Summary of previous episode

- Lambda to create procedures
- Mantra: The value returned by lambda is a procedure
- Difference between creation (lambda) and naming (define)
  - Beware of dirty sugar
- Means of abstraction
  - Lambda captures patterns
  - Define names it

Scheme rules

Evaluation:

Self evaluating → return value
name → return corresponding value
Special → special
Combination → evaluate subexpressions
apply procedure to operands
Lambda → create a procedure

Application:

Primitive procedure → just do it
Compound procedure → evaluate body
with each parameter replaced by arguments

What do the following expressions evaluate to?

```
(lambda (x) (* x x))
((lambda (x) (* x x)) 5 )
(define double (lambda (x) (* 2 x)))
(double (double 6))
(double double)

(average 4 (double 4))
```

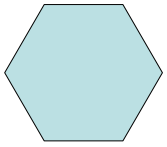
What do the following expressions evaluate to?

```
(define cube (lambda (x) (* x x x)))
(cube 3)
(define + 3)
(define - 6)
(* + -)
```

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 $u_n=\sin \pi/n$…)
We have $\pi \approx n*u_n$
Write a function (u2n un)

Call me sugar

```
(define (f x) (+ x 2))

Always remember to desugar
```

Lambda and variables

```
(define f (lambda (x) (+ x 1)))
(define g (lambda (x) (+ x 3)))
(define x 8)
(f 4)
(g 5)
x
```

Type and errors

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

Where is the error, who is complaining?

If

```
(if predicate
   expression-for-true
   expression-for-false)
```

Example:

```
(if (even? x) (/ x 2) x)
```

If is a special form!
e.g. special division

Write a function that gives the max of two numbers

Write a function sign that returns +1 or -1 depending on the sign of the argument

Syracuse numbers

The Syracuse series is defined as follows:
When a number is even, half it
If it is odd, multiply by three and add one

Write a syracuse function

Study the behavior for various numbers
Difference procedure/process