Whirlwind Tour of Racket

CS 115 | Saelee

IIT College of Science and Letters
ILLINOIS INSTITUTE OF TECHNOLOGY

What's in a name?

- Racket vs. Scheme vs. BSL vs. …
- Different dialects of one language
  - Slightly different features & grammar
- I'll (try to) always say "Racket"

Data

- "Stuff" our programs work on
- Can be used as *input* and *output*

Atomic data

- Discrete (logically indivisible) chunks of information
- E.g., numbers: `0`, `1`, `2`, `3`, `4`, …
  characters: `#\a`, `#\b`, `#\c`, ...
  booleans: `true`, `false`
  symbols: `'red`, `'wednesday`

Compound data

- Can be split into constituent parts
- E.g., strings: `"hello world"`,
  lists: `(list 1 'fish 2 'fish)`
- `(string-ref "hello world" 0)` ⇒ `#\h`
  `(first (list 1 'fish 2 'fish))` ⇒ `1`

Named values

- `(define mypi 3.14)` *binds* the *variable* `mypi` to the constant `3.14`
- Once defined, cannot re-define the value of a variable!

Q:

- How are variables useful?
- Why aren't we allowed to redefine variables?

Arithmetic

- "Primitive" operations: `+`, `-`, `*`, `/`

```
(- 3 4) ⇒ -1

(* 9 1000) ⇒ 9000

(/ 1 (* 3 3)) ⇒ 0.1̄
```

Mathematical functions

- E.g., `abs`, `sqr`, `sqrt`, `expt`, `sin`
  - Many more provided by Racket
- Use in the same way as primitive operators
  - In reality, "primitive operators" are just functions with symbols for names

Mathematical functions

```
(abs -7)   ⇒ 7
(expt 2 3) ⇒ 8
(sqrt (+ (sqr 3) (sqr 4))) ⇒ 5
```

Expressions

- The application of functions to their inputs create *expressions*
  - e.g., `(+ (sqr 3) (sqr 4))`
- All expressions *evaluate* (or *reduce*) to values

Wait a sec…

- `define` doesn't produce a value, when evaluated
  - It isn't a function!
  - `define` is a "special form"

Referential Transparency

- Fancy sounding but simple idea:
  - Applying a *referentially transparent* function to the same input(s) many times will always result in the same value
  - e.g., (+ 2 2) is always = 4 (duh!)

Q:

- Why is referential transparency good?
- Would it ever make sense to not have referential transparency?

Comparisons & Booleans

- Relational operators from math:
  - `=`, `<`, `<=`, `>`, `>=`
- Expressions that uses these operators evaluate to Boolean values
  - `true` or `false`

Comparisons & Booleans

```
(> 10 5)       ⇒ true
(= 1 2)        ⇒ false
(<= -5 2000)   ⇒ true
```

Comparisons & Booleans

- Also other functions for comparison
  - `symbol=?`, `string=?`, etc.

Boolean operations

- For compound expressions, e.g.,
  - time ≥ 1.5 hours or color ≠ pink
- Boolean operations: `and`, `or`, `not`
  - Input: Boolean(s); Output: Boolean

Boolean operations

```
(define time 1.8)
(define color 'pink)

(or (>= time 1.5)
    (not (symbol=? color 'pink)))
```

Conditional expressions

- Likely want to do different things based on value of a Boolean expressions
- E.g., take bicycle if sunny, bus if rainy
- Special form: `cond`

Conditional expressions

```
(define weather 'rainy)

(cond ((symbol=? weather 'sunny) 'bicycle)
      ((symbol=? weather 'rainy) 'bus))
```

- How to apply this easily to different values of `weather`?

Our own functions

- E.g., so we can do

```
(transport-mode 'sunny)
(transport-mode 'rainy)
```

Our own functions

- Use `define` again, but like this:
  `(define (transport-mode weather) …)`
  - `transport-mode` is our function name
  - inputs to function are *bound* to `weather`
  - … is the function body

Our own functions

```
(define (transport-mode weather)
  (cond ((symbol=? weather 'sunny) 'bicycle)
        ((symbol=? weather 'rainy) 'bus)))
```

```
(transport-mode 'sunny) ⇒ 'bicycle
(transport-mode 'rainy) ⇒ 'bus
```

Q:

- Is `cond` referentially transparent?
- What happens when no condition in a `cond` evaluates to true? Why?

Always have a default

```
(define (transport-mode weather)
  (cond ((symbol=? weather 'sunny) 'bicycle)
        ((symbol=? weather 'rainy) 'bus)
        (else                      'car)))

(transport-mode 'freezing) ⇒ 'car
```

Drawing

- See http://docs.racket-lang.org/teachpack/2htdpimage.html for documentation of drawing functions

Just for fun …

- Draw the IIT logo (use triangles and lines)