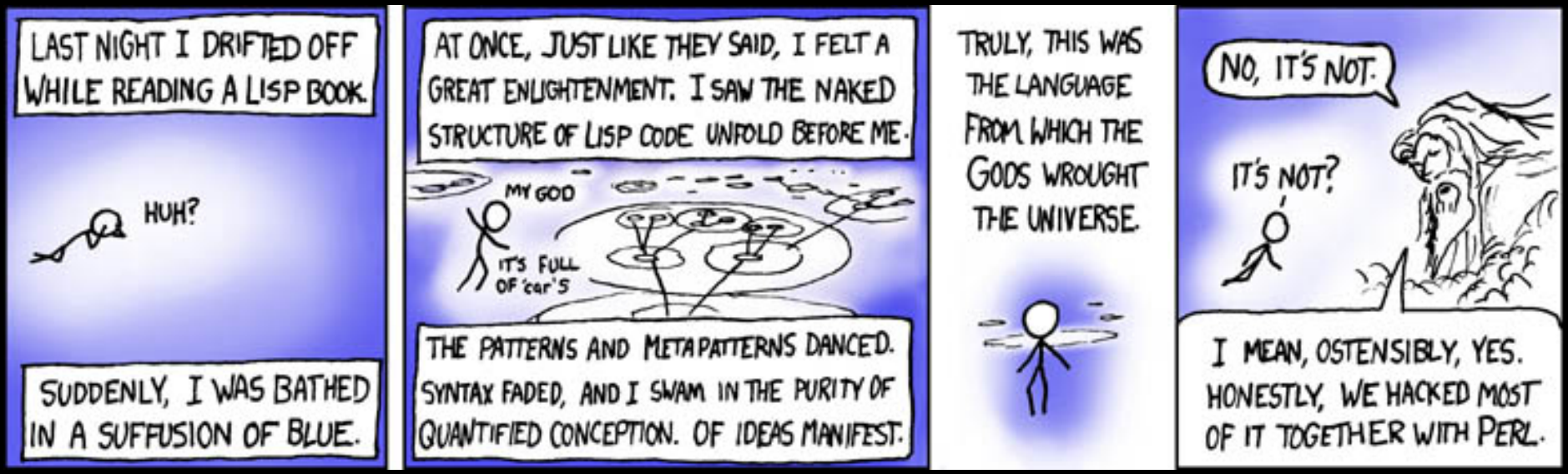
http://xkcd.com/224/



CS 152: *Programming Language Paradigms*

Prof. Tom Austin

San José State University

What are some programming languages?

Taken from http://pypl.github.io/PYPL.html
January 2016

Why are there so many?

Different domains

Different design choices

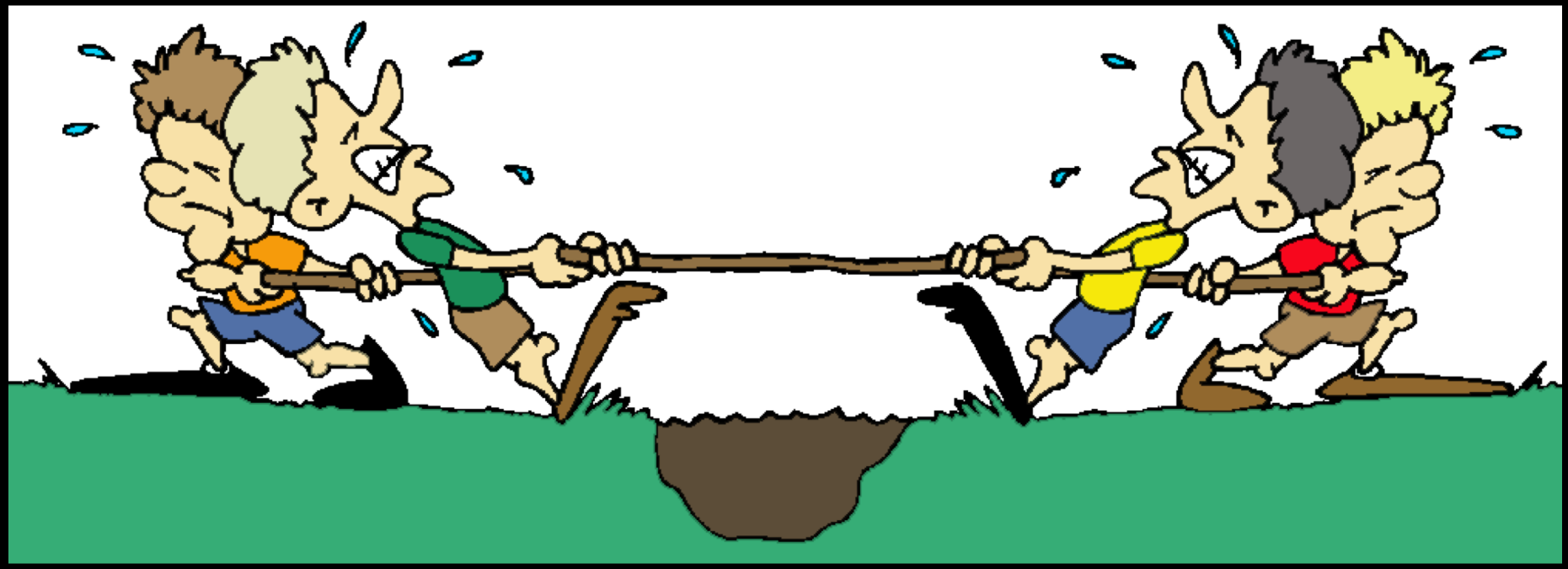
- Flexibility
- Type safety
- Performance
- Build time
- Concurrency

Which language is better?

Good language features

- Simplicity
- Readability
- Learn-ability
- Safety
- Machine independence
- Efficiency

These goals almost always conflict



Conflict: Type Systems

Stop "bad" programs

… *but* ...

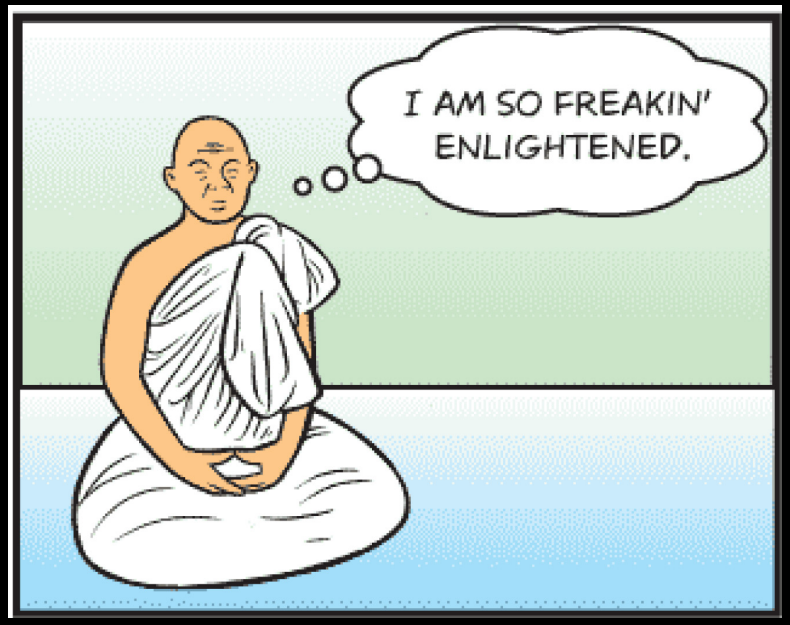
restrict the programmer

Why do we make you take a programming languages course?

- You **might use one** of these languages.
- Perhaps one of these languages is the ***language of the future*** (whatever that means).
- You might see **similar languages** in your job.
- Somebody made us take one, so now we want to make you suffer too.
- But most of all…


We want to warp your minds.

Course goal: change the way that you think about programming.



That will make you a better *Java* programmer.

The "Blub" paradox

"As long as our hypothetical Blub programmer is looking down the power continuum, he knows he's looking down…

[Blub programmers are] satisfied with whatever language they happen to use, because it dictates the way they think about programs."

--Paul Graham

http://www.paulgraham.com/avg.html

So what kind of abstractions do different languages give us?

Data Abstractions

- Representation of common data values
- Variables to represent memory locations.
- Structures: arrays, text files, etc.
- Organizational units:
  - APIs
  - packages
  - classes

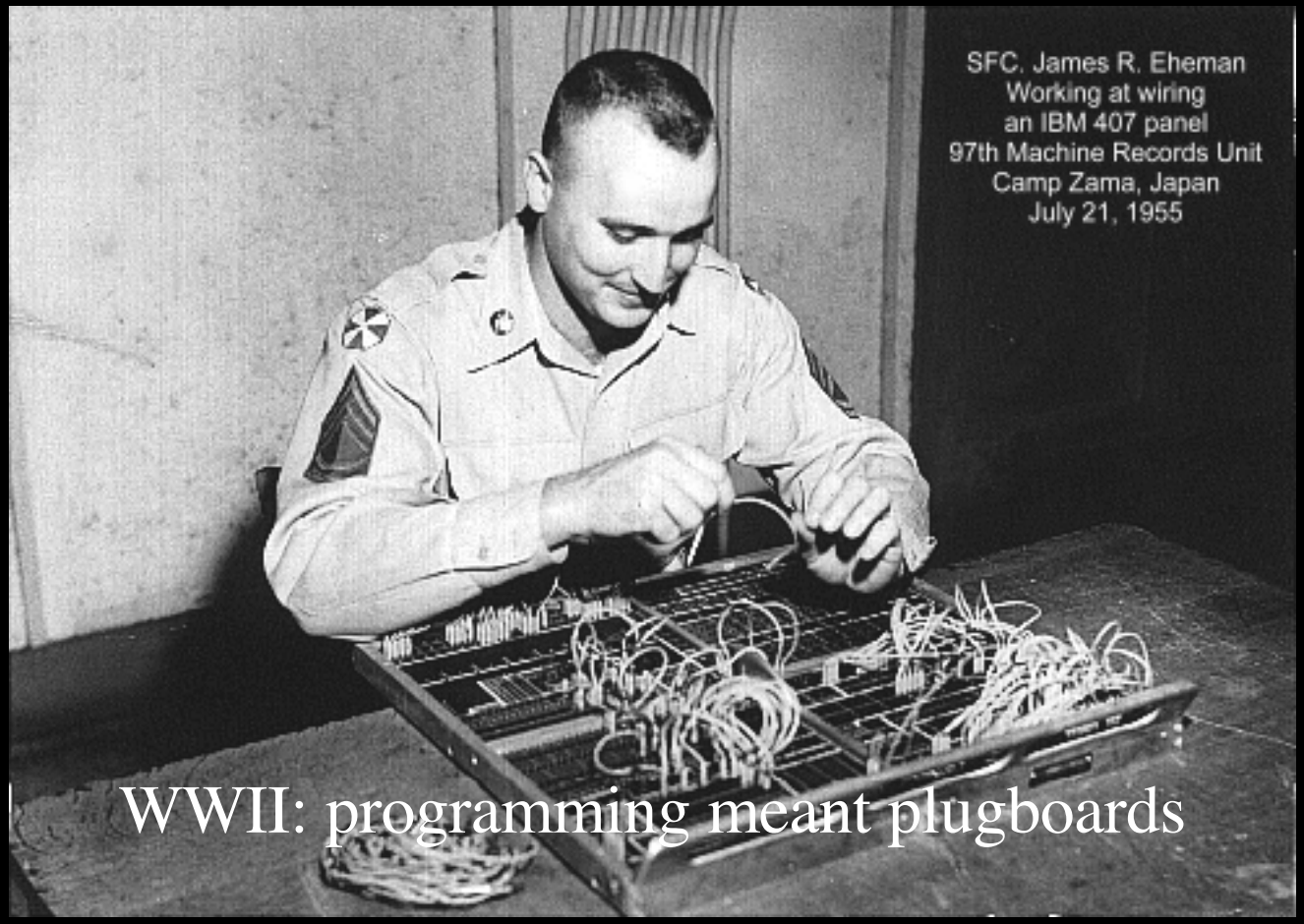
Control Abstractions

- "Syntactic sugar"

  `x+=10` **=>** `x=x+10`

- Branch instructions
  - if statements
  - while loops

A little history



WWII: programming meant plugboards

Von Neumann (late 1940s) created a permanently hardwired general purpose instruction set.

A small, machine language program

```
0010001000000100
0010010000000100
0001011001000010
0011011000000011
1111000000100101
0000000000000101
0000000000000110
0000000000000000
```

Assembly (1950s)

- Translates mnemonic symbols to machine instructions.
- Not a higher level – just easier for humans to read.

```
mov ax, 1234h
mov bx, ax
```

Higher-level languages

- Fortran (1950s):
  - Algebraic notation, floating point math
- Algol (1960):
  - Machine independence (via compiler)
  - Control structures: loops, procedures, if statements
  - Formal specification

Does a formal specification matter?

Languages we will cover
(subject to change)

Administrative Details

- Green sheet: http://www.cs.sjsu.edu/~austin/cs152-fall18/Greensheet.html.
- Homework submitted through Canvas: https://sjsu.instructure.com/
- Academic integrity policy: http://info.sjsu.edu/static/catalog/integrity.html

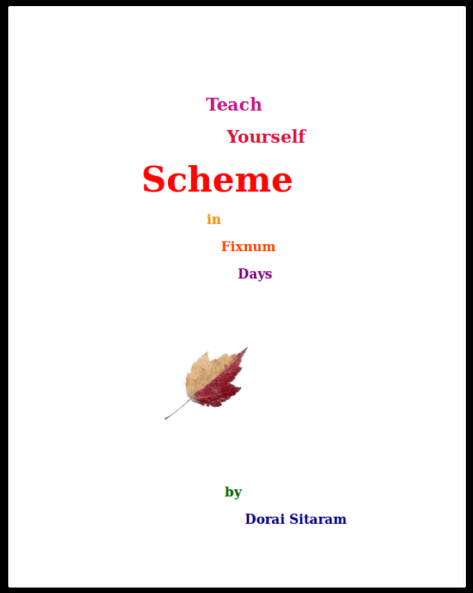
Schedule

- The class schedule is available through Canvas
- Late homeworks will not be accepted
- Check the schedule before every class
- Check the schedule before every class
- And finally, CHECK THE SCHEDULE BEFORE EVERY CLASS.

Prerequisites

- **CS 151 or CMPE 135**, *grade C- or better*
- Show me proof
  - –If you don't, I will drop you.

Resources



Dorai Sitaram
"Teach Yourself Scheme
in Fixnum Days".
http://ds26gte.github.io/tyscheme/

Other references TBD.

Grading

- 30% -- Homework assignments (individual work)
- 20% -- Class project (team work)
- 20% -- Midterm
- 20% -- Final
- 10% -- Participation (labs and drills)

Participation: Labs

- No feedback given (usually)
- I will look at them
- If you have questions, ask me

Homework

- Done *individually*.
- You may *discuss* the assignment with others.
- **Do your own work!**

How to fail yourself and your friend

If two of you turn in similar assignments:

you both get a 0

Project

- Work in groups of up to four people.
- Goal: Build an interpreter.
- Use Java and ANTLR

Office hours

- MacQuarrie Hall room 216
  - Phone being fixed
- Mondays & Wednesdays, 10-11am

Racket/ Scheme

What is Scheme?

- A functional language
  - Describe what things are, not how to do them.
  - More mathematical compared to imperative langs.
- A dialect of Lisp (**Lis**t **P**rocessing)
- (Famously) minimal language
- Racket is a dialect of Scheme

Symbolic Expressions (s-expressions)

The single datatype in Scheme. Includes:

- **Primitive types:** booleans, numbers, characters, and *symbols*.
- **Compound data types:** strings, vectors, pairs, and of course…

LISTS!!!

```
$ racket
Welcome to Racket v6.0.1.
> '(1 2 3 4)
'(1 2 3 4)
> (car '(1 2 3 4))
1
> (cdr '(1 2 3 4))
'(2 3 4)
>  (+ 1 (* 2 4) (- 5 1))
13
>
```

Quote indicates list is data

First element is assumed to be a function

Before next class

- Install Racket from http://racket-lang.org/
- Read chapters 1-2 of *Teach Yourself Scheme*.
- Read Paul Graham's "Beating the Averages" article. http://www.paulgraham.com/avg.html

First homework due June 18th

- This assignment is designed to get you up and running with Racket.
- Available in Canvas.
  - If you don't have access to Canvas, see http://www.cs.sjsu.edu/~austin/cs152-summer18/hw/hw1/ instead.
- Get started now!