SICP Notes 03

https://mitpress.mit.edu/sicp/
http://www.aduni.org/courses/sicp/index.php

Geoffrey Matthews

Department of Computer Science
Western Washington University

April 6, 2015

Higher Order Procedures

- Higher-order procedures are procedures that manipulate other procedures.
- Procedures have first-class status in Scheme.
  - They may be named by variables.
  - They may be passed as arguments to procedures.
  - They may be returned as the results of procedures.
  - They may be included in data structures.

Summing

$$\sum_{i=1}^{5} i^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$$

Summing

$$\sum_{i=1}^{5} i^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$$

- If we were to write `sum` as a procedure, what are the parameters?

Summing

$$\sum_{i=1}^{5} i^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$$

- If we were to write `sum` as a procedure, what are the parameters?

```
(define sum
   (lambda (func a b)
      (if (> a b)
          0
          (+ (func a)
             (sum func
                  (+ 1 a)
                  b)))))
```

Sum examples

How do we compute $\sum_{i=1}^{n} i$?

Sum examples

How do we compute $\sum_{i=1}^{n} i$?

```
(sum (lambda (i) i)
     1
     n)
```

Sum examples

How do we compute $\sum_{i=1}^{n} i$?

```
(sum (lambda (i) i)
     1
     n)
```

How do we compute $\sum_{i=1}^{n} \frac{1}{i}$?

Sum examples

How do we compute $\sum_{i=1}^{n} i$?

```
(sum (lambda (i) i)
     1
     n)
```

How do we compute $\sum_{i=1}^{n} \frac{1}{i}$?

```
(sum (lambda (i) (/ 1 i))
     1
     n)
```

Sum examples

How do we compute $\sum_{i=1}^{n} i$?

```
(sum (lambda (i) i)
     1
     n)
```

How do we compute $\sum_{i=1}^{n} \frac{1}{i}$?

```
(sum (lambda (i) (/ 1 i))
     1
     n)
```

How about $\sum_{i=1}^{n} i^3 + 2i$?

Sum examples

How do we compute $\sum_{i=1}^{n} i$?

```
(sum (lambda (i) i)
     1
     n)
```

How do we compute $\sum_{i=1}^{n} \frac{1}{i}$?

```
(sum (lambda (i) (/ 1 i))
     1
     n)
```

How about $\sum_{i=1}^{n} i^3 + 2i$?

```
(sum (lambda (x) (+ (* x x x) (* 2 x)))
     1
     n)
```

Procedures that return procedures

Here is a procedure for exponentiation:

```
(define expt
   (lambda (b n)
      (if (= n 0)
          1
          (* b (expt b (- n 1))))))
```

How could we use this procedure to allow us to create procedures to calculate n^2, n^3, n^4, *etc.*?

Procedures that return procedures

```
(define make-expt
   (lambda (n)
      (lambda (b) (expt b n))))
```

Now, let's use the make-expt procedure we just wrote to define square.

```
(define square
   (make-expt 2))
```

And use make-expt to write cube:

```
(define cube
   (make-expt 3))
```

How do we define a procedure to return n^8?

```
(define power-8
   (make-expt 8))
```

Multiplying: A Different Take on Sum

$$\prod_{i=1}^{5} i^2 = (1^2)(2^2)(3^2)(4^2)(5^2)$$

```
(define prod
   (lambda (term a b)
      (if (> a b)
          1
          (* (term a)
             (prod term
                   (+ 1 a)
                   b)))))
```

Prod examples

Use prod to calculate $\prod_{i=1}^{n} \frac{1}{i}$

Prod examples

Use prod to calculate $\prod_{i=1}^{n} \frac{1}{i}$

```
(prod (lambda (x) (/ 1 x))
      1
      10)
```

Prod examples

Use prod to calculate $\prod_{i=1}^{n} \frac{1}{i}$

```
(prod (lambda (x) (/ 1 x))
      1
      10)
```

How about $\prod_{i=2}^{100} i^2$

Prod examples

Use prod to calculate $\prod_{i=1}^{n} \frac{1}{i}$

```
(prod (lambda (x) (/ 1 x))
      1
      10)
```

How about $\prod_{i=2}^{100} i^2$

```
(prod (lambda (x) (* x x))
      2
      100)
```

Let there be local variables

The general format of the let statement is

```
(let ((<var1> <exp1>)
      (<var2> <exp2>)
      ...
      (<varn> <expn>))
  <body>)
```

The let statement is equivalent to the following

```
((lambda (<var1> <var2> ... <varn>)
     <body>)
 <exp1>
 <exp2>
 ...
 <expn>)
```

Let example

For example,

```
(let ((a 2)
      (b 3)
      (c 4))
   (+ a b c))
```

de-sugars into

```
((lambda (a b c) (+ a b c)) 2 3 4)
```

which would return 9.

Let examples

```
(define procedure-with-let
   (lambda (x)
      (let ((a (+ x 2))
            (b (* x 3)))
         (+ a b))))
```

What will be returned by the following call?

```
(procedure-with-let 4)
```

Let examples

```
(define procedure-with-let
   (lambda (x)
      (let ((a (+ x 2))
            (b (* x 3)))
         (+ a b))))
```

What will be returned by the following call?

```
(procedure-with-let 4)
```

18