Quote of the Day

"First solve the problem. Then, write the code." - John Johnson

"A language that doesn't affect the way you think about programming, is not worth knowing." - Alan Perlis

COMP 302 is . . .

. . . exposing you to a different way of thinking about problems.

COMP 302 is . . .

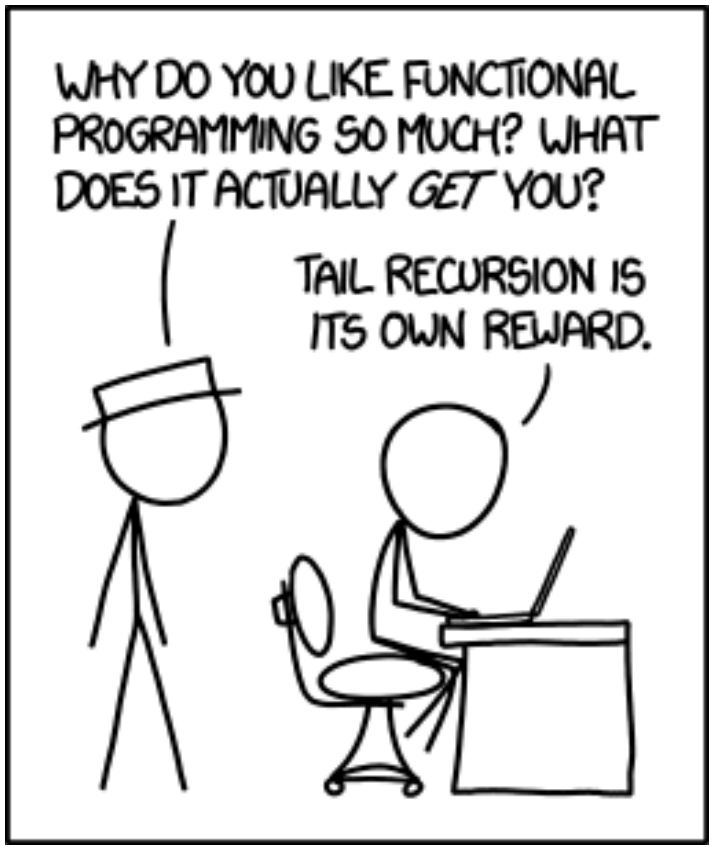
. . . exposing you to a different way of thinking about problems.

- This course: functional programming in OCaml
- Recursion, recursion, recursion
- Types and Abstractions

COMP 302 is . . .

. . . exposing you to a different way of thinking about problems.

- This course: functional programming in OCaml
- Recursion, recursion, recursion
- Types and Abstractions



COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions
- Staged programming and partial evaluation

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions
- Staged programming and partial evaluation
- State-full vs state-free computation

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions
- Staged programming and partial evaluation
- State-full vs state-free computation
- Modelling objects and closure

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions
- Staged programming and partial evaluation
- State-full vs state-free computation
- Modelling objects and closure
- Exceptions

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions
- Staged programming and partial evaluation
- State-full vs state-free computation
- Modelling objects and closure
- Exceptions
- Continuations to defer control

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions
- Staged programming and partial evaluation
- State-full vs state-free computation
- Modelling objects and closure
- Exceptions
- Continuations to defer control
- Lazy programming

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions
- Staged programming and partial evaluation
- State-full vs state-free computation
- Modelling objects and closure
- Exceptions
- Continuations to defer control
- Lazy programming
- Modules

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions
- Staged programming and partial evaluation
- State-full vs state-free computation
- Modelling objects and closure
- Exceptions
- Continuations to defer control
- Lazy programming
- Modules
- . . .

COMP 302 is . . .

. . . introducing you to fundamental concepts found in many languages

- Types
- Polymorphism
- Higher-order functions
- Staged programming and partial evaluation
- State-full vs state-free computation
- Modelling objects and closure
- Exceptions
- Continuations to defer control
- Lazy programming
- Modules
- . . .

Understanding these concepts will make it easier, quicker, and more effective for you to pick up new languages. - E. Dijkstra

Let's practice ...

COMP 302 is . . .

. . . learning to reason about programs

COMP 302 is . . .

. . . learning to reason about programs

- Induction

COMP 302 is . . .

. . . learning to reason about programs

- Induction
- Types systems provide a simple, static, lightweight tool to reason about programs approximating run-time behavior.

COMP 302 is . . .

. . . learning to reason about programs

- Induction
- Types systems provide a simple, static, lightweight tool to reason about programs approximating run-time behavior.

  $\Longrightarrow$ Be able to infer the most general type
- Subtyping, polymorphism
- Evaluation (Environment diagrams, Operational semantics)

Typed functional programming enforces disciplined programming, ultimately making you a better all-round programmer.

Let's practice ...

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

- How do we formally describe the grammar of a language?

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

- How do we formally describe the grammar of a language?
- When is a variable free? When is an expression closed?

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

- How do we formally describe the grammar of a language?
- When is a variable free? When is an expression closed?
- How do we formally describe its execution?

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

- How do we formally describe the grammar of a language?
- When is a variable free? When is an expression closed?
- How do we formally describe its execution? What does it mean for a language to be type-safe?

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

- How do we formally describe the grammar of a language?
- When is a variable free? When is an expression closed?
- How do we formally describe its execution? What does it mean for a language to be type-safe?
- What are good languages?

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

- How do we formally describe the grammar of a language?
- When is a variable free? When is an expression closed?
- How do we formally describe its execution? What does it mean for a language to be type-safe?
- What are good languages?

Why is this important?

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

- How do we formally describe the grammar of a language?
- When is a variable free? When is an expression closed?
- How do we formally describe its execution? What does it mean for a language to be type-safe?
- What are good languages?

Why is this important?

- Gain a deeper understanding of the behavior of a given program

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

- How do we formally describe the grammar of a language?
- When is a variable free? When is an expression closed?
- How do we formally describe its execution? What does it mean for a language to be type-safe?
- What are good languages?

Why is this important?

- Gain a deeper understanding of the behavior of a given program
- Be able to design your own little language

COMP 302 . . .

introduces you to fundamental principles in programming language design and how to realize these ideas in code

- How do we formally describe the grammar of a language?
- When is a variable free? When is an expression closed?
- How do we formally describe its execution? What does it mean for a language to be type-safe?
- What are good languages?

Why is this important?

- Gain a deeper understanding of the behavior of a given program
- Be able to design your own little language
- Be able to implement compilers

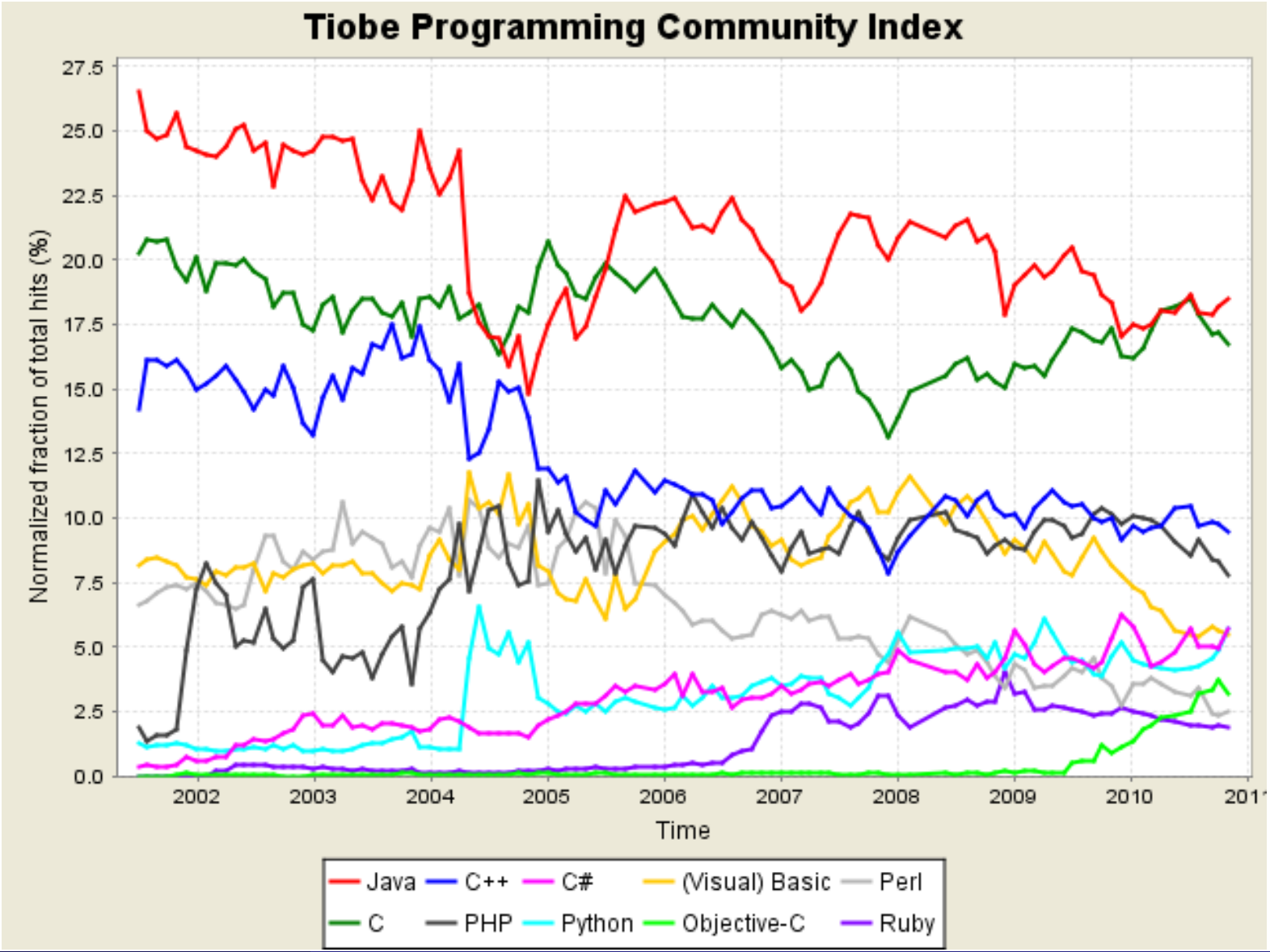
Let's practice ...

A glimpse to the past (www.tiobe.com)

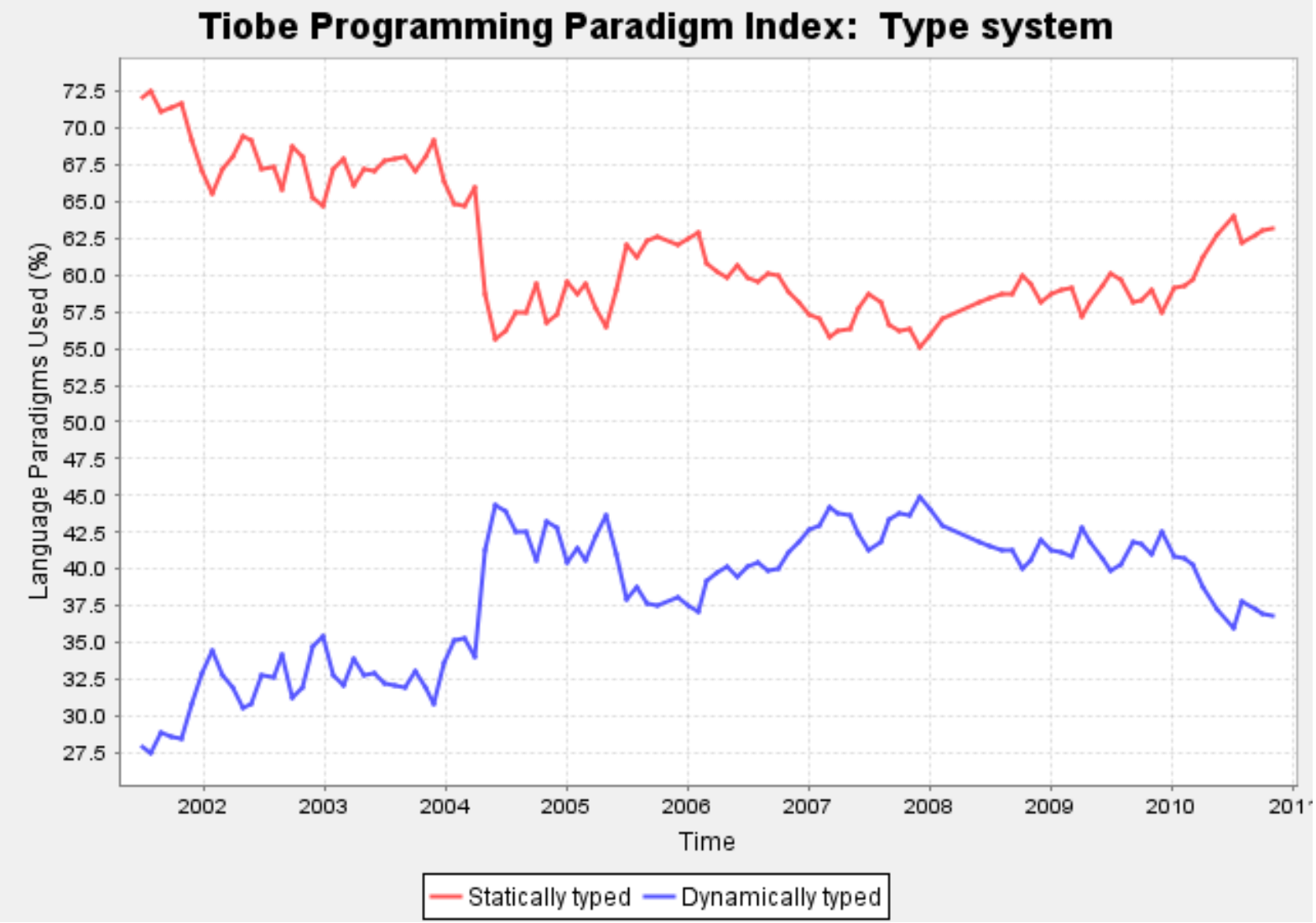
A glimpse to the past (www.tiobe.com)

Programming Language	2015	2010	2005	2000	1995	1990	1985
Java	1	1	2	3	31	-	-
C	2	2	1	1	2	1	1
C++	3	3	3	2	1	2	9
C#	4	5	6	10	-	-	-
Objective-C	5	8	42	-	-	-	-
Python	6	6	7	25	9	-	-
PHP	7	4	4	21	-	-	-
JavaScript	8	10	10	7	-	-	-
Visual Basic .NET	9	192	-	-	-	-	-
Perl	10	7	5	4	5	17	-
Pascal	17	14	17	18	3	7	6
Fortran	25	25	14	17	18	3	5
Lisp	28	15	13	8	10	6	2
Ada	29	22	16	19	4	9	3

A glimpse to the past and current



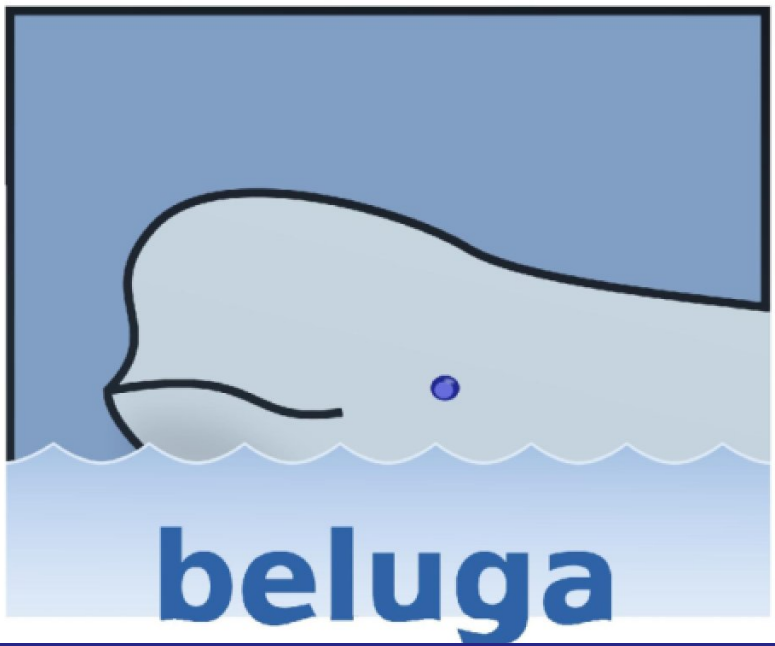
Static vs dynamically typed



A glimpse of the future

Beluga : Programming with proofs

- One of the most advanced languages to program with proofs
- Dependent types to express and track safety policies
- Designed on paper ; Implemented in OCaml



Want more?

- COMP 230: Logic and Computability
- COMP 523: Language-based security
  (really about Types and Programming Languages)
- COMP 527: Logic and computation
  (relationship between Types, Logic, and Programs)
- COMP 396, COMP 400 Projects

Last but not least

Go to Vincent's review lecture!

Last but not least

Go to Vincent's review lecture!

Thank you!

Last but not least

Go to Vincent's review lecture!

Thank you!

. . . And good luck with the final!