A Longitudinal Analysis of the Reid List of First Programming Languages

Robert M. Siegfried
siegfrie@adelphi.edu

Jason P. Siegfried
jasonpsiegfried@yahoo.com

Gina Alexandro
ginaalexandro@gmail.com

Department of Mathematics and Computer Science
Adelphi University
Garden City, NY 11530 USA

Abstract

Throughout the 1990s, Richard Reid of Michigan State University maintained a list showing the first programming language used in introductory programming courses taken by computer science and information systems majors; it was updated for several years afterwards with the most recent update done in 2011. This is a follow-up to that last update of the Reid List. A newer list is shown and compared to the results of four years ago. The changes in popularity of different programming languages are discussed as well as some of the reasons for these changes.

Keywords: introductory programming, programming languages, objects early approach, Java, C++, Python.

1. INTRODUCTION

The choice of a programming language for an introductory programming course has been a topic of debate for over forty years, and the academic community has seen a variety of programming languages gain and then subsequently lose popularity. The difficulties that students encountered compiling Fortran programs with industry-standard compilers led Shantz et al. (1967) to develop WATFOR, a Fortran compiler for student use. Holt (1973) considered the use of PL/I a terrible way of teaching introductory programming. Kernighan (1981) described Pascal as "meant for learning" but unsuitable for serious programming, an assessment with which Haberman (1973) would have concurred. In 1996, Brilliant and Wiseman described Pascal as dated, an assessment with which many educators agreed.

The question that computer science educators have tried to answer since then is whether there exists an ideal language to use when teaching college freshmen how to program. Johnson (1995) considered C too complex a language for beginning programmers. Many college programs switched to using C++ in their introductory programming course, and the Advanced Placement exam in Computer Science switched as well. More recently, the Advanced Placement exam switched to Java, and so did many introductory programming courses. Java was widely considered an easier language to learn than C++ (Hadjerroult 1998; Madden 2002).

While the TeachRacket (originally called TeachScheme) approach has been around for a decade (Felleisen et al. 2004), it is only used in a relatively small number of colleges. More recently, Python has become somewhat popular. Mason and Cooper (2014) found that it has become widely used in programming courses in Australia and New Zealand by programs that choose not to focus on object oriented programming in the first course.

The choice of a programming language to be used when teaching introductory programming has been a "hot button" topic within the computer science and information systems communities. While the AP Computer Science curriculum still uses Java, there are many programs that are questioning whether this is the language that they ought to be using.

The study is a follow-up to a study in 2011 to determine the language of choice in computer science programs (Siegfried et al. 2012). Since most information systems majors take the same introductory programming courses as computer science majors, this is a question that should be of great interest to both communities.

Since the move away from Pascal in the 1990s, it has become more difficult to find consensus on the choice of first programming language to replace it.

2. THE REID LIST

Richard Reid taught computer science at Michigan State University and he began tracking the languages used to teach introductory programming to CS majors in the early 1990s. The List was updated when 10% or more of the included colleges changed the programming language of instruction (Reid 1992). This resulted in a new list being released approximately twice a year until Reid retired in 1999. Frances Van Scoy, a former student of Professor Reid's, updated the list until 2006.

The twenty-fourth Reid List included 410 colleges and universities, with 391 of the colleges representing the District of Columbia and 49 states (Wyoming is the only state without representation). A breakdown by region appears in Table 1. While there is reasonable geographic balance, the mid-Atlantic and southwestern states are overrepresented by the large number of schools in New York, California and Pennsylvania that are on the List. Additionally, the New England states as a whole are significantly overrepresented in comparison to its college-age population, partially due to the presence of all eight Ivy League colleges and MIT.

Table 1. Geographic Breakdown Of The US Colleges In The Reid List

Region	Colleges
New England	41
MidAtlantic (incl. DC)	87
Southeast	72
Kentucky and W. Virginia	10
MidWest	95
SouthWest	68
Northwest	16
Alaska and Hawaii	2

Table 2 shows the breakdown by the highest degree program offered in computing. There is an almost even breakdown between undergraduate, master's- and doctorate-granting departments; however, only nine of the programs were in community colleges, which are significantly underrepresented. There was one vocational/technical school on the list. This was removed from the List because it no longer offered a computing program. Six are no longer on the List for this reason. Lebanon College was removed because it closed.

Table 2. Breakdown by Highest Degree Offered in Computing

Highest Degree Awarded in Computing	Colleges
Associate's	9
Bachelor's	128
Master's	109
Doctorate	157
No longer offering a computing program	7

3. METHODOLOGY

The colleges and universities included in this survey were taken from the twenty-fourth Reid List; many of the 410 schools listed on the twenty-fourth list did not appear on the twenty-fifth list, which only listed 153 schools. The requirements for the Bachelor's program in Computer Science were examined to determine what the first required programming course was. If the school offered both Bachelor of Arts and Bachelor of Science programs, the requirements for the BS were used. In the case of the

community colleges, the requirements for an Associate's degree in Computer Science were examined. Finally, if the school did not have a Computer Science program, the requirements for the Information Systems program were used.

After finding the first programming course, the course description was examined to see if it included the programming language of instruction; however, most did not specify the language. If a current syllabus for the course was available online, then an examination of its content was used to make a determination of the language used in the course. However, if there was no syllabus online, the bookstore's web site was checked for a textbook adoption; in some cases, the bookstore was called in an attempt to get this information. Lastly, if these steps did not provide the programming language in use, then members of the department were contacted to obtain this information.

4. THE TWENTY-SEVENTH REID LIST

Table 3. The Programming Languages Used And The Frequency Of Occurrence

Language	Programs Using it
Java	180
Python	76
C++	74
C	22
Scheme or Racket	9
C/C++	4
JavaScript	2
Visual Basic	2
Ada	1
C#	1
C++ or Java	1
C++ or Scheme	1
C++ then Python	1
Haskell	1
Java and C	1
Java or C++	1
PHP and C	1
Processing or Python or C#	1
Python and C	1
Python or C0	1
Python or Matlab	1
Python or C++	1
Matlab	1
R	1
Scala	1
Visual Basic and Java	1

Of the 398 schools still offering computing programs, we were able to determine the language or languages used in 387 schools. Each language (or combination of languages) is shown in Table 3. Java is still the most common choice of language, with one hundred eighty schools using it, more than double its nearest competitor. It is followed by Python and C++, with seventy-six and seventy-four schools respectively using them. These three languages account for three hundred and thirty of the three hundred eighty-seven schools. Scheme and Racket (a Scheme-derived language) are used at 9 schools, JavaScript and Visual Basic are each used at two schools and there are one school each using Ada, C#, Matlab and Scala. In addition to these languages, there are several programs that use more than language in a given course (in some cases, the choice of language is left to the instructor) and several that offer more than one course with which a student can begin their computing major.

These two most recent lists, the Twenty–seventh list and the twenty-sixth list (which was compiled in 2011) are compared in table 4. The changes in the popularity of the top eight languages on the list are significant: Java's popularity declined somewhat while Python's popularity grew significantly. While C++ is used in beginning programming courses in 4 fewer schools, there are 3 more schools using C than four years. Scheme and Racket's popularity faded somewhat; its significance is noteworthy because the drop represents 25% of the Reid List schools using it in 2011. Java Script appears on the current list after not appearing in 2011 and both Visual Basic and Ada have almost disappeared from the list; where Visual Basic was used in 8 schools (either exclusively or followed by Java), it is now used in only 3 computing programs. While Ada was used in 6 schools in 2001, it is now used in only one. It is also interesting to note the appearance of several languages of lesser popularity that were not on the previous list. These include PHP, Matlab, R and Scala.

Table 5 shows details that one would missed in a simple comparison. Programs that adopted C were more likely to switch from Java than from C++ or Python. And while more programs abandoned Java for Python by a large margin, other programs switched from Java to C, C#, C++ and R. There were four schools that abandoned Python for other languages which included C, C++, Java and Scala.

5. DISCUSSION

Adelphi University switched from C++ to Java in 2002 because Nassau Community College, from

which Adelphi receives a large number of transfer students, had switched to Java and in anticipation of the change in the Advanced Placement exam in Computer Science. There was also a general impression that more computer-savvy students expected to learn Java and its being an object-oriented language made it seem like the immediate future of computer science.

Mason, Cooper , and deRaadt (2012) found that most Australian computing programs based their choice of a language on its perceived pedagogic benefits and its popularity in private industry. Yet 35% of computing programs that Davies, Polack-Wahl and Anewalt (2011) surveyed taught CS1 programming courses where object-oriented programming was not taught. If one is not teaching objects early, or especially if one is not teaching objects at all in a first programming class, why use Java? This led Elliot Koffman to comment on the SIGCSE mailing list (Beaubouef and Mason 2005), "I fear that we have reinvented the 'new math' syndrome and many of us are unaware of it." Decker and Hirschfield (1994) laid out the case for teaching objects early; but there has been no empirical evidence that the objects early approach makes it easier for students to learn object oriented program than an objects later approach does. Bloch (2011) said that teaching objects early gives students an opportunity to see the difficulties in designing classes before they can possibly appreciate any of the benefits.

Prendergast (2006) wrote about the frustration in learning Java and teaching it to beginning programming students. Nor is he alone; several instructors wrote in their e-mail replies about how much easier it was to teach introductory programming in Python and a few other languages. Yadin (2011) saw fewer students fail their programming course when Java was replaced by Python as the programming language of the course.

Many instructors stated that they are still using Java or C++ in a second programming course; the implication is that they are first covering objects in their second course. This shift from one language in CS1 to an object-oriented language in CS 2 corroborates what Davies et al. found in their 2011 study.

The TeachScheme! approach has been heralded as the savior of computer science by its proponents. Bloch has written about the use of Scheme as an introduction to programming before transitioning to Java, crediting it with curtailing attrition in the CS2 course. Yet only one of the schools using Scheme on the sixth Reid List in 1992 was still listed as using it in the twenty-sixth list in 2011. And Scheme and its derivative language Racket are currently only used in 9 schools on the twenty-seventh Reid List. Bloch's former college stopped using Scheme in 2006 in the CS1 course and is currently phasing it out from the programming course for non-majors. A faculty member (Anonymous 2011) at a Reid List school that switched away from Racket explained that the decision was made to reverse the heavy attrition rate in their major after the first programming course. A faculty member (Anonymous 2015) at another Reid List school said that they dropped it because "the Scheme enthusiast finally retired."

The shift toward Python should not be surprising given McIver's (2002) observation about how large an elementary Java program is compared to a comparable program in Python or in C. Mannila and de Raadt examined the suitability of several programming languages for use in an introductory course and favored Python and Eiffel although they did find some merit in Java.

It is difficult to believe that this debate will be resolved any time soon. There was a time when it seemed like PL/I or Pascal would be the programming language of the future of computer science education. And Python is not without its critics. Michael Main (private communication, 2009) indicated that he considers it important that students learn how and why to declare the data types of variables in a program. Similarly, a programmer friend of a colleague said that he did not prefer Python because it is harder to locate certain types of bugs due to its lack type checking (Chays, private communication, 2015).

There will most likely be another language that will usurp the place that Python currently hold in the hearts of computer science faculty. And it will most likely be the cause of continued debate within the computer science and information systems communities.

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the computer science and information systems faculty at the colleges and universities on the Reid List for your gracious assistance in collecting this data.

7. REFERENCES

Anonymous (2011). Private communication.

Anonymous (2015) Private e-mail.

Beaubouef, T., & Mason, J. (2005). Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations. ACM Inroads 37(2), 103-106.

Bloch, S. (2000). Scheme and Java in the First Year. Journal of Computing in Small Colleges 15(5), 157-165.

Bloch, S. (2011). Private communication.

Brilliant, S. S. & Wiseman, T. (1996). The First Programming Paradigm and Language Dilemma. ACM SIGCSE Bulletin 28(1), 338-342.

Chays, D. (2015) Private communication.

Davies, S., Polack-Wahl, J. A., & Anewalt, K. (2011). A Snapshot of Current Practices In Teaching the Introductory Programming Sequence. ACM SIGCSE Bulletin 43(1), 625-630.

Decker, R. & Hirschfield, S. (1994). The Top Ten Reasons Why Object Oriented Programming Can't be Taught in CS 1, ACM SIGCSE Bulletin 26(1), 51-55.

Felleisen , M., Findler, R. B., Flatt M., & Krishnamurthi, S. (2004). The TeachScheme! Project: Computing and Programming for Every Student. Computer Science Education 14(1), 55-57.

Haberman, A. N. (1973). Critical Comments on the Programming Language Pascal.", Acta Informatica 3 (1973), p. 47-57.

Hadjerroult, S. (1998). Java As First Programming Language: A Critical Evaluation. ACM SIGCSE Bulletin 30(2), 43-47.

Holt, R. C. (1973). Teaching the Fatal Disease or Introductory Computer Programming Using PL/I. ACM SIGPLAN Notices 8(5), 8-23.

Johnson, L. F. (1995). C In The First Course Considered Harmful. Communications of the ACM 38(5), 99-101.

Kernighan, B. W. (1981). Why Pascal Is Not My Favorite Programming Language. Computing Science Technical Report No. 100, AT&T Bell Laboratories.

Madden, M. & and Chambers, D. (2002). Evaluation of Student Attitudes to Learning the Java Language. Proceeding of Conference on the Principles and Practice of Programming in Java, (Trinity College Dublin), 125-130.

Main, M. (2009). Private communication.

Mason, R., Cooper G., & De Raadt, M. (2012). Trends In Introductory Programming Courses in Australian Universities – Languages, Environments and Pedagogy. Proceeding of the Fourteenth Australasian Computing Education Conference (ACE 2012), Melbourne, Australia. Conferences in Research and Practice in Information Technology 123.

Mason, R., & Cooper, G. (2014). Introductory Programming Courses in Australia and New Zealand in 2013 – Trends and Reasons. ACE '14 Proceedings of the Sixteenth Australasian Computing Education Conference 148, 139-147.

McIver, L. (2002). Syntactic and Semantic Issues in Introductory Programming Education. Ph. D. Thesis. Monash University, Melbourne Australia. Received from the author, June 30, 2007.

Prendergast, M. O. (2006). Teaching Introductory Programming to IS Students: Java Problems and Pitfalls. Journal of Information Technology Education, 5(1), 491-515. Retrieved from http://www.editlib.org/p/111559/, July 15, 2015.

Reid, R. J. (1992). First Course Language for Computer Science Majors. Retrieved from http://www.csee.wvu.edu/~vanscoy/REID06.html on July 12, 2011

Shantz, P. W., German, R. A., Mitchell, J. G., Shirley, R. S. K, & Zernike, C. R.(1967). WATFOR – The University of Waterloo FORTRAN IV Compiler. Communications of the ACM, 10(1), 41.44.

Siegfried, R. M., Greco, D. M., Miceli, N. G., & P. Siegfried, J. P (2012). Whatever Happened to Richard Reid's List of First Programming Languages? Information Systems Education Journal, 10(4), 24-30. Retrieved from http://isedj.org/2012-0/N4/ISEDJv10n4p24.pdf on July 5, 2015.

Van Scoy, F. (2006). Reid List 25. Retrieved from htpp://groups.google.edu/group/comp.edu/

browse_thread/thread/4f00b5f437ce261a/3267514419052033?q=Reid+List#3267514419052033 on July 15, 2011.

Yadin, A. (2011). Reducing the Dropout Rate in an Introductory Programming Course. ACM Inroads, 2(4), 71-76.

Appendices and Annexures

Table 4. A Comparison of the Twenty-Seventh and Twenty-Sixth Reid Lists

	2015	2011
Java	180	197
Python	76	41
C++	74	80
C	22	19
Scheme or Racket	9	12
C/C++	4	4
JavaScript	2	0
Visual Basic	2	7
Ada	1	5
C#	1	0
C++ or Java	1	2
C++ or Scheme	1	0
C++ then Python	1	0
Haskell	1	1
Java and C	1	0
Java or C++	1	0
PHP and C	1	0
Processing or Python or C#	1	0
Python and C	1	0
Python or C0	1	0
Python or Matlab	1	0
Python or C++	1	0
Matlab	1	0
R	1	0
Scala	1	0
Visual Basic and Java	1	0
Ada or Python	0	1
Alice	0	1
Alice and Java	0	2
Java or Matlab	0	2
Java or Python	0	1
Java or Scheme	0	1
Processing	0	1
Processing and Java	0	1
Python and Java	0	1
Python or Java	0	1
Python Or C# or Matlab	0	1
Various Languages	0	1
Visual C# or Visual Basic	0	1

Table 5. The Languages That Computing Programs Adopted And The Language From Which They Switched:

Schools adopting:	Changed from:	
C	C++	2
C	Java	4
C	Python	1
C#	Java	1
C++	C	1
C++	Java	9
C++	Processing/Java	1
C++	Python	1
Java	Ada	3
Java	C	1
Java	C#	1
Java	C++	11
Java	Processing	1
Java	Python	2
JavaScript	Visual Basic	1
PHP and C	C++	1
Python	Ada	2
Python	Alice/Java	1
Python	C	1
Python	C++	6
Python	Java	28
Python	Scheme	2
R	Java	1
Scala	Python	1