

Difference between procedures and mathematical functions

Mathematical functions are defined declaratively, by stating WHAT the conditions are that their values satisfy.

Example:

$\text{sqrt}(x)$ = the unique y such that $y \geq 0$ and $y * y = x$.

Procedures are defined by stating step by step HOW to find the desired value.

Example:

Newton's method for computing square roots.

Newton's method for square roots

General approach:

- If a guess is good enough, return it.
- If not good enough, compute a better guess.
- Repeat

(As a practical matter, we'll only compute an approximate value.)

Square root of 10

Guess:	Good enough?
2	$2 * 2 = 4$
$(2 + (10/2))/2 = 3.5$	$3.5 * 3.5 = 12.25$
$(3.5 + (10/3.5))/2 = 3.1785$	$3.1785 * 3.1785 = 10.1029$
...	...
3.162277660168...	

Newton's method (code)

```
(define (sqrt x)
  (compute-sqrt 1.0 x))

(define (compute-sqrt guess x)
  (if (good-enough? guess x)
      guess
      (compute-sqrt
       (better guess x)
       x)))
```

(code continued)

```
(define (good-enough? guess x)
  (< (abs (- x (square guess)))
     0.000001))

(define (better guess x)
  (average guess (/ x guess)))

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

How well does it perform?

```
(sqrt 10) = 3.1622776651757
```

```
(sqrt 0.00000001) = 0.00097997
```

Better performance

- $(x - y * y)$ should be small.
 - $\frac{(x - y * y)}{y * y}$ should be small too.
- $$(x - y * y) / (y * y) = x / (y * y) - 1$$

Better code

```
(define (good-enough? guess x)
  (or (< guess 1e-100)
      (< (abs (- (/ x
                  (square guess)
                  1)))
          0.000001))))

(sqrt 0.000000001) =
0.0001000000000000082
```

Procedures as Black-Box abstractions

- A computing problem is often broken down into natural, smaller subproblems.
- Procedures are written for each of these subproblems.
- A procedure may call itself to solve a subproblem that is a smaller version of the original problem. This is called **recursion**.

Square root example

Subproblems (subprocedures)

```
sqrt
  compute-sqrt (also calls itself)
  good-enough?
  square
  better
  average
```

(primitives abs, /, +, - are also called)

Black box



We know what it does, not how

Procedures

Procedures are like Black Boxes. Their definitions (how they work) can be changed without affecting the rest of the program.

Example:

```
(define (square x)
  (exp (* 2 (log x))))
```

Procedural abstraction

- A user-defined procedure is called by name, just as primitive procedures are.
- How the procedure operates is hidden.

```
(square x)
(exp x)
```

Variables

- All symbols in a procedure definition are variables.
- All symbols in the procedure head (procedure name and parameters) are called **bound variables**.
- All occurrences of these variables in the body of the procedure definition are bound occurrences.
- All symbols in the body of the procedure that are not bound are called **free variables**.

Scope

- Bound variables in a procedure definition can be renamed without changing the meaning of the definition.
- The body of a procedure is the **scope** of the bound variables named in the procedure head.
- Changing the name of free variables will change the meaning of the definition.

Local variables

- The formal parameters of a procedure definition are local variables in the body.
- Other variables can become local variables by defining values for them; they become bound.

```
(define (area-of-circle radius)
  (define pi 3.14159)
  (* pi radius radius))
```

Procedure definitions can be nested

```
(define (sqrt x)
  (define (compute-sqrt guess x)
    (if (good-enough? guess x)
        guess
        (compute-sqrt
         (better guess x)
         x))))
```

(sqrt continued)

```
(define (good-enough? guess x)
  (or (< guess 1e-100)
      (< (abs (- (/ x
                   (square guess))
                 1))
          0.000001))))
```

(sqrt continued 2)

```
(define (better guess x)
  (average guess (/ x guess)))
(compute-sqrt 1.0 x))
```

Locally defined procedures must come first in body of definition.

Block structure

- Procedure definitions are often called **blocks**
- The nesting of procedure definitions is called **block structure**
- Variables that are free in an inner block are bound as local variables in an outer block
- Values of local variables don't have to be passed into nested definitions via parameters
- This manner of determining the values of variables is called **lexical scoping**

Using fewer parameters

```
(define (sqrt x)
  (define (compute-sqrt guess)
    (define (good-enough?)
      (or (< guess 1e-100)
          (< (abs (- (square
                     guess))
                 1)))
          0.000001))))
```

(fewer params continued)

```
(define (better)
  (average guess (/ x guess)))
(if (good-enough?
    guess
    (compute-sqrt (better))))
(compute-sqrt 1.0))
```