cond

```
(cond  (predicate1 exp1)
       (predicate2 exp2)
       …
       (else expn))
```

else is not mandatory
no begin is needed (because parenthesis are here
If multiple predicates are true, only the first expression is evaluated
How would you survive without else?
How would you survive without cond?

Formal definition

$R(n)=\Theta(f(n))$
iff there exists N, k_1, k_2
so that
For each n>N,
$k_1 f(n) \leq R(n) \leq k_2 f(n)$

Order of growth

6
n^2+3
$6n^3+4n^2+7$
$\log_{13}(n)$
$5 \log (n^6)$
2^{3n+7}

Why should we care

Name	Notation	n=2	n=10	n=100
Constant	$\Theta(1)$	1	1	1
Logarithmic	$\Theta(\log n)$	1	3.33	6.66
Linear	$\Theta(n)$	2	10	100
Quadratic	$\Theta(n^2)$	4	100	10,000
Exponential	$\Theta(2^n)$	4	1024	1.26×10^{30}

Order of growth space/time

```
(define (fact1 n)
  (if (= n 1) 1
     (* n (fact1 (- n 1)))))
(define (fact2 n)
  (define (helper cur k)
     (if (= k 1) cur
          (helper (* cur k) (- k 1)))))
  (helper 1 n))
```

Order of growth space/time

```
(define (fib1 n)
  (cond   ((= n 0) 0)
          ((= n 1) 1)
          (else (+  (fib1 (- n 1))
                    (fib1 (- n 2))))))
(define (fib2 n)
  (define (fib-iter a b count)
     (if (= count 0) b
          (fib-iter (+ a b) a (- count 1)))))
  (fib-iter 1 0 n))
```

Print all factorials

Write a procedure that prints all factorials from 1 to n

Be dumb, then be smart.

Analyze the orders of growth

Print all factorials

Write a procedure that prints all factorials from 1 to n

Be dumb, then be smart.

Analyze the orders of growth

Now use display-bar. What does it change?

```
(define display-bar (lambda (n)
        (if (= n 0) (display "\n")
            (begin (display ".")
                   (display-bar (- n 1))))))
```

Multiplication

Write a procedure (fastmul m n) that is faster than linear

Binary numbers

```
(define binary (lambda (n)
  (if (> n 0)
      (begin
        (binary (floor (/ n 2)))
        (display (if (even? n) "0" "1"))))))
```

Patterns

Recursive call:

- n calls n-1 → linear
- n calls n/2 → log
- n calls n-1 & n-2 → exponential

2 issues

- How many steps
- Cost of each step

Exponential processes: often, the recursive procedure calls itself twice (or more) at each iteration

Quadratic processes often correspond to two linear processes where the first one calls the second one

Order?

```
(define (dummy x)
  (if (<= 1 x) x
      (+  (dummy (floor (/x 2))
          (dummy (- x (floor (x-x/2)))))
```