Booleans, More BNF, Functions
WiCS
Women In Computer Science
Dedicated to improving diversity and inclusion across gender identity in CS.

Email: wics@lists.cs.brown.edu

Add yourself to the listserv:

https://tinyurl.com/brownuwicwics
Warmup: a new kind of data

- Boolean datatype
  - George Boole, “The Laws of Thought”
- Exactly two values
  - true
  - false
- In Racket, these are written `true, false`
- Both of these are *keywords*
  - *That means you can’t use them as names*
- NB: so far three kinds of data: numbers, strings, booleans
- Three still to come: *functions*, lists, (structures)
Last class

• Broke Racket program text into pieces called “tokens”
• Saw a standard form for definitions
• Learned a way to compactly represent that standard form
  • Backus—Naur form, called BNF
  • Always written in green on slides
  • Some parts informal; written in *italics*
A small note on the behavior of definitions (demo)

• This is not about what’s legal as a racket program (informally, what’s “grammatically correct”), but about what happens when you RUN a racket program

• If you define the same thing twice, you get an error:

(define class 17)
(define class 18)  generates an error!
Why am I obsessing about syntax?

• Syntax: the “grammar” of the language, what it’s legal to write
  
  \[ \text{<prog>} := \text{<defn>}^* \text{[<expr>]} \]
  \[ \text{<defn>} := \text{(define <name> <expr>)...} \]

• Each item on the left-hand side of the BNF for Racket will become a named thing in your Rackette assignment later in the semester!

• As you’re trying to write a program, it’s nice to have a guide to what \textit{might} possibly work.
Our current BNF description of Racket

<prog> ::= <defn>* [<expr>]
<defn> ::= (define <name> <expr>)

<name> ::= <CS17name> | <othername>

<CS17name> ::= sequence of letters, digits, hyphens, starting with a letter, usually lowercase
<othername> ::= token consisting of non-special characters that can’t be interpreted as a number and that isn’t a keyword

<expr> ::= lots to fill in here!

Read as “or”
Abstraction

• Breaking things down into pieces ("tokens", for our language) and
• Giving rules for "legally" assembling pieces...
• Makes describing the legal Racket programs fairly simple
• An example of *abstraction*:
  • Don’t sweat the font
  • Don’t worry about what things *mean* yet
  • *Focus on a narrow task, and ignore everything else*
We can use BNF to describe other stuff

• Example: tokens are single letters. No spaces allowed.
  \(<\text{word}\> ::= \langle\text{capital}\rangle \langle\text{letter}\>^*\)
  \(<\text{capital}\> ::= A | B | C\)
  \(<\text{letter}\> ::= \text{a} | \text{b} | \text{c} \ldots | \text{z}\)
  • Legal “words” defined by this set of rules?
    Abd
    Bed
    Armchair
    Ccccc
    Activity: What’s an example of something that doesn’t fit this definition of word?
    cC, ABA, 14
    Activity: What’s the shortest possible thing that fits this definition? (Multiple correct answers)
    A, B, C
Application of “matching patterns” as in BNF: Eliza

• A program that breaks English sentences into words
• Looks for patterns, like I hate <something>.
  • Here “something” can be any sequence of words
• When eliza gets a pattern it recognizes, it provides a response, like Why do you hate <something>?, filling in exactly the same text.
• The author of the program gets to provide a lot of patterns/ responses, and when you use the program, it feels almost as if you’re having a conversation.
• What about input that doesn’t match any pattern?
  • We always include, as a final pattern, something that matches anything
  • And as a response, we use something like Tell me more.
Demo
Thoughts

• Every program you write has consequences
• They’re really hard to predict
• Sometimes they’re hard to see because of the assumptions we make.
  • Who can’t use Eliza?
A BNF description for the expressions we’ve encountered so far

\[
<\text{expr}> ::= <\text{name}> | <\text{num}> | <\text{misc}>
\]

\[
<\text{num}> ::= \text{stuff that looks like a number}
\]

\[
<\text{misc}> ::= ( <\text{op}> <\text{expr}> <\text{expr}> )
\]

\[
<\text{op}> ::= \pm | \div | \times | /
\]

[NB: those last two lines will soon be replaced]
A few words from math

• Racket is based on the mathematical notion of *function*
• We’ll review that here
• My notion of “function” may be different from yours
  • Yours may have been simplified to make it easy to teach
  • Mine is universally accepted by mathematicians
  • The difference is very small
• Let’s start with “set”
Sets
Ways to describe sets: Natural language
Ways to describe sets: Enumeration
Activity
A special enumerated set
Ways to describe sets: Restriction