sp-10. 構造体

（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

10-1 構造体，構造体の定義，構造体の使用

10-2 パソコン演習

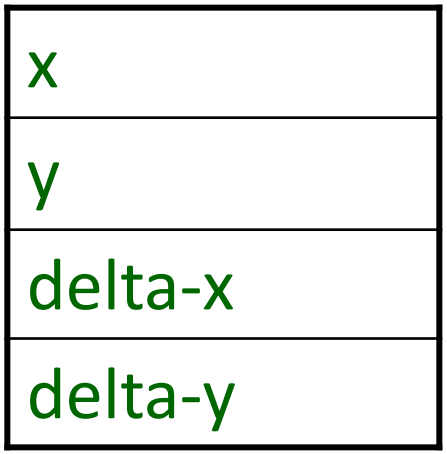
10-3 課題

10-1 構造体，構造体の定義，構造体の使用

構造体とは？

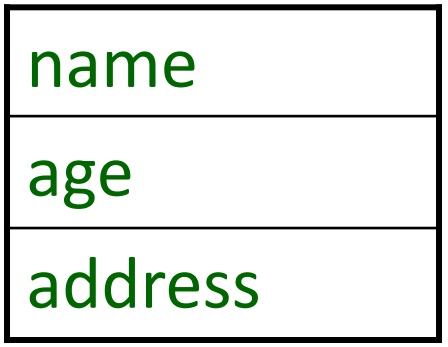
- 複数のデータが集まって，１つのデータを構成
- 新しい型の名前が付いている
- structure ともいう

構造体の例



ball

例題１
例題２



データの
集まり

AddressNote

型の名前

例題３
例題４

define-struct ・・・の機能

名前　　属性の並び

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

- 上記のプログラムの実行によって
  - コンストラクタ：　make-ball
  - セレクタ：　ball-x, ball-y, ball-delta-x, ball-delta-y

  が使えるようになる

コンストラクタとセレクタ

- コンストラクタ
  - 構造体の生成

    （例）　make-ball

- セレクタ
  - 属性値の取得

    （例）　ball-x, ball-y, ball-delta-x, ball-delta-y

10-2 パソコン演習

パソコン演習の進め方

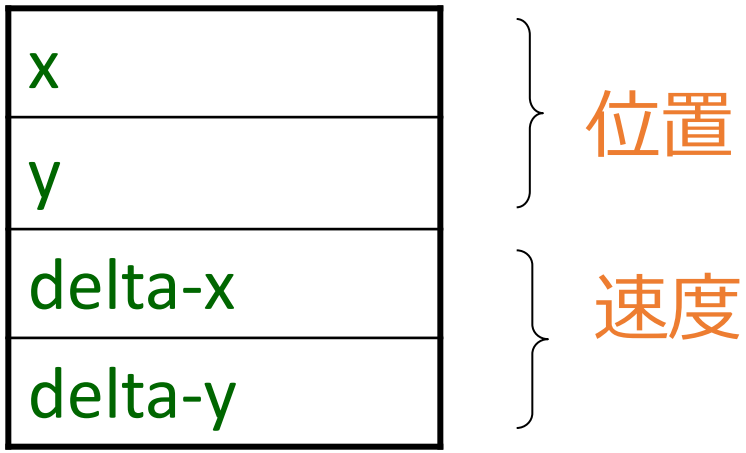
- 資料を見ながら，「例題」を行ってみる
- 各自，「課題」に挑戦する
- 自分のペースで先に進んで構いません

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

例題１． ball 構造体

- ball 構造体(ball structure）を定義するプログラムを書く
  - ball は，x, y, delta-x, delta-y から構成する
  - define-struct 文を使用



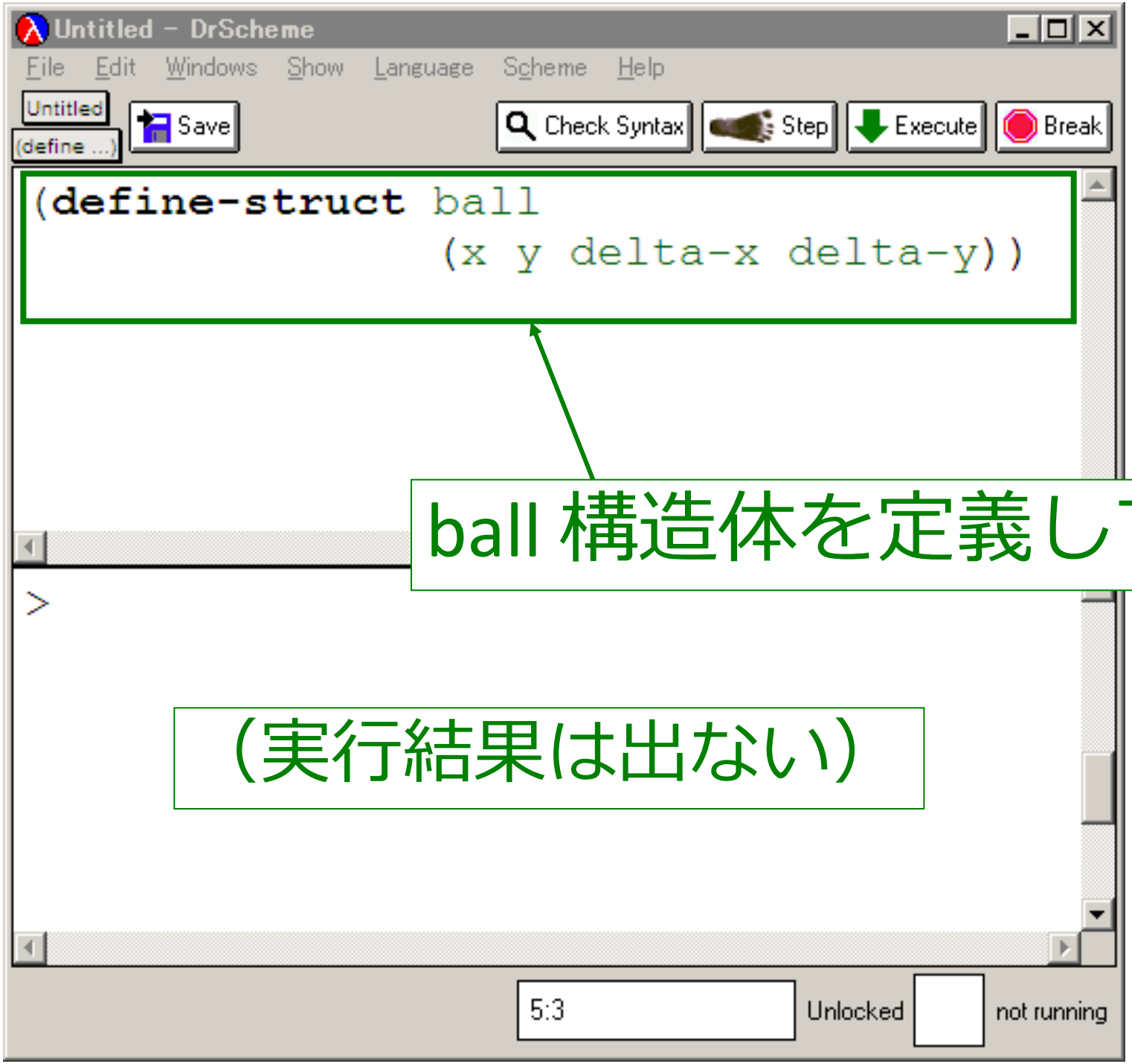
「例題１．ball 構造体」の手順

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))
```

⇒（これは，例題２，３で使います）

☆　次は，例題２に進んでください



ball 構造体を定義している

（実行結果は出ない）

ball 構造体

名前

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

属性の並び
（それぞれの属性にも名前がある）

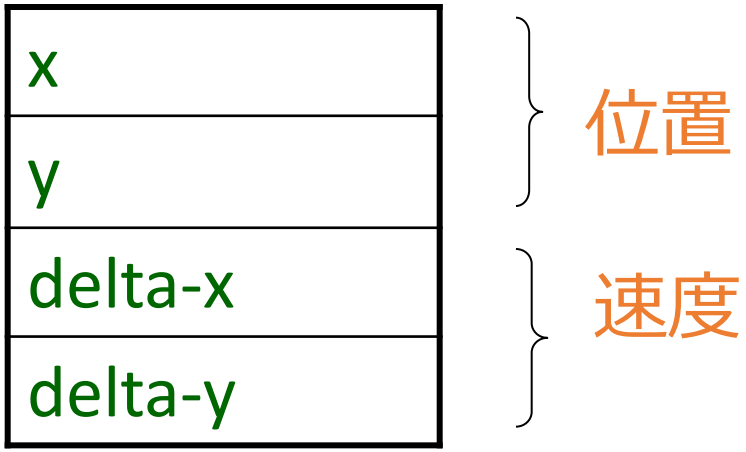
コンストラクタとセレクタ

- 上記の定義によって
  - コンストラクタ：　make-ball
  - セレクタ：　ball-x, ball-y, ball-delta-x, ball-delta-y

  が使えるようになる

例題２． ball の原点からの距離

- ボールの座標値から，原点からの距離を求める関数 distance-to-0 を作り，実行する
  - 例題１の ball 構造体を使用
  - 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 の原点からの距離」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

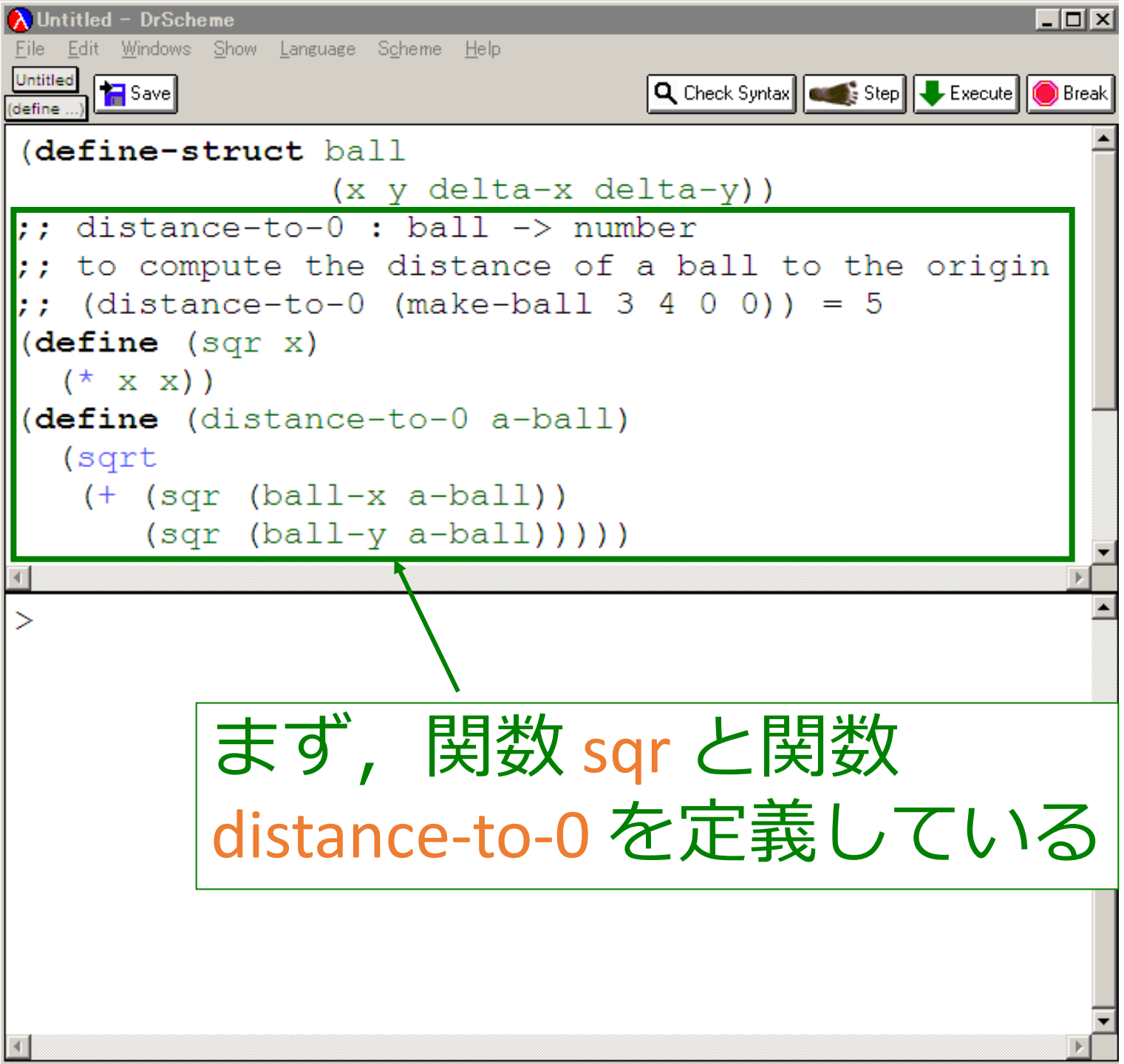
```
(define-struct ball
   (x y delta-x delta-y))
;; distance-to-0: ball -> number
;; to compute the distance of a ball to the origin
;; (distance-to-0 (make-ball 3 4 0 0)) = 5
(define (sqr x)
  (* x x))
(define (distance-to-0 a-ball)
   (sqrt
     (+ (sqr (ball-x a-ball))
        (sqr (ball-y a-ball)))))
```

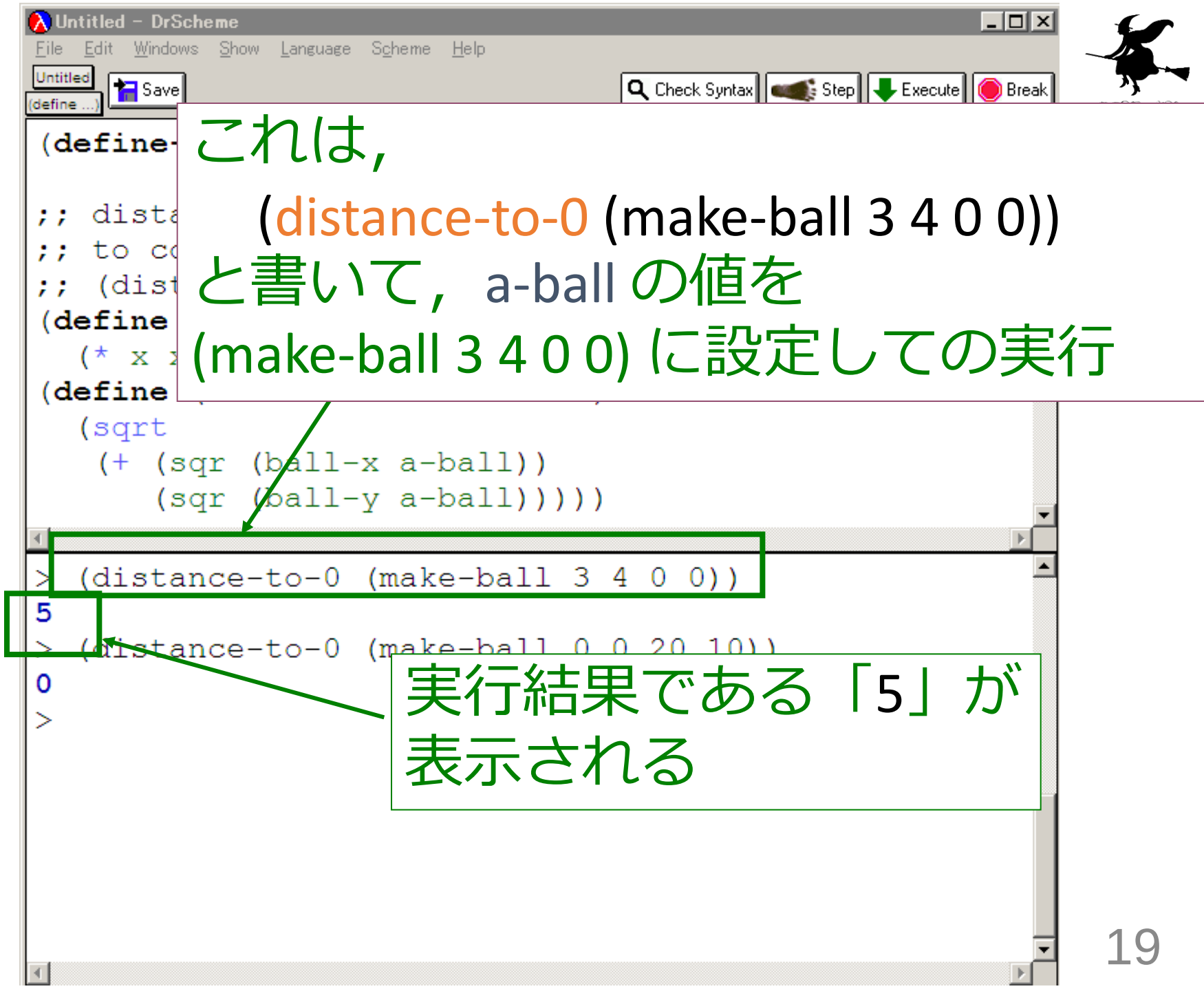
例題１と同じ

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(distance-to-0  (make-ball 3 4 0 0))
```

☆　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))
;; distance-to-0 : ball -> number
;; to compute the distance of a ball to the origin
;; (distance-to-0 (make-ball 3 4 0 0)) = 5
(define (sqr x)
(* x x))
(define (distance-to-0 a-ball)
(sqrt
(+ (sqr (ball-x a-ball))
(sqr (ball-y a-ball)))))
>
まず，関数 sqr と関数
distance-to-0 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-
;; dista
;; to c
;; (dist
(define
(* x x
(define
(sqrt
(+ (sqr (ball-x a-ball))
(sqr (ball-y a-ball)))))
> (distance-to-0 (make-ball 3 4 0 0))
5
> (distance-to-0 (make-ball 0 0 20 10))
0
>
これは,
(distance-to-0 (make-ball 3 4 0 0))
と書いて，a-ball の値を
(make-ball 3 4 0 0) に設定しての実行
実行結果である「5」が
表示される

distance-to-0 の入力と出力

a-ball の値：

(make-ball 3 4 0 0)

入力は ball 構造体

出力は数値


```
(define-struct ball (x y delta-x delta-y))
;; distance-to-0 : ball -> number
;; to compute the distance of a ball to the origin
;; (distance-to-0 (make-ball 3 4 0 0)) = 5
(define (sqr x)
  (* x x))
(define (distance-to-0 a-ball)
   (sqrt
      (+ (sqr (ball-x a-ball))
           (sqr (ball-y a-ball)))))
```

```
(define-struct ball (x y delta-x delta-y))
;; distance-to-0 : ball -> number
;; to compute the distance of a ball to the origin
;; (distance-to-0 (make-ball 3 4 0 0)) = 5
(define (sqr x)
  (* x x))
(define (distance-to-0 a-ball)
   (sqrt
      (+ (sqr (ball-x a-ball))
         (sqr (ball-y a-ball)))))
```

「ball-x」は，x の値の取得

「ball-y」は，x の値の取得

例題３．ステップ実行

- 関数 distance-to-0 （例題２）について，実行結果に至る過程を見る
  - (distance-to-0 (make-ball 3 4 0 0)) から 5 に至る過程を見る
  - DrScheme の stepper を使用する

```
(distance-to-0 (make-ball 3 4 0 0))
= (sqrt (+ (sqr (ball-x (make-ball 3 4 0 0)))
           (sqr (ball-y (make-ball 3 4 0 0)))))
= (sqrt (+ (sqr 3)
           (sqr (ball-y (make-ball 3 4 0 0)))))
= ...
= (sqrt (+ 9
           (sqr (ball-y (make-ball 3 4 0 0)))))
= (sqrt (+ 9
           (sqr 4)))
= ...
= (sqrt (+ 9 16))
= (sqrt 25)
= 5
```

「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

```
(define-struct ball
   (x y delta-x delta-y))
;; distance-to-0: ball -> number
;; to compute the distance of a ball to the origin
;; (distance-to-0 (make-ball 3 4 0 0)) = 5
(define (sqr x)
  (* x x))
(define (distance-to-0 a-ball)
   (sqrt
      (+ (sqr (ball-x a-ball))
         (sqr (ball-y a-ball)))))
(distance-to-0 (make-ball 3 4 0 0))
```

← 例題２と同じ

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４に進んでください

(distance-to-0 (make-ball 3 4 0 0)) から 5 に至る過程の概略

(distance-to-0 (make-ball 3 4 0 0)) 最初の式

```
= (sqrt (+ (sqr (ball-x (make-ball 3 4 0 0)))
           (sqr (ball-y (make-ball 3 4 0 0)))))
= (sqrt (+ (sqr 3)
           (sqr (ball-y (make-ball 3 4 0 0)))))
= ...
= (sqrt (+ 9
           (sqr (ball-y (make-ball 3 4 0 0)))))
= (sqrt (+ 9
           (sqr 4)))
= ...
= (sqrt (+ 9 16))
= (sqrt 25)
```

コンピュータ内部での計算

= 5 実行結果

(distance-to-0 (make-ball 3 4 0 0)) から 5 に至る過程の概略

(distance-to-0 (make-ball 3 4 0 0))

= (sqrt (+ (sqr (ball-x (make-ball 3 4 0 0)))
(sqr (ball-y (make-ball 3 4 0 0)))))

= (sqrt (+ (sqr 3)
(sqr (ball-y (make-ball 3 4 0 0)))))

= .

= (

= (

= .

= (

= (sqrt 25)

= 5

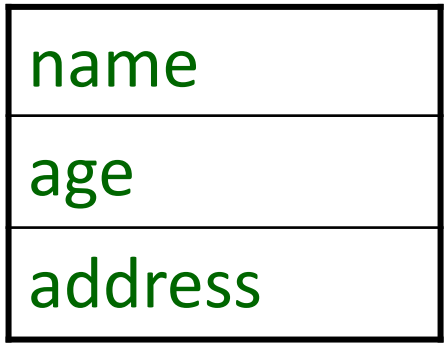
これは,

```
(define (distance-to-0 a-ball)
  (sqrt
    (+ (sqr (ball-x a-ball))
       (sqr (ball-y a-ball)))))
```

の a-ball を (make-ball 3 4 0 0) で置き換えたもの

例題４．AddressNote 構造体

- AddressNote 構造体を定義するプログラムを書く
  - AddressNote は，name, age, address から構成する
  - define-struct 文を使用



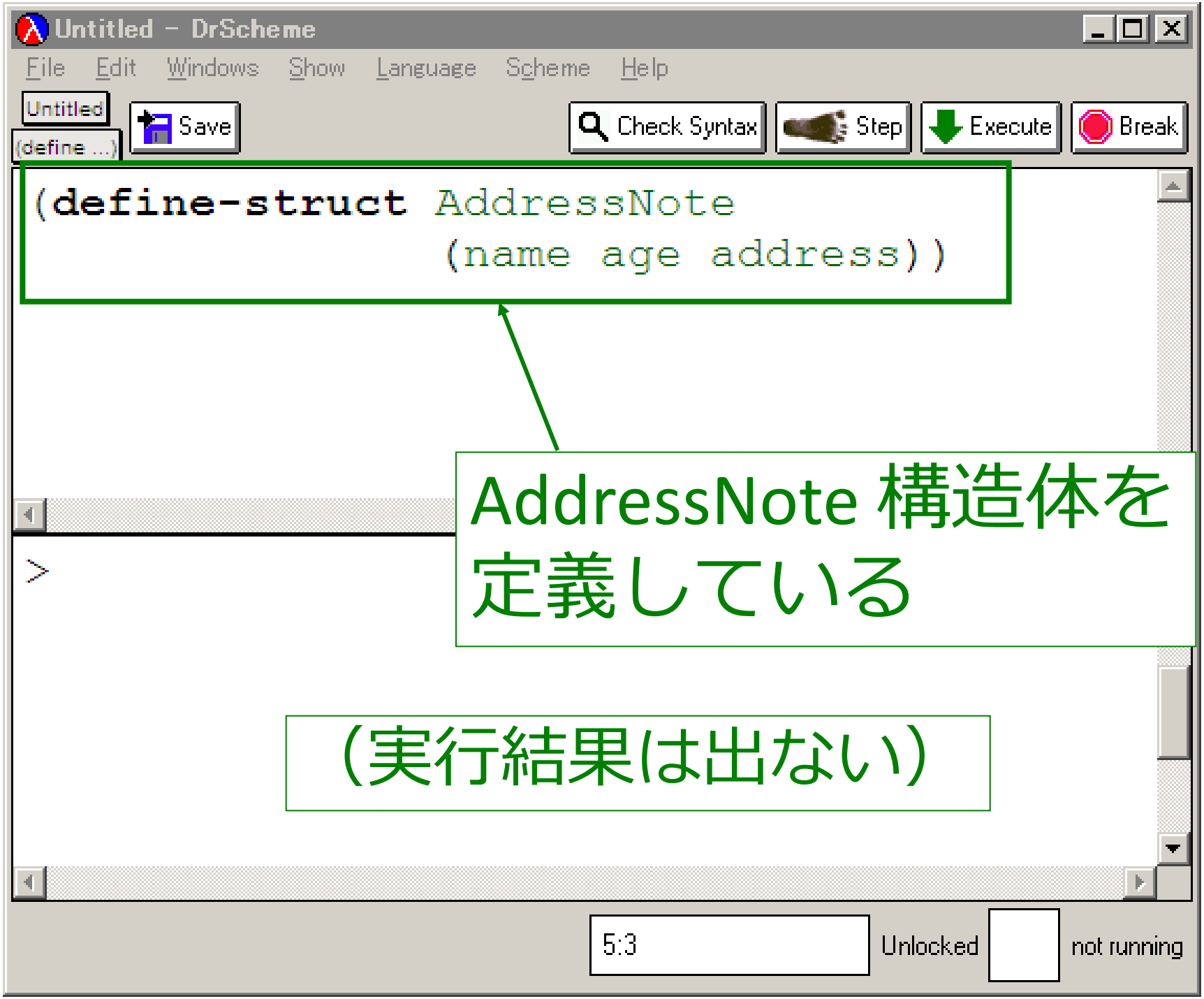
「例題４． ball 構造体」の手順

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 AddressNote
                (name age address))
```

⇒ （これは，例題５で使います）

☆　次は，例題５に進んでください



AddressNote 構造体を定義している

（実行結果は出ない）

AddressNote 構造体

名前

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

属性の並び
（それぞれの属性にも名前がある）

コンストラクタとセレクタ

- 上記のプログラムの実行によって
  - make-AddressNote
  - AddressNote-name, AddressNote-age, AddressNote-address

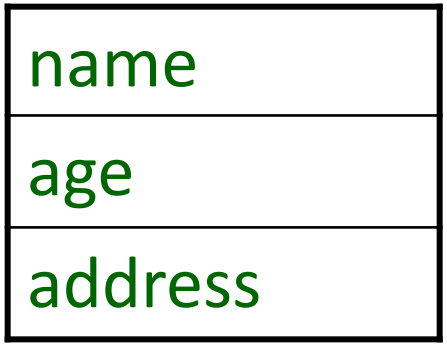
  が使えるようになる

例題５．住所録

- AddressNote 構造体のリストから，名前（name）のリストを得る関数 select-name を作り，実行する
  - 例題３の AddressNote 構造体を使用

    構造体１つ　＝　１人分

    構造体のリスト　＝　複数人
  - 属性 name を取り出すために AddressNote-name（セレクタ）を使う



AddressNote

「例題５．住所録」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 AddressNote
          (name age address))
```

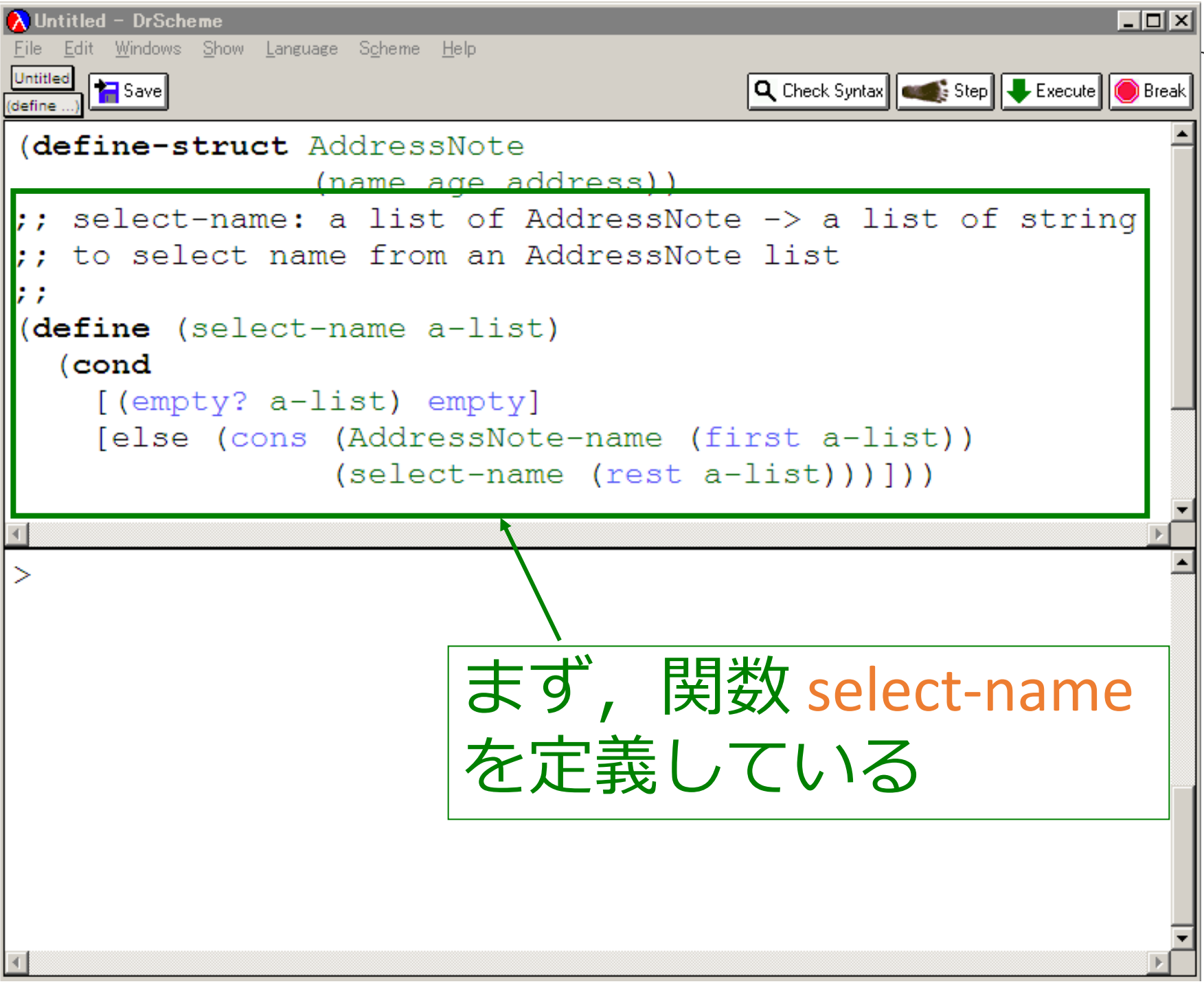
← 例題４と同じ

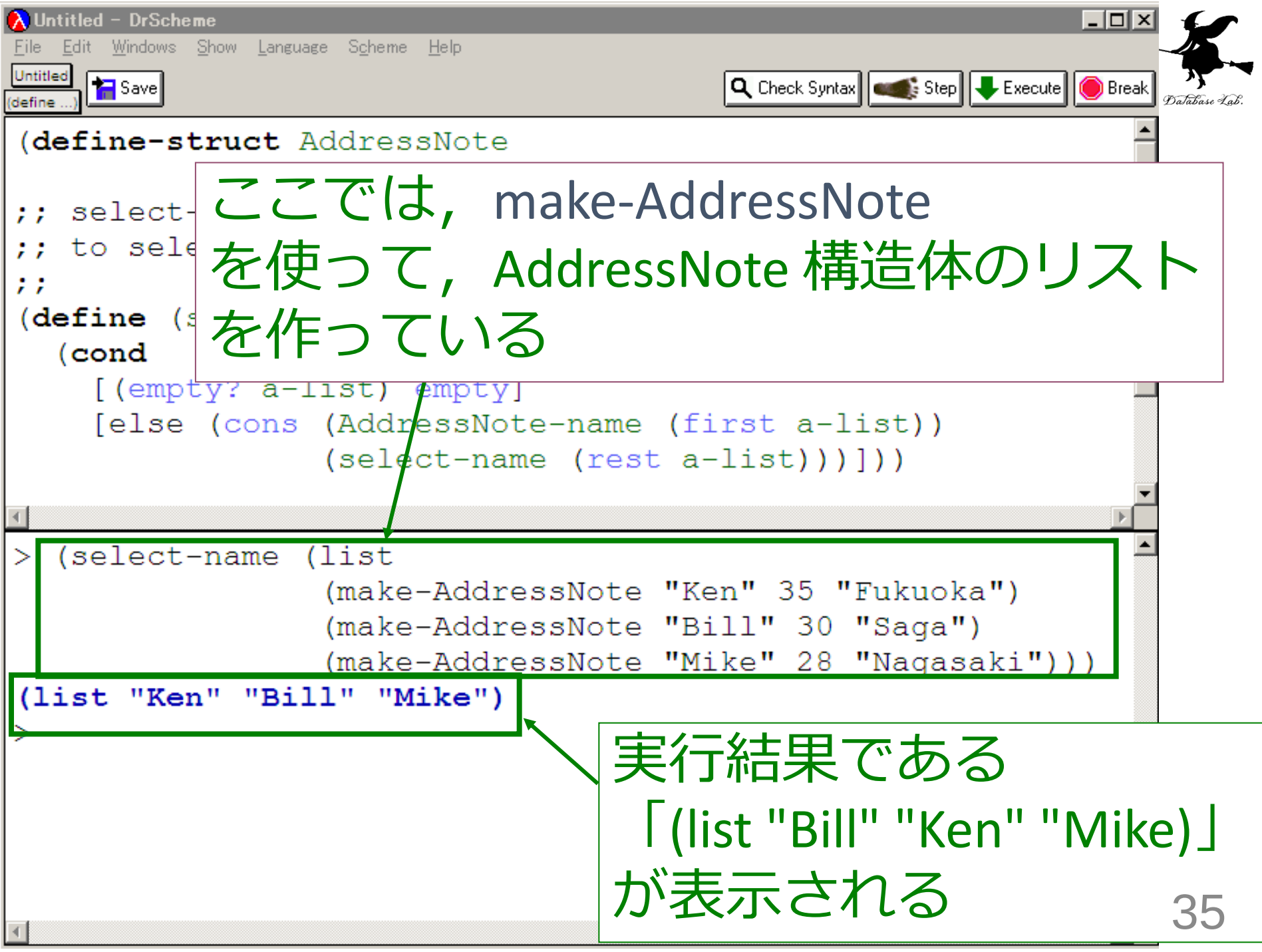
```
;; select-name: a list of AddressNote -> a list of string
;; to select name from an AddressNote list
(define (select-name a-list)
  (cond
    [(empty? a-list) empty]
    [else (cons (AddressNote-name (first a-list))
                (select-name (rest a-list)))]))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(select-name (list
          (make-AddressNote "Ken" 35 "Fukuoka")
          (make-AddressNote "Bill" 30 "Saga")
          (make-AddressNote "Mike" 28 "Nagasaki")))
```

☆　次は，例題６に進んでください

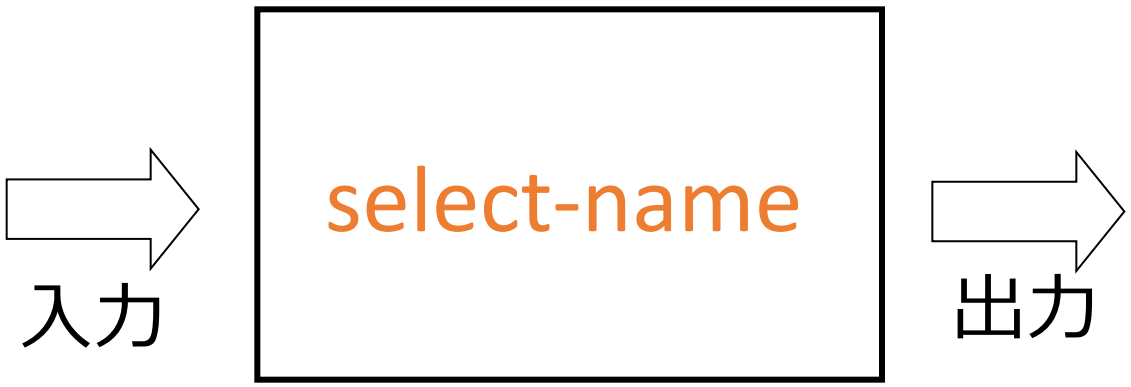

Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Save
Check Syntax
Step
Execute
Break
(define-struct AddressNote
(name age address))
;; select-name: a list of AddressNote -> a list of string
;; to select name from an AddressNote list
;;
(define (select-name a-list)
(cond
[(empty? a-list) empty]
[else (cons (AddressNote-name (first a-list))
(select-name (rest a-list)))]))
>
まず，関数 select-name 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 AddressNote
;; select-
;; to sele
;;
(define (s
(cond
[(empty? a-list) empty]
[else (cons (AddressNote-name (first a-list))
(select-name (rest a-list)))]))
ここでは，make-AddressNote
を使って，AddressNote 構造体のリスト
を作っている
> (select-name (list
(make-AddressNote "Ken" 35 "Fukuoka")
(make-AddressNote "Bill" 30 "Saga")
(make-AddressNote "Mike" 28 "Nagasaki")))
(list "Ken" "Bill" "Mike")
>
実行結果である
「(list "Bill" "Ken" "Mike)」
が表示される

select-name の入力と出力

```
(list
  (make-AddressNote "Ken" 35 "Fukuoka")
  (make-AddressNote "Bill" 30 "Saga")
  (make-AddressNote "Mike" 28 "Nagasaki"))
```

```
(list "Ken" "Bill" "Mike")
```



入力は AddressNote 構造体のリスト

出力はリスト

```
(define-struct AddressNote
                (name age address))
;; select-name: a list of AddressNote -> a list of string
;; to select name from an AddressNote list
(define (select-name a-list)
  (cond
    [(empty? a-list) empty]
    [else (cons (AddressNote-name (first a-list))
                (select-name (rest a-list)))]))
```

select-name の実行では

```
(select-name (list
        (make-AddressNote "Ken" 35 "Fukuoka")
        (make-AddressNote "Bill" 30 "Saga")
        (make-AddressNote "Mike" 28 "Nagasaki")))
```

- select-name の入力は　⇒　リスト
- AddressNote 構造体のリストを作るために，
  make-AddressNote（コンストラクタ） を並べている

例題６．複素数の計算

- DrScheme にすでに組み込み済みの構造体 rectangular を使って，複素数の計算を行ってみる

足し算	(1+2i) + (3+4i)
引き算	(5+7i) – (3+4i)
かけ算	(1+2i) ×(3+4i)
割り算	(-5+10i) / (3+4i)

複素数の計算

DrScheme では　⇒　make-rectangular を使用

実数部が x で，虚数部が y であるような複素数は，
(make-rectangular x y)

例：　「(make-rectangular 3 4)」は 「3+4*i*」

「rectangular」　は，DrScheme に組み込み済みの構造体

よくある間違い

```
> (+ (1+2i) (3+4i))
illegal application: first term in application
must be a function name
```

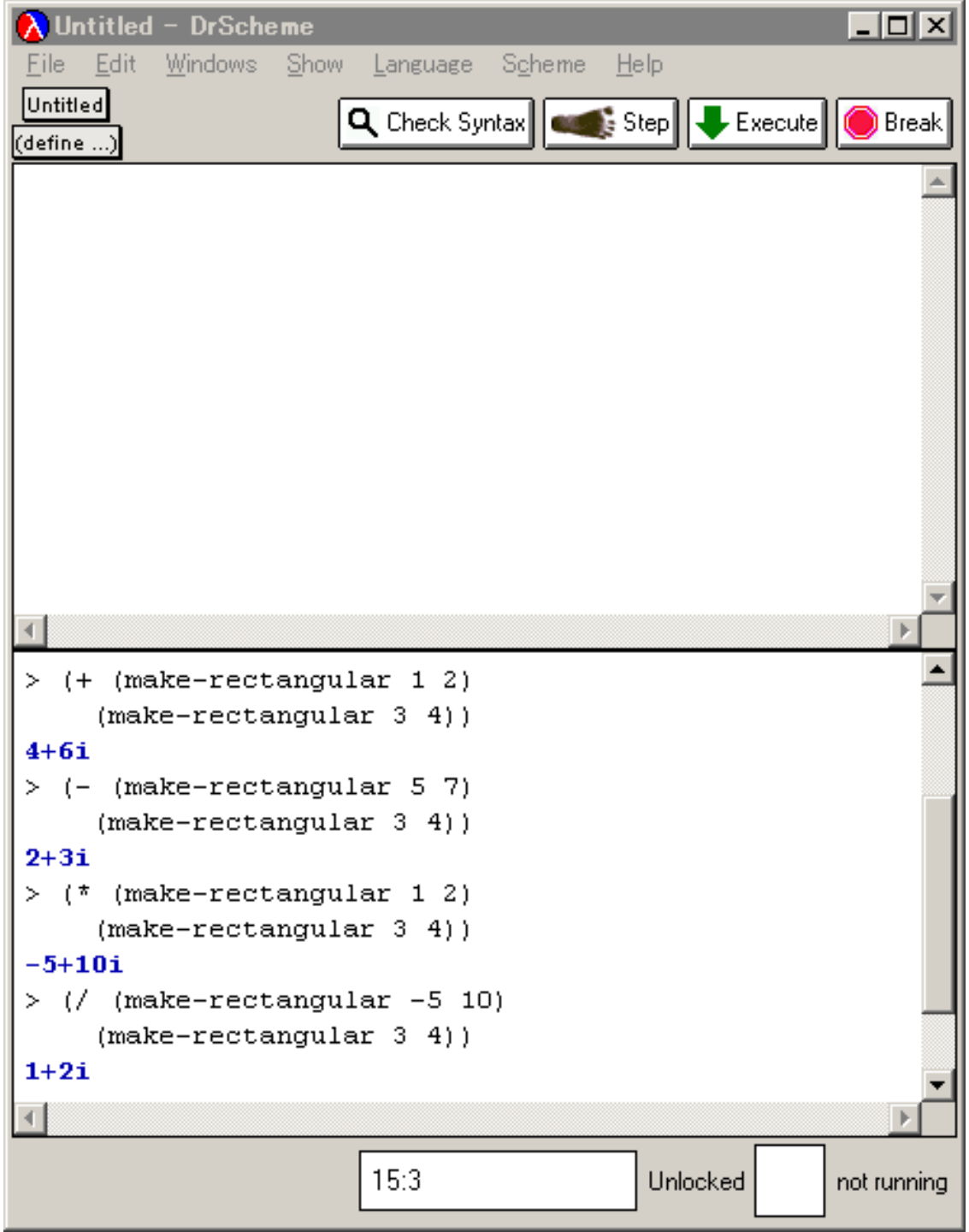
(+ (1+2i) (3+4i)) は，シンタックスエラー（文法エラー）
→　(+ (make-rectangular 1 2) (make-rectangular 3 4)) と書くべき

「例題６．複素数の計算」の手順

1. 次の式を「実行用ウインドウ」で，実行しなさい

```
(+ (make-rectangular 1 2)
   (make-rectangular 3 4))
(- (make-rectangular 5 7)
   (make-rectangular 3 4))
(* (make-rectangular 1 2)
   (make-rectangular 3 4))
(/ (make-rectangular -5 10)
   (make-rectangular 3 4))
```

☆　次は，例題７に進んでください


Untitled - DrScheme
File Edit Windows Show Language Scheme Help
Untitled
(define ...)
Check Syntax
Step
Execute
Break
> (+ (make-rectangular 1 2)
(make-rectangular 3 4))
4+6i
> (- (make-rectangular 5 7)
(make-rectangular 3 4))
2+3i
> (* (make-rectangular 1 2)
(make-rectangular 3 4))
-5+10i
> (/ (make-rectangular -5 10)
(make-rectangular 3 4))
1+2i
15:3
Unlocked
not running

例題７．複素数のべき乗

- ２つの値 theta と n から $(\cos\theta + i \sin\theta)^n$ を計算するプログラム myexp を作り，実行する
  - θはの単位はラジアン
  - $(\cos\theta + i \sin\theta)^n$ の計算: expt を使用

    →　 expt は複素数にも使える

「例題７．複素数のべき乗」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

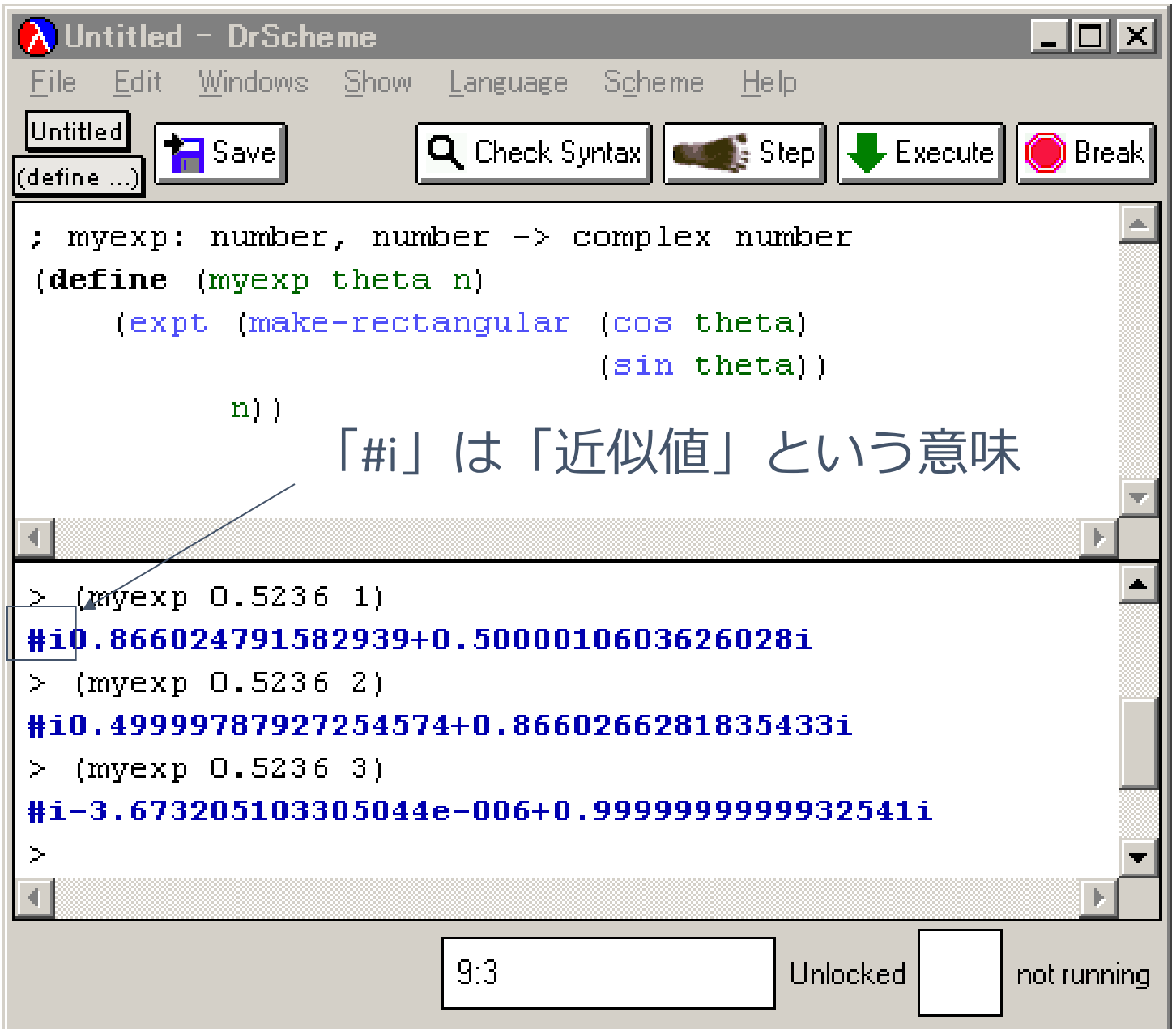
```
(define (myexp theta n)
   (expt (make-rectangular (cos theta)
                           (sin theta))
         n))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(myexp 0.5236 1)
(myexp 0.5236 2)
(myexp 0.5236 3)
```

☆　次は，課題に進んでください

「例題７．複素数のべき乗」の実行結果



10-3 課題

課題１

- 関数 distance-to-0 （授業の例題２）についての問題
  - 次の式を実行し，実行結果を報告せよ

1. (distance-to-0 (make-posn 1 2))
2. (distance-to-0 (make-posn (- 5 3) (- 4 6))
3. (distance-to-0 (make-posn (* 2 3) (* 4 5))

課題 2

- ball 構造体（授業の例題 1 ）についての問題
  - ball のデータ a-ball の x の値が 0 と 100 の間にあるときに限り true を返し，範囲外 のときには false を返すような関数 in を作成し，実行結果を報告しなさい
  - 但し，x = 0 あるいは x = 100 のときには false を返すこと.
  - ヒント：　次の空欄を埋めなさい

```
(define-struct ball (x y delta-x delta-y))
(define (in a-ball)
  (cond
    [                                    ]
    [ else                                  ]))
```

課題３

- AddressNote 構造体（授業の例題４）についての問題
  - AddressNote のデータ a-person の氏名を取り出す関数 get-name は，セレクタ AddressNote-name を使って，次のように書ける.

```
(define (get-name a-person)
        address-note-name a-person)
```

  - では，AddressNote のデータ a-person の年齢(age)が２０以上ならば「'Adult」を，２０以下なら「'Clild」を出力する関数を作成し，実行結果を報告しなさい

課題４

- AddressNote 構造体（授業の例題４）についての問題
  - a-person の年齢が、ある年齢 an-age と一致していれば名前（name）を、さもなければ，文字列 "none" を返す関数 check-by-age を作成し，実行結果を報告しなさい

    例えば

    (check-by-age (make-address-note "Kunihiko Kaneko" 35 "Hakozaki") 35) = "Kunihiko Kaneko"

    (check-by-age (make-address-note "Kunihiko Kaneko" 35 "Hakozaki") 30) = "none"

課題５

- AddressNote 構造体（授業の例題４）についての問題
  - AddressNote 構造体のリスト a-list と年齢 an-age から，年齢が an-age に一致する人のリストを出力する関数 selection-by-age を作成し，実行結果について報告しなさい

例えば

```
(selection-by-age
   (list (make-address-note “Kunihiko Kaneko” 35 “Hakozaki”)
         (make-address-note “Taro Tanaka” 34 “Kaizuka”)
         (make-address-note “Hanako Saito” 35 “Tenjin”)) 35)
= (list (make-address-note “Kunihiko Kaneko” 35 “Hakozaki”)
        (make-address-note “Hanako Saito” 35 “Tenjin”))
```

課題６

- AddressNote 構造体のリストから，年齢(age)の平均を求める関数 sum-age を作り，実行結果を報告しなさい

さらに勉強したい人への補足説明資料

- ドモアブルの定理

ドモアブルの定理

- n = 1, 2, …, 20 について，$(\cos\theta + i \sin\theta)^n$ と cos nθ+ i sin nθを計算するプログラム　f1, f2 を書く
  - 結果は，長さ２０のリストとして得る
  - 近似値を使った計算を繰り返すと，誤差が積み重なることを確かめる

    → $(\cos\theta + i \sin\theta)^n$ は，(cosθ+ i sinθ)の値（これは近似値）を使っての計算

「ド・モアブルの定理」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

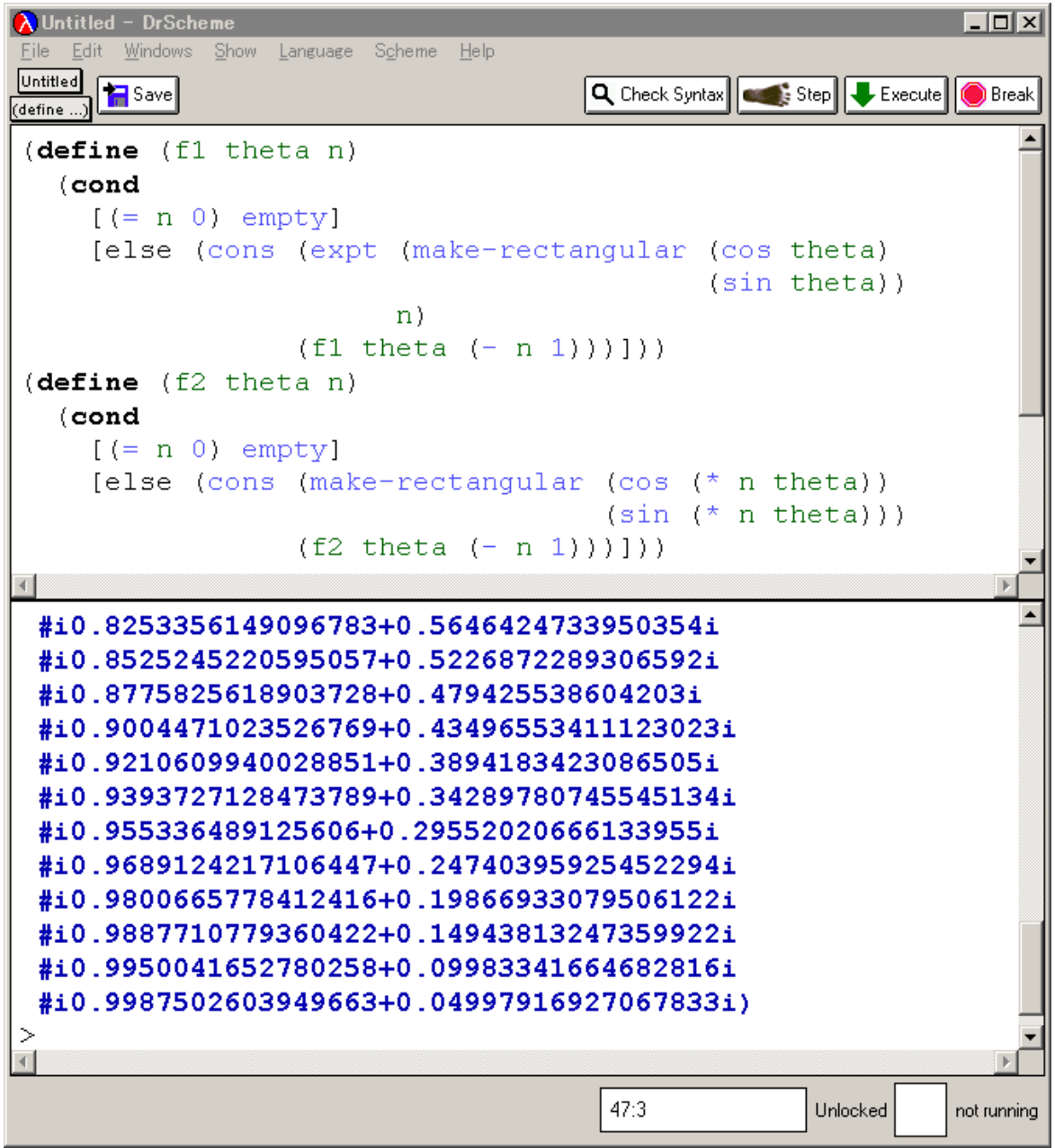
```
(define (f1 theta n)
   (cond
      [(= n 0) empty]
      [else (cons (expt (make-rectangular (cos theta)
                                          (sin theta))
                        n)
                  (f1 theta (- n 1)))]))
(define (f2 theta n)
   (cond
      [(= n 0) empty]
      [else (cons (make-rectangular (cos (* n theta))
                                    (sin (* n theta)))
                  (f2 theta (- n 1)))]))
```

「ド・モアブルの定理」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

```
(f1 0.05 20)
(f2 0.05 20)
```

「ド・モアブルの定理」の実行結果



実行結果の比較

$(\cos\theta + i \sin\theta)^n$

```
> (f1 0.05 20)
(list
 #i0.5403023058681402+0.8414709848078972i
 #i0.5816830894638839+0.8134155047893742i
 #i0.6216099682706648+0.7833269096274839i
 #i0.6599831458849826+0.7512804051402932i
 #i0.6967067093471658+0.7173560908995231i
 #i0.731688868873821+0.6816387600233345i
 #i0.7648421872844886+0.6442176872376914i
 #i0.796083798549056+0.6051864057360399i
 #i0.8253356149096784+0.5646424733950356i
 #i0.8525245220595062+0.5226872289306593i
 #i0.8775825618903731+0.47942553860420317i
 #i0.9004471023526772+0.43496553411123035i
 #i0.9210609940028853+0.3894183423086506i
 #i0.939372712847379+0.34289780745545145i
 #i0.9553364891256061+0.29552020666133966i
 #i0.968912421710645+0.24740395925452296i
 #i0.9800665778412417+0.19866933079506124i
 #i0.9887710779360424+0.14943813247359924i
 #i0.9950041652780258+0.09983341664682816i
 #i0.9987502603949663+0.04997916927067833i)
```

$\cos n\theta + i \sin n\theta$

```
> (f2 0.05 20)
(list
 #i0.5403023058681398+0.8414709848078965i
 #i0.5816830894638836+0.8134155047893737i
 #i0.6216099682706644+0.7833269096274834i
 #i0.6599831458849822+0.7512804051402927i
 #i0.6967067093471654+0.7173560908995228i
 #i0.7316888688738209+0.6816387600233341i
 #i0.7648421872844885+0.644217687237691i
 #i0.7960837985490559+0.6051864057360396i
 #i0.8253356149096783+0.5646424733950354i
 #i0.8525245220595057+0.5226872289306592i
 #i0.8775825618903728+0.479425538604203i
 #i0.9004471023526769+0.43496553411123023i
 #i0.9210609940028851+0.3894183423086505i
 #i0.9393727128473789+0.34289780745545134i
 #i0.95336489125606+0.29552020666133955i
 #i0.9689124217106447+0.24740395925452294i
 #i0.9800665778412416+0.19866933079506122i
 #i0.9887710779360422+0.14943813247359922i
 #i0.9950041652780258+0.09983341664682816i
 #i0.9987502603949663+0.04997916927067833i)
```

不一致

値は、ほぼ一致しているが，わずかに食い違う

ド・モアブルの定理

- ド・モアブルの定理は：

$$(\cos\theta + i\sin\theta)^n$$

$$= \cos n\theta + i\sin n\theta$$

  - なお，i は虚数単位

$(\cos\theta + i\ \sin\theta)^n = \cos n\theta + i \sin n\theta$

$(\cos\theta + i\ \sin\theta)^2 = \cos^2\theta - \sin^2\theta + 2i \cos\theta\sin\theta$

$= \cos 2\theta + i \sin 2\theta$

$(\cos\theta + i\ \sin\theta)^3 = (\cos\theta + i\ \sin\theta)^2 (\cos\theta + i\ \sin\theta)$

$= (\cos 2\theta + i \sin 2\theta) (\cos\theta + i\ \sin\theta)$

$= \cos 2\theta\cos\theta - \sin 2\theta\sin\theta$

$+ i\ (\cos 2\theta\sin\theta - \sin 2\theta\cos\theta)$

$= \cos (2\theta+\theta) + i \sin (2\theta+\theta)$

$= \cos 3\theta + i \sin 3\theta$

(以下同様に考える．数学的帰納法で証明できる)

複素数の掛け算

$z_1 = x_1 + i\,y_1$

$z_2 = x_2 + i\,y_2$ のとき

$$z_1 z_2 = (x_1 + i\,y_1)(x_2 + i\,y_2)$$

$$= x_1x_2 + x_1 i\,y_2 + i\,y_1x_2 + i\,y_1 i\,y_2$$

$$= \underbrace{x_1x_2 - y_1y_2}_{\text{実数部}} + \underbrace{i\,(x_1y_2 + y_1x_2)}_{\text{虚数部}}$$

実際の計算では

- 本来なら「 $(\cos\theta + i \sin\theta)^n = \cos n\theta + i \sin n\theta$ 」が成り立つはず

- しかし，ここで実行される計算は，あくまでも近似計算

  sin, cos, log 等の計算結果は，近似値でしか得られない

  例：「(sin 0.1)」を実行すると →

  #i0.09983341664682816

- 計算を繰り返す（つまり、計算結果を使った計算）と，誤差が積み重なる
  - $(\cos\theta + i \sin\theta)^n$ は，$(\cos\theta + i \sin\theta)$の値（これは近似値）を使っての計算