CSCI 431: Principles of Functional Programming

Welcome and Introduction

Mukund Raghothaman

Fall 2025

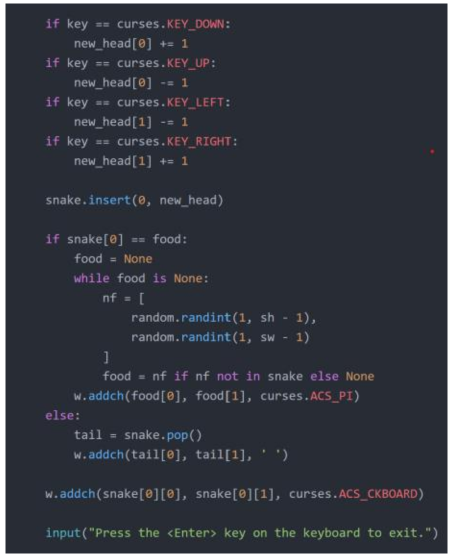
About This Course

- We’ll study several different **programming languages**
- What languages do you know?
- What makes them different?

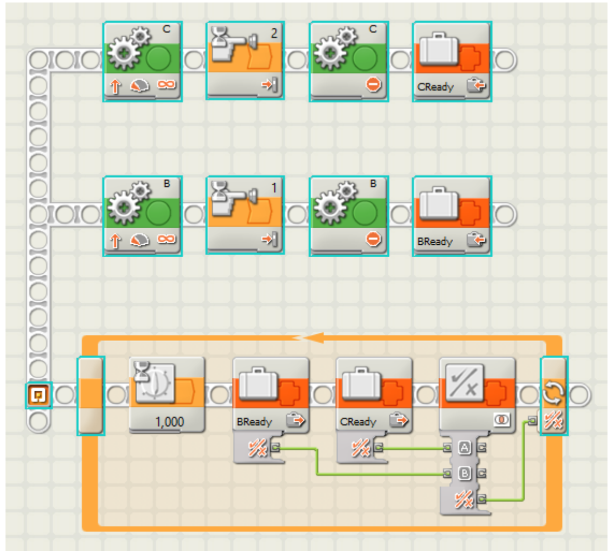
About This Course

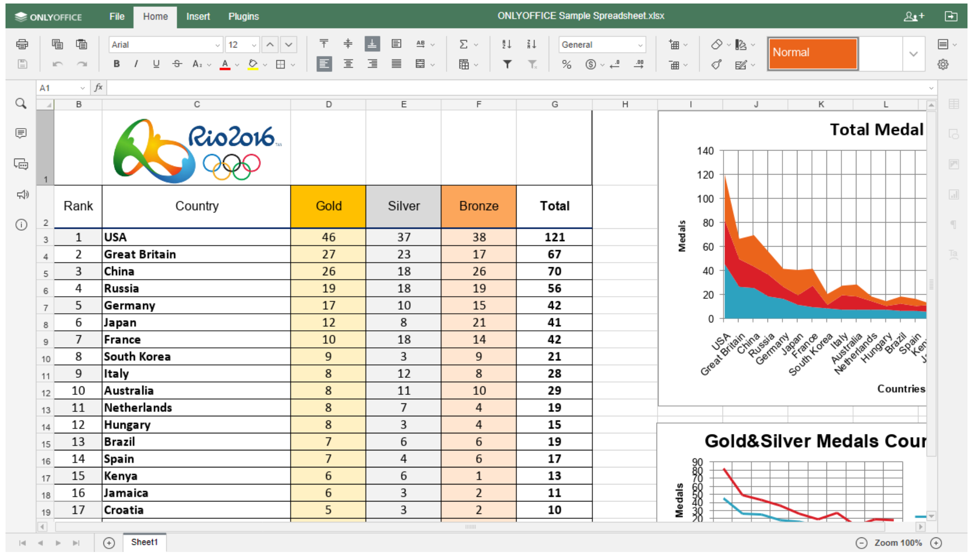
- What is a programming language?

- **Notation for specifying computations**



```
if key == curses.KEY_DOWN:
    new_head[0] += 1
if key == curses.KEY_UP:
    new_head[0] -= 1
if key == curses.KEY_LEFT:
    new_head[1] -= 1
if key == curses.KEY_RIGHT:
    new_head[1] += 1

snake.insert(0, new_head)

if snake[0] == food:
    food = None
    while food is None:
        nf = [
            random.randint(1, sh - 1),
            random.randint(1, sw - 1)
        ]
        food = nf if nf not in snake else None
    w.addch(food[0], food[1], curses.ACS_PI)
else:
    tail = snake.pop()
    w.addch(tail[0], tail[1], ' ')

w.addch(snake[0][0], snake[0][1], curses.ACS_CKBOARD)

input("Press the <Enter> key on the keyboard to exit.")
```

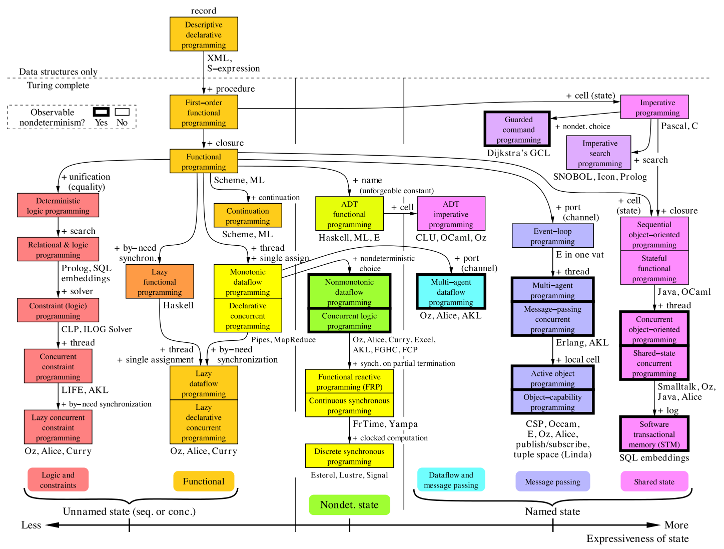




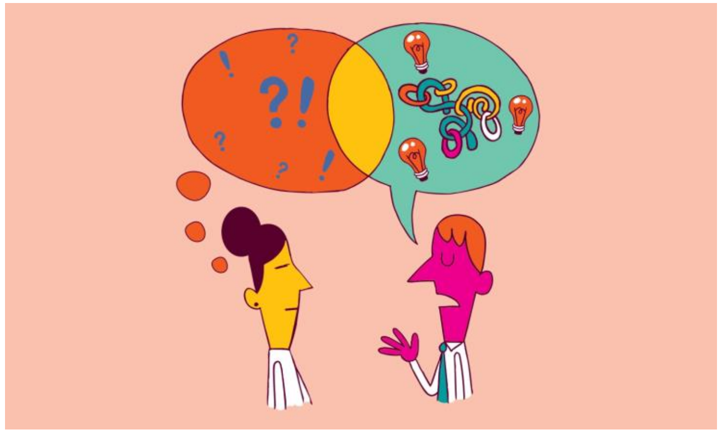
Rank	Country	Gold	Silver	Bronze	Total
1	USA	46	37	38	121
2	Great Britain	27	23	17	67
3	China	26	18	26	70
4	Russia	19	18	19	56
5	Germany	17	10	15	42
6	Japan	12	8	21	41
7	France	10	18	14	42
8	South Korea	9	3	9	21
9	Italy	8	12	8	28
10	Australia	8	11	10	29
11	Netherlands	8	7	4	19
12	Hungary	8	3	4	15
13	Brazil	7	6	6	19
14	Spain	7	4	6	17
15	Kenya	6	6	1	13
16	Jamaica	6	3	2	11
17	Croatia	5	3	2	10

About This Course

- Why so many languages?



record
Descriptive declarative programming
XML, S-expression
Data structures only
Turing complete
+ procedure
Observable nondeterminism? Yes No
First-order functional programming
+ cell (state)
Imperative functional programming
Pascal, C
Guarded command programming
+ nondet. choice
Dijkstra's GCL
Imperative search programming
+ search
SNOBOL, Icon, Prolog
+ closure
Functional programming
Scheme, ML
+ unification (equality)
Deterministic logic programming
+ search
Relational & logic programming
Prolog, SQL embeddings
+ solver
Constraint (logic) programming
CLP, ILOG Solver
+ thread
Concurrent constraint programming
LIFE, AKL
+ by-need synchronization
Lazy concurrent constraint programming
Oz, Alice, Curry
+ continuation
Continuation programming
Scheme, ML
+ by-need synchron.
Lazy functional programming
Haskell
+ thread
+ single assign.
Monotonic dataflow programming
Declarative concurrent programming
Pipes, MapReduce
+ thread
+ single assignment
+ by-need synchronization
Lazy dataflow programming
Lazy declarative concurrent programming
Oz, Alice, Curry
+ name (unforgeable constant)
ADT functional programming
Haskell, ML, E
+ cell
ADT imperative programming
CLU, OCaml, Oz
+ nondeterministic choice
Nonmonotonic dataflow programming
Concurrent logic programming
Oz, Alice, Curry, Excel, AKL, FGHC, FCP
+ synch. on partial termination
Functional reactive programming (FRP)
Continuous synchronous programming
FrTime, Yampa
+ clocked computation
Discrete synchronous programming
Esterel, Lustre, Signal
+ port (channel)
Multi-agent dataflow programming
Oz, Alice, AKL
+ port (channel)
Event-loop programming
E in one vat
+ thread
Multi-agent programming
Message-passing concurrent programming
Erlang, AKL
+ local cell
Active object programming
Object-capability programming
CSP, Occam, E, Oz, Alice, publish/subscribe, tuple space (Linda)
+ cell (state)
+ closure
Sequential object-oriented programming
Stateful functional programming
Java, OCaml
+ thread
Concurrent object-oriented programming
Shared-state concurrent programming
Smalltalk, Oz, Java, Alice
+ log
Software transactional memory (STM)
SQL embeddings
Logic and constraints
Functional
Dataflow and message passing
Message passing
Shared state
Unnamed state (seq. or conc.)
Nondet. state
Named state
Less
More
Expressiveness of state



Why So Many Languages?

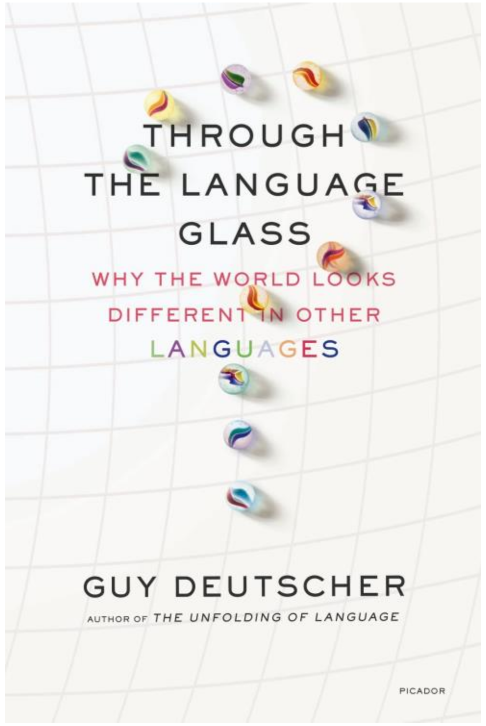
- Different languages for different tasks
  - Verilog for describing hardware
  - Assembly for low-level computations
  - JavaScript for manipulating the DOM

- **Question:** How might one program a quantum computer?

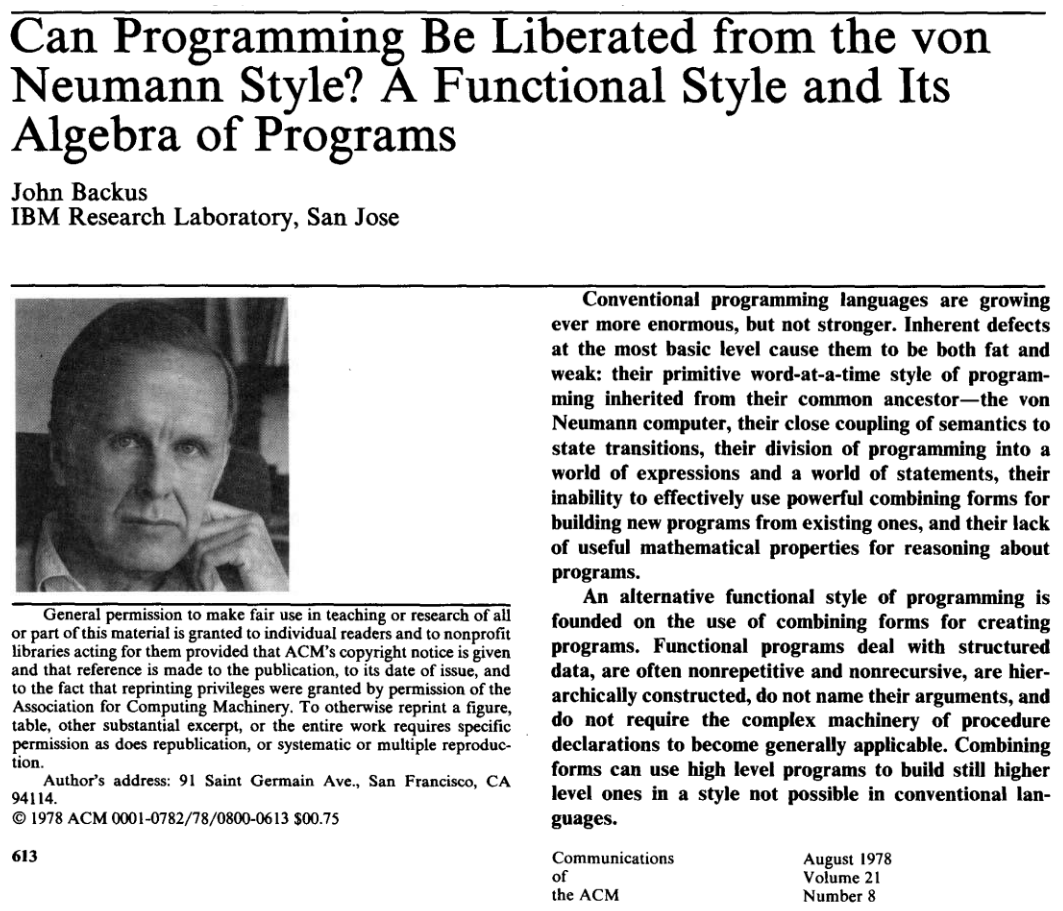
Why Study Programming Languages?

- **Linguistic Relativity:** Structure of a language affects the speaker's thought
- Radiolab podcast on colors in ancient texts: https://radiolab.org/episodes/211213-sky-isnt-blue
- **"A language that doesn't affect the way you think about programming, is not worth knowing."**

  -- Alan Perlis, Epigrams on Programming


THROUGH THE LANGUAGE GLASS
WHY THE WORLD LOOKS DIFFERENT IN OTHER LANGUAGES
GUY DEUTSCHER
AUTHOR OF THE UNFOLDING OF LANGUAGE
PICADOR

Why Study Programming Languages?



Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs

John Backus
IBM Research Laboratory, San Jose

Conventional programming languages are growing ever more enormous, but not stronger. Inherent defects at the most basic level cause them to be both fat and weak: their primitive word-at-a-time style of programming inherited from their common ancestor—the von Neumann computer, their close coupling of semantics to state transitions, their division of programming into a world of expressions and a world of statements, their inability to effectively use powerful combining forms for building new programs from existing ones, and their lack of useful mathematical properties for reasoning about programs.

An alternative functional style of programming is founded on the use of combining forms for creating programs. Functional programs deal with structured data, are often nonrepetitive and nonrecursive, are hierarchically constructed, do not name their arguments, and do not require the complex machinery of procedure declarations to become generally applicable. Combining forms can use high level programs to build still higher level ones in a style not possible in conventional languages.

Author's address: 91 Saint Germain Ave., San Francisco, CA 94114.

Our Goals in this Course

- Make you **better programmers** …
- … by exposing you to **powerful new languages** and programming **constructs**

- **Demystify** some of the magic

howstuffworks

- Make you **informed leaders** who can influence technical decisions
- Change the way you **think** about computation

Ideas to Take Away

Unit 1
Functional Programming

1. Immutable data
2. First-class functions
3. Static types
4. Pattern matching

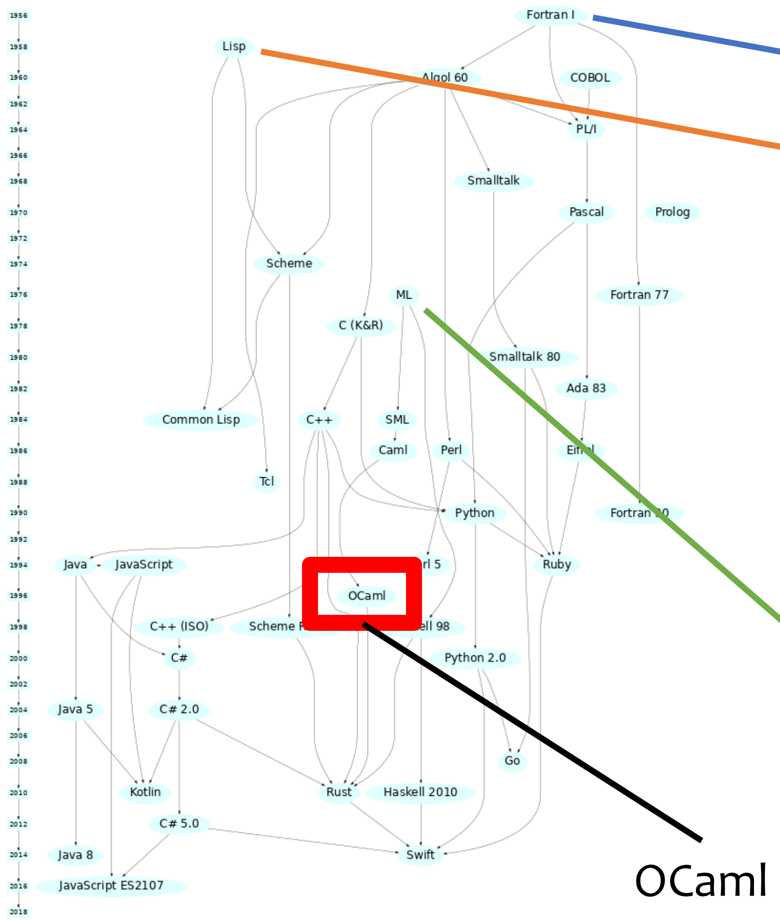
Unit 2
Language Implementation

1. Processing syntax
2. Type inference
3. Garbage collection

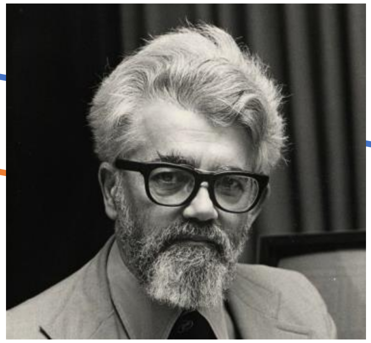
Unit 3
Relational Programming

1. Spreadsheets and DAGs
2. Declarative languages
3. Recursive queries

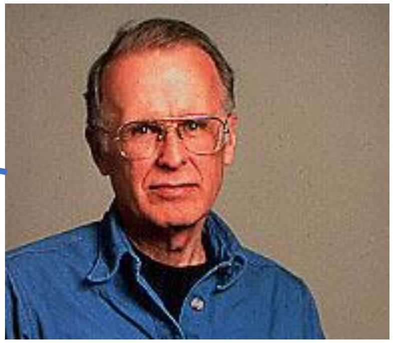
History of Programming Languages



[Pascal Rigaux]



John McCarthy
1927—2011
Turing Award 1971



John Backus
1924—2007
Turing Award 1977

Robin Milner
1934—2010
Turing Award 1991

Why Functional Programming?

- **Encourage immutability**
  Programs are easier to think about
- **Algebraic data types and pattern matching**
  Elegant ways to construct and destruct data
- **First-class functions**
  Functions can be passed around just like values
- **Static type checking**
  Programs have fewer bugs
- **Automatic type inference**
  Make the compiler work for you
- **Parametric polymorphism**
  Can generalize computation across many types
- **Garbage collection**
  Make the runtime work for you
- **Modules**
  Elegant ways of structuring large systems

Functional Influences on Programming Practice

- Garbage collection:
  Lisp (1958) → Python (1990), Java (1995)

- Parametric polymorphism / Generics / Templates:
  ML (1975) → C++ (1986), Java (2004)

- Higher-order functions:
  Lisp (1958) → C# (2007), C++ (2011), Java (2014)

- Type inference:
  ML (1982) → C++ (2011), Java (2011), TypeScript (2012)

- Linear and affine types, resource ownership:
  Linear logic (1987) → Cyclone (2002) → C++ unique_ptr (2011), Rust (2006)

Functional Programming in Industry

- **OCaml**: Jane Street, Bloomberg, Citrix
- **Scala**: X / Twitter, Foursquare, LinkedIn
- **Haskell**: Facebook, Barclays, AT&T
- **Erlang**: WhatsApp, Amazon, T-Mobile

- See StackOverflow Developer Survey for economic motivation 🤑

- More recently: Lean + Mathlib, AlphaProof

Today's Plan

- Motivation and Overview
- **Course Logistics**
- Diving into OCaml

Textbooks

- Yaron Minsky, Anil Madhavapeddy, and Jason Hickey
  Real World OCaml
- Michael Clarkson
  OCaml Programming: Correct + Efficient + Beautiful
- Harold Abelson, Gerald Jay Sussman, and Julie Sussman
  Structure and Interpretation of Computer Programs
- All books are open access

Classes and Office Hours

- Tuesdays and Thursdays
- 4pm—5:50pm Los Angeles time
- Will be streamed over Zoom, recorded*

- Website: https://**r-mukund.github.io**/teaching/fa2025-csci431/

- Office Hours:
  - Thursdays, 1:30pm—3:30pm
  - Or by appointment

Assessment (1)

- 3 homework assignments × 17% each = 50%
- 2 quizzes and 1 final exam, 17% each = 50%
  - You are allowed to consult one handwritten letter paper-sized crib sheet

- Welcome to collaborate with a partner on homeworks
- **But! Identify your partner, write answers by yourselves**

- GPT / LLM policy
  - **Welcome to use**
  - But… (1) clearly reflect on why you used it (or not). And how it was helpful. Tell us
  - That said, **(2) this policy is experimental. Subject to change at any time**
  - (3) No technology access during quizzes or exams

Assessment (2)

- HW0 is posted to Brightspace
- Not graded
- Due at 10pm, Monday, 1 September 2025

- Going forward…

- Assignments due 10pm on Monday after unit is complete
- Reference solutions will be posted on Wednesday at 2pm
- Submissions after reference solutions are posted will not be graded
- Talk to me in advance if you are unable to submit an assignment on time

Accommodations

- Talk to me if you:
  - are unable to submit an assignment on time,
  - are finding it difficult to keep up,
  - need any help to make the most of this course, or
  - want to bring anything to my attention

About Me

- Mukund Raghothaman
- PhD from UPenn, 2017
- Joined USC in Fall 2019

- Research Area: “How do we reason about programs?”
- Find bugs; prove correctness; **synthesize** code!
- Can data (i.e., GitHub) help?
- Can we use probabilities and / or machine learning?

Tell Us About Yourselves

- Name, program
- Background in programming
- Languages you have used + Familiarity

For Next Class

- Install OCaml on your computers
- Start work on Homework 0, due 10pm, Monday, 1 September
- **Reading:**
  - RWO, Chapter 1: Guided Tour
  - MC, Chapters 1, 2