CSCI 599: An Introduction to Programming Languages

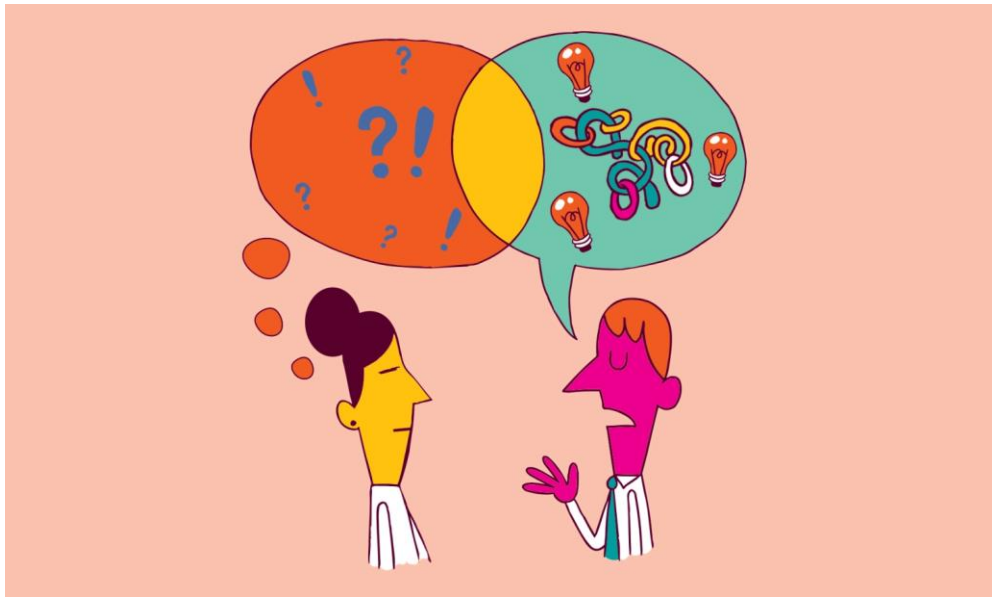
Welcome and Introduction

Mukund Raghothaman

Fall 2020

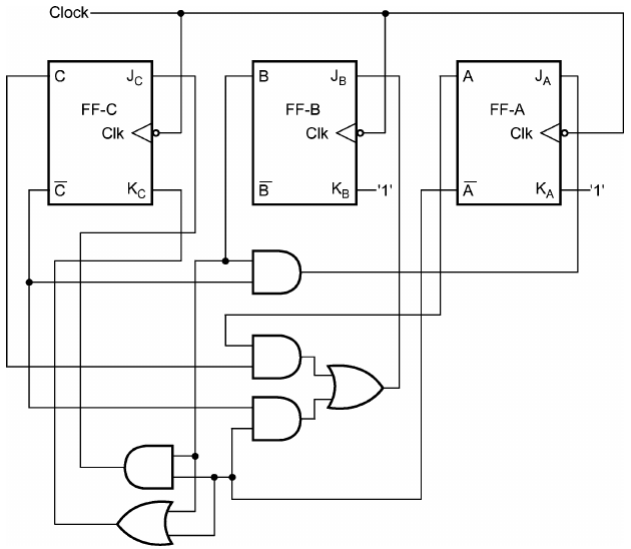
Programming Languages

- System of communication used by a country or community



Programming Languages

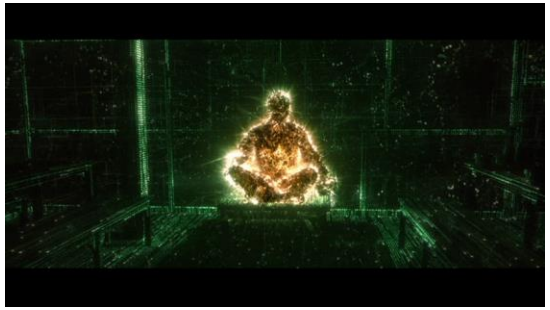
- Notation for specifying a computation




```
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.")
```

Why Study Programming Languages?

- Programming is not hard
- **Programming well is very hard**



- **Linguistic relativity:** Structure of a language affects its speaker's worldview (Controversial)

https://www.wnycstudios.org/podcasts/radiolab/segments/211213-sky-isnt-blue

- **Programming languages shape programming thought** (Dogma for the purposes of this course)

Our Goals in this Course

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

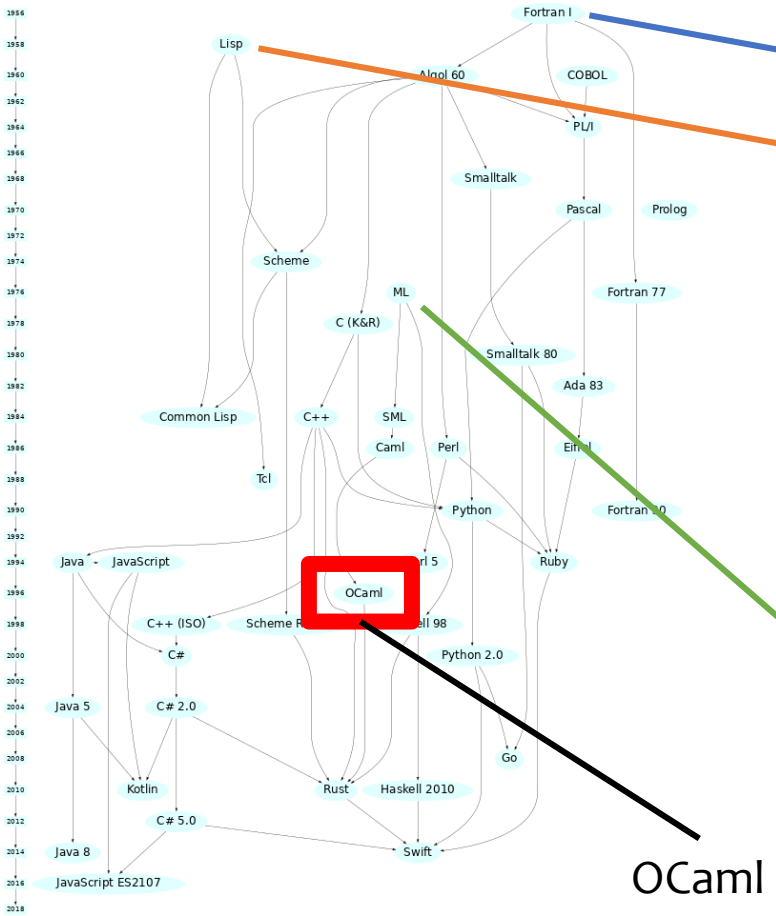
- **Demystify** some of the magic howstuffworks?

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

Course Outline

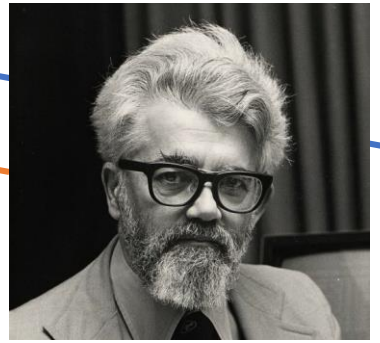
- 3 units, ~4—5 weeks each
- **Functional programming** in Ocaml
- Relational programming: Spreadsheets, SQL and **logic programs**
- Implementation details: Syntax, type systems, runtime (tail call optimization and garbage collection), unification and evaluation algorithms

History of Programming Languages

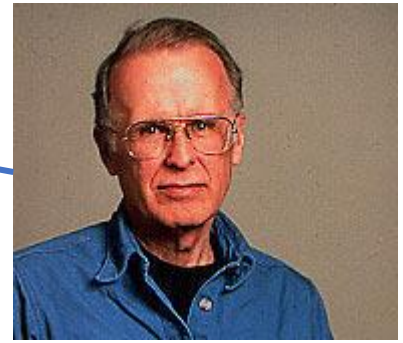


[Pascal Rigaux]

OCaml



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 Languages Predict the Future

- 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)

Functional Programming in Industry

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

Today's Plan

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

Classes and Office Hours

- Tuesdays and Thursdays
- 4pm—6pm Los Angeles time
- Will be recorded

- Website: https://r-mukund.github.io/teaching/fa2020-csci599/
- Zoom: https://usc.zoom.us/j/98960729161
- Piazza: https://piazza.com/usc/fall2020/csci599/home

- Office Hours: Mondays, 4pm—6pm, or by appointment
- Zoom: https://usc.zoom.us/j/95662027167

Evaluation

- 4 homework assignments × 15% each = 60%
- Midterm = 20%
- Final exam = 20%

- All homeworks and exams are take-home
- No collaboration / internet use during exams
- Welcome to collaborate with a partner on homeworks
- But! Identify your partner, write answers by yourselves

Course Staff

- 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 Me About Yourselves

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

- Any functional languages?
- Have you heard of monads, categories, lambda (calculus)?
- This course will not require or cover any of these

Today's Plan

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

What's the Difference Between …?