sp-11. 構造体とグラフィックス

（Scheme プログラミング）

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

金子邦彦

CC BY NC SA

アウトライン

11-1 define による変数定義

11-2 DrScheme でのグラフィックス

11-3 パソコン演習

11-4 課題

11-1 define による変数定義

変数

- 変数には，名前，値がある

名前　　値

例）

```
(define PI 3.14)
(define A (list 15 8 6))
```

リストの変数定義

(define A (list 15 8 6))

変数名　リストの本体

変数とは

- 「オブジェクト」に名前を付けたもの
  - オブジェクトとは，数値，文字列，リストなど．「値」を持つ
- コンピュータは，変数の値と名前の関係を記憶している

11-2 DrScheme でのグラフィックス

DrScheme でのグラフィックス

- DrScheme のグラフィックス機能である draw.ss teachpack

  - start: 「描画用ウインドウ」を開く
  - draw-solid-line:線
  - draw-solid-rect: 四角形
  - draw-solid-disk: 塗りつぶされた円
  - clear-solid-disk: 一度描いた「塗りつぶされた円」を消す
  - draw-circle: 円
  - stop: 「描画用ウインドウ」を閉じる

11-3 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
  - 遠慮なく質問してください
- 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

draw.ss teach pack のロード

- DrScheme の描画機能である draw.ss teachpack を使うために,

draw.ss teach pack をロードせよ

この操作は１回だけでよい
（次回からは自動的にロードされるようになる）

draw.ss teach pack のロード

Language

→ Add Teachpack

→ htdp ディレクトリを選択

→ draw.ss を選択

→ Execute ボタン

例題１．簡単な絵を描く

- DrScheme の描画機能である draw.ss teachpack を使って，簡単な絵を描く

  - start: 「描画用ウインドウ」を開く
  - draw-solid-line:線
  - draw-solid-rect: 四角形
  - draw-solid-disk: 塗りつぶされた円
  - clear-solid-disk: 一度描いた「塗りつぶされた円」を消す
  - draw-circle: 円
  - stop: 「描画用ウインドウ」を閉じる

「例題１．簡単な絵を描く」の手順

1. 次を「実行用ウインドウ」で実行しなさい

```
(start 100 100)
(draw-solid-line (make-posn 0 0) (make-posn 80 80) 'red)
(draw-solid-rect (make-posn 50 50) 50 20 'green)
(draw-circle (make-posn 20 20) 20 'blue)
(draw-solid-disk (make-posn 70 70) 10 'red)
  (stop)
```

☆　次は，例題２に進んでください

Untitled

(define ...)

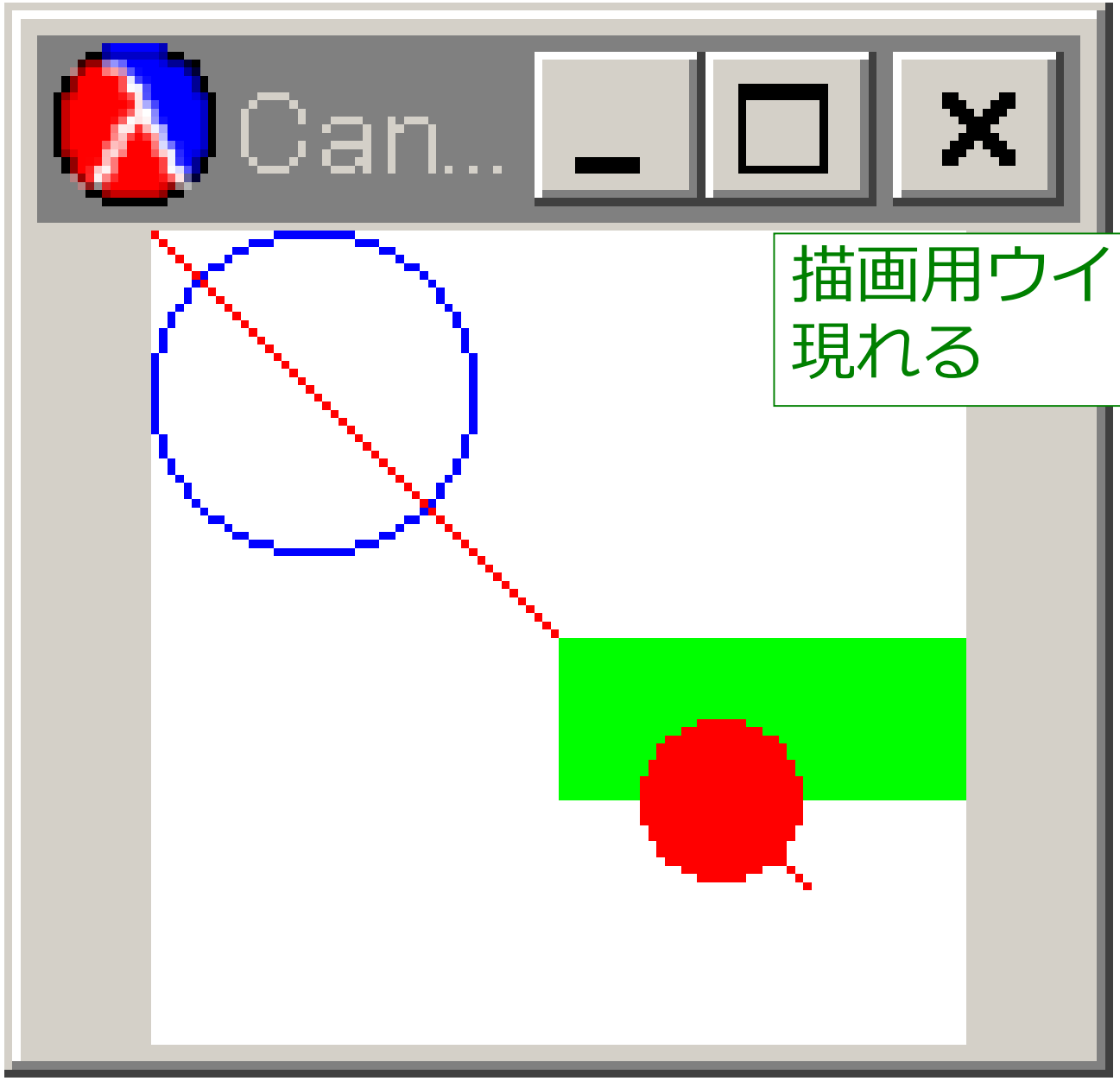
Save

Check Syntax | Step | Execute | Break

```
>  (start 100 100)
> (draw-solid-line (make-posn 0 0) (make-posn 80 80) 'red)
true
> (draw-solid-rect (make-posn 50 50) 50 20 'green)
true
> (draw-circle (make-posn 20 20) 20 'blue)
true
> (draw-solid-disk (make-posn 70 70) 10 'red)
true
```

28:3 Unlocked not running

例題１の実行結果



描画用ウインドウが
現れる

例題１のまとめ

- 描画用のウインドウを開く

```
(start 100 100)
```

- 描画の実行

```
(draw-solid-line (make-posn 0 0) (make-posn 80 80) 'red)
(draw-solid-rect (make-posn 50 50) 50 20 'green)
(draw-circle (make-posn 20 20) 20 'blue)
(draw-solid-disk (make-posn 70 70) 10 'red)
```

- 描画用ウインドウを閉じる

```
(stop)
```

Untitled

(define ...)

Save

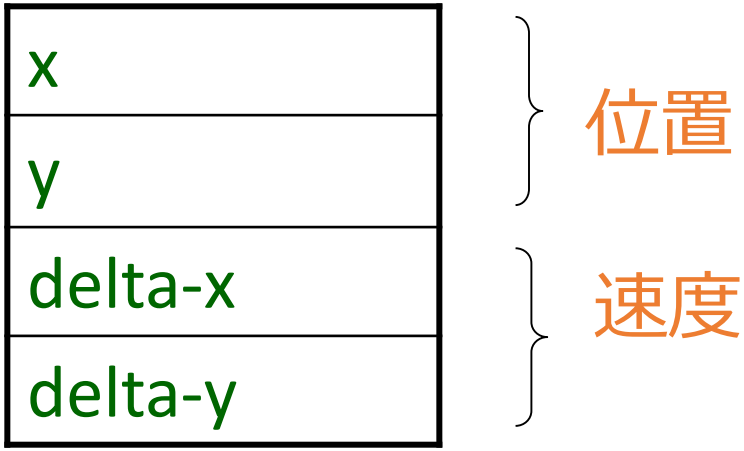
Check Syntax | Step | Execute | Break

```
>  (start 100 100)
> (draw-solid-line (make-posn 0 0) (make-posn 80 80) 'red)
true
> (draw-solid-rect (make-posn 50 50) 50 20 'green)
true
> (draw-circle (make-posn 20 20) 20 'blue)
true
> (draw-solid-disk (make-posn 70 70) 10 'red)
true
```

28:3 Unlocked not running

例題２． ball 構造体を描く

- ball 構造体を使って，(x, y) の位置に円を描くプログラム draw-ball を書く
  - 構造体とグラフィックス処理の組み合わせ
  - ball 構造体の属性値を設定するために make-ball （コンストラクタ）を使う
  - 属性 x, y を取り出すために ball-x, ball-y （セレクタ）を使う



「例題２． ball 構造体を描く」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-struct ball
   (x y delta-x delta-y))
;; draw-ball: a ball -> none
;; draw a solid disk at (x,y)
(define (draw-ball a-ball)
  (draw-solid-disk (make-posn
                    (ball-x a-ball)
                    (ball-y a-ball))
                   5 'red))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(start 100 100)
(draw-ball (make-ball 10 10 0 0))
(stop)
```

☆　次は，例題３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-struct ball
               (x y delta-x delta-y))
```

>

5:3 Unlocked not running

まず，ball 構造体を定義している

Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled (define ...) Save Check Syntax Step Execute Break

```
(define-struct ball (x y delta-x delta-y))
;; draw-ball: a ball -> none
;; draw a solid disk at (x,y)
(define (draw-ball a-ball)
  (draw-solid-disk (make-posn
                    (ball-x a-ball)
                    (ball-y a-ball))
                   5 'red))
```

>

次に，関数 draw-ball を定義している

4:3 Unlocked not running

Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled (define ...) Save Check Syntax Step Execute Break

```
(define-struct ball
               (x y delta-x delta-y))
(define (draw-ball a-ball)
  (draw-solid-disk (make-posn
                    (ball-x a-ball) (ball-y a-ball)) 5 'red))
```

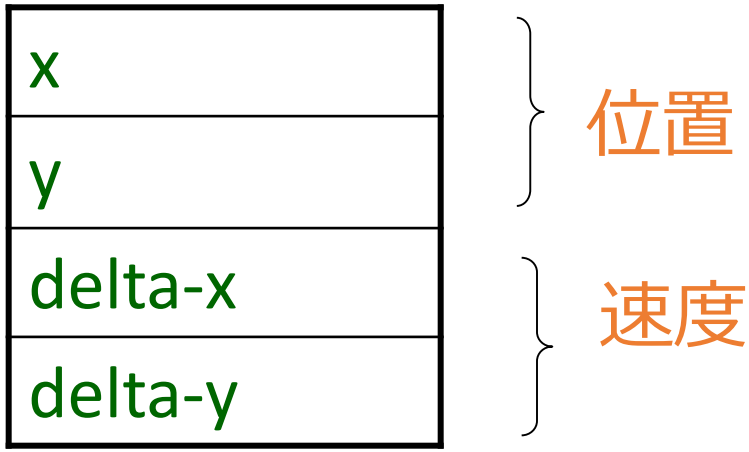
```
> (start 100 100)
> (draw-ball (make-ball 10 10 0 0))
true
>
```

7:2 Unlocked not running

描画用ウインドウが現れる

ball 構造体

- ball は，x, y, delta-x, delta-y から構成する



名前

```
(define-struct ball
                (x y delta-x delta-y))
```

属性の並び

```
(define-struct ball
   (x y delta-x delta-y))
;; draw-ball: a ball -> none
;; draw a solid disk at (x,y)
(define (draw-ball a-ball)
  (draw-solid-disk (make-posn
                    (ball-x a-ball)
                    (ball-y a-ball))
                   5 'red))
```

```
(start 100 100)
(draw-ball (make-ball 10 10 0 0))
(stop)
```

例題３．リストの変数定義

- リスト (list 15 8 6) を値として持つ変数Aを定義する
  - 変数を定義するために define を使う
  - リストを作るために cons を使う

「例題３．リストの変数定義」の手順

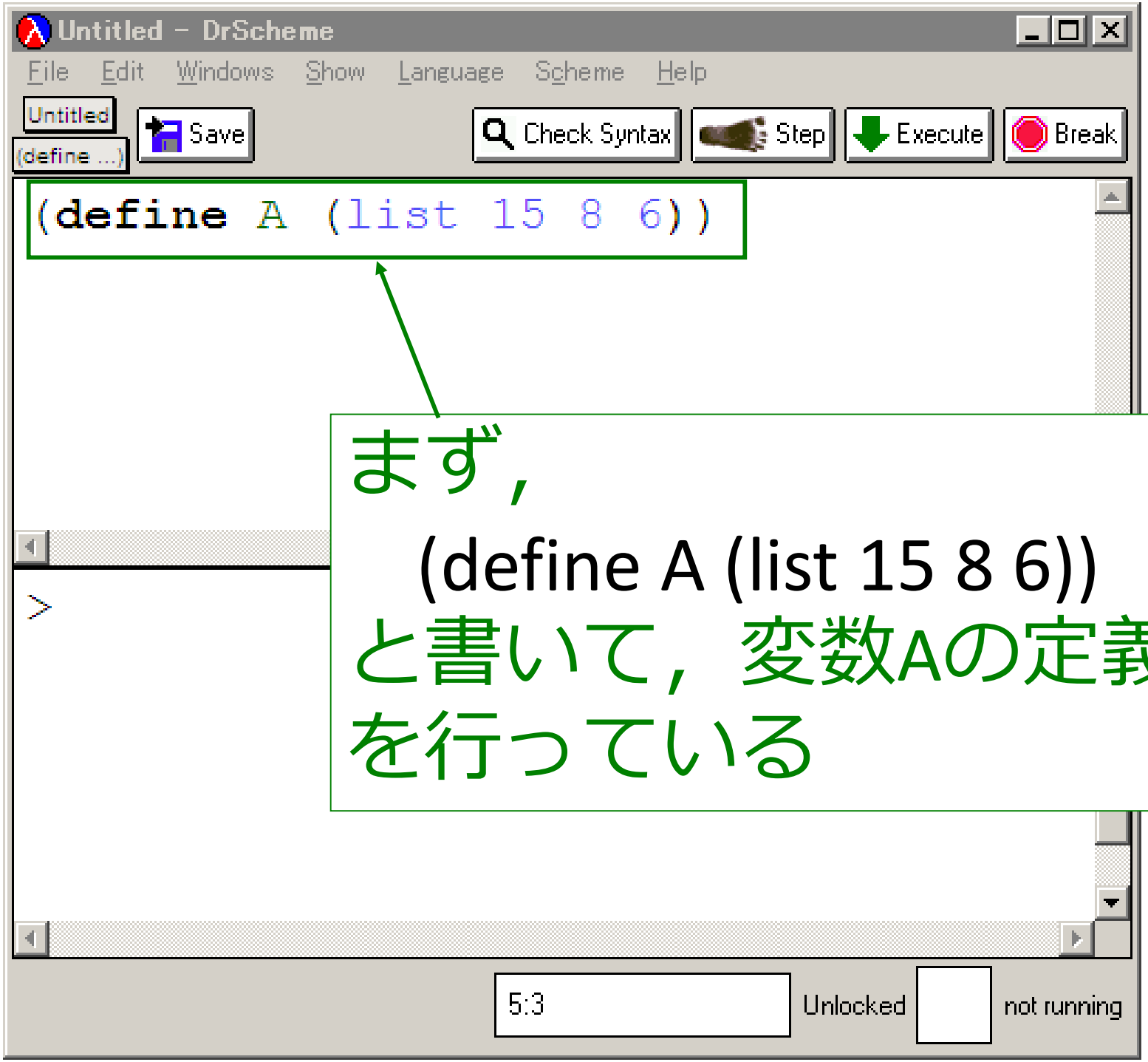
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Execute ボタンを押す

```
(define A (list 15 8 6))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
A
(first A)
(rest A)
```

☆　次は，例題４に進んでください



まず，

(define A (list 15 8 6))

と書いて，変数Aの定義を行っている

Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled (define ...) Save Check Syntax Step Execute Break

```
(define A (list 15 8 6))
```

```
> A
(list 15 8 6)
>
```

A

と実行すると，Aの値である

(list 15 8 6)

が表示される

Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled (define ...) Save Check Syntax Step Execute Break

```
(define A (list 15 8 6))
```

(first A), (rest A) を実行している

```
> A
(list 15 8 6)
> (first A)
15
> (rest A)
(list 8 6)
>
```

例題４．構造体のリスト

- 次のプログラムを実行し，「(name-list book)」から「(list "Mike" "Bill" "Ken")」に至る過程の概略を見る

```
(define-struct address-record
            (name age address))
(define book (list
            (make-address-record "Mike" 10 "Fukuoka")
            (make-address-record "Bill" 20 "Saga")
            (make-address-record "Ken" 30 "Nagasaki")))
(define (name-list a-book)
  (cond
    [(empty? a-book) empty]
    [else
     (cons (address-record-name (first a-book))
          (name-list (rest a-book)))]))
```

「例題４．構造体のリスト」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-struct address-record
                (name age address))
(define book (list
                (make-address-record "Mike" 10 "Fukuoka")
                (make-address-record "Bill" 20 "Saga")
                (make-address-record "Ken" 30 "Nagasaki")))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
book
(first book)
(address-record-address (first book))
```

☆　次は，例題 5 に進んでください

Untitled
(define ...)
Save
Check Syntax
Step
Execute
Break

```
(define-struct address-record
        (name age address))
```

AddressNote 構造体の定義

```
(define book (list
                (make-address-record "Mike" 10 "Fukuoka")
                (make-address-record "Bill" 20 "Saga")
                (make-address-record "Ken" 30 "Nagasaki")))
```

変数 book の定義
（AddressNote 構造体のリス

```
> book
(list
 (make-address-record "Mike" 10 "Fukuoka")
 (make-address-record "Bill" 20 "Saga")
 (make-address-record "Ken" 30 "Nagasaki"))
> (first book)
(make-address-record "Mike" 10 "Fukuoka")
> (address-record-address (first book))
"Fukuoka"
```

変数 book の操作

13:3 Unlocked not running

例題５．頂点のリストを値とする変数の定義

- 「頂点」のリストを値として持つ変数 P を定義する
  - 変数を定義するために define を使う
  - １つの頂点　＝　posn 構造体 make-posn を使用
  - リストを作るために cons を使う

「例題５．頂点のリストを値とする変数の定義」の手順

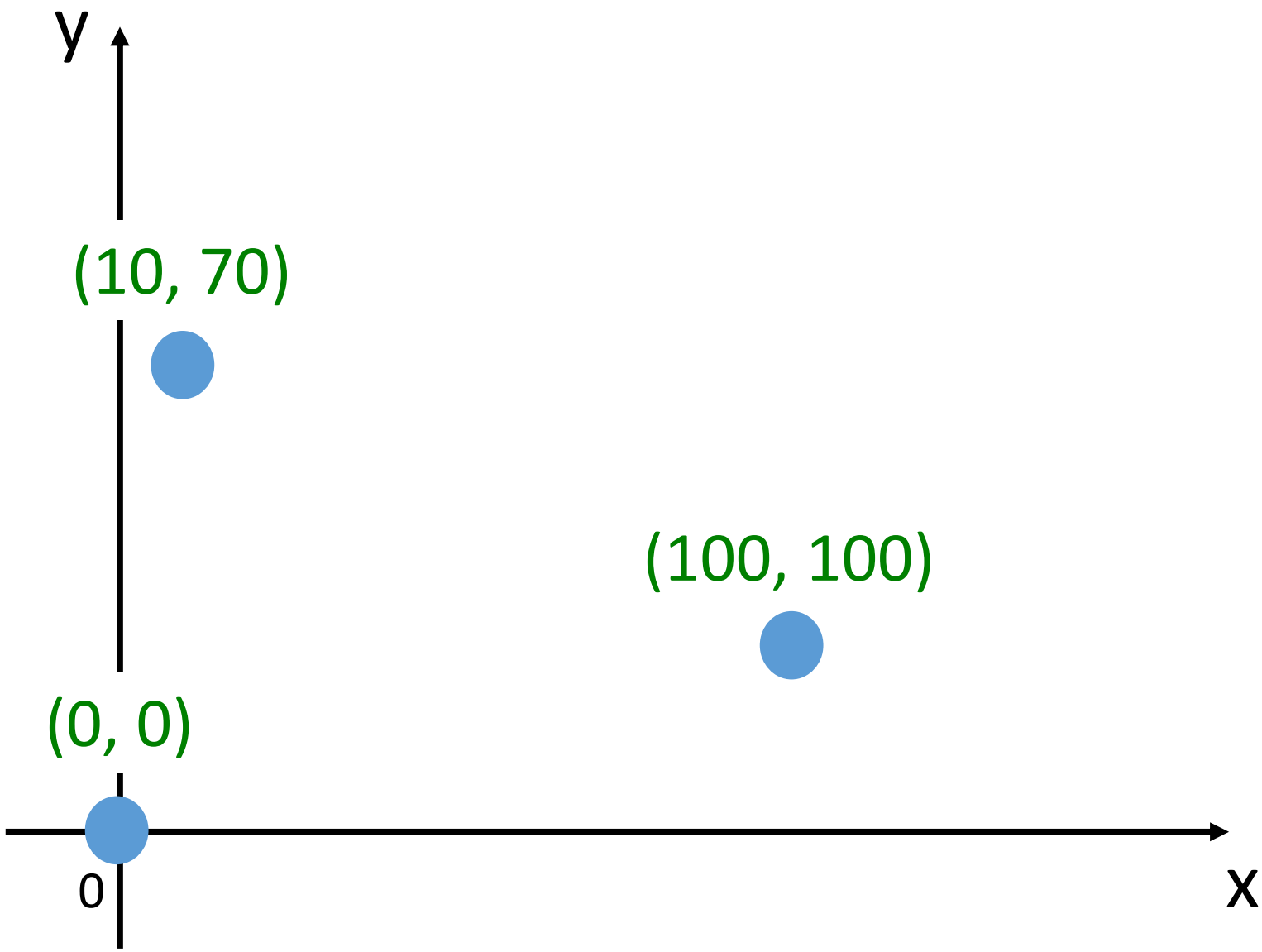
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Execute ボタンを押す

```
(define P
  (cons (make-posn 0 0)
    (cons (make-posn 10 70)
      (cons (make-posn 100 30) empty))))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
P
(first P)
(rest P)
```

☆　次は，例題６に進んでください


y
(10, 70)
(100, 100)
(0, 0)
0
x

Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled (define ...) Save Check Syntax Step Execute Break

```
(define P
    (cons (make-posn 0 0)
          (cons (make-posn 10 70)
                (cons (make-posn 100 30) empty))))
```

>

4:3 Unlocked not running

まず，変数Pの定義を行っている

Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled (define ...) Save Check Syntax Step Execute Break

```
(define P
    (cons (make-posn 0 0)
          (cons (make-posn 10 70)
                (cons (make-posn 100 30) empty))))
```

```
> P
(list
 (make-posn 0 0)
 (make-posn 10 70)
 (make-posn 100 30))
>
```

Pの値が表示されている

9:3 Unlocked not running

Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled (define ...) Save Check Syntax Step Execute Break

```
(define P
    (cons (make-posn 0 0)
          (cons (make-posn 10 70)
                (cons (make-posn 100 30) empty))))
```

```
> P
(list
 (make-posn 0 0)
 (make-posn 10 70)
 (make-posn 100 30))
> (first P)
(make-posn 0 0)
> (rest P)
(list (make-posn 10 70) (make-posn 100 30))
>
```

(first P), (rest P) を実行している

変数定義

変数名

```
(define P
  (cons (make-posn 0 0)
    (cons (make-posn 10 70)
      (cons (make-posn 100 30)
 empty))))
```

リストの本体

posn 構造体

- posn 構造体は，すでに DrScheme に組み込み済み

- (define-struct 構造名) を実行していなくても，posn 構造体を使うことができる

例題６．折れ線

- 折れ線を描くプログラム draw-polyline を作り，実行する
  - １つの「頂点」　→　構造体
  - 折れ線　→　「頂点」のリスト

「例題６．折れ線」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

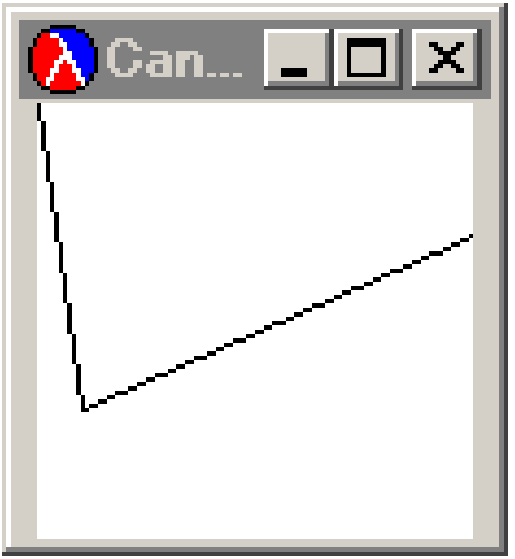
```
(define P
   (cons (make-posn 0 0)
      (cons (make-posn 10 70)
      (cons (make-posn 100 30) empty))))
(define (draw-polyline a-poly)
  (cond
    [(empty? (rest a-poly)) true]
    [else (and
            (draw-solid-line (first a-poly)
                             (first (rest a-poly)))
            (draw-polyline (rest a-poly)))]))
```

← 例題５と同じ

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(start 100 100)
(draw-polyline P)
(stop)
```

☆　次は，例題７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 P
  (cons (make-posn 0 0)
        (cons (make-posn 10 70)
              (cons (make-posn 100 30) empty))))
(define (draw-polyline a-poly)
  (cond
    [(empty? (rest a-poly)) true]
    [else (and
           (draw-solid-line (first a-poly)
                            (first (rest a-poly)))
           (draw-polyline (rest a-poly)))]))
```

Welcome to DrScheme, version 103p1.
Language: Intermediate Student.
Teachpack: C:\Program Files\PLT\teachpack\htdp\draw.ss.

```
> (start 100 100)
> (draw-polyline P)
true
>
```

7:3 Unlocked not running

入力と出力

a-poly の値：

```
(cons (make-posn 0 0)
   (cons (make-posn 10 70)
         (cons (make-posn 100 30) empty)))
```

入力は posn 構造体のリスト

出力は常に true

```
(define (draw-polyline a-poly)
  (cond
    [(empty? (rest a-poly)) true]
    [else (and
            (draw-solid-line (first a-poly)
                             (first (rest a-poly)))
            (draw-polyline (rest a-poly)))]))
```

終了条件

自明な解

折れ線 draw-polyline

1. (rest a-poly) が空ならば：　→　終了条件

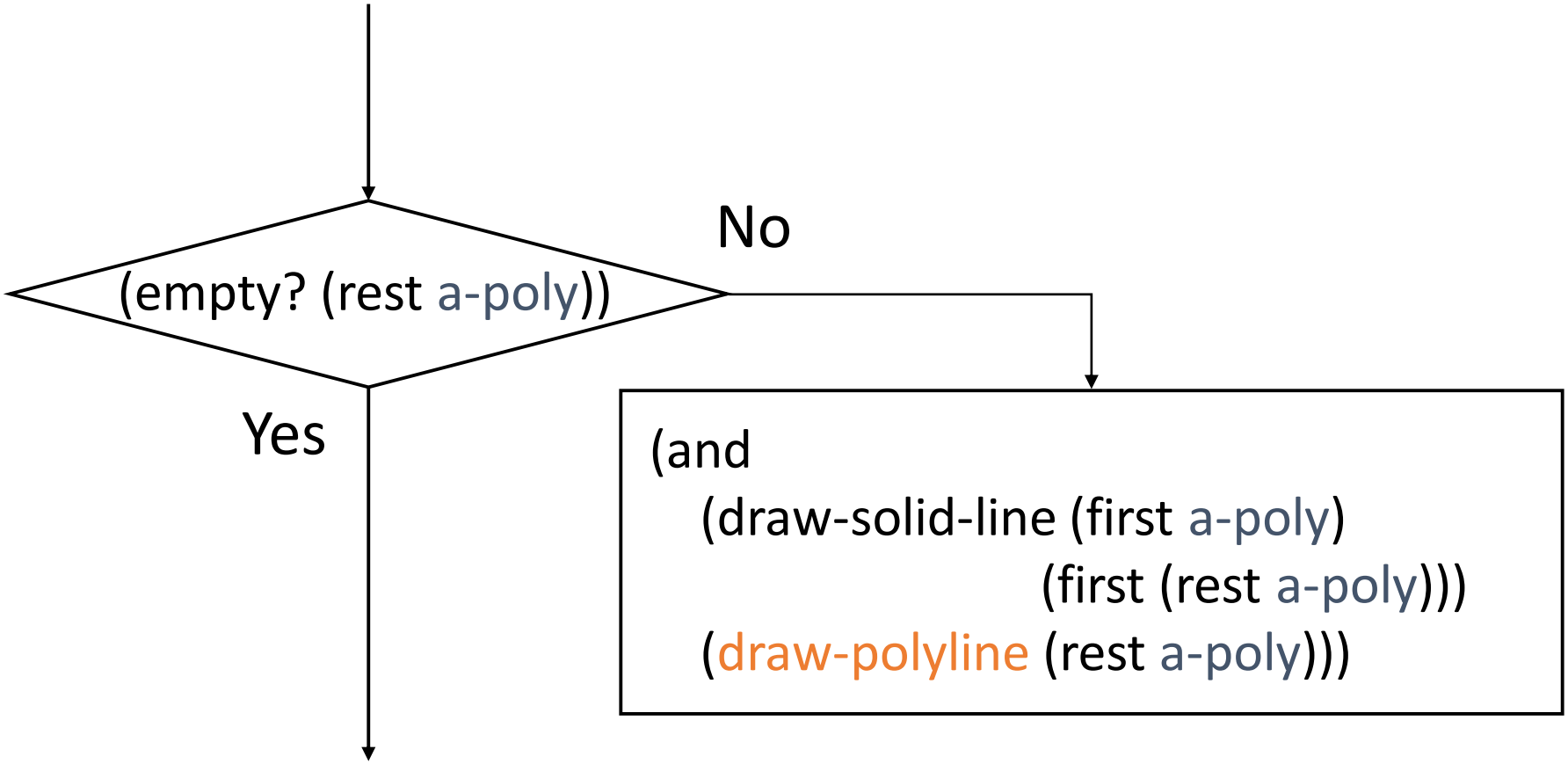
   true　→　自明な解

2. そうで無ければ：

   - ２点：　(first a-poly) と (first (rest a-poly)) を使って，線分を書き，その後 (draw-polyline (rest a-poly)) を実行

     ⇒ 結局，すべての点について，線分を描くことを繰り返す

折れ線の終了条件



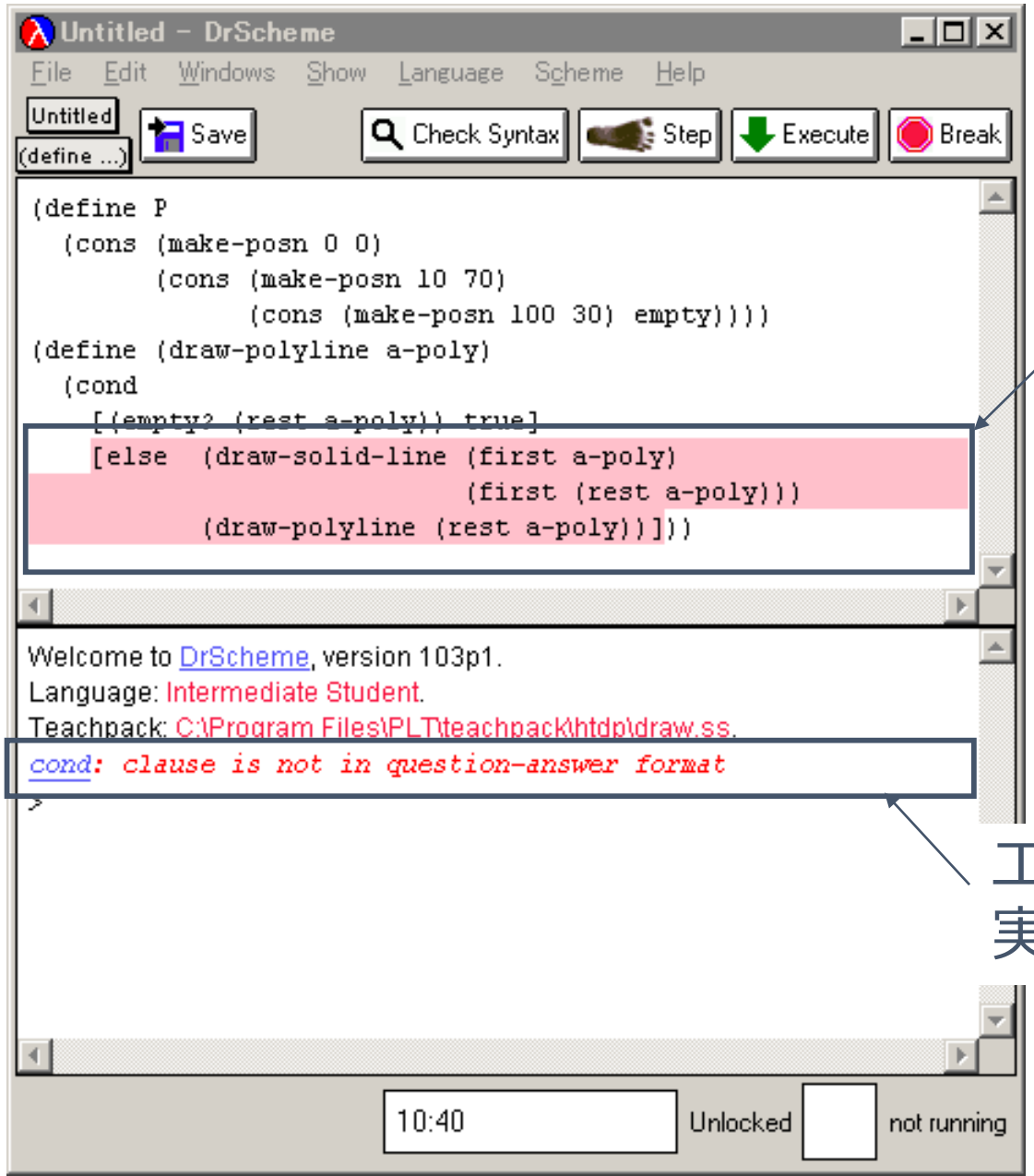
true が自明の解

描画命令と他の命令を並べるときは「and」でつなぐ

```
(and
    (draw-solid-line (first a-poly)
                     (first (rest a-poly)))
    (draw-polyline (rest a-poly)))
```

描画命令 draw-solid-line
と animation とを and でつないでいる

よくある間違い



描画命令と他の命令が並んでいるのだが、「and」を書き忘れている

エラーが出て，実行できない

(draw-polyline P) から true が得られる過程の概略 (1/2)

(draw-polyline P) 最初の式

= (draw-polyline (list (make-posn 0 0) (make-posn 10 70) (make-posn 100 30)))

= ...

= (and

(draw-solid-line (makeposn 0 0) (make-posn 10 70))

(draw-polyline (rest (list (make-posn 0 0) (make-posn 10 70) (make-posn 100 30)))))

= …

= (draw-polyline (rest (list (make-posn 0 0) (make-posn 10 70) (make-posn 100 30))))

= (draw-polyline (list (make-posn 10 70) (make-posn 100 30)))

コンピュータ内部での計算

(draw-polyline P) から true が得られる過程の概略 (2/2)

= ...

= (and

(draw-solid-line (makeposn 10 70) (make-posn 100 30))

(draw-polyline (rest (list (make-posn 10 70) (make-posn 100 30)))))

= ...

= (draw-polyline (rest (list (make-posn 10 70) (make-posn 100 30))))

= (draw-polyline (list (make-posn 100 30)))

= ...

= true

終了条件が成立

コンピュータ内部での計算

実行結果

例題７．多角形

- 折れ線を描くプログラム draw-polygon を作り，実行する
  - １つの「頂点」　→　構造体
  - 折れ線　→　「頂点」のリスト
  - 終点と始点を結ぶ（これが，例題６との違い）

「例題７．多角形」の手順 (1/2)

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 P
   (cons (make-posn 0 0)
      (cons (make-posn 10 70)
            (cons (make-posn 100 30) empty))))
     (define (draw-polyline a-poly)
  (cond
   [(empty? (rest a-poly)) true]
   [else (and
            (draw-solid-line (first a-poly)
                             (first (rest a-poly)))
            (draw-polyline (rest a-poly)))]))
(define (last a-poly)
  (cond
    [(empty? (rest a-poly)) (first a-poly)]
    [else (last (rest a-poly))]))
(define (draw-polygon a-poly)
   (draw-polyline (cons (last a-poly) a-poly)))
```

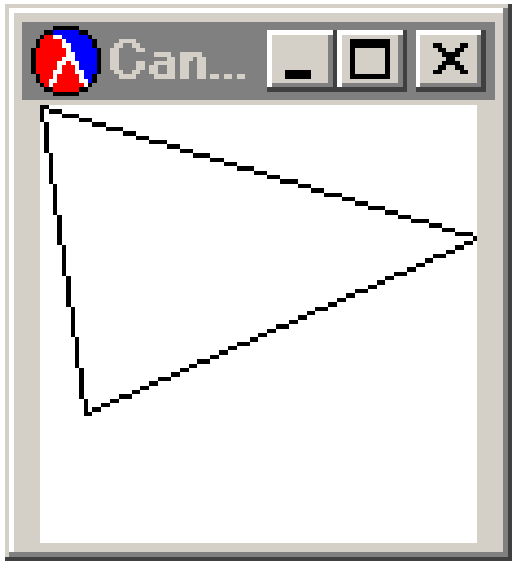
例題６と同じ

「例題７．多角形」の手順 (2/2)

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(start 100 100)
(draw-polygon P)
(stop)
```

☆　次は，課題に進んでください



Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled (define ...) Save Check Syntax Step Execute Break

```
(define P
  (cons (make-posn 0 0)
        (cons (make-posn 10 70)
              (cons (make-posn 100 30) empty))))
(define (draw-polyline a-poly)
  (cond
    [(empty? (rest a-poly)) true]
    [else (and
            (draw-solid-line (first a-poly)
                             (first (rest a-poly)))
            (draw-polyline (rest a-poly)))]))
(define (last a-poly)
  (cond
    [(empty? (rest a-poly)) (first a-poly)]
    [else (last (rest a-poly))]))
(define (draw-polygon a-poly)
  (draw-polyline (cons (last a-poly) a-poly)))
```

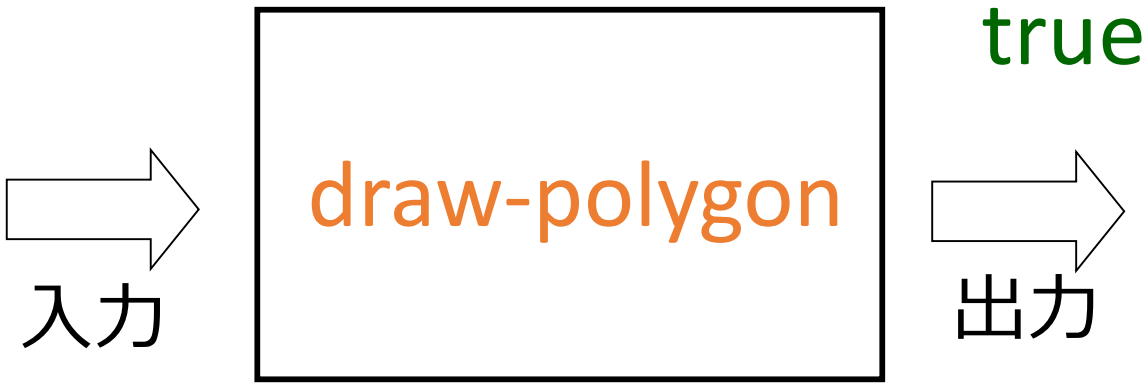
```
Welcome to DrScheme, version 103p1.
Language: Intermediate Student.
Teachpack: C:\Program Files\PLT\teachpack\htdp\draw.ss.
> (start 100 100)
> (draw-polygon P)
true
>
```

7:3 Unlocked not running

入力と出力

a-poly の値：

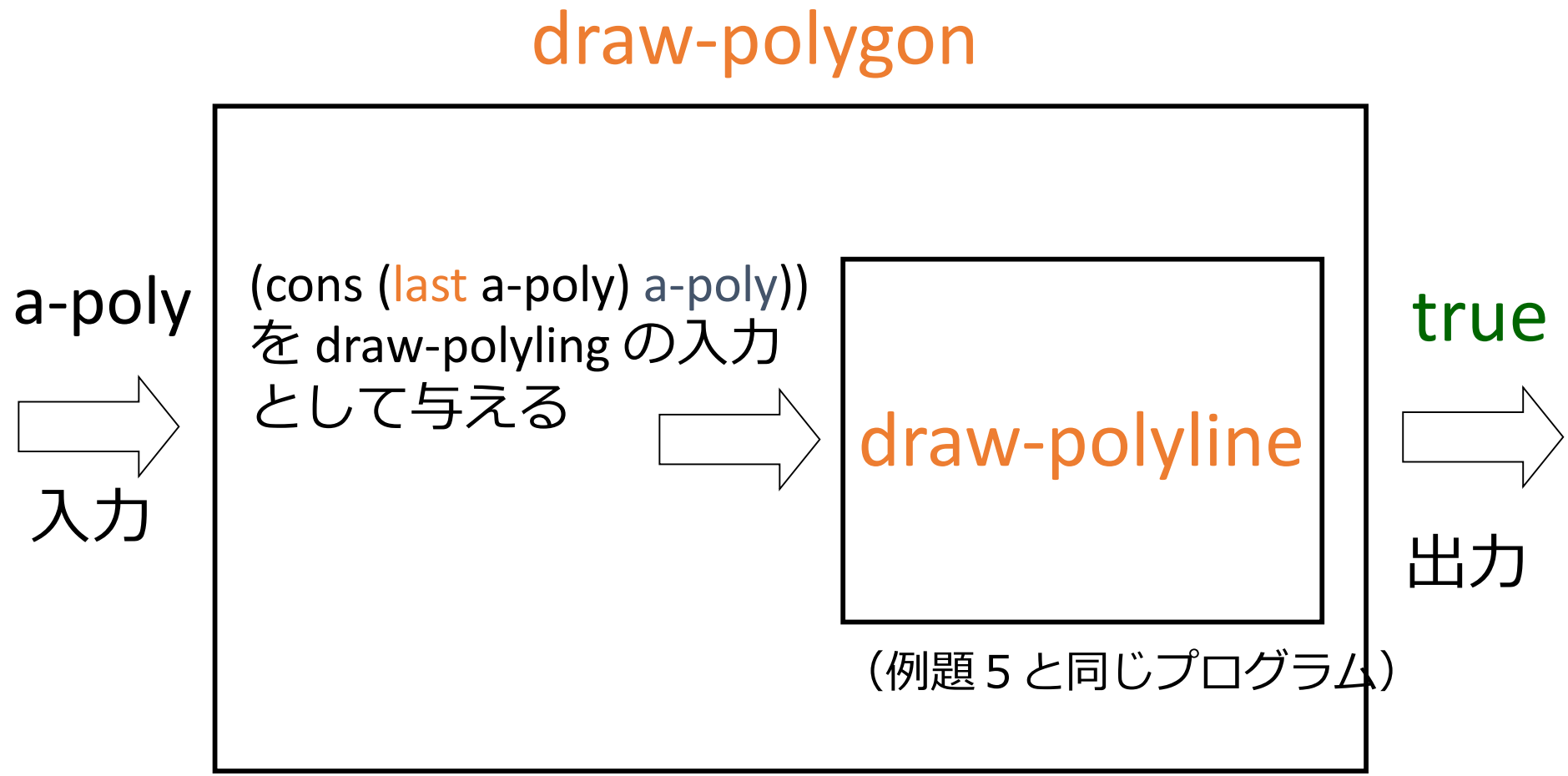
```
(cons (make-posn 0 0)
   (cons (make-posn 10 70)
         (cons (make-posn 100 30) empty)))
```



入力は posn 構造体のリスト

出力は常に true

draw-polygon の中で行っていること



入力は posn 構造体
のリスト

出力は常に true

```
(define (draw-polyline a-poly)
  (cond
    [(empty? (rest a-poly)) true]
    [else (and
            (draw-solid-line (first a-poly)
                             (first (rest a-poly)))
            (draw-polyline (rest a-poly)))]))
(define (last a-poly)
  (cond
    [(empty? (rest a-poly)) (first a-poly)]
    [else (last (rest a-poly))]))
(define (draw-polygon a-poly)
  (draw-polyline (cons (last a-poly) a-poly)))
```

(draw-polygon P) からの過程の概略

(draw-polygon P)

= (draw-polygon (list (make-posn 0 0) (make-posn 10 70) (make-posn 100 30)))

= ...

= (draw-polyline (list (make-posn 100 30) (make-posn 0 0) (make-posn 10 70) (make-posn 100 30)))

以下省略

(cons (last a-poly) a-poly)) の実行結果

折れ線のリストからの描画

- 例題１との関係
  - draw_function: 入力：　折れ線のリスト
  - draw_接線
  - draw_小区画‘

11-4 演習課題

課題 1

- 次の式を順に実行し，実行結果を報告せよ.

1. (start 300 300)
2. (draw-solid-line (make-posn 100 100) (make-posn 200 200) ’red)
3. (draw-circle (make-posn 200 100) 50 ’red)
4. (draw-solid-disk (make-posn 100 200) 50 ’green)
5. (stop)

課題２

- ball 構造体（授業の例題２）についての問題
  - ball のデータ a-ball から，ボールの速さを求める関数を作成し，実行結果を報告しなさい
  - ボールの速さは：　$\sqrt{\delta x^2 + \delta y^2}$

課題３

- ball 構造体（授業の例題２）についての問題
  - ball のデータ a-ball について，「x < 0 または y < 0 または x > 100 または y > 100 のときのみ true を出力し，そうでなければfalseを出力」 するような関数を作成し，実行結果を報告しなさい

課題４

- x-y 平面上の２点 a と b から，その間の距離を求める関数 distance を作成し，実行結果を報告しなさい
  - ２点 a, b のための構造体を定義すること

課題 5

- 複数のball を描画するプログラムの作成.
  - 次の関数 draw-ball は， ball のデータ a-ball から１つのballを描くプログラムである.
  - これを参考にして，複数のballを描く関数 draw-all-balls を作成しなさい．draw-all-balls の入力はball構造体のリストである.
  - 複数の ball を扱うので，ball 構造体のリストを扱うことになる
  - 必ず動作確認まで行うこと.

```
(define (draw-ball a-ball)
  (draw-solid-disk (make-posn
                    (ball-x a-ball) (ball-y a-ball)) 5 'red))
```

課題 6

- 複数のball を消すプログラムの作成.
  - 次の関数 clear-ball は， ball のデータ a-ball から１つのballを消すプログラムである.
  - これを参考にして，複数のballを描く関数 clear-all-balls を作成しなさい． clear-all-balls の入力はball構造体のリストである.
  - 必ず動作確認まで行うこと.

```
(define (clear-ball a-ball)
  (clear-solid-disk (make-posn
                     (ball-x a-ball) (ball-y a-ball)) 5))
```

さらに勉強したい人への補足説明資料

DrScheme でのアニメーション

内容

- Dr Scheme でのタイマー機能とアニメーション

例題８． ball を１秒描く

- ball 構造体（授業の例題２）を使って，(x, y) の位置に，ball を１秒だけ描くための関数 draw-and-clear を作り，実行する
  - 円を描く関数：　draw-solid-disk 関数を使う
  - 円を消す関数：　clear-solid-disk 関数を使う
  - 円を書いた後に，sleep-for-a-while 関数を使って１秒待ち，円を消す
  - and 文を使い，draw-solid-disk, sleep-for-a-while, clear-solid-disk を順次実行する

「例題８．ball を１秒描く」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-struct ball
                (x y delta-x delta-y))
(define DELAY 1)
;;draw-and-clear: ball -> true
;;draw, sleep, clear a disk from the canvas
;;structural design, Scheme knowledge
(define (draw-and-clear a-ball)
  (and
    (draw-solid-disk (make-posn (ball-x a-ball) (ball-y a-ball)) 5 'red)
    (sleep-for-a-while DELAY)
    (clear-solid-disk (make-posn (ball-x a-ball) (ball-y a-ball)) 5 'red)))
```

← 例題２と同じ

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(start 100 100)
(draw-and-clear (make-ball 10 10 0 0))
(stop)
```

File Edit Windows Show Language Scheme Help

Untitled

(define ...)

Save | Check Syntax | Step | Execute | Break

```
(define DELAY 1)
(define-struct ball (x y delta-x delta-y))
;;draw-and-clear:a-ball->true
;;draw, sleep, clear a disk from the canvas
;;structural design, Scheme knowledge
(define (draw-and-clear a-ball)
  (and
    (draw-solid-disk (make-posn (ball-x a-ball) (ball-y a-ball)) 5 'red)
    (sleep-for-a-while DELAY)
    (clear-solid-disk (make-posn (ball-x a-ball) (ball-y a-ball)) 5 'red)))
```

Welcome to DrScheme, version 103p1.
Language: Intermediate Student.
Teachpack: C:\Program Files\PLT\teachpack\htdp\draw.ss.

```
> (start 100 100)
> (draw-and-clear (make-ball 10 10 0 0))
true
>
```

1:36 Unlocked not running

ball を 1 秒描く

- (draw-and-clear (make-ball 10 10 0 0))

  の実行時に，「描画用ウインドウ」に，赤い円が現れて消えるので，確認すること

```
(define DELAY 1)
```

変数 DELAY の定義

```
(define-struct ball (x y delta-x delta-y))
```

ball 構造

```
;;draw-and-clear:a-ball->true
;;draw, sleep, clear a disk from the canvas
;;structural design, Scheme knowledge
(define (draw-and-clear a-ball)
  (and
    (draw-solid-disk (make-posn (ball-x a-ball) (ball-y a-ball)) 5 'red)
    (sleep-for-a-while DELAY)
    (clear-solid-disk (make-posn (ball-x a-ball) (ball-y a-ball)) 5 'red)))
```

関数 draw-and-clear

```
(strart 100 100)
(draw-and-clear (make-ball 10 10 0 0))
(stop)
```

draw-and-clear を使っている部分

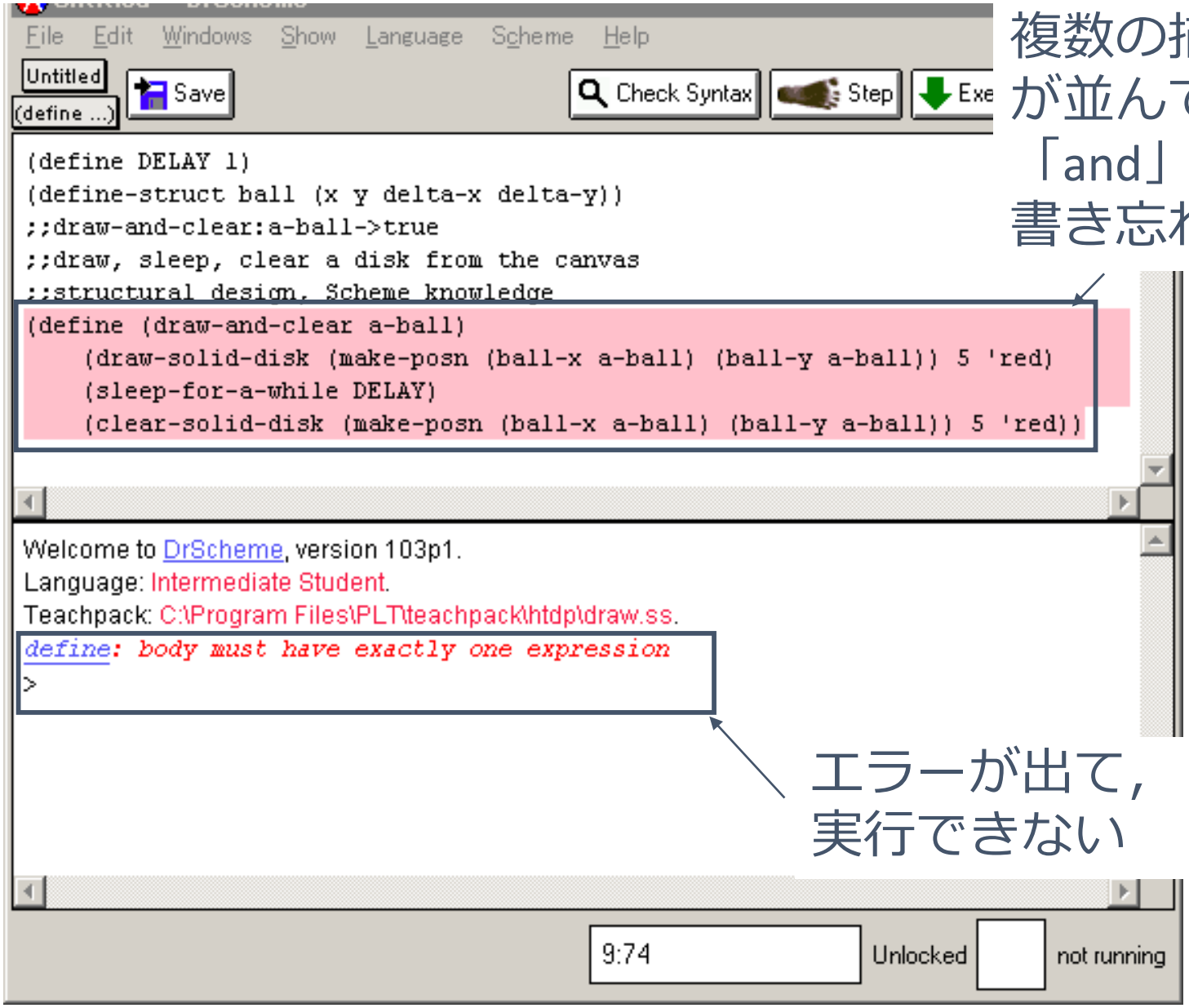
複数の描画命令を並べるときは「and」でつなぐ

```
(and
  (draw-solid-disk (make-posn (ball-x a-ball) (ball-y a-ball)) 5 'red)
  (sleep-for-a-while DELAY)
  (clear-solid-disk (make-posn (ball-x a-ball) (ball-y a-ball)) 5 'red))
```

3 つの描画命令

draw-solid-disk, sleep-for-a-while, clear-solid-disk を and でつないでいる

よくある間違い



複数の描画命令
が並んでいるのだが、
「and」を
書き忘れている

エラーが出て，
実行できない

例題９．動く ball を描く

- ball 構造体（授業の例題２）を使って，動くボールのアニメーションのプログラム animation を作る
  - ボールを動かすための関数 move-ball を作る
  - 動くボールのアニメーションの関数 animation は，move-ball と draw-and-clear（授業の例題８） を呼び出すと共に，animation（自分自身） を再帰的に呼び出す

「例題９．動く ball を描く」の手順 (1/2)

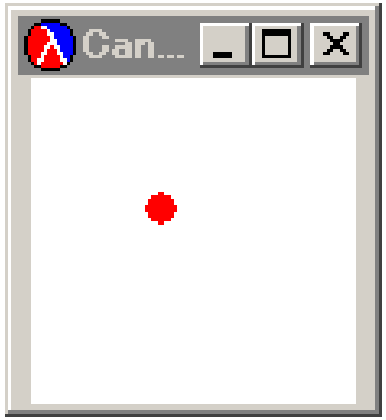
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Execute ボタンを押す

```
(define-struct ball
                (x y delta-x delta-y))
(define DELAY 1)
(define (draw-ball a-ball)
  (draw-solid-disk (make-posn
                (ball-x a-ball) (ball-y a-ball)) 5 'red))
(define (clear-ball a-ball)
  (clear-solid-disk (make-posn
                (ball-x a-ball) (ball-y a-ball)) 5))
 (define (move-ball a-ball)
   (make-ball
       (+ (ball-x a-ball) (ball-delta-x a-ball))
       (+ (ball-y a-ball) (ball-delta-y a-ball))
       (ball-delta-x a-ball) (ball-delta-y a-ball)))
(define (animation a-ball)
  (and
    (draw-ball a-ball)
    (sleep-for-a-while DELAY)
    (clear-ball a-ball)
    (animation (move-ball a-ball))))
```

「例題９．動く ball を描く」の手順 (2/2)

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(start 100 100)
(animation (make-ball 0 0 10 5))
(stop)
```



Untitled - DrScheme

File Edit Windows Show Language Scheme Help

Untitled (define ...) Save Check Syntax Step Execute Break

```
(define-struct ball
                      (x y delta-x delta-y))
(define DELAY 1)
(define (draw-ball a-ball)
  (draw-solid-disk (make-posn
                    (ball-x a-ball) (ball-y a-ball)) 5 'red))
(define (clear-ball a-ball)
  (clear-solid-disk (make-posn
                    (ball-x a-ball) (ball-y a-ball)) 5))
 (define (move-ball a-ball)
   (make-ball
        (+ (ball-x a-ball) (ball-delta-x a-ball))
        (+ (ball-y a-ball) (ball-delta-y a-ball))
        (ball-delta-x a-ball) (ball-delta-y a-ball)))
(define (animation a-ball)
  (and
    (draw-ball a-ball)
    (sleep-for-a-while DELAY)
    (clear-ball a-ball)
    (animation (move-ball a-ball))))
```

```
Welcome to DrScheme, version 103p1.
Language: Intermediate Student.
Teachpack: C:\Program Files\PLT\teachpack\htdp\draw.ss.
> (start 100 100)
> (animation (make-ball 0 0 10 5))
```

動く ball を描く

- 「描画用ウインドウ」に，赤い円が動く様子を確認すること
- 満足したら「(stop)」を実行して，「描画用ウインドウ」を閉じる

```
(define-struct ball
                (x y delta-x delta-y))
```

ball 構造

```
(define DELAY 1)
```

変数 DELAY の定義

```
(define (draw-ball a-ball)
  (draw-solid-disk (make-posn
                    (ball-x a-ball) (ball-y a-ball)) 5 'red))
```

draw-ball 関数

```
(define (clear-ball a-ball)
  (clear-solid-disk (make-posn
                     (ball-x a-ball) (ball-y a-ball)) 5))
```

clear-ball 関数

```
(define (move-ball a-ball)
  (make-ball
      (+ (ball-x a-ball) (ball-delta-x a-ball))
      (+ (ball-y a-ball) (ball-delta-y a-ball))
      (ball-delta-x a-ball) (ball-delta-y a-ball)))
```

move-ball 関数

```
(define (animation a-ball)
  (and
    (draw-ball a-ball)
    (sleep-for-a-while DELAY)
    (clear-ball a-ball)
    (animation (move-ball a-ball))))
```

animation 関数

描画命令と他の命令を並べるときは「and」でつなぐ

```
(define (animation a-ball)
  (and
    (draw-ball a-ball)
    (sleep-for-a-while DELAY)
    (clear-ball a-ball)
    (animation (move-ball a-ball))))
```

描画命令の１種である sleep-for-a-while と他の式を and でつないでいる

課題 7

- 複数のball を動かすプログラムの作成.
  - 次の関数 move-ball は，１つの ball を動かすプログラムである．速度からの位置の計算を行っている
  - これを参考にして，複数のballを動かす関数 move-all-balls を作成しなさい．move-all-balls の入力はball構造体のリストである.
  - 必ず動作確認まで行うこと.

```
(define (move-ball a-ball)
  (make-ball
      (+ (ball-x a-ball) (ball-delta-x a-ball))
      (+ (ball-y a-ball) (ball-delta-y a-ball))
      (ball-delta-x a-ball) (ball-delta-y a-ball)))
```

課題 8

- 課題 5 で作成したプログラムを変更して，全てのballが「描画用ウインドウ」の外にあれば「描画用ウインドウ」を閉じるようにしなさい.
- また，「描画用ウインドウ」の外にあるボールについては，draw-solid-disk を実行しないようにしなさい
  - 描画用ウインドウを閉じるには「(stop)」を用いる

課題９

- 例題９を参考にして，複数のballのアニメーションのプログラムを作成し，実行結果を報告しなさい
  - 作成したアニメーションプログラムが，全てのball が「描画用ウインドウ」の外に出たら，アニメーションが終わるようになっていることを確認すること
  - もし，アニメーションが終わらないのなら，必ず終わるように変更すること

課題１０

- 課題９についての問題
  - 動いている間に大きさや色が変るようにプログラムを変更しなさい

課題１１

- 課題９についての問題
  - 「描画用ウインドウ」の境界まで達したら，跳ね返りの角度や速度などを変えて，跳ね返るようにプログラムを変更しなさい