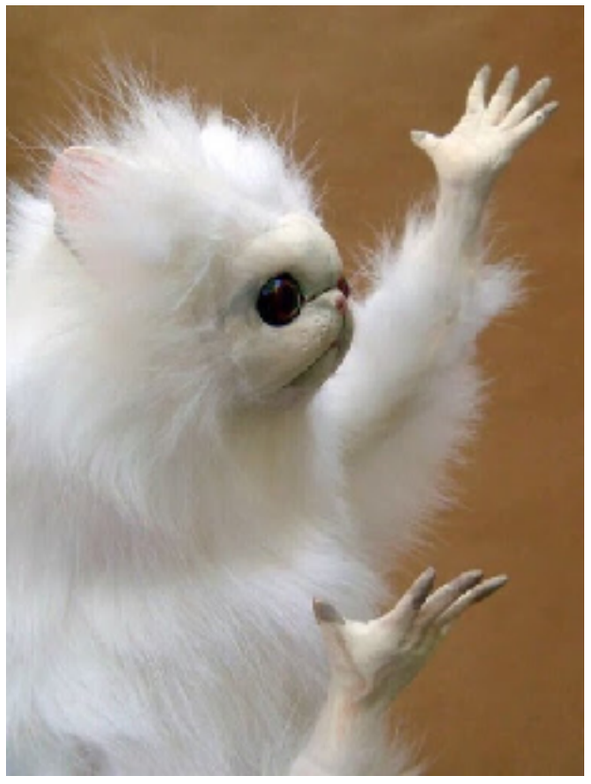
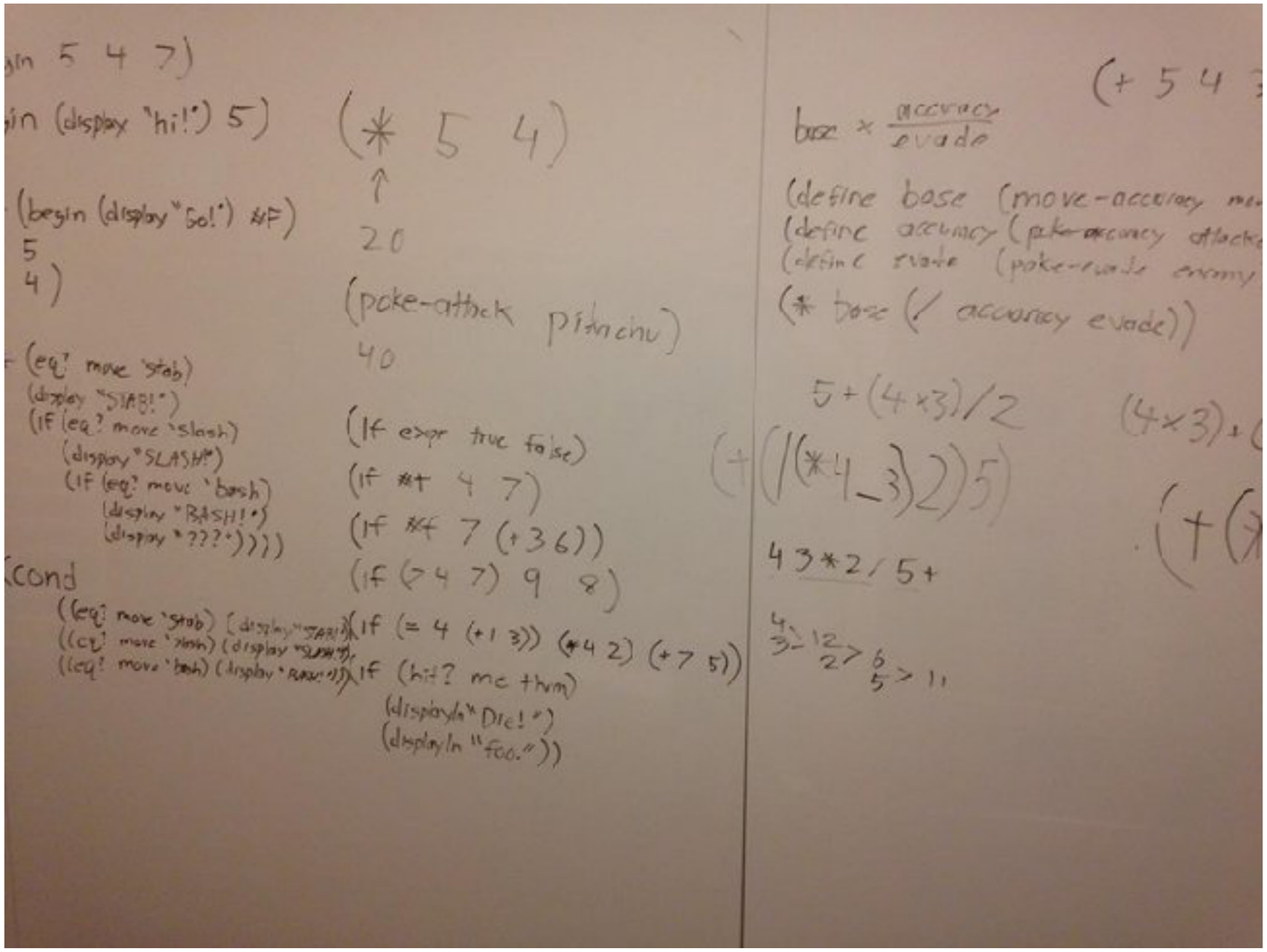
Ahoy world!

«Мне нравится функциональное программирование, потому что оно проще, а я люблю простые вещи.»

«Кажется, что Python — это шаг назад, но своего сына я буду учить в первую очередь Python.»

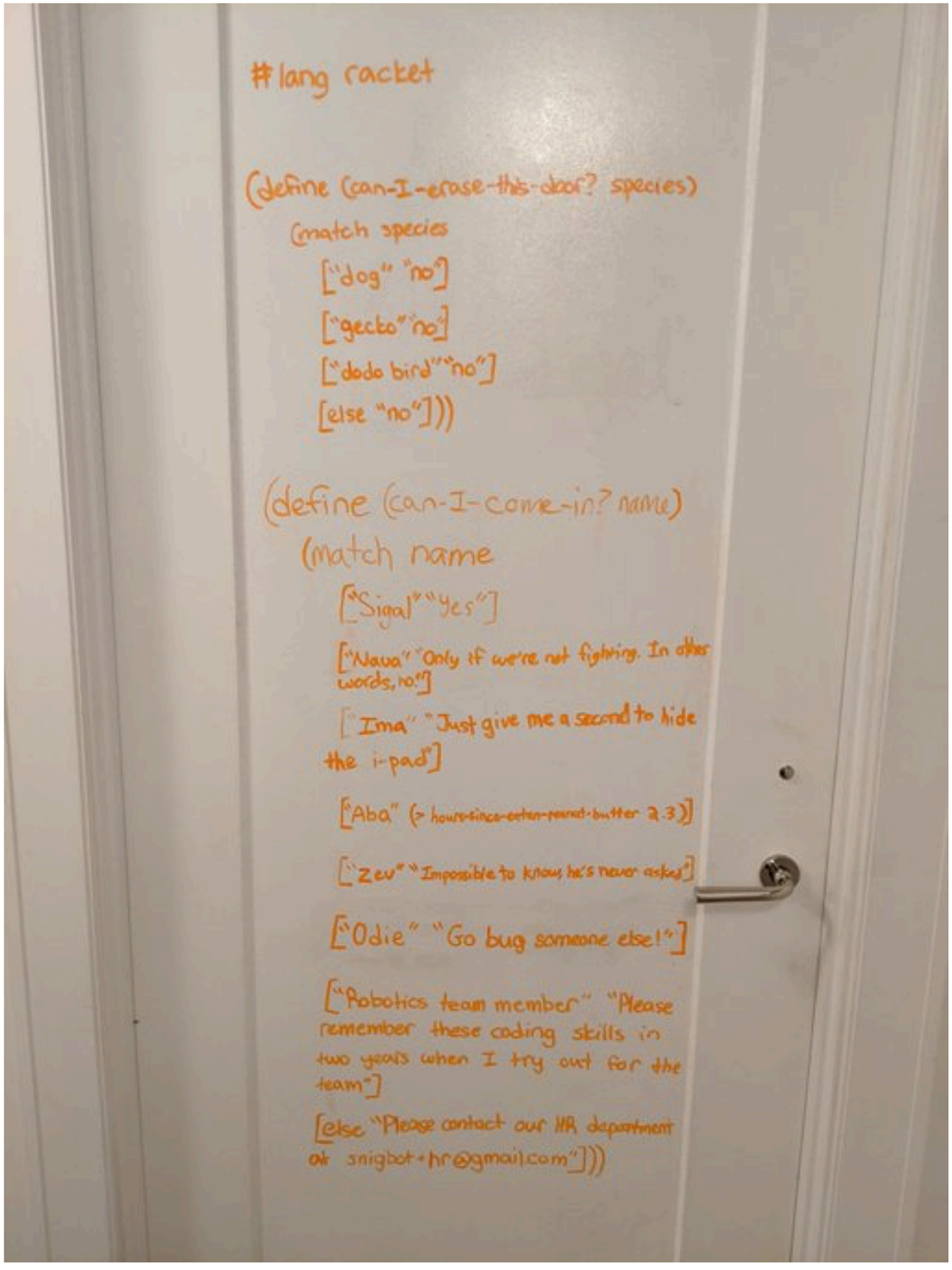




John Carmack @ID_AA_Carmack · Feb 23, 2015

Teaching my older son lisp (Racket). Working on a Pokemon battle simulator.

http://bit.ly/carmack_racket

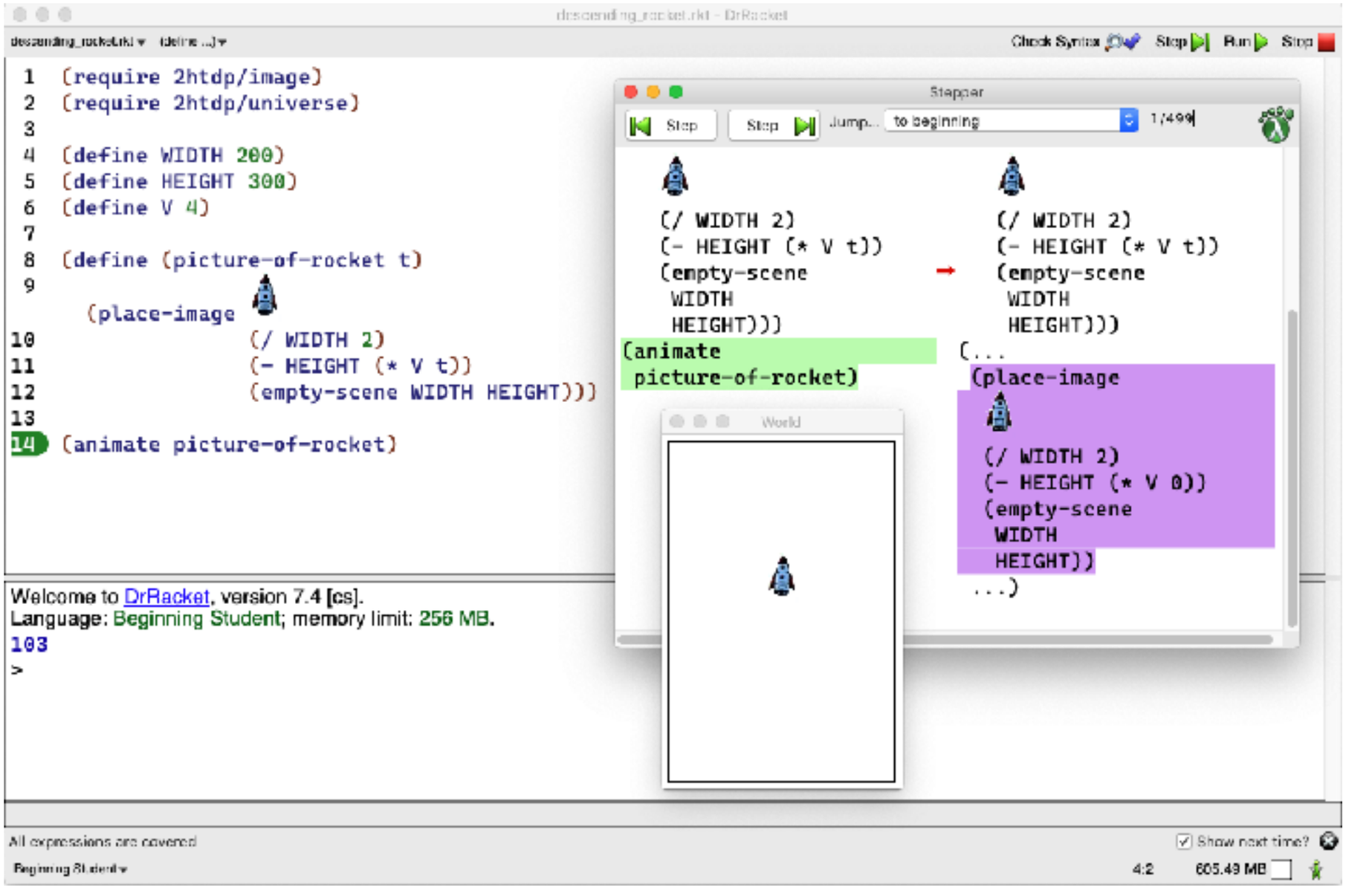


Yaron Minsky
@yminsky

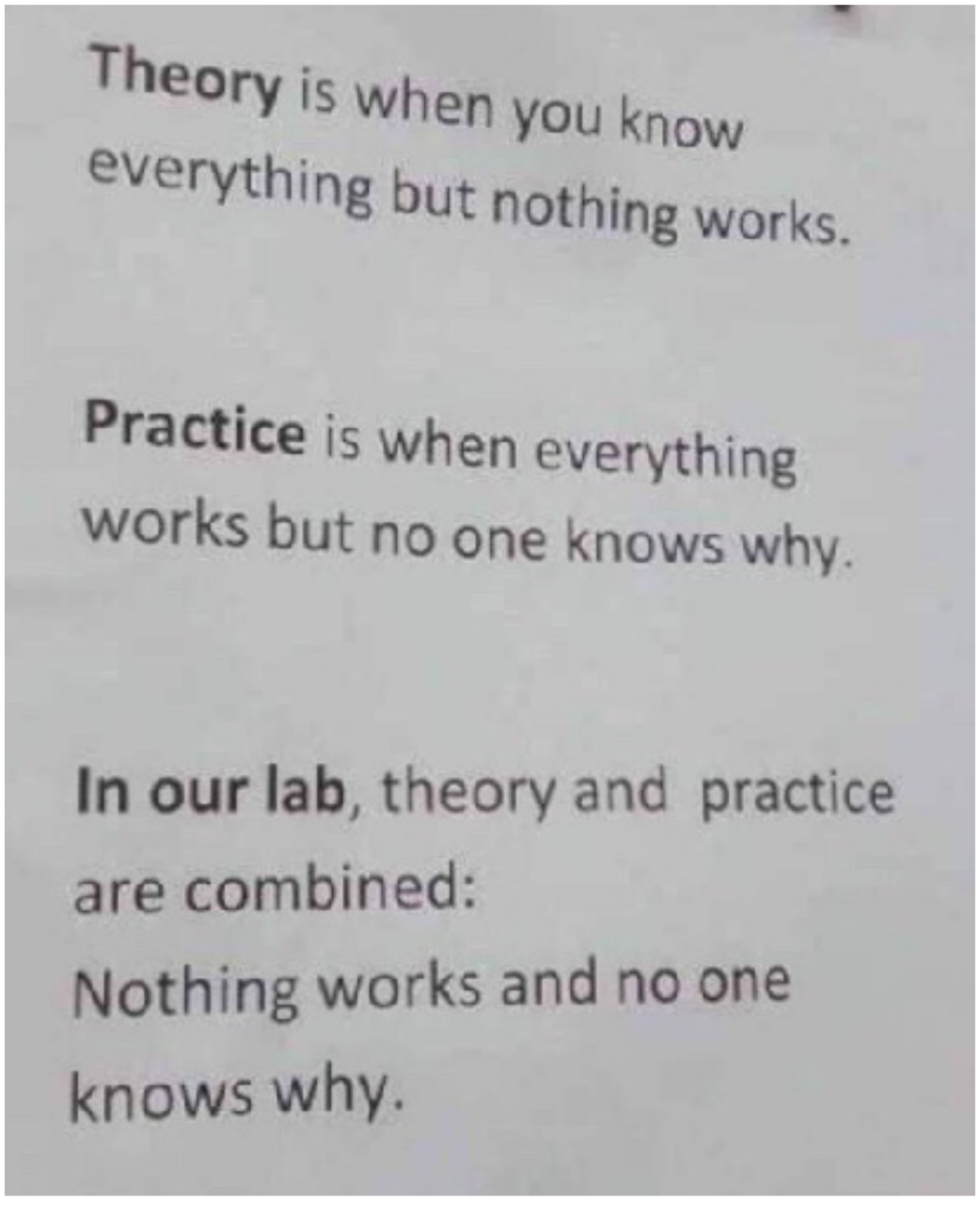
My daughter's door.

http://bit.ly/minsky_racket

How to Design Programs + Student Languages + DrRacket



Systematic Program Design VS Tinkering



functional programming, because of the data structures she's using, or because she's teaching higher-level functions? We don't really know what makes programming so hard, and we don't yet have enough theory to explain why it works when we get it right.

- Mark Guzdial (BLOG@CACM: Learning Computer Science is Different than Learning Other STEM Disciplines, Jan. 5, 2018)

Systematic Program Design VS Tinkering

Задача: определять следующий цвет светофора.

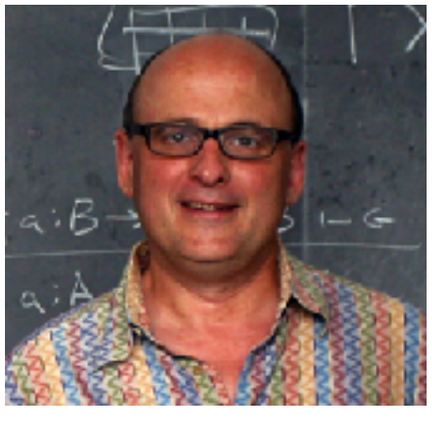
Проектируем по рецепту: сначала данные, потом функцию для них.

Данные

```
;; TLColor is one of:
;;  - 0
;;  - 1
;;  - 2
;; interp. 0 means red, 1 yellow, 2 green
#;
(define (fn-for-tlcolor c)
  (cond [(= c 0) (...)]
        [(= c 1) (...)]
        [(= c 2) (...)]))
```

Функция

```
;; TLColor -> TLColor
;; produce next color of traffic light

(check-expect (next-color 0) 2)
(check-expect (next-color 1) 0)
(check-expect (next-color 2) 1)

;(define (next-color c) 0)  ;stub

; Template from TLColor

(define (next-color c)
  (cond [(= c 0) 2]          ; red -> green
        [(= c 2) 1]          ; green -> yellow
        [(= c 1) 0]))        ; yellow -> red
```



Роберт Харпер (StandardML) одобряет

Systematic Program Design

Рецепты снижают когнитивную нагрузку:

- Начинаем с известных шагов, откладывая реализацию максимально на потом.
- Структура данных определяет её обработку.
- Примеры (они же тесты) — неотъемлемая часть проектирования.
- Неявно мыслим типами данных с самого начала.

*What HtDP teaches is the principle of **data-driven design**. This is the philosophy that underlies both functional and object-oriented programming, but it is rarely articulated clearly or well.*

https://qr.ae/TW4tUA

Systematic Program Design

А что если…

- Алгебраические типы вместо комментариев.
- Синтаксис а-ля Python.
- IDE прямо в браузере.
- Программы доступны онлайн, легко делиться с друзьями.
- Гибкий набор возможностей для разных стилей преподавания.
- Driven by the HtDP philosophy.
- Независимость от индустрии (язык можно развивать без оглядки на легаси).

code.pyret.org/editor#program=1B6fECCg-1MNBPtZE9DyqvTU

File (simple_ufo_landing) | Insert | Publish | Run | Stop

```
include reactors
include image

UFO = image-url("https://cs.brown.edu/~sk/tmp/ufo.png")
BG = rectangle(400, 400, "solid", "white")
INIT = { x: 100, y: 10 }

type Pos = { x :: Number, y :: Number }

fun update(p :: Pos) -> Pos:
  { x: p.x,
    y: p.y + 5 }
end

fun drawer(p :: Pos) -> Image:
  place-image(UFO, p.x, p.y, BG)
end

fun stopper(p :: Pos) -> Boolean:
  (p.y < 0) or (p.y > 250)
end

fun typer(p :: Pos, k :: String) -> Pos:
  ask:
    | k == "up" then: {x: p.x, y: p.y - 10}
    | k == "down" then: {x: p.x, y: p.y + 10}
    | otherwise: p
  end
end

r = reactor:
  init: INIT,
```

```
reactor({x: 100, y: 255})
```

Looks shipshape, your test passed, mate!

update — Show Details

The test in this block passed.

›››

Reactor

Systematic Program Design with Types

```
data TLColor:
     | Red
     | Yellow
     | Green
end

fun next-color(c :: TLColor) -> TLColor:
  cases (TLColor) c:
     | Red => Green
     | Green => Yellow
     | Yellow => Red
  end
end

examples:
  next-color(Red) is Green
  next-color(Yellow) is Red
end
```

```
fun advice(tc :: TLColor) -> String:
    cases(TLColor) tc:
        | Red => "Wait!"
        | Yellow => "Get ready!"
        | Green => "Go!"
    end
where:
    advice(Red) is "Wait!"
    advice(Yellow) is "Get ready!"
    advice(Green) is "Go!"
end
```

https://code.pyret.org/editor#share=1ccaiQCJztil5xH08StmzfSt1VBg-QJFh&v=36e00a0

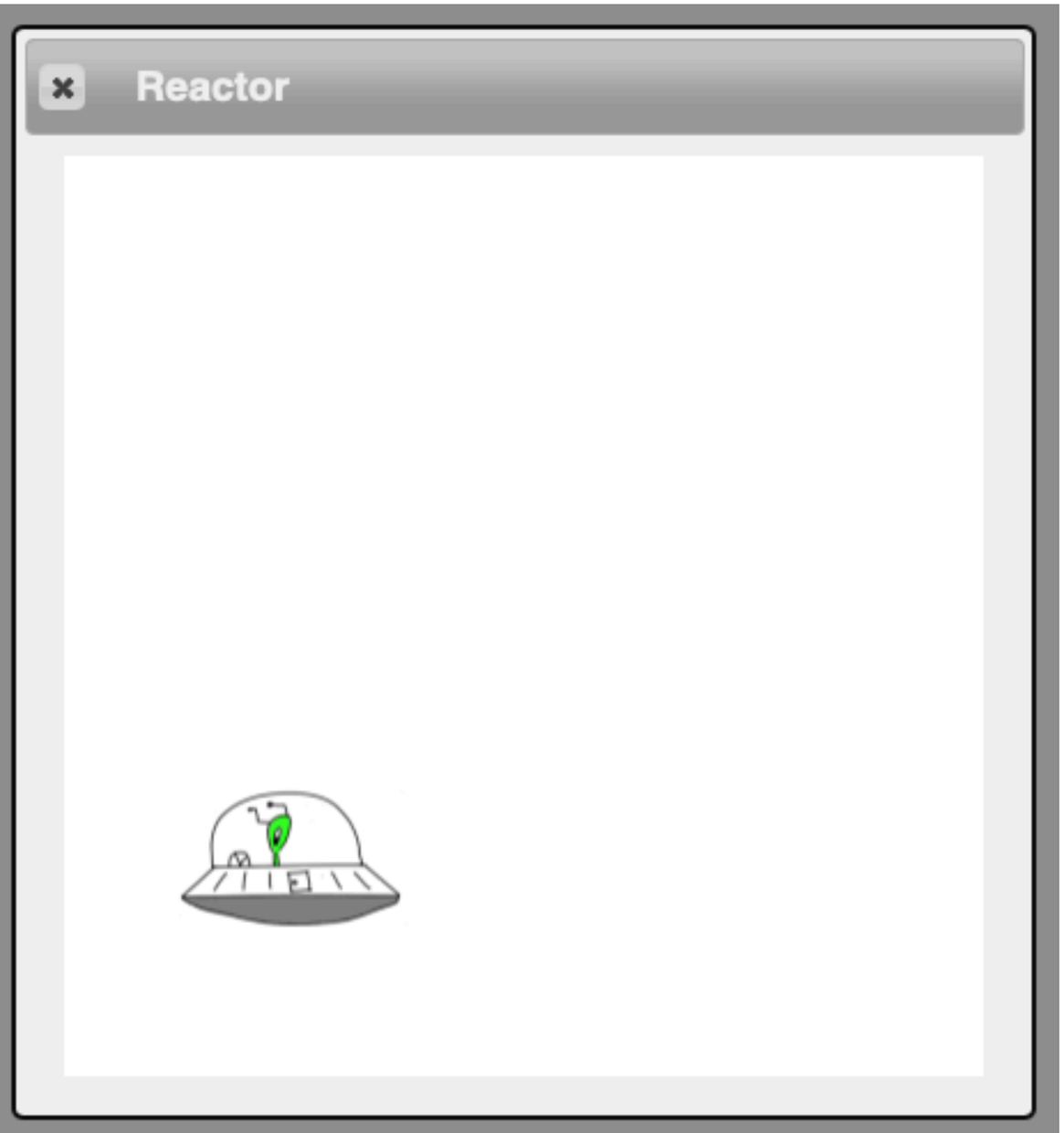
Работа с данными: таблицы, анализ

```
include math
include statistics

playlist = table: name, artist, plays
    row: 'Silver Bride', 'Amorphis', 31
    row: 'Supergood', 'Leningrad', 3
    row: 'The Man Who Sold the World', 'Nirvana', 1
    row: 'Where do I Belong', 'Infected Mushroom', 7
    row: "Why Don't You Get a Job", 'The Offspring', 1
    row: 'Sampo', 'Amorphis', 27
end

play-counts = extract plays from playlist end
most-played-qty = max(play-counts)

most-played-song = sieve playlist using plays:
  plays == most-played-qty
end

print(most-played-song) # row with 'Silver Bride' song
```

https://code.pyret.org/editor#share=1NvsqEwr0nUcOWOuike6e18mkWKZ-qybC&v=36e00a0

Анимация: Event Loops as First-Class Values

```
type Pos = { x :: Number, y :: Number } # Model

fun update(p :: Pos) -> Pos:
  { x: p.x,
    y: p.y + 5 }
where:
  update({x: 2, y: 3}) is {x: 2, y: 8}
end

fun drawer(p :: Pos) -> Image:
  place-image(UFO, p.x, p.y, BG)
end

fun stopper(p :: Pos) -> Boolean:
  (p.y < 0) or (p.y > 300)
end

fun typer(p :: Pos, k :: String) -> Pos:
  ask:
    | k == "up" then: {x: p.x, y: p.y - 20}
    | k == "down" then: {x: p.x, y: p.y + 20}
    | otherwise: p
  end
end

r = reactor: # Event Loop
  init: INIT,
  on-tick: update,
  to-draw: drawer,
  stop-when: stopper,
  on-key: typer
end

interact(r) # Controller
```



https://code.pyret.org/editor#share=1fClH_aSYIuNYrg3V4Q34UCfg_5rZqw6c&v=36e00a0

FUNCTIONAL PROGRAMMING FOR K-12

BOOTSTRAP

www.bootstrapworld.org

http://bootstrapworld.org

Учебные материалы

- pyret.org — сайт языка.
- papl.cs.brown.edu — основной учебник.
- Курс лекций Accelerated Introduction to CS от Brown University (есть видео).

Методика

- Bootstrapworld — набор методических материалов для старших классов.
- Curriculum Design as an Engineering Problem — Шрирам о проектировании учебных курсов.

Прочее

- В. Брагилевский про выбор первого языка программирования — в т.ч. про HtDP, Racket и Pyret.
- Программирование для начинающих: методики и языки — моя статья по теме.

http://andreymiskov.ru

https://github.com/amiskov