Typical recursion

```
(define recursive-proc
  (lambda (something)
     (if test base-case
        (operation something
          (recursive-proc something-simpler)
```

Typical problem

Forget the if

Forget to use something-simpler
(keep calling with same argument)

Forget to perform operation

Recursive vs. iterative

Similar procedure

Different process

Recursive has pending operations

Iterative typically requires helper function with more arguments to count and keep track of result

Proving programs

Often do the opposite: try to prove it fails

e.g. aeronautics

Multiplication

Write a procedure that multiplies, using only additions

Recursive? Iterative?

begin

```
(begin exp1 exp2… exp n)
Special form (evaluated in
  order)
Useful for if
(if predicate
  (begin 1 (display “true”) ex1)
  (begin (display “false”) ex2) )
```

Counting

Consider the following two procedures.

```
(define (count-down x by)
  (if (< x 0) #t
    (begin (print x)
      (count-down (- x by) by))))
(define (count-up x by)
  (if ((< x 0) #t)
    (begin
      (count-up (- x by) by)
      (print x))))
```

What happens for each of

```
(count-down 11 3)
(count-up 11 3)
```

Display bar

Write a procedure
  (display-bar n)
  that displays n dots

e.g.
  (display-bar 6)
    ......

Use (display “.”)

Syracuse a.k.a. Hailstone series

```
(define syra (lambda(n)
    (if (even? n) (/ n 2)
        (+ 1 (* 3 n)))))
```

Implement Syracuse that writes all the numbers in a series that starts at n

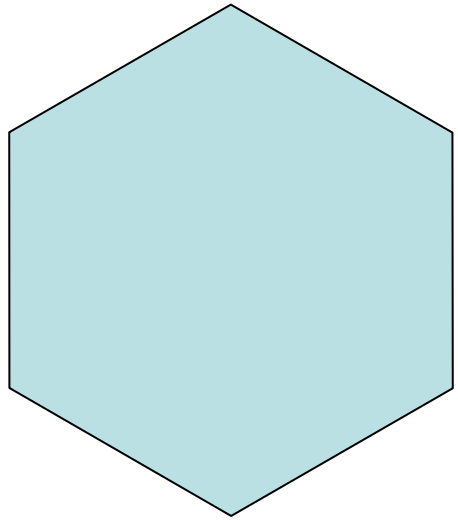
Binary

Write a procedure that prints the binary version of a number

Computing pi

U_n: edge length of n-gon

$$u_{2n} = \sqrt{\frac{1}{2}(1 - \sqrt{1 - u_n^2})}$$



And u_6=1/2

(because u_n=sin π/n…)

We have π ¼ n*u_n

We have a function (u2n un)

Write a function (mypi n)

Power

Compute x^y with only multiplications

Do it more efficiently