

Computer Science I

Professor Tom Ellman
Lecture 3

A Pattern May Occur Over and Over

```
Welcome to DrScheme
> (* 8 8)
64
> (* 12 12)
144
> (* 7 7)
49
> (* 16 16)
256
```

What is the Pattern?

```
Welcome to DrScheme
> (* <something> <something>)
<something-squared>
```

Another Pattern May Be Repeated

```
Welcome to DrScheme
> (/ (+ 22 48) 2)
35
> (/ (+ 91 101) 2)
96
> (/ (+ 3 27) 2)
15
```

What is the Pattern?

```
Welcome to DrScheme
> (/ (+ <thing1> <thing2>) 2)
<average-thing1-&-thing2>
```

Procedural Abstraction

- Captures a pattern in expressions that occur over and over.
- Uses the same “Define” mechanism that we saw earlier.
- Along with a special notation for expressing patterns.

Defining the “Square” Procedure

Welcome to DrScheme

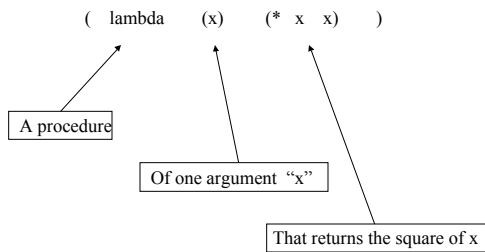
```
> (define square (lambda (x) (* x x)))  
  
> (square 8)  
64  
> (square 12)  
144  
> (square 7)  
49
```

Defining the “Average” Procedure

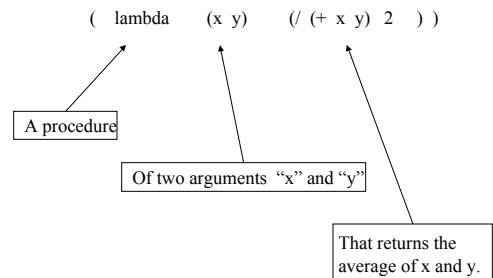
Welcome to DrScheme

```
> (define average (lambda (x y) (/ (+ x y) 2)))  
  
> (average 22 48)  
35  
> (average 91 101)  
96  
> (average 3 27)  
15
```

The Meaning of a “Lambda” Expression



The Meaning of a “Lambda” Expression



General form of Procedure Definitions

```
(define <variable> (lambda (<arguments>) <expression> ))
```

Example:

```
(define square (lambda (x) (* x x) ))
```

Example:

```
(define average (lambda (x y) (/ (+ x y) 2) ))
```

Argument Names Carry No Meaning

These Expressions Define the Same Procedure:

```
(define square (lambda (x) (* x x) ))
```

```
(define square (lambda (fred) (* fred fred) ))
```

```
(define square (lambda (ethel) (* ethel ethel) ))
```

```
(define square (lambda (lucy) (* lucy lucy) ))
```

Argument Names Carry No Meaning

These Expressions Define the Same Procedure:

```
(define average (lambda (x y) (/ (+ x y) 2)))  
  
(define average (lambda (romeo juliet) (/ (+ romeo juliet) 2)))  
  
(define average (lambda (jekyll hyde) (/ (+ jekyll hyde) 2)))  
  
(define average (lambda (clinton starr) (/ (+ clinton starr) 2)))
```

Argument Names Must be Unique

```
(define average (lambda (x x) (/ (+ x x) 2)))
```

- This procedure definition is syntactically incorrect.
- An attempt to process this definition will result in an error message.

A Lambda Expression Has a Value (Just Like Any Other Expression)

Welcome to DrScheme

```
> (lambda (x) (* x x))  
#<procedure>  
  
> (define square (lambda (x) (* x x)))  
> square  
#<procedure:square>
```

Making a Single Item list

Welcome to DrScheme

```
> (define embed (lambda (x) (cons x '())))  
  
> (embed 'fred)  
(fred)  
  
> (embed (embed 'fred))  
((fred))
```

Increment and Decrement

```
Welcome to DrScheme  
> (define increment (lambda (x) (+ x 1)))  
> (define decrement (lambda (x) (- x 1)))  
> (increment 0)  
1  
> (increment 1)  
2  
> (decrement 2)  
1  
> (decrement 1)  
0
```

Double and Half

```
Welcome to DrScheme  
> (define double (lambda (x) (* x 2)))  
> (define half (lambda (x) (/ x 2)))  
> (double 1)  
2  
> (double 2)  
4  
> (half 4)  
2  
> (half 2)  
1
```

The Substitution Model of Procedure Application

(Example)

```
Welcome to DrScheme
> (define seven 7)
> (define square (lambda (x) (* x x)))
> (square seven)
49
```

The Substitution Model of Procedure Application

1. Start with: (square seven).
2. Evaluate variables "square" and "seven".
 - The value of square is (lambda (x) (* x x)).
 - The value of seven is 7.
 Now we have: ((lambda (x) (* x x)) 7).
3. Replace x with 7 in the body of (lambda (x) (* x x)).
 - Now we have: (* 7 7).
4. Evaluate (* 7 7) to get 49.

Positional Association

```
(average 1066 2000)
((lambda (x y) (/ (+ x y) 2)) 1066 2000)
```

Formal Arguments Actual Arguments

- Which actual argument is substituted for x?
- The one in the first position, since x is first.
- Which actual argument is substituted for y?
- The one in the second position, since y is second.

Quadratic Procedure

```
(define quadratic
  (lambda (a b c x)
    (+ (* a (square x)) (+ (* b x) c))))
```

Solving a Quadratic Equation

```
(define discriminant (lambda (a b c) (- (square b) (* 4 (* a c)))))
(define root1
  (lambda (a b c)
    (/ (+ (- b) (sqrt (discriminant a b c))) (* 2 a))))
(define root2
  (lambda (a b c)
    (/ (- (- b) (sqrt (discriminant a b c))) (* 2 a))))
```

Solving Equation $x^2 - 2 = 0$

```
> (root1 1 0 -2)
1.4142135623730951
> (quadratic 1 0 -2 1.4142135623730951)
4.440892098500626e-16
> (root2 1 0 -2)
-1.4142135623730951
> (quadratic 1 0 -2 -1.4142135623730951)
4.440892098500626e-16
```

Solving Equation $x^2 - 7x + 12 = 0$

```
>(root1 1 -7 12)
4
```

```
>(quadratic 1 -7 12 4)
0
```

```
>(root2 1 -7 12)
3
```

```
>(quadratic 1 -7 12 3)
0
```

Solving a Quadratic Equation

```
(define discriminant
  (lambda (alpha bravo charlie)
    (- (square bravo) (* 4 (* alpha charlie)))))
```

```
(define root1
  (lambda (a b c)
    (/ (+ (- b) (sqrt (discriminant a b c))) (* 2 a))))
```

```
(define root2
  (lambda (a b c)
    (/ (- (- b) (sqrt (discriminant a b c))) (* 2 a))))
```

Using the Substitution Model

```
(root1 1 -7 12)
```

```
((lambda (a b c) (/ (+ (- b) (sqrt (discriminant a b c))) (* 2 a)))
  1 -7 12)
```

```
(/ (+ (- -7) (sqrt (discriminant 1 -7 12)))) (* 2 1))
```

```
(/ (+ 7 (sqrt (discriminant 1 -7 12)))) 2)
```

```
(/ (+ 7 (sqrt ((lambda (alpha bravo charlie)
  (- (square bravo) (* 4 (* alpha charlie)))))
  1
  -7
  12)))
```

```
2)
```

Using the Substitution Model

```
(/ (+ 7 (sqrt ((lambda (alpha bravo charlie)
  (- (square bravo) (* 4 (* alpha charlie)))))
  1
  -7
  12)))
```

```
2)
```

```
(/ (+ 7 (sqrt (- (square -7) (* 4 (* 1 12))))) 2)
```

```
(/ (+ 7 (sqrt (- (square -7) 48))) 2)
```

```
(/ (+ 7 (sqrt (- 49 48))) 2)
```

```
(/ (+ 7 (sqrt 1)) 2)
```

```
4
```

Procedural Abstraction

- Captures a pattern in commonly occurring expressions.
- Defines a procedure that allows user to use the pattern over and over.
- Gives a name to the procedure.
- When the user uses the name, he/she does not need to remember the details of the pattern.
- The user needs only to remember what arguments the procedure accepts, and what value the procedure returns.