sp-8. プログラムの設計法と種々のエラー

（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

8-1 プログラム設計法

8-2 種々のエラー

8-3 パソコン演習

8-4 課題

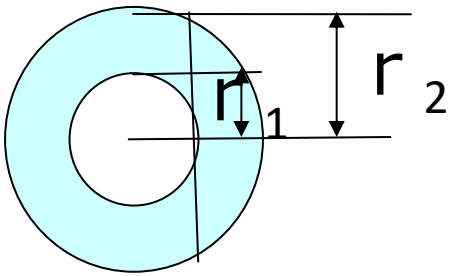
8-1 プログラム設計法

リングの面積

- 第１回講義の「リングの面積」について，「プログラム設計法(Design Recipe)」を示す

- 内径と外径からリングの面積を求めるプログラム area-of-ring を書く
  - 円の面積を求めるプログラム area-of-disk を使用
  - 名前： area-of-ring
  - パラメータ： outer（外径）, inner（内径）

真ん中に穴のあいた
円の面積と考える



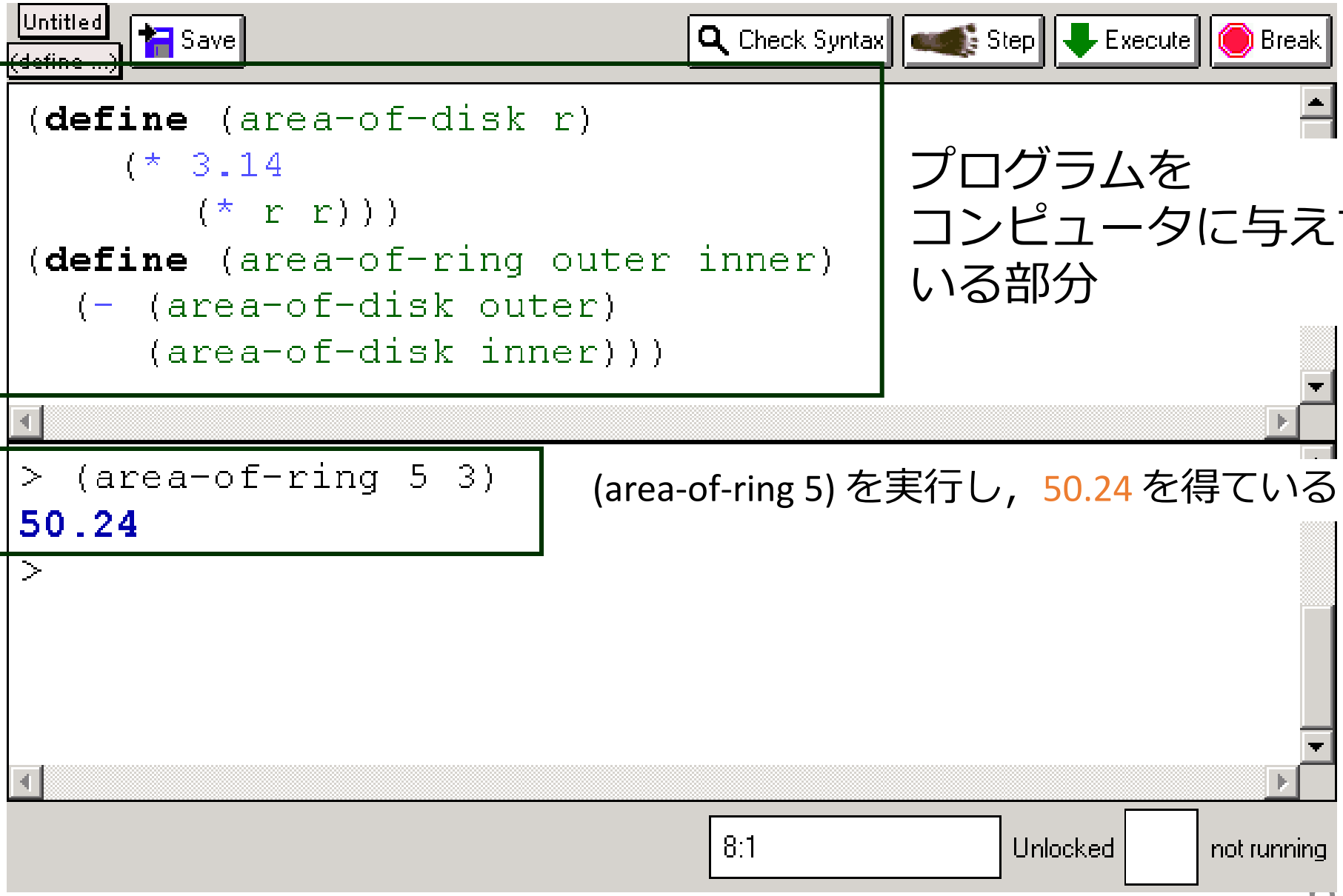
リングの面積のプログラム

```
(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

実行結果の例



プログラムを
コンピュータに与えて
いる部分

(area-of-ring 5) を実行し，50.24 を得ている

プログラム設計法

- 何をすべきか，そして，行うべきことの順番についての段階的な規定

- Design Recipe （デザインレシピ）ともいう

プログラム設計法

プログラム開発に必要な活動

1. Contract：
   - 問題の解析，関数の名前と入出力の定義
2. Purpose：
   - プログラムの目的の理解．プログラムの振る舞いの設計
3. Example
   - プログラムの振る舞いを「例」で記述．
4. Definition
   - プログラムの定義
5. Test
   - 検査を通じた誤り（エラー）の発見

Contract

- 問題の解析
- 問題が取り扱うデータの種類の記述

area-of-ring : number number -> number

プログラム名
(意味を良く表す名前)

プログラムの
入力と出力

プログラム化すると

```
;; area-of-ring : number number -> number
(define (area-of-ring outer inner) ...)
```

Purpose

- プログラムの振る舞いに関する仕様の明確化
  - プログラムの仕様，目的など

to compute the area of a ring whose radius is outer and whose hole has a radius of inner

プログラムが何をするか，
何を計算するかを書いた文章

プログラム化すると

```
;; area-of-ring : number number -> number
;; to compute the area of a ring whose radius is
;; outer and whose hole has a radius of inner
(define (area-of-ring outer inner) ...)
```

Purpose の効果

- プログラムが何をするものかのコメント．引数名をコメント中に盛り込むと，よりわかりやすい

```
;; area-of-ring : number number -> number
;; to compute the area of a ring whose radius is
;; outer and whose hole has a radius of inner
(define (area-of-ring outer inner) ...)
```

Example

- 例題を使った，プログラムの振る舞いの例示
- 入出力関係を特徴づけるような「例」での記述

area-of-ring should produce 50.24

for the inputs 5 and 3

入力と期待される出力
（プログラムの理解の助けになる）

プログラム化すると

```
;; area-of-ring : number number -> number
;; to compute the area of a ring whose radius is
;; outer and whose hole has a radius of inner
;; example: (area-of-ring 5 3) should produce 50.24
(define (area-of-ring outer inner) ...)
```

Example の効用

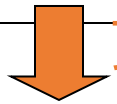
特定の入力に対する出力の記述

1. 論理的なエラーの発見の手段.
2. プログラムを正確に理解するための手段（関数の入力と出力の関係や，計算過程など）
3. 後でプログラムを読み返すときのための「メモ」として

Definition

- Example まで終わったら，プログラム本体の記述に取りかかる.
- Contract において「...」としていた部分を実際に作成

ドーナツ型の面積は，外側の面積から
内側の面積を引いたもの
プログラムの振る舞い



プログラム化すると

```
;; area-of-ring : number number -> number
;; to compute the area of a ring whose radius is
;; outer and whose hole has a radius of inner
;; example: (area-of-ring 5 3) should produce 50.24
(define (area-of-ring outer inner)
  (- (area-of-disk outer)
     (area-of-disk inner)))
```

Definition

- 与えられた入力から，プログラムがどのように出力を計算するのかを理解した後に行う
- 記述していた「Example」を再度見て，特定の入力に対する出力をどのように計算したかを理解すること
- 入出力関係が，数式で与えられていれば，　→　直ちにプログラムを書ける
- 言葉で問題が与えられていれば，
  →　注意深くプログラムを作る

Test

- テストを通したエラーの発見
  - 少なくとも，「Example」で書いた場合のテストは行う
  - この段階で，エラーの発見に努める

Test では

- 「Example」に対して，期待された出力を計算するかを確かめる.
- 間違った出力に対しては

  → 「Example」と「プログラムの中身そのもの」を確認
  - Example の間違い
  - プログラムの中身の間違い

プログラム設計法の例（まとめ）

Contract:

area-of-ring : number number -> number

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

Purpose:

to compute the area of a ring whose radius is outer and whose hole has a radius of inner

Example:

(area-of-ring 5 3) should produce 50.24

Definition:

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

Tests:

```
(area-of-ring 5 3) ;; expected value 50.24
```

参考 Web ページ

http://www.htdp.org/2001-11-21/Book/node14.htm

ここまでのまとめ

1. Contract
   - 問題の解析．問題が取り扱うデータの種類の記述
2. Purpose
   - プログラムの振る舞いに関する仕様の明確化
     （プログラムの仕様，目的など）
3. Example
   - 例題を使った，プログラムの振る舞いの例示
4. Definition
   - プログラムの作成
5. Test
   - テストを通したエラーの発見

8-2 種々のエラー

種々のエラー

DrSchemeが見つける.

- 構文エラー(Syntax Errors)
  - 言語Schemeの書式に合っていない場合

  e.g. (10) , (10 + 20), (define (P x) (+ (x) 10))
- 実行エラー(Run-time Errors)
  - Schemeの書式に合っているプログラムの全てが，結果を返すとは限らない.

  e.g. (/ 1 0),　(define (f n) (+ (/ n 3) 2))に対する(f 5 8)
- 論理的エラー(Logical Errors)
  - Schemeの書式に合っていて，答えも出すが，その答えが間違っている.

  e.g. (define (area-of-triangle b h) (+ b h))

  (define (wage h) (+ 12 h))

注意深く，規則正しくプログラムを設計することで発見できる.

分野の知識(Domain Knowledge)

- プログラムを定義するには，プログラムをしようとする領域の知識（分野の知識）が必要.
  - e.g. 数学の知識，音楽，生物学，土木工学，芸術..
  - その領域の言葉を理解することが求められる
  - 時には，その応用領域のデータを表現する言語を発明しなければならない.
- コンピュータ言語にとらわれない，しっかりとした基礎の理解が不可欠.

8-3 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

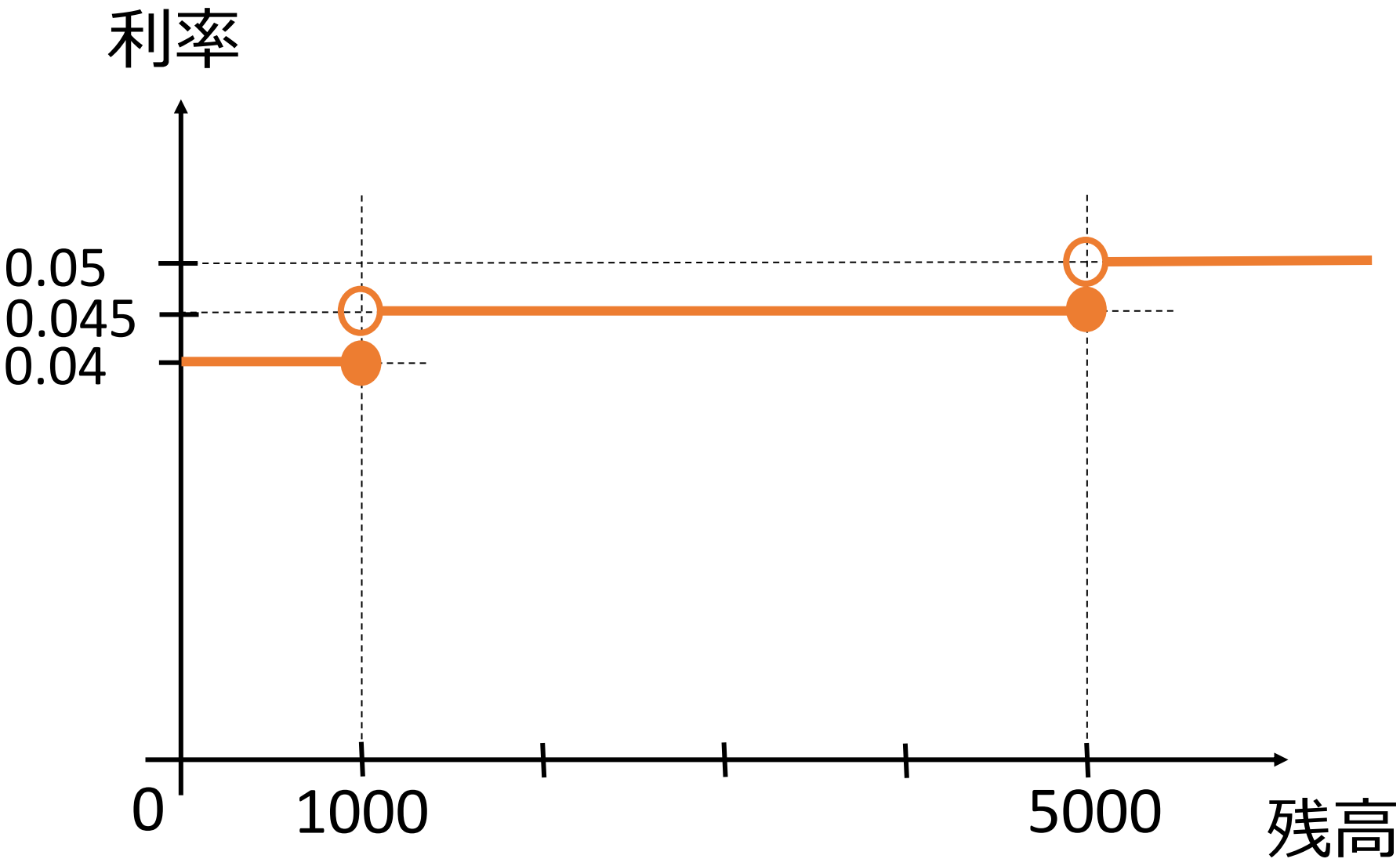
DrScheme の使用

- 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

例題１．条件式

- 残高 amount から利率を求める関数 interest-rate を作る
  - 残高が $1000 以下ならば： 4%
  - 残高が $5000 以下ならば： 4.5%
  - 残高が $5000より多ければ： 5%
- 残高を条件とする条件式を使う
- cond 句が登場

例題１．条件式



入力と出力

入力　　出力

条件式のプログラム

```
;; interest-rate: number -> number

;; to determine the interest rate

;; for the given amount
(define (interest-rate amount)
  (cond
    [(<= amount 1000) 0.040]
    [(<= amount 5000) 0.045]
    [(>  amount 5000) 0.050]))
```

amount ≦ 1000 のとき

1000 ＜ amount ≦ 5000 のとき

5000 ＜ amount のとき

条件式

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

(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

実行エラーの例

- first, rest に関する実行エラー(runtime error)

	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

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

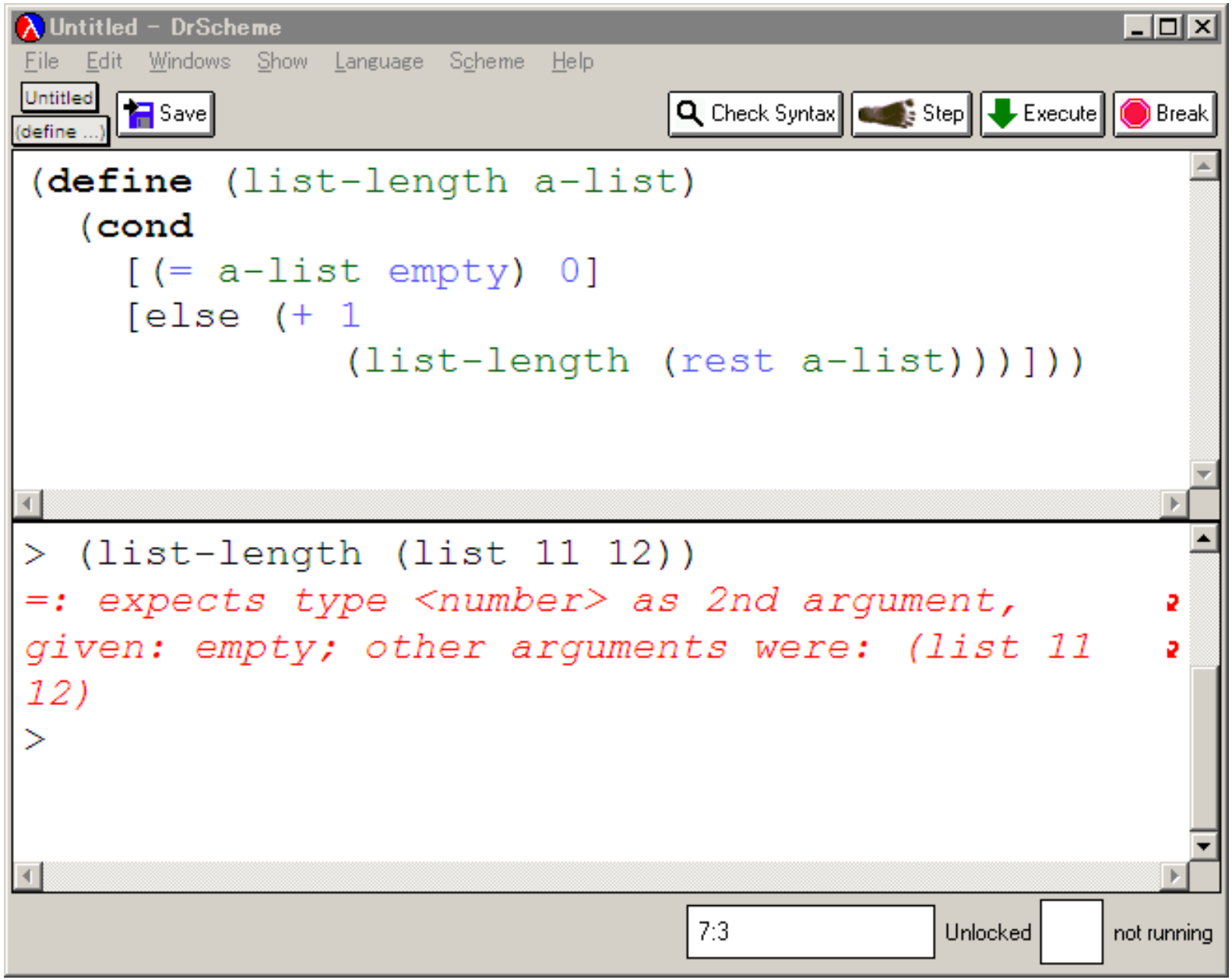
よくある勘違い

```
(define (list-length a-list)
   (cond
      [(= a-list empty) 0]
      [else (+ 1
               (list-length (rest a-list)))]))
```

終了条件の判定：

- 正しくは「(empty? a-list)」
- x がリストのとき、(= x empty) は実行エラー
- 「=」は数値の比較には使えるが，リスト同士の比較には使えない

実行エラーの例


Untitled – DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Save
Check Syntax
Step
Execute
Break
(define (list-length a-list)
(cond
[(= a-list empty) 0]
[else (+ 1
(list-length (rest a-list)))]))
> (list-length (list 11 12))
=: expects type <number> as 2nd argument,
given: empty; other arguments were: (list 11
12)
>
7:3
Unlocked
not running

プログラム作製のプロセス

エラーメッセージ

- 期待された入力以外の時には，警告のメッセージを出したい．error関数を使う．

```
(error  SYMBOL  STRING)

(define (give-name name)
   (cond
     [(symbol? name) (format “Hello, ~s” name)]
    [else (error ‘give-name “symbol expected”)]))
```

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

プログラムの作成手順と問題の分析について

1. 関数の名前とパラメータを決める

2. 関数の機能を，文章で書く

3. 関数の中身を書く

8-4 課題

課題 1

- 次のように関数 f が定義されている時，「(f 5 20)」 を DrScheme で実行すると，どうなるか，説明せよ。

```
(define (f n)
  (* n 20))
```

課題２

- 次の Scheme式のエラーの原因について，分かりやすく説明しなさい
  - 次のScheme式を DrScheme を使って実行すると，赤い文字で，エラーメッセージが表示される

1. (10 + 20)　　・・・構文エラー
2. (define (f y)　・・・実行エラー
   (+ x 10))のとき，(f 5)を実行
3. (define (g x)　・・・構文エラー
   + x 10)
4. (define h(x)　・・・構文エラー
   (+ x 10))
5. (define (f n)　・・・実行エラー
   (+ (/ n 3) 2))のとき，(f 5 8)を実行
6. (+ 5 (/ 1 0))　・・・実行エラー