Taxonomy of Programming Languages

Sandeep Koranne

April 1, 2015

Motivation

Motivation

Critical view of language features

Classification of Programming Languages

Style

- Imperative (Procedural)

Classification of Programming Languages

Style

- Imperative (Procedural)
- Functional

Classification of Programming Languages

Style

- Imperative (Procedural)
- Functional
- Logic

Classification of Programming Languages

Style

- Imperative (Procedural)
- Functional
- Logic
- Object Oriented

Classification of Programming Languages

Style

- Imperative (Procedural)
- Functional
- Logic
- Object Oriented
- Declarative

Evolution of Programming Languages

- The original (circa 1955) concern was efficiency

$$y = 2 * (y + 5y^2) + \sin(y)\cos(y) \tag{1}$$

$$dx = \frac{x - y}{step} \tag{2}$$

$$dy = \sin(y)dx \tag{3}$$

- It was widely believed that no computer program would ever be able to optimize these sort of programs
- Today, writing such code even in Fortran and C++ is considered low level
- Only in the most critical cases is assembly language programming used.

Evolution of Programming Languages

- The original (circa 1955) concern was efficiency
- consider

$$
\begin{aligned}
y &= 2 * (y + 5y^2) + \sin(y)\cos(y) && (1)\\
dx &= \frac{x - y}{step} && (2)\\
dy &= \sin(y)dx && (3)
\end{aligned}
$$

- It was widely believed that no computer program would ever be able to optimize these sort of programs
- Today, writing such code even in Fortran and C++ is considered low level
- Only in the most critical cases is assembly language programming used.

Language features and their contrast

- There is no such thing as a *perfect* language

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic
- Such as the use of abstract data types in C++ for templates

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic
- Such as the use of abstract data types in C++ for templates
- Use of asynchronous messaging in Erlang

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic
- Such as the use of abstract data types in C++ for templates
- Use of asynchronous messaging in Erlang
- The use of map-reduce (from Common Lisp)

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic
- Such as the use of abstract data types in C++ for templates
- Use of asynchronous messaging in Erlang
- The use of map-reduce (from Common Lisp)
- Array and slice operations from Fortran

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic
- Such as the use of abstract data types in C++ for templates
- Use of asynchronous messaging in Erlang
- The use of map-reduce (from Common Lisp)
- Array and slice operations from Fortran
- List comprehensions from Common Lisp

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic
- Such as the use of abstract data types in C++ for templates
- Use of asynchronous messaging in Erlang
- The use of map-reduce (from Common Lisp)
- Array and slice operations from Fortran
- List comprehensions from Common Lisp
- Garbage collection (from Lisp and then Java)

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic
- Such as the use of abstract data types in C++ for templates
- Use of asynchronous messaging in Erlang
- The use of map-reduce (from Common Lisp)
- Array and slice operations from Fortran
- List comprehensions from Common Lisp
- Garbage collection (from Lisp and then Java)
- Use of packages (Python)

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic
- Such as the use of abstract data types in C++ for templates
- Use of asynchronous messaging in Erlang
- The use of map-reduce (from Common Lisp)
- Array and slice operations from Fortran
- List comprehensions from Common Lisp
- Garbage collection (from Lisp and then Java)
- Use of packages (Python)
- Strength of awk, sed, grep, bash, Tcl,

Language features and their contrast

- There is no such thing as a *perfect* language
- However, certain language features have become idiomatic
- Such as the use of abstract data types in C++ for templates
- Use of asynchronous messaging in Erlang
- The use of map-reduce (from Common Lisp)
- Array and slice operations from Fortran
- List comprehensions from Common Lisp
- Garbage collection (from Lisp and then Java)
- Use of packages (Python)
- Strength of awk, sed, grep, bash, Tcl,
- Use of SQL for declarative database programming

The goal of this course is to give an introduction to the above

Outline of this course

Lecture	Content	Lab	HW
4/2	Chap 1-2	C language	HW1
4/9	Chap 3-4	C language	HW2
4/16	Chap 5-6	Fortran	HW3
4/23	Chap 7-8	Common Lisp	HW4
4/30	Chap 9-10	C++	HW5
5/7	Chap 11	C++	HW6
5/14	Chap 12	C++	HW7
5/16	Chap 13	Erlang	HW8
5/21	No class	-	-
5/28	Chap 14	Python	HW9
6/4	Chap 15	Various	HW10
6/11	Chap 16	Various	Final

Each class may have a in-class quiz