

CSCI 431: Principles of Functional Programming

Welcome and Introduction

Mukund Raghothaman

Fall 2026

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



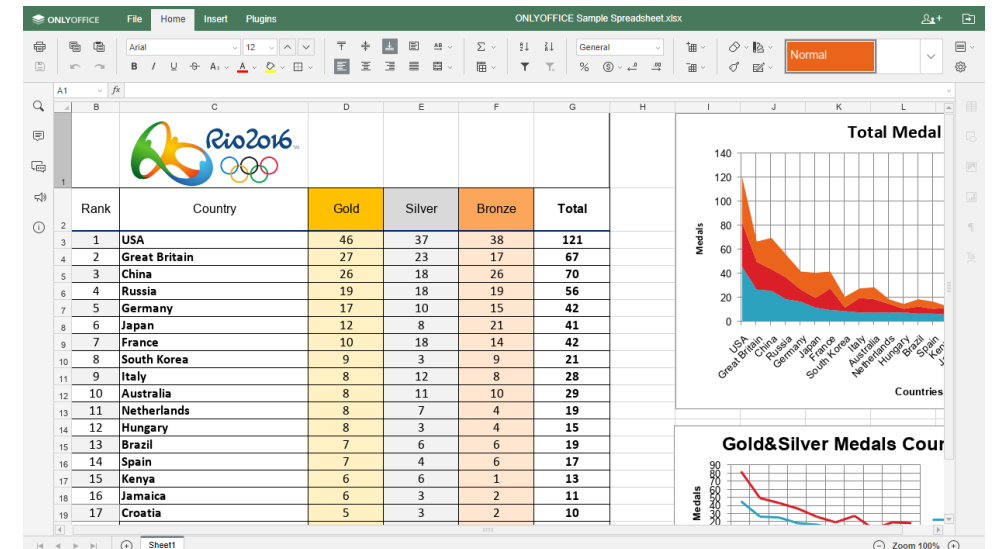
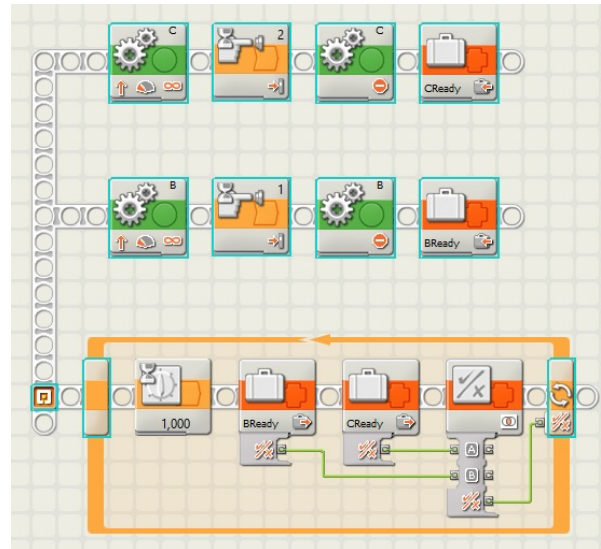
```
if key == curses.KEY_DOWN:
    new_head[0] += 1
if key == curses.KEY_UP:
    new_head[0] -= 1
if key == curses.KEY_LEFT:
    new_head[1] -= 1
if key == curses.KEY_RIGHT:
    new_head[1] += 1

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], ' ')

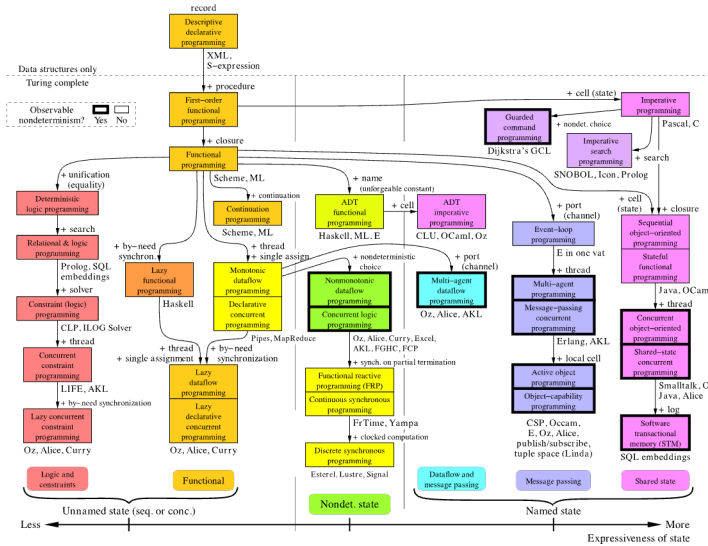
w.addch(snake[0][0], snake[0][1], curses.ACS_CKBOARD)

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



About This Course

- Why so many languages?

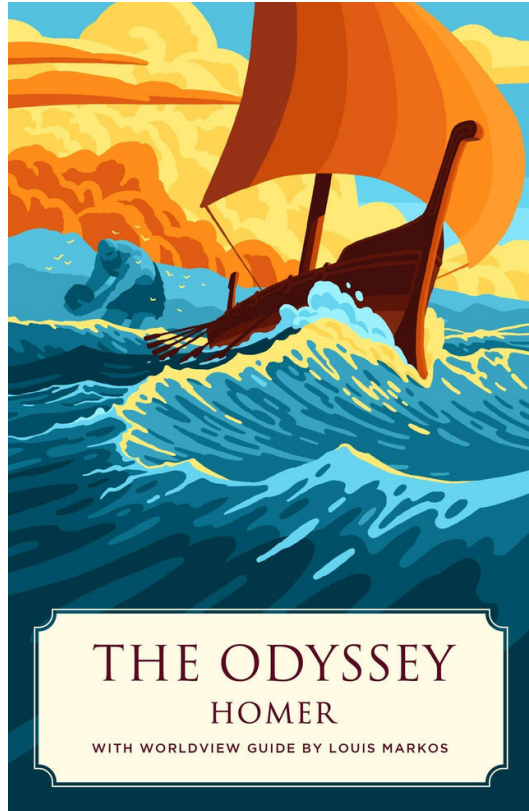


Why So Many Languages?

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



“The Wine-Dark Sea”



Volodymyr Vlasenko. View from Karpathos, near the island of Crete.

“The Wine-Dark Sea”

- Homer uses the word “porphyreos” to describe blood, dark clouds, waves, rainbows, ...
- Porphyreos in later Greek roughly means purple or dark red
- Describes the sea as “oinops” (wine-looking)



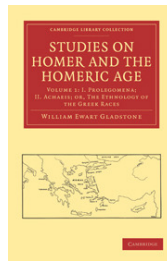
Volodymyr Vlasenko. View from Karpachos, near the island of Crete.

“The Wine-Dark Sea”

- Gladstone: Ancient Greeks categorized colours in terms of lightness, darkness, or saturation, rather than hue
- Conspiracy theory: Homer was colour-blind



Portrait of William Gladstone by George Frederic Watts.

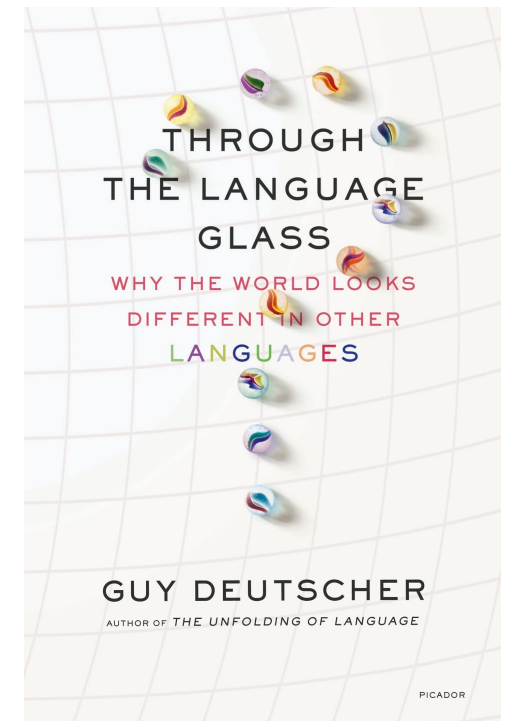


Volodymyr Vlasenko. View from Karpathos, near the island of Crete.

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.”

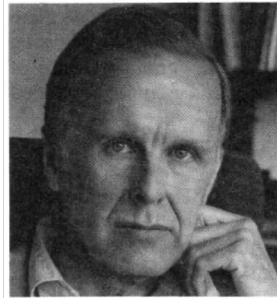
--- Alan Perlis, Epigrams on Programming



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



General permission to make fair use in teaching or research of all or part of this material is granted to individual readers and to nonprofit libraries acting for them provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery. To otherwise reprint a figure, table, other substantial excerpt, or the entire work requires specific permission as does republication, or systematic or multiple reproduction.

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

© 1978 ACM 0001-0782/78/0800-0613 \$00.75

613

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.

Communications
of
the ACM

August 1978
Volume 21
Number 8

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

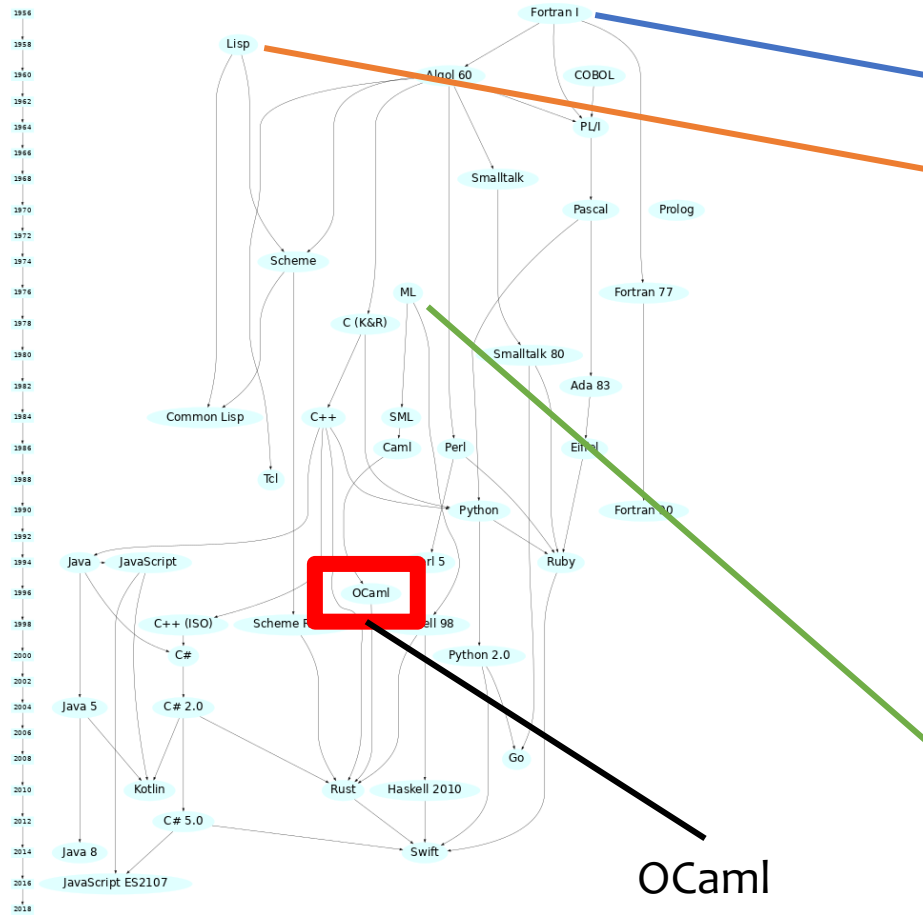
Unit 3

Relational Programming

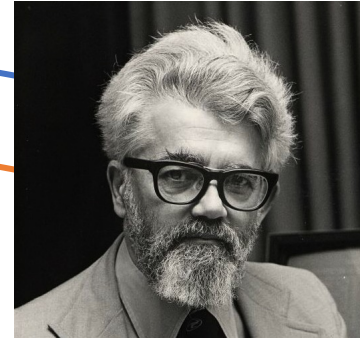
1. Spreadsheets and DAGs
2. Declarative languages
3. Recursive queries



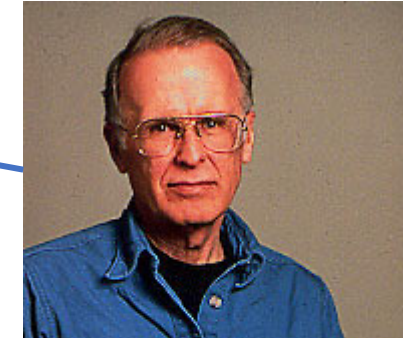
History of Programming Languages



[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, 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/fa2026-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**
- 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**
 - (2) No technology access during quizzes or exams

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 Yourself

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

For Next Class

- Install OCaml on your computers
- **Reading:**
 - RWO, Chapter 1: Guided Tour
 - MC, Chapters 1, 2