Lecture 1

Comparison of programming languages

Iztok Savnik, FAMNIT

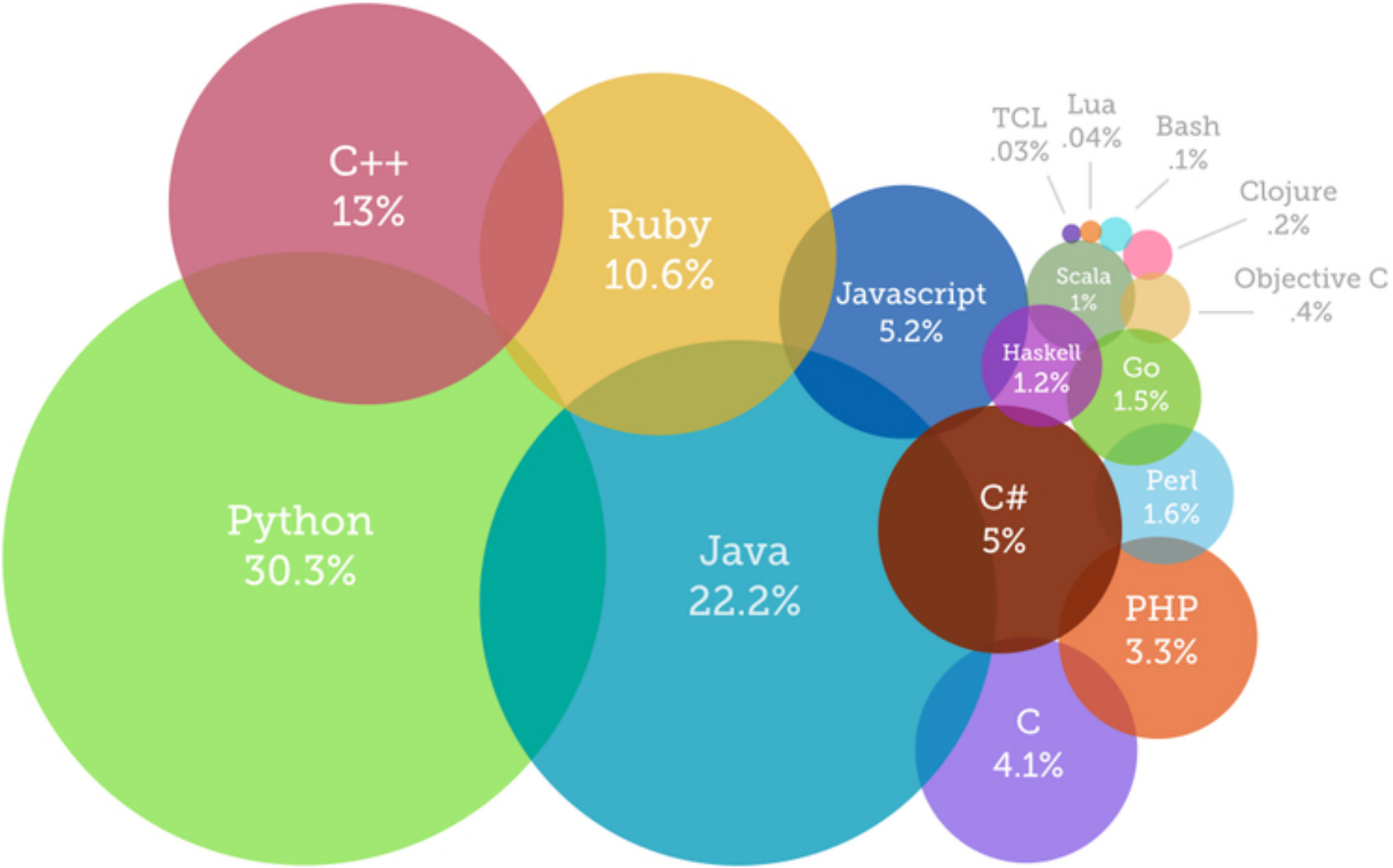
2015-19.

Presented material was gathered from 2015 to 2019.

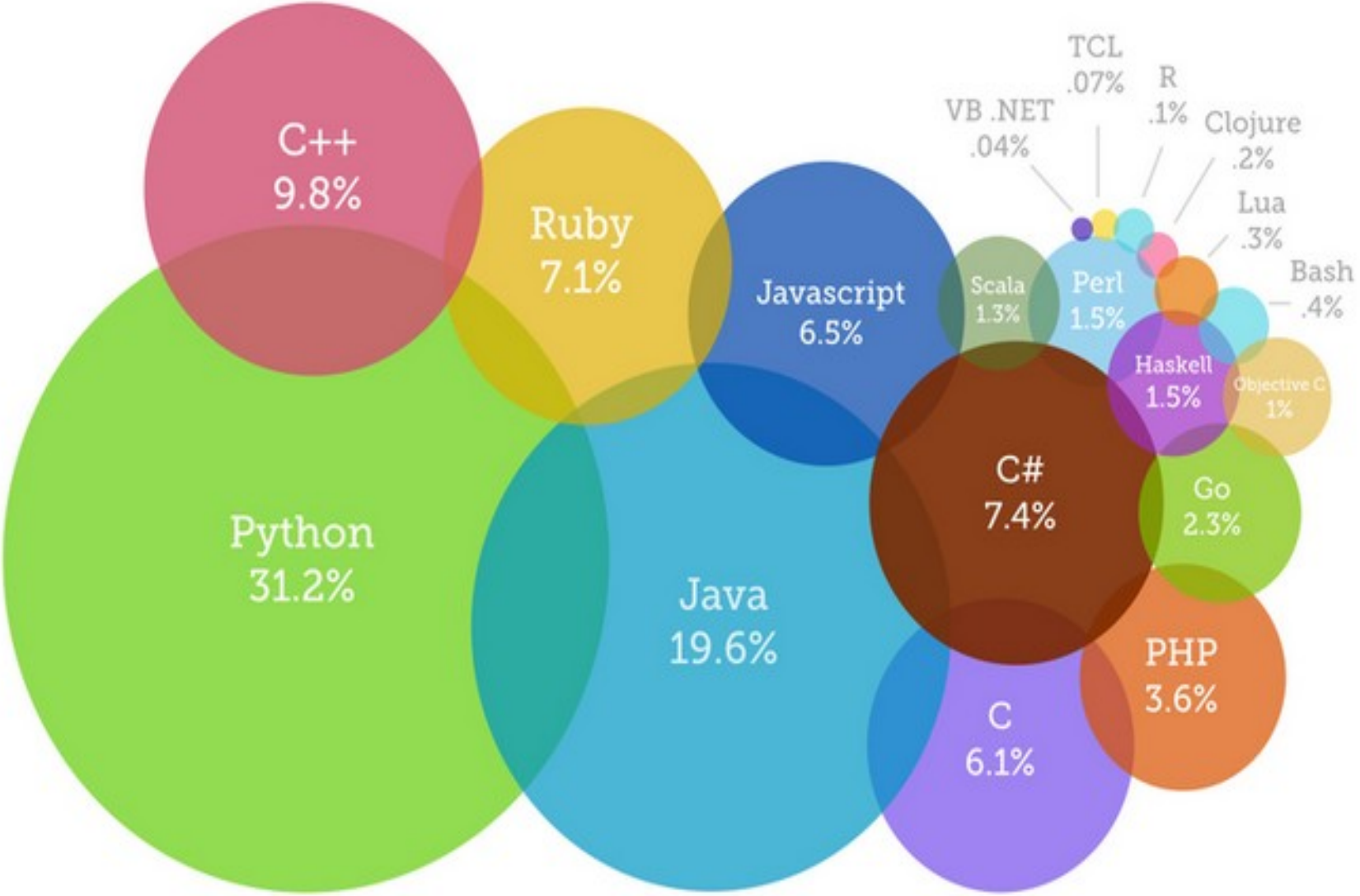
Comparisons of PL

- Codeval comparison
- A Comparison of Programming Languages in Economics
- In-Deman Programming languages
- The Computer Language Benchmarks Game
- Githut
- TIOBE Index

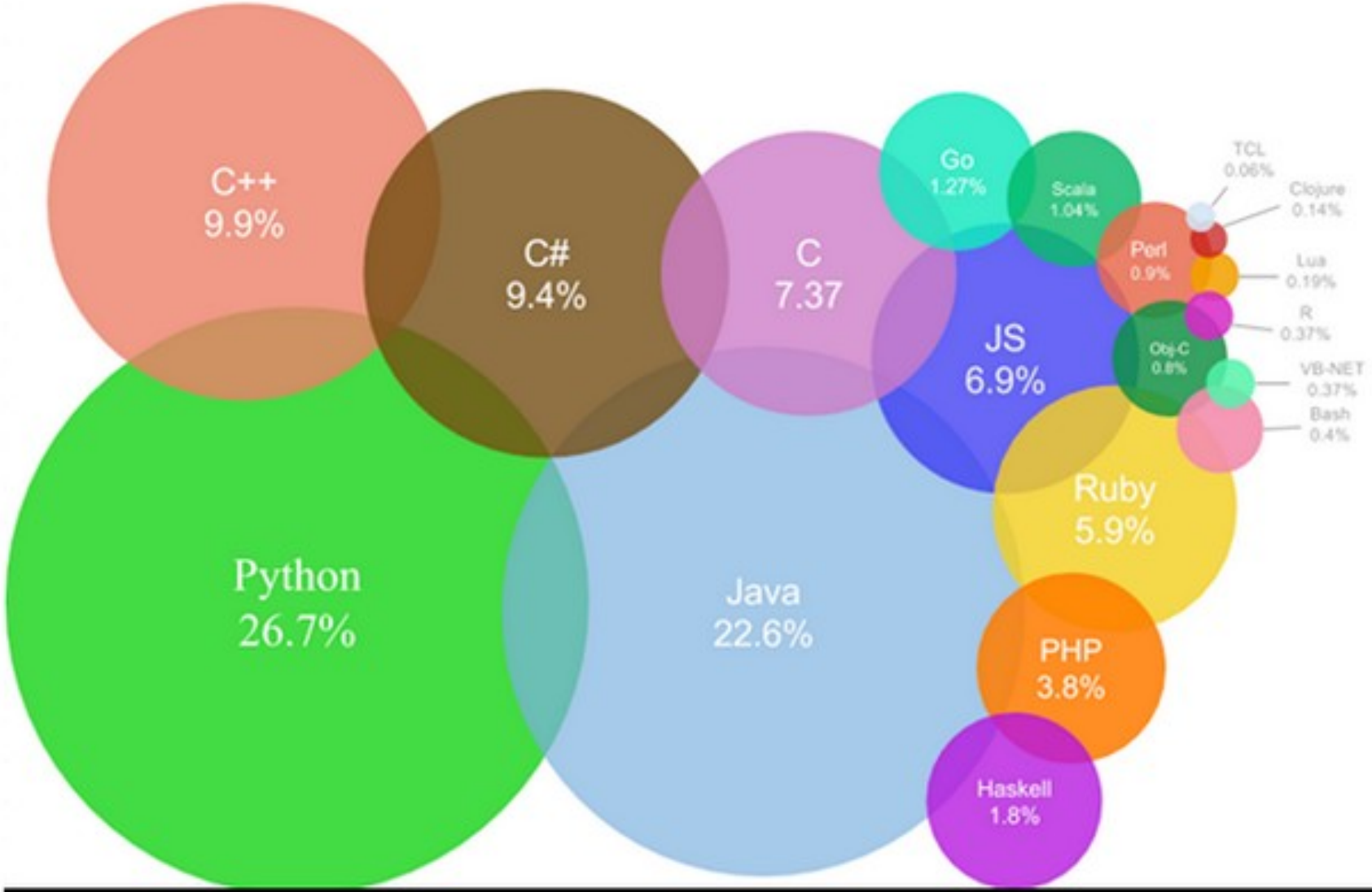
Most Popular Coding Languages of 2014



Most Popular Coding Languages of 2015

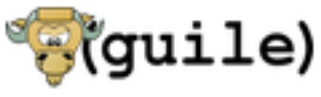


Most Popular Coding Languages of 2016



Programming language change percentage by year.

2015 Rank		2015	Change%	2014	Change%	2013	Change%
1	Python	26.67%	-14.64%	31.24%	3.10%	30.30%	5.21%
2	Java	22.58%	15.37%	19.57%	-11.85%	22.20%	-13.95%
3	C++	9.96%	1.76%	9.79%	-24.70%	13.00%	3.17%
4	C#	9.39%	27.37%	7.37%	47.37%	5.00%	100.00%
5	C	7.37%	21.37%	6.07%	48.14%	4.10%	-16.33%
6	JavaScript	6.88%	6.09%	6.48%	24.66%	5.20%	33.33%
7	Ruby	5.88%	-17.27%	7.11%	-32.90%	10.60%	10.42%
8	PHP	3.82%	5.45%	3.62%	9.84%	3.30%	-54.79%
9	Haskell	1.77%	17.24%	1.51%	25.83%	1.20%	
10	Go	1.27%	-44.00%	2.26%	50.67%	1.50%	-25.00%
11	Scala	1.04%	-17.80%	1.27%	27.00%	1.00%	66.67%
12	Perl	0.95%	-37.33%	1.52%	-6.17%	1.62%	
13	Objective-C	0.82%	-17.62%	1.00%	265.76%	0.27%	173.40%
14	Bash	0.46%	7.21%	0.43%	290.91%	0.11%	
15	R	0.37%	165.71%	0.14%	-30.00%	0.20%	
16	Visual Basic,NET	0.37%	825.50%	0.04%			
17	Lua	0.19%	-44.51%	0.35%	337.50%	0.08%	
18	Clojure	0.14%	-8.53%	0.15%	-48.28%	0.29%	-63.75%
19	Tcl	0.06%	-8.57%	0.07%	133.33%	0.03%	50.00%

We've Added 5 New Programming Languages To CodeEval: D, Fortran, Guile, OCaml and Scheme.

January 23, 2016

CodeEval now supports D, Guile, OCaml, Fortran and Scheme.

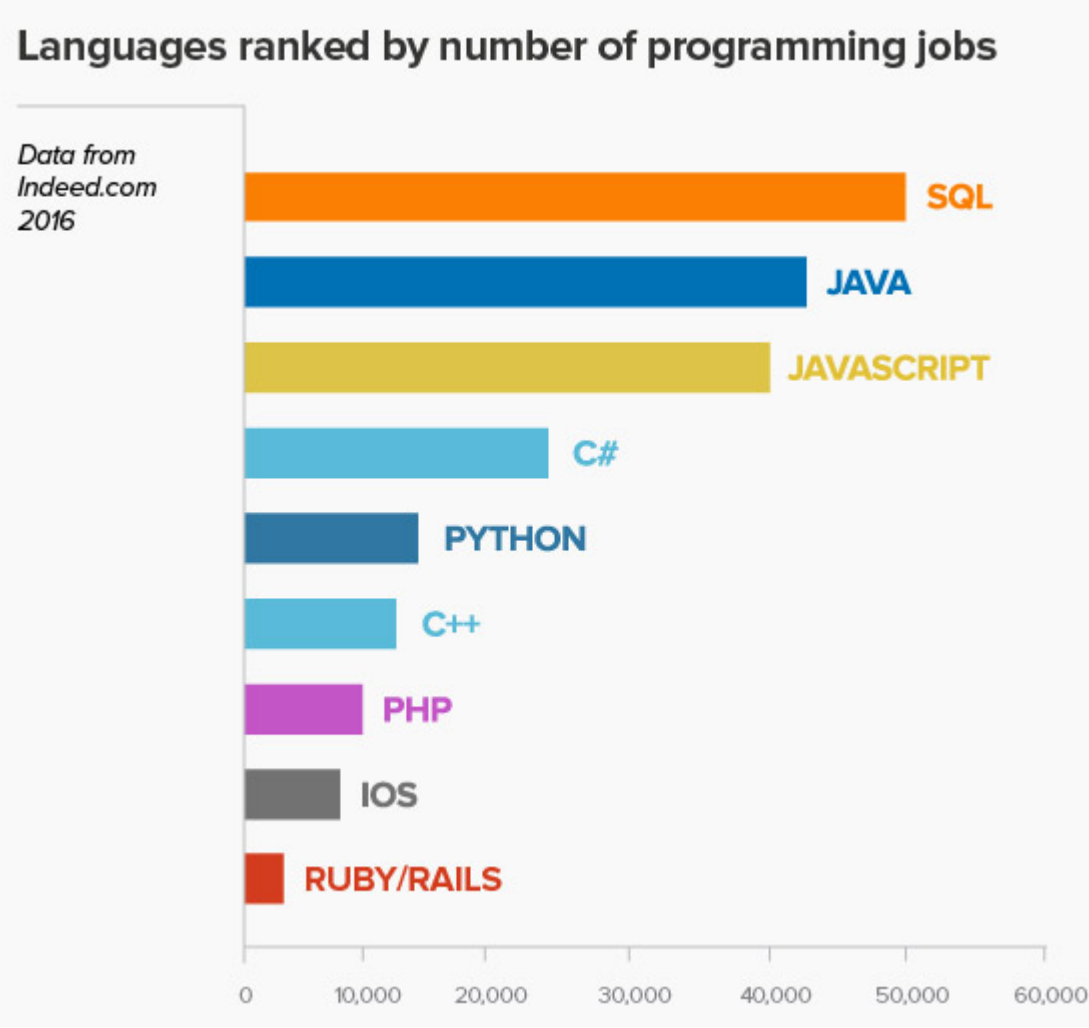
It's been a while since we added new languages to CodeEval — the last one was R — but we've taken the opportunity to add these 5 new languages.

A Comparison of Programming Languages in Economics, Aruoba and Fernandez-Villaverde, 2014

1. C++ and Fortran are still considerably faster than any other alternative, although one needs to be careful with the choice of compiler.
2. C++ compilers have advanced enough that, contrary to the situation in the 1990s and some folk wisdom, C++ code runs slightly faster (5-7 percent) than Fortran code.
3. Julia delivers outstanding performance. Execution speed is only between 2.64 and 2.70 times slower than the execution speed of the best C++ compiler.
4. Baseline Python was slow. Using the Pypy implementation, it runs around 44 times slower than in C++. Using the default CPython interpreter, the code runs between 155 and 269 times slower than in C++.
5. Matlab is between 9 to 11 times slower than the best C++ executable.
6. R runs between 475 to 491 times slower than C++. If the code is compiled, the code is between 243 to 282 times slower.
7. Hybrid programming and special approaches can deliver considerable speed ups. For example, when combined with Mex files, Matlab is only 1.24 to 1.64 times slower than C++ and when combined with Rcpp, R is between 3.66 and 5.41 times slower. Similar numbers hold for Numba (a just-in-time compiler for Python that uses decorators) and Cython (a static compiler for writing C extensions for Python) in the Python ecosystem.
8. Mathematica is only about three times slower than C++, but only after a considerable rewriting of the code to take advantage of the peculiarities of the language. The baseline version of the algorithm in Mathematica is considerably slower.

The 9 Most In-Demand Programming Languages of 2016

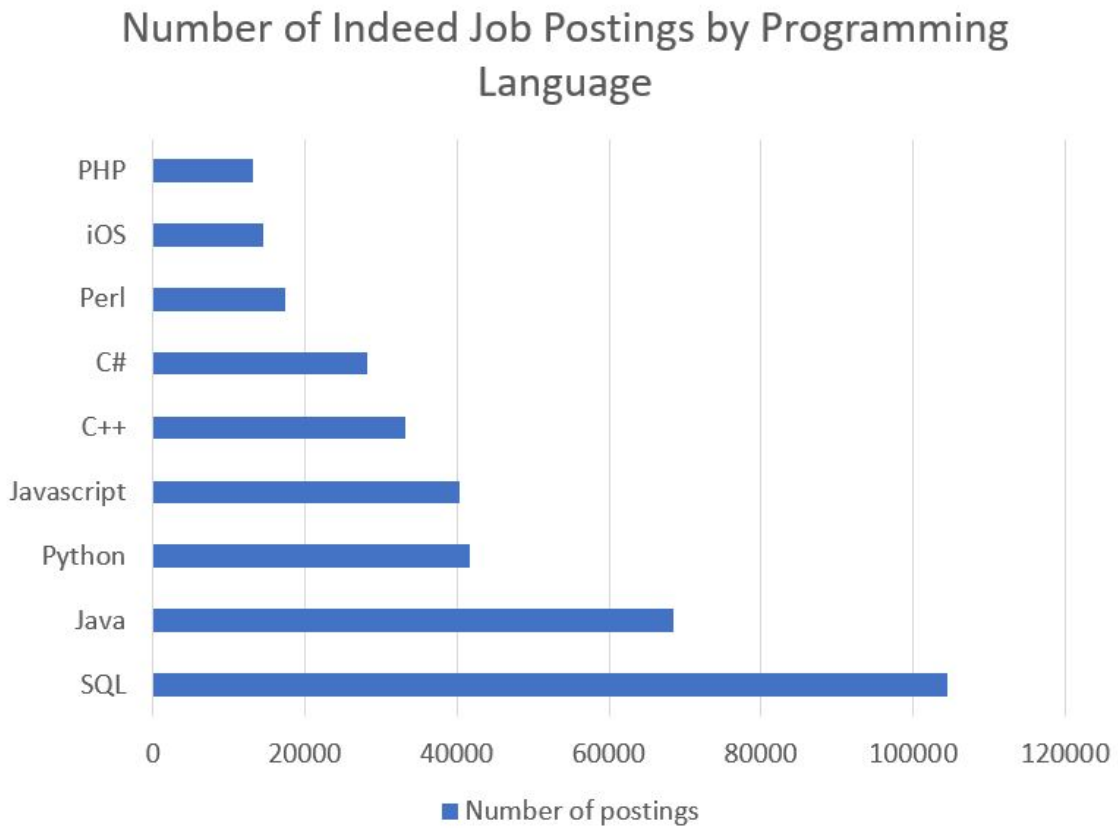
January 27, 2016



http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2016/

The 9 Most In-Demand Programming Languages of 2017

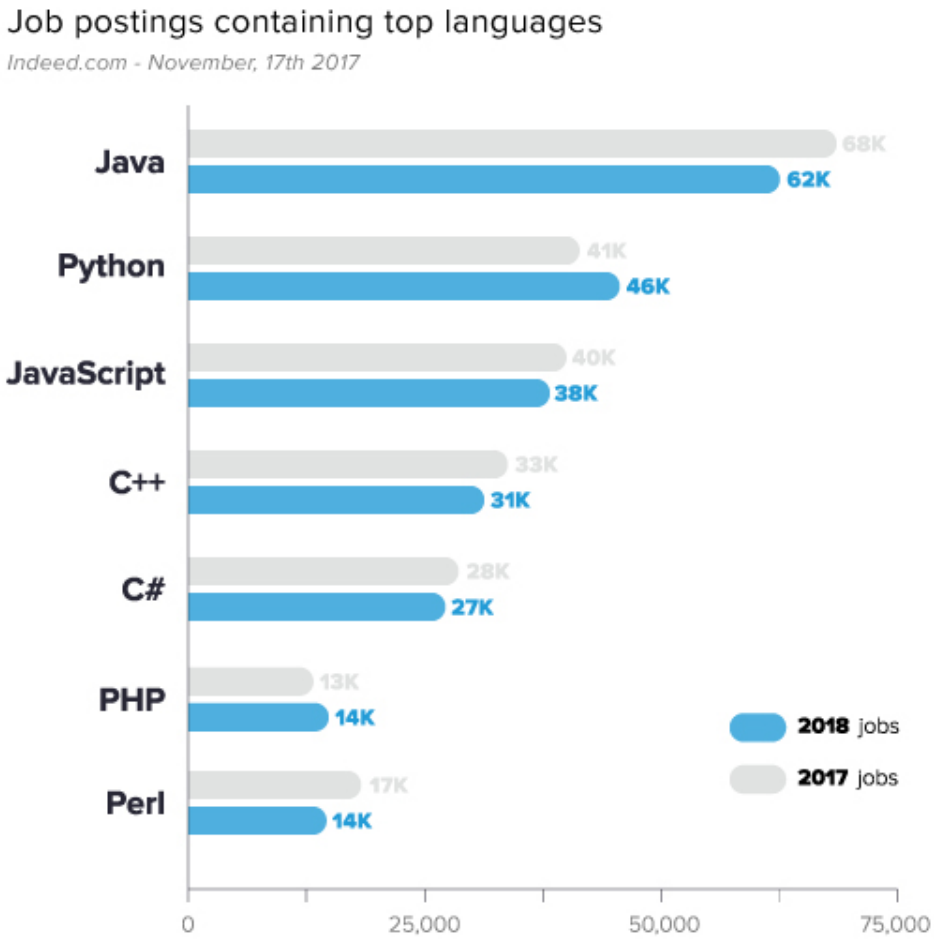
February 2, 2017



http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2017/

The 7 Most In-Demand Programming Languages of 2018

December 13, 2017



http://www.codingdojo.com/blog/7-most-in-demand-programming-languages-of-2018/

The 7 Most In-Demand Programming Languages of 2019

March 15, 2019

- Java – 65,986 jobs
- Python – 61,818 jobs
- Javascript – 38,018 jobs
- C++ – 36,798 jobs
- C# – 27,521 jobs
- PHP – 16,890 jobs
- PERL – 13, 727 jobs

https://www.codingdojo.com/blog/the-7-most-in-demand-programming-languages-of-2019

9 cutting-edge programming languages worth learning now

These strong alternatives to the popular languages are gaining steam -- and may be the perfect fit for your next project

By Peter Wayner

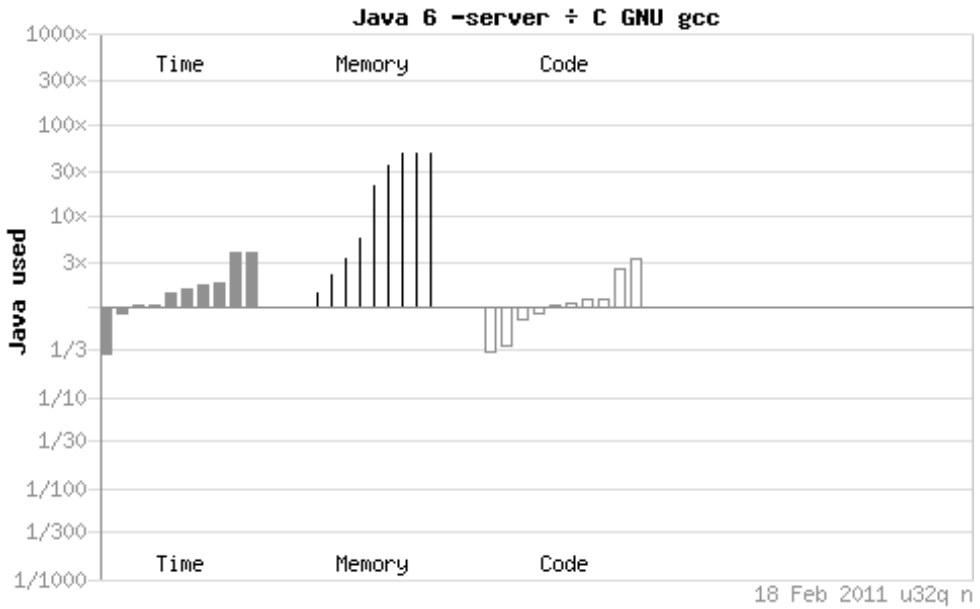
Contributing Editor, InfoWorld | JUL 12, 2017

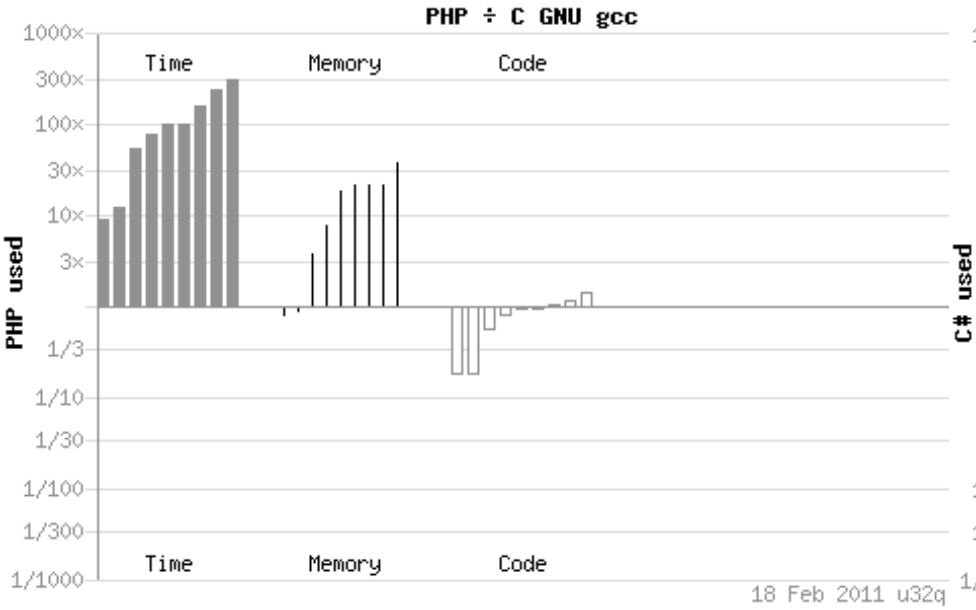
- Kotkin: Java reconsidered
- Erlang: Functional programming for real-time systems
- Go: Simple and dynamic
- OCaml: Complex data hierarchy juggler
- TypeScript: JavaScript you'll like
- Rust: Safe and usable systems language
- ...

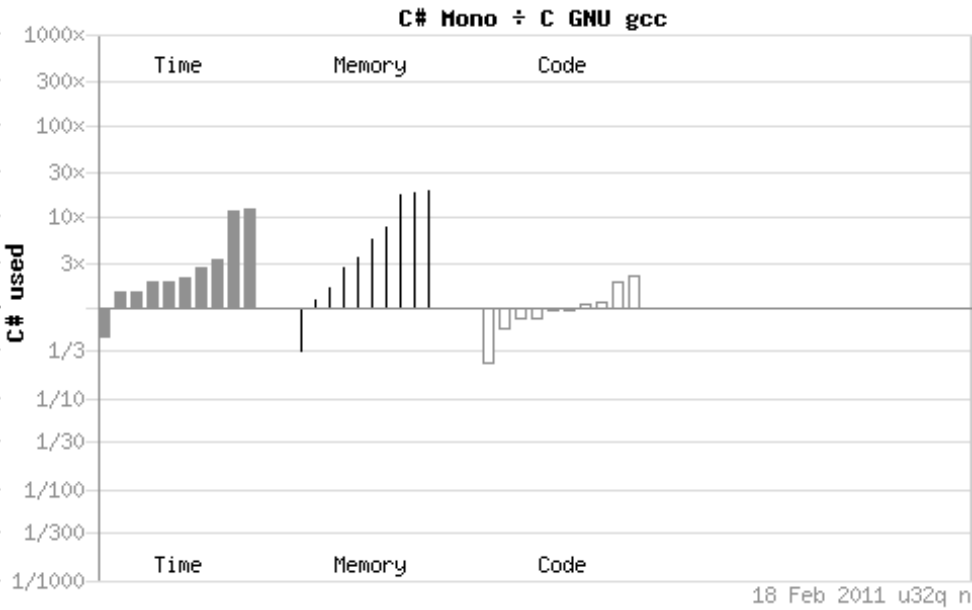
https://www.infoworld.com/article/2840235/application-development/9-cutting-edge-programming-languages-worth-learning-next.html

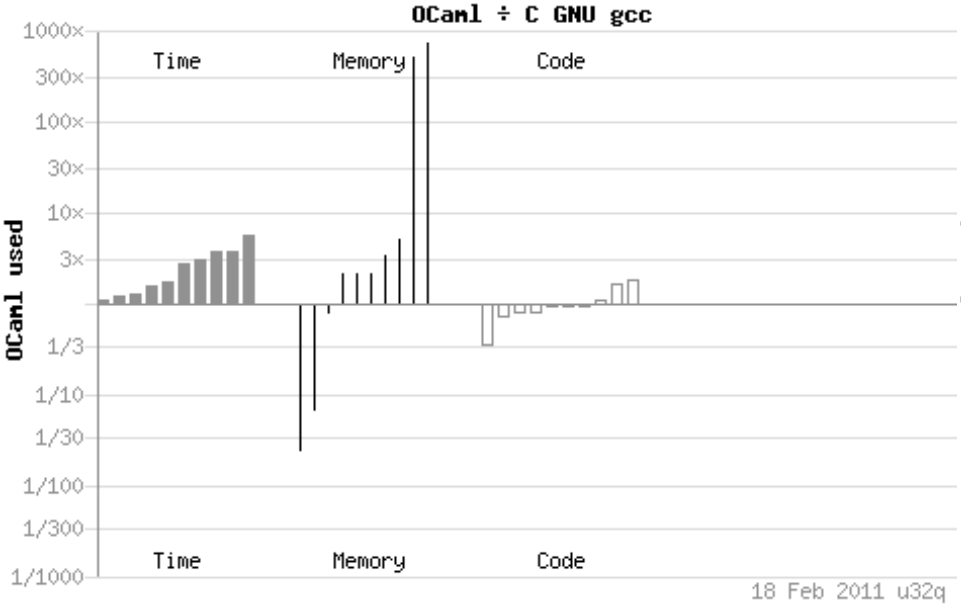
The Computer Language Benchmarks Game

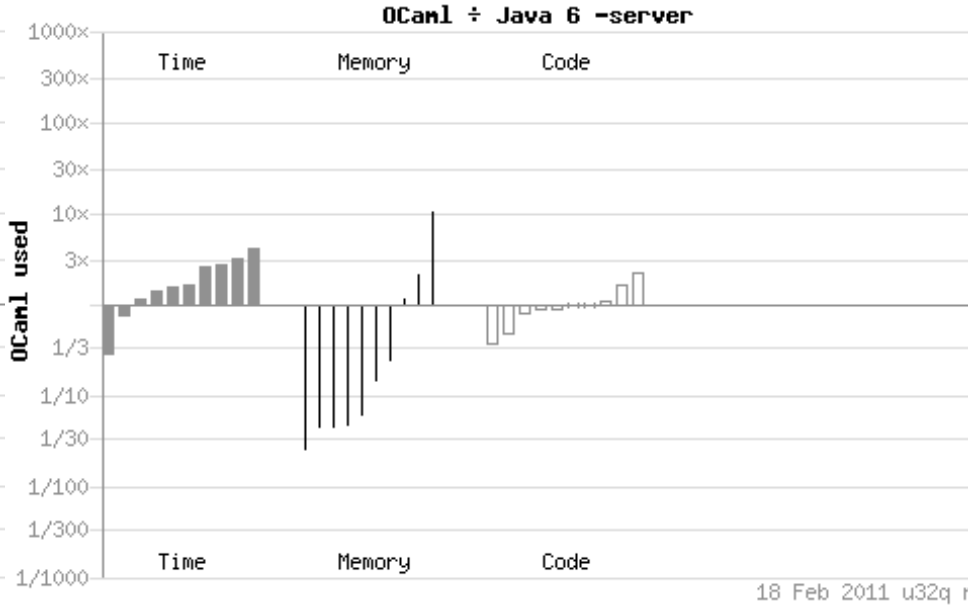
- 33 languages, 4 systems, 13 test programs
- https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html

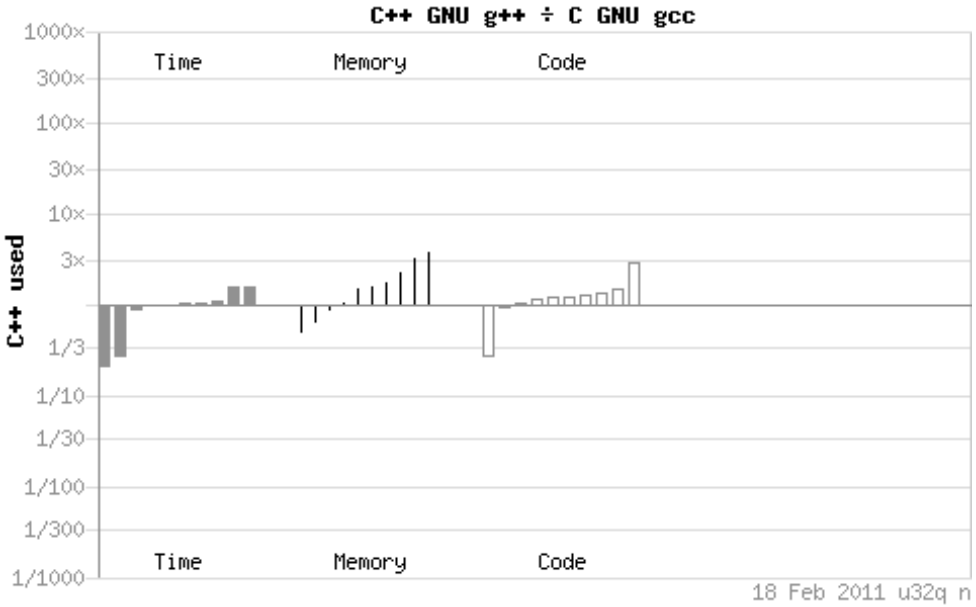



PHP ÷ C GNU gcc
1000×
300×
100×
30×
10×
3×
1/3
1/10
1/30
1/100
1/300
1/1000
PHP used
Time
Memory
Code
Time
Memory
Code
18 Feb 2011 u32q


C# Mono ÷ C GNU gcc
1000×
300×
100×
30×
10×
3×
1/3
1/10
1/30
1/100
1/300
1/1000
C# used
Time
Memory
Code
Time
Memory
Code
18 Feb 2011 u32q n


OCaml ÷ C GNU gcc
1000×
300×
100×
30×
10×
3×
1/3
1/10
1/30
1/100
1/300
1/1000
OCaml used
Time
Memory
Code
Time
Memory
Code
18 Feb 2011 u32q


OCaml ÷ Java 6 -server
1000×
300×
100×
30×
10×
3×
1/3
1/10
1/30
1/100
1/300
1/1000
OCaml used
Time
Memory
Code
Time
Memory
Code
18 Feb 2011 u32q



- High-level programming languages can be very efficient
- Java and Ocaml are efficient languages

The Computer Language Benchmarks Game

OCaml programs versus Java

all other OCaml programs & measurements

by benchmark task performance

reverse-complement

source	secs	KB	gz	cpu	cpu load
OCaml	**0.74**	?	1314	?	63% 61% 42% 57%
Java	1.11	345,308	1661	2.44	33% 58% 54% 80%

fannkuch-redux

source	secs	KB	gz	cpu	cpu load
OCaml	**16.14**	?	1017	?	100% 100% 100% 100%
Java	17.74	30,048	1282	69.90	98% 98% 100% 99%

n-body

source	secs	KB	gz	cpu	cpu load
OCaml	21.69	1,084	1239	21.68	1% 1% 100% 1%
Java	21.50	27,240	1489	21.52	1% 1% 100% 0%

spectral-norm

source	secs	KB	gz	cpu	cpu load
OCaml	4.46	4,792	907	15.50	90% 93% 88% 90%
Java	4.29	31,428	950	16.56	97% 96% 98% 97%

regex-dna

source	secs	KB	gz	cpu	cpu load
OCaml	8.18	?	1050	?	86% 100% 86% 84%
Java	4.26	667,416	1085	11.69	59% 69% 90% 59%

binary-trees

source	secs	KB	gz	cpu	cpu load
OCaml	23.42	173,568	784	70.56	84% 90% 77% 55%
Java	11.52	643,924	889	40.54	89% 92% 85% 89%

mandelbrot

source	secs	KB	gz	cpu	cpu load
OCaml	13.78	7,800	710	54.76	100% 100% 100% 99%
Java	5.89	89,504	796	23.08	98% 98% 98% 99%

k-nucleotide

source	secs	KB	gz	cpu	cpu load
OCaml	19.76	256,940	1789	54.18	43% 96% 43% 94%
Java	8.02	467,004	1802	25.57	76% 98% 73% 74%

fasta

source	secs	KB	gz	cpu	cpu load
OCaml	5.79	202,460	1161	5.79	1% 2% 100% 1%
Java	2.14	36,192	2457	5.68	71% 58% 62% 77%

pidigits

source	secs	KB	gz	cpu	cpu load
OCaml	Make Error				
Java	0.24	828	938	0.24	71% 38% 8% 4%

OCaml	The OCaml native-code compiler, version 4.04.0
Java	java version "1.8.0_121" Java(TM) SE Runtime Environment (build 1.8.0_121-b13) Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)

The Computer Language Benchmarks Game

OCaml programs versus C++ g++

all other OCaml programs & measurements

by benchmark task performance

fannkuch-redux

source	secs	KB	gz	cpu	cpu load
OCaml	16.14	?	1017	?	100% 100% 100% 100%
C++ g++	13.17	1,944	1439	51.41	93% 100% 98% 100%

reverse-complement

source	secs	KB	gz	cpu	cpu load
OCaml	0.74	?	1314	?	63% 61% 42% 57%
C++ g++	0.59	217,564	2275	0.84	26% 78% 12% 34%

regex-dna

source	secs	KB	gz	cpu	cpu load
OCaml	8.18	?	1050	?	86% 100% 86% 84%
C++ g++	3.89	242,200	695	6.50	0% 77% 1% 91%

spectral-norm

source	secs	KB	gz	cpu	cpu load
OCaml	4.46	4,792	907	15.50	90% 93% 88% 90%
C++ g++	2.01	1,792	1044	8.00	100% 100% 100% 100%

n-body

source	secs	KB	gz	cpu	cpu load
OCaml	21.69	1,084	1239	21.68	1% 1% 100% 1%
C++ g++	9.30	1,712	1763	9.29	100% 1% 1% 0%

mandelbrot

source	secs	KB	gz	cpu	cpu load
OCaml	13.78	7,800	710	54.76	100% 100% 100% 99%
C++ g++	5.82	33,952	726	22.40	96% 95% 95% 100%

k-nucleotide

source	secs	KB	gz	cpu	cpu load
OCaml	19.76	256,940	1789	54.18	43% 96% 43% 94%
C++ g++	7.15	164,080	1252	24.21	82% 83% 80% 96%

binary-trees

source	secs	KB	gz	cpu	cpu load
OCaml	23.42	173,568	784	70.56	84% 90% 77% 55%
C++ g++	7.23	362,820	892	24.45	53% 92% 100% 96%

fasta

source	secs	KB	gz	cpu	cpu load
OCaml	5.79	202,460	1161	5.79	1% 2% 100% 1%
C++ g++	1.47	9,484	2291	5.13	88% 88% 87% 88%

pidigits

source	secs	KB	gz	cpu	cpu load
OCaml	Make Error				
C++ g++	0.07	?	508	0.07	0% 13% 100% 0%

OCaml	The OCaml native-code compiler, version 4.04.0
C++ g++	g++ (Ubuntu 5.4.0-6ubuntu1~16.04.1) 5.4.0 20160609

Energy Efficiency across Programming Languages: How does Energy, Time and Memory Relate?

Table 3. Results for binary-trees, fannkuch-redux, and fasta

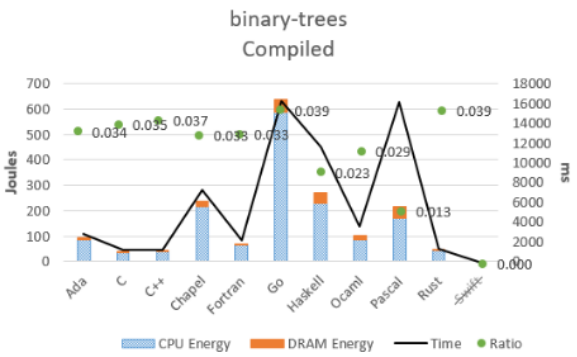
binary-trees				
	Energy	Time	Ratio	Mb
(c) C	39.80	1125	0.035	131
(c) C++	41.23	1129	0.037	132
(c) Rust $\Downarrow_2$	49.07	1263	0.039	180
(c) Fortran $\Uparrow_1$	69.82	2112	0.033	133
(c) Ada $\Downarrow_1$	95.02	2822	0.034	197
(c) Ocaml $\downarrow_1 \Uparrow_2$	100.74	3525	0.029	148
(v) Java $\uparrow_1 \Downarrow_{16}$	111.84	3306	0.034	1120
(v) Lisp $\downarrow_3 \Downarrow_3$	149.55	10570	0.014	373
(v) Racket $\downarrow_4 \Downarrow_6$	155.81	11261	0.014	467
(i) Hack $\uparrow_2 \Downarrow_9$	156.71	4497	0.035	502
(v) C# $\downarrow_1 \Downarrow_1$	189.74	10797	0.018	427
(v) F# $\downarrow_3 \Downarrow_1$	207.13	15637	0.013	432
(c) Pascal $\downarrow_3 \Uparrow_5$	214.64	16079	0.013	256
(c) Chapel $\uparrow_5 \Uparrow_4$	237.29	7265	0.033	335
(v) Erlang $\uparrow_5 \Uparrow_1$	266.14	7327	0.036	433
(c) Haskell $\uparrow_2 \Downarrow_2$	270.15	11582	0.023	494
(i) Dart $\downarrow_1 \Uparrow_1$	290.27	17197	0.017	475
(i) JavaScript $\downarrow_2 \Downarrow_4$	312.14	21349	0.015	916
(i) TypeScript $\downarrow_2 \Downarrow_2$	315.10	21686	0.015	915
(c) Go $\uparrow_3 \Uparrow_{13}$	636.71	16292	0.039	228
(i) Jruby $\uparrow_2 \Downarrow_3$	720.53	19276	0.037	1671
(i) Ruby $\Uparrow_5$	855.12	26634	0.032	482
(i) PHP $\Uparrow_3$	1,397.51	42316	0.033	786
(i) Python $\Uparrow_{15}$	1,793.46	45003	0.040	275
(i) Lua $\downarrow_1$	2,452.04	209217	0.012	1961
(i) Perl $\uparrow_1$	3,542.20	96097	0.037	2148
(c) Swift	n.e.			

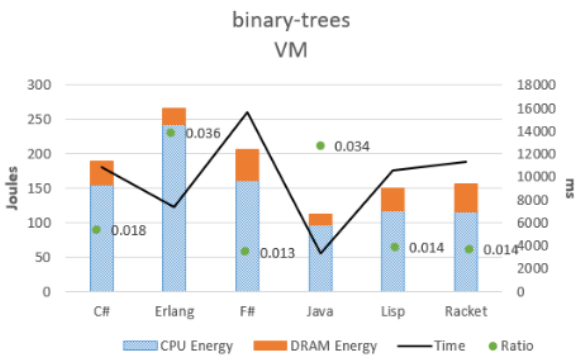
fannkuch-redux				
	Energy	Time	Ratio	Mb
(c) C $\Downarrow_2$	215.92	6076	0.036	2
(c) C++ $\Uparrow_1$	219.89	6123	0.036	1
(c) Rust $\Downarrow_{11}$	238.30	6628	0.036	16
(c) Swift $\Downarrow_5$	243.81	6712	0.036	7
(c) Ada $\Downarrow_2$	264.98	7351	0.036	4
(c) Ocaml $\downarrow_1$	277.27	7895	0.035	3
(c) Chapel $\uparrow_1 \Downarrow_{18}$	285.39	7853	0.036	53
(v) Lisp $\downarrow_3 \Downarrow_{15}$	309.02	9154	0.034	43
(v) Java $\uparrow_1 \Downarrow_{13}$	311.38	8241	0.038	35
(c) Fortran $\Downarrow_1$	316.50	8665	0.037	12
(c) Go $\uparrow_2 \Uparrow_7$	318.51	8487	0.038	2
(c) Pascal $\Uparrow_{10}$	343.55	9807	0.035	2
(v) F# $\downarrow_1 \Downarrow_7$	395.03	10950	0.036	34
(v) C# $\uparrow_1 \Downarrow_5$	399.33	10840	0.037	29
(i) JavaScript $\downarrow_1 \Downarrow_2$	413.90	33663	0.012	26
(c) Haskell $\uparrow_1 \Uparrow_8$	433.68	14666	0.030	7
(i) Dart $\Downarrow_7$	487.29	38678	0.013	46
(v) Racket $\Uparrow_3$	1,941.53	43680	0.044	18
(v) Erlang $\Uparrow_3$	4,148.38	101839	0.041	18
(i) Hack $\Downarrow_6$	5,286.77	115490	0.046	119
(i) PHP	5,731.88	125975	0.046	34
(i) TypeScript $\downarrow_4 \Uparrow_4$	6,898.48	516541	0.013	26
(i) Jruby $\uparrow_1 \Downarrow_4$	7,819.03	219148	0.036	669
(i) Lua $\downarrow_3 \Uparrow_{19}$	8,277.87	635023	0.013	2
(i) Perl $\uparrow_2 \Uparrow_{12}$	11,133.49	249418	0.045	12
(i) Python $\uparrow_2 \Uparrow_{14}$	12,784.09	279544	0.046	12
(i) Ruby $\uparrow_2 \Uparrow_{17}$	14,064.98	315583	0.045	8

fasta				
	Energy	Time	Ratio	Mb
(c) Rust $\Downarrow_9$	26.15	931	0.028	16
(c) Fortran $\downarrow_6$	27.62	1661	0.017	1
(c) C $\uparrow_1 \Downarrow_1$	27.64	973	0.028	3
(c) C++ $\uparrow_1 \Downarrow_2$	34.88	1164	0.030	4
(v) Java $\uparrow_1 \Downarrow_{12}$	35.86	1249	0.029	41
(c) Swift $\Downarrow_9$	37.06	1405	0.026	31
(c) Go $\downarrow_2$	40.45	1838	0.022	4
(c) Ada $\downarrow_2 \Uparrow_3$	40.45	2765	0.015	3
(c) Ocaml $\downarrow_2 \Downarrow_{15}$	40.78	3171	0.013	201
(c) Chapel $\uparrow_5 \Downarrow_{10}$	40.88	1379	0.030	53
(v) C# $\uparrow_4 \Downarrow_5$	45.35	1549	0.029	35
(i) Dart $\Downarrow_6$	63.61	4787	0.013	49
(i) JavaScript $\Downarrow_1$	64.84	5098	0.013	30
(c) Pascal $\downarrow_1 \Uparrow_{13}$	68.63	5478	0.013	0
(i) TypeScript $\downarrow_2 \Downarrow_{10}$	82.72	6909	0.012	271
(v) F# $\uparrow_2 \Uparrow_3$	93.11	5360	0.017	27
(v) Racket $\downarrow_1 \Uparrow_5$	120.90	8255	0.015	21
(c) Haskell $\uparrow_2 \Downarrow_8$	205.52	5728	0.036	446
(v) Lisp $\Downarrow_2$	231.49	15763	0.015	75
(i) Hack $\Downarrow_3$	237.70	17203	0.014	120
(i) Lua $\Uparrow_{18}$	347.37	24617	0.014	3
(i) PHP $\downarrow_1 \Uparrow_{13}$	430.73	29508	0.015	14
(v) Erlang $\uparrow_1 \Uparrow_{12}$	477.81	27852	0.017	18
(i) Ruby $\downarrow_1 \Uparrow_2$	852.30	61216	0.014	104
(i) JRuby $\uparrow_1 \Downarrow_2$	912.93	49509	0.018	705
(i) Python $\downarrow_1 \Uparrow_{18}$	1,061.41	74111	0.014	9
(i) Perl $\uparrow_1 \Uparrow_8$	2,684.33	61463	0.044	53

https://sites.google.com/view/energy-efficiency-languages/home
http://greenlab.di.uminho.pt/wp-content/uploads/2017/10/sleFinal.pdf

Energy Efficiency across Programming Languages: How does Energy, Time and Memory Relate?





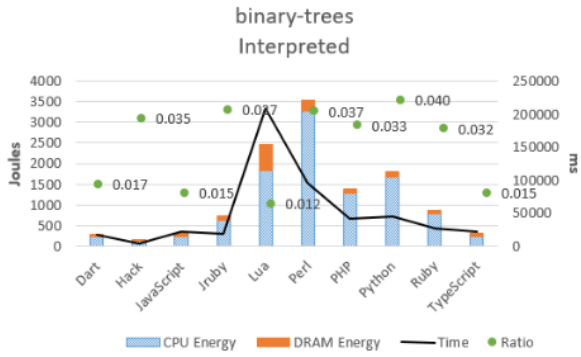
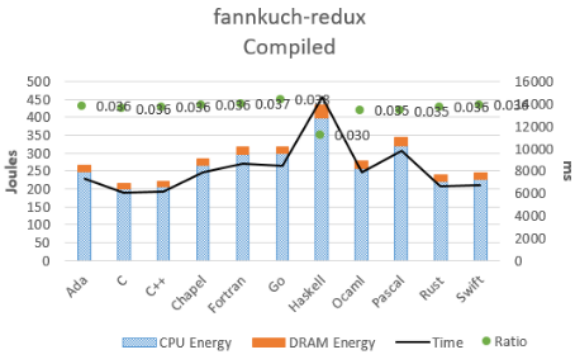
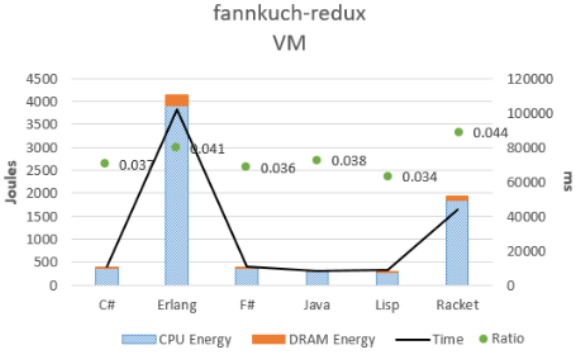
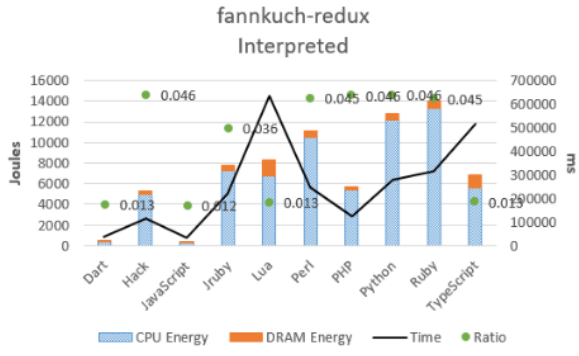


Figure 1. Energy and time graphical data for binary-trees







https://sites.google.com/view/energy-efficiency-languages/home
http://greenlab.di.uminho.pt/wp-content/uploads/2017/10/sleFinal.pdf

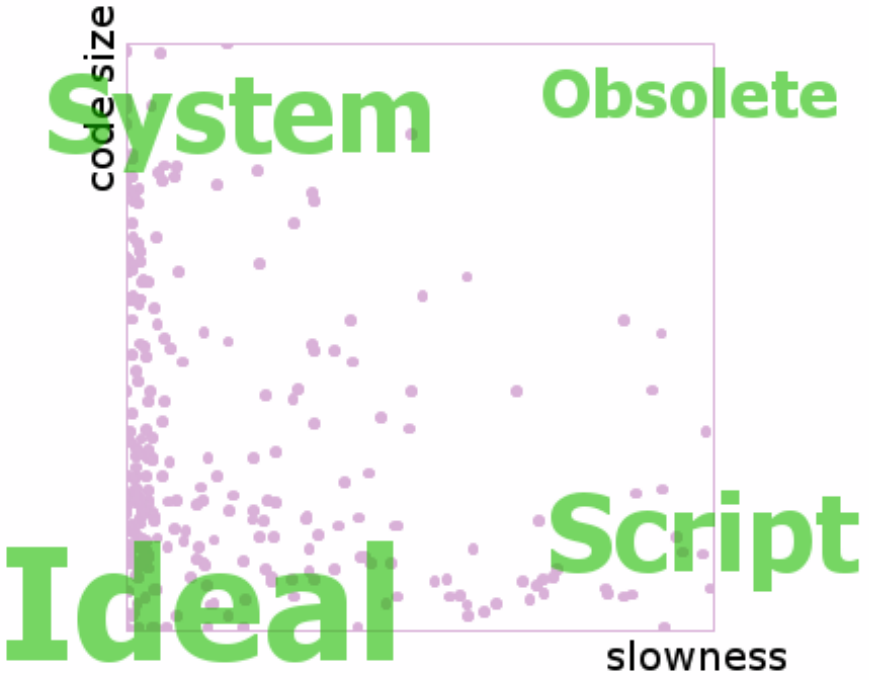
Speed, size and memory usage of PL

http://blog.gmarceau.qc.ca/2009/05/speed-size-and-dependability-of.html

429 programs

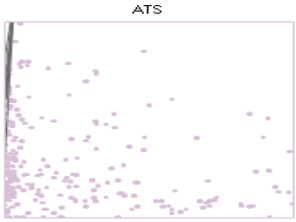
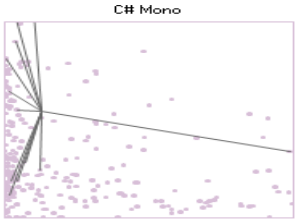
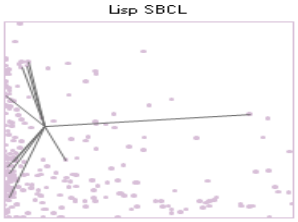
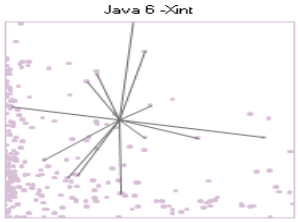
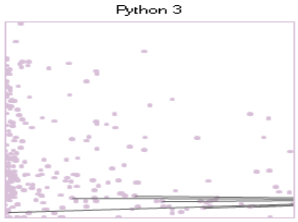
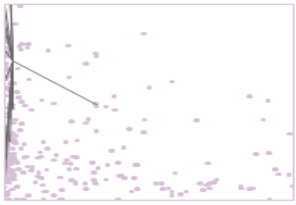
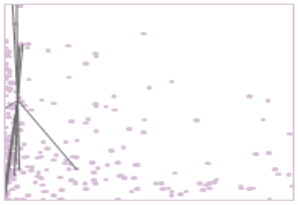
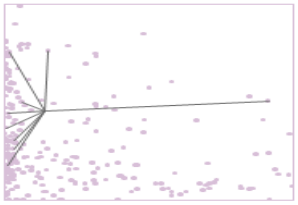
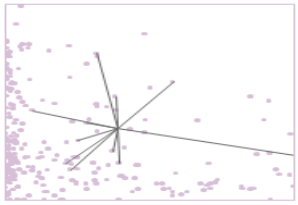
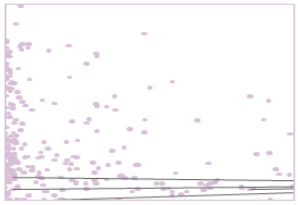
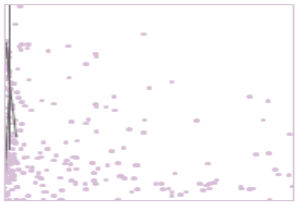
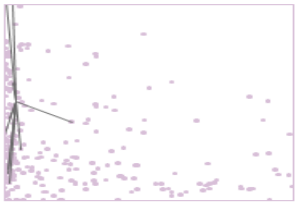
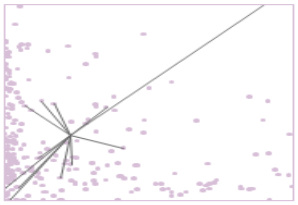
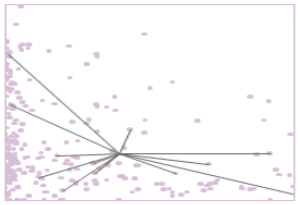
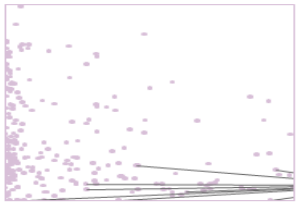
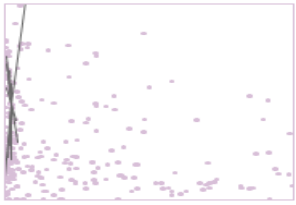
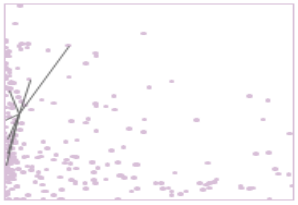
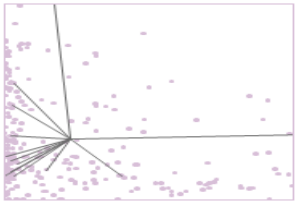
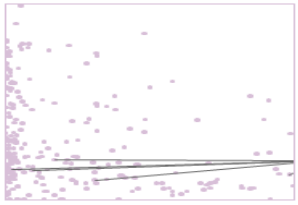
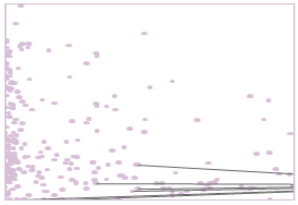
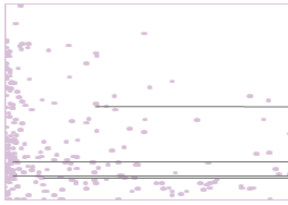
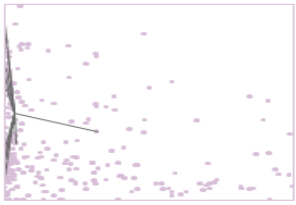
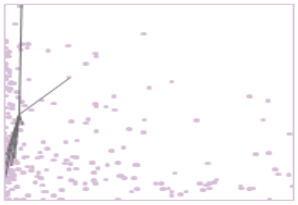
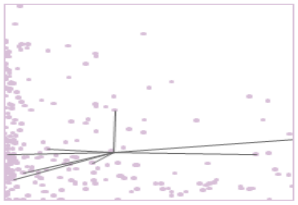
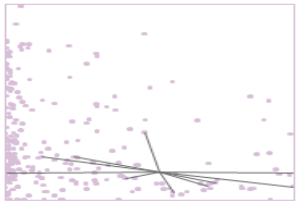
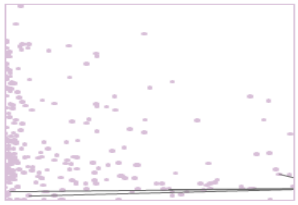
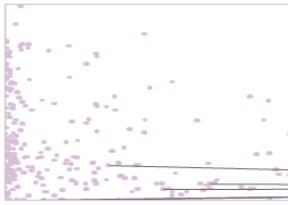
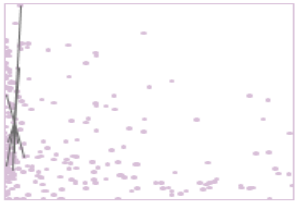
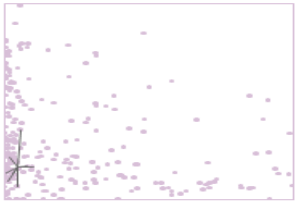
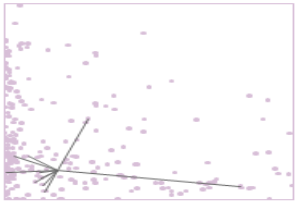
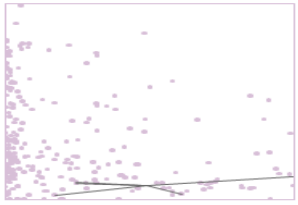
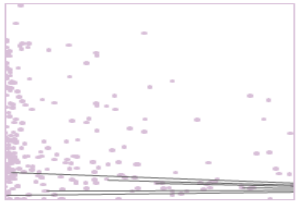
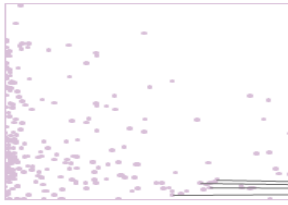
13 test programs

33 PL



Experiment:

Guillaume Marceau


ATS

C# Mono

Lisp SBCL

Java 6 -Xint

Python 3

C++ GNU g++

Haskell GHC

Ada 2005 GNAT

Smalltalk VisualWorks

Perl

*Java 6 steady state

Pascal Free Pascal

Erlang HiPE

Scheme PLT

Ruby JRuby

Java 6 -server

Lisaac

Scala

Mozart/Oz

Ruby 1.9

PHP

C GNU gcc

Fortran Intel

F# Mono

Lua

JavaScript TraceMonkey

Ruby

Clean

OCaml

Lua LuaJIT

JavaScript V8

Python

Python pypy

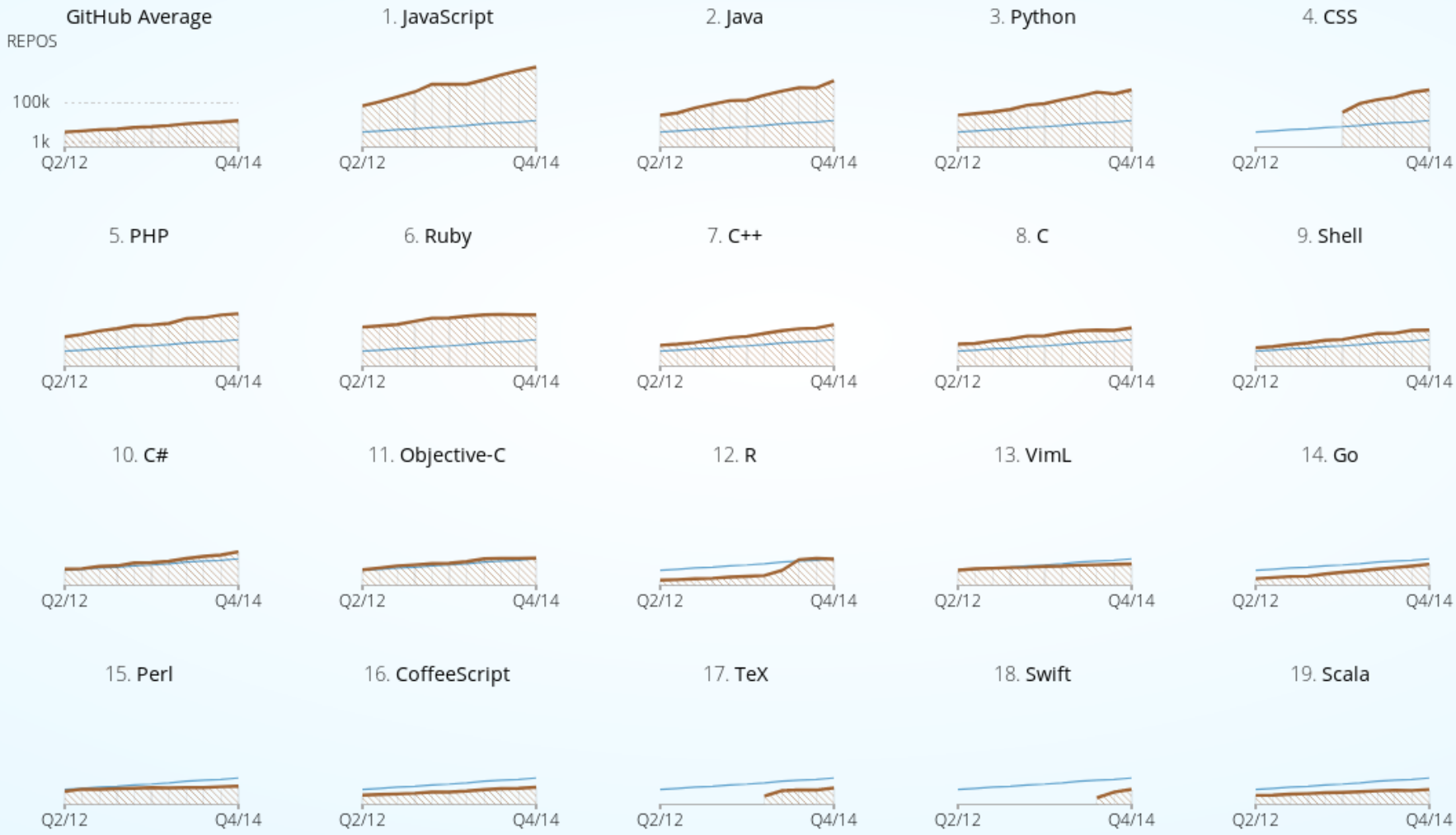
githut

- Programming languages on github, 2014
- http://githut.info/
- 2M active repositories
- Parameters
  - No. of projects on github
  - No. of push operations
  - No. of push/repository operations
  - No. of fork operations
  - No. of opened issues
  - No. of watchers

TOP ACTIVE LANGUAGES

A split by language view of active repositories

%

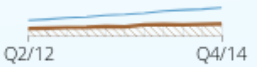


20. Emacs Lisp

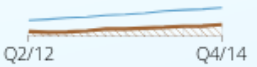
21. Haskell

22. Lua

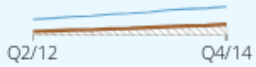
23. Clojure



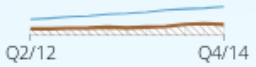
24. Matlab



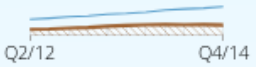
25. Arduino



26. Groovy



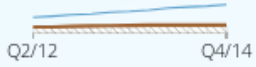
27. Puppet



28. Rust

29. PowerShell

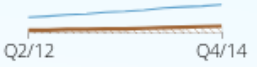
30. Erlang



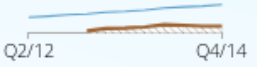
31. Visual Basic

32. Processing

33. Assembly

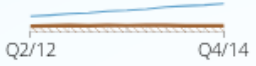


34. TypeScript

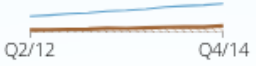


35. XSLT

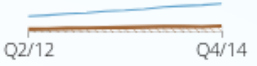
36. ActionScript



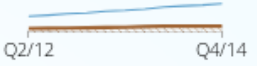
37. ASP



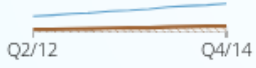
38. OCaml



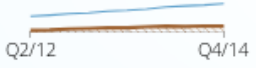
39. D



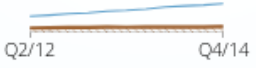
40. Scheme



41. Dart



42. Common Lisp



43. Julia

44. F#

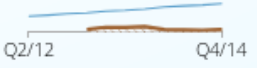
45. Elixir

46. FORTRAN

47. Haxe

48. Racket

49. Logos



TIOBE Indeks

- https://www.tiobe.com/tiobe-index/
- Indeks based on:
  - Web search engines: Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube, Baidu
- TIOBE index is NOT:
  - About best PL
  - About the most used PL

Feb 2018	Feb 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.988%	-1.69%
2	2		C	11.857%	+3.41%
3	3		C++	5.726%	+0.30%
4	5	^	Python	5.168%	+1.12%
5	4	v	C#	4.453%	-0.45%
6	8	^	Visual Basic .NET	4.072%	+1.25%
7	6	v	PHP	3.420%	+0.35%
8	7	v	JavaScript	3.165%	+0.29%
9	9		Delphi/Object Pascal	2.589%	+0.11%
10	11	^	Ruby	2.534%	+0.38%
11	-	^^	SQL	2.356%	+2.36%
12	16	^^	Visual Basic	2.177%	+0.30%
13	15	^	R	2.086%	+0.16%
14	18	^^	PL/SQL	1.877%	+0.33%
15	13	v	Assembly language	1.833%	-0.27%
16	12	vv	Swift	1.794%	-0.33%
17	10	vv	Perl	1.759%	-0.41%
18	14	vv	Go	1.417%	-0.69%
19	17	v	MATLAB	1.228%	-0.49%
20	19	v	Objective-C	1.130%	-0.41%

Feb 2019	Feb 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.876%	+0.89%
2	2		C	12.424%	+0.57%
3	4	⌃	Python	7.574%	+2.41%
4	3	⌄	C++	7.444%	+1.72%
5	6	⌃	Visual Basic .NET	7.095%	+3.02%
6	8	⌃	JavaScript	2.848%	-0.32%
7	5	⌄	C#	2.846%	-1.61%
8	7	⌄	PHP	2.271%	-1.15%
9	11	⌃	SQL	1.900%	-0.46%
10	20	⌃⌃	Objective-C	1.447%	+0.32%
11	15	⌃⌃	Assembly language	1.377%	-0.46%
12	19	⌃⌃	MATLAB	1.196%	-0.03%
13	17	⌃⌃	Perl	1.102%	-0.66%
14	9	⌄⌄	Delphi/Object Pascal	1.066%	-1.52%
15	13	⌄	R	1.043%	-1.04%
16	10	⌄⌄	Ruby	1.037%	-1.50%
17	12	⌄⌄	Visual Basic	0.991%	-1.19%
18	18		Go	0.960%	-0.46%
19	49	⌃⌃	Groovy	0.936%	+0.75%
20	16	⌄⌄	Swift	0.918%	-0.88%

Feb 2020	Feb 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.358%	+1.48%
2	2		C	16.766%	+4.34%
3	**3**		**Python**	**9.345%**	**+1.77%**
4	4		C++	6.164%	-1.28%
5	7	⌃	C#	5.927%	+3.08%
6	5	⌄	Visual Basic .NET	5.862%	-1.23%
7	6	⌄	JavaScript	2.060%	-0.79%
8	8		PHP	2.018%	-0.25%
9	9		SQL	1.526%	-0.37%
10	20	⌃⌃	Swift	1.460%	+0.54%
11	18	⌃⌃	Go	1.131%	+0.17%
12	11	⌄	Assembly language	1.111%	-0.27%
13	15	⌃	R	1.005%	-0.04%
14	23	⌃⌃	D	0.917%	+0.28%
15	16	⌃	Ruby	0.844%	-0.19%
16	12	⌄⌄	MATLAB	0.794%	-0.40%
17	21	⌃⌃	PL/SQL	0.764%	-0.05%
18	14	⌄⌄	Delphi/Object Pascal	0.748%	-0.32%
19	13	⌄⌄	Perl	0.697%	-0.40%
20	10	⌄⌄	Objective-C	0.688%	-0.76%

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	22	21	-	-
C++	4	4	4	3	2	1	2	12
C#	5	5	5	8	8	-	-	-
Visual Basic .NET	6	10	-	-	-	-	-	-
JavaScript	7	8	8	9	6	-	-	-
PHP	8	6	3	4	27	-	-	-
SQL	9	-	-	97	-	-	-	-
Objective-C	10	3	21	37	-	-	-	-
Lisp	31	18	16	13	14	5	3	2
Ada	35	29	24	15	15	6	4	3
Pascal	229	16	13	65	11	3	15	5

Programming Language Hall of Fame

The hall of fame listing all "Programming Language of the Year" award winners is shown below. The award is given to the programming language that has the highest rise in ratings in a year.

Year	Winner
2019	C
2018	Python
2017	C
2016	Go
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++