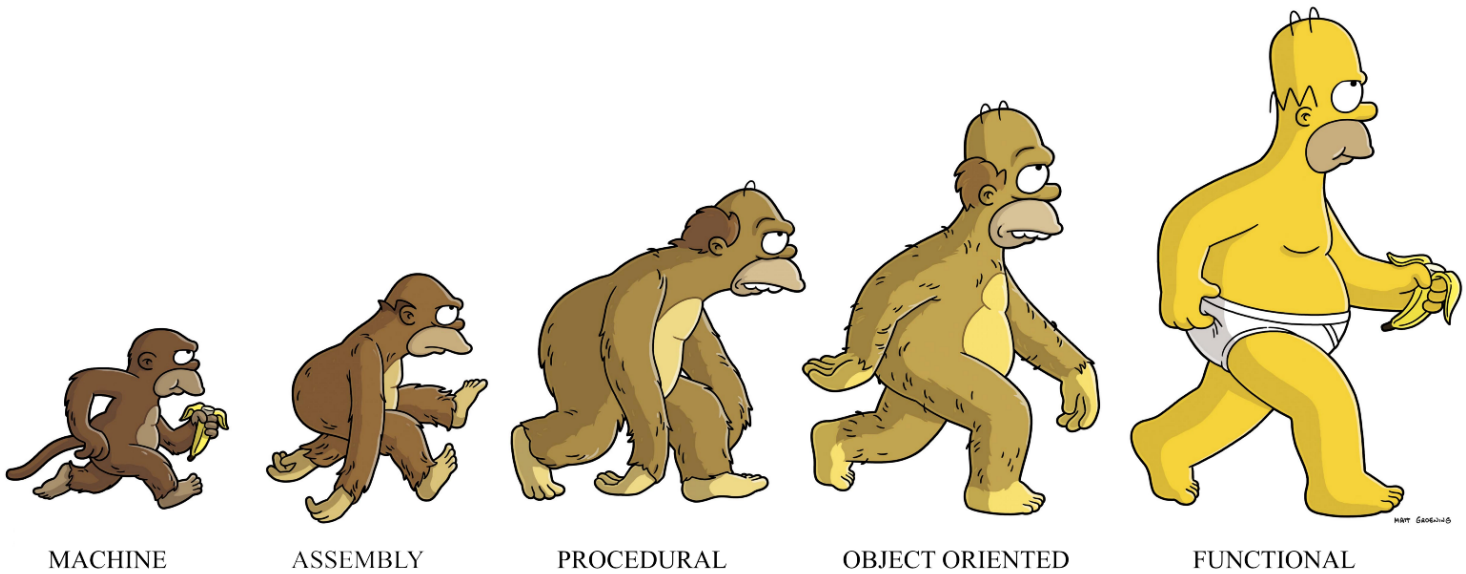
COMP302: Programming Languages and Paradigms

Prof. Brigitte Pientka (Sec 01)
bpientka@cs.mcgill.ca

Francisco Ferreira (Sec 02)
fferre8@cs.mcgill.ca

School of Computer Science
McGill University

Week 7-2, Fall 2017



Functional Tidbit: What's the future of programming?

"If you want to see which features will be in mainstream programming languages tomorrow, then take a look at functional programming languages today."

"Languages like Haskell and OCaml have served as a laboratory for new ideas to be developed, some of which have then made the transition into the mainstream."

– Simon Peyton Jones
(Principal Researcher at Microsoft)

Check out the article: "What's the future of programming? The answer lies in functional languages" at `www.techrepublic.com`

The four main goals of COMP 302

The four main goals of COMP 302

1. Provide a thorough introduction to fundamental concepts in programming languages
   Higher-order functions, State-full vs state-free computation, Modelling objects and closures, Exceptions to defer control, Continuations to defer control, Polymorphism, Partial evaluation, Lazy programming, Modules, ...
2. Show different ways to reason about programs
   Type checking, Induction, Operational semantics, ...
3. Introduce fundamental principles in programming language design
   Grammars and parsing, Operational semantics and interpreters, Type checking, polymorphism, and subtyping
4. Expose students to a different way of thinking about problems
   It's like going to the gym; it's good for you!

Today: Modules

Primary benefits:

- Control complexity of developing and maintaining software
- Split large programs into separate piece
- Name space separation
- Allows for separate compilation
- Incremental development
- Clear specifications at module boundaries
- Programs are easier to maintain and reuse (!)
- Enforces abstractions
- Isolates bugs
- ...