

A programming language is low level when its programs require attention to the irrelevant.

- Alan Perlis

CSE341

Programming Languages

Lecture 1.1 – September 19, 2016

Introduction

© 2012-2016 Yakup Genç

Largely adapted from V. Shmatikov, J. Mitchell and R.W. Sebesta

Today

- Introductions
- Administrative Details
- Introduction to Programming Languages

Introductions

- Dr. Genc – Instructor:
 - PhD from CS @ UIUC in 1999
 - 15+ years research and R&D management & 4+ years academic experience
 - yakup.genc@gyte.edu.tr
 - Tel: x2220
- TAs:
 - TBA

Admin: Communication

- The course communication will be done via Moodle
 - <http://193.140.134.13/moodle>
- Make sure you register as soon as possible

Admin: Prerequisites

- You should know/have
 - Data structures
 - Good working knowledge of C/C++ & Java

Admin: Attendance etc.

- Policy:
 - Attendance is mandatory
 - To be kept for each lecture
 - Failure to meet requirement will result in a fail
 - No late admittance (class starts exactly at 9:15am)

Admin: Grading

- Grading (tentative)
 - %40 – Homework and projects
 - %25 – Midterm
 - %35 – Final
- Rules
 - No attendance → no pass
 - Zero tolerance on cheating

Homework and Projects

- Homework
 - Details to come soon
- You will need to learn “Lisp”
- Rules
 - Hand in before the class on the due date
 - No late submission will be accepted
 - Discuss among yourselves but do your own work, no cheating allowed

Reading Material

- Book:
 - Concepts of Programming Languages by R. W. Sebesta, Fifth Edition
 - Chapters will be assigned for reading
- Other reading materials will be provided as needed

Programming Languages

What's a programming language?

A programming language is an artificial language designed to communicate instructions to a machine, particularly a computer [Wikipedia].

A language is a “conceptual universe” providing a framework for problem-solving and useful concepts and programming methods [Perlis].

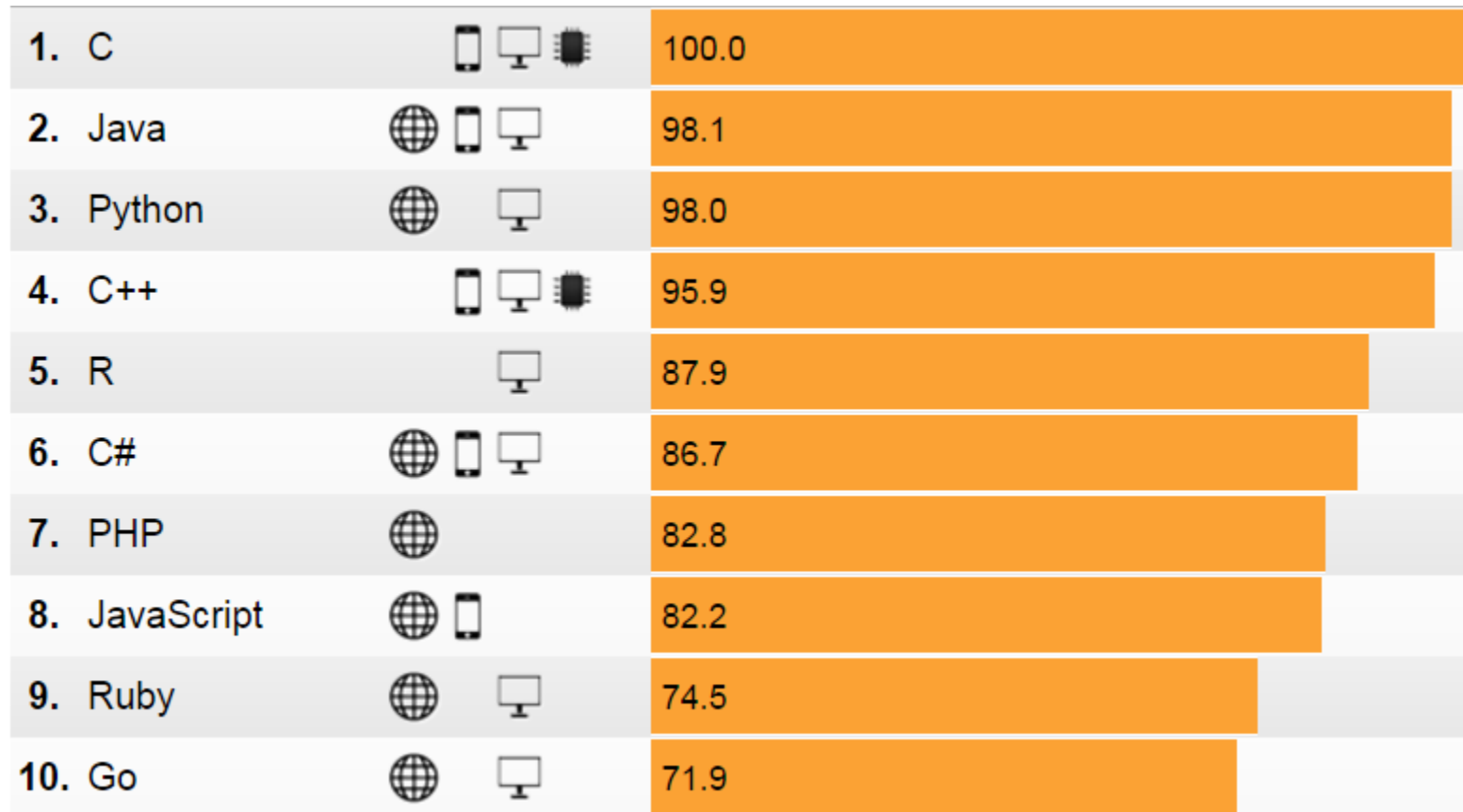
How many programming languages are there?

Thousands!

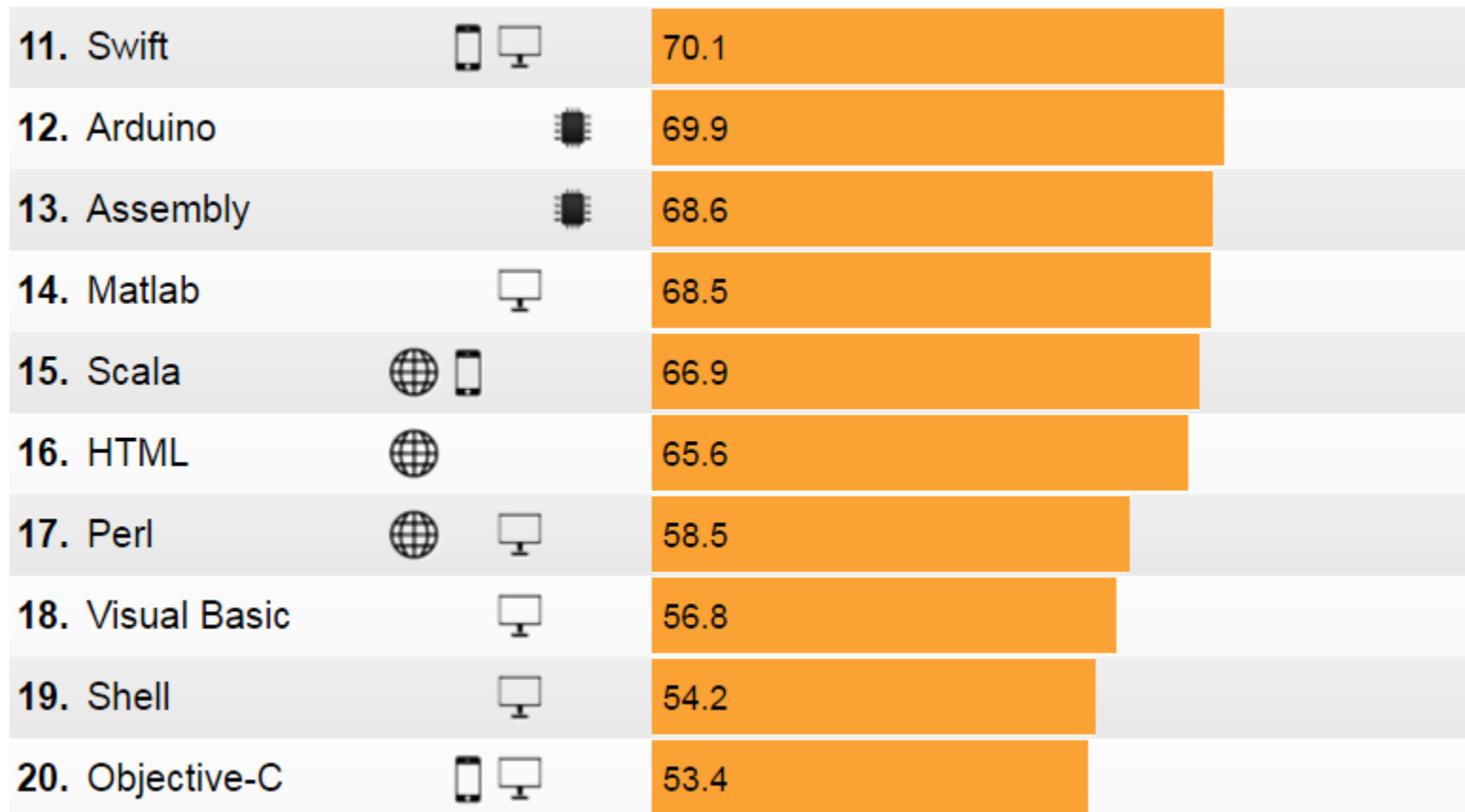
Which one to use?

We'll have a good idea after completing this course.


















Top Programming Languages – IEEE Spectrum

















Top Programming Languages – IEEE Spectrum



Top Programming Languages – IEEE Spectrum

| | | | |
|----------------|--|---|------|
| 21. Cuda | |  | 53.2 |
| 22. Lua |  |  | 52.1 |
| 23. Processing |  |  | 50.5 |
| 24. SQL | |  | 49.8 |
| 25. Haskell | |   | 44.1 |
| 26. Rust |  |  | 43.3 |
| 27. Fortran | |  | 42.2 |
| 28. Delphi | |   | 42.1 |
| 29. D |  |  | 38.9 |
| 30. LabView | |   | 35.7 |

Top Programming Languages – IEEE Spectrum

| | | |
|------------------|---|------|
| 31. VHDL |  | 35.4 |
| 32. Lisp |  | 34.9 |
| 33. Julia |  | 32.8 |
| 34. Ladder Logic |  | 28.1 |
| 35. Erlang |   | 28.0 |
| 36. Verilog |  | 26.7 |
| 37. Prolog |  | 26.1 |
| 38. Clojure |   | 24.1 |
| 39. SAS |  | 23.4 |
| 40. Ada |   | 22.0 |
| 41. Cobol |  | 19.2 |

Top Programming Languages – TIOBE

TIOBE Index for September 2016

September Headline: Julia enters top 50 for the first time

It was a matter of time until Julia would hit the top 50. This month it did so. The Julia programming language is meant for numerical computing. It combines functional programming paradigms with high speed. In other words, readable and stable code that performs. Chances are high that Julia will gain even more popularity the next few months. This new entry might make you curious what other languages are expected to reach the top 50 soon. I would put my bets on Hack (number 77), Kotlin (entering the top 100 this month at position 99) and TypeScript (still 183 but watch my words).

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the best programming language or the language in which most lines of code have been written.

<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

Top Programming Languages – TIOBE

| Sep 2016 | Sep 2015 | Change | Programming Language | Ratings | Change |
|----------|----------|--------|----------------------|---------|--------|
| 1 | 1 | | Java | 18.236% | -1.33% |
| 2 | 2 | | C | 10.955% | -4.67% |
| 3 | 3 | | C++ | 6.657% | -0.13% |
| 4 | 4 | | C# | 5.493% | +0.58% |
| 5 | 5 | | Python | 4.302% | +0.64% |
| 6 | 7 | ▲ | JavaScript | 2.929% | +0.59% |
| 7 | 6 | ▼ | PHP | 2.847% | +0.32% |
| 8 | 11 | ▲ | Assembly language | 2.417% | +0.61% |
| 9 | 8 | ▼ | Visual Basic .NET | 2.343% | +0.28% |
| 10 | 9 | ▼ | Perl | 2.333% | +0.43% |
| 11 | 13 | ▲ | Delphi/Object Pascal | 2.169% | +0.42% |
| 12 | 12 | | Ruby | 1.965% | +0.18% |
| 13 | 16 | ▲ | Swift | 1.930% | +0.74% |
| 14 | 10 | ✚ | Objective-C | 1.849% | +0.03% |
| 15 | 17 | ▲ | MATLAB | 1.826% | +0.65% |

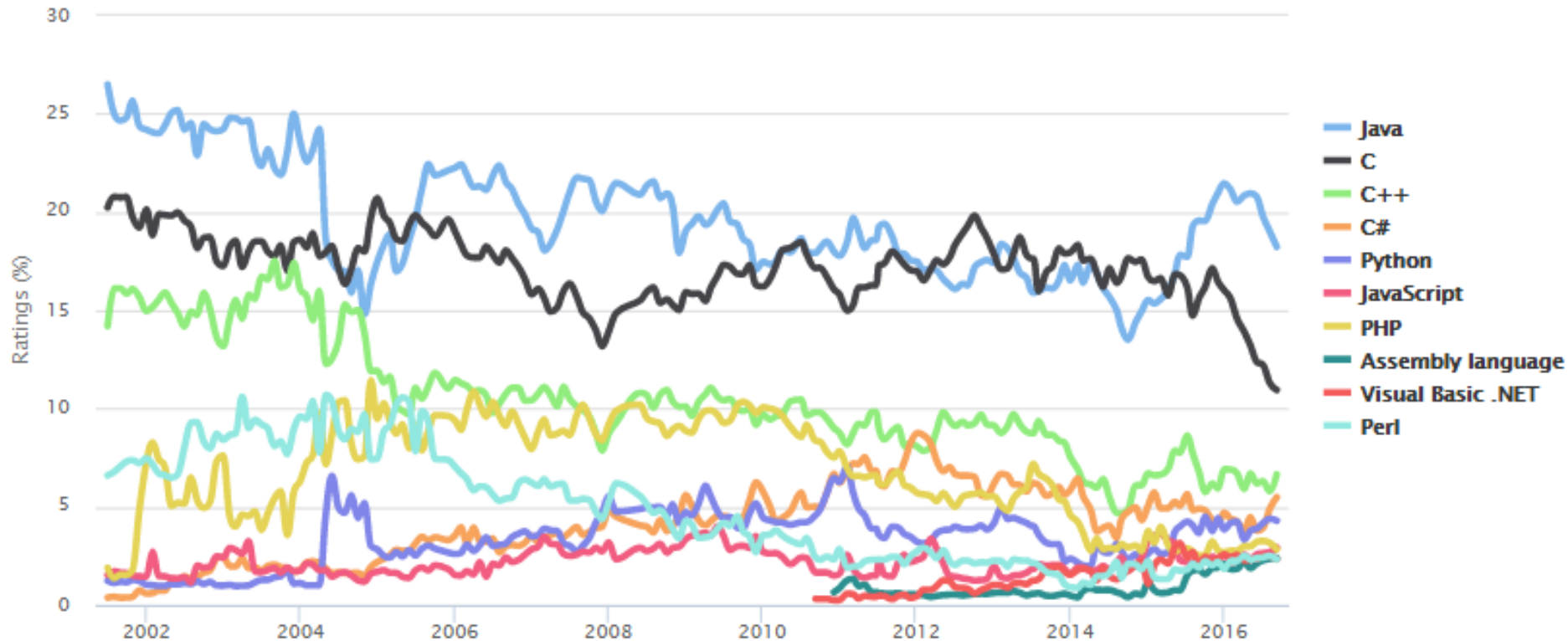
TIOBE Programming Community Index for September 2016

Top Programming Languages – TIOBE

| Sep 2016 | Sep 2015 | Change | Programming Language | Ratings | Change |
|----------|----------|--------|----------------------|---------|--------|
| 1 | 1 | | Java | 18.236% | -1.33% |
| 2 | 2 | | C | 10.955% | -4.67% |
| 3 | 3 | | C++ | 6.657% | -0.13% |
| 4 | 4 | | C# | 5.493% | +0.58% |
| 5 | 5 | | Python | 4.302% | +0.64% |
| 6 | 7 | ▲ | JavaScript | 2.929% | +0.59% |
| 7 | 6 | ▼ | PHP | 2.847% | +0.32% |
| 8 | 11 | ▲ | Assembly language | 2.417% | +0.61% |
| 9 | 8 | ▼ | Visual Basic .NET | 2.343% | +0.28% |
| 10 | 9 | ▼ | Perl | 2.333% | +0.43% |
| 11 | 13 | ▲ | Delphi/Object Pascal | 2.169% | +0.42% |
| 12 | 12 | | Ruby | 1.965% | +0.18% |
| 13 | 16 | ▲ | Swift | 1.930% | +0.74% |
| 14 | 10 | ✚ | Objective-C | 1.849% | +0.03% |
| 15 | 17 | ▲ | MATLAB | 1.826% | +0.65% |
| 16 | 34 | ▲ | Groovy | 1.818% | +1.31% |
| 17 | 14 | ▼ | Visual Basic | 1.761% | +0.23% |
| 18 | 19 | ▲ | R | 1.684% | +0.64% |
| 19 | 44 | ▲ | Go | 1.625% | +1.37% |
| 20 | 18 | ▼ | PL/SQL | 1.443% | +0.36% |

TIOBE Programming Community Index for September 2016

Top Programming Languages – TIOBE



TIOBE Programming Community Index for September 2015

Top Programming Languages – TIOBE

| Programming Language | 2016 | 2011 | 2006 | 2001 | 1996 | 1991 | 1986 |
|----------------------|------|------|------|------|------|------|------|
| Java | 1 | 1 | 1 | 3 | 14 | - | - |
| C | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| C++ | 3 | 3 | 3 | 2 | 2 | 2 | 5 |
| C# | 4 | 5 | 6 | 11 | - | - | - |
| Python | 5 | 6 | 7 | 24 | 23 | - | - |
| PHP | 6 | 4 | 4 | 8 | - | - | - |
| JavaScript | 7 | 9 | 8 | 7 | 19 | - | - |
| Visual Basic .NET | 8 | 30 | - | - | - | - | - |
| Perl | 9 | 8 | 5 | 4 | 3 | - | - |
| Ruby | 10 | 10 | 18 | 32 | - | - | - |
| Lisp | 27 | 12 | 12 | 15 | 7 | 5 | 3 |
| Ada | 28 | 16 | 15 | 16 | 6 | 3 | 2 |

TIOBE Programming Community Index for September 2016

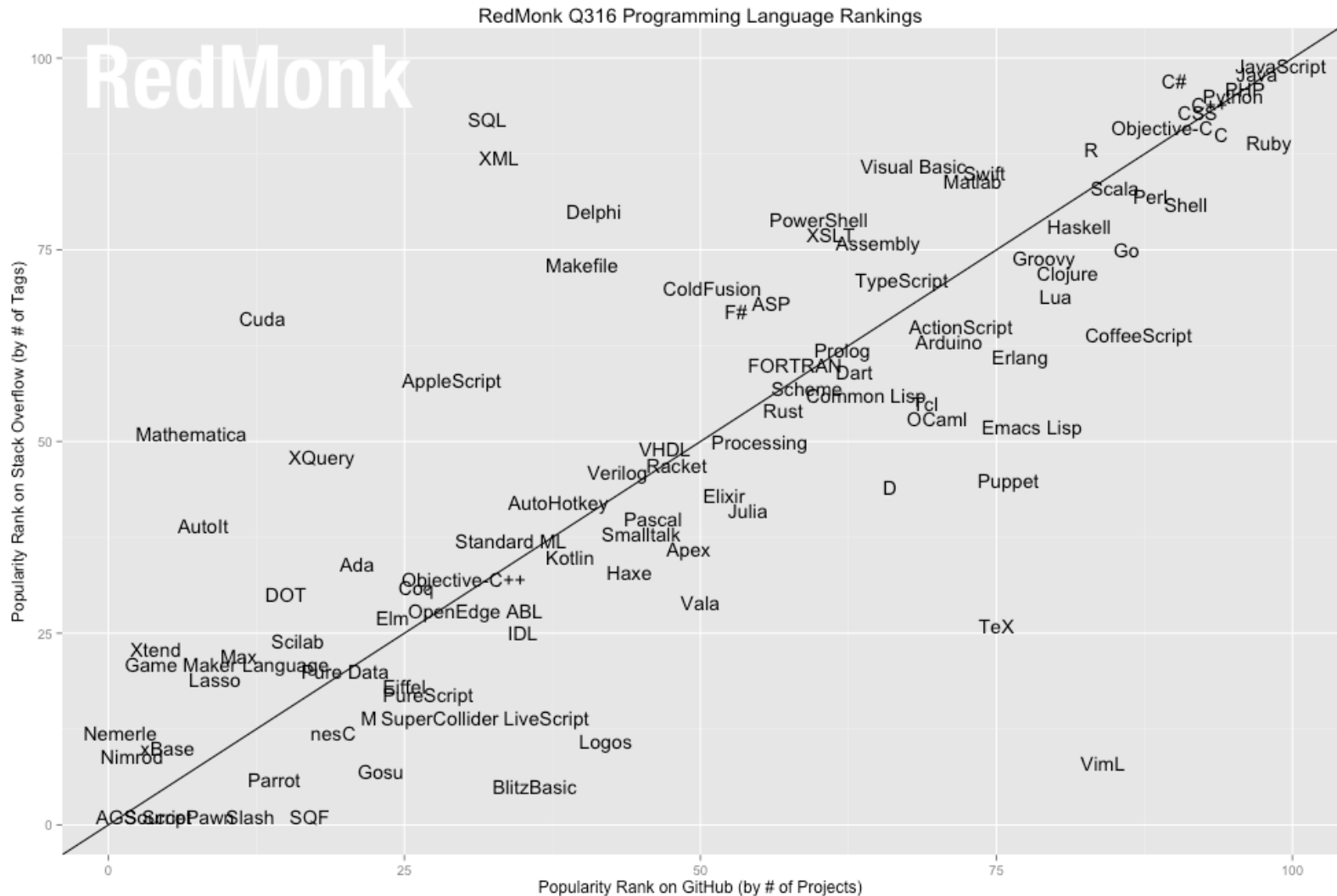
Top Programming Languages – TIOBE

| Year | Winner |
|------|--------------|
| 2015 | Java |
| 2014 | JavaScript |
| 2013 | Transact-SQL |
| 2012 | Objective-C |
| 2011 | Objective-C |
| 2010 | Python |
| 2009 | Go |
| 2008 | C |
| 2007 | Python |
| 2006 | Ruby |
| 2005 | Java |
| 2004 | PHP |
| 2003 | C++ |

TIOBE Programming Community Index for September 2016

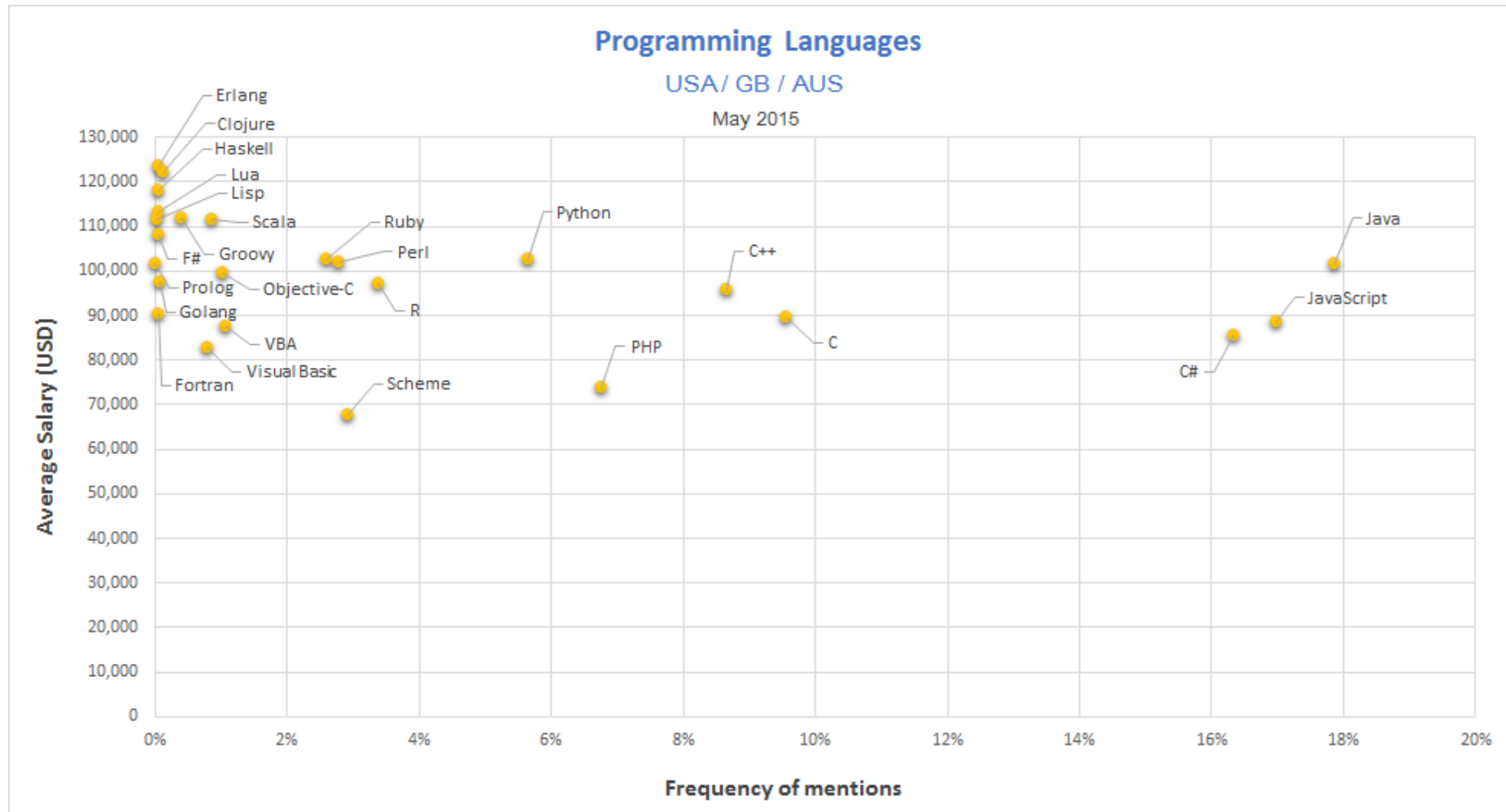
Programming Languages – Popularity

http://sograpy-media.redmonk.com/sograpy/files/2016/07/lang.rank_Q316.plot-WM.png



Programming Languages – Jobs

<https://gooroo.io/GoorooTHINK/Article/16300/Programming-languages--salaries-and-demand-May-2015/18672#.V96phvI97IU>



Why this Course?

Programming Language Concepts

- A language is a “conceptual universe” (Perlis)
 - Framework for problem-solving
 - Useful concepts and programming methods
- Understand the languages you use, by comparison
- Appreciate history, diversity of ideas in programming
- Be prepared for new programming methods, paradigms, tools

J. Mitchell

Why this Course?

Critical thought

- Identify properties of language, not syntax or sales pitch
- Language and implementation

Every convenience has its cost

- Recognize the cost of presenting an abstract view of machine
- Understand trade-offs in programming language design

J. Mitchell

Trends

Commercial trend over past 5+ years

- Increasing use of type-safe languages: Java, C#, ...
- Scripting languages, other languages for web applications

Teaching trends

- Java replaced C as most common intro language
 - Less emphasis on how data, control represented in machine
- Objective C

J. Mitchell

Trends

Research and development trends

- Modularity
 - Java, C++: standardization of new module features
- Program analysis
 - Automated error detection, programming env, compilation
- Isolation and security
 - Sandboxing, language-based security, ...
- Web 2.0
 - Increasing client-side functionality, mashup isolation problems

J. Mitchell

Example



What is Lua?

Lua is a powerful, fast, lightweight, embeddable scripting language.

Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics. Lua is dynamically typed, runs by interpreting bytecode for a register-based virtual machine, and has automatic memory management with incremental garbage collection, making it ideal for configuration, scripting, and rapid prototyping.

<http://www.lua.org/about.html>

Example



Where does Lua come from?

Lua is designed, implemented, and maintained by a [team](#) at [PUC-Rio](#), the Pontifical Catholic University of Rio de Janeiro in Brazil. Lua was born and raised in [Tecgraf](#), formerly the Computer Graphics Technology Group of PUC-Rio. Lua is now housed at [LabLua](#), a laboratory of the [Department of Computer Science](#) of PUC-Rio.

What's in a name?

"Lua" (pronounced **LOO-ah**) means "Moon" in Portuguese. As such, it is neither an acronym nor an abbreviation, but a noun. More specifically, "Lua" is a name, the name of the Earth's moon and the name of the language. Like most names, it should be written in lower case with an initial capital, that is, "Lua". Please do not write it as "LUA", which is both ugly and confusing, because then it becomes an acronym with [different meanings](#) for different people. So, please, write "Lua" right!

<http://www.lua.org/about.html>

Example



Why choose Lua?

Lua is a proven, robust language

Lua has been used in [many industrial applications](#) (e.g., [Adobe's Photoshop Lightroom](#)), with an emphasis on embedded systems (e.g., the [Ginga](#) middleware for digital TV in Brazil) and [games](#) (e.g., [World of Warcraft](#) and Angry Birds). Lua is currently [the leading scripting language in games](#). Lua has a solid [reference manual](#) and there are [several books about it](#). Several [versions](#) of Lua have been released and used in [real applications](#) since its creation in 1993. Lua featured in [HOPL III, the Third ACM SIGPLAN History of Programming Languages Conference](#), in June 2007.

Lua is fast

Lua has a deserved reputation for performance. To claim to be "as fast as Lua" is an aspiration of other scripting languages. Several benchmarks show Lua as the fastest language in the realm of interpreted scripting languages. Lua is fast not only in fine-tuned benchmark programs, but in real life too. Substantial fractions of large applications have been written in Lua.

If you need even more speed, try [LuaJIT](#), an independent implementation of Lua using a just-in-time compiler.

<http://www.lua.org/about.html>

Example



Why choose Lua?

Lua is portable

Lua is [distributed](#) in a small package and builds out-of-the-box in all platforms that have a standard C compiler. Lua runs on all flavors of Unix and Windows, on mobile devices (running Android, iOS, BREW, Symbian, Windows Phone), on embedded microprocessors (such as ARM and Rabbit, for applications like Lego MindStorms), on IBM mainframes, etc.

For specific reasons why Lua is a good choice also for constrained devices, read [this summary](#) by Mike Pall. See also a [poster](#) created by Timm Müller.

Lua is embeddable

Lua is a fast language engine with small footprint that you can embed easily into your application. Lua has a simple and well documented API that allows strong integration with code written in other languages. It is easy to extend Lua with libraries written in other languages. It is also easy to extend programs written in other languages with Lua. Lua has been used to extend programs written not only in C and C++, but also in Java, C#, Smalltalk, Fortran, Ada, Erlang, and even in other scripting languages, such as Perl and Ruby.

<http://www.lua.org/about.html>

Example



Why choose Lua?

Lua is powerful (but simple)

A fundamental concept in the design of Lua is to provide *meta-mechanisms* for implementing features, instead of providing a host of features directly in the language. For example, although Lua is not a pure object-oriented language, it does provide meta-mechanisms for implementing classes and inheritance. Lua's meta-mechanisms bring an economy of concepts and keep the language small, while allowing the semantics to be extended in unconventional ways.

Lua is small

Adding Lua to an application does not bloat it. The [tarball for Lua 5.2.3](#), which contains source code and documentation, takes 246K compressed and 960K uncompressed. The source contains around 20000 lines of C. Under Linux, the Lua interpreter built with all standard Lua libraries takes 182K and the Lua library takes 244K.

Lua is free

Lua is free open-source software, distributed under a [very liberal license](#) (the well-known MIT license). It may be used for any purpose, including commercial purposes, at absolutely no cost. Just [download](#) it and use it.

<http://www.lua.org/about.html>

Tentative Schedule

Week 1: Introduction, Paradigms

Weeks 2&3: Lexical and Syntax Analysis

Week 4: Holiday Break

Week 5: Names, Bindings and Scope

Week 6: Data Types

Week 7: Expressions and Assignments



Week 8: Control

Week 9: Subprograms

Week 10: Encapsulation

Week 11: Object Oriented

Week 13: Concurrency & Exception Handling

Week 14: Functional & Logic Programming

Next

- Start with programming paradigms...
- Reminders:
 - Read chapters 1 and 2