Numbers, Expressions, and Simple Programs
Today’s Goals

- Discipline! Order! Developing programs requires care!
- Programs are useful but delicate entities.
- They run fast but can also easily get derailed.
  - If program running car engine fails, can lead to loss of life
- To understand programs (“computation”),
  - We should know how to write them correctly
  - … this is more subtle than it seems at first.
- Coolest thing in this course: Design Recipes.
Why the Scheme Programming Language?

- Simple syntax (and we'll use only a subset)
  - Function definition
  - Function application
  - Conditionals
  - Structure definitions
  - Local definitions
  - Assignment

- Simple semantics
  - Can be understood abstractly
  - Can be understood rigorously
  - Nice integrated design environment (IDE)
Computing with Numbers

- Naturals: 0, 1, 2, ...
- Integers: -1, -12, ...
- Rationals: (3/4), (-5/6), ...
- Reals: pi, e, phi, ...
- Complex numbers: (sqrt(-1))

Note: Exact computation is still an active research topic.
Computer Representations

- All the above types of numbers can be represented
- More typically, more limited forms are used:
  - "int" : 5 as (0101)
  - "float" : 0.5*10^6 as (1000) , (0110)
- DrScheme has arbitrarily large integers. But reals are fixed.
Basic "data types" are good. But operations on them are more fun!

- (+ 1 2), (* 3 4), or
  - (\* 4 2) where \* \in \{-,/,exp,mod,div\}
- (+ (+ 1 2) 3)
- (+ 1 (+ 2 3))
- (* (+ 1 2) (+3 4))

How does this compare to 1+2*3+4?

- Good: Unambiguous: result is 11, 13, 15, or 21!?
- Bad: Longer!
Defining Our First Program

- Example: The area of a circle
  - Mathematically, \( A(r) = \pi \times r^2 \)

If we approximate \( \pi \) by 3.14, in DrScheme:

```scheme
(define (area-of-disk radius) (* 3.14 (* radius radius)))
```

- Note: “\( A \)” became “area-of-disk”, “\( r \)” became “radius”.

- Longer names optional, but help readability!

- What language constructs do we use here?
  - 1) function definition: “(define …)”, 2) variables: “radius”, 3) primitive operators: “*”, 4) literal constants: “3.14”
Using our first program

- We've just made a big investment in implementing the formula.
- What's the benefit from doing that? Now, all we need to do is to type:
  - (area-of-disk 5)
- This evaluates to
  - \( = (*\ 3.14\ (*\ 5\ 5))\)
  - \( = (*\ 3.14\ 25)\)
  - \( = 78.5\)
- Without our procedure definition, we have to type \( (*\ 3.14\ (*\ x\ x))\) over and over again…
Now that we know how to implement “area-of-circle”…

- What other things can we implement?
  - volume of a cone
  - GPA for fixed number of courses
  - sum of a list fixed length list

- What things can we NOT implement yet
  - GPA for arbitrary number of courses
  - Sum for list of arbitrary list

- So, we can implement a lot of algorithms, but not all.