

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

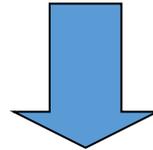
3-2 パソコン演習

3-3 課題

- 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

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: $(\text{area-of-disk } 5) = (* 3.14 (* 5 5)) = \dots$

$(+ (\text{sqr } 2) (\text{sqr } 4)) = (+ (* 2 2) (\text{sqr } 4)) = \dots$

- Scheme 処理系と実行モデルを理解する
- 変数と関数の違い

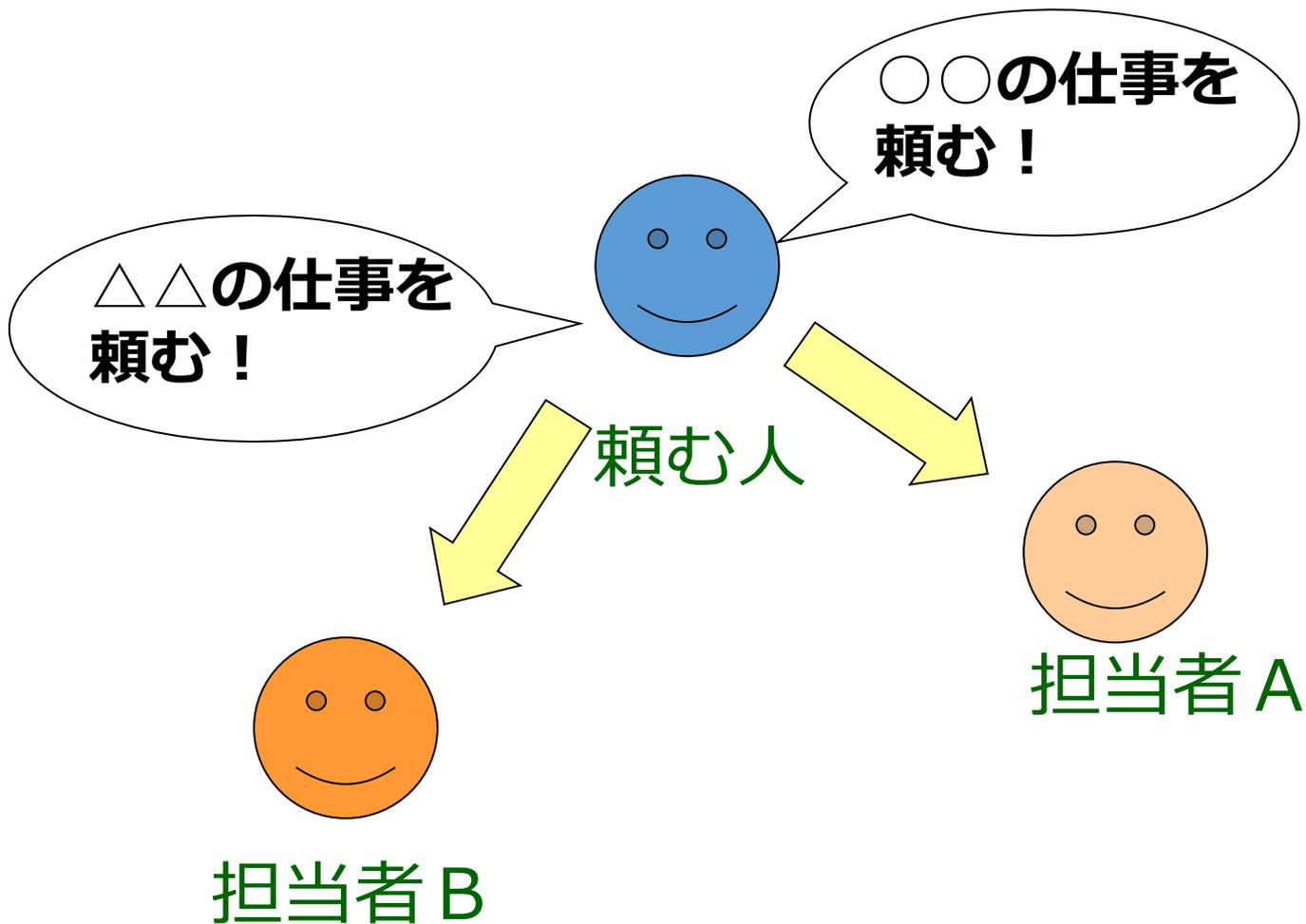
Scheme の関数の振る舞い



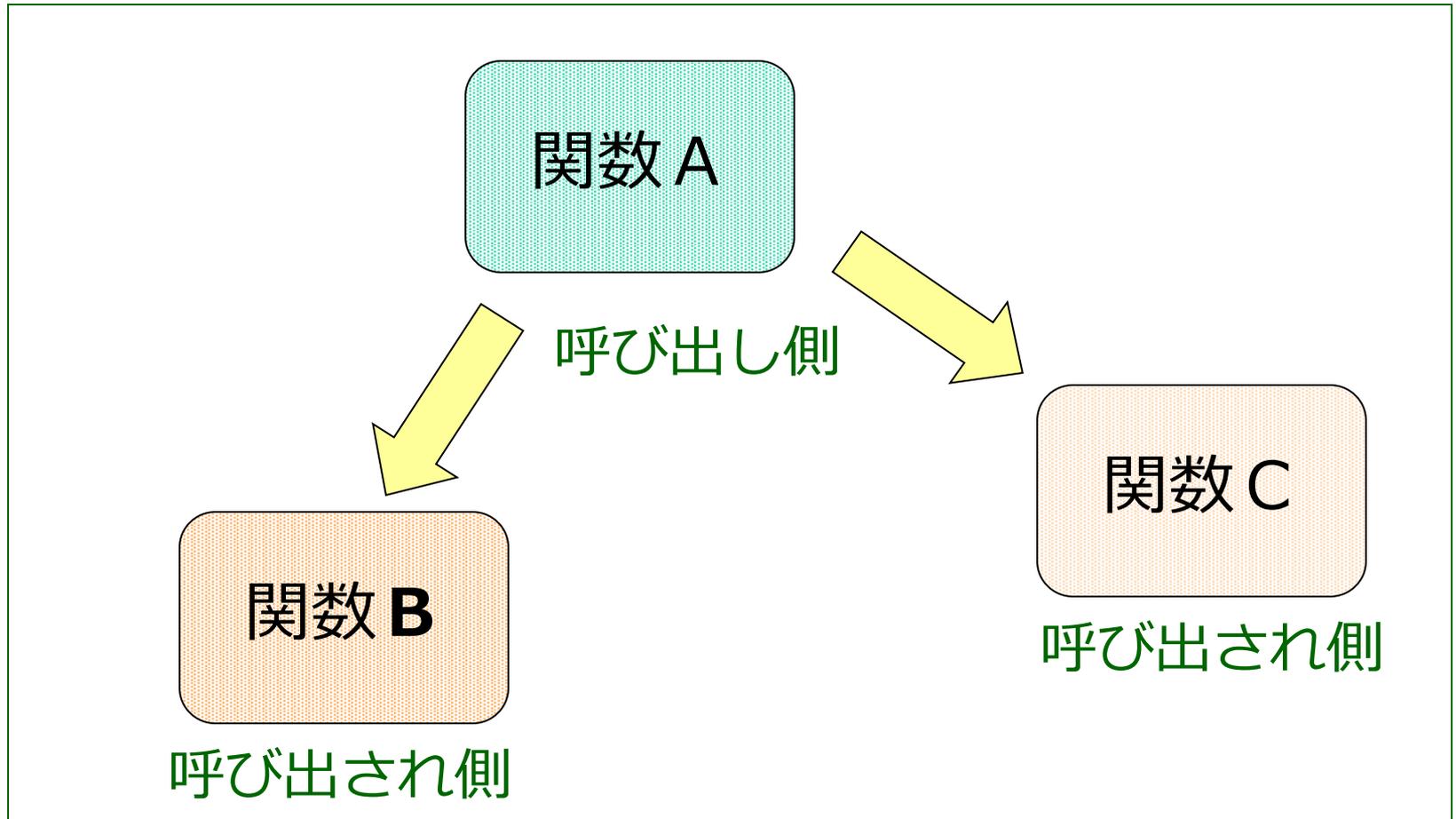
関数 (procedure arg₁ ... arg_n) に対して

- procedure の値を取得する
- arg₁ から arg_n の値を取得する
- procedure の値に arg₁ 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



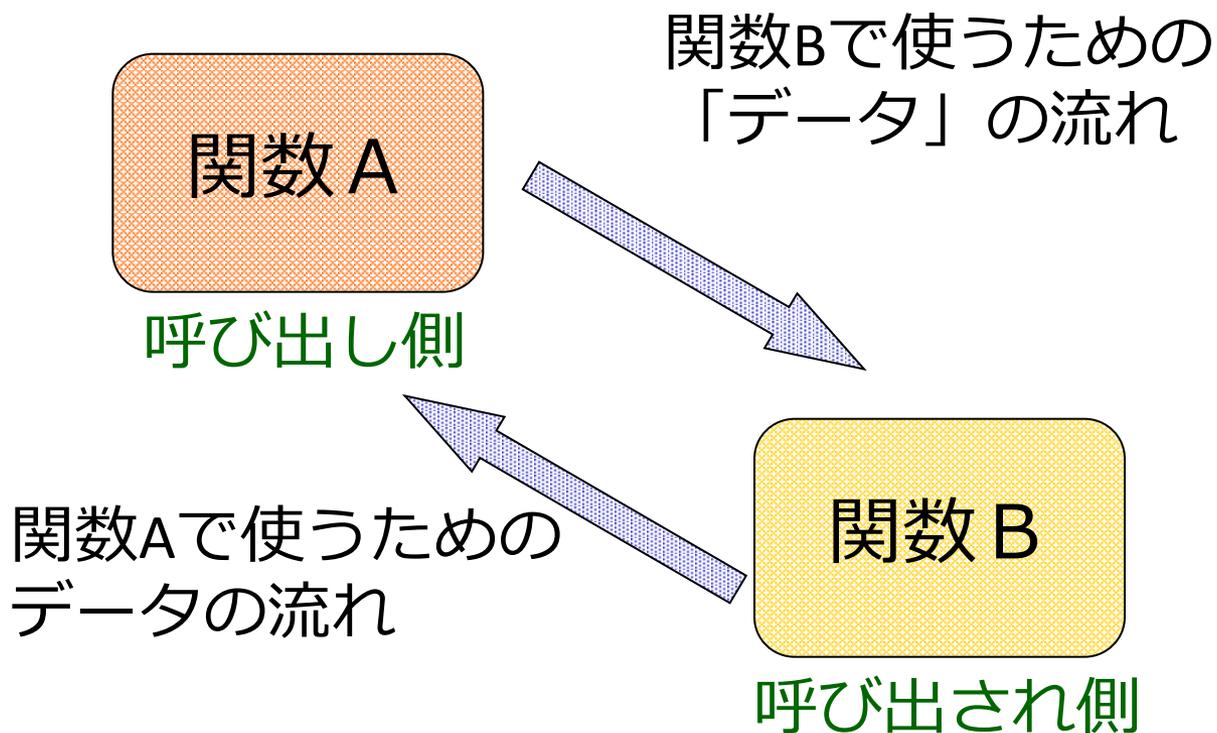
- Scheme のプログラムは, 一般に, 複数の関数の集まり

```
(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

- 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
- 「定義用ウィンドウ」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



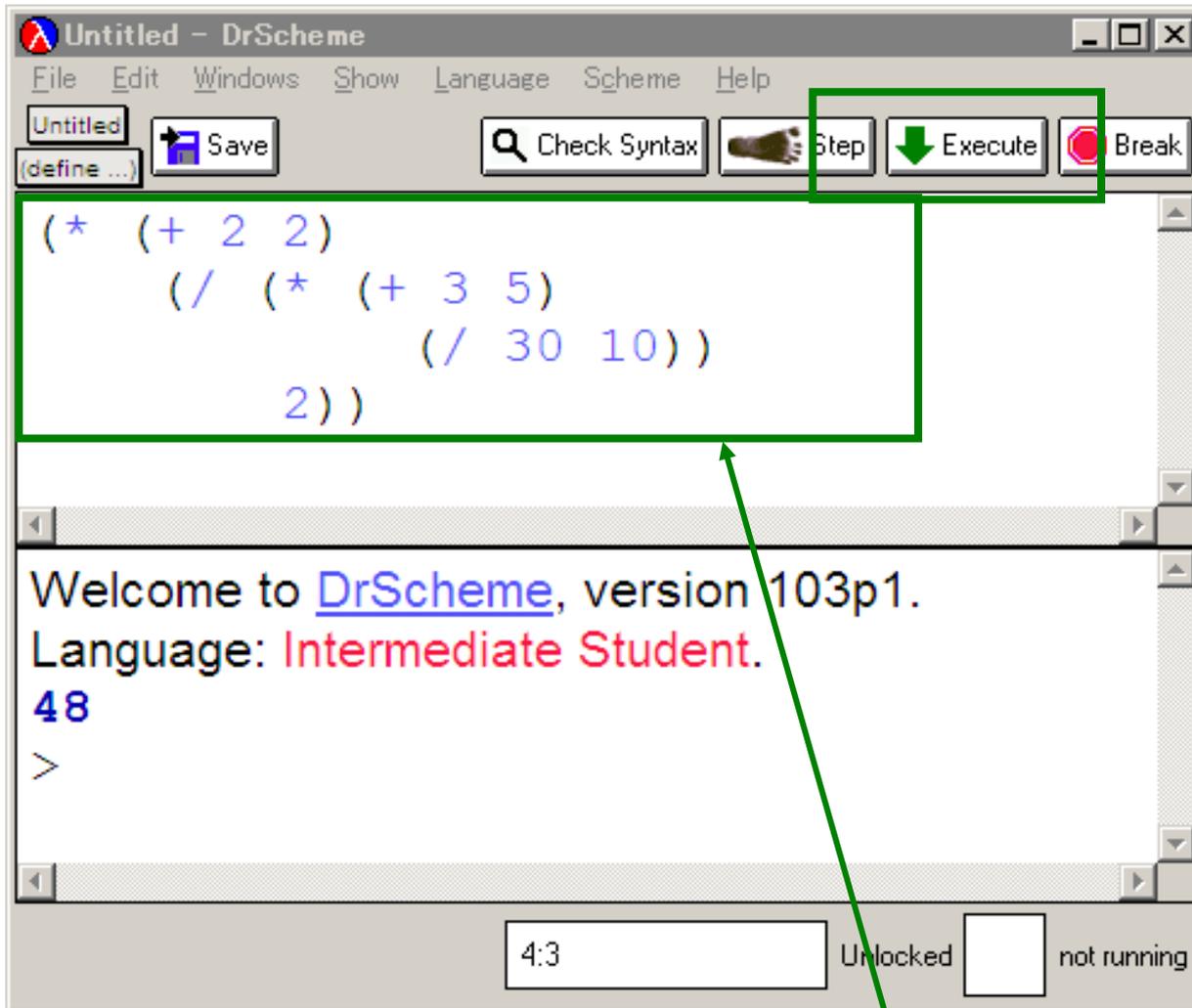
1. 次を「定義用ウィンドウ」で，実行しなさい
 - Intermediate Student で実行すること
 - 入力した後に，Execute ボタンを押す

```
(* (+ 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



定義用ウィンドウに入力して、
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 >>

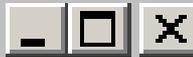
```
( * 4 ( / ( * 8 3 ) 2 ) )  
4  
( / ( * 8 ( / 30 10 ) )  
2 ) )
```

→

「(/ 30 10)」は「3」で置き換わる



Stepper



File Edit Windows Help

Home

<< Previous

Next >>

(* 4 (/ (* 8 3) 2)) → (* 4 (/ 24 2))

「(* 8 3)」は「24」で
置き換わる



Stepper



File Edit Windows Help

Home << Previous Next >>

(* 4 (/ 24 2)) → (* 4 12)

「(/ 24 2)」は「12」で
置き換わる



Stepper



File Edit Windows Help

Home

<< Previous

Next >>

(* 4 12)

→ 48

「(* 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

$$= (* 4 (/ (* (+ 3 5) (/ 30 10)) 2)) (+ 2 2) \rightarrow 4$$

$$= (* 4 (/ (* 8 (/ 30 10)) 2)) (+ 3 5) \rightarrow 8$$

$$= (* 4 (/ (* 8 3) 2)) (/ 30 10) \rightarrow 3$$

$$= (* 4 (/ 24 2)) (* 8 3) \rightarrow 2$$

$$= (* 4 12) (/ 24 2) \rightarrow 12 \quad \text{コンピュータ内部での計算}$$

$$= \boxed{48} \quad \text{実行結果}$$

実行結果「48」が得られる過程



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

最初の式

$= (\square (/ (* (+ 3 5) (/ 30 10)) 2)) (+ 2 2) \rightarrow 4$

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

$= \square (/ 24 2) \rightarrow 12$
 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「定義用ウィンドウ」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

例題 1 と同じ

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

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

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

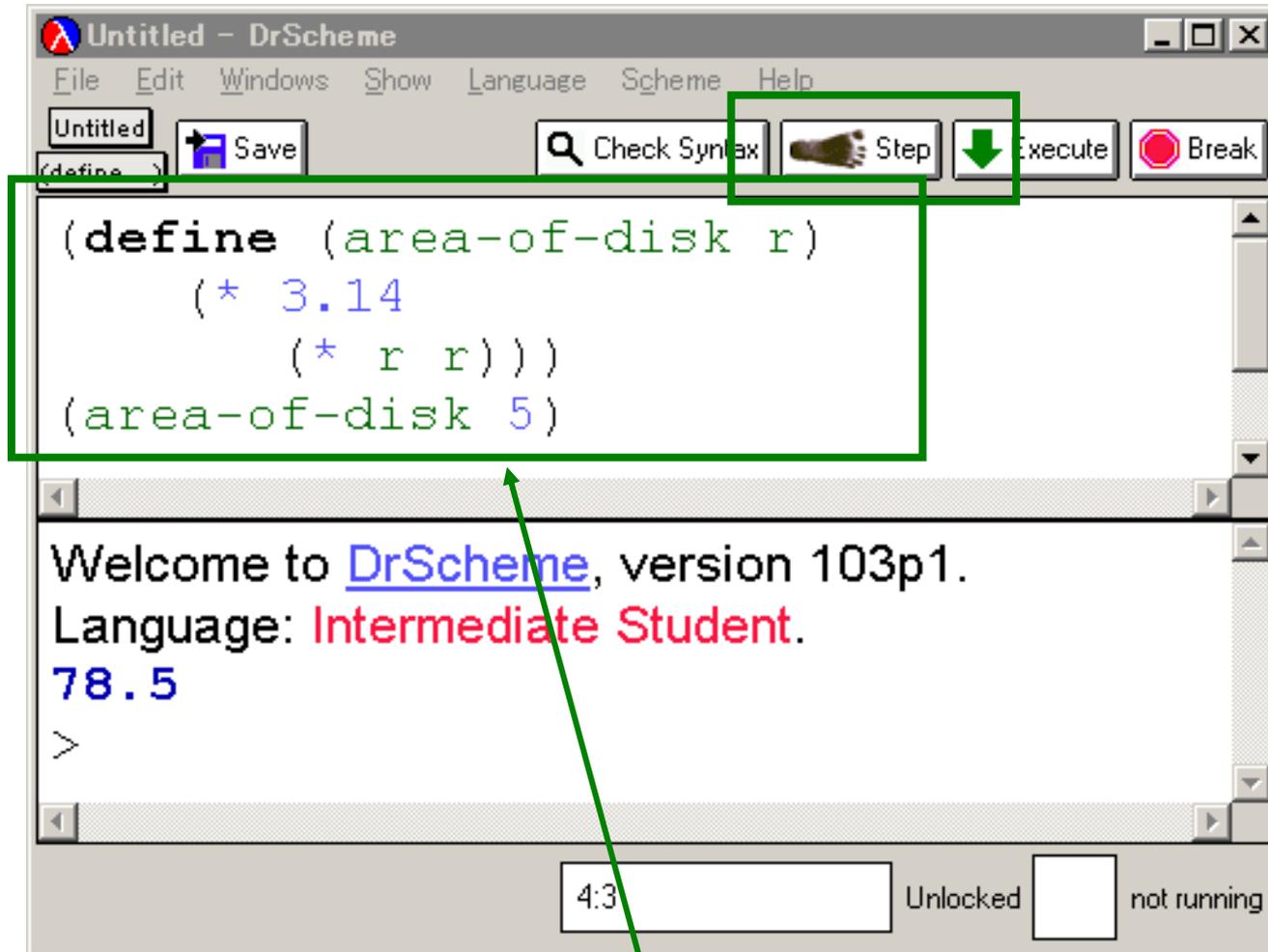
例題 1 に
1 行書き加える

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



```
Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled Save Check Syntax Step Execute Break
(define ...)
(define (area-of-disk r)
  (* 3.14
    (* r r)))
(area-of-disk 5)

Welcome to DrScheme, version 103p1.
Language: Intermediate Student.
78.5
>

4:3 Unlocked not running
```

定義用ウィンドウに入力して、
Execute ボタンを押した後、Step ボタンを押すと

```
(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

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）

```
(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)

最初の式

最初の式

(area-of-disk 5)
から始まって、計算を繰り返して、実行結果
78.5
に至る

= 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)

= (* 3.14 (* 5 5))

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「定義用ウィンドウ」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「実行用ウィンドウ」で実行しなさい

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

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

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

>

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

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

```
> (sum-of-squares 2 4)
```

20

実行結果である「20」が
表示される

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

今回は、
(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

```
> (sum-of-squares 2 4)
20
> (sum-of-squares 20 30)
1300
```

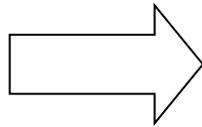
実行結果である「1300」が
表示される

入力と出力

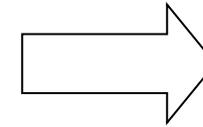


xの値 :

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)

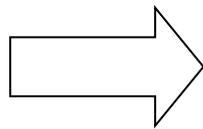
値を1つ受け取る (入力)
(xのことを「パラメータ」という)

入力と出力

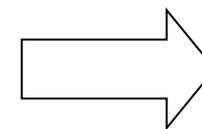


x, y の値

2, 4



sum-of-squares



20

入力は
2つの数値

出力は
1つの数値

sum-of-squares 関数



「関数である」ことを
示すキーワード 関数の名前

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

1つの関数

「 $x^2 + y^2$ 」
を計算 (出力)

値を2つ受け取る (入力)

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)
```

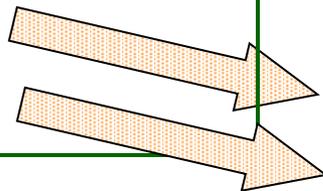
```
  (+ (sqr x) (sqr y)))
```

sqr
の部分

sum-of-squares
の部分

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「定義用ウィンドウ」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)  
  (sqr 30))
```

→

```
(+  
  (* 20 20)  
  (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)))
```

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

$$= (+ (\text{sqr } 20) (\text{sqr } 30)) \quad \boxed{(+ (\text{sqr } x) (\text{sqr } y))}$$

に $x=2, y=4$ が代入される

$$= (+ (* 20 20) (\text{sqr } 30)) \quad \boxed{(* x x)}$$

に $x=2$ が代入される

$$= (+ 400 (\text{sqr } 30)) \quad (* 2 2) \rightarrow 4$$

$$= (+ 400 (* 30 30)) \quad \boxed{(* x x)}$$

に $x=4$ が代入される

$$= (+ 400 900) \quad (* 4 4) \rightarrow 16 \quad \text{コンピュータ内部での計算}$$

= 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))

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

= (+ 400 900)

= 1300

例題 5 . リングの面積

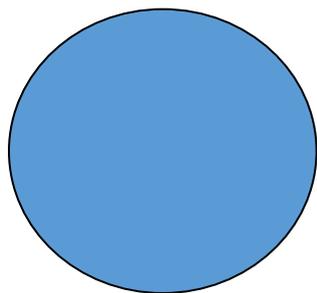


- 外径 `outer`, 内径 `inner` からリングの面積を求めるプログラム `area-of-ring` を作り, 実行する
 - 円の面積を求める関数 `area-of-disk` と組み合わせる

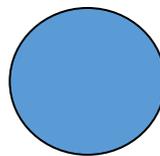
真ん中に穴のあいた
円の面積と考える

リングの面積

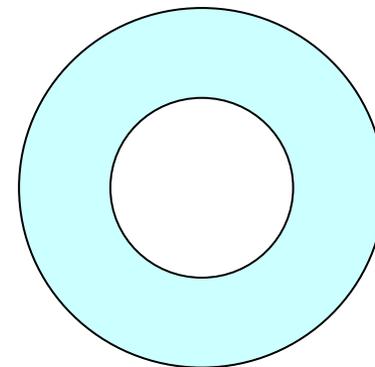
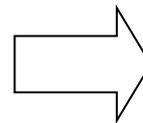
- 求める面積は、外側の円の面積から、内側の穴の円の面積を引いたもの



外側の円



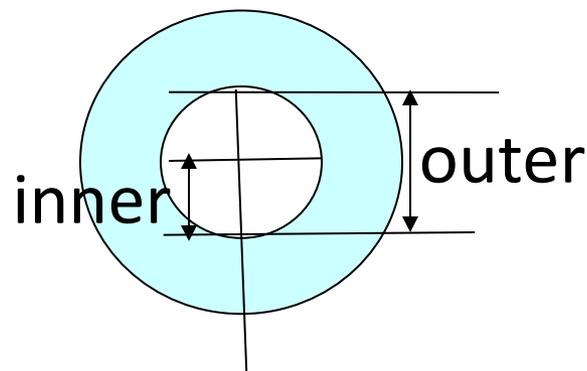
内側の円



リングの面積

外径 : outer

内径 : inner



リングの面積

$$= \underbrace{\text{外側の円の面積}}_{\text{半径 outer の円}} - \underbrace{\text{内側の円の面積}}_{\text{半径 inner の円}}$$

「例題 5. リングの面積」の手順

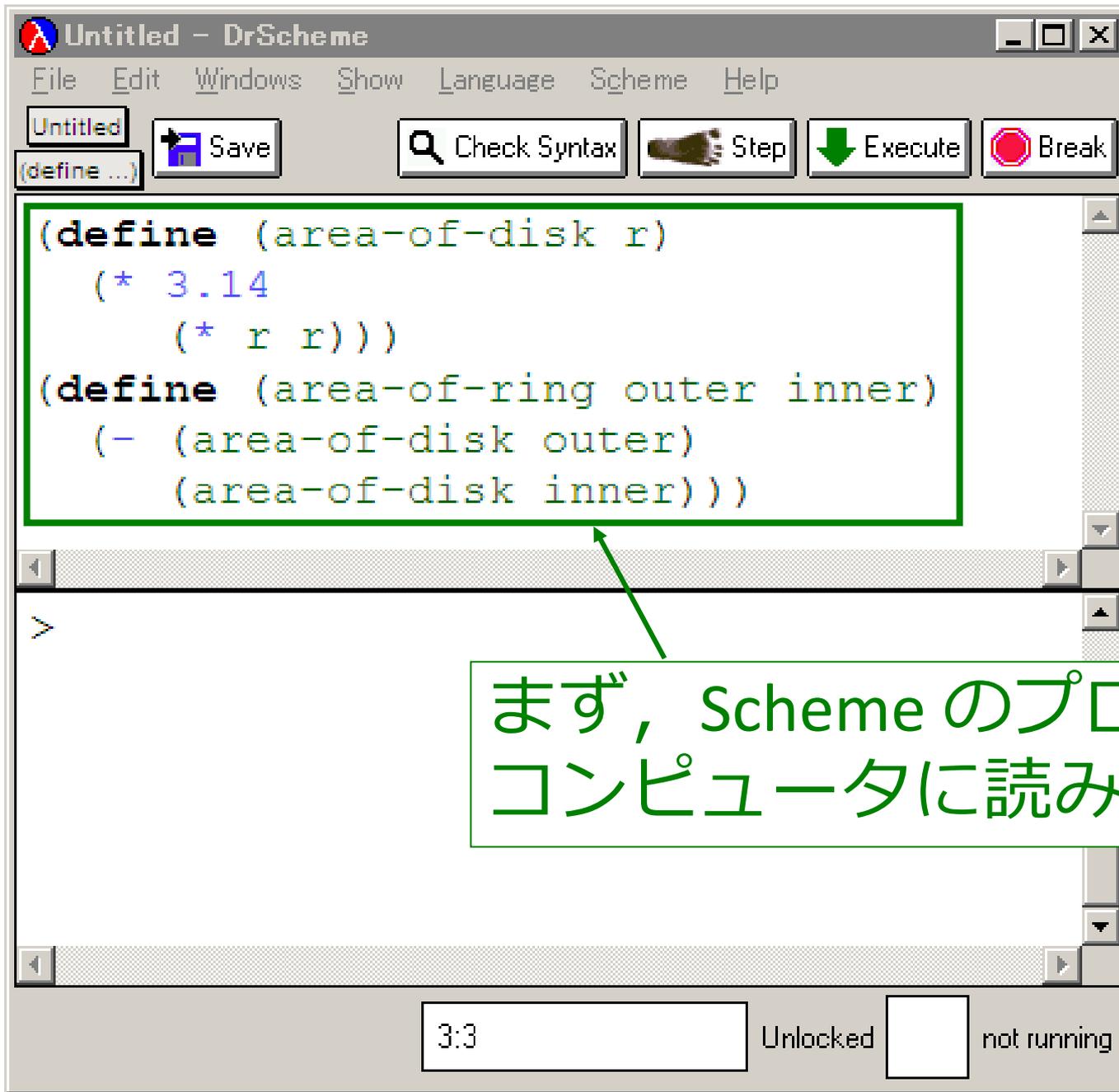
1. 次を「定義用ウィンドウ」で、実行しなさい
 - 入力した後に、Execute ボタンを押す

```
(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「実行用ウィンドウ」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



Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled Save Check Syntax Step Execute Break

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

>

3:3 Unlocked not running

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



The screenshot shows the DrScheme IDE with the following content:

```
Untitled - DrScheme
File Edit Windows Show Language
Untitled Save Check
(define ...)
```

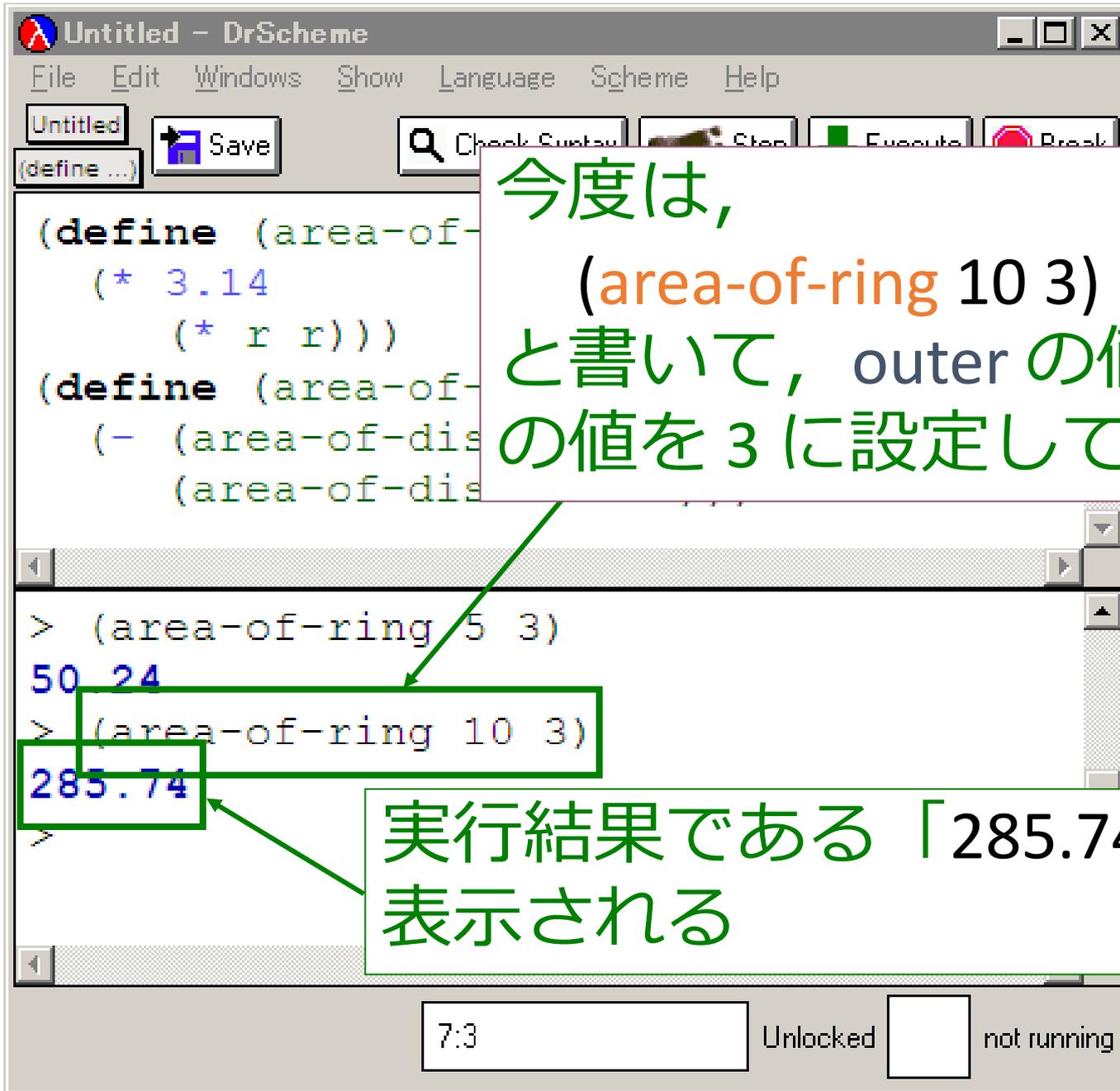
```
(define (area-of-disk r)
  (* 3.14
     (* r r)))
(define (area-of-ring outer inner)
  (- (area-of-disk outer)
     (area-of-disk inner)))
```

```
> (area-of-ring 5 3)
50.24
```

At the bottom of the IDE, there is a control panel with a text box containing "5:3", a button labeled "Unlocked", and a status indicator "not running".

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



The screenshot shows the DrScheme IDE with the following content:

```
Untitled - DrScheme
File Edit Windows Show Language Scheme Help
[Untitled] [Save] [Check Syntax] [Step] [Execute] [Break]
(define ...)
(define (area-of-ring
  (* 3.14
    (* r r)))
(define (area-of-ring
  (- (area-of-disk
    (area-of-disk
      5 3)
      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

> `(area-of-ring 5 3)`
50.24

> `(area-of-ring 10 3)`
285.74

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

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



「関数である」ことを
示すキーワード 関数の名前

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

} 1つの関数

「外側の円の面積
- 内側の円の面積」
を計算 (出力)

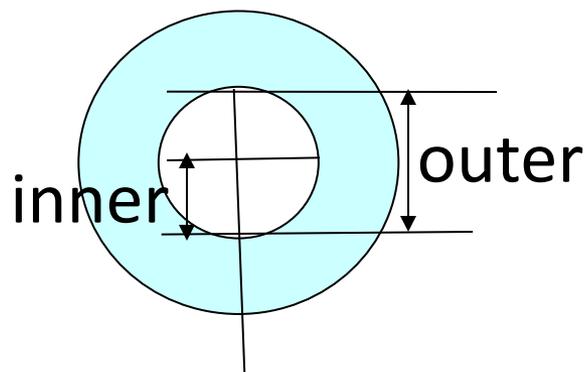
値を2つ受け取る (入力)

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)))
```

area-of-disk
の部分

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

area-of-ring
の部分

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「定義用ウィンドウ」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



```
(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)
```

Welcome to [DrScheme](#), version 103p1.
Language: **Intermediate Student**.
50.24
>

4:3 Unlocked not running

定義用ウィンドウに入力して、
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



The screenshot shows a window titled "Stepper" with a menu bar (File, Edit, Windows, Help) and navigation buttons (Home, << Previous, Next >>). The main area contains Scheme code. The first part defines two functions: `(define (area-of-disk r) (* 157/50 (* r r)))` and `(define (area-of-ring outer inner) (- (area-of-disk outer) (area-of-disk inner)))`. A horizontal blue line separates this from the next line of code, `(area-of-ring 5 3)`, which is highlighted in green. A red arrow points from this line to a purple-shaded area containing the expanded execution steps: `(-`, `(area-of-disk 5)`, and `(area-of-disk 3))`. A second horizontal blue line is below this. The bottom right corner of the window shows a page number "78".

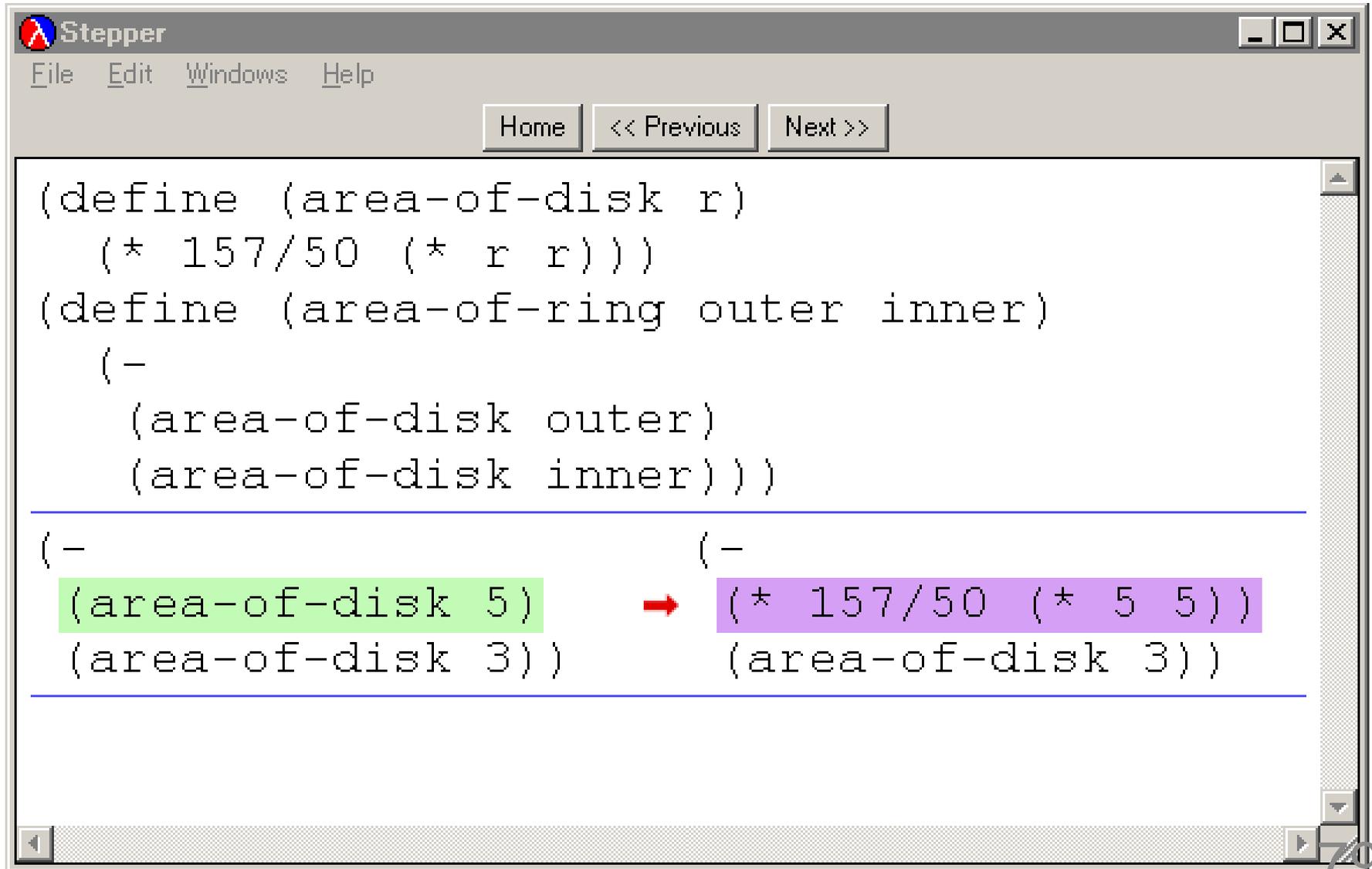
```
(define (area-of-disk r)
  (* 157/50 (* r r)))
(define (area-of-ring outer inner)
  (-
   (area-of-disk outer)
   (area-of-disk inner)))

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

→

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

例題 6. ステップ実行



The screenshot shows a window titled "Stepper" with a menu bar (File, Edit, Windows, Help) and navigation buttons (Home, << Previous, Next >>). The main area contains Scheme code. A red arrow points from the original code to the code after one step of execution.

```
(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



The screenshot shows a window titled "Stepper" with a menu bar (File, Edit, Windows, Help) and navigation buttons (Home, << Previous, Next >>). The main text area contains Scheme code for calculating the area of a disk and a ring. A blue horizontal line separates the code from the execution steps. The first step shows the evaluation of `(* 157/50 25)` resulting in `157/2`. A red arrow points from the first expression to the second, indicating the next step in the execution process.

```
(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

A screenshot of a software application window titled "Stepper". The window has a menu bar with "File", "Edit", "Windows", and "Help". Below the menu bar are three buttons: "Home", "<< Previous", and "Next >>". The main area of the window contains a text editor with the following code:

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

A horizontal blue line separates the code from the execution steps below. The first step shows the code for the `area-of-disk` function being executed with the argument `3`. The second step shows the code for the `area-of-ring` function being executed with arguments `3` and `3`. A red arrow points from the first step to the second. The code in the first step is highlighted in light green, and the code in the second step is highlighted in light purple.

```
(-
 157/2
(area-of-disk 3))
```

```
(-
 157/2
(* 157/50 (* 3 3)))
```

例題 6. ステップ実行

A screenshot of a software application window titled "Stepper". The window has a menu bar with "File", "Edit", "Windows", and "Help". Below the menu bar are three buttons: "Home", "<< Previous", and "Next >>". The main area contains a text editor with code. The code is divided into two sections by horizontal blue lines. The first section shows the definition of two functions: (define (area-of-disk r) (* 157/50 (* r r))) and (define (area-of-ring outer inner) (- (area-of-disk outer) (area-of-disk inner))). The second section shows a comparison of two function calls. On the left, (- 157/2 (* 157/50 (* 3 3))) is shown, with the inner expression (* 3 3) highlighted in light green. A red arrow points to the right, where (- 157/2 (* 157/50 9)) is shown, with the constant 9 highlighted in light purple. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

例題 6. ステップ実行



The screenshot shows a window titled "Stepper" with a menu bar (File, Edit, Windows, Help) and navigation buttons (Home, << Previous, Next >>). The main area contains Scheme code for calculating the area of a disk and a ring. A blue horizontal line separates the code from the execution steps. The first step shows the evaluation of the first function call, with the result $1413/50$ highlighted in purple. The second step shows the evaluation of the second function call, with the result 9 highlighted in green. A red arrow points from the $1413/50$ result to the 9 result, indicating the flow of execution.

```
(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  → 9
```

例題 6. ステップ実行



The screenshot shows a window titled "Stepper" with a menu bar (File, Edit, Windows, Help) and navigation buttons (Home, << Previous, Next >>). The main area contains Scheme code defining two functions: `(define (area-of-disk r) (* 157/50 (* r r)))` and `(define (area-of-ring outer inner) (- (area-of-disk outer) (area-of-disk inner)))`. A blue horizontal line separates the code from the execution result. The expression `(- 157/2 1413/50)` is highlighted in green, and the result `1256/25` is highlighted in purple, with a red arrow pointing from the expression to the result. The window has standard OS window controls (minimize, maximize, close) in the top right and a scrollbar on the right side.

```
(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

$$= (- (\text{area-of-disk } 5) (\text{area-of-disk } 3))$$

(- (area-of-disk outer)
(area-of-disk inner))

(こ outer = 5, inner = 3 が代入される)

$$= (- (* 3.14 (* 5 5)) (\text{area-of-disk } 3))$$

(* 3.14
(* r r))

に r = 5 が代入される

$$= (- (* 3.14 25) (\text{area-of-disk } 3)) \quad (* 5 5) \rightarrow 25$$

$$= (- 78.5 (\text{area-of-disk } 3)) \quad (* 3.14 25) \rightarrow 78.5$$

$$= (- 78.5 (* 3.14 (* 3 3))) \quad (* 3.14 (* r r)) \quad \text{に } r = 3 \text{ が代入される}$$

$$= (- 78.5 (* 3.14 9)) \quad (* 3 3) \rightarrow 9$$

$$= (- 78.5 28.26) \quad (* 3.14 9) \rightarrow 28.26$$

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

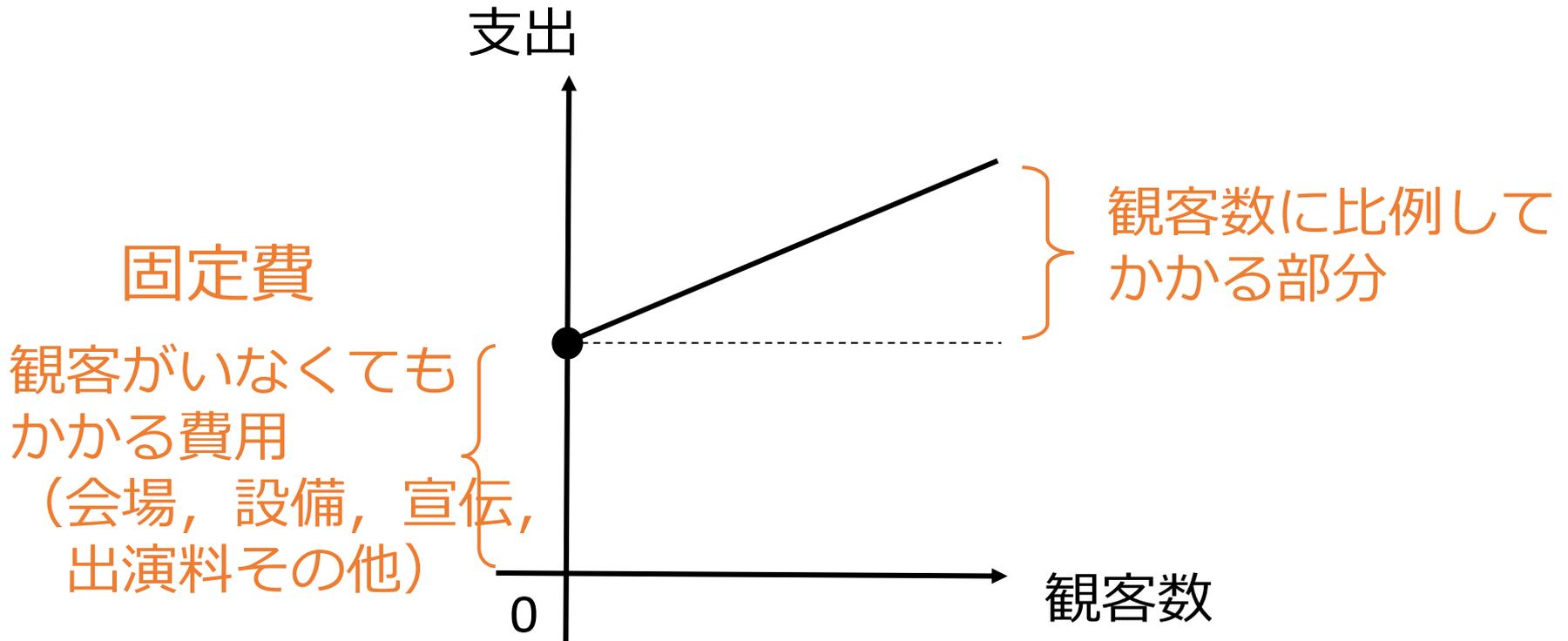
= 50.24

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

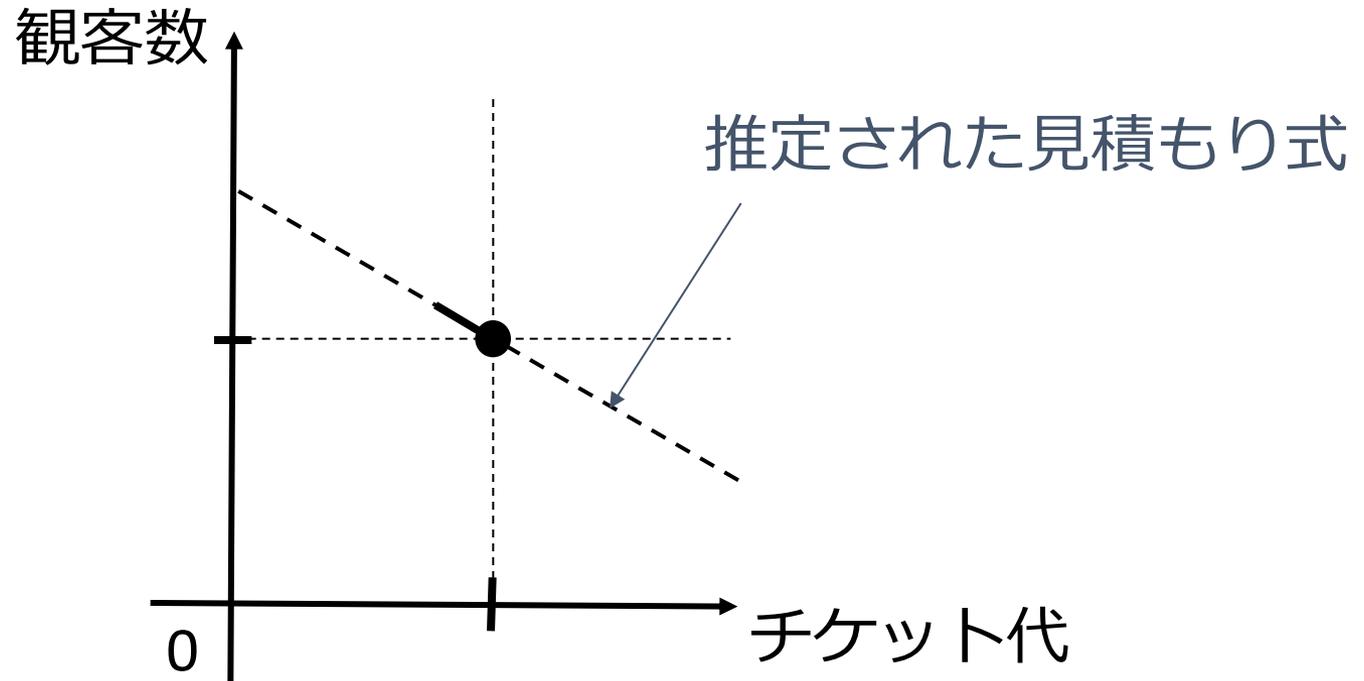


$$\bullet \text{ 支出} = \text{固定費} + \text{観客数} \times \text{費用}$$

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

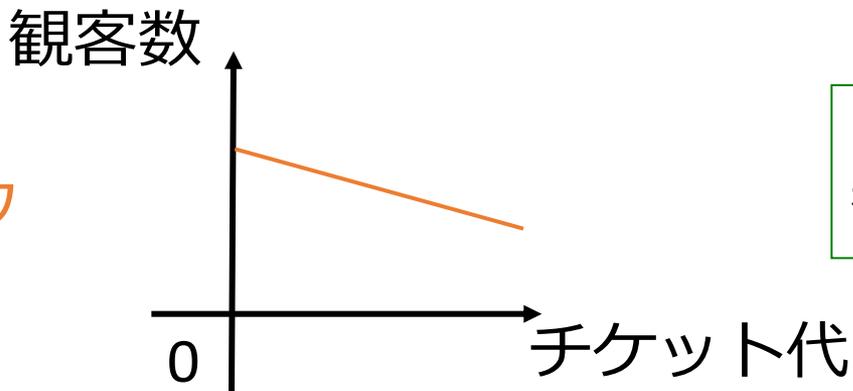
チケット代: \$ 0.1 値下げすると15人増えた

⇒ 観客数 = $-(15 / 0.1) \times (\text{チケット代} - \$ 5) + 120$ と見積もる

- 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

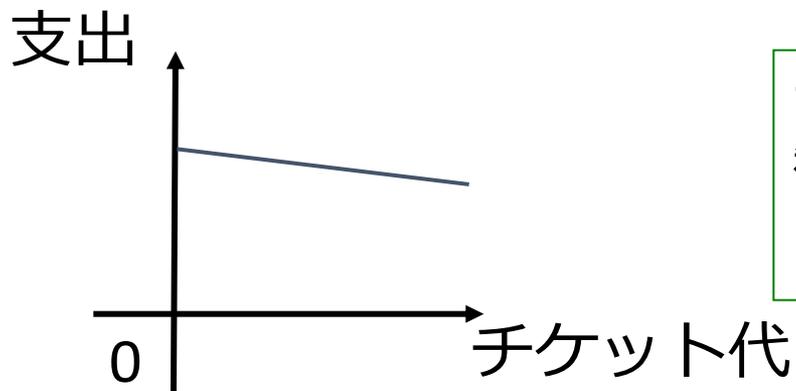


観客数
のグラフ



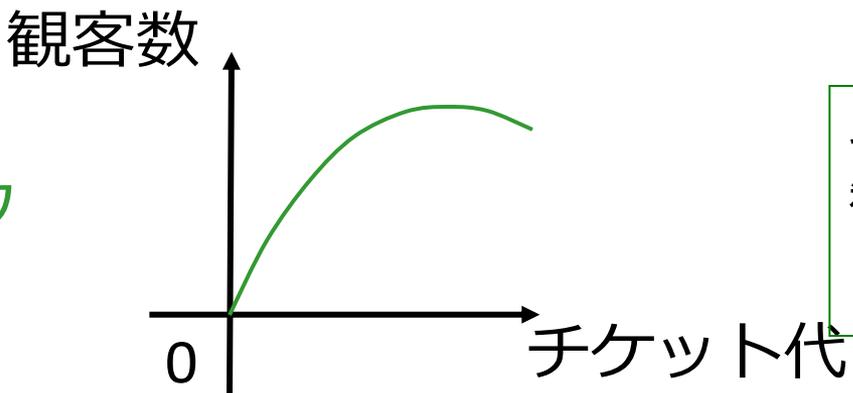
チケット代を上げると
観客数が減る

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「定義用ウィンドウ」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) ... )
```

「例題 7. 利益の計算」の手順 (2/4)



2. 「定義用ウィンドウ」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. 「定義用ウィンドウ」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))
```

「例題 7. 利益の計算」の手順 (3/4)



4. 「定義用ウィンドウ」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. 「定義用ウィンドウ」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))))
```

「例題 7 . 利益の計算」の手順 (4/4)



6. その後, 次を「実行用ウィンドウ」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)

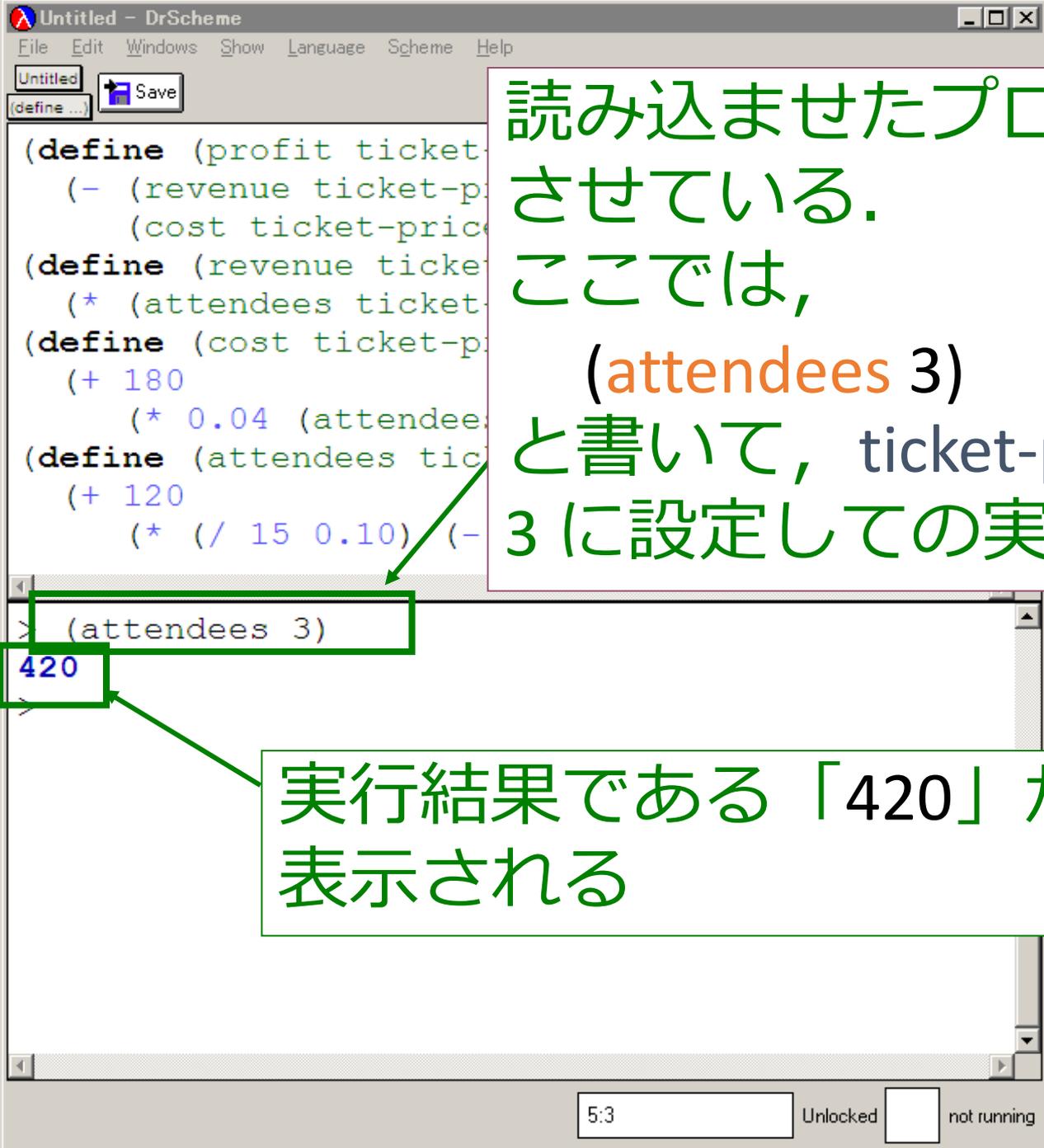
☆ 次は, 例題 8 に進んでください

```
Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled Save
define ... Check Syntax Step Execute Break

(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))))

>
```

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



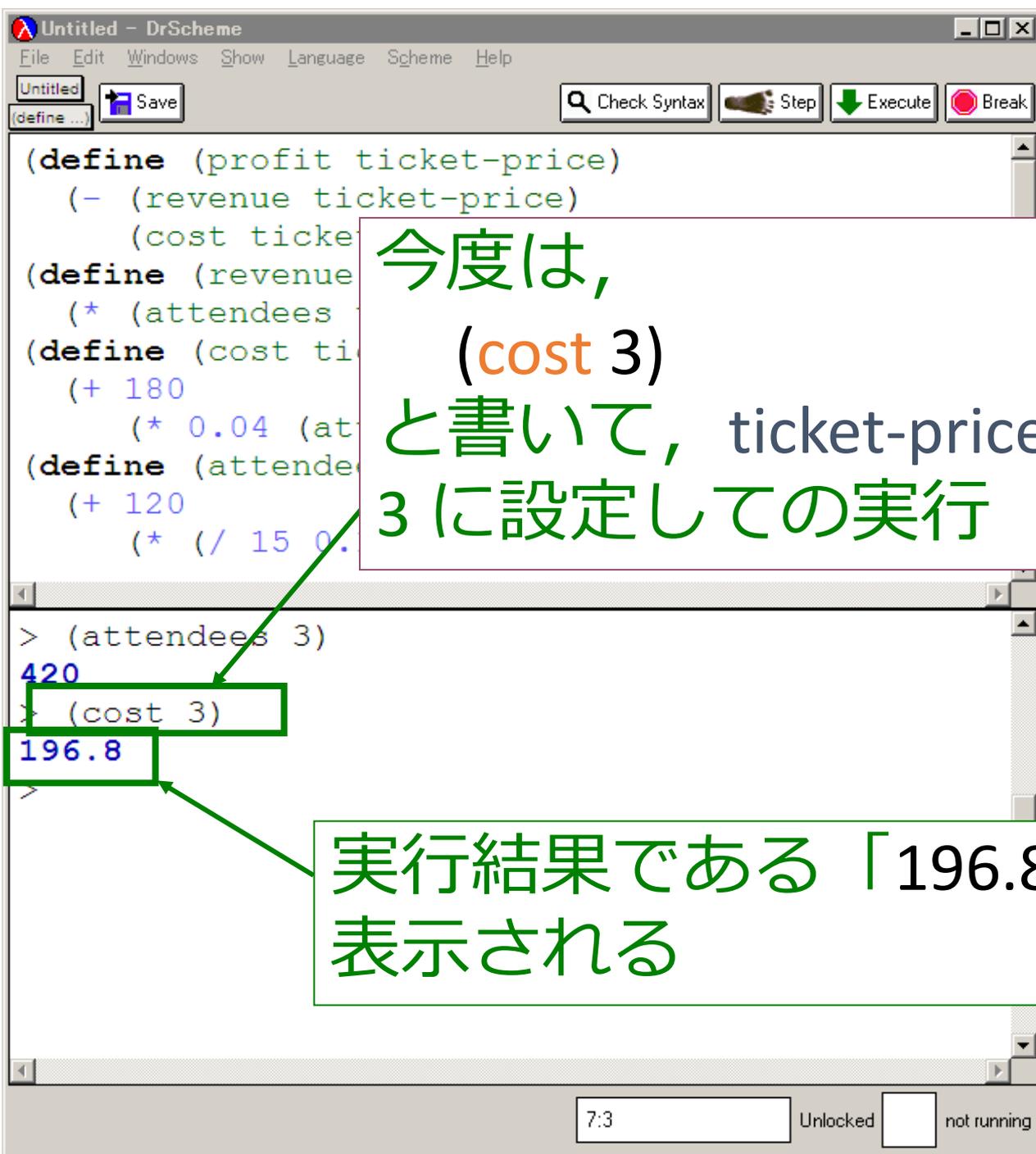
読み込ませたプログラムを実行
させている。

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



The screenshot shows the DrScheme IDE with the following code in the editor:

```
(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 0.04) (ticket-price))))
```

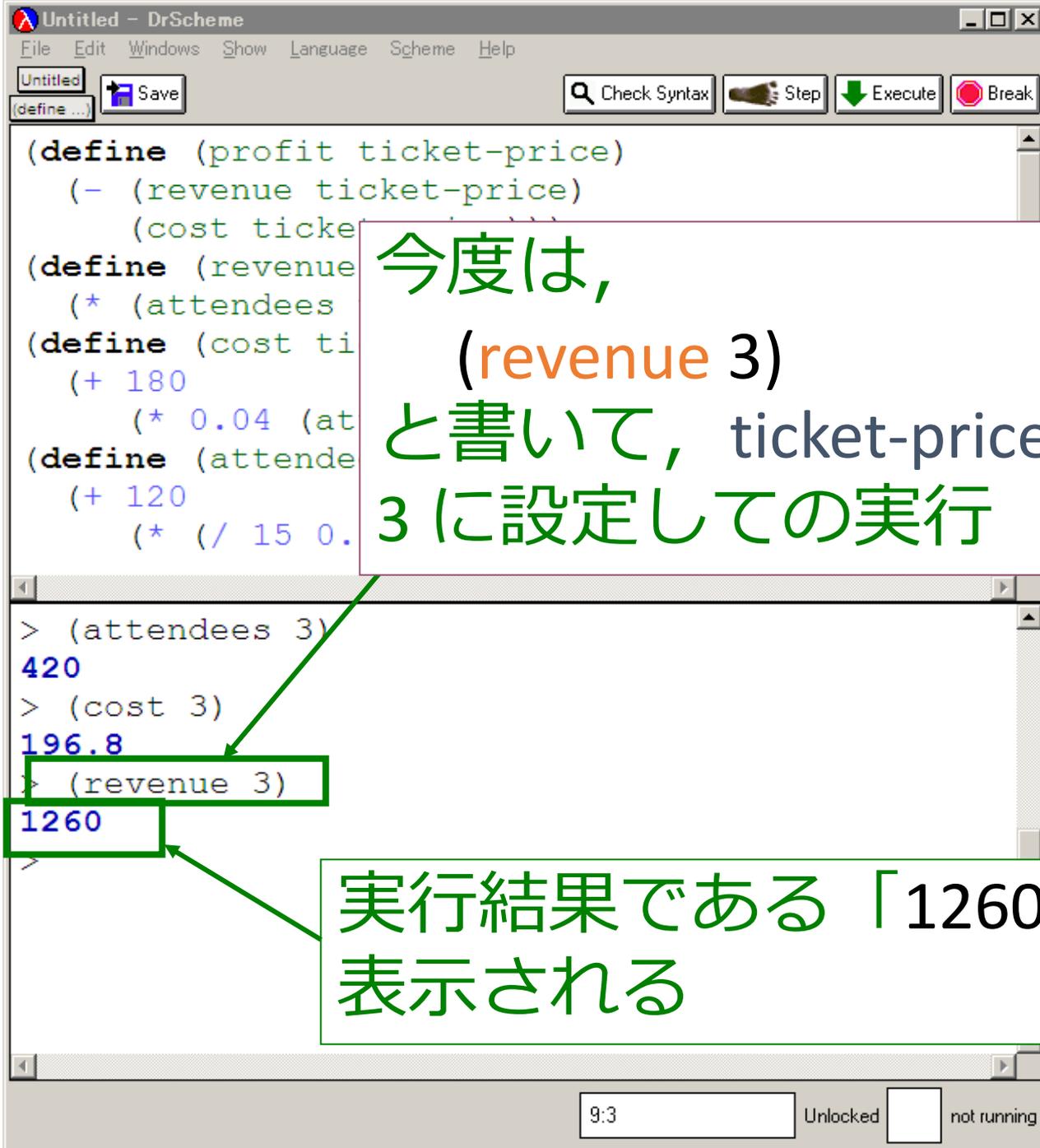
The console shows the following execution steps:

```
> (attendees 3)
420
> (cost 3)
196.8
```

The status bar at the bottom indicates the current time is 7:3, the window is Unlocked, and the program is not running.

今回は、
(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



The screenshot shows the DrScheme IDE with the following content:

- Editor:** Scheme code defining functions for profit, revenue, and cost calculations.
- Console:** Execution results for the defined functions with the value 3.
- Bottom Panel:** Status bar showing "9:3", "Unlocked", and "not running".

```
(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 0.04) (ticket-price))))
```

Console output:

```
> (attendees 3)
420
> (cost 3)
196.8
> (revenue 3)
1260
```

今回は、
(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

```
Untitled - DrScheme
File Edit Windows Show Language Scheme Help
[define ...] Save [Check Syntax] Step Execute Break

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

> (attendees 3)
420
> (cost 3)
196.8
> (revenue 3)
1260
> (profit 3)
1063.2
>
```

今回は、
(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



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)))
```

profit 関数

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

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

revenue 関数

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

```
  (+ 180
```

```
     (* 0.04 (attendees ticket-price))))
```

cost 関数

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

```
  (+ 120
```

```
     (* (/ 15 0.10) (- 5.00 ticket-price))))
```

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)))) 106
```

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「定義用ウィンドウ」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

$$\begin{aligned} &= (- (\text{revenue } 3) (\text{cost } 3)) \\ &= (- (* (\text{attendees } 3) 3) (\text{cost } 3)) \\ &= (- (* (+ 120 (* (/ 15 0.10) (- 5.00 3))) 3) (\text{cost } 3)) \\ &= (- (* (+ 120 (* 150 (- 5.00 3))) 3) (\text{cost } 3)) \\ &= (- (* (+ 120 (* 150 2)) 3) (\text{cost } 3)) \\ &= (- (* (+ 120 300) 3) (\text{cost } 3)) \\ &= (- (* 420 3) (\text{cost } 3)) \\ &= (- 1260 (\text{cost } 3)) \\ &= (- 1260 (+ 180 (* 0.04 (\text{attendees } 3)))) \\ &= (- 1260 (+ 180 (* 0.04 (+ 120 (* (/ 15 0.10) (- 5.00 3))))) \\ &= (- 1260 (+ 180 (* 0.04 (+ 120 (* 150 (- 5.00 3))))) \\ &= (- 1260 (+ 180 (* 0.04 (+ 120 (* 150 2))))) \\ &= (- 1260 (+ 180 (* 0.04 (+ 120 300)))) \\ &= (- 1260 (+ 180 (* 0.04 420))) \\ &= (- 1260 (+ 180 16.8)) \\ &= (- 1260 196.8) \end{aligned}$$

コンピュータ内部での計算

=1063.2 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

- 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