sp-3. 関数の組み合わせ

（Scheme プログラミング）

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

金子邦彦

3-1 Scheme の関数

アウトライン

3-1 Scheme の関数

3-2 パソコン演習

3-3 課題

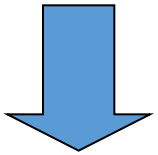
本日の内容

- Scheme の式から出発して，実行結果に至るステップを読み，理解する
- 複数の関数から構成されたプログラムを読んで，意味を理解できる能力，知識を身に付ける
- 見やすいプログラムを書くために，ブロック単位での字下げを行う

実行結果に至る過程

なぜ？

コンピュータの振る舞いを理解するためには



Scheme の式から出発して，
実行結果に至るステップを理解する

実行結果に至る過程

- 演算子を含む式が計算される
  - 括弧の内側が優先される

  例： (* 4 12) = 48

  (* 3.14 (* 5 5)) = (* 3.14 25) = 78.5
- 関数が，その「中身」で置き換わる
  - 同時に，関数のパラメータは，実際の値で置き換わる

  例: (area-of-disk 5) = (* 3.14 (* 5 5)) = ...

  (+ (sqr 2) (sqr 4)) = (+ (* 2 2) (sqr 4)) = ...

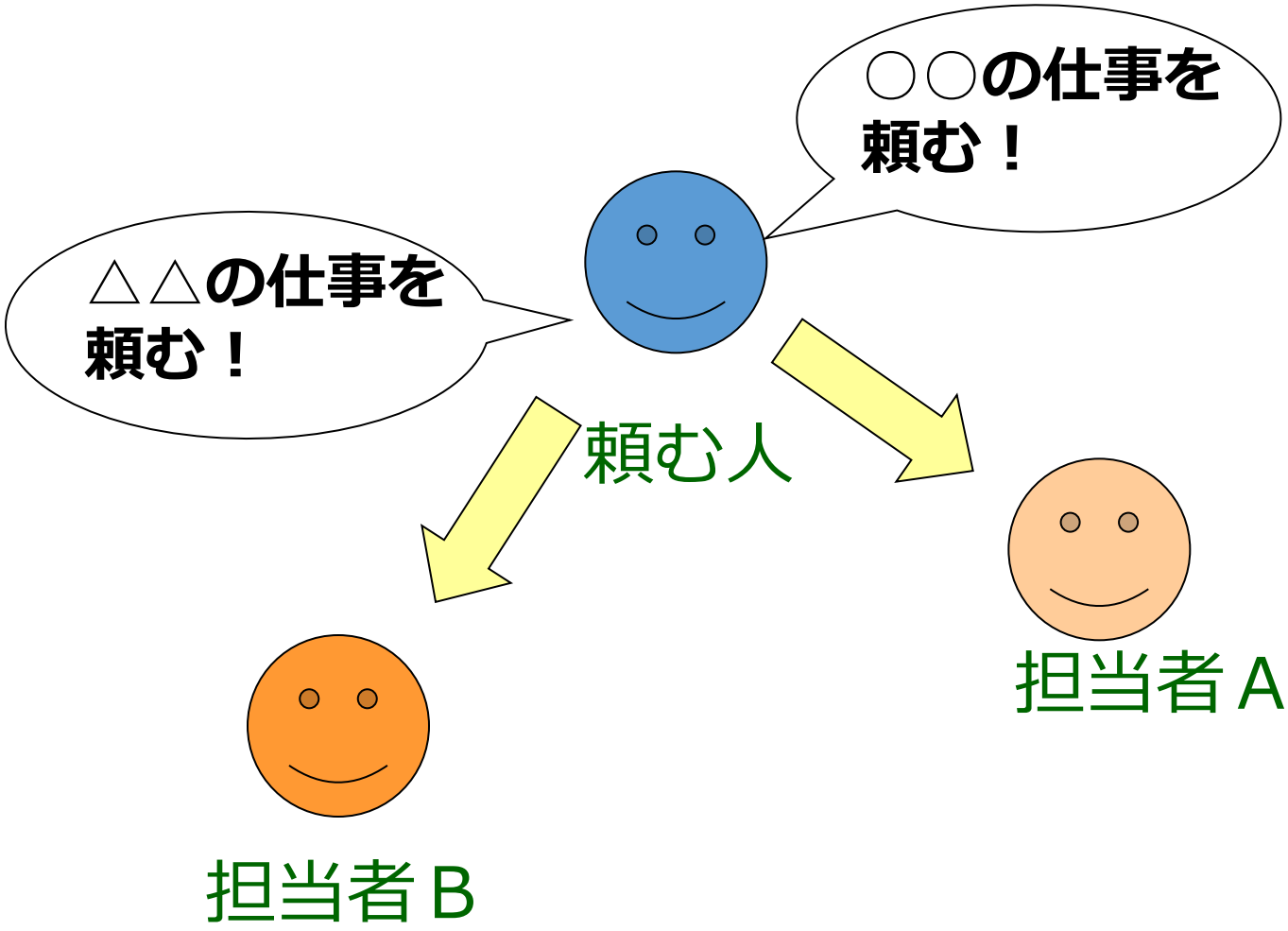
- Scheme 処理系と実行モデルを理解する
- 変数と関数の違い

Scheme の関数の振る舞い

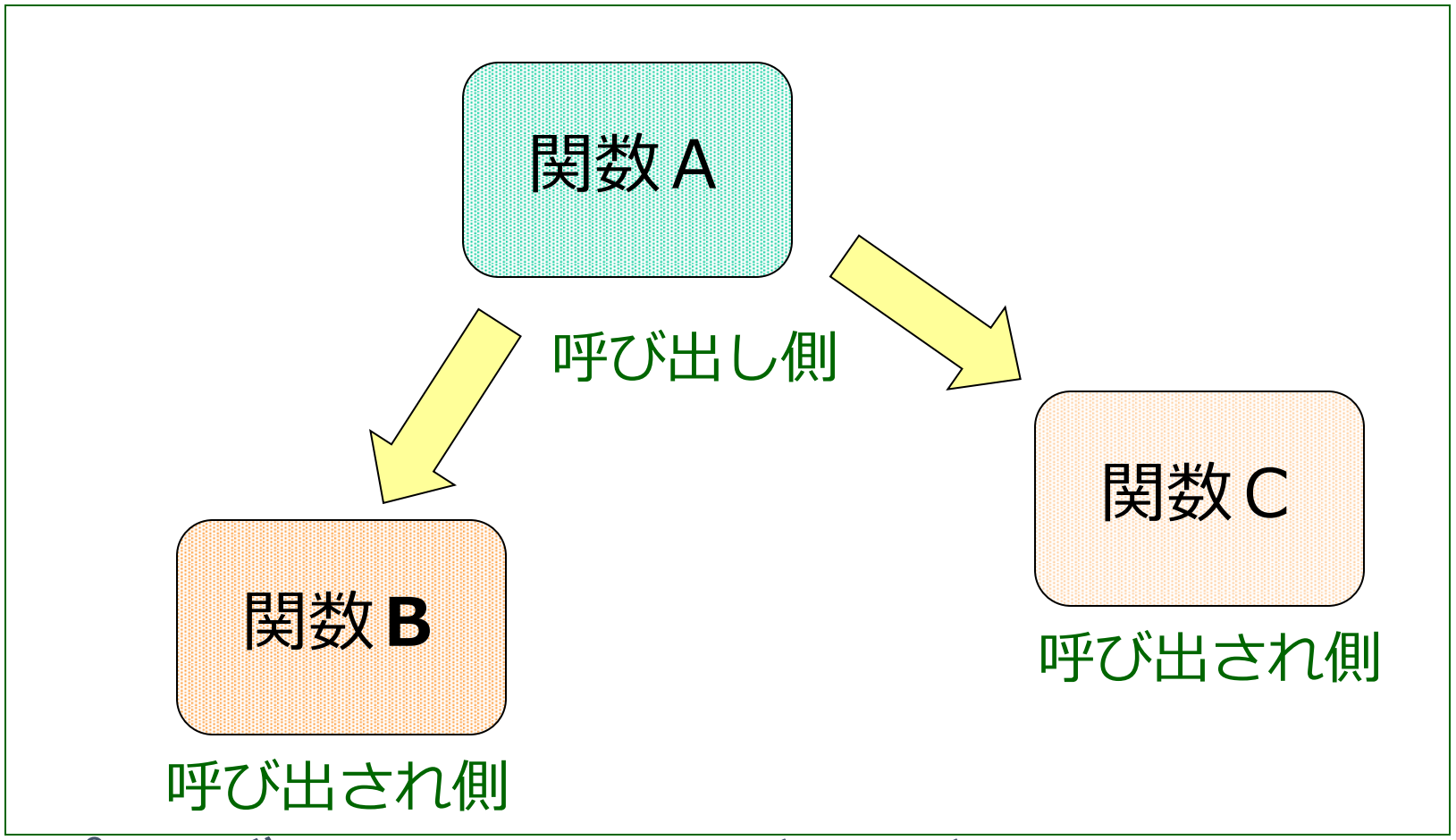
関数 (procedure arg_1 ... arg_n) に対して

- procedure の値を取得する
- arg_1 から arg_n の値を取得する
- procedure の値に arg_1 から arg_n の値を引き渡す

仕事の分割



関数とは



プログラムは，しばしば，複数の関数に「分割」される

Scheme のプログラムと関数

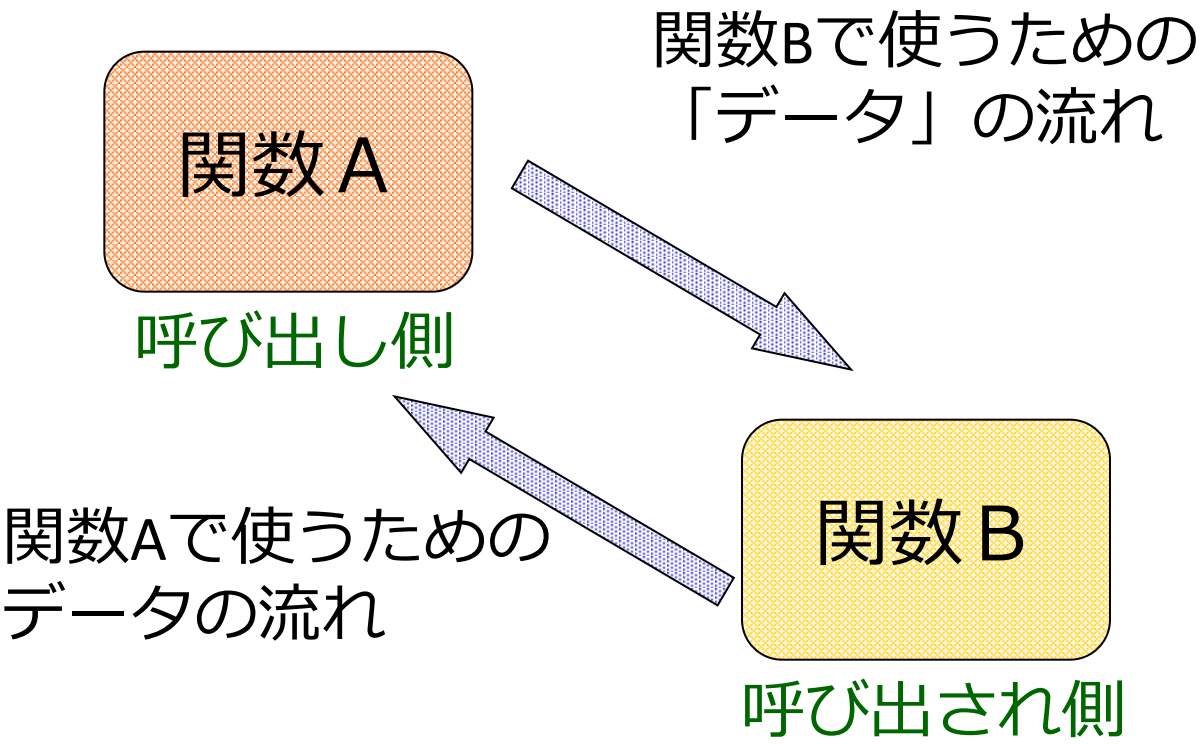
- Scheme のプログラムは，一般に，複数の関数の集まり

```
(define   (area-of-disk   r)
    (* 3.14
       (*  r   r)))
(define (area-of-ring outer inner)
    (-  (area-of-disk outer)
        (area-of-disk inner)))
```

これらは Scheme の式

⇒　参照：　例題３、４

関数でのデータの流れ



3-2 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
- 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

ステップ実行とは

- DrScheme の機能
- プログラム実行の振る舞いを観察するためのツール
- 「定義用ウインドウ」のみを使用

  ←　普通のプログラム実行とは違う
- 「Intermediate Student」に設定する必要あり

例題１．実行結果に至る過程

- 次の式について，実行結果「48」に至る過程を見る
- DrScheme の stepper を使用する

```
(* (+ 2 2)
   (/ (* (+ 3 5)
         (/ 30 10))
      2))
```

Scheme の式

「*」とあるのは，「乗算」の意味

「例題１．実行結果に至る過程」の手順

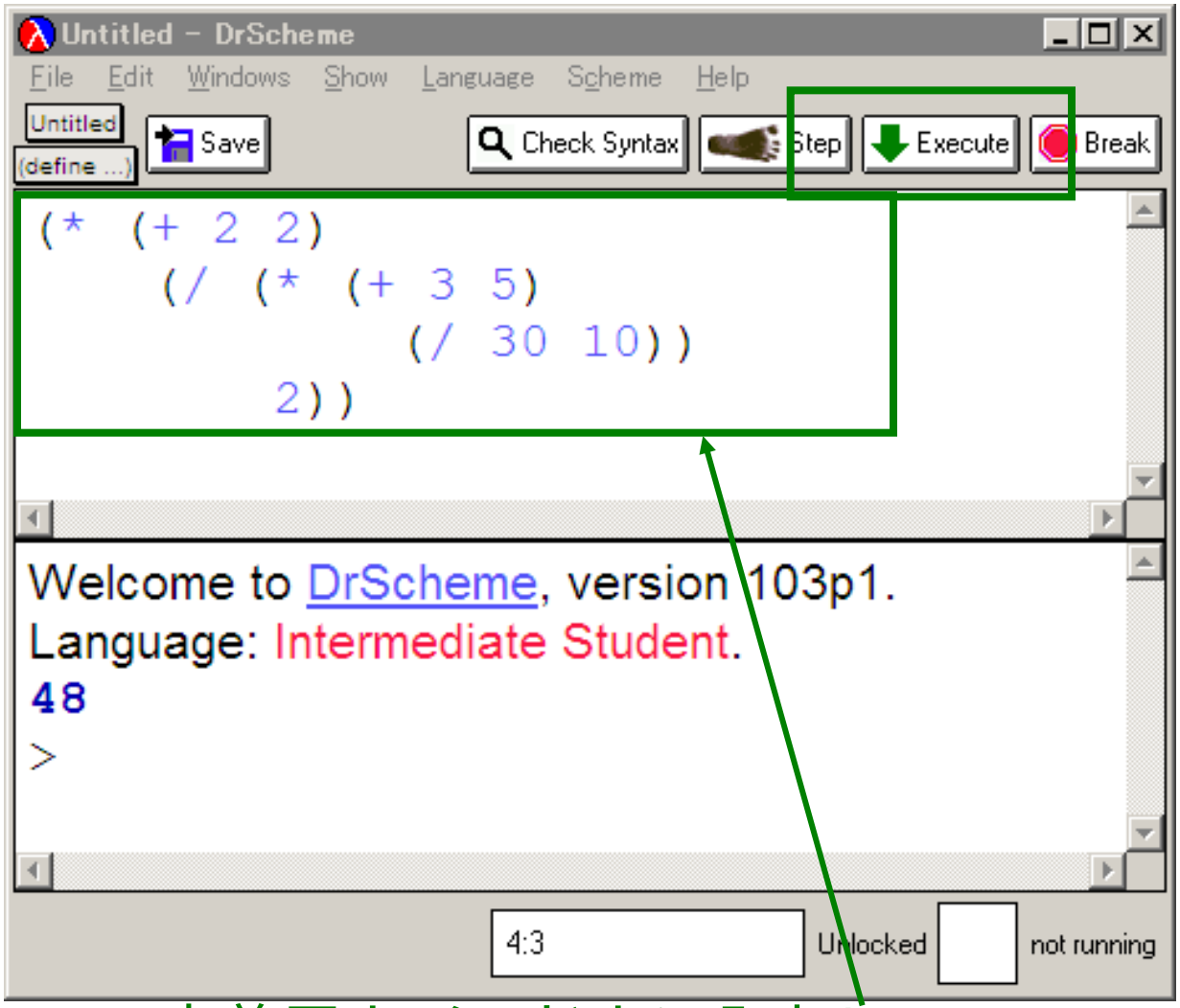
1. 次を「定義用ウインドウ」で，実行しなさい
   - Intermediate Student で実行すること
   - 入力した後に，Execute ボタンを押す

```
(* (+ 2 2)
   (/ (* (+ 3 5)
         (/ 30 10))
      2))
```

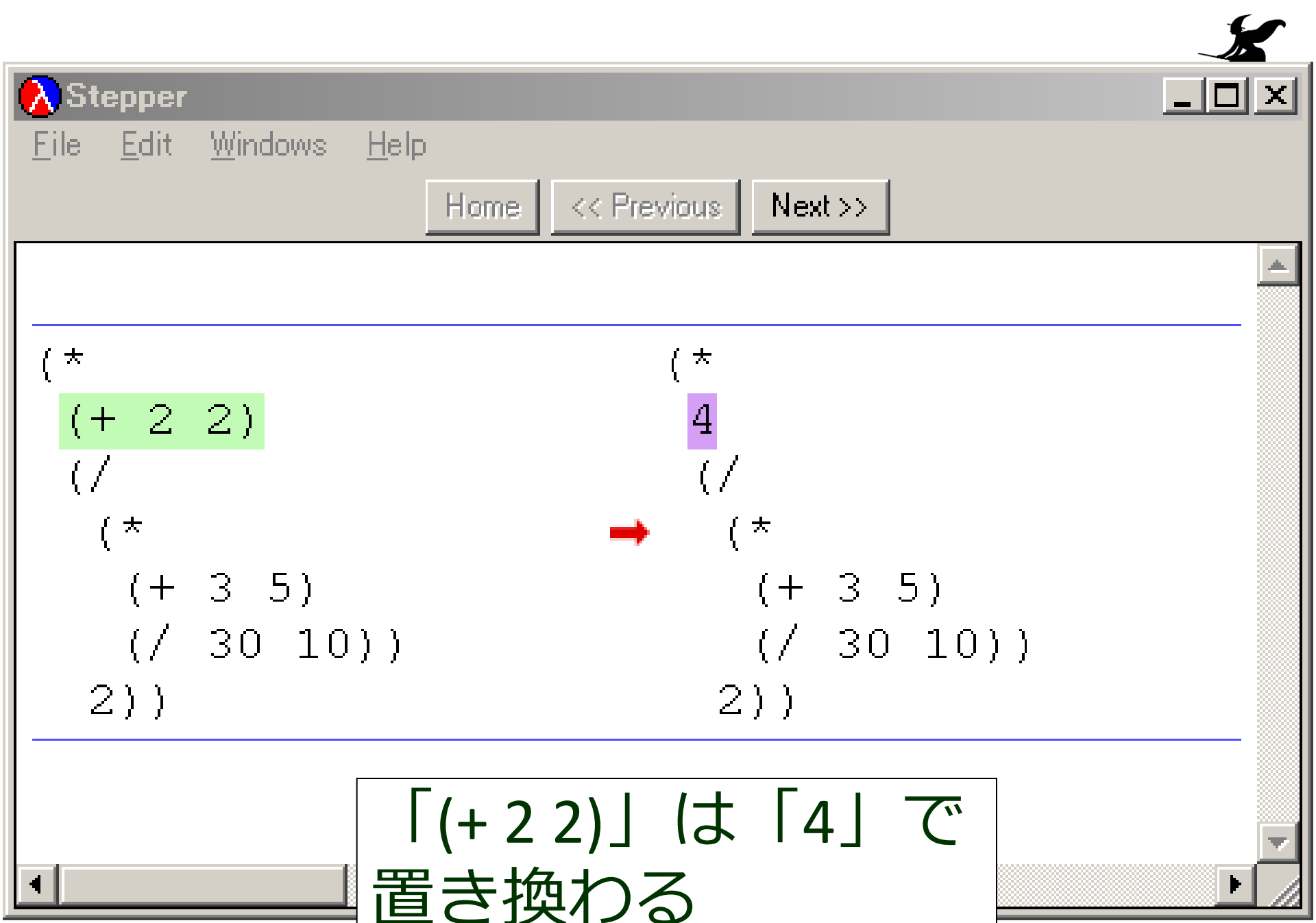
2. DrScheme を使って，ステップ実行の様子を確認しなさい　（Step ボタン，Next ボタンを使用）
   - 理解しながら進むこと

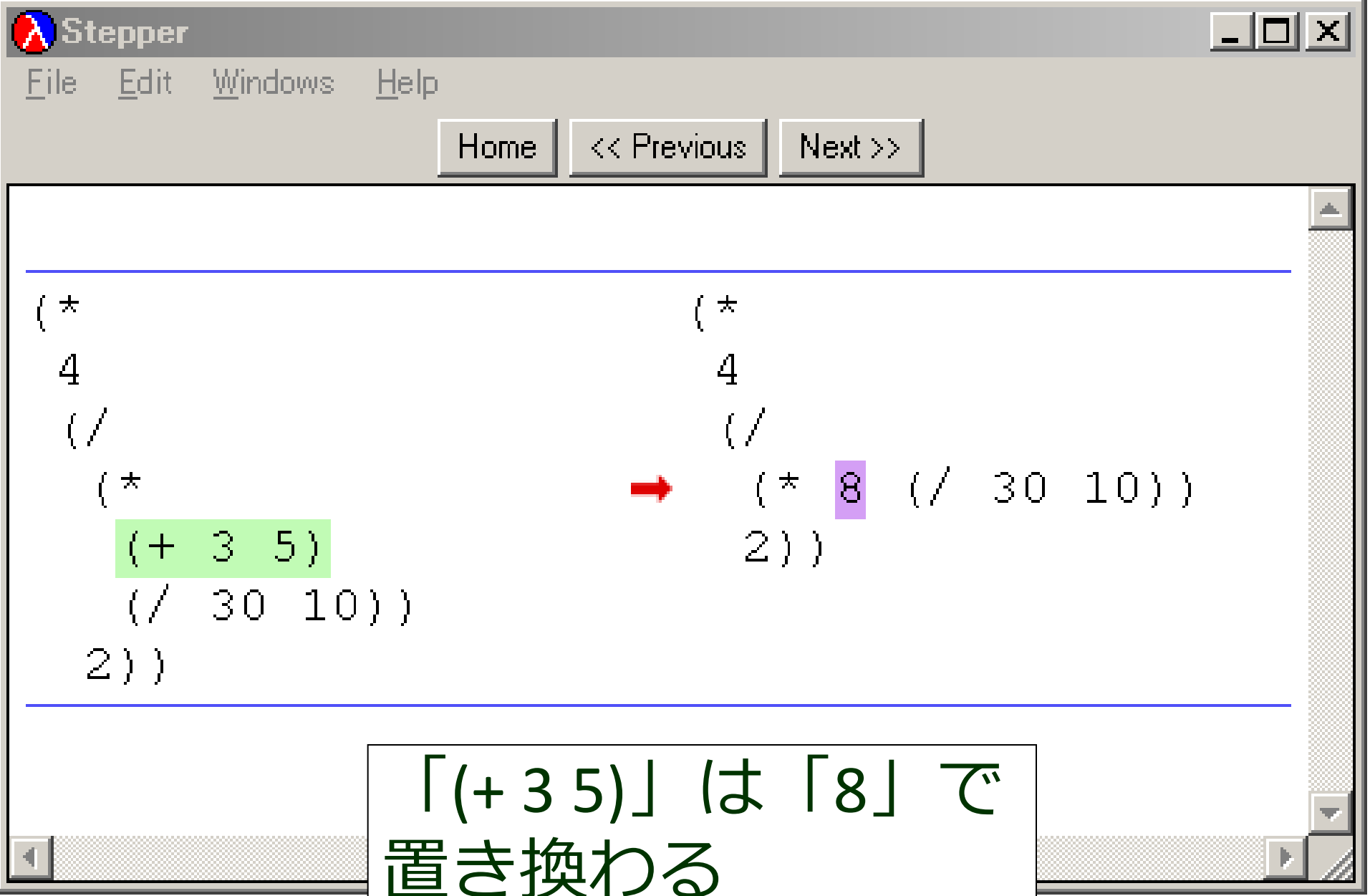
☆　次は，例題２に進んでください

例題２．式のステップ実行

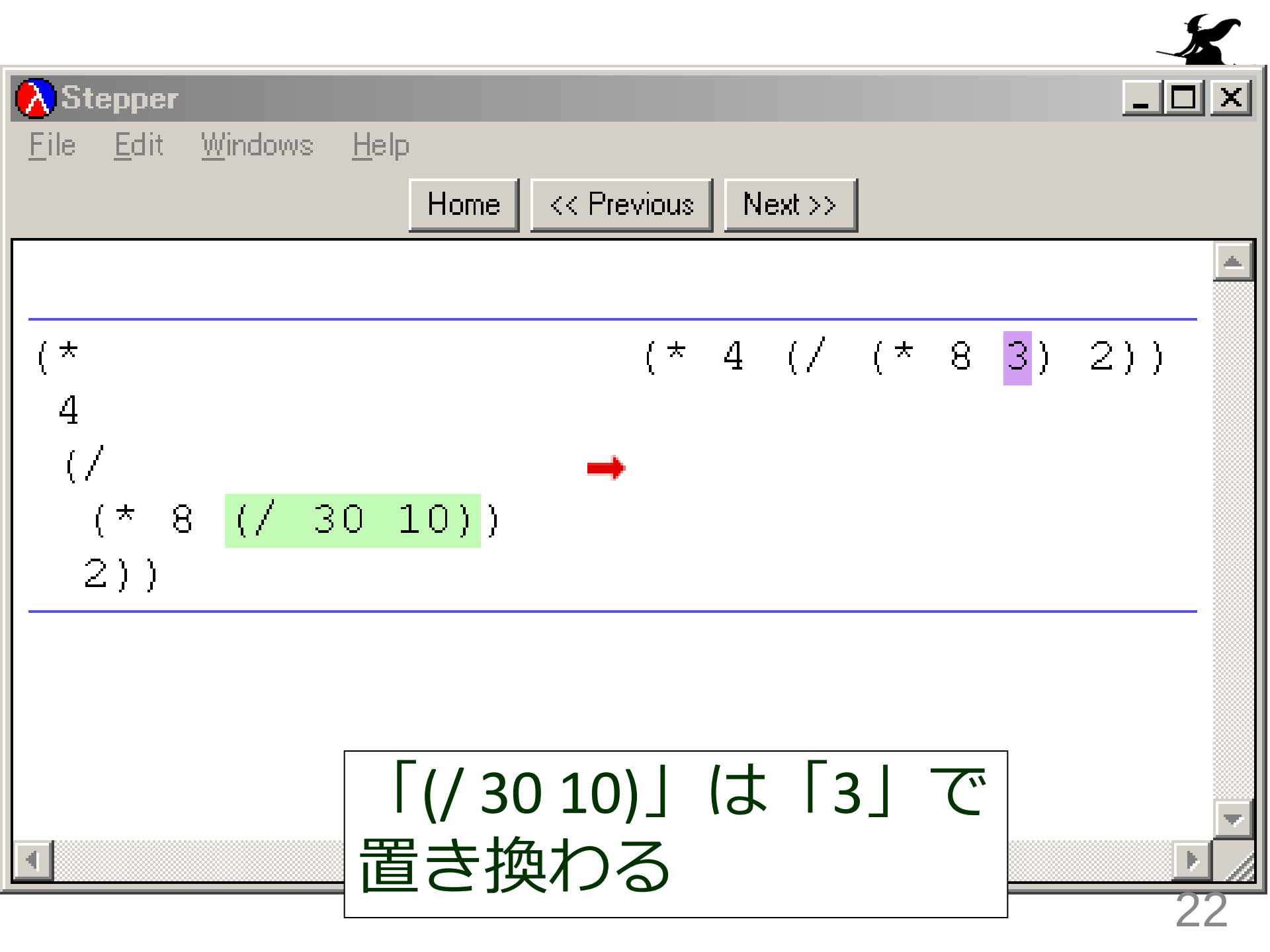


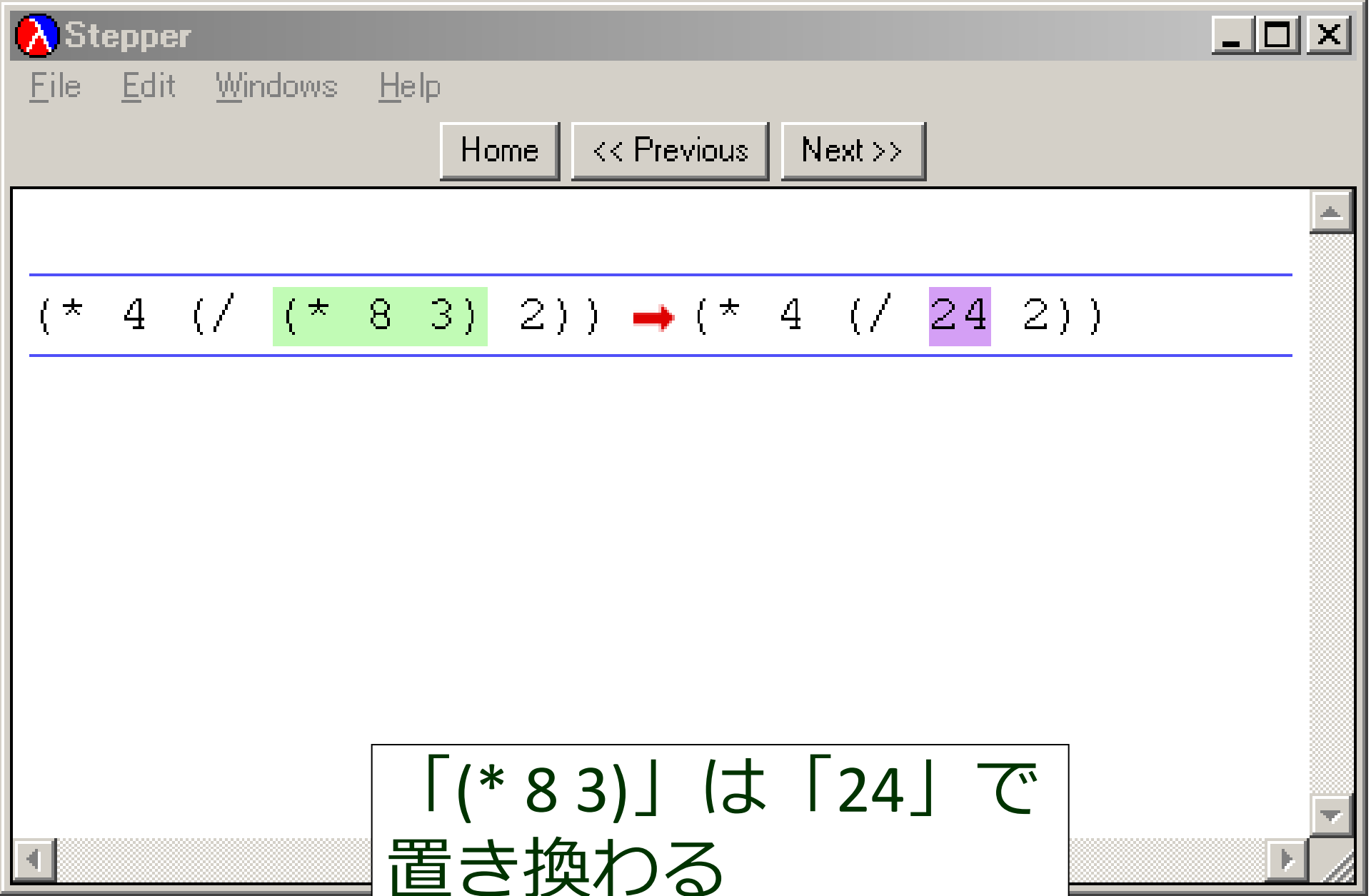
定義用ウインドウに入力して，
Execute ボタンを押した後，Step ボタンを押すと


Stepper
File Edit Windows Help
Home << Previous Next >>
(*
(+ 2 2)
(/
(*
(+ 3 5)
(/ 30 10))
2))
(*
4
(/
(*
(+ 3 5)
(/ 30 10))
2))
「(+ 2 2)」は「4」で置き換わる

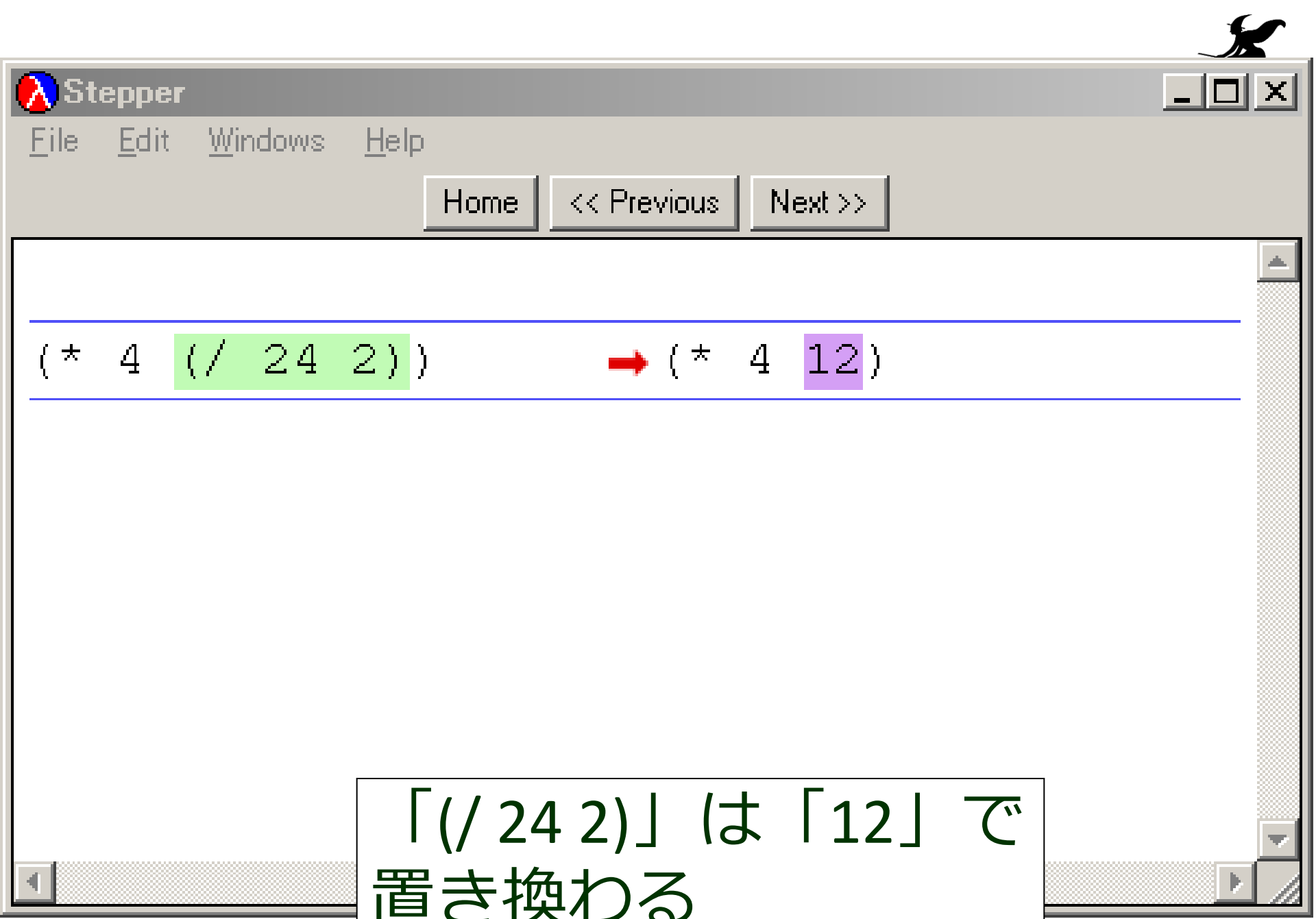

Stepper
File Edit Windows Help
Home << Previous Next >>
(*
4
(/
(*
(+ 3 5)
(/ 30 10))
2))
(*
4
(/
(* 8 (/ 30 10))
2))

「(+ 3 5)」は「8」で置き換わる

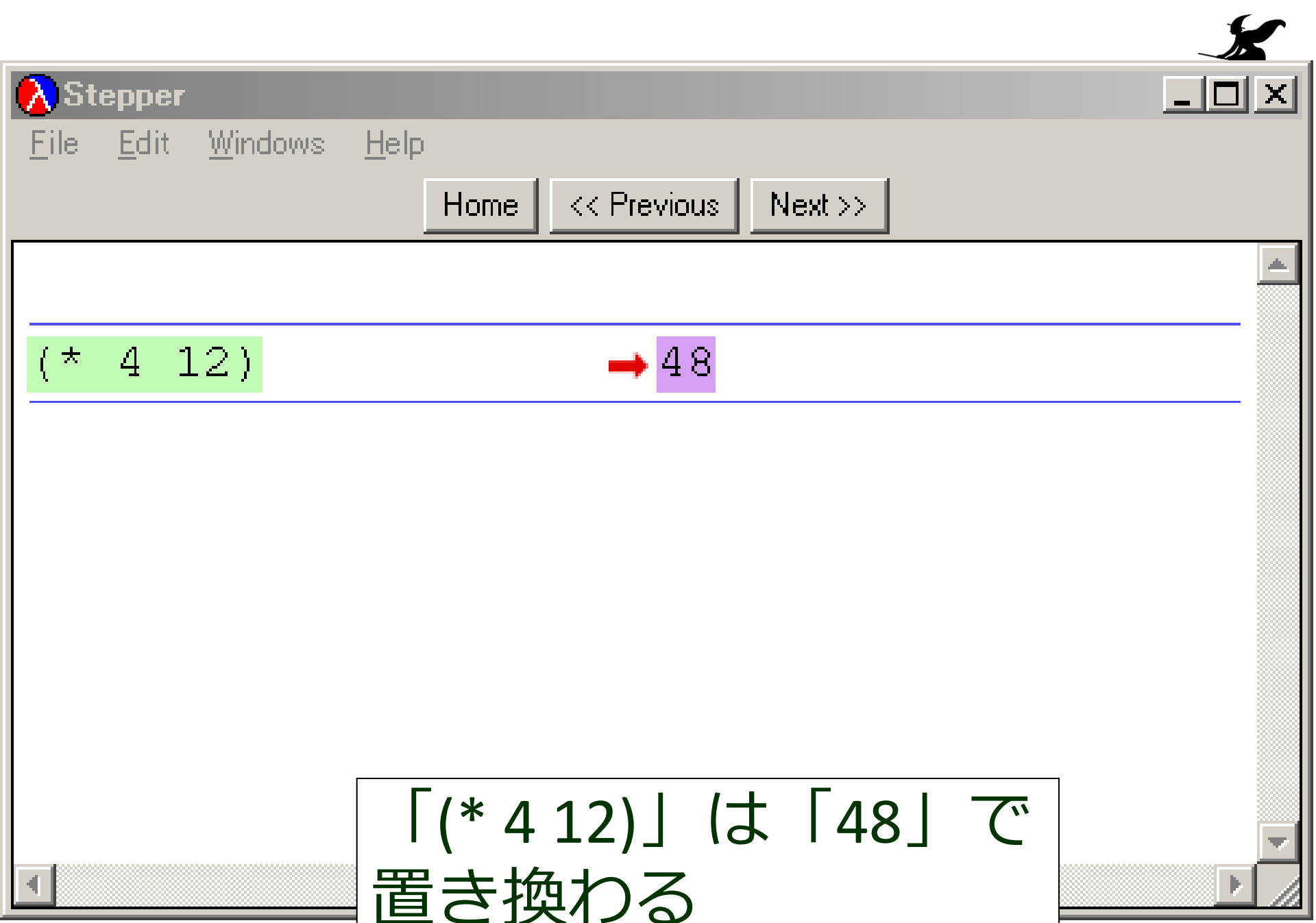

Stepper
File Edit Windows Help
Home << Previous Next >>
(* (* 4 (/ (* 8 3) 2))
4
(/
(* 8 (/ 30 10))
2))
「(/ 30 10)」は「3」で置き換わる



「(* 8 3)」は「24」で置き換わる



「(/ 24 2)」は「12」で置き換わる



「(* 4 12)」は「48」で置き換わる

実行結果「４８」が得られる過程

(* (+ 2 2) (/ (* (+ 3 5) (/ 30 10)) 2))　最初の式

= (* 4 (/ (* (+ 3 5) (/ 30 10)) 2))　(+ 2 2) → 4

= (* 4 (/ (* 8 (/ 30 10)) 2))　(+ 3 5) → 8

= (* 4 (/ (* 8 3) 2))　(/30 10) → 3

= (* 4 (/ 24 2))　(* 8 3) → 2

= (* 4 12)　(/24 2) → 12　コンピュータ内部での計算

= 48　実行結果

実行結果「４８」が得られる過程

(* (+ 2 2) (/ (* (+ 3 5) (/ 30 10)) 2))

最初の式

= ((/ (* (+ 3 5) (/ 30 10)) 2))

(+ 2 2) → 4

最初の式

(* (+ 2 2) (/ (* (+ 3 5) (/ 30 10)) 2))

から始まって，計算を繰り返して，実行結果

48

に至る

(/24 2) → 12

= 48

実行結果

例題２．関数のステップ実行

- 円の半径 r から面積を求める関数 area-of-disk について，実行結果に至る過程を見る
  - (area-of-disk 5) から 78.5 に至る過程を見る
  - DrScheme の stepper を使用する

```
(define (area-of-disk r)
   (* 3.14
      (* r r)))
```

```
(area-of-disk 5)
= (* 3.14 (* 5 5))
= (* 3.14 25)
= 78.5
```

例題２．関数のステップ実行

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

```
(define  (area-of-disk  r)
   (* 3.14
      (*  r  r)))
(area-of-disk 5)
```

例題１と同じ

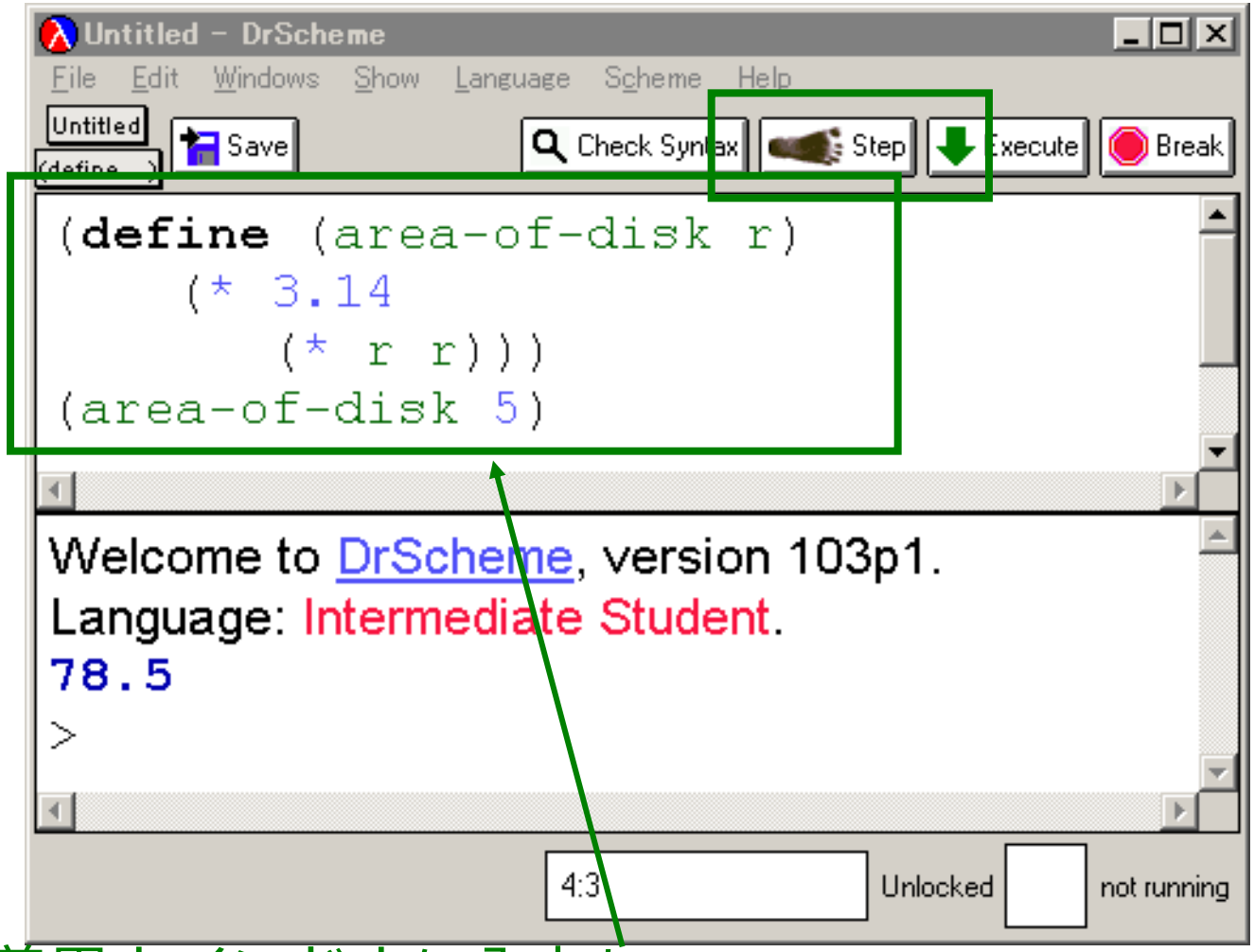
ステップ実行したいので，入力済みのプログラムは，消さずに残しておく

例題１に１行書き加える

2. DrScheme を使って，ステップ実行の様子を確認しなさい　（Step ボタン，Next ボタンを使用）
   - 理解しながら進むこと

☆　次は，例題３に進んでください

例題２．式のステップ実行



定義用ウインドウに入力して，
Execute ボタンを押した後，Step ボタンを押すと

Stepper

File Edit Windows Help

Home << Previous Next >>

```
(define (area-of-disk r)
  (* 157/50 (* r r)))
```

```
(area-of-disk 5)
```

```
(* 157/50 (* 5 5))
```

「(area-of-disk 5)」は
「(* 3.14 (* 5 5))」で置き換わる

注：「3.14」は「157/50」のように表示されている

Stepper

File Edit Windows Help

Home << Previous Next >>

```
(define (area-of-disk r)
  (* 157/50 (* r r)))
```

```
(* 157/50 (* 5 5))   ➡ (* 157/50 25)
```

「(* 5 5)」は
「25」で置き換わる

「157/50」とあるのは 3.14 のこと（有理数表示）

Stepper

File Edit Windows Help

Home << Previous Next >>

```
(define (area-of-disk r)
  (* 157/50 (* r r)))
```

```
(* 157/50 25)
```

➡ `157/2`

「(* 3.14 25)」は
「78.5」で置き換わる

「157/2」とあるのは 78.5 のこと（有理数表示）

(area-of-disk 5) から 78.5 が得られる過程

(area-of-disk 5)　最初の式

= (* 3.14 (* 5 5))

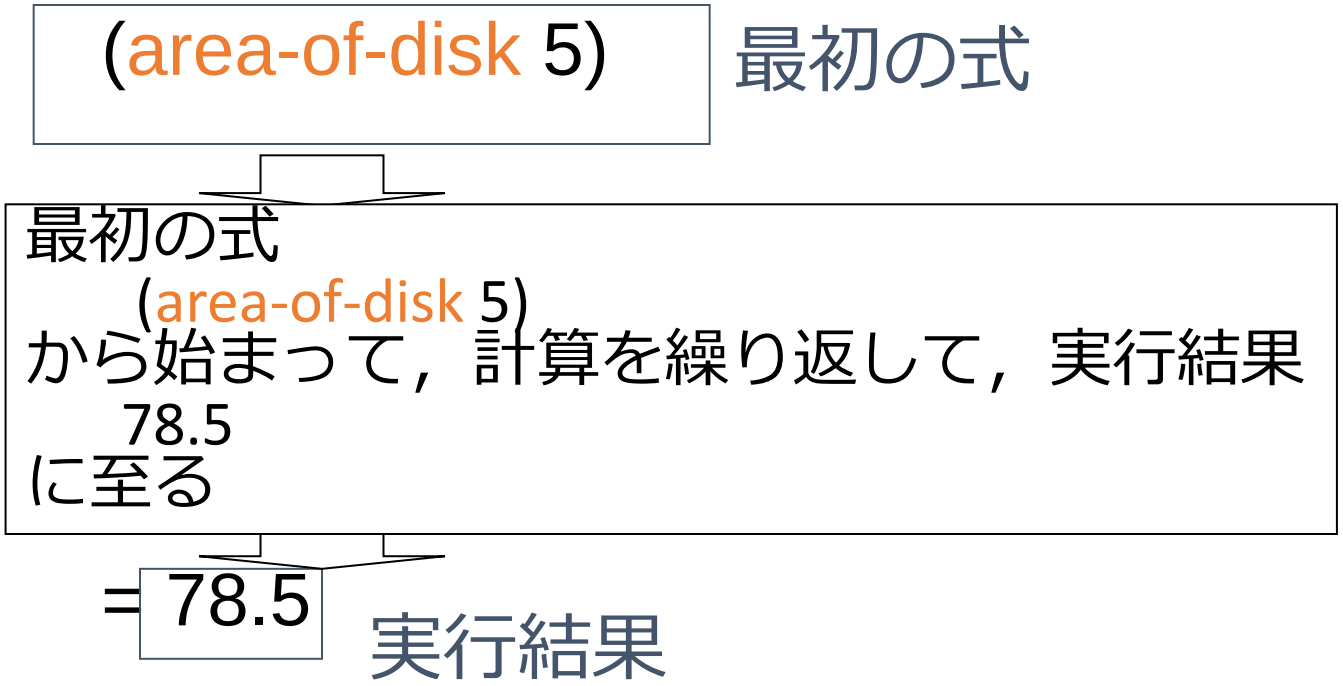
(* 3.14
(* r r))
に r = 5 が代入される

= (* 3.14 25)

(* 5 5) → 25 コンピュータ内部での計算

= 78.5　実行結果

(area-of-disk 5) から 78.5 が得られる過程



(area-of-disk 5) から 78.5 が得られる過程

(area-of-disk 5)

= (* 3.14 (* 5 5))

= (* 3.14 25)

これは,

```
(define (area-of-disk r)
  (* 3.14
     (* r r)))
```

の r を 5 で置き換えたもの

例題３．２乗の和

- x と y とから，x^2+y^2 を求めるプログラム sum-of-squares を作り，実行する
  - x の値から x^2 を求める関数 sqr と組み合わせる

「例題３．２乗の和」の手順

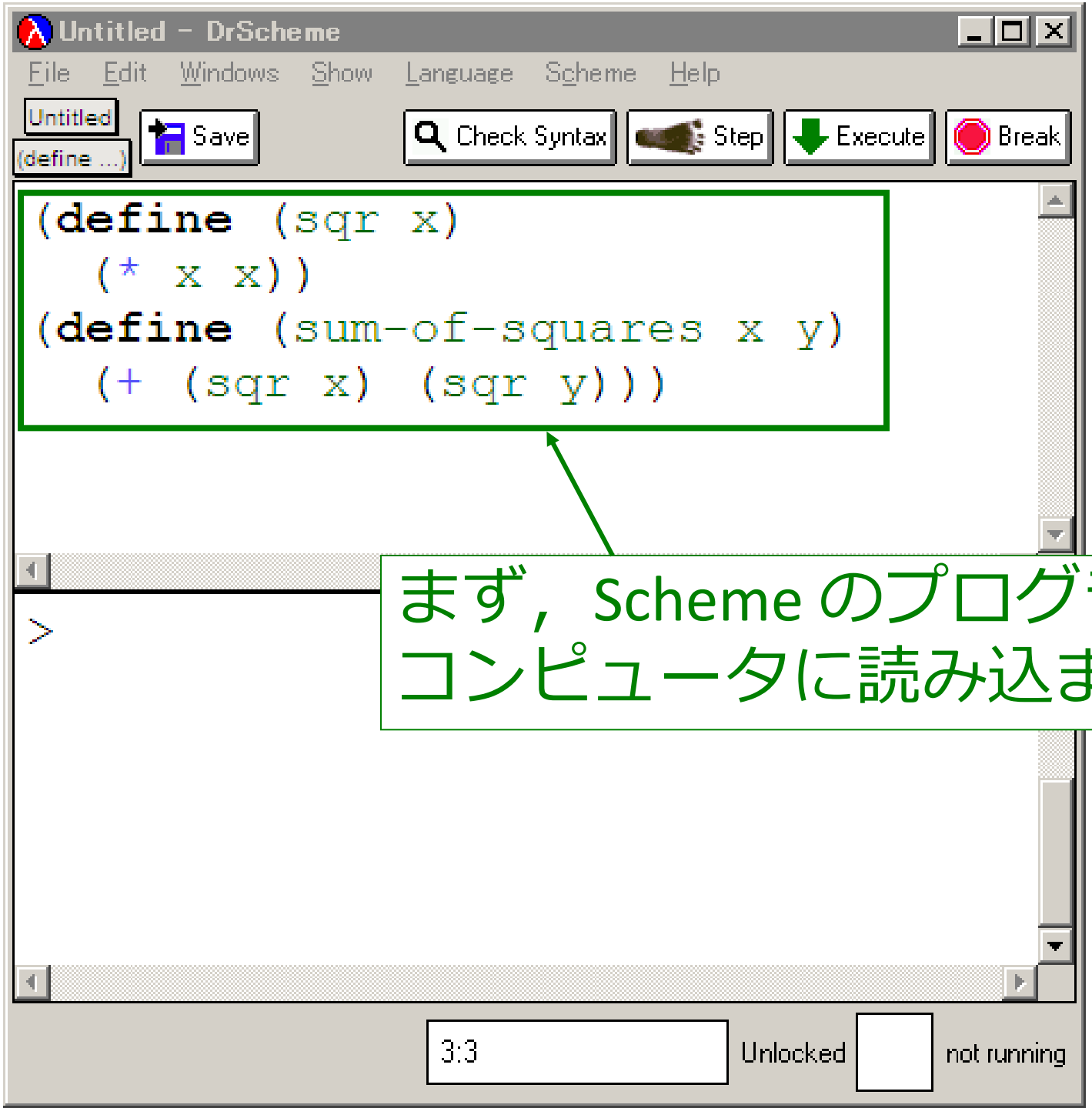
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Execute ボタンを押す

```
(define (sqr x)
  (* x x))
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

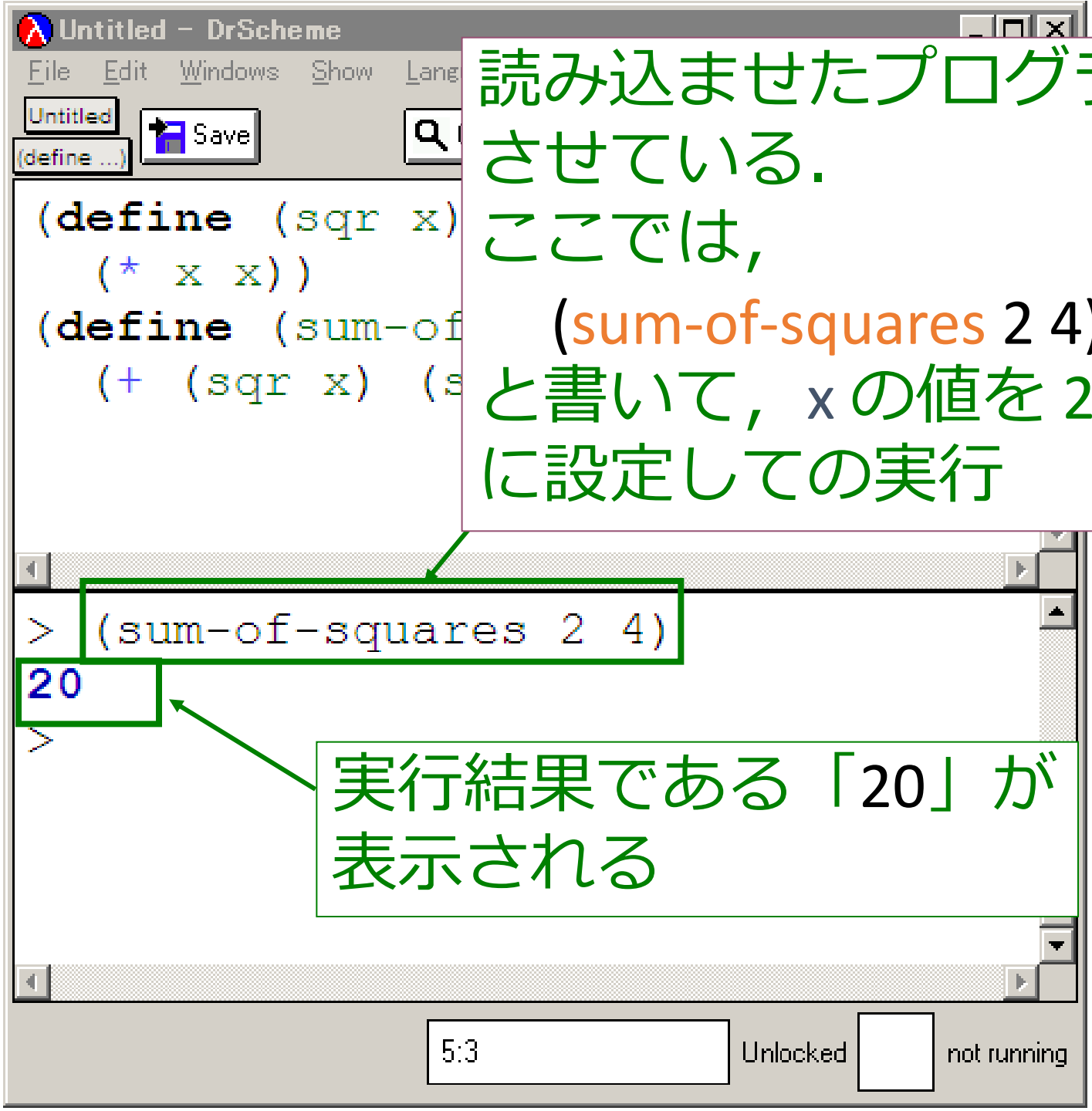
2. その後，次を「実行用ウインドウ」で実行しなさい

```
(sum-of-squares 2 4)
(sum-of-squares 20 30)
```

☆　次は，例題４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

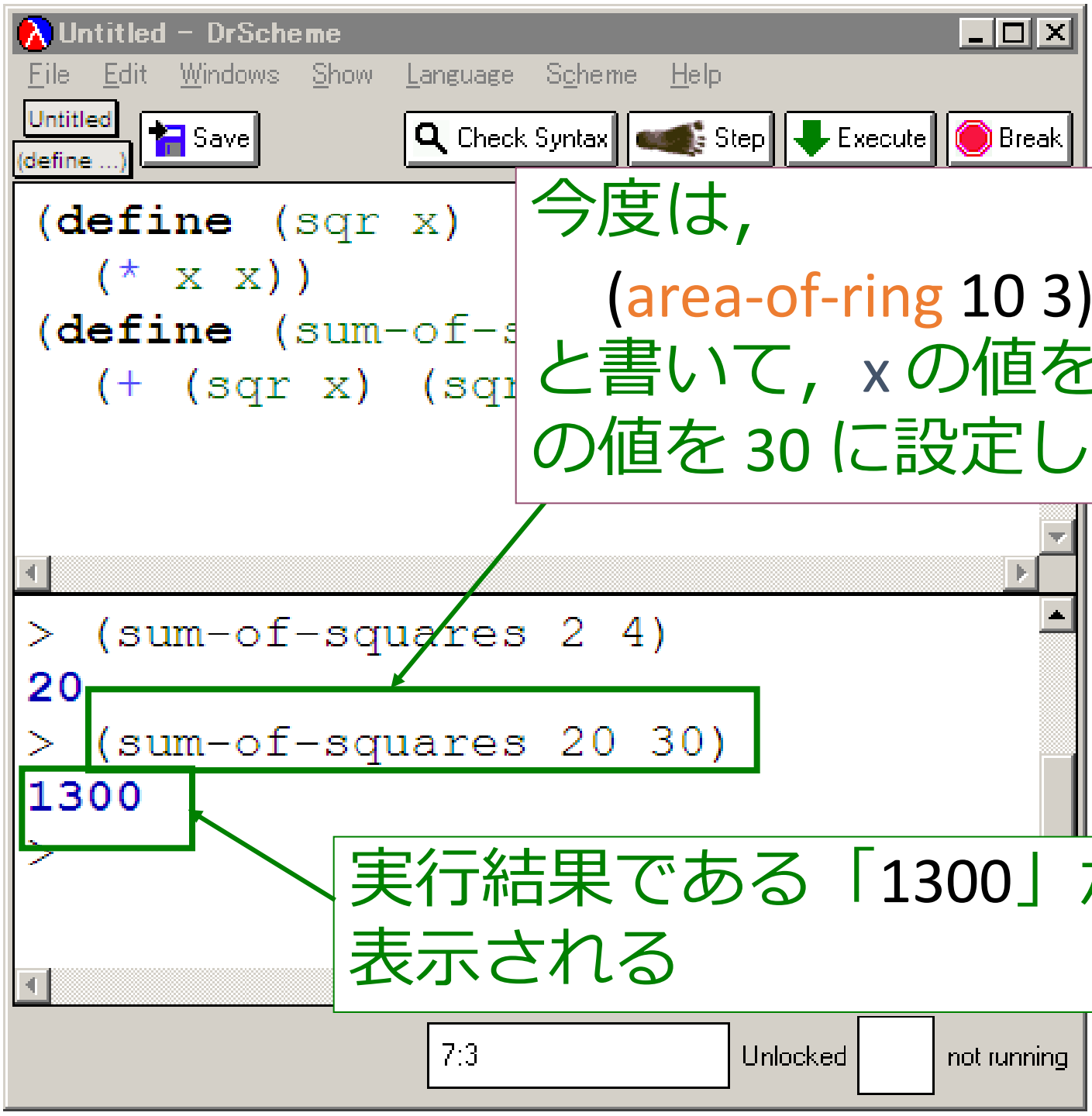


読み込ませたプログラムを実行させている.
ここでは,

(sum-of-squares 2 4)

と書いて，x の値を 2, yの値を 4 に設定しての実行

実行結果である「20」が表示される



今度は,

(area-of-ring 10 3)

と書いて，x の値を 20, y の値を 30 に設定しての実行

実行結果である「1300」が表示される

入力と出力

xの値：

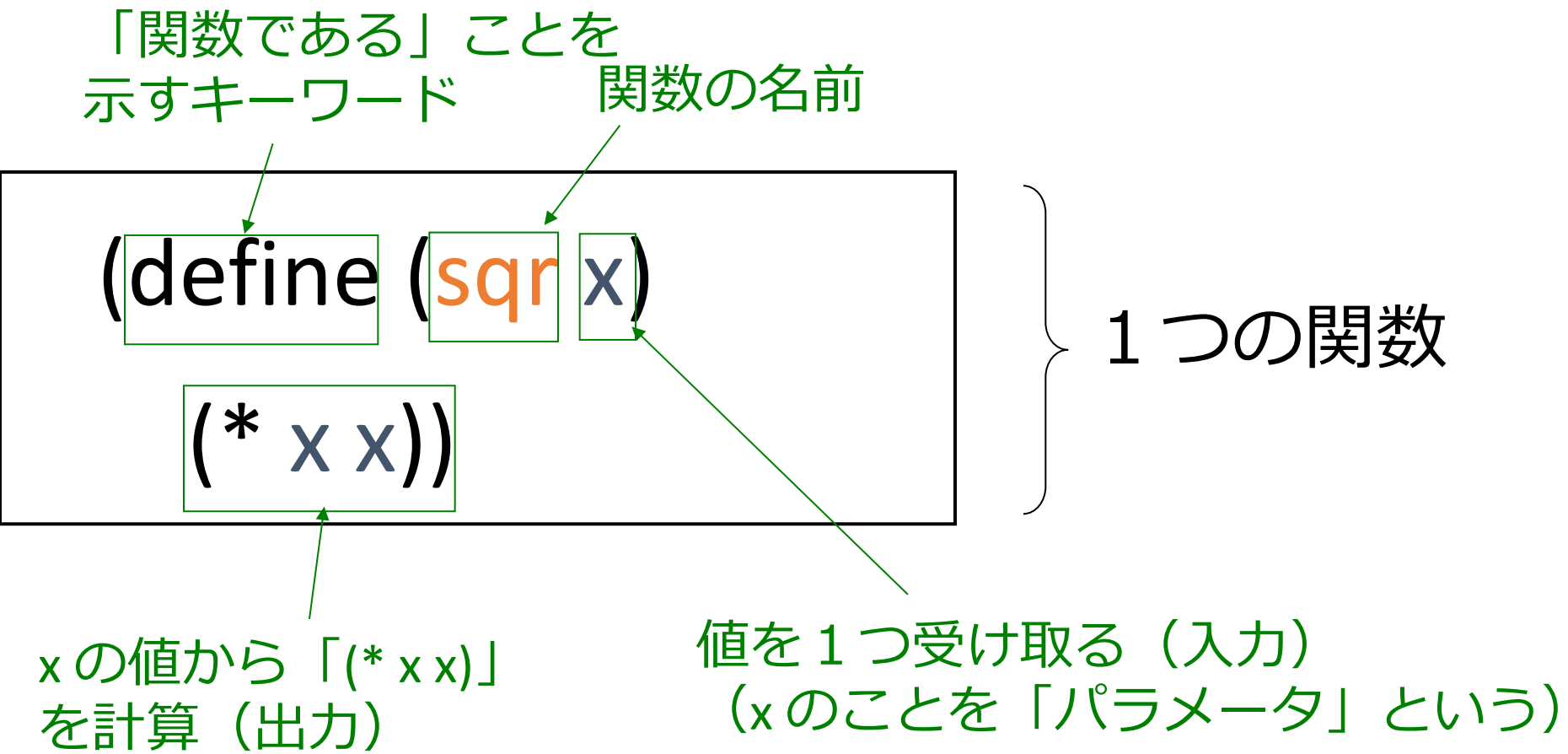
4

16

入力は
１つの数値

出力は
１つの数値

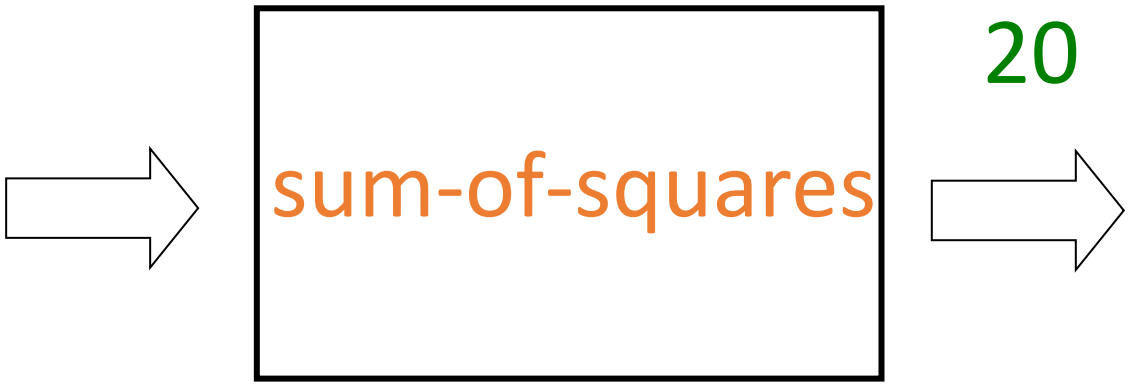
sqr 関数


「関数である」ことを
示すキーワード
関数の名前
(define (sqr x)
(* x x))
１つの関数
x の値から「(* x x)」
を計算（出力）
値を１つ受け取る（入力）
（x のことを「パラメータ」という）

入力と出力

x, y の値

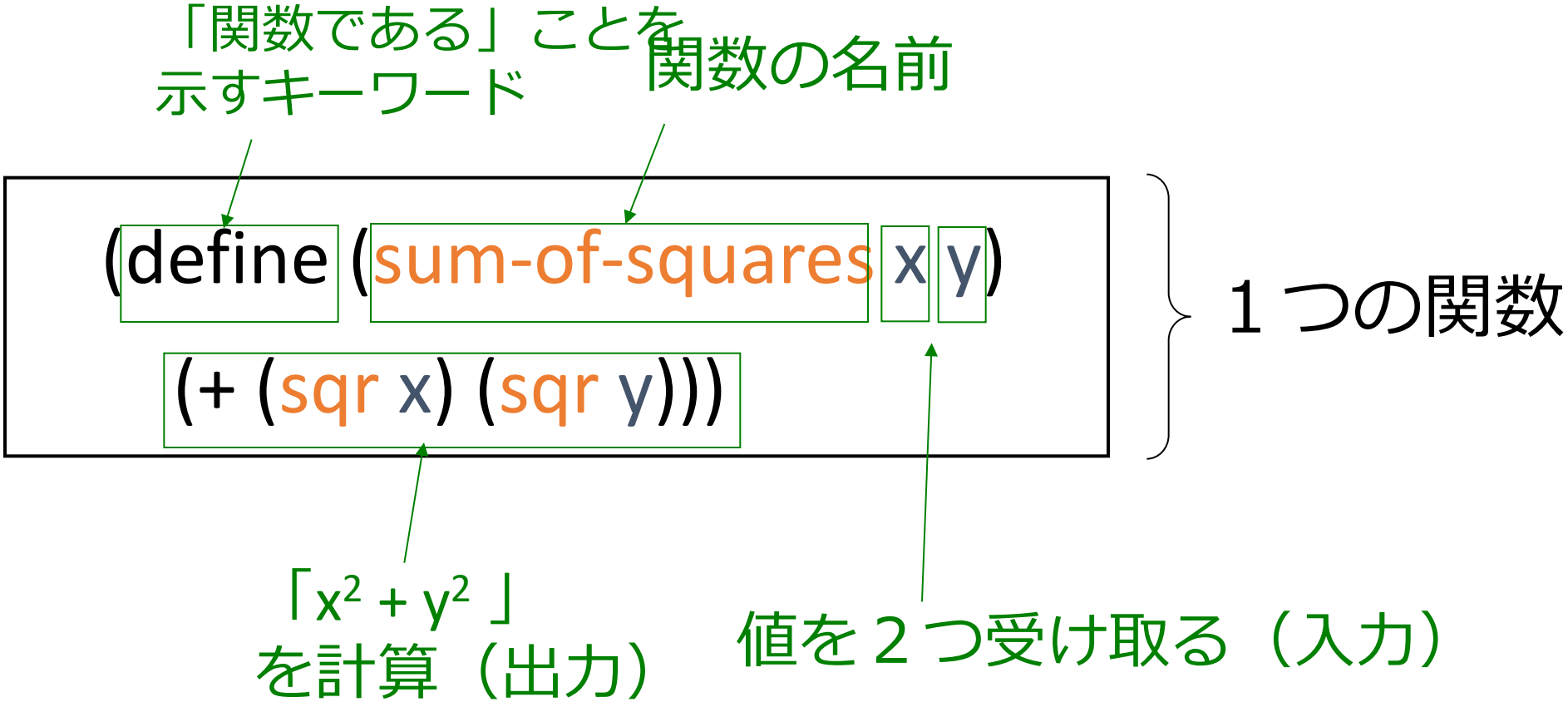
2, 4



入力は
２つの数値

出力は
１つの数値

sum-of-squares 関数



２乗の和のプログラム

- x^2+y^2　を求める

```
(define (sqr x)
  (* x x))
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

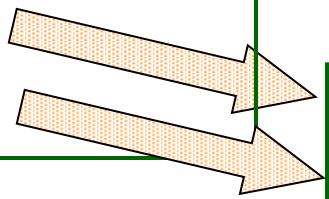
sqr
の部分

sum-of-squares
の部分

関数の関係

sum-of-squares 関数

```
(define (sum-of-squares x y)
   (+ (sqr x) (sqr y)))
```



sqr 関数

```
(define (sqr x)
   (* x x))
```

sum-of-squares 関数の中で，
sqr 関数を使っている（２箇所）

データの流れ

sum-of-squares 関数

```
(define (sum-of-squares x y)
   (+ (sqr x) (sqr y)))
```

① 数値を，
sqr 関数に渡す

②渡された値を，
「x」という名前で使う

sqr 関数

```
(define (sqr x)
   (* x x))
```

③実行結果を
sum-of-squares 関数に返す

関数を分割する理由

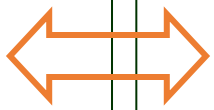
分割する場合

```
(define (sqr x)
    (* x x))
(define (sum-of-squares x
  y)
    (+ (sqr x) (sqr y)))
```

分割しない場合

```
(define (sum-of-squares x y)
    (+ (* x x) (* y y)))
```

「働き」は同じ



自分にとって「分かりやすい」書き方で書くことが重要

例題４．ステップ実行

- 関数 sum-of-squares （例題３）について，実行結果に至る過程を見る
  - (sum-of-squares 20 30) から 1300 に至る過程を見る
  - DrScheme の stepper を使用する

```
(define (sqr x)
   (* x x))
(define (sum-of-squares x y)
   (+ (sqr x) (sqr y)))
```

```
(sum-of-squares 20 30)
 = (+ (sqr 20) (sqr 30))
 = (+ (* 20 20) (sqr 30))
 = (+ 400 (sqr 30))
 = (+ 400 (* 30 30))
 = (+ 400 900)
 = 1300
```

例題４．ステップ実行

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

```
(define (sqr x)
   (* x x))
(define (sum-of-squares x y)
   (+ (sqr x) (sqr y)))
(sum-of-squares 20 30)
```

例題３と同じ

ステップ実行したいので，入力済みのプログラムは，消さずに残しておく

例題３に１行書き加える

2. DrScheme を使って，ステップ実行の様子を確認しなさい　（Step ボタン，Next ボタンを使用）
   - 理解しながら進むこと

☆　次は，例題５に進んでください

Home << Previous Next >>

```
(define (sqr x) (* x x))
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

```
(sum-of-squares
  20
  30)
```

```
(+
 (sqr 20)
 (sqr 30))
```

「(+ (sqr x) (sqr y))」の
「x」は「20」で「y」は「30」で置き換わる

Home | << Previous | Next >>

```
(define (sqr x) (* x x))
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

```
(+                          (+
 (sqr 20)          ➡        (* 20 20)
 (sqr 30))                   (sqr 30))
```

「(* x x)」の
「x」は「20」で置き換わる

Home << Previous Next >>

```
(define (sqr x) (* x x))
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

```
(+                          (+ 400 (sqr 30))
 (* 20 20)
 (sqr 30))
```

乗算により
「(* 20 20)」は「400」で
置き換わる

Home << Previous Next >>

```
(define (sqr x) (* x x))
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

```
(+ 400 (sqr 30)) ➡ (+ 400 (* 30 30))
```

「(* x x)」の
「x」は「30」で置き換わる

Home << Previous Next >>

```
(define (sqr x) (* x x))
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

```
(+ 400 (* 30 30)) → (+ 400 900)
```

乗算により
「(* 30 30)」は「900」で
置き換わる

Home << Previous Next >>

```
(define (sqr x) (* x x))
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

```
(+ 400 900)
```

➡ 1300

加算により
「(+ 400 900)」は「1300」で
置き換わる

(sum-of-squares 20 30) から 1300 が得られる過程

(sum-of-squares 20 30) 最初の式

= (+ (sqr 20) (sqr 30))　(+ (sqr x) (sqr y)) に x = 2, y = 4 が代入される

= (+ (* 20 20) (sqr 30))　(* x x)) に x = 2 が代入される

= (+ 400 (sqr 30))　(* 2 2) → 4

= (+ 400 (* 30 30))　(* x x)) に x = 4 が代入される

= (+ 400 900)　(* 4 4) → 16　コンピュータ内部での計算

= 1300 実行結果

(sum-of-squares 20 30) から 1300 が得られる過程

(sum-of-squares 20 30)

= (+ (sqr 20) (sqr 30))

= (+ (* 20 20) (sqr 30))

=

=

= (+ 400 900)

= 1300

これは,

```
(define (sum-of-squares x y)
  (+ (sqr x) (sqr y)))
```

の x を 20 で，y を 30 で置き換えたもの

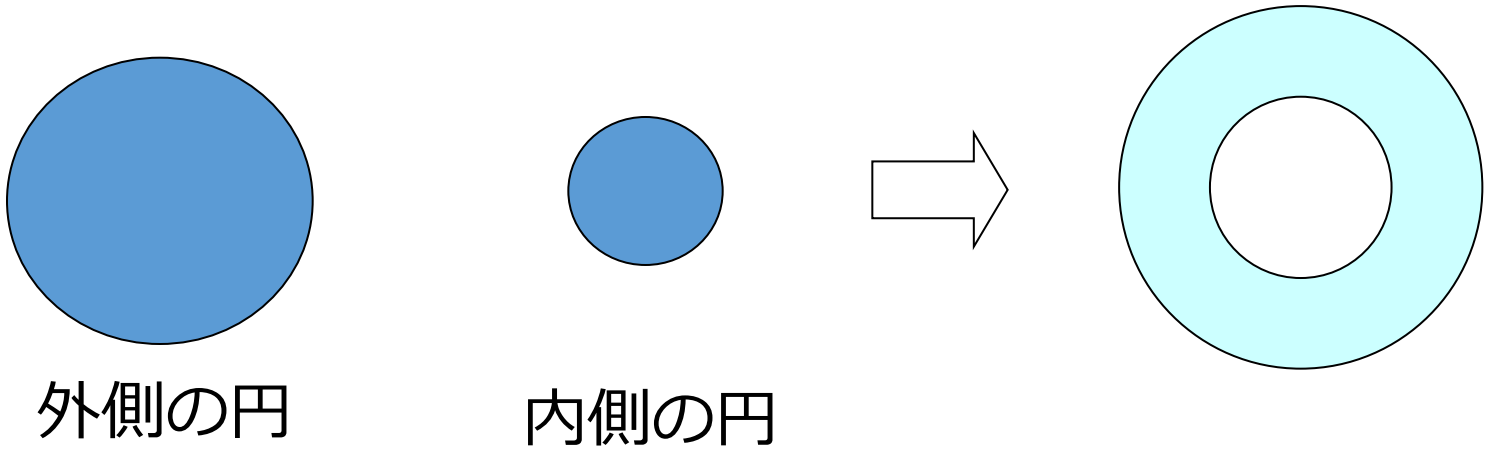
例題５．リングの面積

- 外径 outer, 内径 inner からリングの面積を求めるプログラム area-of-ring を作り，実行する
  - 円の面積を求める関数 area-of-disk と組み合わせる

真ん中に穴のあいた
円の面積と考える

リングの面積

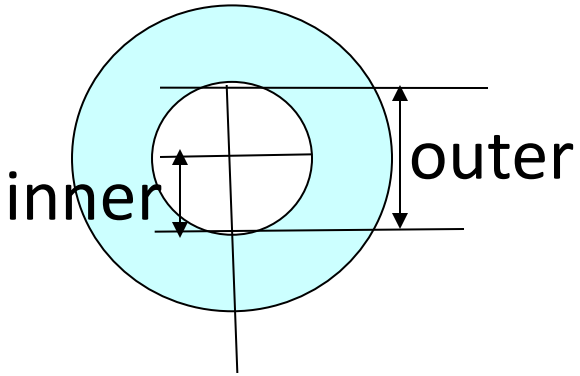
- 求める面積は，外側の円の面積から，
  内側の穴の円の面積を引いたもの



リングの面積

外径：outer

内径： inner



リングの面積

＝　外側の円の面積　－　内側の円の面積

半径 outer の円　　半径 inner の円

「例題５．リングの面積」の手順

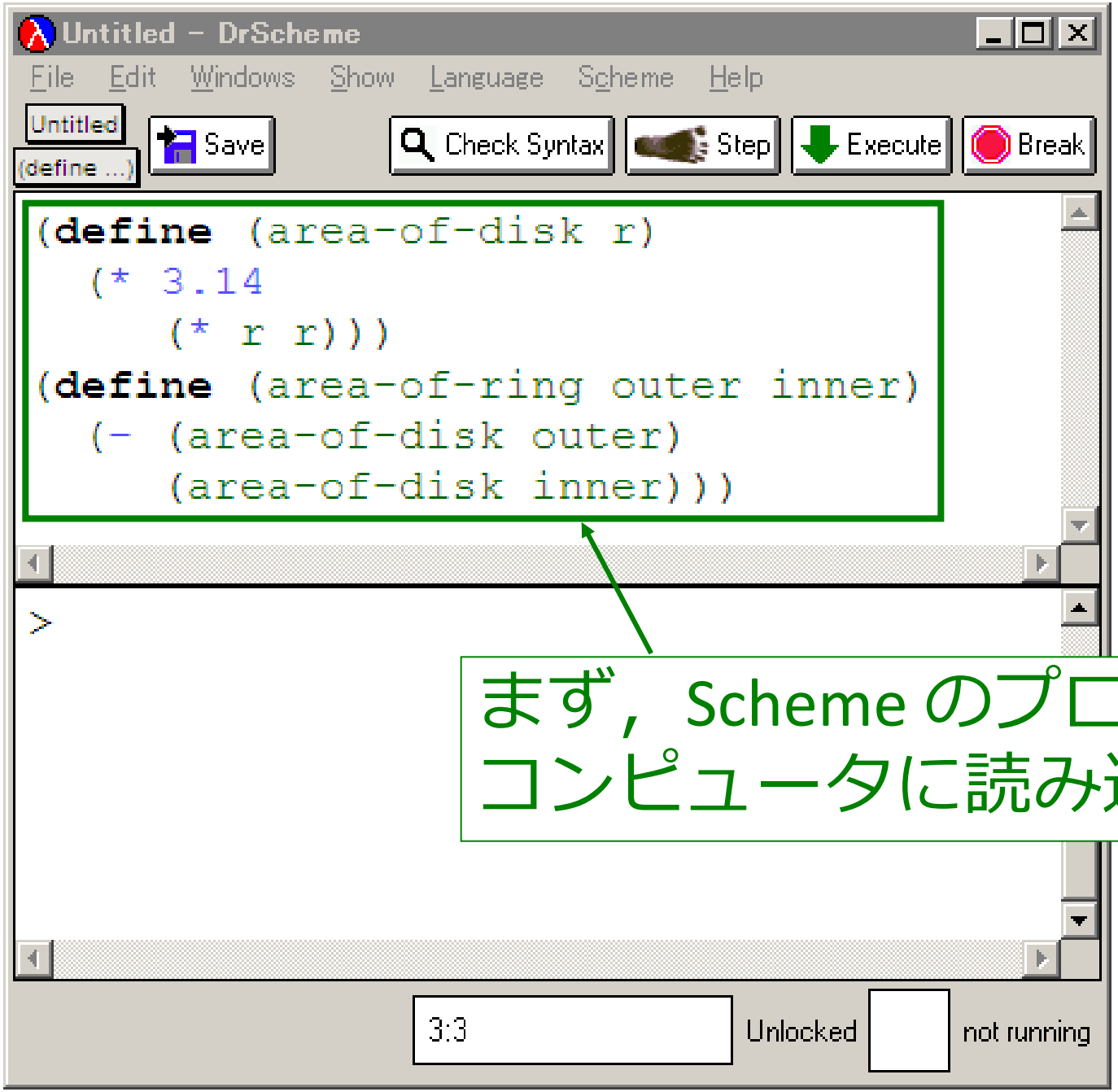
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Execute ボタンを押す

```
(define (area-of-disk r)
  (* 3.14
     (* r r)))
(define (area-of-ring outer inner)
  (- (area-of-disk outer)
     (area-of-disk inner)))
```

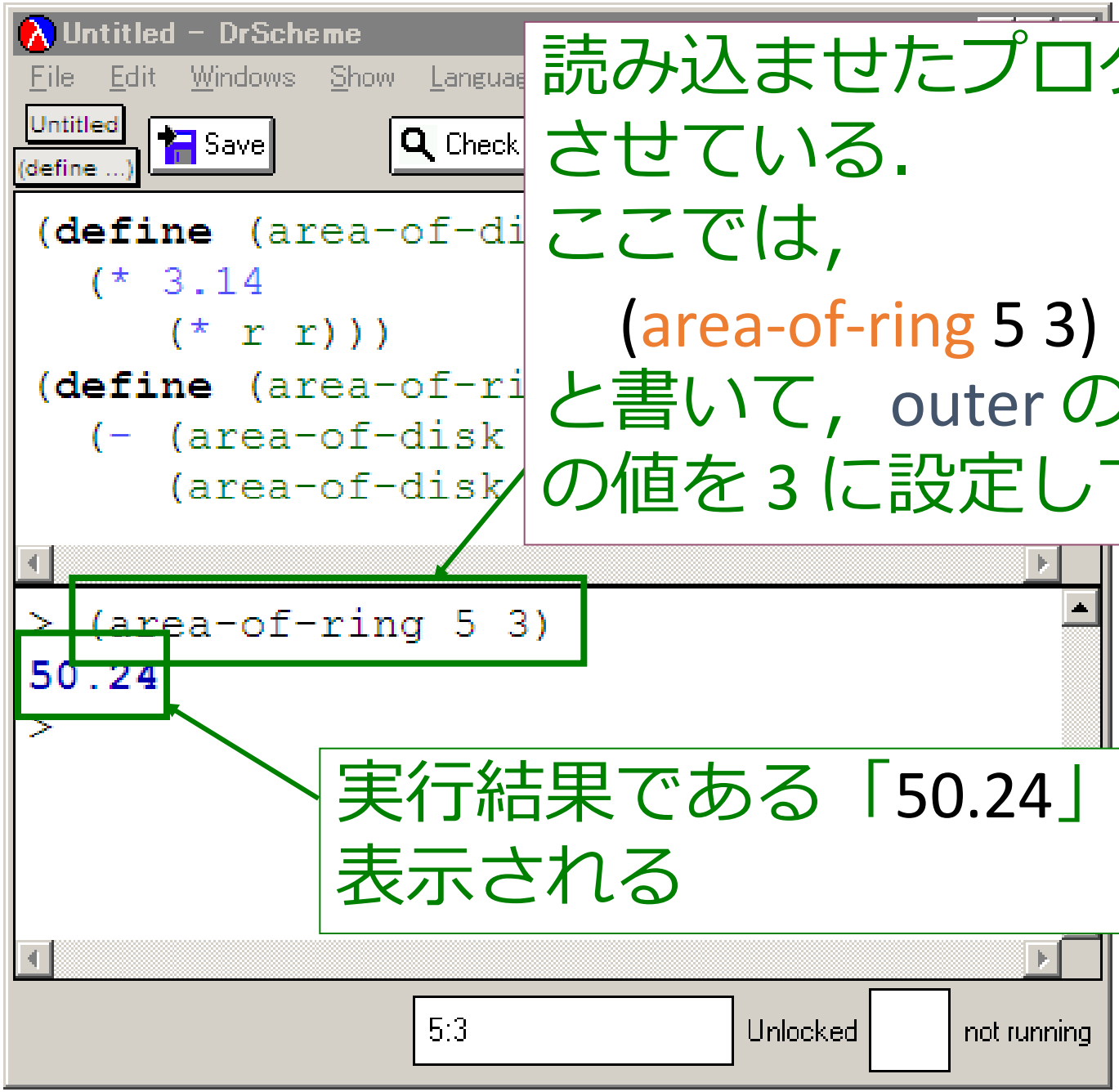
2. その後，次を「実行用ウインドウ」で実行しなさい

```
(area-of-ring 5 3)
```

☆　次は，例題６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

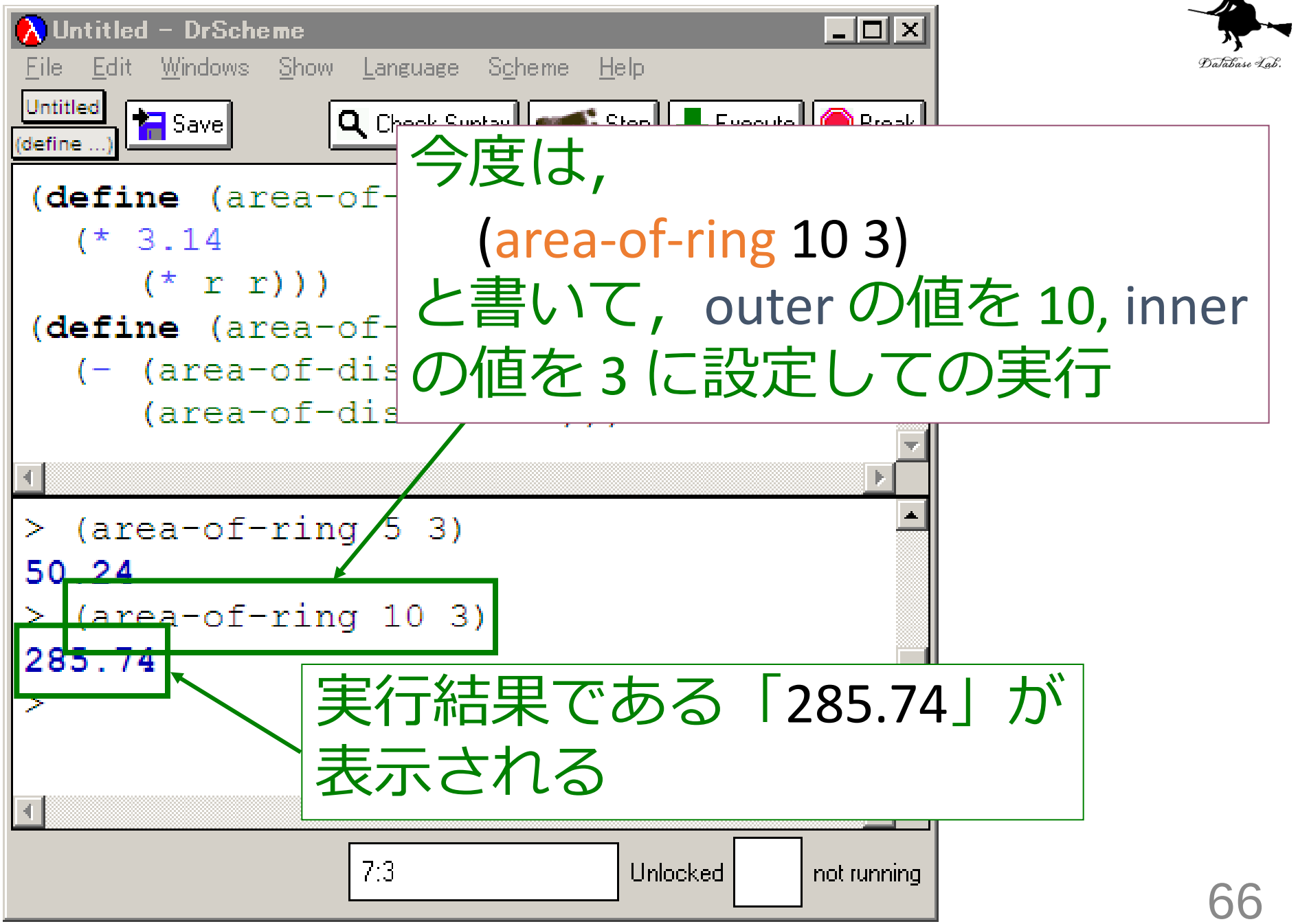


読み込ませたプログラムを実行させている．
ここでは，

(area-of-ring 5 3)

と書いて，outer の値を 5, inner の値を 3 に設定しての実行

実行結果である「50.24」が表示される

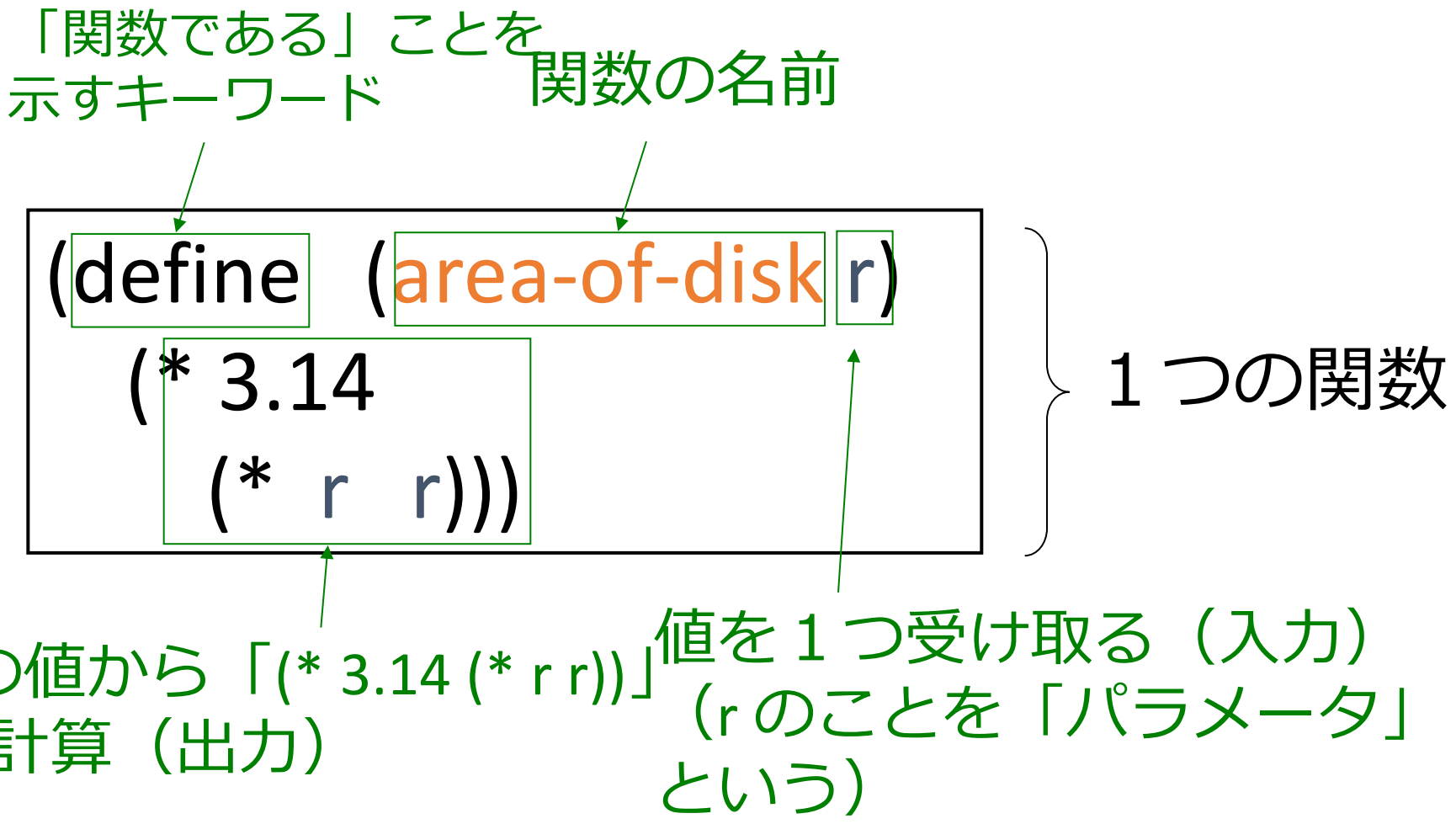

Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Save
(define (area-of-
(* 3.14
(* r r)))
(define (area-of-
(- (area-of-dis
(area-of-dis
> (area-of-ring 5 3)
50.24
> (area-of-ring 10 3)
285.74
>
7:3
Unlocked
not running
今度は,
(area-of-ring 10 3)
と書いて，outer の値を 10, inner の値を 3 に設定しての実行
実行結果である「285.74」が表示される

入力と出力

入力は
１つの数値

出力は
１つの数値

area-of-disk 関数

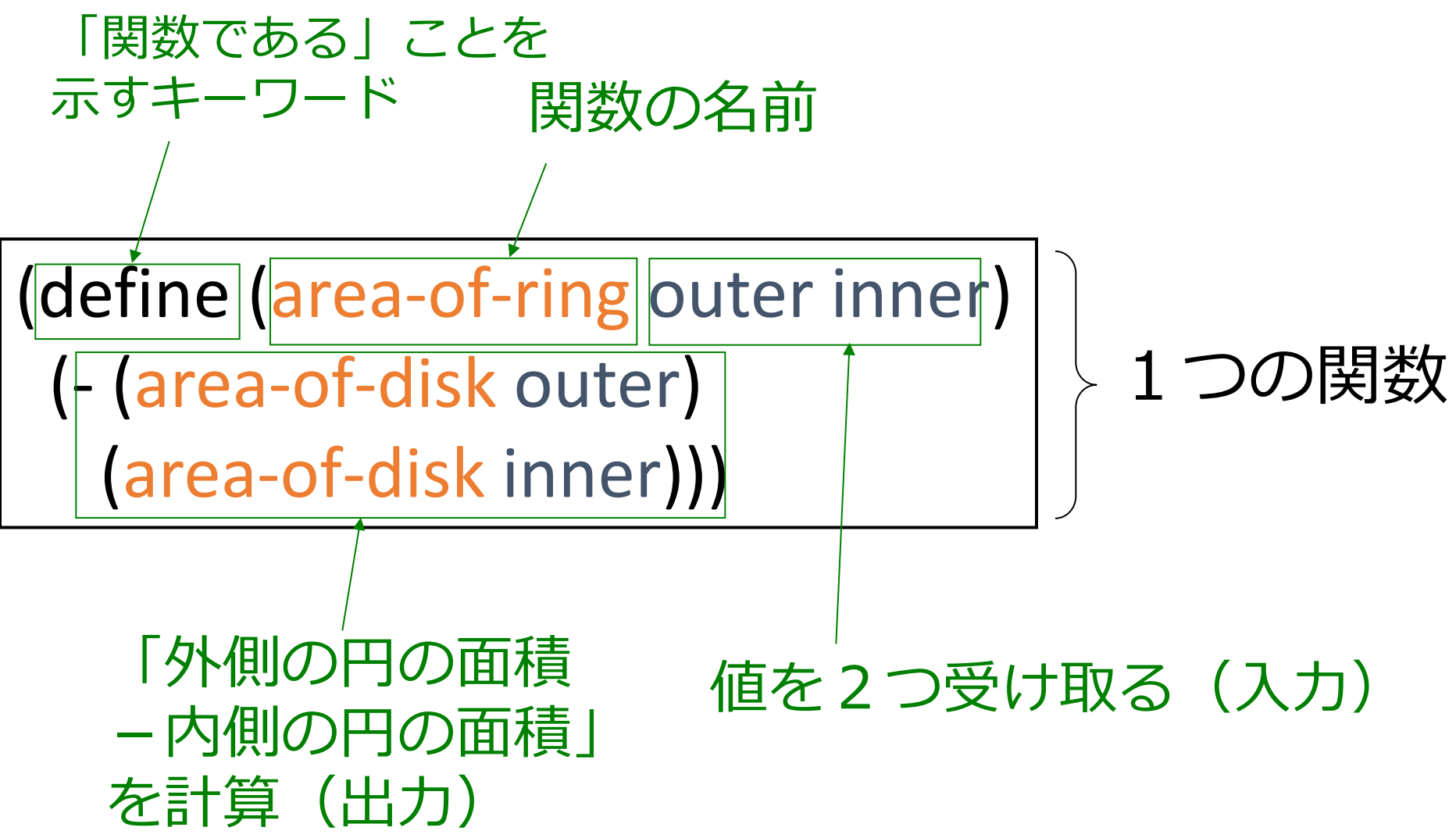


入力と出力

入力は
２つの数値

出力は
１つの数値

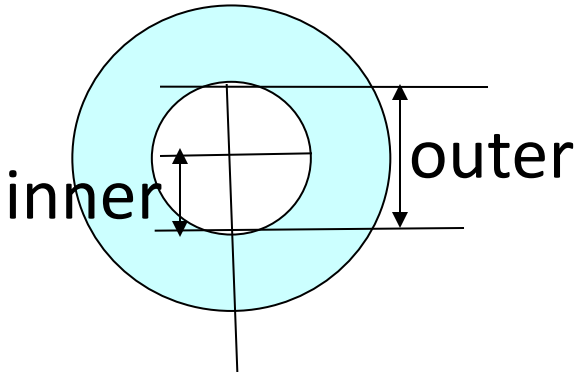
area-of-ring 関数



リングの面積のプログラム

外径： outer

内径： inner




```
(define (area-of-disk r)
  (* 3.14
     (* r r)))
(define (area-of-ring outer inner)
  (- (area-of-disk outer)
     (area-of-disk inner)))
```

area-of-disk の部分

area-of-ring の部分

関数の関係

area-of-ring 関数

```
(define (area-of-ring outer inner)
   (-  (area-of-disk outer)
       (area-of-disk inner)))
```

area-of-disk 関数

```
(define   (area-of-disk   r)
   (* 3.14
      (*  r   r)))
```

area-of-ring 関数の中で，
area-of-disk 関数を使っている（２箇所）

データの流れ

area-of-ring 関数

```
(define (area-of-ring outer inner)
   (-  (area-of-disk outer)
       (area-of-disk inner)))
```

① 数値を，
area-of-disk 関数に渡す

②渡された値を，
「r」という名前で使う

area-of-disk 関数

```
(define   (area-of-disk   r)
   (* 3.14
      (*  r   r)))
```

③実行結果を
area-of-ring 関数に返す

関数を分割する理由

分割する場合

```
(define   (area-of-disk   r)
   (* 3.14
      (*  r   r)))
(define (area-of-ring outer
  inner)
   (-  (area-of-disk outer)
      (area-of-disk inner)))
```

分割しない場合

```
(define (area-of-ring outer
    inner)
   (-  (* 3.14 (* outer outer))
      (* 3.14 (* inner inner))))
```

「働き」は同じ

リングの面積は，「外側の円の面積」－「内側の円の面積」であることがひと目で分かる

例題６．ステップ実行

- 関数 area-of-ring （例題５）について，実行結果に至る過程を見る
  - (area-of-ring 5 3) から 50.24 に至る過程を見る
  - DrScheme の stepper を使用する

```
(define (area-of-disk r)
  (* 3.14
     (* r r)))
(define (area-of-ring outer inner)
  (- (area-of-disk outer)
     (area-of-disk inner)))
```

```
(area-of-ring 5 3)
= (- (area-of-disk 5) (area-of-disk 3))
= (- (* 3.14 (* 5 5)) (area-of-disk 3))
= (- (* 3.14 25) (area-of-disk 3))
= (- 78.5 (area-of-disk 3))
= (- 78.5 (* 3.14 (* 3 3)))
= (- 78.5 (* 3.14 9))
= (- 78.5 28.26)
= 50.24
```

例題６．ステップ実行

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

```
(define   (area-of-disk   r)
   (* 3.14
      (*  r   r)))
(define (area-of-ring outer inner)
 (-  (area-of-disk outer)
      (area-of-disk inner)))
(area-of-ring 5 3)
```

例題５と同じ

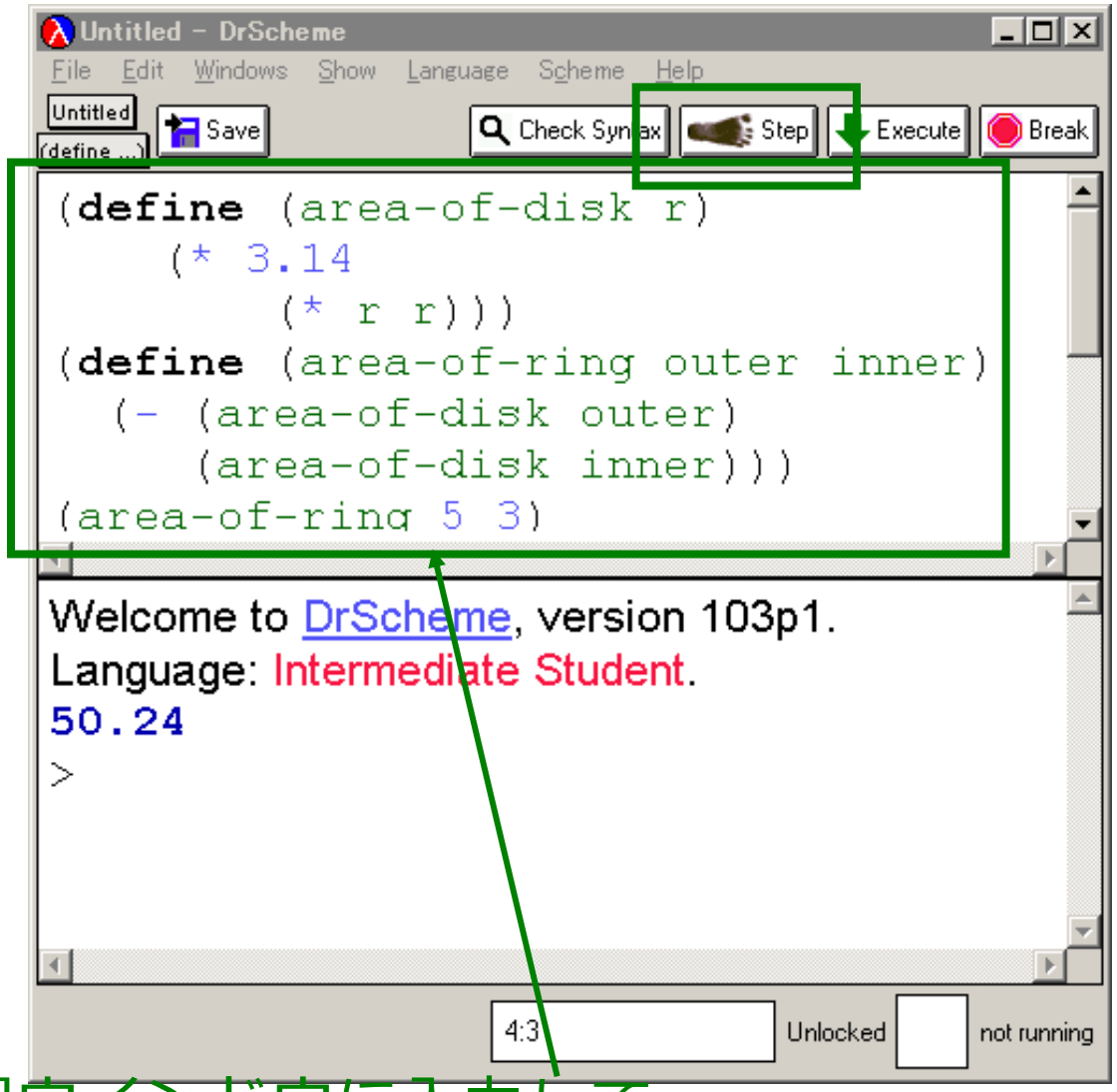
ステップ実行したいので，入力済みのプログラムは，消さずに残しておく

例題５に１行書き加える

2. DrScheme を使って，ステップ実行の様子を確認しなさい　（Step ボタン，Next ボタンを使用）
   - 理解しながら進むこと

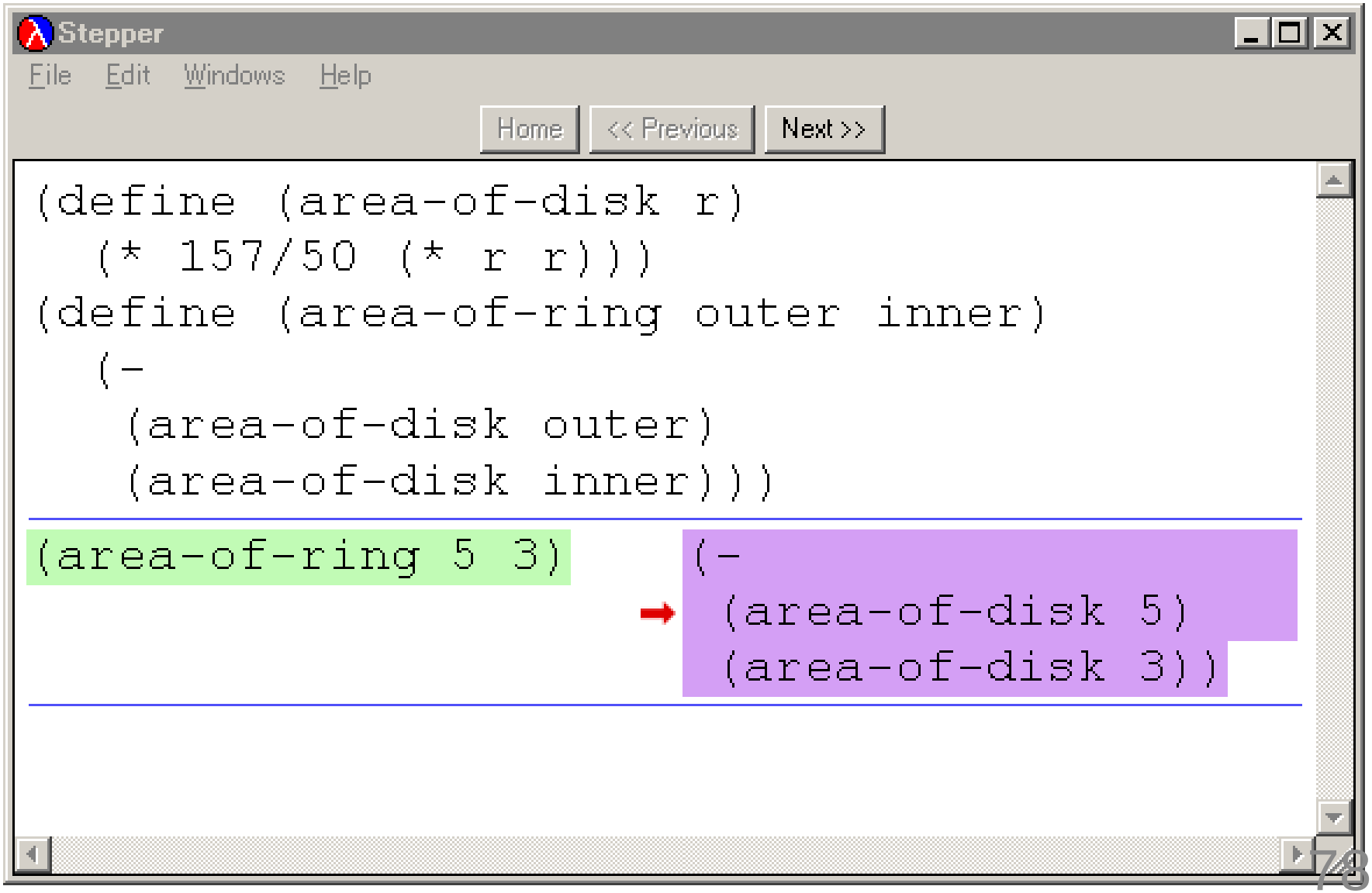
☆　次は，例題７に進んでください

例題６．ステップ実行

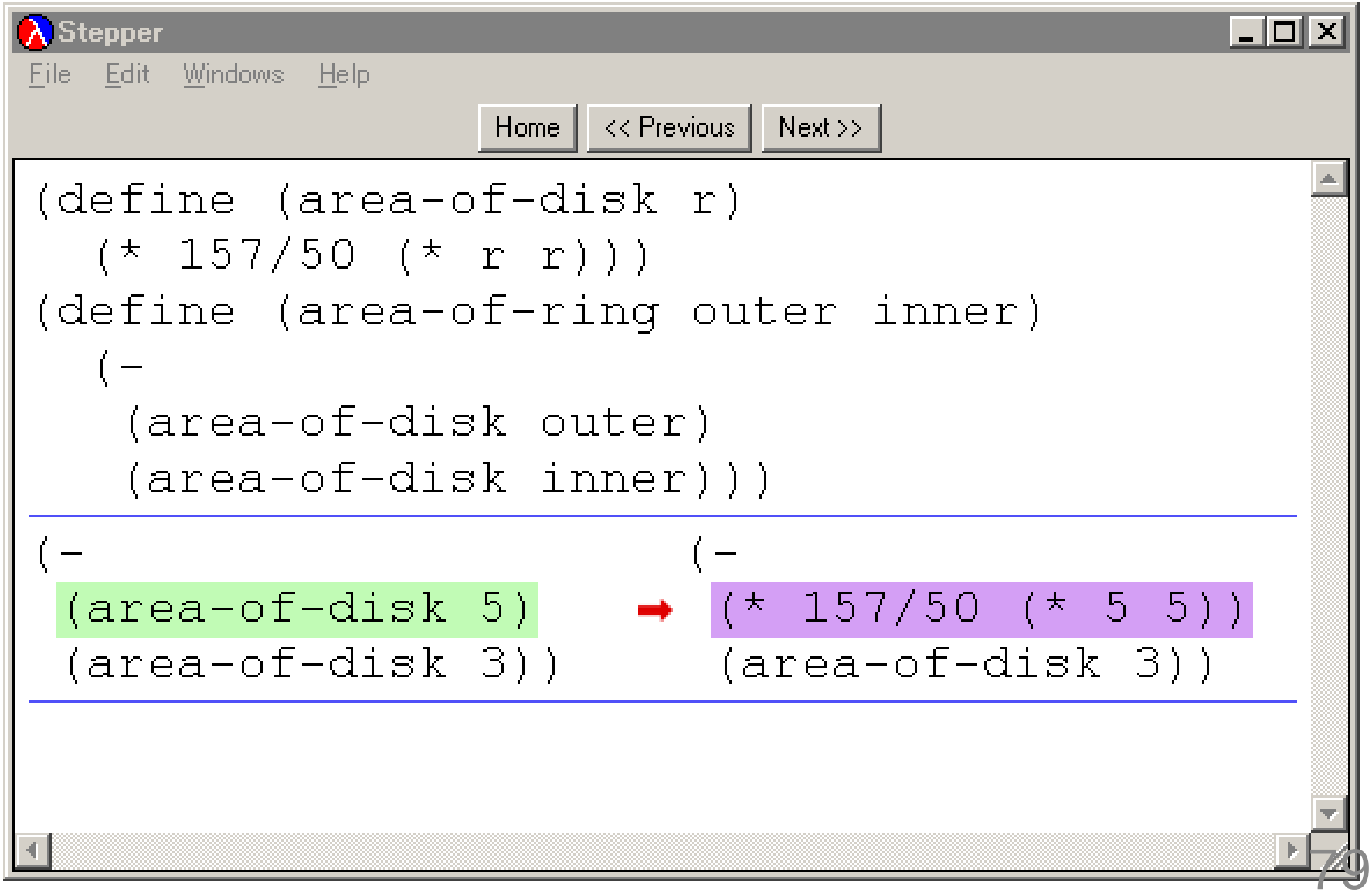


定義用ウインドウに入力して，
Execute ボタンを押した後，Step ボタンを押すと

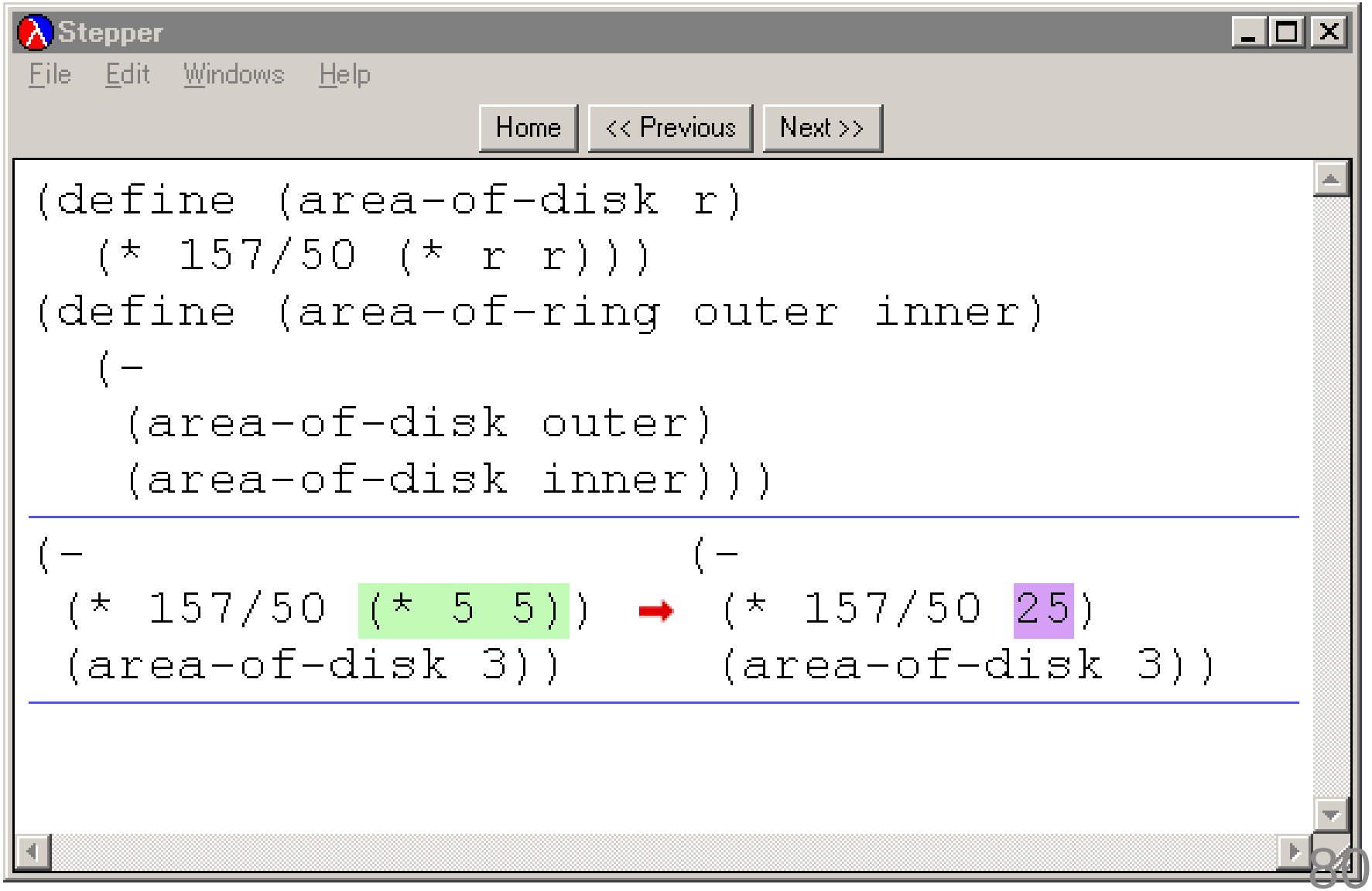
例題６．ステップ実行


Stepper
File Edit Windows Help
Home << Previous Next >>
(define (area-of-disk r)
(* 157/50 (* r r)))
(define (area-of-ring outer inner)
(-
(area-of-disk outer)
(area-of-disk inner)))
(area-of-ring 5 3)
(-
(area-of-disk 5)
(area-of-disk 3))

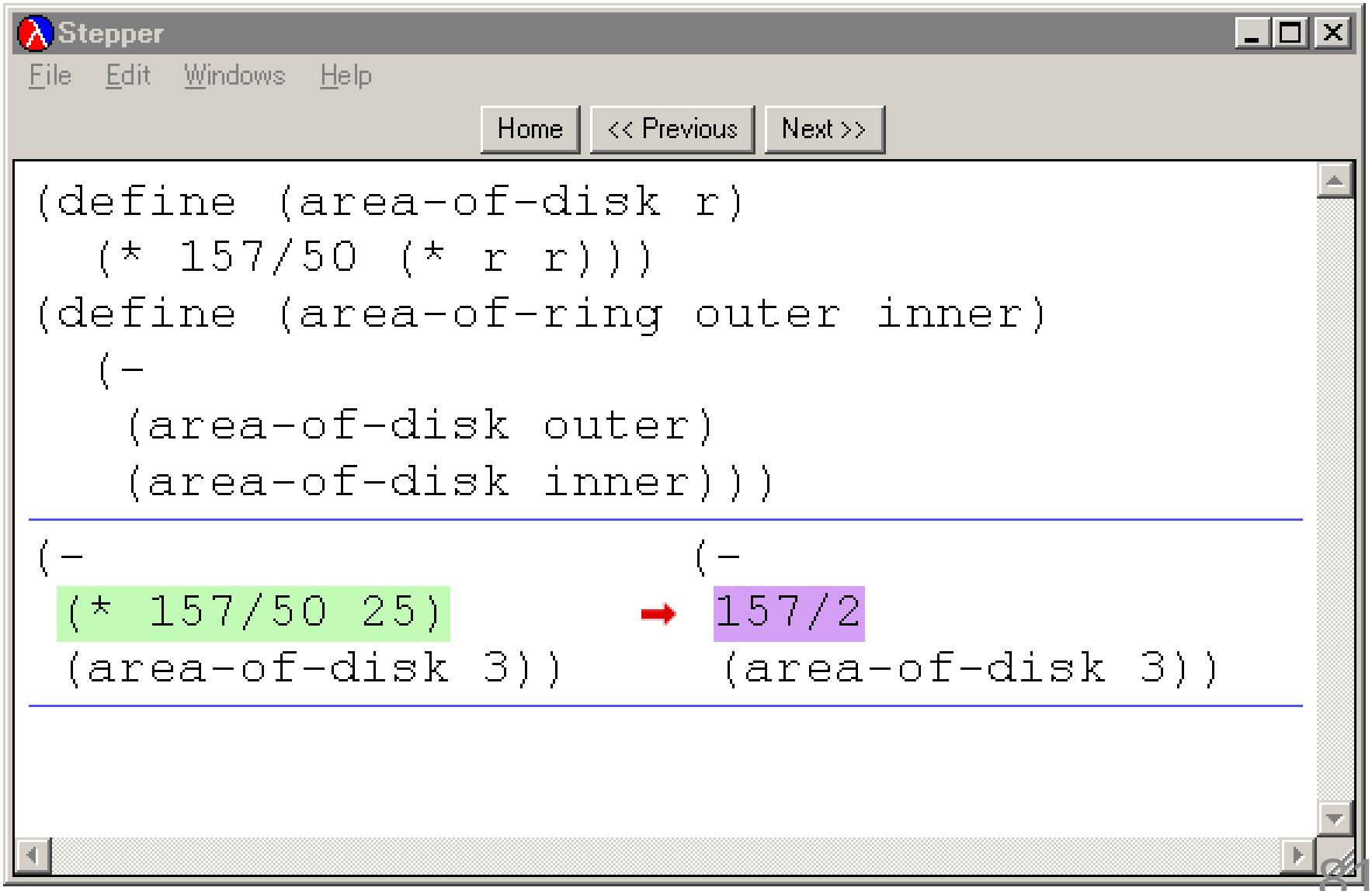
例題６．ステップ実行


Stepper
File Edit Windows Help
Home << Previous Next >>
(define (area-of-disk r)
(* 157/50 (* r r)))
(define (area-of-ring outer inner)
(-
(area-of-disk outer)
(area-of-disk inner)))
(-
(area-of-disk 5)
(area-of-disk 3))
(-
(* 157/50 (* 5 5))
(area-of-disk 3))

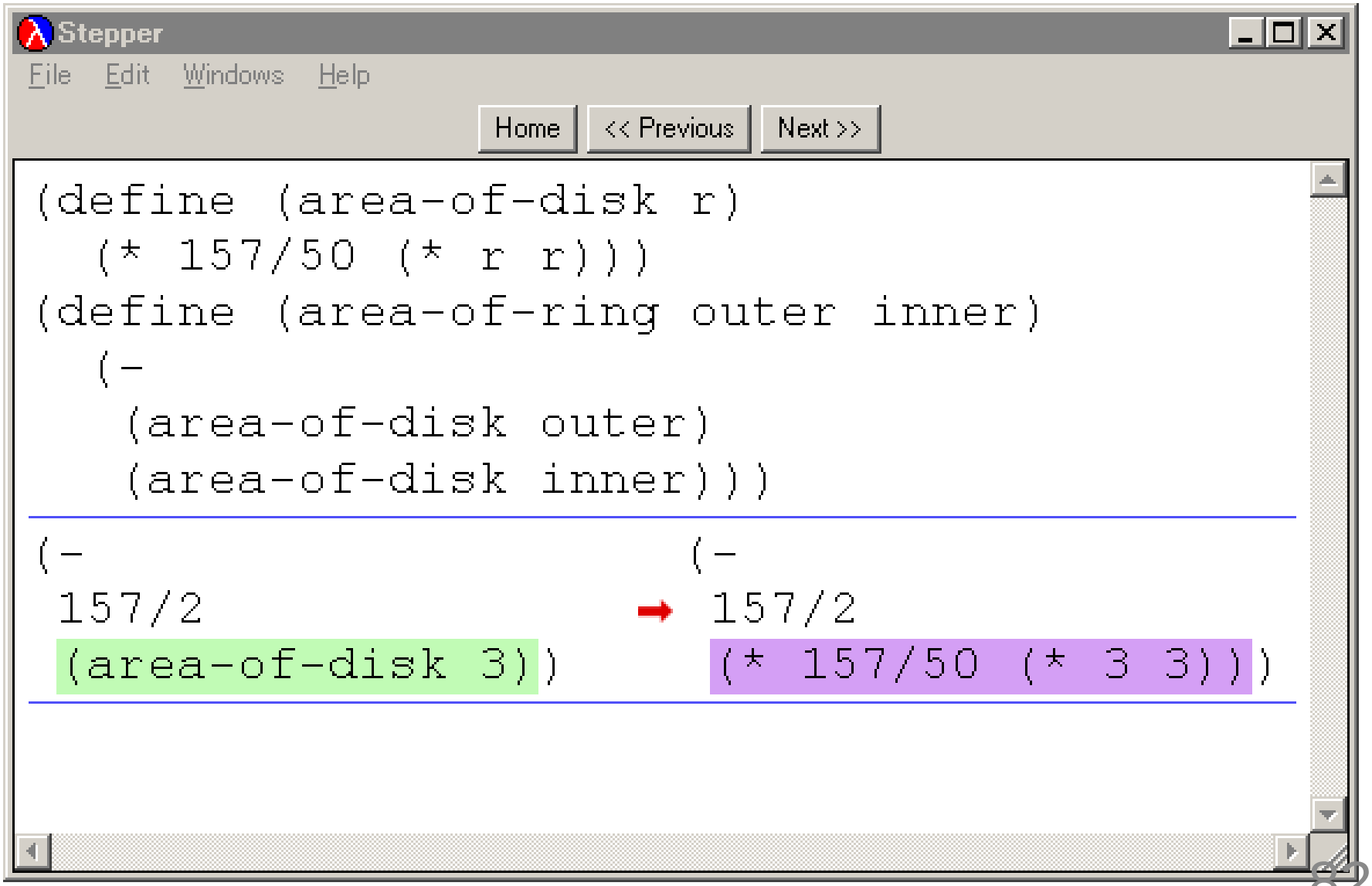
例題６．ステップ実行


Stepper
File Edit Windows Help
Home
<< Previous
Next >>
(define (area-of-disk r)
(* 157/50 (* r r)))
(define (area-of-ring outer inner)
(-
(area-of-disk outer)
(area-of-disk inner)))
(-
(* 157/50 (* 5 5))
(area-of-disk 3))
(-
(* 157/50 25)
(area-of-disk 3))

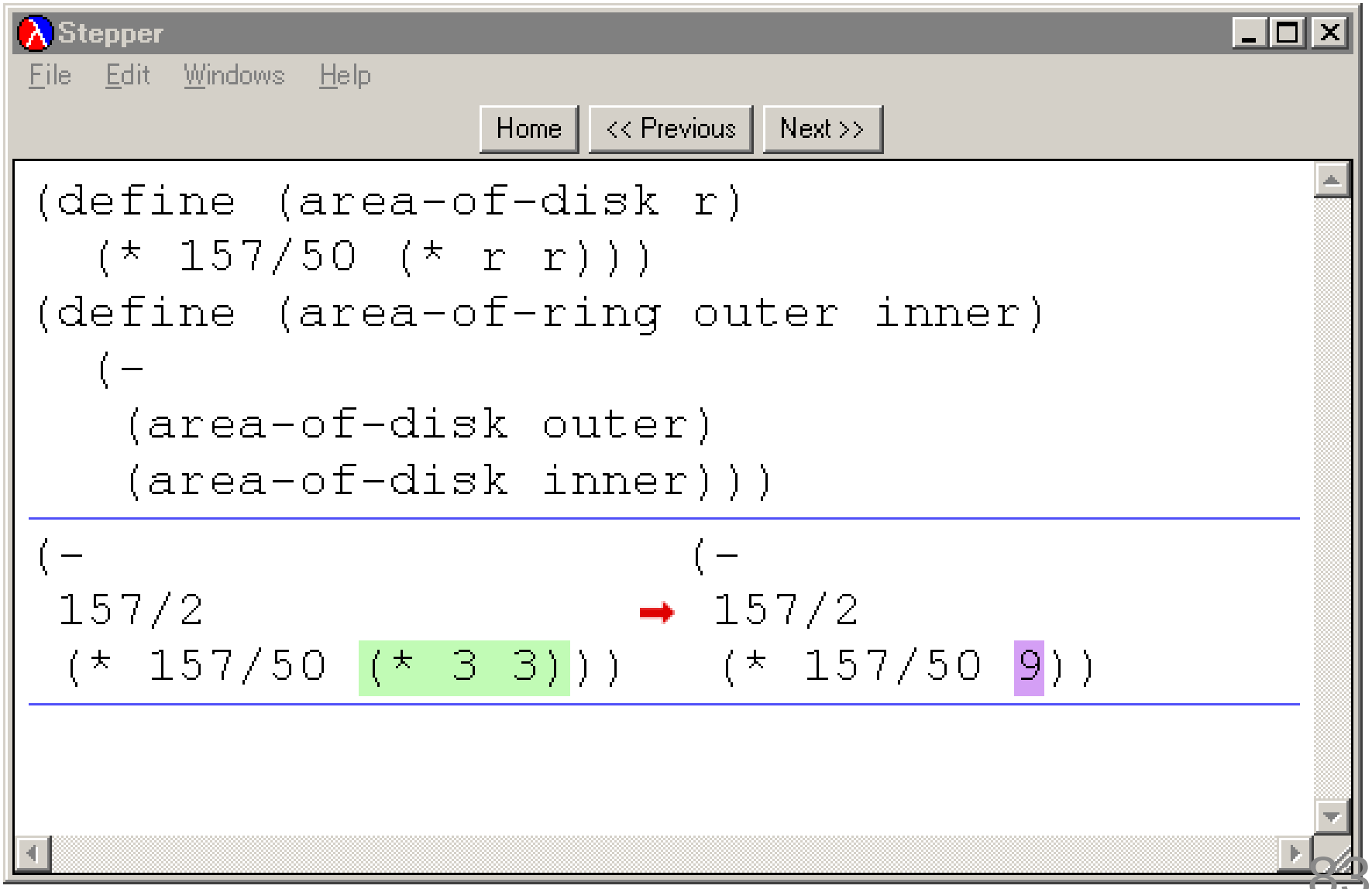
例題６．ステップ実行


Stepper
File Edit Windows Help
Home
<< Previous
Next >>
(define (area-of-disk r)
(* 157/50 (* r r)))
(define (area-of-ring outer inner)
(-
(area-of-disk outer)
(area-of-disk inner)))
(-
(* 157/50 25)
(area-of-disk 3))
(-
157/2
(area-of-disk 3))

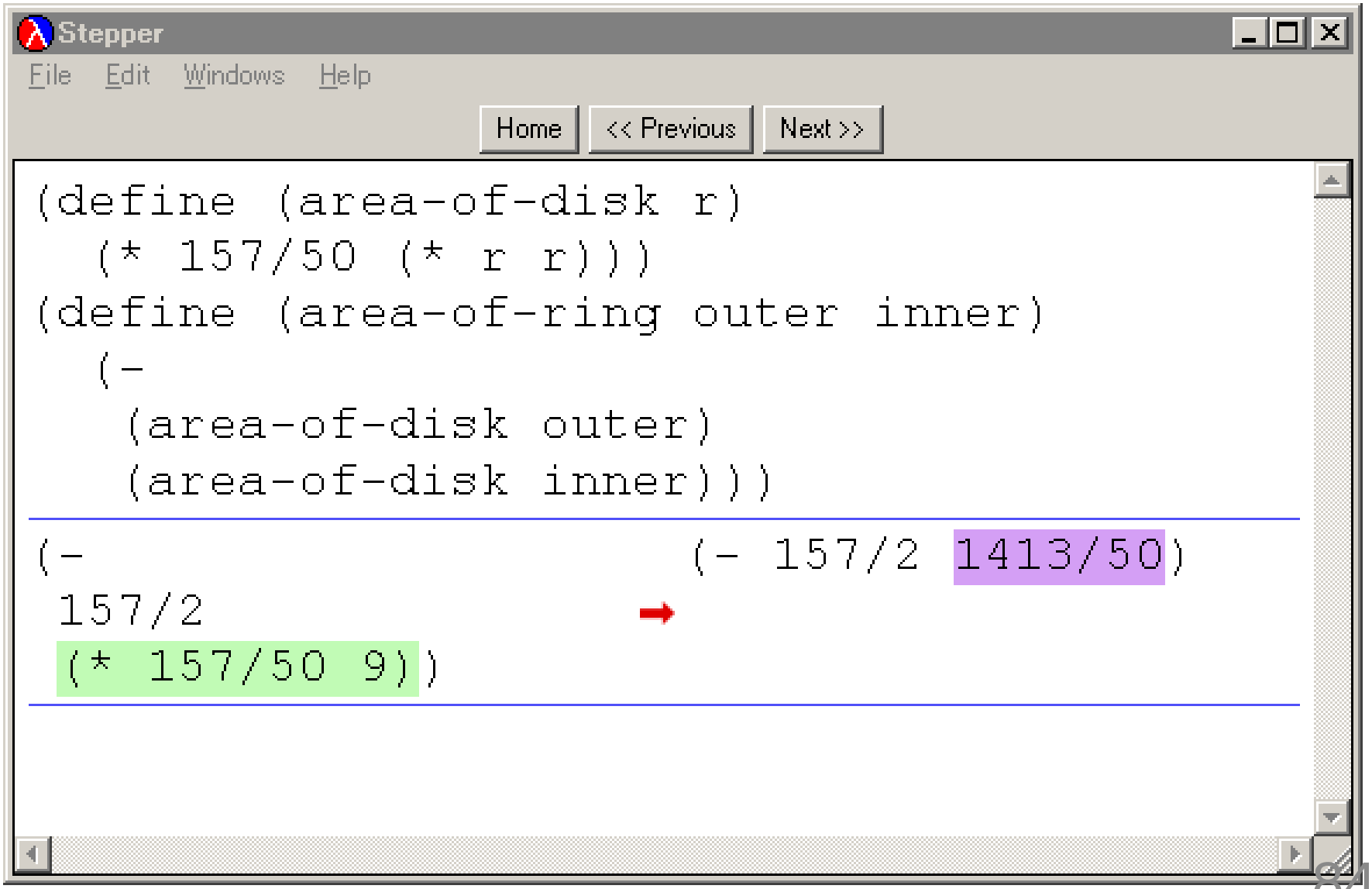
例題６．ステップ実行


Stepper
File Edit Windows Help
Home << Previous Next >>
(define (area-of-disk r)
(* 157/50 (* r r)))
(define (area-of-ring outer inner)
(-
(area-of-disk outer)
(area-of-disk inner)))
(-
157/2
(area-of-disk 3))
(-
157/2
(* 157/50 (* 3 3)))

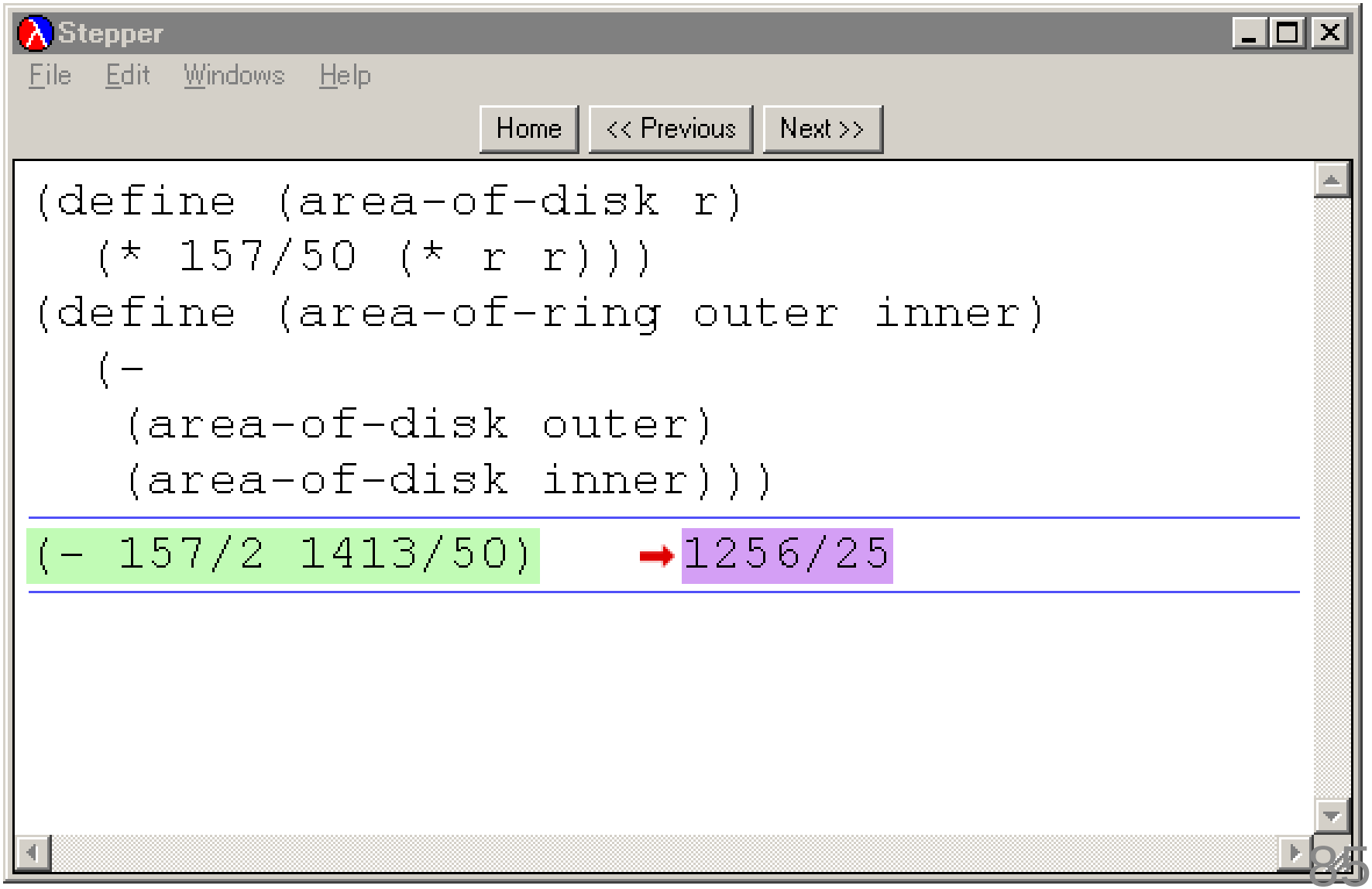
例題６．ステップ実行


Stepper
File Edit Windows Help
Home
<< Previous
Next >>
(define (area-of-disk r)
(* 157/50 (* r r)))
(define (area-of-ring outer inner)
(-
(area-of-disk outer)
(area-of-disk inner)))
(-
157/2
(* 157/50 (* 3 3)))
(-
157/2
(* 157/50 9))

例題６．ステップ実行


Stepper
File Edit Windows Help
Home << Previous Next >>
(define (area-of-disk r)
(* 157/50 (* r r)))
(define (area-of-ring outer inner)
(-
(area-of-disk outer)
(area-of-disk inner)))
(- (- 157/2 1413/50)
157/2
(* 157/50 9))

例題６．ステップ実行


Stepper
File Edit Windows Help
Home << Previous Next >>
(define (area-of-disk r)
(* 157/50 (* r r)))
(define (area-of-ring outer inner)
(-
(area-of-disk outer)
(area-of-disk inner)))
(- 157/2 1413/50)
1256/25

(area-of-ring 5 3) から 50.24 が得られる過程

(area-of-ring 5 3) 最初の式

= (- (area-of-disk 5) (area-of-disk 3))
(- (area-of-disk outer) (area-of-disk inner)) に outer = 5, inner = 3 が代入される

= (- (* 3.14 (* 5 5)) (area-of-disk 3))
(* 3.14 (* r r)) に r = 5 が代入される

= (- (* 3.14 25) (area-of-disk 3))
(* 5 5) → 25

= (- 78.5 (area-of-disk 3))
(* 3.14 25) → 78.5

= (- 78.5 (* 3.14 (* 3 3)))
(* 3.14 (* r r)) に r = 3 が代入される

= (- 78.5 (* 3.14 9))
(* 3 3) → 9

= (- 78.5 28.26)
(* 3.14 9) → 28.26
コンピュータ内部での計算

= 50.24 実行結果

(area-of-ring 5 3) から 50.24 が得られる過程

(area-of-ring 5 3)

= (- (area-of-disk 5) (area-of-disk 3))

= (- (* 3.14 (* 5 5)) (area-of-disk 3))

これは,

```
(define (area-of-ring outer inner)
  (- (area-of-disk outer)
     (area-of-disk inner)))
```

の outer を 5 で，inner を 3 で置き換えたもの

= 50.24

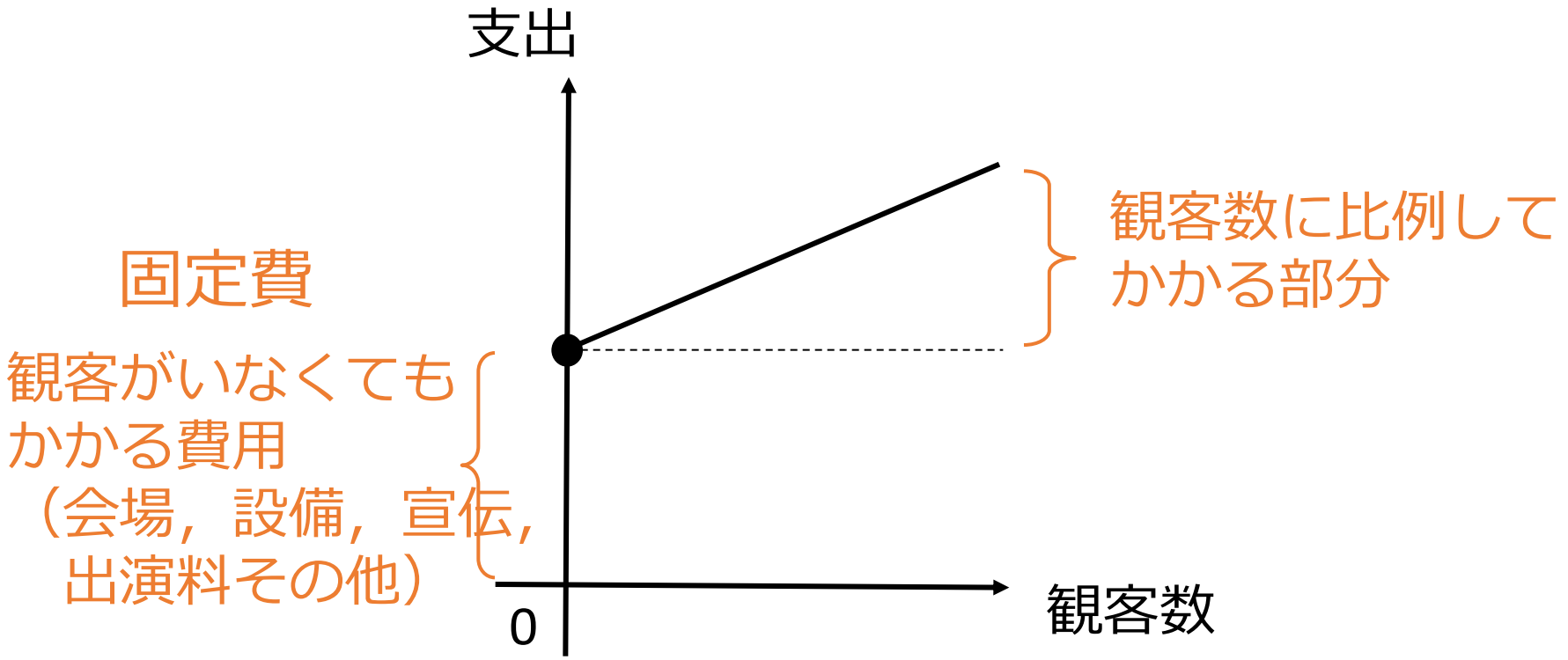
例題７．利益の計算

- 公演での利益を求めるプログラムを作り，実行する
- チケット代 ticket-price から利益、収入、支出、観客数を求める関数 profit, revenue, cost, attendees の作成
  - profit:　利益 ＝ 収入(revenue) － 費用(cost)
  - revenue: 収入 ＝ 観客数(attendees) × チケット代(ticket-price)
  - cost:　支出 ＝ 固定費 ＋ 観客数(attendees) × 費用

    → 「固定費」と「費用」は公演ごとに異なる
  - attendees: チケット代(ticket-price)と観客数には関係がある

    → この「関係」は公演ごとに異なる

支出の見積もり式

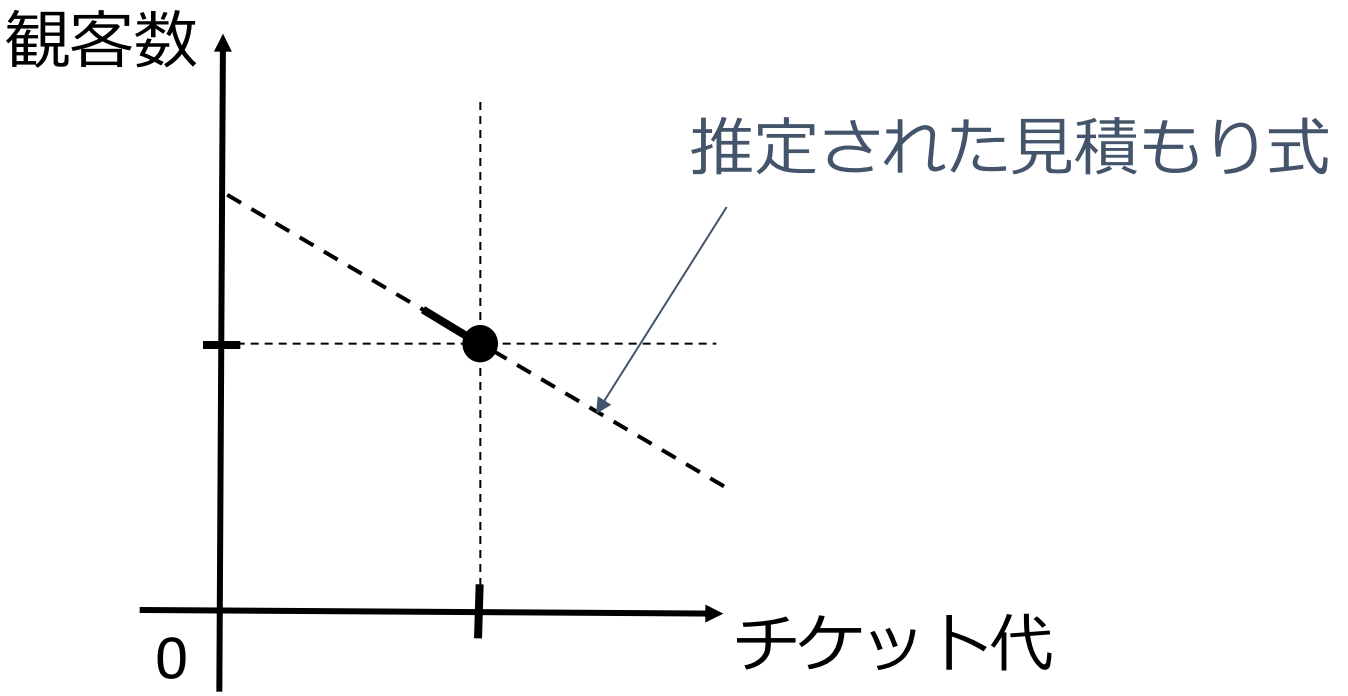


- 支出 ＝ 固定費 ＋ 観客数 × 費用

例） 固定費： ＄１８０

費用： 観客１人あたり＄０．０４

観客数の見積もり式



- チケット代と観客数には関係がある

  例） チケット代：＄５ のとき，観客数は１２０人だった

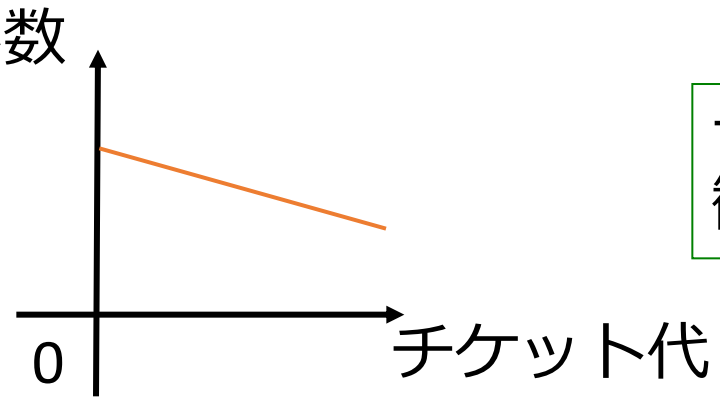
  チケット代：＄０．１ 値下げすると１５人増えた

  ⇒ 観客数 ＝ －（１５／０．１）×（チケット代－＄５） ＋ １２０ と見積もる

利益の計算

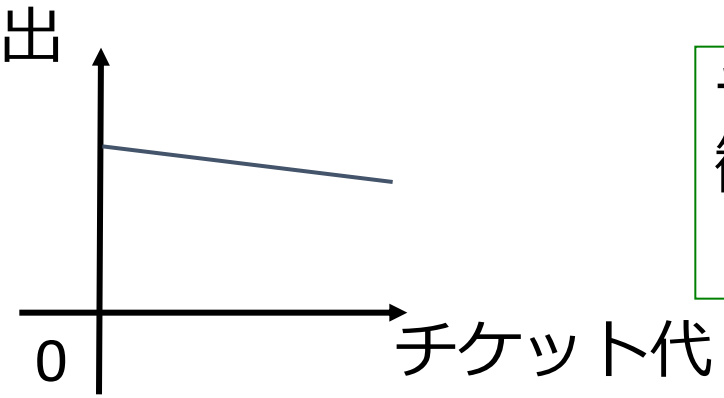
- あなたが「劇場」の所有者であるとする
  - チケット代は，あなたが自由に決める
  - チケット代から，収入，支出，利益などを見積もりたい

観客数
のグラフ



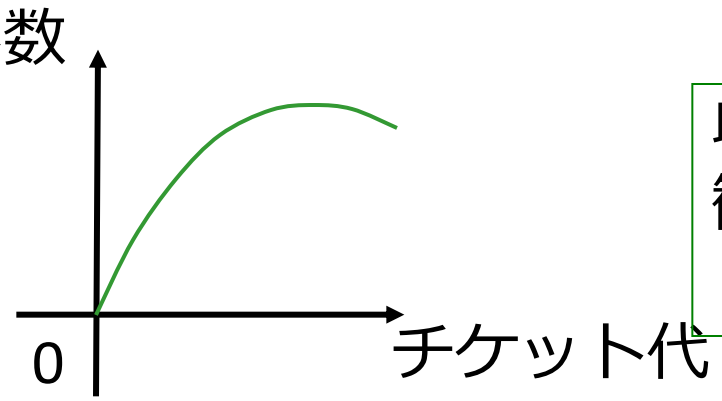
チケット代を上げると，
観客数が減る

支出
のグラフ



チケット代を上げると，
観客数が減り，結果
として支出も減る

収入
のグラフ



収入は，
観客数×チケット代
（これは２次曲線）

「例題７．利益の計算」の手順 (1/4)

1. 次を「定義用ウインドウ」で，実行しなさい
   - 文法的な「間違い」が無いことを確認するため，入力した後 Execute ボタンを押す

```
;; profit : number -> number
;; to compute the profit as the difference between
;; revenue and costs at some given ticket-price
(define (profit ticket-price) ... )
;; revenue:number number → number
;; to compute the revenue, given ticket-price
(define (revenue ticket-price) ... )
;; cost : number -> number
;; to compute the cost, given ticket-price
(define (cost ticket-price) ... )
;; attendees:number → number
;; to compute the number of attendees,
;; given ticket-price
(define (attendees ticket-price) ... )
```

☆　次ページに進んでください

「例題７．利益の計算」の手順 (2/4)

2. 「定義用ウインドウ」で，profit 関数の中身を書く．
   - 文法的な「間違い」が無いことを確認するため，入力した後に，Execute ボタンを押す

```
(define (profit ticket-price)
  (- (revenue ticket-price)
    (cost ticket-price)))
```

3. 「定義用ウインドウ」で，revenue 関数の中身を書く．
   - 文法的な「間違い」が無いことを確認するため，入力した後に，Execute ボタンを押す

```
(define (revenue ticket-price)
  (* (attendees ticket-price) ticket-price))
```

☆　次ページに進んでください

「例題７．利益の計算」の手順 (3/4)

4. 「定義用ウインドウ」で，cost 関数の中身を書く．
   - 文法的な「間違い」が無いことを確認するため，入力した後に，Execute ボタンを押す

```
(define (cost ticket-price)
  (+ 180
    (* .04 (attendees ticket-price))))
```

5. 「定義用ウインドウ」で，attendees 関数の中身を書く．
   - 文法的な「間違い」が無いことを確認するため，入力した後に，Execute ボタンを押す

```
(define (attendees ticket-price)
  (+ 120
    (* (/ 15 .10) (- 5.00 ticket-price))))
```

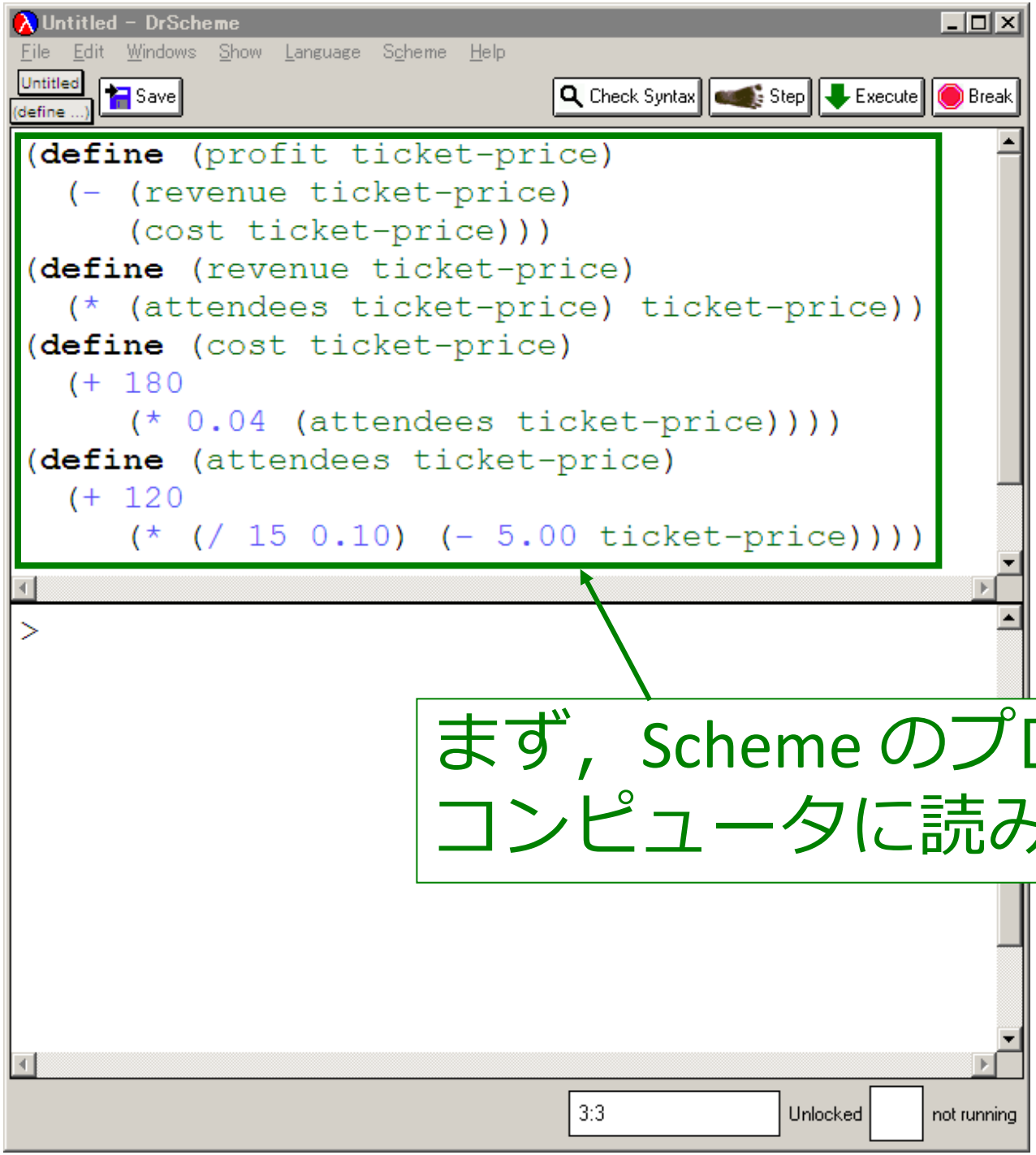
☆　次ページに進んでください

「例題７．利益の計算」の手順 (4/4)

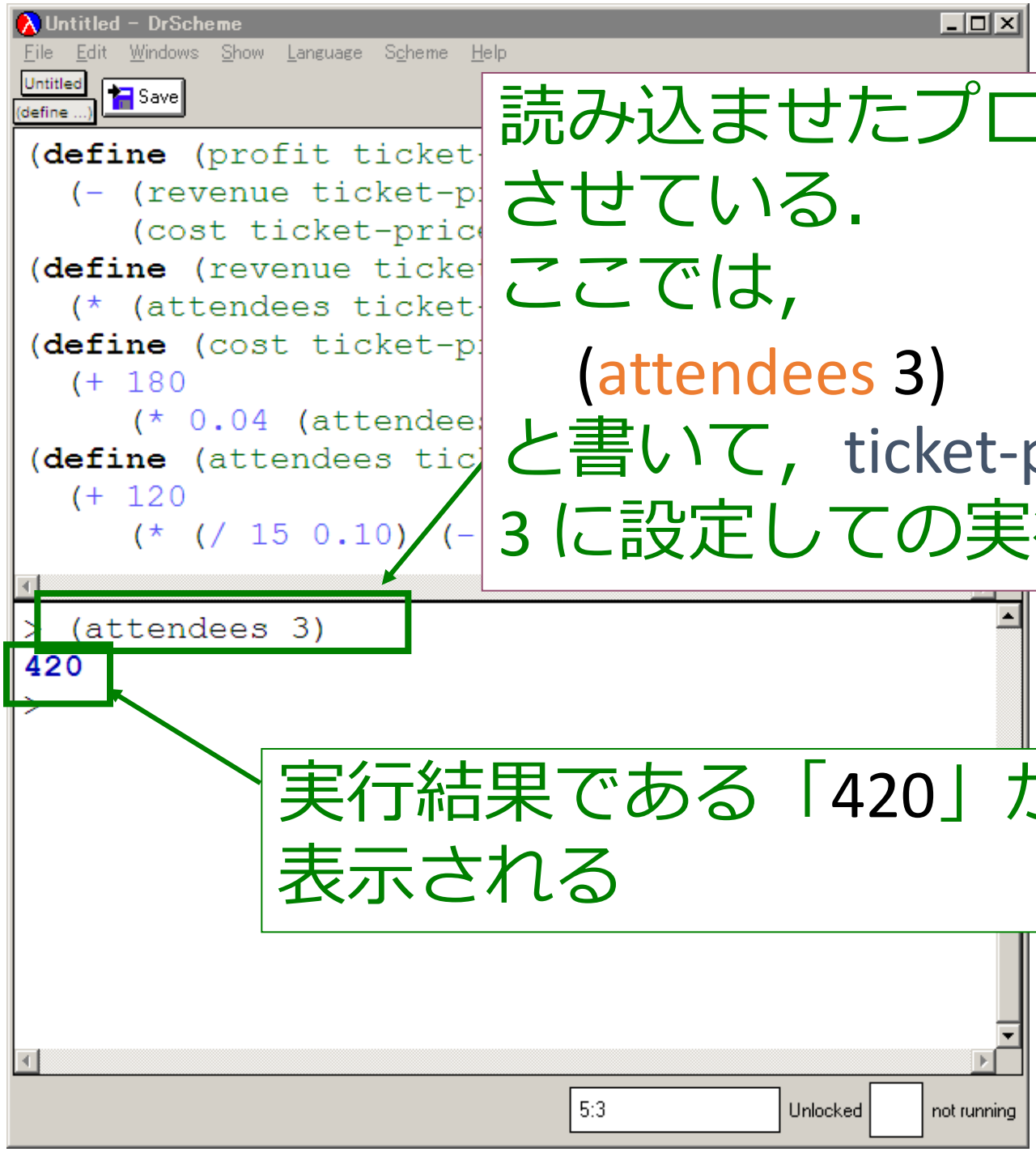
6. その後，次を「実行用ウインドウ」で実行しなさい
   - それぞれの実行結果が，予想通りであることを確認しながら行うこと

```
(attendees 3)
(cost 3)
(revenue 3)
(profit 3)
```

☆　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

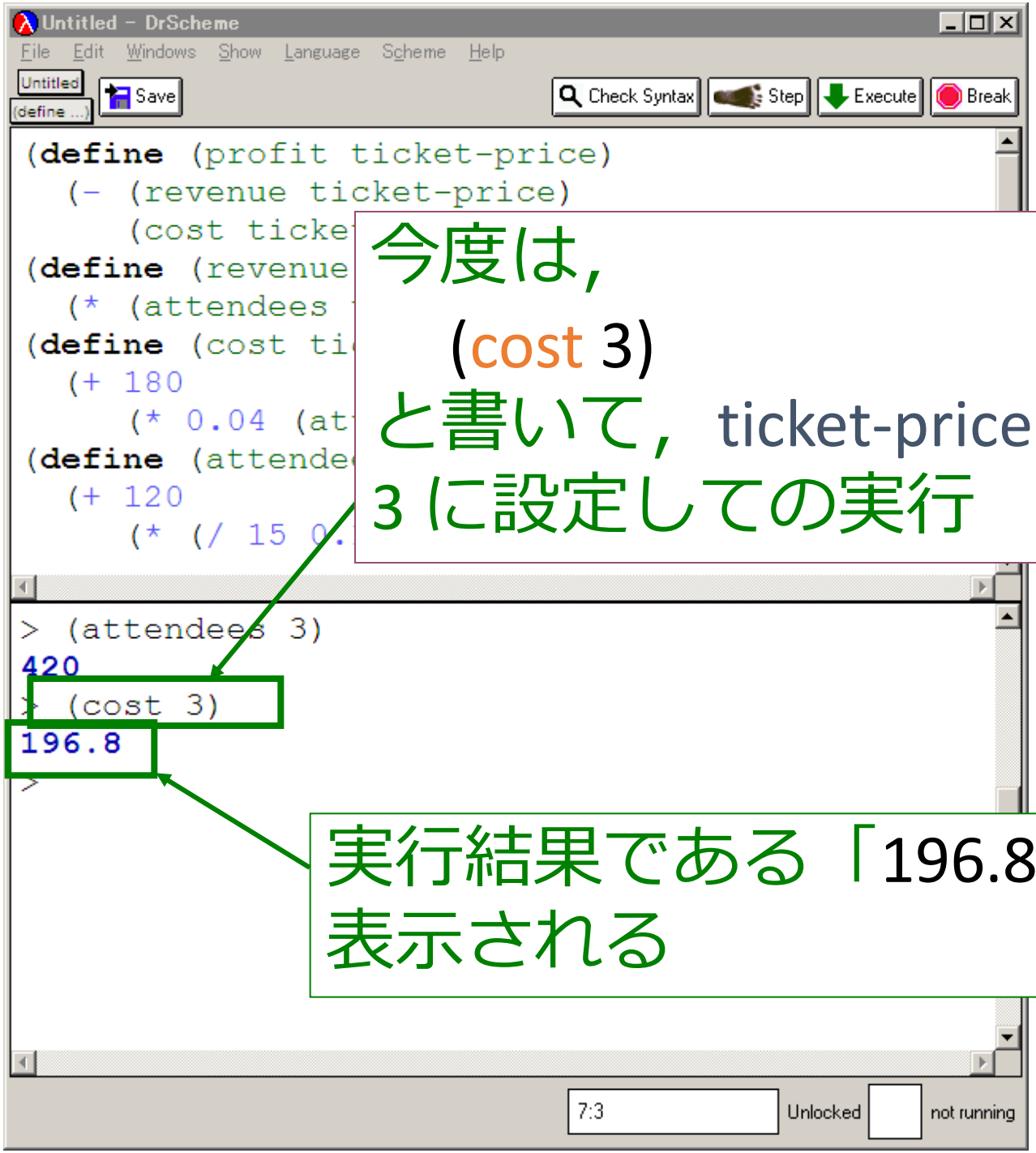


読み込ませたプログラムを実行させている.
ここでは,

(attendees 3)

と書いて，ticket-price の値を3 に設定しての実行

実行結果である「420」が表示される

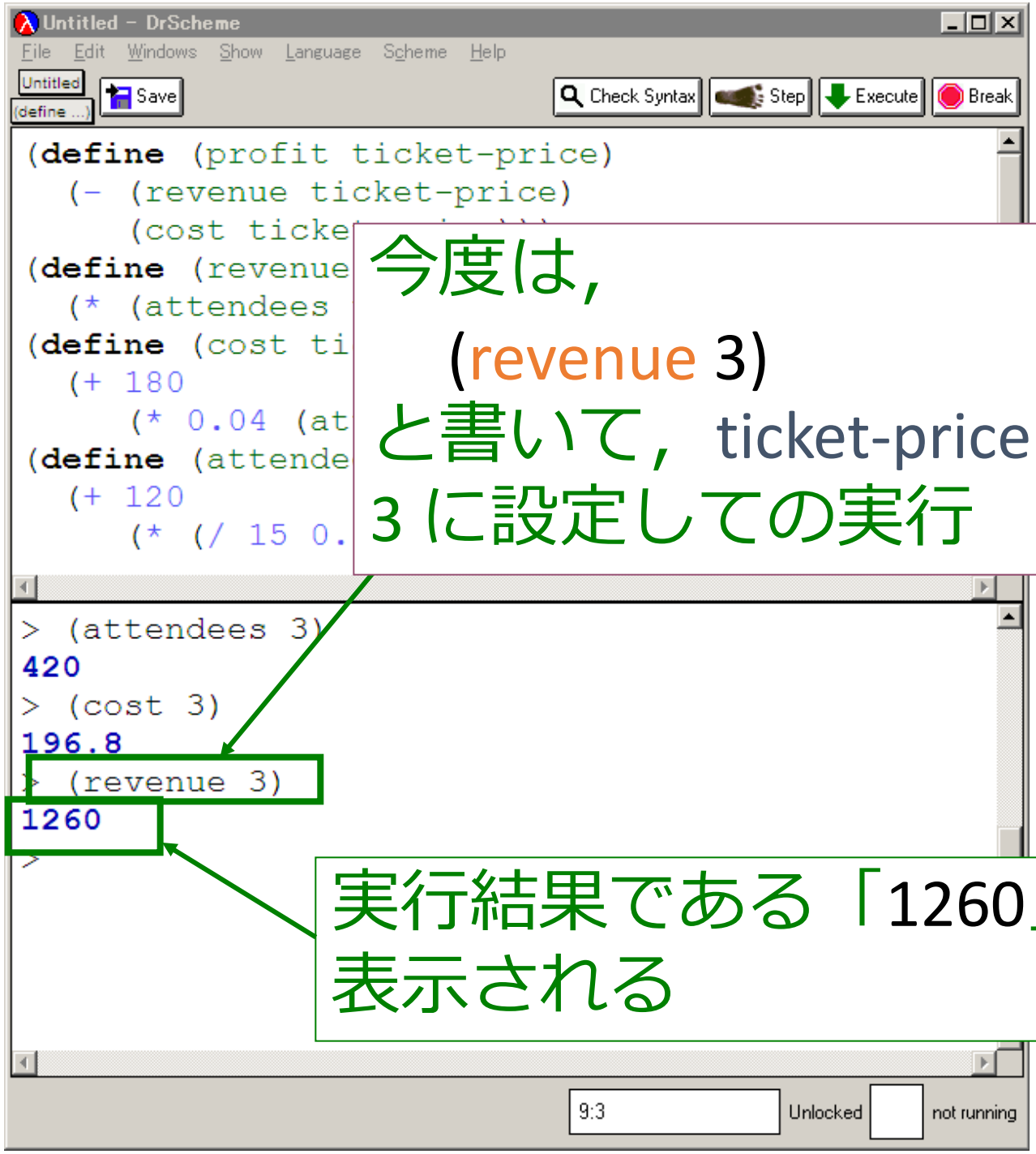


今度は，

(cost 3)

と書いて，ticket-price の値を 3 に設定しての実行

実行結果である「196.8」が表示される

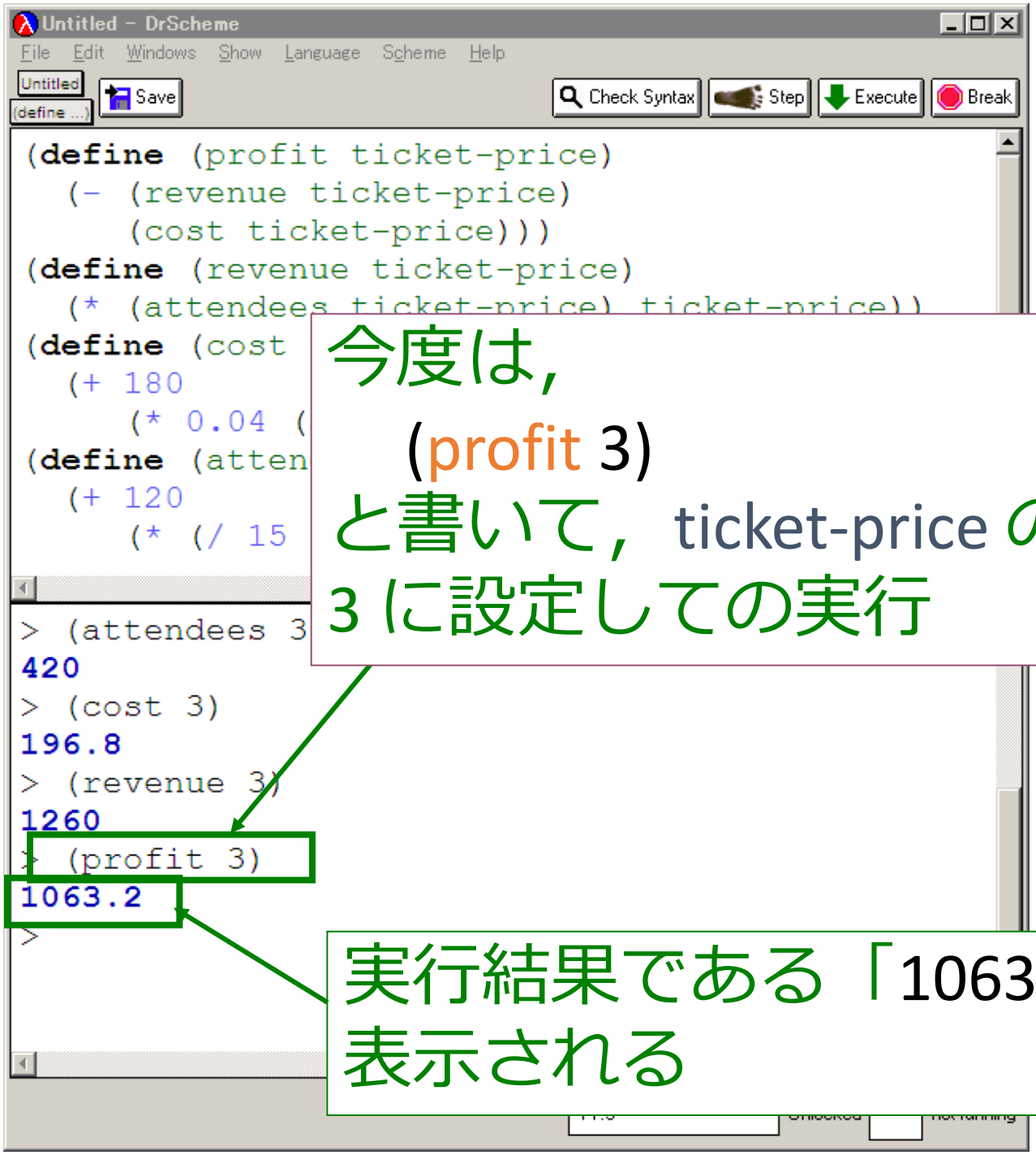


今度は，

(revenue 3)

と書いて，ticket-price の値を3 に設定しての実行

実行結果である「1260」が表示される



今度は,

(profit 3)

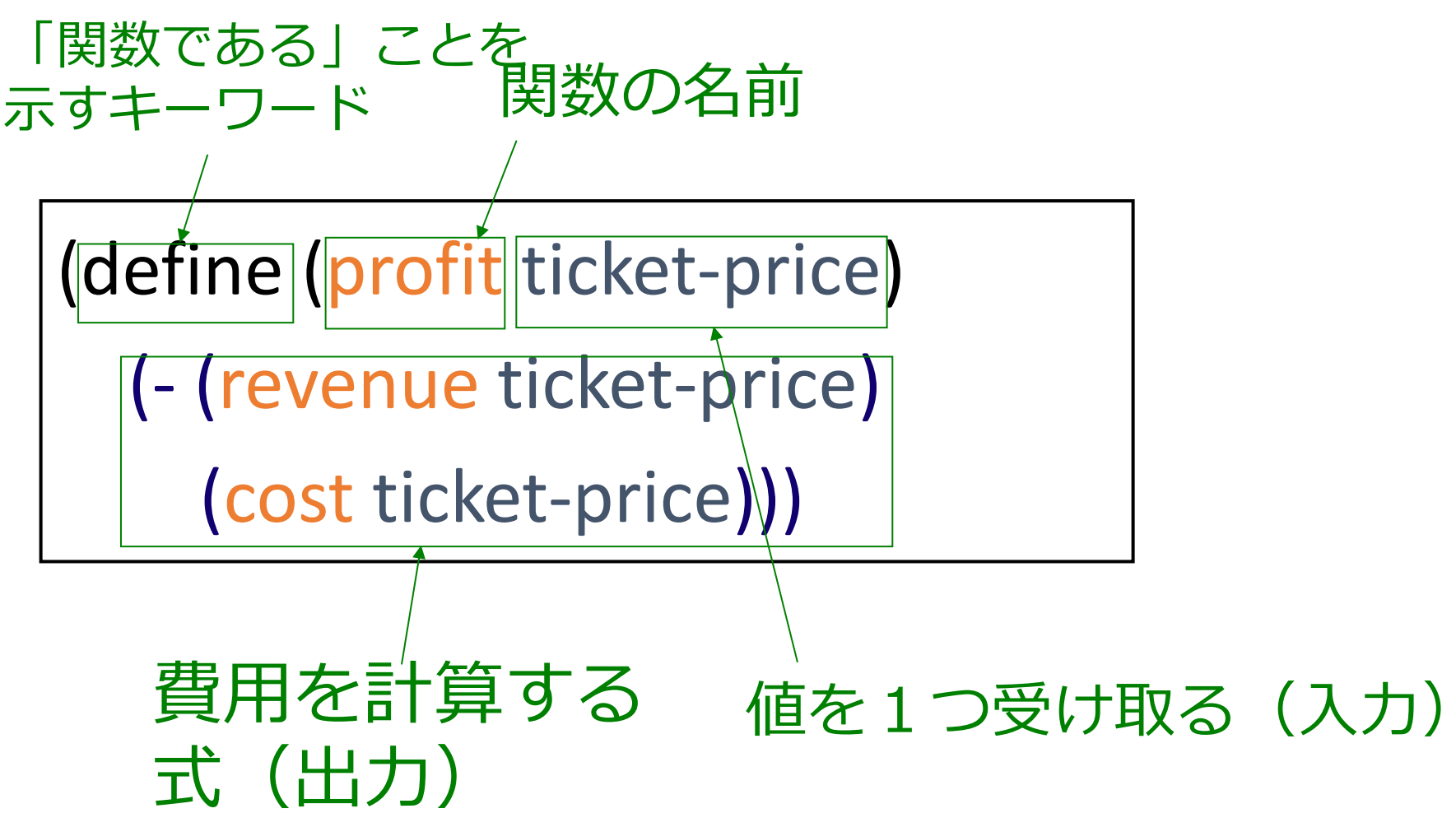
と書いて，ticket-price の値を 3 に設定しての実行

実行結果である「1063.2」が表示される

入力と出力

入力　　出力

profit 関数


「関数である」ことを
示すキーワード
関数の名前
(define (profit ticket-price)
(- (revenue ticket-price)
(cost ticket-price)))
費用を計算する
式（出力）
値を１つ受け取る（入力）

```
(define (profit ticket-price)
    (- (revenue ticket-price)
       (cost ticket-price)))
(define (revenue ticket-price)
    (* (attendees ticket-price) ticket-price))
(define (cost ticket-price)
    (+ 180
       (* 0.04 (attendees ticket-price))))
(define (attendees ticket-price)
    (+ 120
       (* (/ 15 0.10) (- 5.00 ticket-price))))
```

profit 関数

revenue 関数

cost 関数

attendees 関数

関数の関係

profit 関数

```
(define (profit ticket-price)
  (- (revenue ticket-price)
     (cost ticket-price)))
```

revenue, cost 関数を使っている

revenue 関数

```
(define (revenue ticket-price)
  (* (attendees ticket-price) ticket-price))
```

attendees 関数を使っている

cost 関数

```
(define (cost ticket-price)
  (+ 180
     (* 0.04 (attendees ticket-price))))
```

attendees 関数を使っている

attendees 関数

```
(define (attendees ticket-price)
  (+ 120
     (* (/ 15 0.10) (- 5.00 ticket-price))))
```

例題８．ステップ実行

- 関数 profit （例題７）について，実行結果に至る過程を見る
  - (profit 3) から 1063.2 に至る過程を見る
  - DrScheme の stepper を使用する

```
(profit 3)
= (- (revenue 3) (cost 3))
= (- (* (attendees 3) 3) (cost 3))
= (- (* (+ 120 (* (/ 15 0.10) (- 5.00 3))) 3) (cost 3))
= (- (* (+ 120 (* 150 (- 5.00 3))) 3) (cost 3))
= (- (* (+ 120 (* 150 2)) 3) (cost 3))
= (- (* (+ 120 300) 3) (cost 3))
= (- (* 420 3) (cost 3))
= (- 1260 (cost 3))
= (- 1260 (+ 180 (* 0.04 (attendees 3))))
= (- 1260 (+ 180 (* 0.04 (+ 120 (* (/ 15 0.10) (- 5.00 3))))))
= (- 1260 (+ 180 (* 0.04 (+ 120 (* 150 (- 5.00 3))))))
= (- 1260 (+ 180 (* 0.04 (+ 120 (* 150 2))))))
= (- 1260 (+ 180 (* 0.04 (+ 120 300))))
= (- 1260 (+ 180 (* 0.04 420)))
= (- 1260 (+ 180 16.8))
= (- 1260 196.8)
=1063.2
```

「例題３．ステップ実行」の手順

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

```
;; profit: number -> number
;; to compute the profit as the difference between
;; revenue and costs at some given ticket-price
(define (cost ticket-price)
  (+ 180
    (* .04 (attendees ticket-price))))
;; revenue:number number → number
;; to compute the revenue, given ticket-price
(define (revenue ticket-price)
  (* (attendees ticket-price) ticket-price))
;; cost : number -> number
;; to compute the cost, given ticket-price
(define (cost ticket-price)
  (+ 180
    (* .04 (attendees ticket-price))))
;; attendees:number → number
;; to compute the number of attendees,
;; given ticket-price
(define (attendees ticket-price)
  (+ 120
    (* (/ 15 .10) (- 5.00 ticket-price))))
(profit 3)
```

例題７と同じ

ステップ実行したいので，入力済みのプログラムは，消さずに残しておく

例題７に１行書き加える

2. DrScheme を使って，ステップ実行の様子を確認しなさい　（Step ボタン，Next ボタンを使用）

(profit 3) から 1063.2 が得られる過程

(profit 3) 最初の式

```
= (- (revenue 3) (cost 3))
= (- (* (attendees 3) 3) (cost 3))
= (- (* (+ 120 (* (/ 15 0.10) (- 5.00 3))) 3) (cost 3))
= (- (* (+ 120 (* 150 (- 5.00 3))) 3) (cost 3))
= (- (* (+ 120 (* 150 2)) 3) (cost 3))
= (- (* (+ 120 300) 3) (cost 3))
= (- (* 420 3) (cost 3))
= (- 1260 (cost 3))
= (- 1260 (+ 180 (* 0.04 (attendees 3))))
= (- 1260 (+ 180 (* 0.04 (+ 120 (* (/ 15 0.10) (- 5.00 3))))))
= (- 1260 (+ 180 (* 0.04 (+ 120 (* 150 (- 5.00 3))))))
= (- 1260 (+ 180 (* 0.04 (+ 120 (* 150 2)))))
= (- 1260 (+ 180 (* 0.04 (+ 120 300))))
= (- 1260 (+ 180 (* 0.04 420)))
= (- 1260 (+ 180 16.8))
= (- 1260 196.8)
=1063.2
```

コンピュータ内部での計算

実行結果

(profit 3) から 1063.2 が得られる過程

(profit 3) 最初の式
= (- (revenue 3) (cost 3))
= (- (* (attendees 3) 3) (cost 3))
= (- (* (+ 120 (* (/ 15 0.10) (- 5.00 3))) 3) (cost 3))

これは,

```
(define (profit ticket-price)
  (- (revenue ticket-price)
     (cost ticket-price)))
```

の ticket-price を 3 で置き換えたもの

= (- 1260 (+ 180 (* 0.04 (+ 120 (* 150 (- 5.00 3))))))
= (- 1260 (+ 180 (* 0.04 (+ 120 (* 150 2))))))
= (- 1260 (+ 180 (* 0.04 (+ 120 300))))
= (- 1260 (+ 180 (* 0.04 420)))
= (- 1260 (+ 180 16.8))
= (- 1260 196.8)
=1063.2 実行結果

3-3 課題

課題１

- 関数 profit （授業の例題７）についての問題
  - 関数 profit を実行し，チケット代が 3, 4, 5の時の実行結果を報告しなさい

課題２

- 関数 profit （授業の例題７）についての問題
  - 固定費が０になるように例題７のプログラムを変更しなさい
  - その後，関数 profit を実行し，チケット代が 3, 4, 5の時の実行結果を報告しなさい