Scheme/LISP

A functional and dynamically typed language with good support for lists er order functions.

CMPT 383
Summer 2022
SFU Surrey

1975

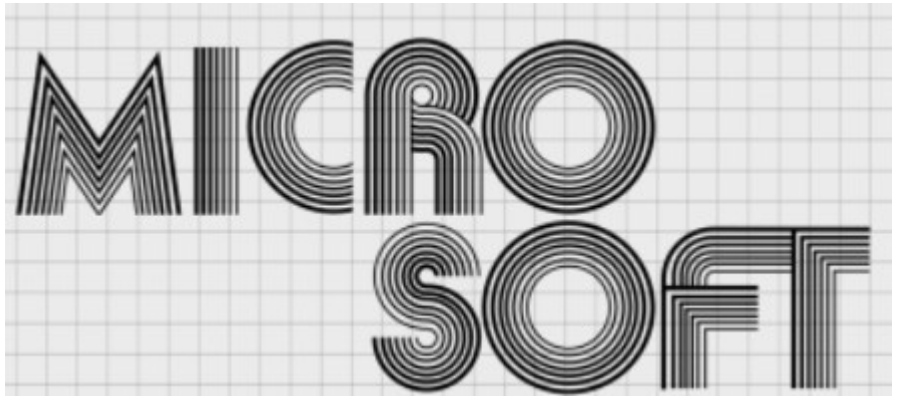
LISP: 1958

What wa in 1975?

1975: First Appearances

Cromemco Cyclops
First all-digital camera
32x32 pixel resolution

Betamax and VHS
Consumer video recorder



Colossal Cave Adventure
First text-based role-playing game

1975: Highest-grossing Arcade Game

Tank / Tank II

e Games (Atari)

1975: Highest-Movie

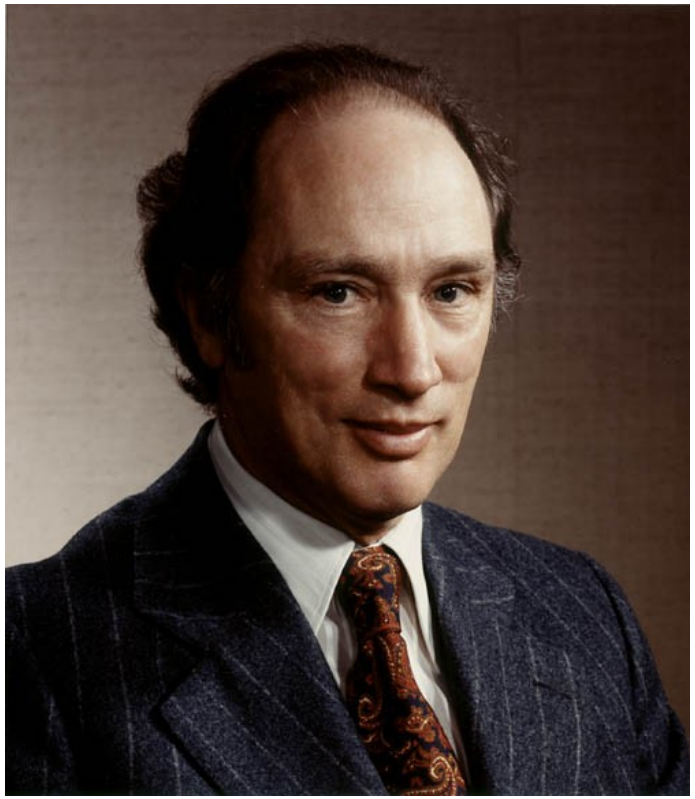
1975: Most Popular Song

Love will Keep us Together

Captain and Tennille

1975

Canadian PM



Pierre Trudeau

US President

Gerald Ford

This TV show started

Saturday Night Live

1975

- Ilton the talking toaster sold Pop Tarts

LISP	1958
Scheme	1975
Racket	~1995

LISP and Scheme's Original Designers

John McCarthy (1927-2011)
Original designer of **LISP** (1958)

Also coined the term **Artificial Intelligence**, and for many years was a leading AI researcher.

Guy Steele and **Gerald Sussman**
Original designers of Scheme (1975)

Racket is a modern dialect of Scheme, with many new ideas added.

Scheme Probably **Can't** Get You a Job!

It's mostly a **research language** used by a subset of academic programming language resea

The original version of **JavaScript** was implemented as a Scheme-style language, before being modified to look more C-like.

A variant of LISP called **Clojure** has mo recently gained some popularity for practical programming (but it's still pretty niche compared to other languages)

Clojure

What's Scheme/LISP Good for?

Traditional **symbolic** AI research.

Inspir

kinds of y

of which have been mainstream lang

Writing programming languages and programming language tools.

What's Scheme Not so Good For?

- Scheme is generally not as efficient as other languages.
  - Dynamically typed
  - Reliance on singly-lists
- Compared to the most popular main languages, Scheme is not used much, so it can be hard to find support for it.
- Scheme's notation is simple, but maybe *too* simple. ()-style notation results in lots and lots of parentheses, which many programmers dislike.

Measuring the performance of different programming languages is surprisingly difficult to do fairly. Check out the Computer Language Benchmarks Game

Major Features of Scheme

Functions are first-class objects

Functions can be passed and returned (including closures)

Interpreted

Easy to evaluate individu

Memory-safe and garbage-collected

No dangling pointers, no manual memory

Homoiconic

Scheme programs are written as Scheme lists

Flexible

Powerful macro system allows for major changes to the language

Major Features of Scheme

Types

- Dynamically typed, i.e. types checked at run-time
- Symbols
- Singly-linked lists are
- Continuations

Lexical Scoping

- One of the first languages to fully implement the lexical scoping rules we now take for granted

Tail Recursion

- Tail-recursive functions are automatically optimized into loops

Major Features Not in Scheme

Features **not** part of Scheme

- No type declarations
- No pointers
- No built-in notion of obj

Features We Won't Be Using

- No loops
- No mutatio ging values of variables
- Instead, we'll use fu al style, which is what our next language (Haskell) will force us to use.

Hello, World!

```
;; hello_world.rkt

(print "Hello, world!")
```


Download and run this in the **DrRacket** IDE. We'll use both the interpreter and file editor in DrRacket.