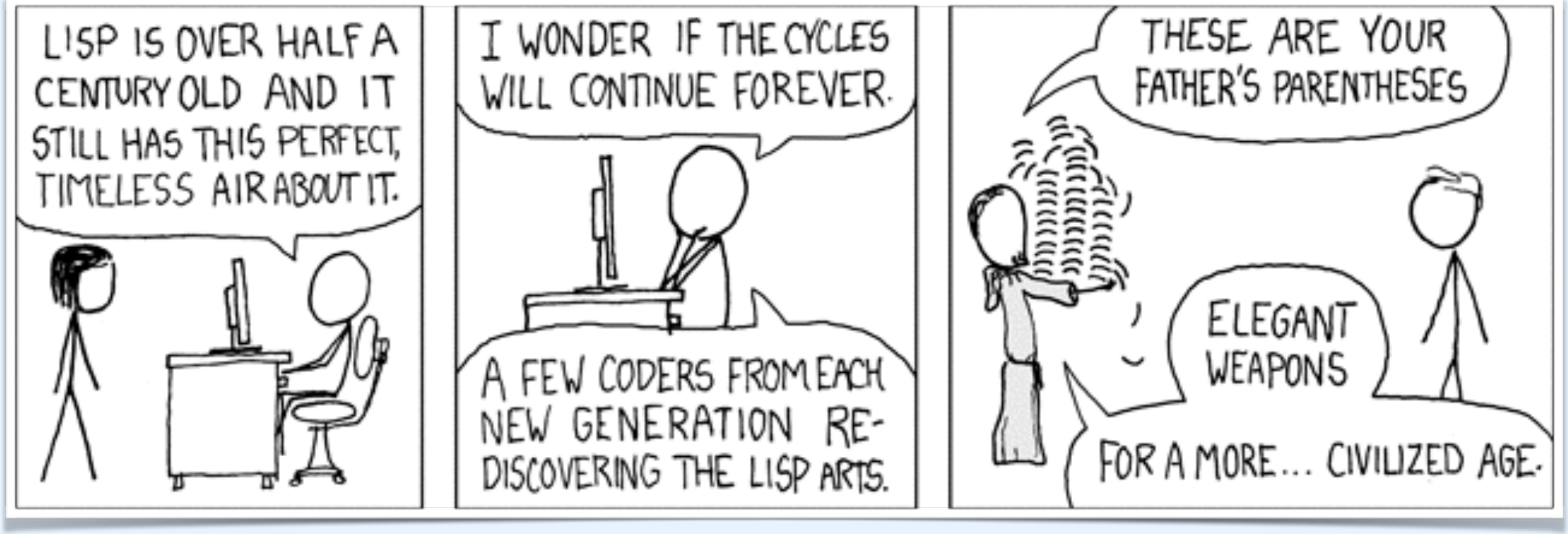
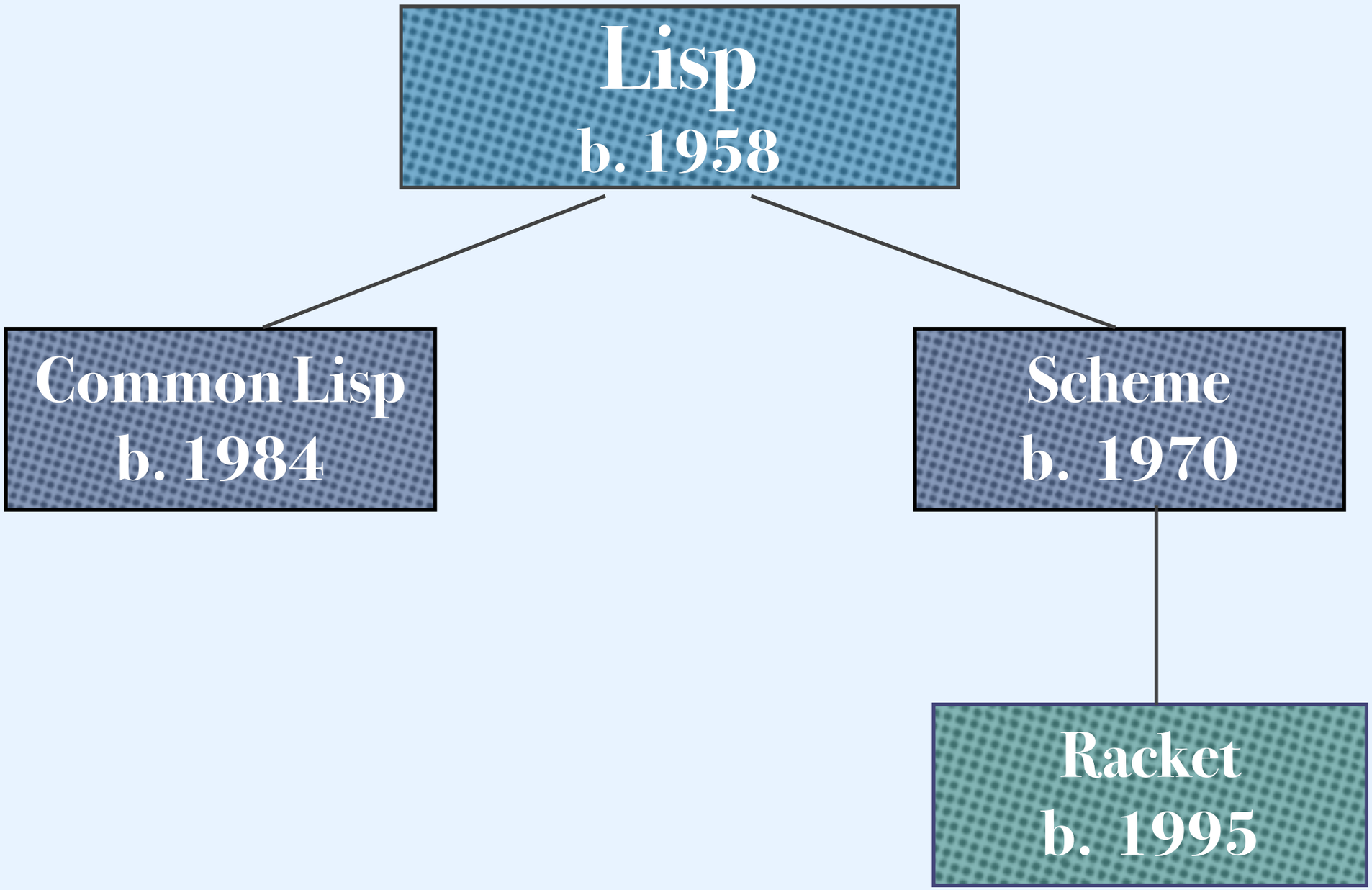



xkcd comic 297

# Introduction to Racket

*September 11, 2018*

# What is Racket?

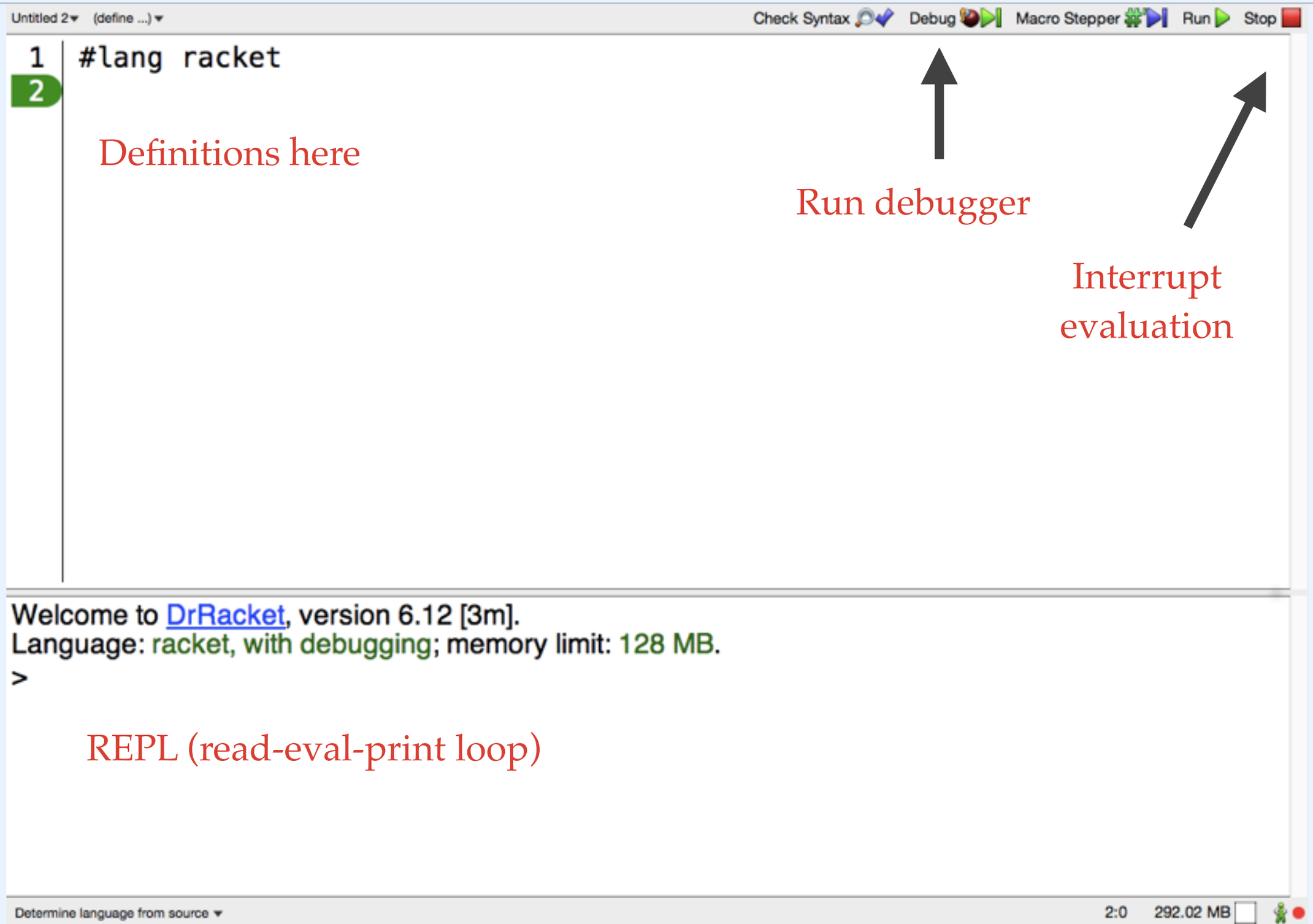


Racket is a general purpose functional-programming language

We're going to use just a subset:

a subset without mutation

# Dr. Racket


Untitled 2
(define ...)
Check Syntax
Debug
Macro Stepper
Run
Stop
1
#lang racket
2
Definitions here
Run debugger
Interrupt evaluation
Welcome to DrRacket, version 6.12 [3m].
Language: racket, with debugging; memory limit: 128 MB.
>
REPL (read-eval-print loop)
Determine language from source
2:0
292.02 MB

# Basic datatypes

## Booleans

#t

#f

## Numbers

1

1/2

1.0

## Strings

“hi”

“h”

## Characters

#\h

#\λ

# Functions

```
(define (hello-world)
        (printf “Hello world!”))
```

# Control Flow

```
(if (= x 5)        test
    #t             value if true
    #f)            value if false
```

```
(cond ((= x 0) (printf “x is 0”))
      ((= x 1) (printf “x is 1”))
      (else (printf “x is greater than 1”)))
```

# Why are there so many parentheses?


xkcd comic 312

# Euclid's algorithm for GCD

Find greatest common divisor of r1 and r2:

base case:

If r1 = 0:

    return r2

If r2 = 0:

    return r1

kth step:

If r1 and r2 are greater than 0:

    r1 / r2

    GCD(r2, remainder)

# Local binding

A let expression binds a set of variables for use in the body of the let block.

```
(define (greet str)
        (let ((greeting (string-append “hi “ str))
              printf(greeting))
```