rest (list 10))
empty
> (rest empty)
rest: expects argument of type <non-empty list>; given empty
> (rest 10)
rest: expects argument of type <non-empty list>; given 10
```

DrScheme の実行画面

empty?

- empty? は，リストが空リストであるかを調べる

```
> (empty? (list 11 12 13))
false
> (empty? empty)
true
> (empty? 10)
false
```

- empty? の意味：
  - 「空リスト」ならば true （さもなければ false）

リストに関するキーワード

- リストに関する演算子
  - first　　リストの先頭の要素
    （例） (first (rest (rest a-list)))
  - rest　　リストから先頭の要素を除いた残り
    （例） (rest a-list)
  - empty?　リストが空リストであるか調べる
    （例） (empty? a-list)
- list
  - list　　リストを記述するためのキーワード
    （例） (list 15 8 6 32 23)

5-3 リストに関する演算子

リストに関する演算子

- length

  リストの要素の個数

- list-ref

  リストのｎ番目の要素（先頭は０番目）

- append

  リストの連結

length

```
> (length (list 11 12 13))
3
> (length (list 11 12))
2
> (length empty)
0
> (length 10)
length: expects argument of type <proper
list>; given 10
```

リストで無い場合には
実行エラーとなる

- length の意味：
  - リストの要素の個数

list-ref

```
> (list-ref (list 11 12 13 14) 3)
14
> (list-ref (list 11 12 13 14) 0)
11
> (list-ref (list 11 12 13 14) 20)
list-ref: index 20 too large for list:
(list 11 12 13 14)
> (list-ref 10 0)
list-ref: index 0 too large for list (not a
proper list): 10
```

指定した番号が大きすぎると実行エラーとなる

リストで無い場合には実行エラーとなる

- list-ref の意味：
  - リストのｎ番目の要素（先頭は０番目）

append

```
> (append (list 1 2 3) (list 4 5 6))
(list 1 2 3 4 5 6)
> (append (list 11 12 13) empty)
(list 11 12 13)
> (append empty (list 4 5 6))
(list 4 5 6)
> (append 10 20)
append: last argument must be of type
<list>, given 20; all args: (10 20)
```

リストで無い場合には
実行エラーとなる

- append の意味：
  - リストの連結

5-4 数字，シンボル，文字列を含むリスト

シンボル

- 「記号」や「単語」を表す
- カッコ、ダブルクオーテーションマーク，空白，コンマはシンボルとして使えない

(例）　'the

'a

'cat!

'tow^3

'and%so%on

- 「'」が無いと，変数名の意味になる

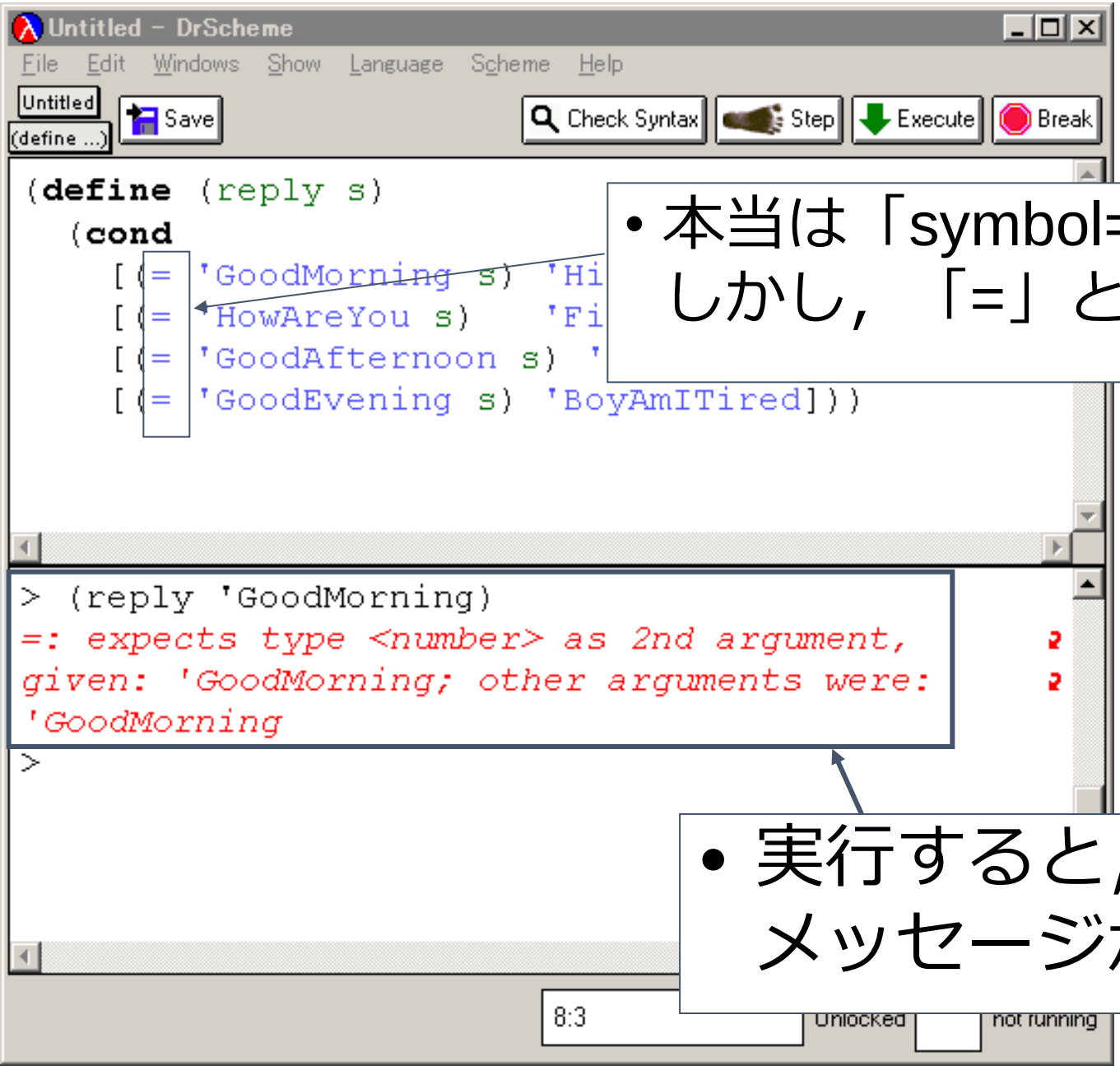
symbol=? の意味

- symbolの比較演算子

(symbol=? 'Hello 'Hello) → true

(symbol=? 'Hello 'Hallo) → false

よくある間違い



- 本当は「symbol=?」と書くべき．しかし，「=」と書いている．

- 実行すると，エラーメッセージが出る

文字列

- 文字列：　「"」で囲む

  "This is a string"

- 空白文字なども「文字列」として使える

さまざまな比較演算

- 数値同士の比較

  <, <=, =, >=, > など

- シンボル同士の比較

  symbol=? など

- 文字列同士の比較

  string<?, string<=?, string=?, string>=?, string>? など

Schemeの式

- atomic 式
  - 数値
  - true, false 値
  - 変数名
  - シンボル
  - 文字列
  - empty,

  など
- 括弧の入った式 (compound)

  (演算子 式の並び)　(関数名 式の並び)

  (cond [(条件式) 式] ...)　(list 式の並び)

  など のパターン
- 関数の定義式

  (define (関数名 変数の並び)
  　式)

  のパターン

Scheme の式

- 数値：
  5, -5, 0.5 など
- true, false 値
  true, false
- シンボル，文字列
- 変数名
- empty
- 四則演算子：
  +, -, *, /
- 比較演算子
  <, <=, >, >=, =
- 奇数か偶数かの判定
  odd?, even?
- 論理演算子
  and, or, not
- リストに関する演算子
  first, rest, empty?, length, list-ref, append
- その他の演算子：
  remainder, quotient, max, min,
  abs, sqrt, expt, log, sin, cos, tan
  asin, acos, atan など

- 括弧
  (,), [,]
- 関数名
- define
- cond
- list

5-5 パソコン演習

パソコン演習の進め方

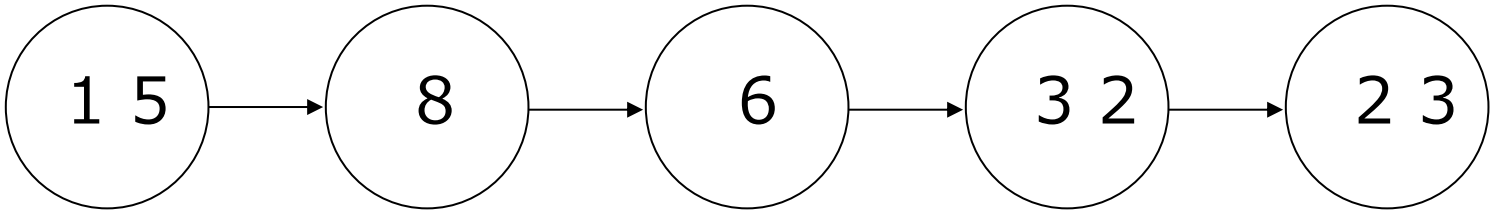
- 資料を見ながら，「例題」を行ってみる
- 各自，「課題」に挑戦する
  - 各自で自習　＋　巡回指導
- 自分のペースで先に進んで構いません

DrScheme の使用

- DrScheme の起動
  プログラム　→ PLT Scheme → DrScheme
- 今日の演習では「Intermediate Student」に設定
  Language
  → Choose Language
  → Intermediate Student
  → Execute ボタン

例題１．リストの式

- リストの式を書く
  - リストの式を書くために 「list」 を使う



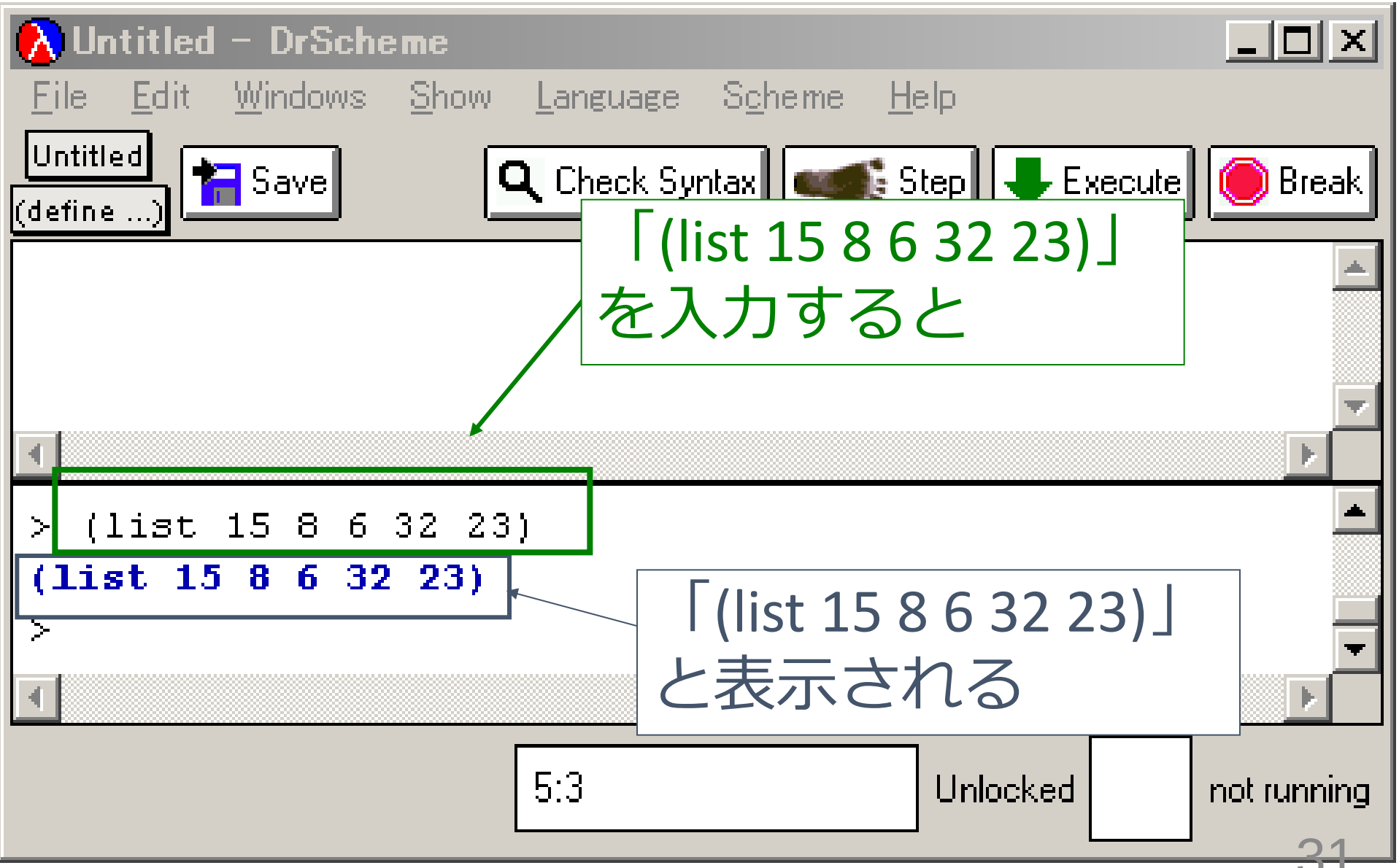
「例題１．リストの式」の手順

1. 次の式を「実行用ウインドウ」で，
   実行しなさい

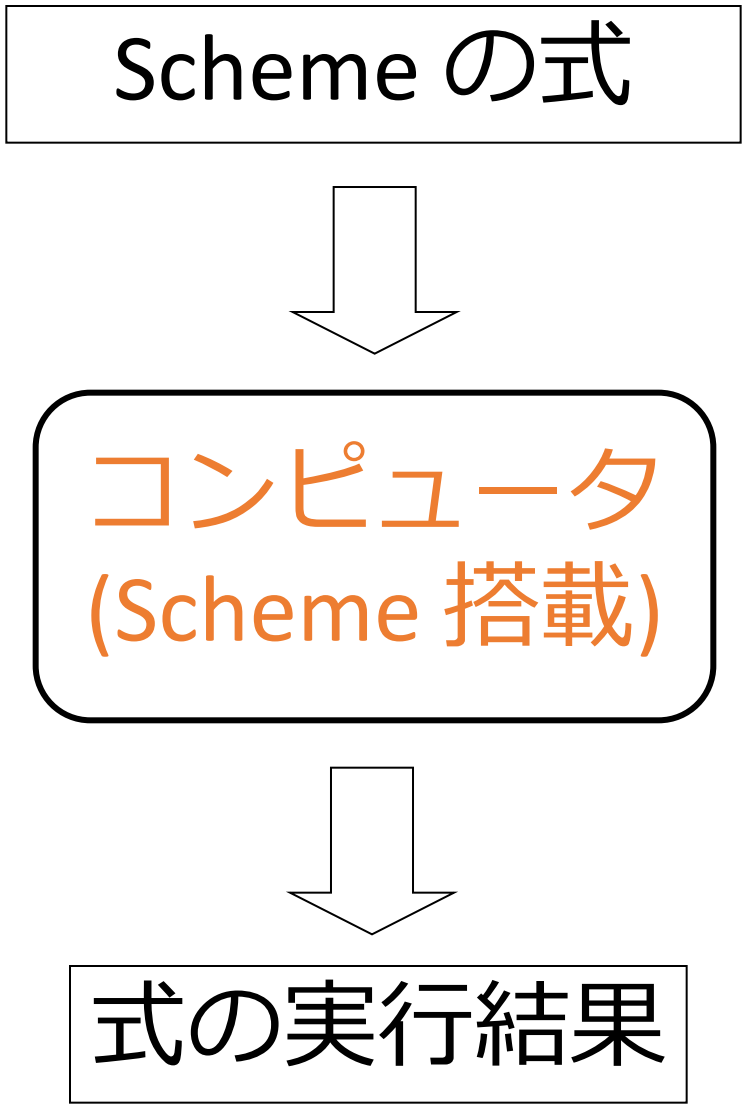
```
(list 15 8 6 32 23)
```

☆　次は，例題２に進んでください

実行結果の例


Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Save
Check Syntax
Step
Execute
Break
「(list 15 8 6 32 23)」
を入力すると
> (list 15 8 6 32 23)
(list 15 8 6 32 23)
>
「(list 15 8 6 32 23)」
と表示される
5:3
Unlocked
not running

コンピュータが行っていること



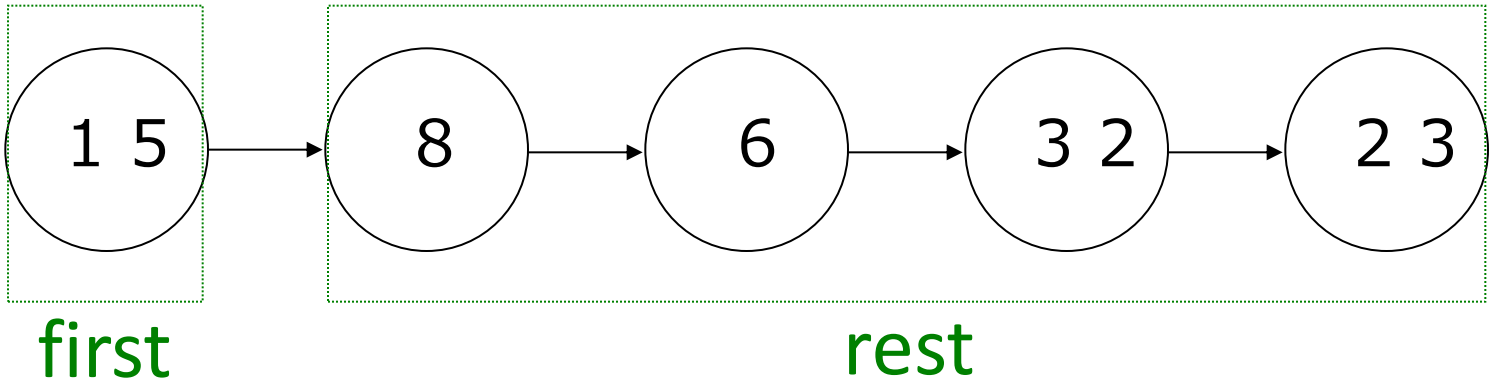
(list 15 8 6 32 23)
を入力する
と・・・

(list 15 8 6 32 23)
がそのまま表示される

例題２．リストの first と rest

- リスト (list 15 8 6 32 23) に対して，first と rest を実行する

例）



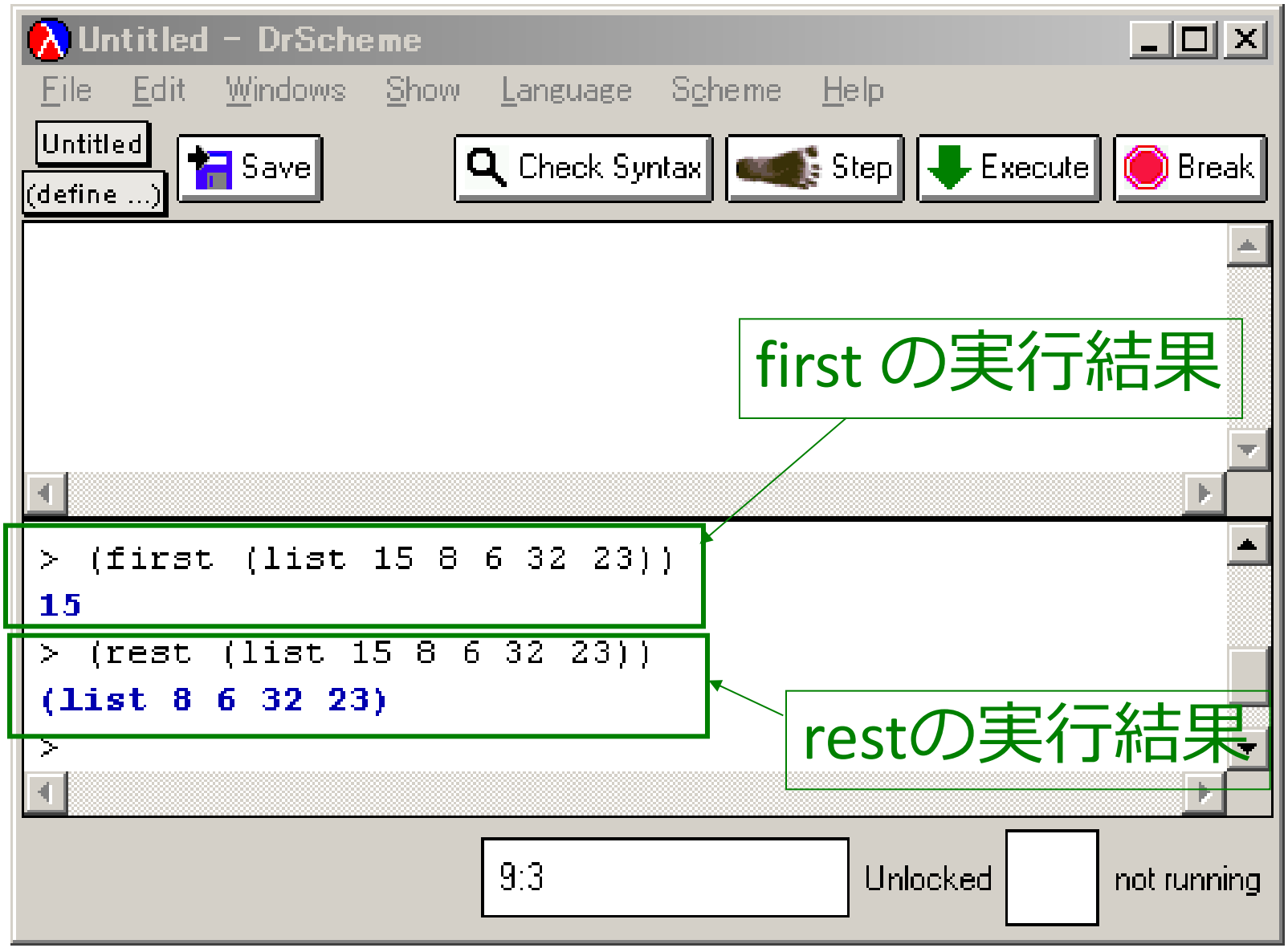
「例題２．リストの first と rest」の手順

1. 次の式を「実行用ウインドウ」で，実行しなさい

```
(first (list 15 8 6 32 23))
(rest (list 15 8 6 32 23))
```

☆　次は，例題３に進んでください

実行結果の例

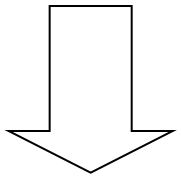


first の実行結果

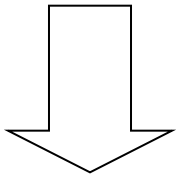
restの実行結果

first

Scheme の式



コンピュータ
(Scheme 搭載)



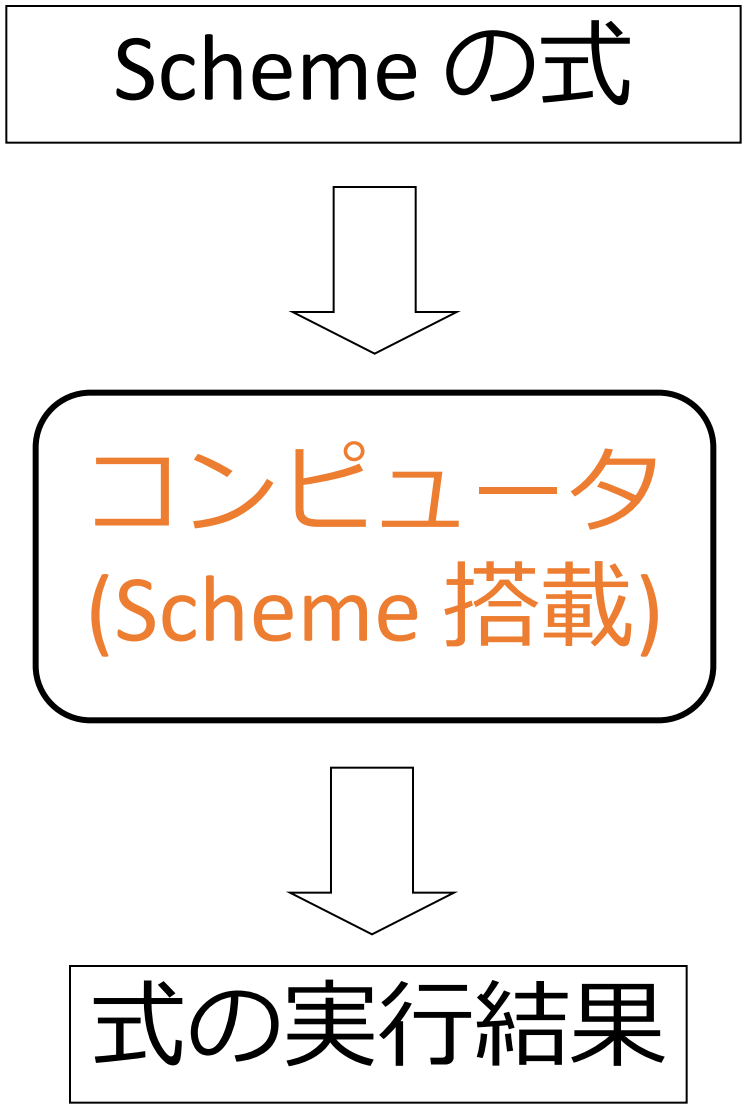
式の実行結果

(first (list 15 8 6 32 23))
を入力すると・・・

15
が表示される

リストの先頭の要素

rest



(rest (list 15 8 6 32 23))
を入力すると・・・

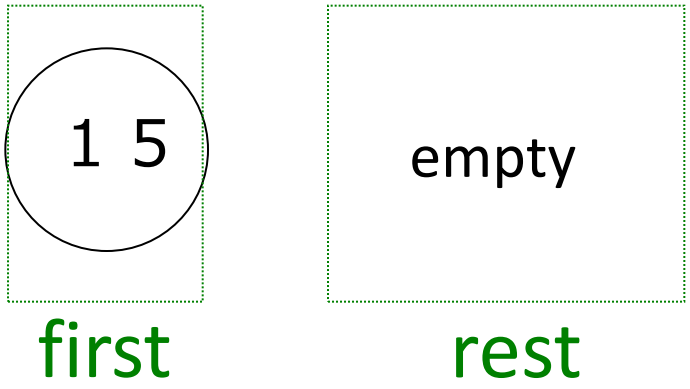
(list 8 6 32 23)
が表示される

リストから先頭の要素を除いた残り

例題３．リストの first と rest

- 要素が１つしか無いリスト (list 15) に対して，first と rest を実行する

例）



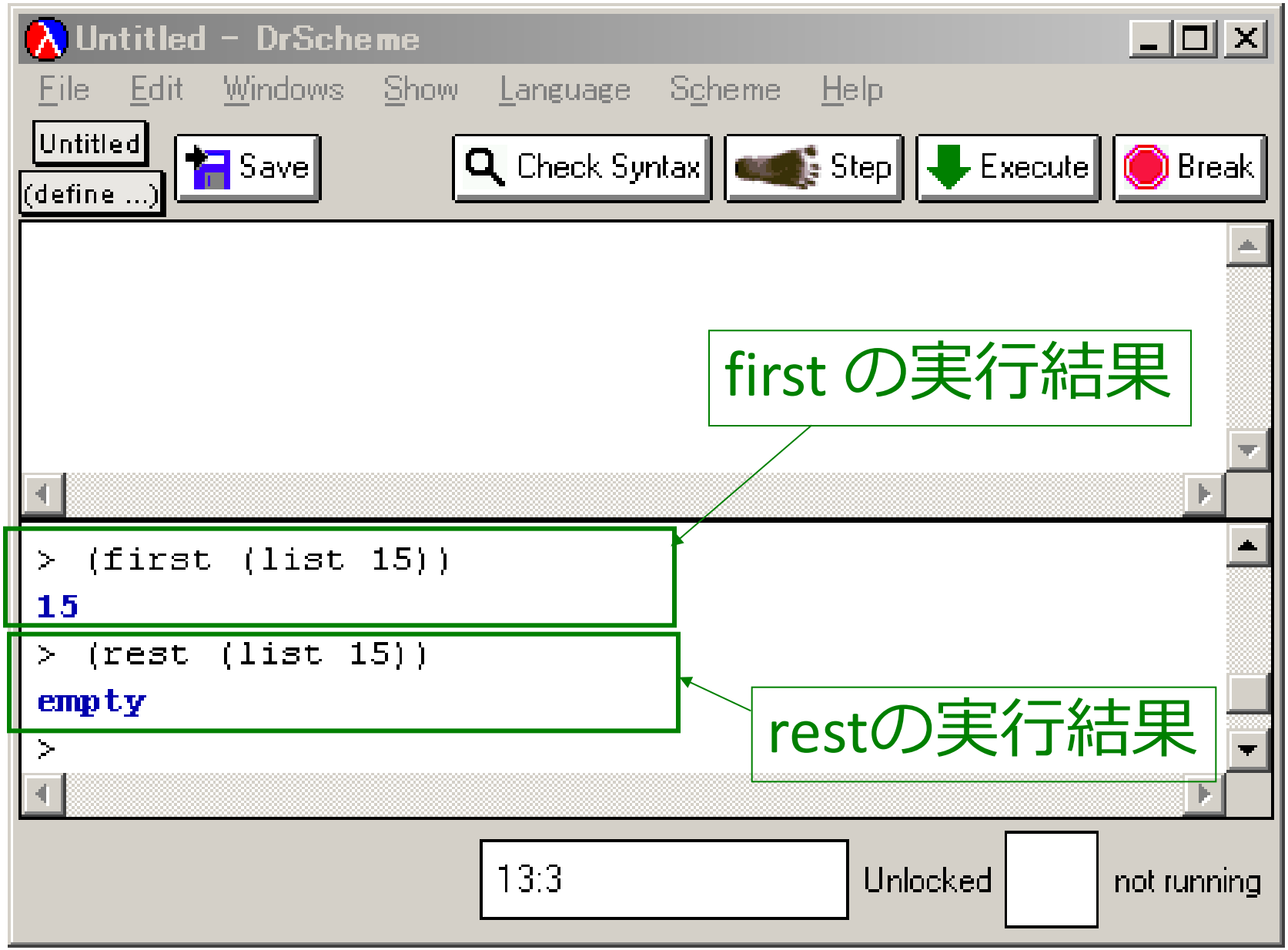
「例題３．リストの first と rest」の手順

1. 次の式を「実行用ウインドウ」で，実行しなさい

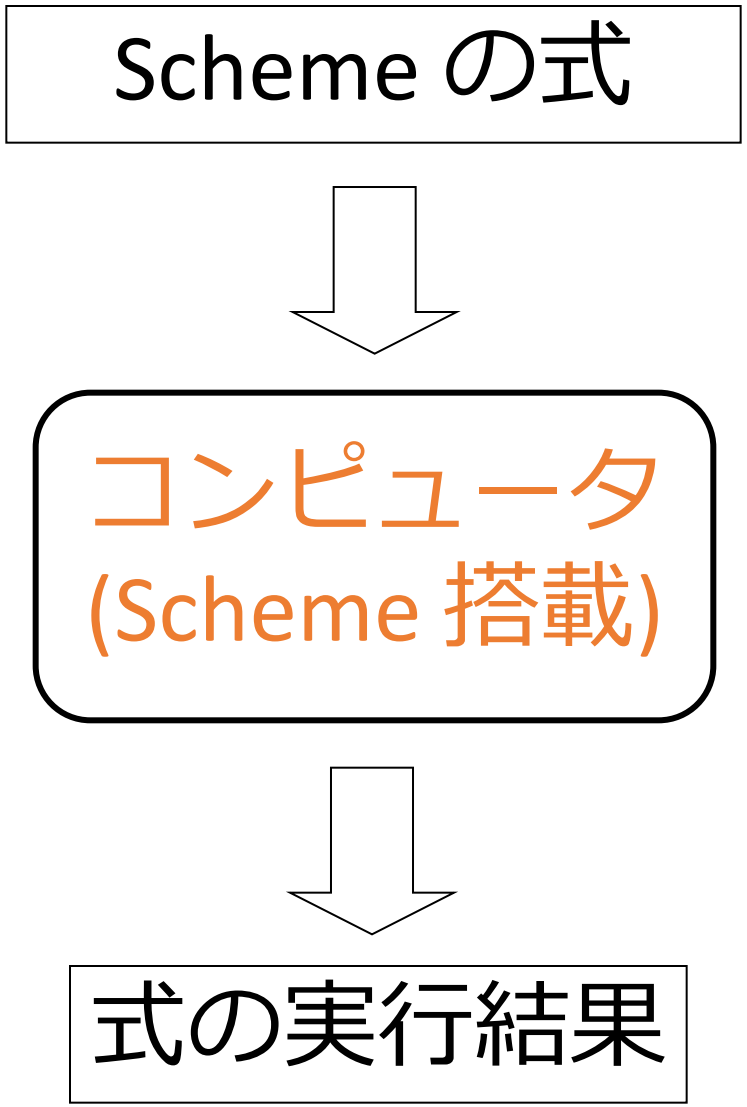
```
(first (list 15))
(rest (list 15))
```

☆　次は，例題４に進んでください

実行結果の例


Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Save
Check Syntax
Step
Execute
Break
first の実行結果
> (first (list 15))
15
> (rest (list 15))
empty
>
restの実行結果
13:3
Unlocked
not running

first

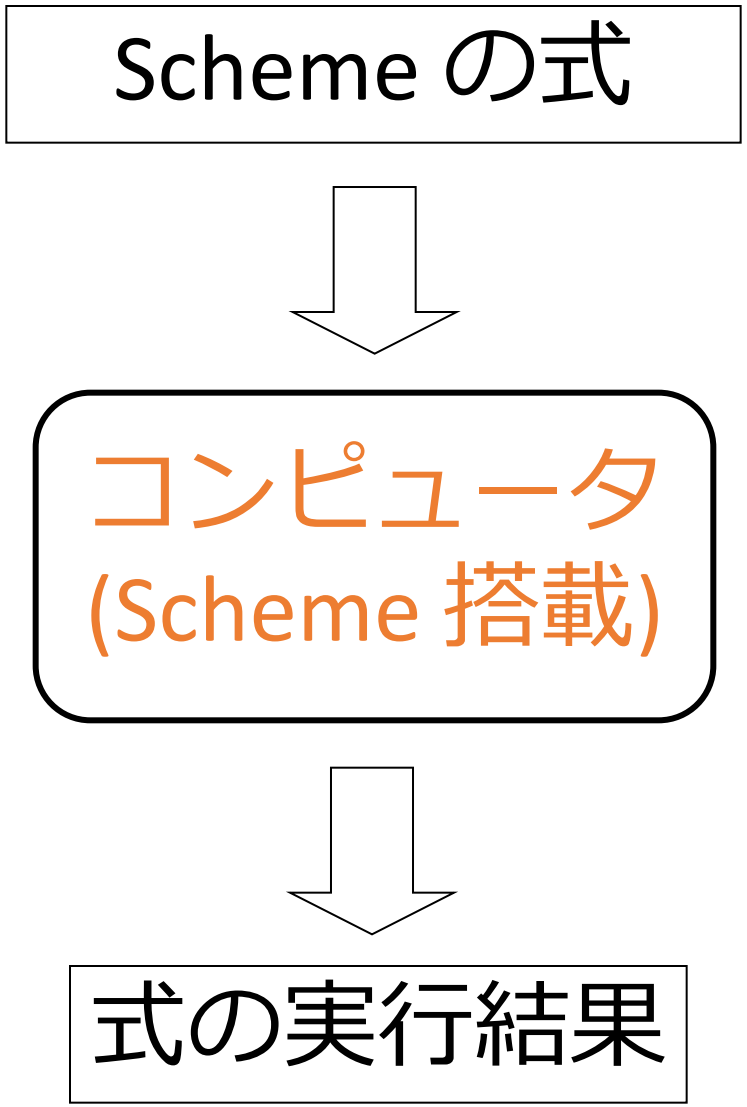


(first (list 15))
を入力すると・・・

15
が表示される

リストの先頭の要素

rest



(rest (list 15))
を入力すると・・・

empty
が表示される

リストから先頭の要素を除いた残り

例題４．append

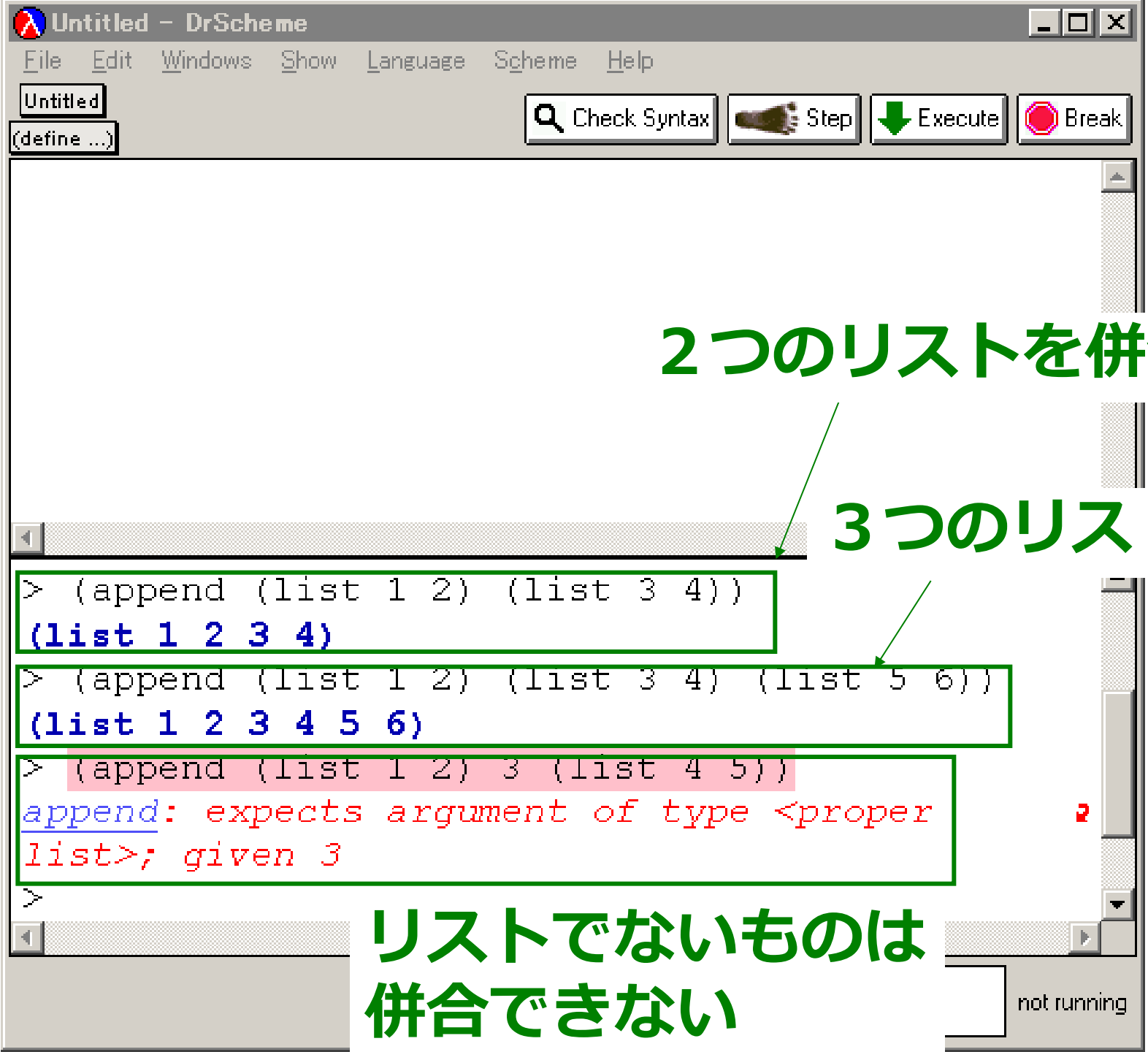
- リストをつなげる関数 append を使ってみる
  - append は Scheme が備えている関数

「例題４． append 」の手順

1. 次の式を「実行用ウインドウ」で，実行しなさい

```
(append (list 1 2) (list 3 4))
(append (list 1 2) (list 3 4) (list 5 6))
(append (list 1 2) 3 (list 4 5))
```

☆　次は，例題５に進んでください



２つのリストを併合

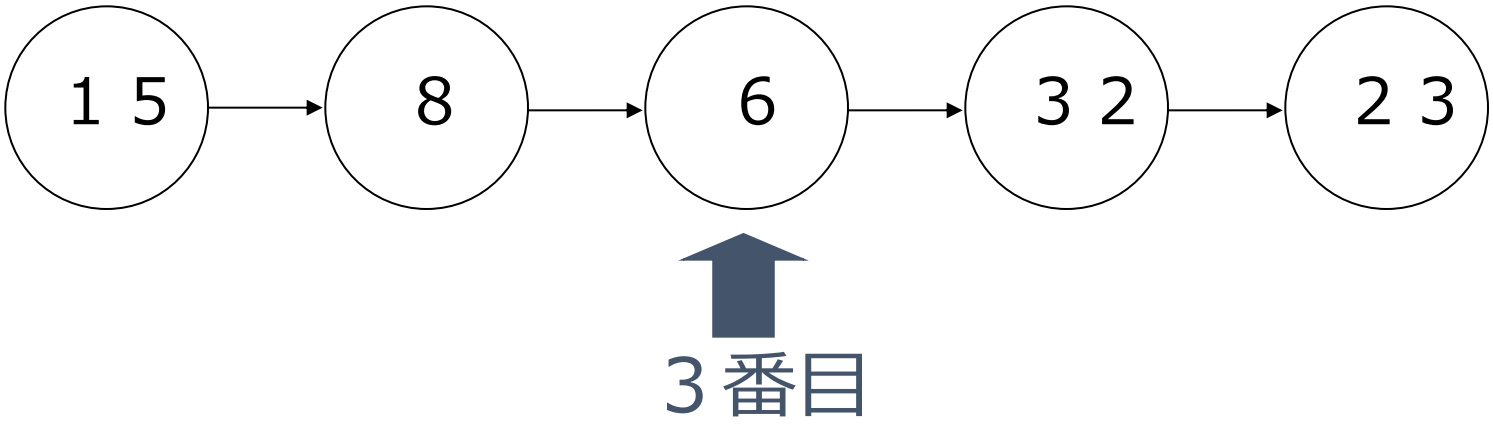
３つのリストを併合

リストでないものは
併合できない

例題５．リストの基本操作

- リストの３番目の要素を得る関数 element3 を作り，実行する
  - first, rest の組み合わせ

例）

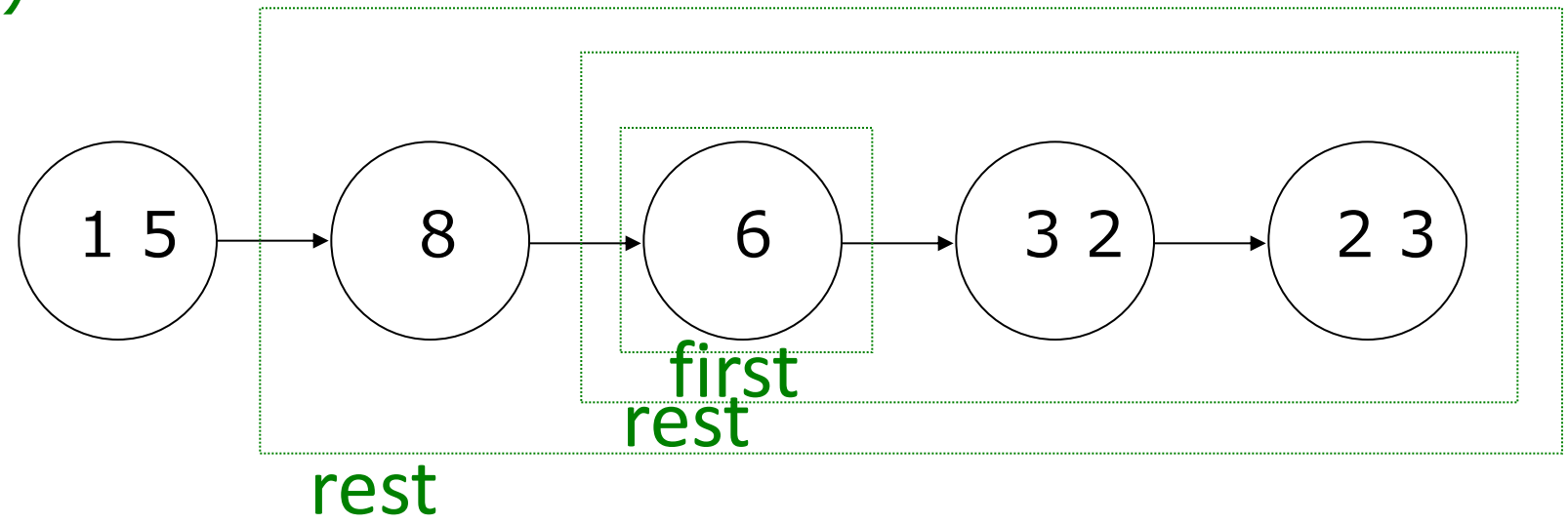


リストの基本操作

- リストの３番目

  ＝ リストの rest の rest の first

例）



「例題５．リストの基本操作」の手順

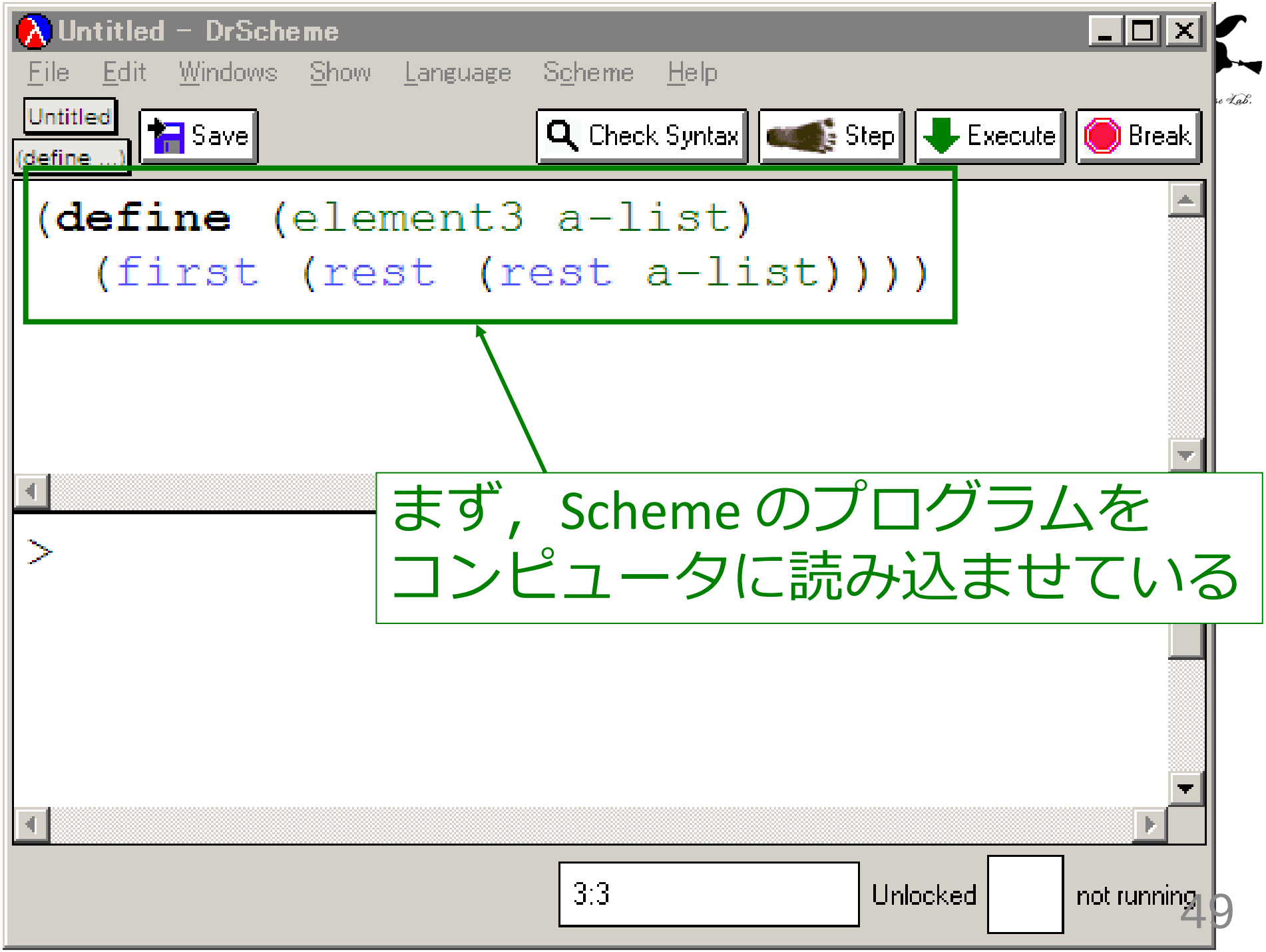
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Execute ボタンを押す

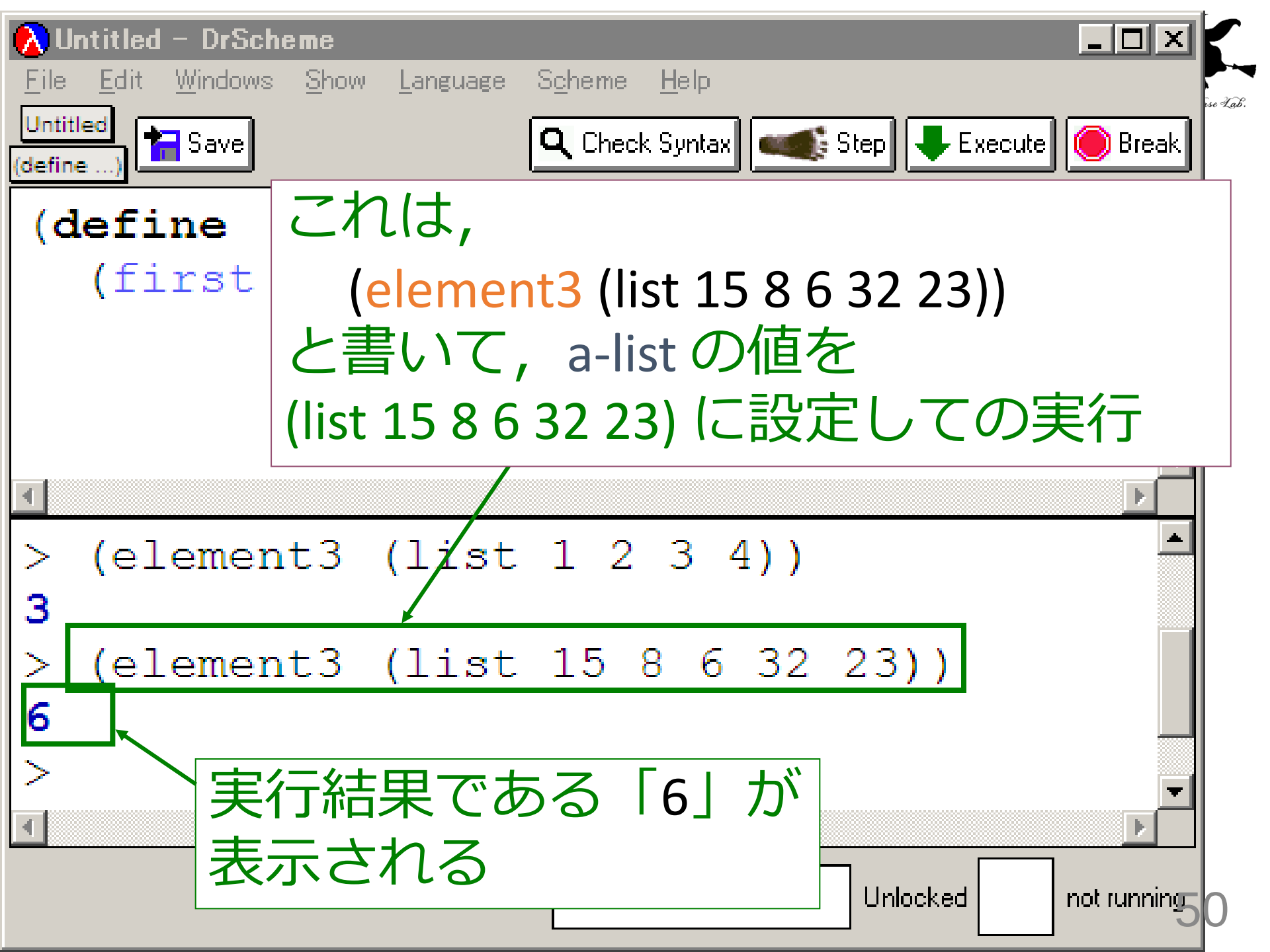
```
(define (element3 a-list)
   (first (rest (rest a-list))))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(element3 (list 1 2 3 4))
(element3 (list 15 8 6 32 23))
```

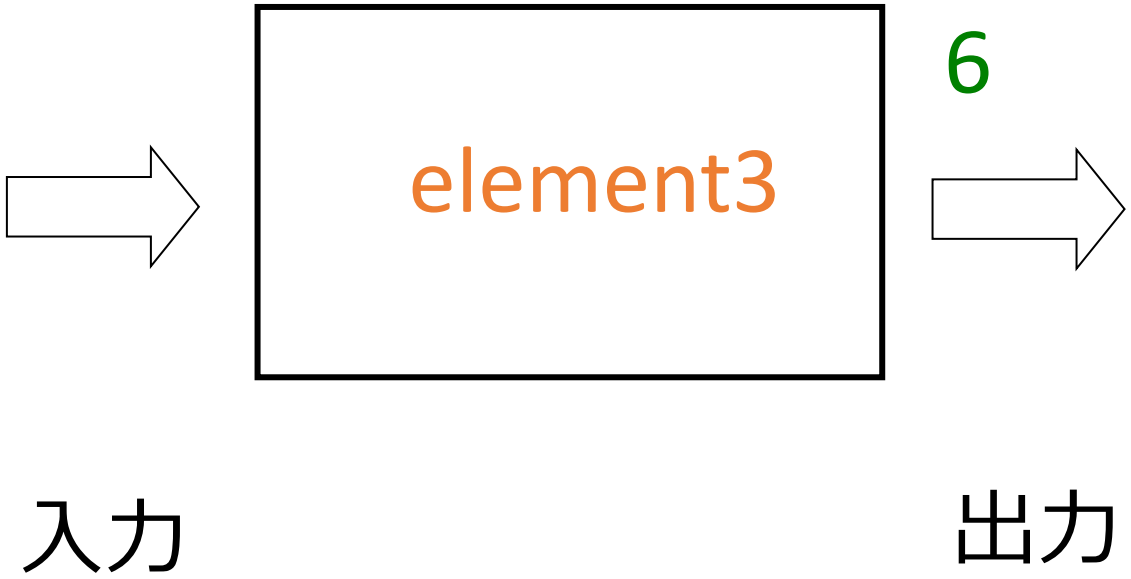
☆　次は，例題６に進んでください


Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Save
Check Syntax
Step
Execute
Break
(define (element3 a-list)
(first (rest (rest a-list))))
>
まず，Scheme のプログラムを
コンピュータに読み込ませている
3:3
Unlocked
not running

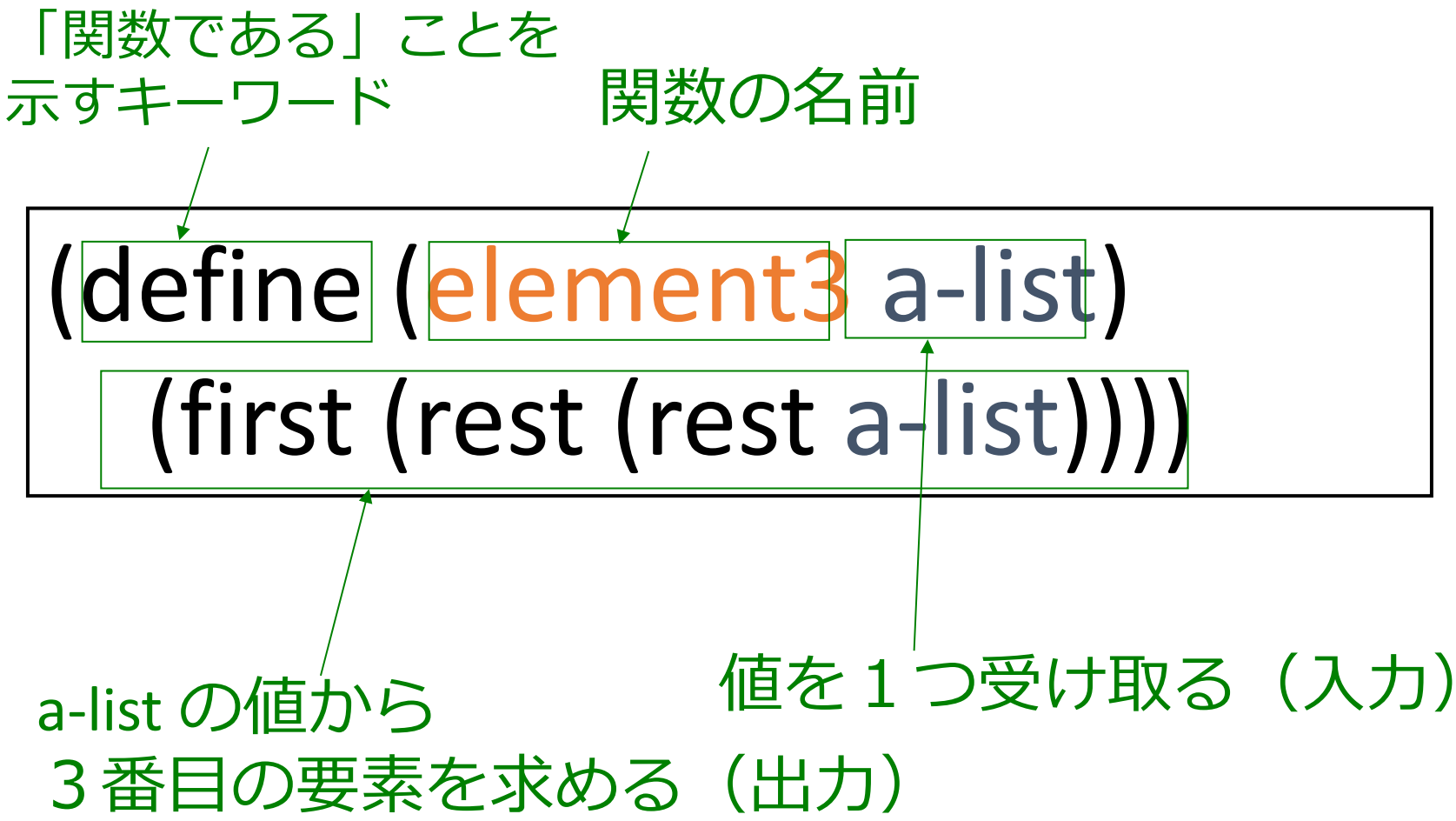

Untitled – DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Save
Check Syntax
Step
Execute
Break
(define
(first
これは,
(element3 (list 15 8 6 32 23))
と書いて，a-list の値を
(list 15 8 6 32 23) に設定しての実行
> (element3 (list 1 2 3 4))
3
> (element3 (list 15 8 6 32 23))
6
>
実行結果である「6」が
表示される
Unlocked
not running

入力と出力

(list 15 8 6 32 23)



element3 関数


「関数である」ことを
示すキーワード
関数の名前
(define (element3 a-list)
(first (rest (rest a-list))))
a-list の値から
３番目の要素を求める（出力）
値を１つ受け取る（入力）

(element3 (list 15 8 6 32 23)) から6が得られる過程

(element3 (list 15 8 6 32 23)) 最初の式

= (first (rest (rest (list 15 8 6 32 23)))) (first (rest (rest a-list)))
に a-list = (list 15 8 6 32 23) が代入される

= (first (rest (list 8 6 32 23))) (rest (list 15 8 6 32 23)) → (list 8 6 32 23)

= (first (list 6 32 23)) (rest (list 8 6 32 23)) → (list 6 32 23)

コンピュータ内部での計算

= 6 実行結果

(element3 (list 15 8 6 32 23)) から6が得られる過程

(element3 (list 15 8 6 32 23))

= (first (rest (rest (list 15 8 6 32 23))))

= (first (rest (list 8 6 32 23)))

= (first (list 6 32 23))

= 6

これは,

(define (element3 a-list)
  (first (rest (rest a-list)))

の a-list を (list 15 8 6 32 23) で置き換えたもの

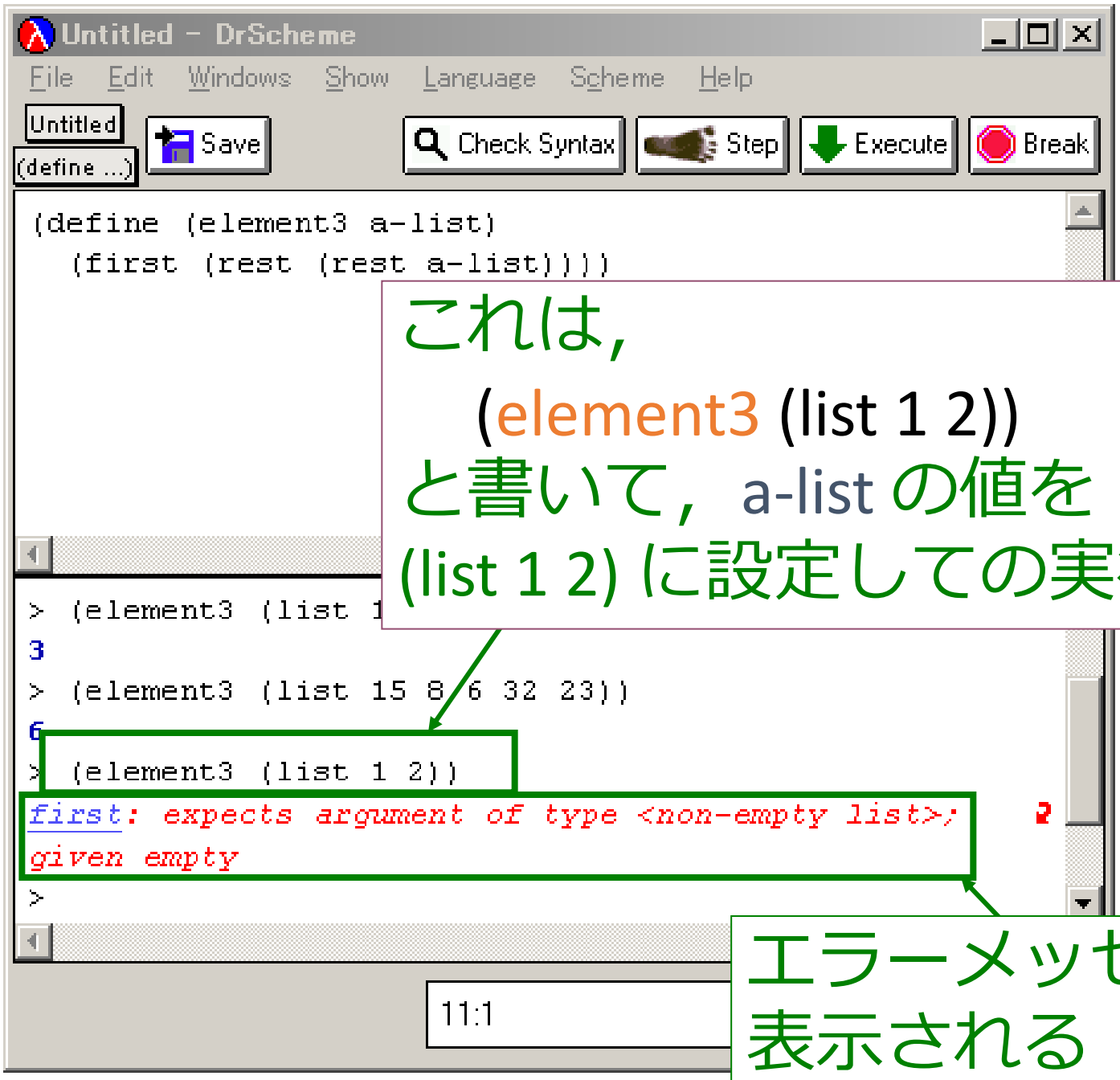
関数 element3 について

- リストの長さが 2 以下の時には，「エラーメッセージ」が表示される

例）

(element3 (list 1 2))

→　エラーメッセージが表示される



これは，

(element3 (list 1 2))

と書いて，a-list の値を

(list 1 2) に設定しての実行

エラーメッセージが表示される

(element3 (list 1 2)) から実行エラーに至る過程

(element3 (list 1 2)) 最初の式

= (first (rest (rest (list 1 2))))

(first (rest (rest a-list)))
に a-list = (list 1 2) が
代入される

= (first (rest (list 2))) (rest (list 1 2)) → (list 2)

= (first empty) (rest (list 1)) → empty

コンピュータ内部での計算

→ 「空リスト empty に対して first を実行できない」という決まりがあるので，実行エラー

例題６．シンボル

- x の値から，３種類のシンボル（'Cold, 'Warm, 'Hot）のどれかを出力する関数 judge を作り，実行する

$x \leqq 20 \quad \rightarrow$ 'Cold

$20 < x \leqq 30 \quad \rightarrow$ 'Warm

$30 < x \quad \rightarrow$ 'Hot

シンボル

「例題６．シンボル」の手順

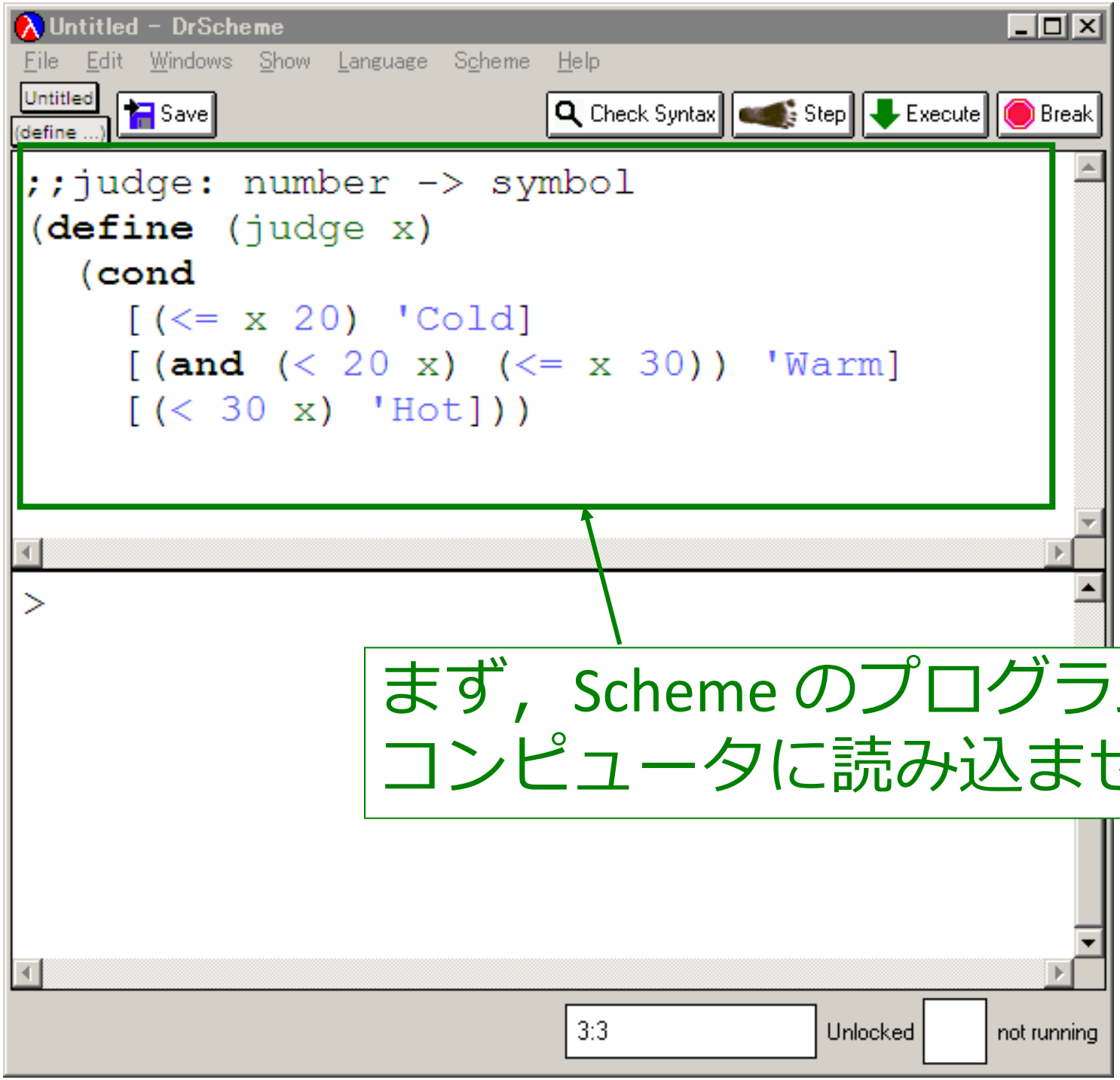
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Execute ボタンを押す

```
;;judge: number -> symbol
(define (judge x)
  (cond
    [(<= x 20) 'Cold]
    [(and (< 20 x) (<= x 30)) 'Warm]
    [(< 30 x) 'Hot]))
```

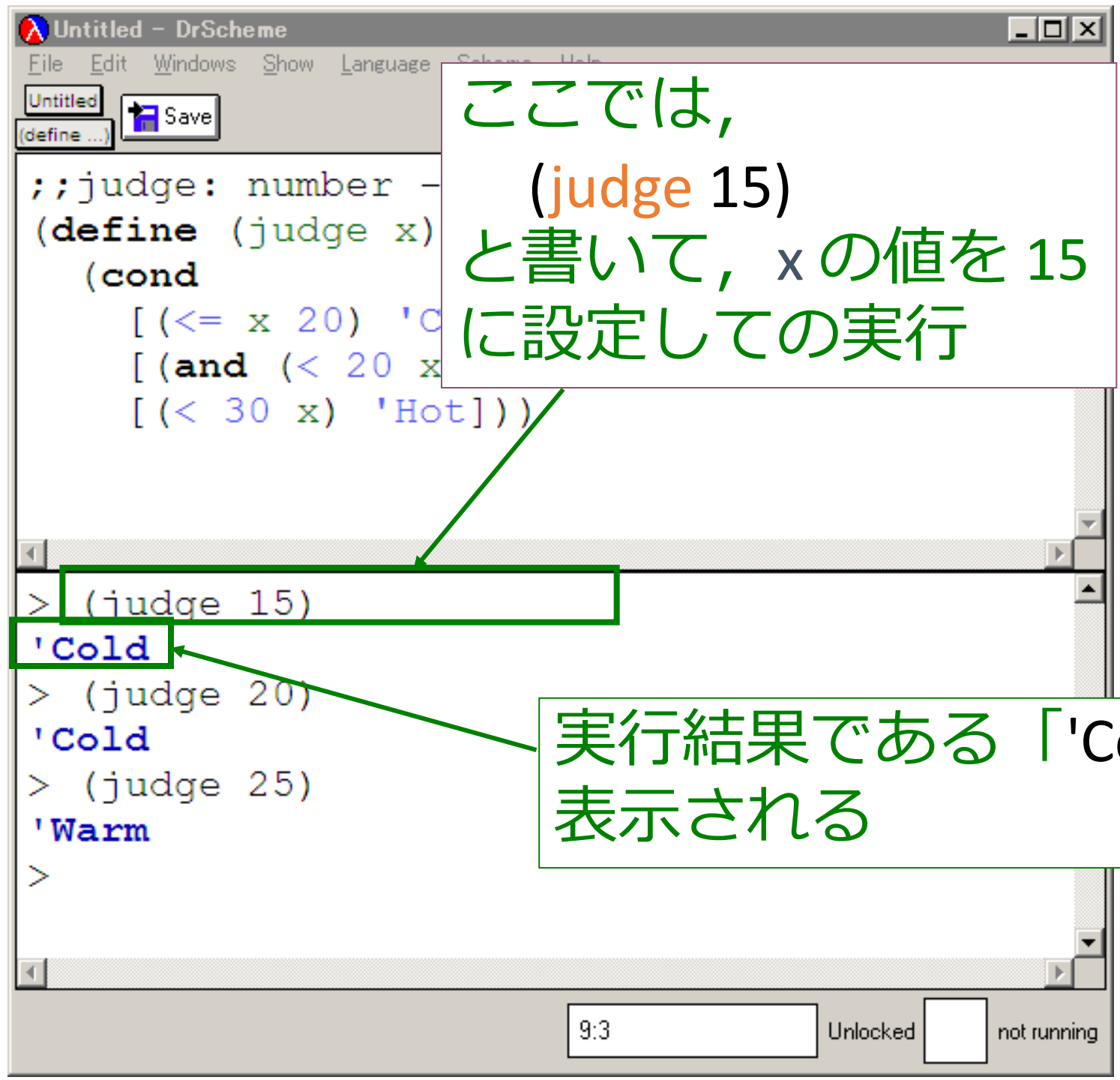
2. その後，次を「実行用ウインドウ」で実行しなさい

```
(judge 15)
(judge 20)
(judge 25)
```

☆　次は，例題７に進んでください



まず，Scheme のプログラムを
コンピュータに読み込ませている



ここでは,

(judge 15)

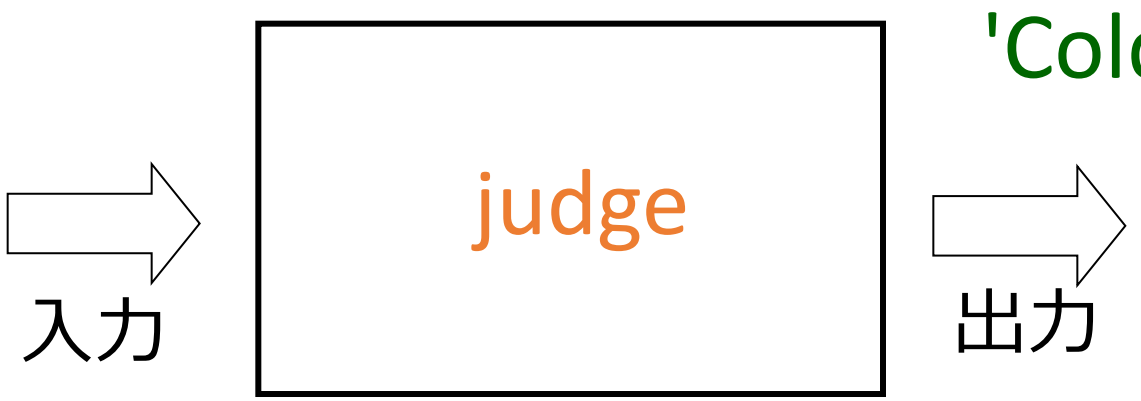
と書いて，x の値を 15 に設定しての実行

実行結果である「'Cold」が表示される

入力と出力

x の値：

15



入力は
１つの数値

出力は
１つのシンボル

judge 関数

「関数である」ことを
示すキーワード

関数の名前

```
;; judge: number -> symbol
(define (judge x)
  (cond
    [(<= x 20) 'Cold]
    [(and (< 20 x) (<= x 30)) 'Warm]
    [(< 30 x) 'Hot]))
```

値を１つ受け取る（入力）

例題７．数字かシンボルを出力

- x の値から，数字あるいはシンボルを出力する関数 ast を作り，実行する
  - x > 0 ならば：　x の値を出力する
  - x ≦ 0 ならば：　「'*」を出力する

「例題７．数値かシンボルを出力」の手順

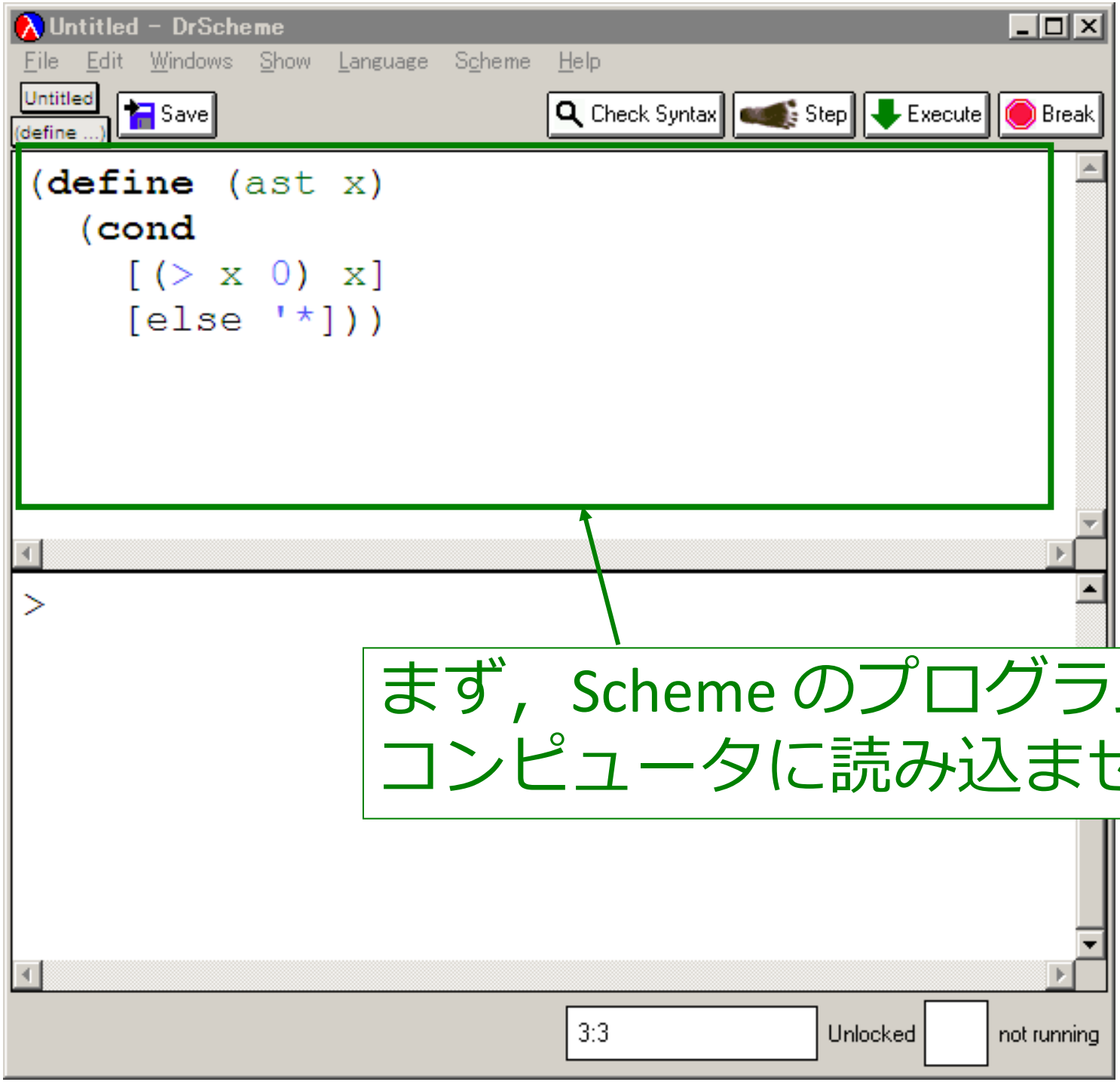
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Execute ボタンを押す

```
(define (ast x)
   (cond
      [(> x 0) x]
      [else '*]))
```

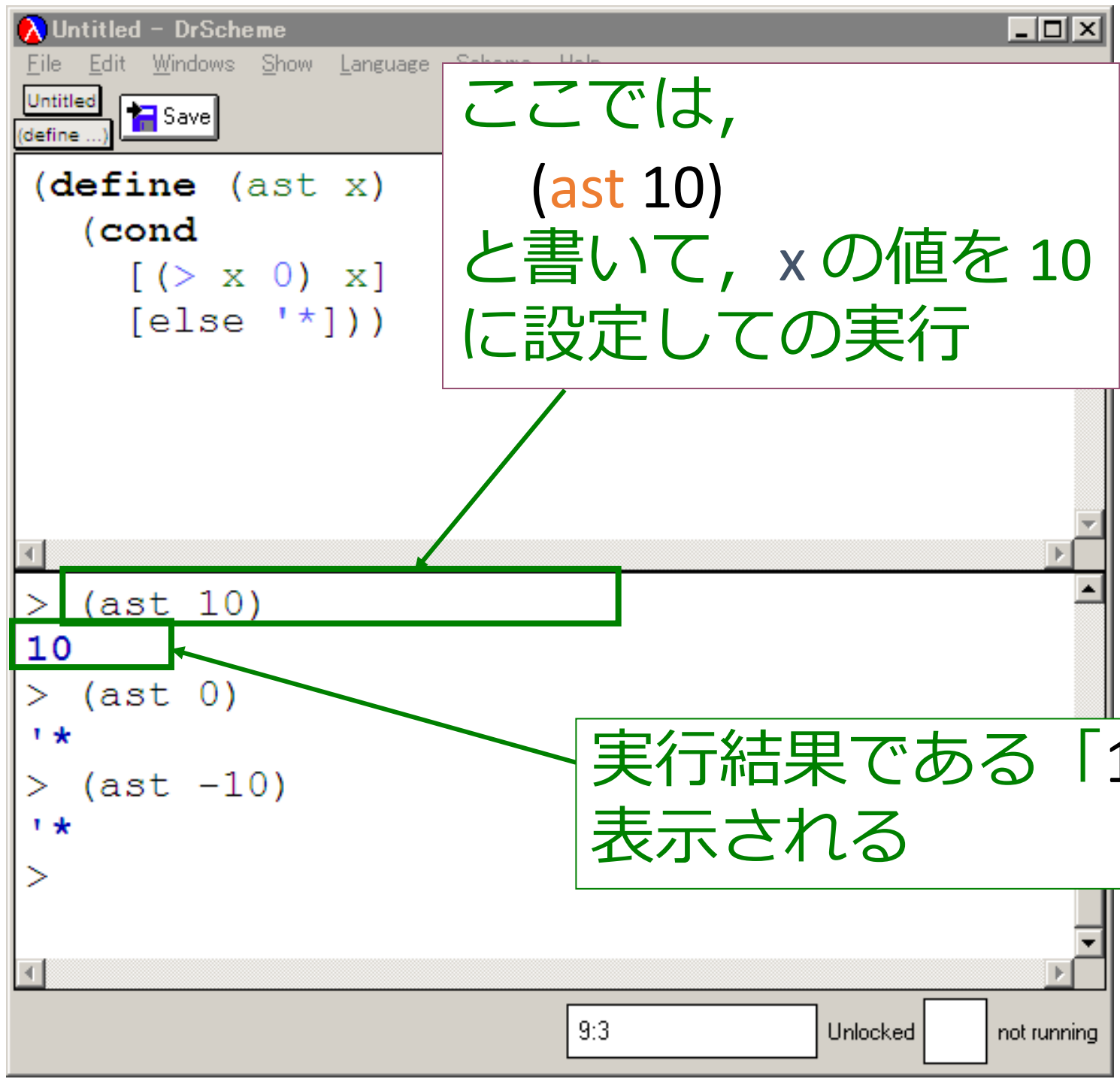
2. その後，次を「実行用ウインドウ」で実行しなさい

```
(ast 10)
(ast 0)
(ast -10)
```

☆　次は，例題８に進んでください



まず，Scheme のプログラムを
コンピュータに読み込ませている



ここでは,

(ast 10)

と書いて，x の値を 10 に設定しての実行

実行結果である「10」が表示される

入力と出力

x の値：

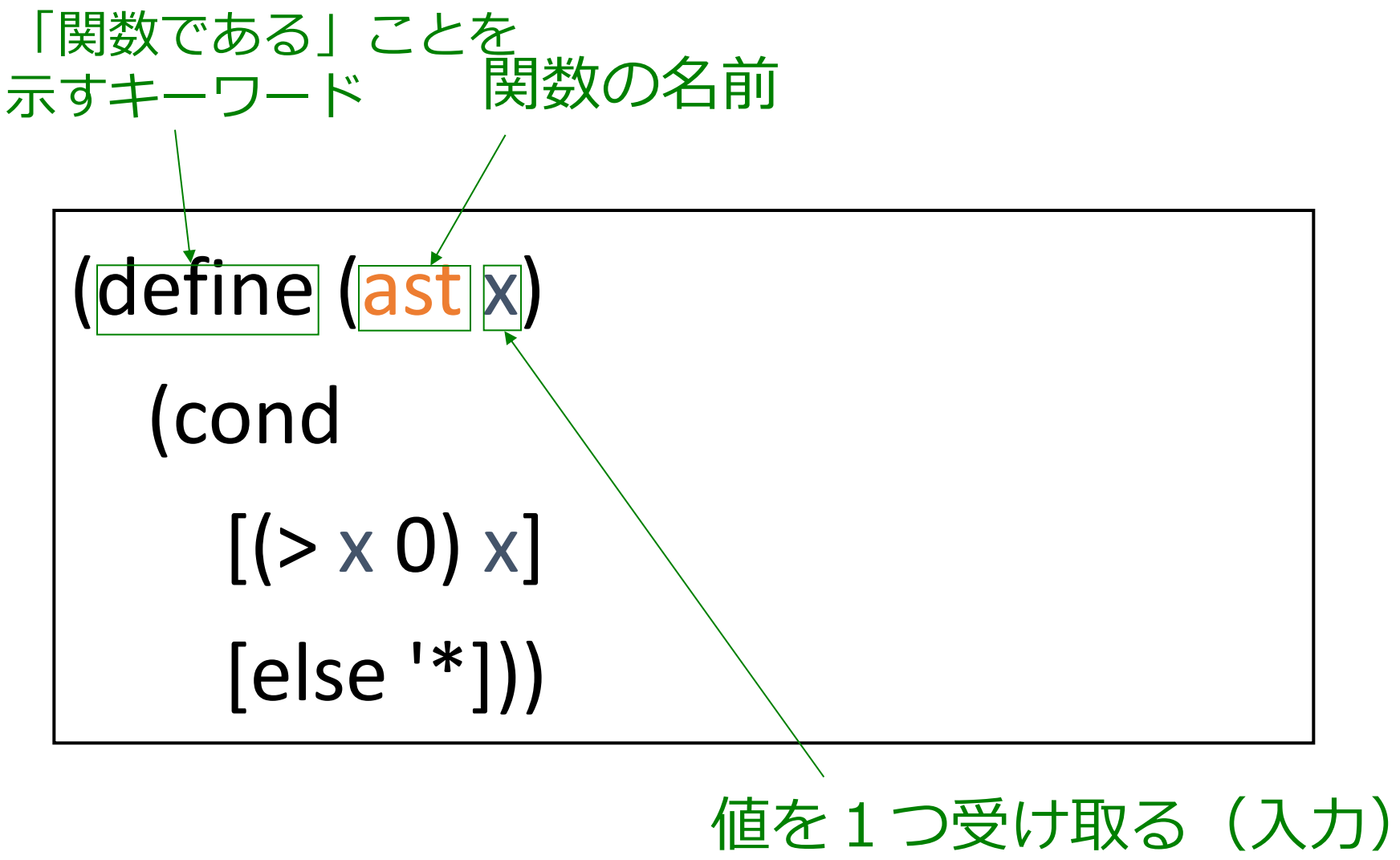
10

15

入力は
１つの数値

出力は
１つの数値
あるいはシンボル

ast 関数


「関数である」ことを
示すキーワード
関数の名前
(define (ast x)
(cond
[(> x 0) x]
[else '*]))
値を１つ受け取る（入力）

例題８．シンボル

- 次の４種のシンボルから，「答え」を返すような関数 reply を作り，実行する

		答え
'GoodMorning	→	'Hi
'HowAreYou	→	'Fine
'GoodAfternoon	→	'NeedANap
'GoodEvening	→	'BoyAmITired

これ以外の入力に対しては，実行エラー

「例題８．シンボル」の手順

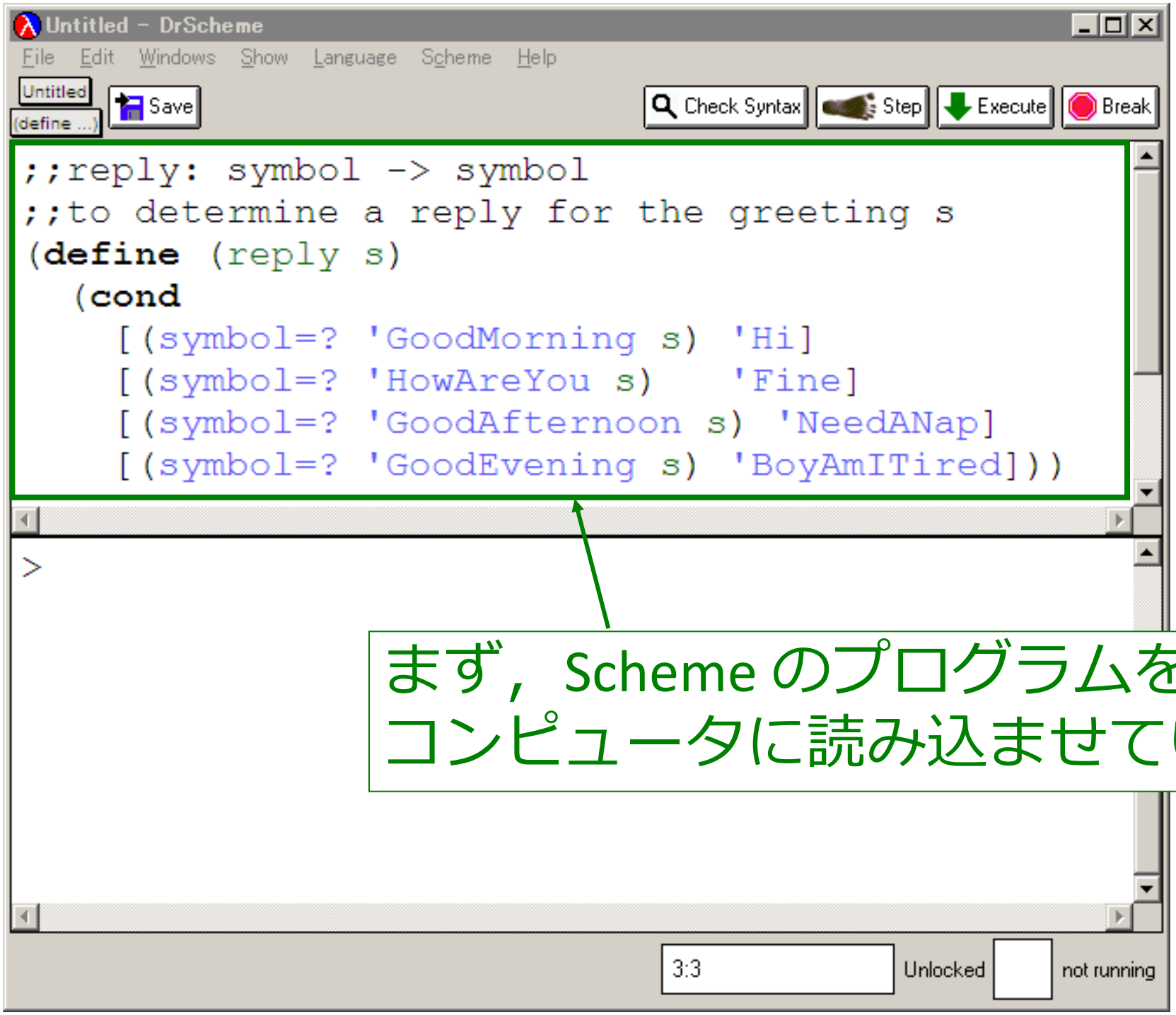
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Execute ボタンを押す

```
;;reply: symbol -> symbol
;;to determine a reply for the greeting s
(define (reply s)
  (cond
    [(symbol=? 'GoodMorning s) 'Hi]
    [(symbol=? 'HowAreYou s)   'Fine]
    [(symbol=? 'GoodAfternoon s) 'NeedANap]
    [(symbol=? 'GoodEvening s) 'BoyAmITired]))
```

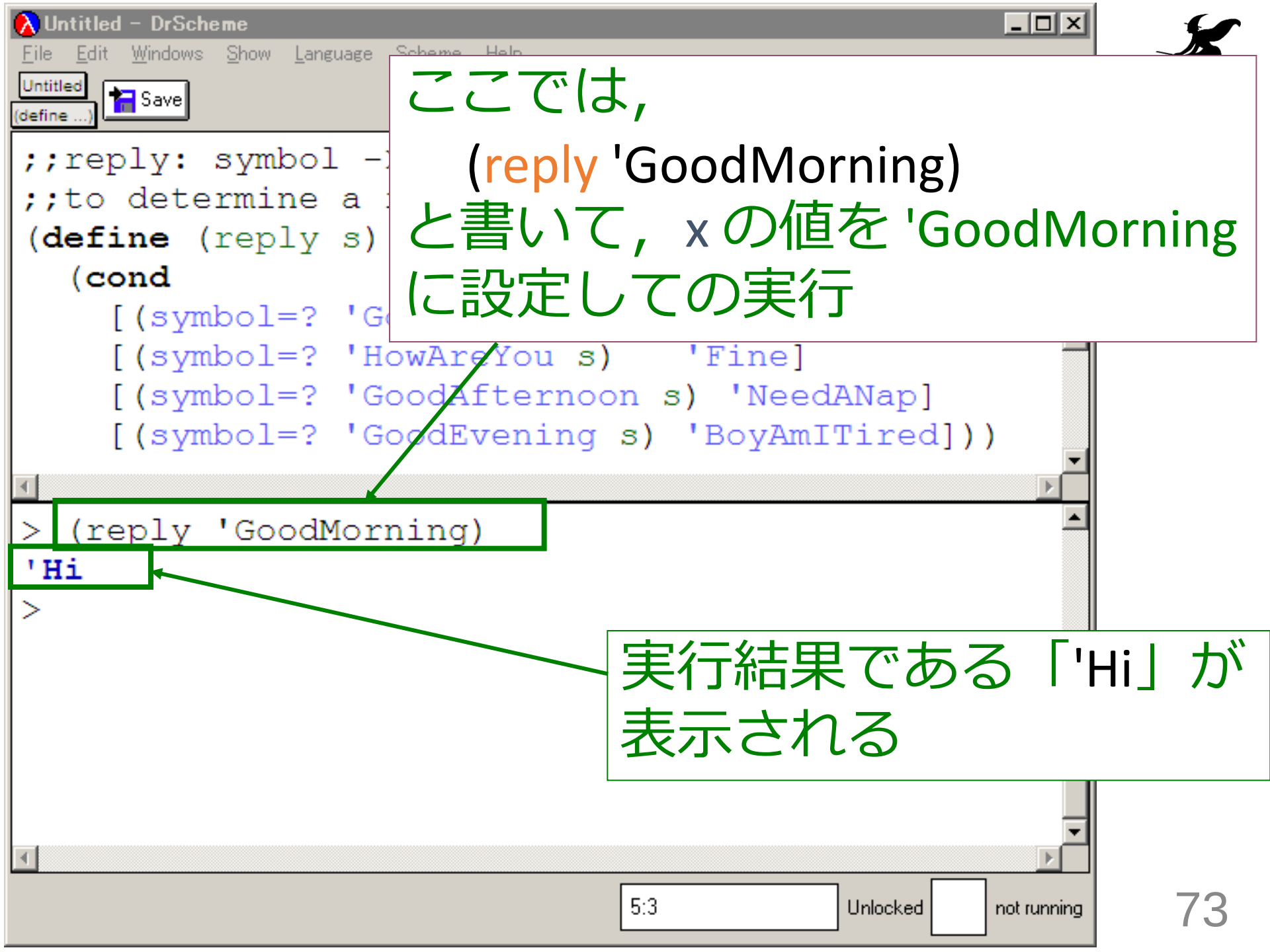
2. その後，次を「実行用ウインドウ」で実行しなさい

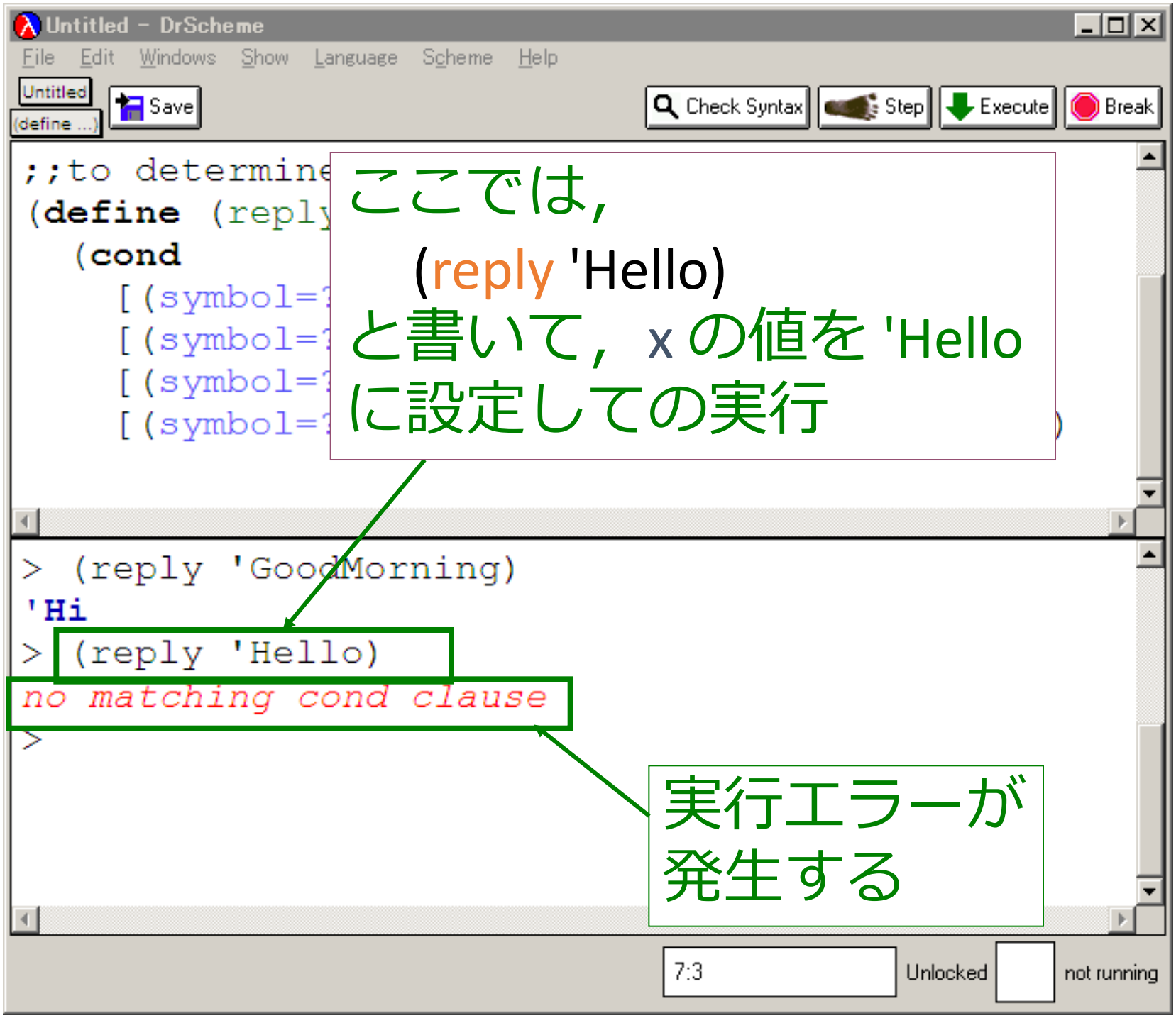
```
(reply 'GoodMorning)
(reply 'Hello)
```

☆　次は，課題に進んでください



まず，Scheme のプログラムを
コンピュータに読み込ませている

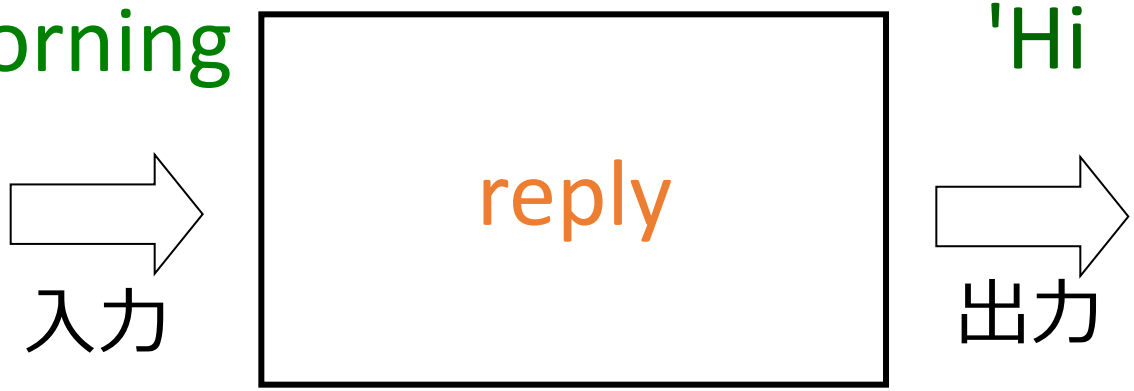

Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Save
;;reply: symbol -
;;to determine a
(define (reply s)
(cond
[(symbol=? 'G
[(symbol=? 'HowAreYou s) 'Fine]
[(symbol=? 'GoodAfternoon s) 'NeedANap]
[(symbol=? 'GoodEvening s) 'BoyAmITired]))
> (reply 'GoodMorning)
'Hi
>
5:3
Unlocked
not running
ここでは,
(reply 'GoodMorning)
と書いて，x の値を 'GoodMorning に設定しての実行
実行結果である「'Hi」が表示される


Untitled – DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Save
Check Syntax
Step
Execute
Break
;;to determine
(define (reply
(cond
[(symbol=?
[(symbol=?
[(symbol=?
[(symbol=?
ここでは,
(reply 'Hello)
と書いて，x の値を 'Hello に設定しての実行
> (reply 'GoodMorning)
'Hi
> (reply 'Hello)
no matching cond clause
>
実行エラーが発生する
7:3
Unlocked
not running

入力と出力

x の値：

'GoodMorning



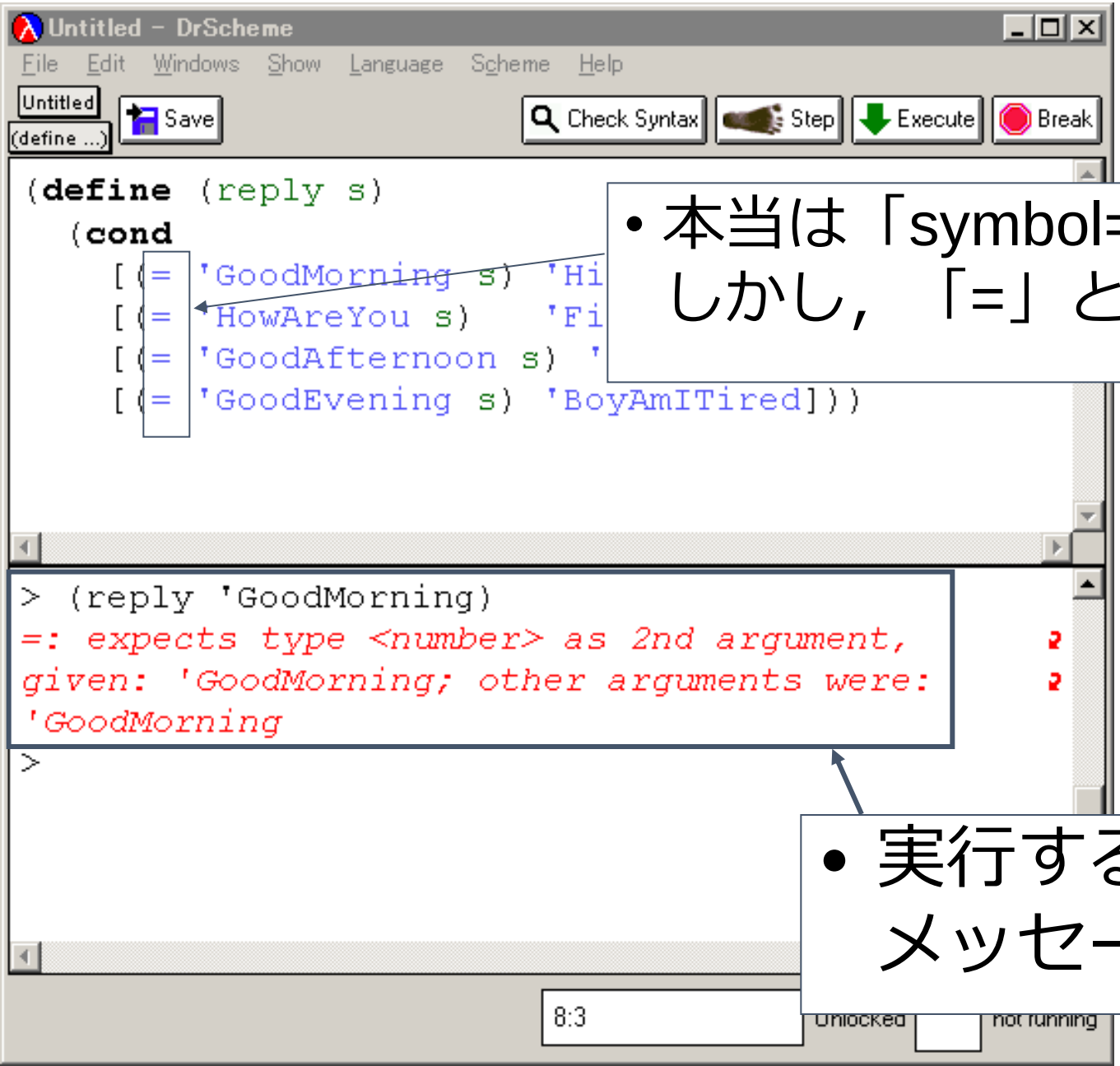
'Hi

入力はシンボル　　出力はシンボル

reply 関数

```
;; reply: symbol -> symbol
;; to determine a reply for the greeting s
(define (reply s)
  (cond
    [(symbol=? 'GoodMorning s) 'Hi]
    [(symbol=? 'HowAreYou s)   'Fine]
    [(symbol=? 'GoodAfternoon s) 'NeedANap]
    [(symbol=? 'GoodEvening s) 'BoyAmITired]))
```

よくある間違い



- 本当は「symbol=?」と書くべき．しかし，「=」と書いている．

- 実行すると，エラーメッセージが出る

条件式の判定順

```
(define (reply s)
  (cond
    ① [(symbol=? 'GoodMorning s) 'Hi]
    ② [(symbol=? 'HowAreYou s)   'Fine]
    ③ [(symbol=? 'GoodAfternoon s) 'NeedANap]
    ④ [(symbol=? 'GoodEvening s) 'BoyAmITired]))
```

判定順

- cond 文に並べた条件式は，上から順に判定される
  - 上の例では，①，②，③，④の順に判定が行われ，①が成り立てば，②，③，④は判定されない
- 条件式の並べ方に意味がある

(reply 'GoodMorning) から 'Hi が得られる過程


```
(reply 'GoodMorning)
= (cond
    [(symbol=? 'GoodMorning 'GoodMorning) 'Hi]
    [(symbol=? 'HowAreYou 'GoodMorning)   'Fine]
    [(symbol=? 'GoodAfternoon 'GoodMorning) 'NeedANap]
    [(symbol=? 'GoodEvening 'GoodMorning) 'BoyAmITired])
= (cond
    [true 'Hi]
    [(symbol=? 'HowAreYou 'GoodMorning)   'Fine]
    [(symbol=? 'GoodAfternoon 'GoodMorning) 'NeedANap]
    [(symbol=? 'GoodEvening 'GoodMorning) 'BoyAmITired]) =
'Hi
```

(reply 'GoodMorning) から 'Hi が得られる過程

```
(reply 'GoodMorning)
= (cond
    [(symbol=? 'GoodMorning 'GoodMorning) 'Hi]
    [(symbol=? 'HowAreYou 'GoodMorning)   'Fine]
    [(symbol=? 'GoodAfternoon 'GoodMorning) 'NeedANap]
    [(symbol=? 'GoodEvening 'GoodMorning) 'BoyAmITired])
= (cond
```

これは,

```
(cond
  [(symbol=? 'GoodMorning s) 'Hi]
  [(symbol=? 'HowAreYou s)   'Fine]
  [(symbol=? 'GoodAfternoon s) 'NeedANap]
  [(symbol=? 'GoodEvening s) 'BoyAmITired])
```

の s を 'GoodMorning で置き換えたもの

5-6 課題

課題 1

- 実行結果を報告しなさい
  - 「DrScheme の実行用ウインドウ」で実行して，実行結果を報告しなさい
  - 「エラー」が出た場合には，「エラー」と記入すること

	(list 1)	(list 1 2)	(list 1 2 3)
(first (list ...)) の実行結果			
(rest (list ...)) の実行結果			
(first (rest (list ...))) の実行結果			
(first (rest (rest(list ...)))) の実行結果			

課題２

- 最高気温 high と最低気温 low から，真夏日，夏日，冬日，真冬日を判定する関数 summer-winter-day を作成し，実行結果を報告しなさい
  - "Tropical Day"
    （真夏日，１日の最高気温が３０度以上の日）
  - "Summer Day"
    （夏日，１日の最高気温が２５度以上の日）
  - "Frost Day"
    （冬日，１日の最低気温が０度未満の日）
  - "Ice Day"
    （真冬日，１日の最高気温が０度未満の日）

課題 3

- ある年 y のある月 m のある日 d が存在するかを調べ，存在すれば d を，存在しなければシンボル「'*」を返す関数を作成し，実行結果を報告しなさい
  - 例えば，

    2004 10 10 ⇒ 10 を出力

    2004 10 0　⇒ '* を出力

    2004 10 32 ⇒ '* を出力