Lecture 1

CS-151

Course logistics

Homework - After every class
Due before the next class. 1:30pm CST

Midterm - Take home, open book
Final - Take home, open book

Lab - If won't be graded.
No submissions.
Building a checkers app
- Lab sessions, Homework

Lecture slides are already available.

Zoom recording - Class sessions.
+ Tutorials (explaining how to use Dr Racket or other tools)
This week's homework already available.

Canvas Left panel - assignments

Make sure you sign-up for piazza.

Programming using Racket

- Variant Racket (Typed Racket)

- Racket is a multi-paradigm

programing language.

But we will be focussing on the functional part.

- Not like C, python, java
- No loops
- Imperative and functional
  Difference - pure functions.

function — Math (Input → Output)
         — CS (set of instructions)

Functional program : Math-def
≡ CS-def

Crash course on how compiler

Works

Dr Racket ← Code (text file or .rkt).

Your computer doesn't understand racket language.
(Machine code)

Dr Racket: rkt file
→ Machine code.
→ execute machine code
→ output on your screen.

Expressions

9 * (10 − 7) + 15.5

prefix notation

((9 * (10 − 7)) + 15.5)

(+ (* 9 (-10 7)) 15.5) 15.5

(182 / (9+4))

≡ (/ 182 (+ 9 4))

(5+6)

≡ (+ 5 6)

1+ 2+ 3

≡ (+ 1 2 3)

Common errors

(5+6)

≡ (+ 5 6)

((+ 5 6)) — Incorrect

Extra parenthesis.

parenthesis - special meaning

(+ 10 6) ⇒ 16

operation

((+ 10 6)) ⇒ (16) ✗

Not an operation

Braces should start with functions, operations or keywords

Data Types

→ Program - Input is some data, Output is data

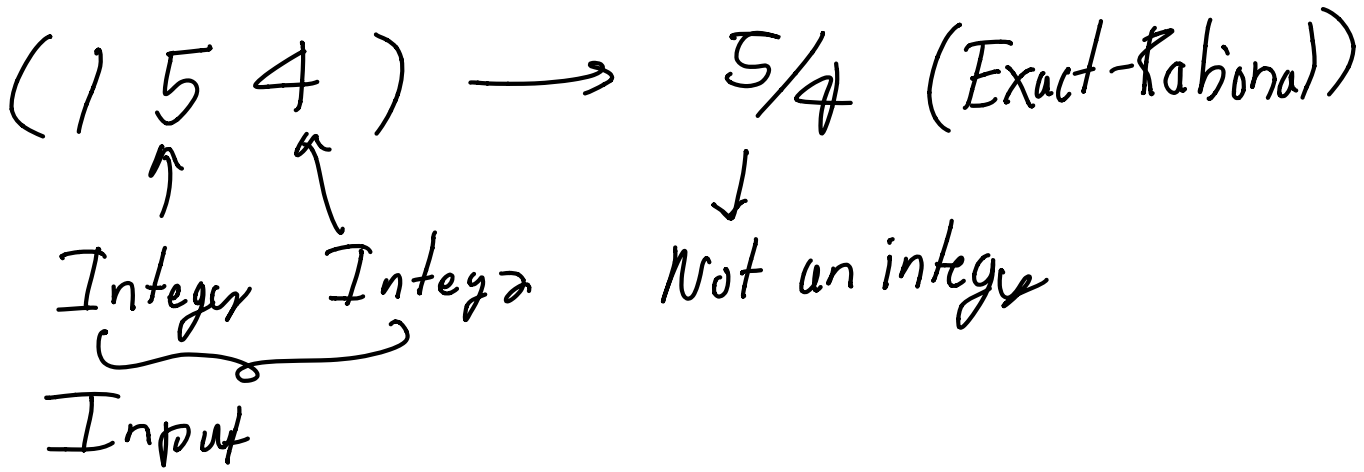
Input Data → [program/function] → Output Data

Data types

Integers - 0, -1, 2, 5

Exact-Rational → 5 1/6, 3/4

Reals - $\sqrt{2}$, $\sqrt{3}$.



Suppose we want to compute the volume of Earth.

$$Vol = \frac{4}{3}\pi r^3.$$ r - radius of the earth.

$Vol = (4/3) \cdot \pi \cdot r \cdot r \cdot r$

$\equiv$ (* (1 4 3) pi 6400 6400 6400)

Input: Real Output → Real.

Boolean Types

- Boolean → #t, #f.

(< 3 5) $\equiv$ (3 < 5)

Racket form

(< 3 5) $\Longrightarrow$ #t.

and, or and not operators

(3<5) and (3>0) Is this true?

((3<5) and (3>0))

|||

(and (< 3 5) (> 3 0)) (Racket equivalent)

(< 3 5) ⇒ #t

(> 3 0) ⇒ #t

(and #t #t) ⇒ #t

If expressions

(if (> 81 75) 8 5)

(if condition output1 output2)

condition ⇒ #t

Value of the expression will be output1

condition ⇒ #f

Value of the expression will be output 2

(if (> 81 75) 8 5)

what does this expression evaluates to?

(if (> 81 75) 8 5)

(> 81 75) - condition

(81 > 75) ⇒ #t

(> 81 75) ⇒ #t

(if (> 81 75) 8 5)

⇒ 8

(if (> 60 70) 8 5)

⇒ 5.

(if (> 81 75) → True

8

5) ⇒ 8.

(if (> 60 80) → False

8

5) ⇒ 5

(if statement output1 output2)

statement is true output1
statement is false output2

Example

The price of a ticket to "Hamilton" is $50 if day of the month ends in 0 (otherwise it's $100)

```
(if (= (remainder 22 10) 0)
    50
    100)
```

= operator

(= 2 2) (2 is equal to 2)
⇒ #t

(= 2 3) ⇒ #f

(remainder 22 10) ⇒ 2

(remainder 18 5) ⇒ 3

(remainder 15 3) ⇒ 0

(remainder x y)
⇒ remainder when x is divided by y.

Variables

Variables are like containers where you can store data. or variable are aliases for data.

Example: (Hamilton play)

Variable

(: day-of-month : Integer)

(define day-of-month 22)

↳ Day-of-month is storing 22 or is an alias for 22.

(define day-of-month 22)

(if (= (remainder day-of-month 10) 0)
    50
    100) ⇒ 100

(remainder day-of-month 10)
⇒ (remainder 22 10) ⇒ 2

Functions

Input Data → [Function] → Output Data

Write down a function to compute volume of a sphere.

Volume

$$vol = \frac{4}{3}\pi r^3$$

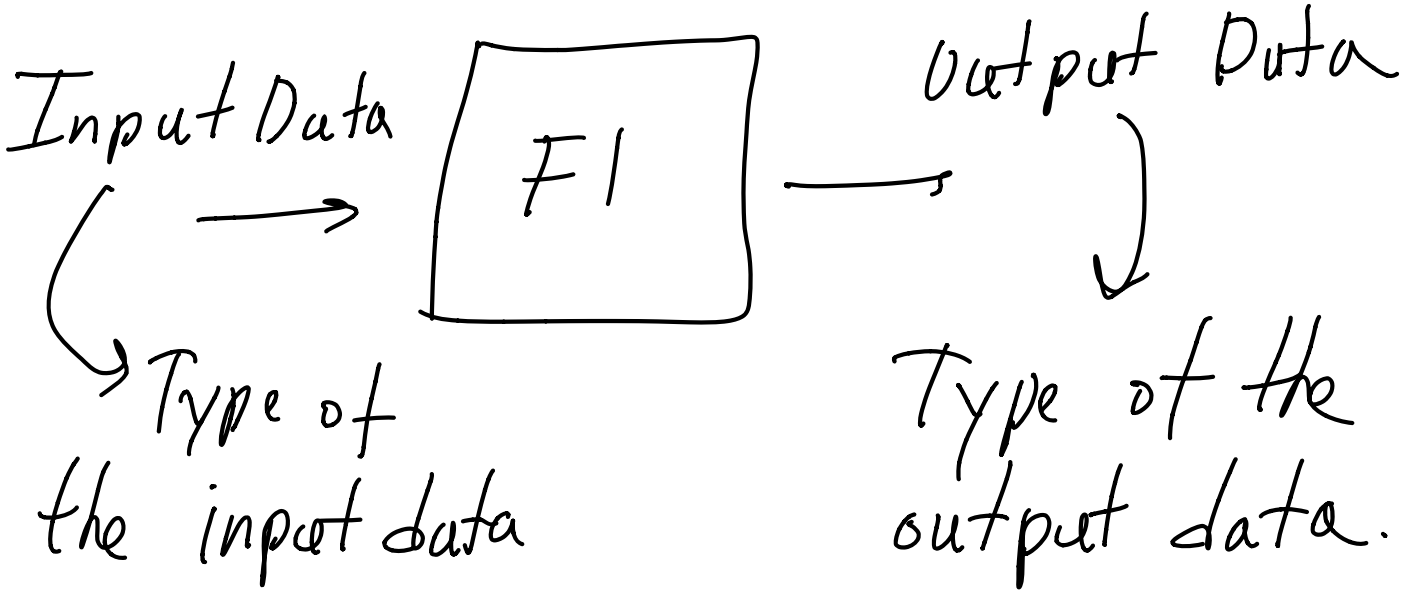
3 steps of writin down a typed Racket function

1: Function annotation

2. Define function

3. Write test cases

Typed Racket vs Ordinary Racket
is Function annotation
Ordinary Racket doesn't have
Function annotation.



We want to compute volume of
a sphere.

1. What is the input?
   Radius (Real)

2. What is the output?
Volume (Real)

```
;; Type annotation

(: compute-volume : Real -> Real)
(define (compute-volume radius)
 (* (/ 4 3) pi radius radius radius)
 )
```

(volume 1) ⇒ volume of a sphere of radius 1.

(volume 5) ⇒ volume of a sphere of radius 5.

(check-within (volume 1) 1.345 0.001)

(volume 1) = 1.345)

check that (volume 1) is with in 1.345 ± 0.001.
i.e (1.343, 1.344)

Example

Counting legs

Problem: We have some humans and some sheep total number of legs.

Write down a racket function which takes number of humans and number as input and the output is number of legs.

No. of legs = 2 * No. of humans + 4 * No. of sheep.

Step 1: Function annotation

Number of inputs: 2
Integer.

Outputs: Integer

```
(: count-legs : Integer Integer -> Integer)
;; We are counting legs
(define (count-legs num-humans num-sheep)
  (+ (* 2 num-humans) (* 4 num-sheep))
  )
(check-expect (count-legs 1 1) 6)

(check-expect (count-legs 2 1) 8).
```

Strings

String is another data type

"Hello world!"

"Bob"

Write a program which outputs the string "It's 5!!!" if the input is 5 or the output will be "Not 5..."

1. Function annotation.
Input - Integer.
Output - String

```
(: is-it-5 : Integer → String)

(define (is-it-5 num)
 (if (= num 5) "It's 5!!!" "Not 5..."))

(check-expect (is-it-5 5) "It's 5!!!")
```

(check-expect (is-it-5 10) "Not 5..")

Example

Write a function named "whats-for-breakfast" take a string as input and output "Coming right up" if the input string is yogurt or it will print "We don't have that".

(: whats-for-breakfast : String → String)

(define (whats-for-breakfast str f)
(if (string=? str "yogurt")
"Coming right up"

"We dont have that"))

(if (string=? str "yogurt")

(string=? str "yogurt") ⟹ #f
if str = "abc"

if str = "yogurt"
(string=? str "yogurt") ⟹ #t

(string=? str "abc")

(whats-for-breakfast "abc")

"abc" → str

```
(string=? "abc" "yogurt" "
```

```
(define (whats-for-breakfast s1 s2)
```

```
(if (or (string=? s1 "yogurt")
        (string=? s2 "yogurt"))
```

```
    "Coming right up"
```

```
    "We don't have that"))
```