Trends Of Commonly Used Programming Languages in CS1 And CS2 Learning

1st Robert M. Siegfried
Department of Mathematics and Computer Science
Adelphi University
Garden City, NY, USA
siegfrie@adelphi.edu

2nd Katherine G. Herbert-Berger
Department of Computer Science
Montclair State University
Montclair, NJ, USA
herbertk@montclair.edu

3rd Kees Leune
Department of Mathematics and Computer Science
Adelphi University
Garden City, NY, USA
leune@adelphi.edu

4th Jason P. Siegfried
Department of Mathematics and Computer Science
Adelphi University
Garden City, NY, USA
jasonsiegfried@mail.adelphi.edu

2021 16th International Conference on Computer Science & Education (ICCSE) | 978-1-6654-1468-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCSE51940.2021.9569444

***Abstract*—Computer science educators have tried to identify the best language for their first-year college students to use when learning how to program. Consequently, selecting a computer programming language for use in an introductory programming course has been a hot-button topic within computing education communities. This paper builds on the work started by Richard Reid in the early 1990s, which surveys institutions providing post-secondary education. It provides educators with up-to-date information about common choices for computer programming languages used in first computer programming courses. This year's survey results indicate clearly that, at the moment, Java and Python are the most commonly used languages. A new element in the survey is that we have now started collecting data regarding the second programming course (CS2). Our findings show that 88% of all surveyed schools only use one of four languages (Java, Python, C++, and C), with the adoption of Python growing steadily at the expense of the other languages. Java continues to be the most popular choice for CS2 courses, followed by C++ after a significant gap. However, we are finding potential trends where institutions that start in Java or C++ tend to stay with their choice, while schools starting with other languages are likely to transition their students over to a second programming language in CS2.**

I. Introduction

During the last four decades, many languages have been used for teaching introductory programming. The language choice is usually made locally, based on factors such as faculty preference, industry relevance, technical aspects of the language, and the availability of useful tools and materials. The process has become increasingly cumbersome as the number of languages has grown [1].

CS1 remains a subject of great interest in the same way it has been for over half a century. We usually consider it to be one of the foundations of an education in computer science. Since it is usually where students begin the study of the discipline, CS1 courses are critically important. While we concern ourselves with different approaches to teaching beginners on how to develop an algorithm, and turn it into a program that works correctly and is easy to understand and modify, we cannot ignore the role that our choice of programming languages plays.

Many programming languages have been used in CS1 courses; several of them became popular, yet were eventually replaced by other languages. FORTRAN was a choice to be used as an introductory language, but the difficulties that students had with industry-standard compilers led to the development of WATFOR by Shantz et al. [2]. The advent of structured programming led to the use of PL/I as an instructional language; Holt considered the use of PL/I a terrible way of teaching introductory programming [3].

Despite the development of the PL/C compiler by Conway and Wilcox [4], PL/I died out because of the difficulties students had in mastering it, and its compilers were too complex to run on the early generations of personal computers.

Niklaus Wirth derived the Pascal programming language from Algol and it was intended for use in teaching good programming habits. However, it had shortcomings of its own. Kernighan described Pascal as "meant for learning", but unsuitable for serious programming [5], an assessment with which Haberman concurred [6]. In 1996, Brilliant and Wiseman described Pascal as dated, an assessment with which many educators agreed [7].

The question that computer science educators have tried to answer since the early 1990s, is whether there exists an ideal language to use when teaching college freshmen how to program. Johnson considered C too complex a language for beginning programmers [8]. Many college programs switched to using C++ in their introductory programming course. The College Board decided to embrace C++ for the Advanced Placement Computer Science exams in 1994 [9]. The Advanced Placement exam subsequently switched to Java in 2003 [10], and so did many introductory programming courses. Java was widely considered an easier language to learn than C++ [11], [12]. While the TeachRacket (originally called TeachScheme) approach has been around for a decade [13], it is only used in

a relatively small number of colleges. More recently, Python has become popular. Mason and Cooper 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 [14].

The choice of a programming language to be used in an introductory programming course has been a "hot button" topic within the computer science and information systems communities. While the AP Computer Science curriculum continues to be based on Java, there are many programs that are questioning whether this is the language that they ought to be using. Some schools never used Java in their CS1 courses, while others have either made a change or are contemplating it.

This paper builds on the work started by Richard Reid in the early 1990s, which surveys institutions providing post-secondary education and attempts to determine what computer programming languages are in use for their first computer programming course. This year's survey results indicate clearly that Java and Python are currently the most common languages of choice for teaching CS1. A new element in the survey is that we have now started collecting data regarding the second programming course (CS2).

Leveraging the longevity of Reid list data set, the work presented in this paper provides a meaningful and substantial context for further research into the choice of programming languages used to teach CS1 and CS2 education. Our ultimate goal is to uncover the decision-making rationale for adoption of programming languages. This paper provides initial overview of what these languages currently are.

Specifically, we asked the following questions: What is the most common programming language of choice in post-secondary CS1 courses? What is the most common programming language of choice in post-secondary CS2 courses? And, finally: Is there a relationship between the languages used for CS1 courses and CS2 courses taught in post-secondary programs of study? If so, what is the relationship?

In this paper, we begin to answer these questions. Section II gives a brief overview of the history of the Reid List and its methodologies. Section III further delves into related literature. Section IV summarizes our research and previous work in the area. Section V discusses our current active research and findings, and provides answers to our research questions. Section VI addresses our conclusions and future work.

II. The Reid List

Richard Reid was a Professor of 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 Reid List was updated when 10% or more of the included colleges changed the programming language of instruction [15]. This resulted in a new list being released approximately twice a year until Reid retired in 1999. Frances Van Scoy, an Associate Professor of Computer Science at West Virginia University, updated the list from 2001 until 2006 [16]].

TABLE I
Regional distribution of Reid List Schools (2019)

Region	Schools ($n = 402$)
New England	39
Mid Atlantic (incl. DC)	83
Southeast	94
Kentucky and West Virginia	8
Midwest	95
Southwest	40
Northwest	23
Alaska and Hawaii	2
Non-US	18

TABLE II
Breakdown by Highest Degree Offered in Computing

Highest Degree Awarded in Computing	2011	2019
Associate's	9	9
Bachelor's	128	117
Master's	109	104
Doctoral	157	166

The last list compiled by Reid included 527 colleges and universities, of which 352 were located in the United States, and 25 were located in Canada. With Van Scoy taking over the list, the number of colleges and universities outside the US fell to 11, with only 3 from Canada. The schools appearing on lists compiled in 2011, 2015, and 2019 were based on the 24th Reid List, which Van Scoy released in 2004.

The current Reid List (compiled in 2019) includes 402 colleges and universities. The geographic distribution of the Reid List schools over the past two decades is shown in Table I, with 384 of the colleges representing the District of Columbia and 49 states.

Wyoming is the only state without representation despite several attempts to obtain data from schools in the state.

While there is a reasonable geographic balance, the mid-Atlantic and southwestern states are overrepresented by the relatively 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 several Ivy League colleges and MIT.

Table II shows the breakdown by the highest degree program offered in computing for the surveys in 2011 and 2019. There is an almost even distribution between undergraduate, master's- and doctorate-granting departments in 2001; however, only nine of the programs were in community colleges, which are significantly underrepresented in the sample. There was one vocational/technical school on the list. It was removed because the school no longer offered a computing program. Five other schools are also no longer included for the same reason. Three colleges have been removed after they closed.

III. Literature Review

The earliest peer-reviewed and published survey of programming language choice was by Levy in 1996 [17], who

noted that there were 151 schools on the 11th Reid List from 1994 and that of the remaining schools, C and Ada were the most common language.

Levy also noted the lack of consensus on what language, if any, was the best language for teaching beginners how to program.

Several studies were conducted in Australia and New Zealand in 2001 [18], 2003 [19], 2010 [20] , 2013 [14], and 2016 [21]. They ranked the popularity of programming languages used in introductory programming courses weighted by student enrollment. The studies show that in 2001, the most popular languages were Java, Visual Basic, and C++; over the next fifteen years, this changed significantly and in 2016 the most popular programming languages were Python, Java, and C.

Similarly, in 2001, the most common criteria for choosing a language for CS1 was its industry relevance and marketability, which was twice as common as pedagogic benefits (39 universities to 19). In 2016, pedagogic benefits were slightly more common than industry relevance (81% compared to 78%).

There have been three other surveys looking at the popularity of programming languages at European universities. Murphy et al. surveyed universities in the United Kingdom in 2016 [22] and found that the most commonly used language was Java, followed by the C family of languages (C, C++ and C#) with Python a distant third.

The most common reason for the choice was industry relevance effectively tied with object-orientation and availability/cost, and only after these was pedagogic benefits a consideration. Becker [23] found similar results, where 49% of courses used Java, 28% used Python and 18% used JavaScript. Industry relevance, availability/cost and pedagogic benefit were the most common reasons in that order. Avouris [24] compiled data from 121 courses at 44 higher education institutions in Greece, and found that 73% of these courses used C and 9% used Python.

Davies et al. surveyed 371 computer science programs in 2010 to determine the programming languages used in CS0, CS1 and CS2 courses [25]. They found that 179 programs used Java in their CS1 course, 107 used C++, 48 used Python and 27 used C. In their CS2 courses, 207 used Java, 134 used C++, 18 used C and 12 used Python. While 75% used the object-oriented paradigm in CS1, 92% used objects in their CS2 courses. Nevins surveyed the computer science programs in the California Community college system and found that 82% used a single language, with 48% using C++ and 31% using Java [26].

In a 2019 publication, Silva et. al [27] investigated the use of programming languages in CS1 and CS2 education by comparing academic publications mentioning specific programming languages. Their findings were that 41.3% of publications related to CS1 education discussed using Java, and 39.1% discussed the use of Python. For CS2 education, they found that 34.8% of publications were Java-centric.

IV. Methodology

Updates to the Reid List were compiled in 2011, 2015, and 2019, with a few of the programs listed being updated in July 2020. The colleges and universities included in the 2011 survey for the twenty-sixth Reid List were taken from the twenty-fourth Reid List, published online in 2004; many of these did not appear on the twenty-fifth list from 2006, which only listed 153 schools.

The methodology used to collect the data used to compile the list has not changed significantly, despite the work having been performed by different authors.

The authors recognize that there are challenges in using the existing Reid list as a basis for data collection. Specifically, the list is US-centric and it under-represents community colleges. Furthermore, the scope of the study specifically excludes programming courses that are taught as part of an interdisciplinary curriculum. As such, the results found in this study are self-contained. However, given the historic availability of the Reid list data, understanding the evolution of language adoption can help researchers understand the meta-issues that impact language adoption and facilitate consideration in a longitudinal study. By studying a large number of schools over a long period of time, we can then further identify patterns that may be helpful for future trends.

Figure 1 illustrates the process followed to locate data for each school. The methodology was designed to obtain objective data points in a repeatable way while minimizing the inconvenience to individual faculty members.

Using previous lists as a starting point, step 1 (Identify Program) identified what program (if any) would be evaluated for the current revision.

The school's public web sites were reviewed to determine whether the school offered both Bachelor of Arts (BA) and Bachelor of Science (BS) programs. If so, the requirements for the BS were used. If the highest degree offered was an Associate's degree, as is common at community colleges, the requirements for the Associate's degree in Computer Science were examined. For schools that offered both Computer Science and Information systems programs, Computer Science was preferred over Information Systems. Computer Engineering, Information Technology, and Data Science or similarly named programs were not included in this review. If no match was found, the school was removed from the list, and no data was collected.

The second stage in data collection was to review degree requirements for the selected programs. As before, websites containing programs of study were analyzed, and the requirements for a degree program in Computing were examined to determine the first required programming course. If program requirements could be found with certainty and confidence, it was included in this year's revision. In all other cases, it was omitted from data collection. Course descriptions were examined to see if explicit references to the programming language of instruction were included. If so, the course description was deemed authoritative, and the data was recorded.

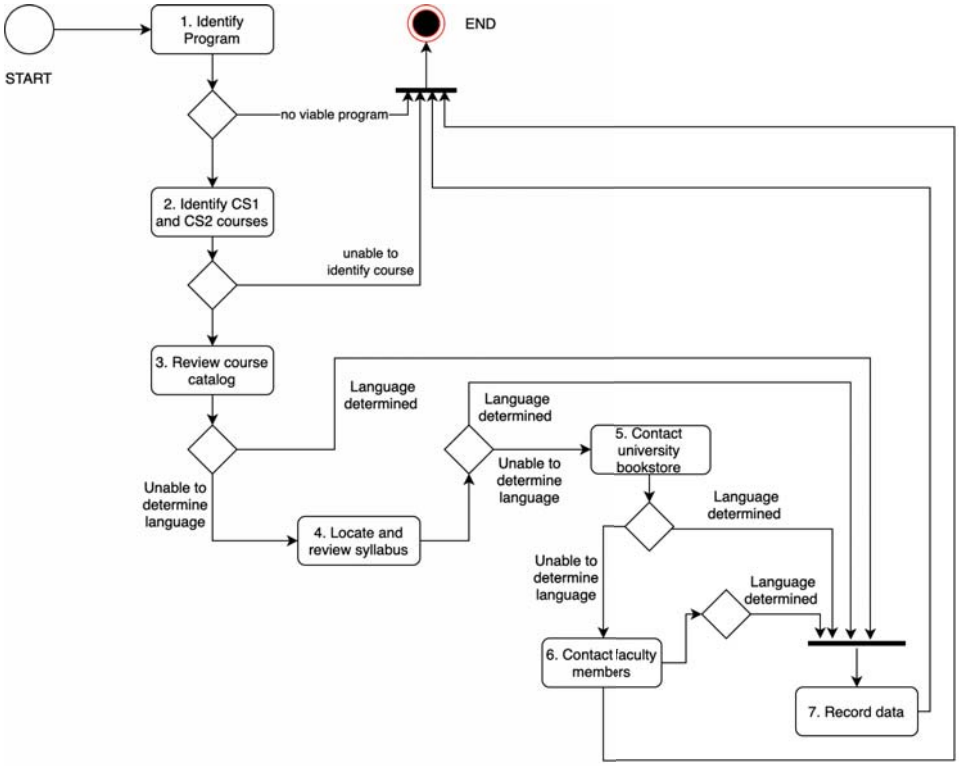

Fig. 1. Reid List Data Collection Methodology

However, as expected, most programs did not specify the language used in their CS1 and/or CS2 courses. Our next step was to determine if a current syllabus for the course was available online. Only the syllabus for the current academic year was accepted. Only if a current syllabus was identified and contained specific references to the programming languages used for the course, we included the data point. If multiple current syllabi were located, they were cross-referenced for consistency. In this, we follow the same approach as Becker and Fitzpatrick [28].

If a current syllabus could not be obtained, or if no language reference could be found in one, the web site of the bookstore that is affiliated with the school's web site was checked for a textbook adoption; in some cases, the bookstore was contacted in an attempt to get this information.

As before, only textbook listings for the current term were considered, and results were only accepted if the choice of textbook(s) clearly identified a specific programming language.

Lastly, if these steps did not yield results, then department members were contacted to obtain this information. Emails to faculty members became standard procedure in the 2015 and 2019 updates because they frequently provided insightful anecdotal information.

By following the organically evolved Reid List, the current data set primarily includes institutions that responded to the original requests for information made by professor Reid. As such, the sample might not be fully representative of the current composition of higher education. In future work, we aim to evolve the data set by using standardized data collection and sampling techniques to normalize the data set.

TABLE III
THE 28TH REID LIST OF CS1 LANGUAGES (2019)

Language	Number of schools ($n = 409$)
Java	163
Python	111
C++	83
C	17
Racket/Scheme	6
JavaScript	2
Scala	2
C#	1
Haskell	1
Visual Basic	1
Logic	1
2 Languages	5
Choice of Language	16

V. RESULTS

The 28th Reid List (2019) appears in Table III. It immediately illustrates that Java remains the predominant language for CS1 courses taught at Reid List schools, with Python and C++ in second and third place respectively. C and Scheme/Racket (the Teach Racket group's dialect of Scheme) follow them distantly. JavaScript, Scala, C#, Haskell, SNAP and Visual Basic are each used in only one or two schools. Lastly, there was one program that did not use a programming language but

TABLE IV
A COMPARISON OF PROGRAMMING LANGUAGE USAGE IN CS1 COURSES IN THE 3 MOST RECENT SURVEYS

Language	2011	2015	2019
C	18	21	17
C++	85	76	69
Java	193	179	159
Python	41	75	107
Racket/Scheme	11	9	5
Other Languages	16	11	11
2 or More Languages	12	8	4
Choice of 2 or more	8	8	15

TABLE V
THE POPULARITY OF PROGRAMMING LANGUAGE IN CS2

Language	Number of schools ($n = 353$)
Java	196
C++	90
Python	22
C	15
C#	4
Scala	3
Alice	1
Haskell	1
2 or more Languages	11
Choice of 2 or more Language	10

taught the concepts of programming using pseudocode and the programming constructs using basic logic.

Five schools used two languages in the same courses, such as Python with C++ or Java. These schools are in addition to schools covering C and C++ in the same introductory course; these four schools are counted together with the courses taught in C++, since it is reasonable to assume that mastery of C++ was the main learning goal for the course, with mastery of C being an additional positive outcome.

Sixteen schools had different course sections or most commonly, different introductory courses in different languages. In the latter case, these different CS1 courses had different themes or were intended for different audiences. It was not unusual for there to be as many as four different courses serving as a first programming course for computer science majors.

Table IV shows the changes in language popularity between the 26th (2011), the 27th (2015), and the 28th (2019) Reid Lists. The only schools included in this analysis were those for which there was data for all three lists. The most obvious changes are in the loss in popularity of Java and the gain by Python. C++ also became less popular as a CS1-language, although far less so than Java. The increased popularity of Python seems to be almost fully at the expense of Java and C++. The other languages were each used at three or fewer colleges and are not listed separately.

Schools employing more than one language for CS1 instruction have remained relatively rare, with the use of multiple languages in more courses decreasing and the number using different languages in different courses or sections increasing.

TABLE VI
COMBINATIONS OF LANGUAGES USED IN THE CS1–CS2 SEQUENCE ($n = 312$)

CS1	CS2	Amount
C	C++	7
	C	4
	Java	5
C++	C++	51
	Java	5
Java	Java	122
	C++	11
	C	4
	C#	2
	Python	1
Python	Java	54
	Python	18
	C++	13
	C	1
	C#	1
	Haskell	1

The languages used by Reid List colleges in their CS2 courses appear in Table V. Java is the most popular by a sizable margin with 196 schools using it to teach CS2 courses (55.5%).

C++ was used by 90 schools to teach CS2, followed by Python and C, which are used in 22 and 15 schools respectively. C#, Scala, Alice, and Haskell accounted for the languages which collectively were used in nine Reid List schools. Eleven schools used multiple languages in their CS2 courses and another ten offered multiple courses or sections using different languages.

The languages used by the various colleges in the CS1-CS2 sequence is shown in Table VI. The most commonly observed sequence consists of schools starting with Java for CS1 and continuing its use in CS2.

Schools starting with Python for CS1 and who switch to Java to teach CS2 follow in second place. Schools that opt to adopt C++ for CS1 instruction commonly continue to use it for CS2. These three combinations account for 73% of schools included in our data set.

Similarly, most of the schools using Python in their CS1 course used Java or C++ for CS2, with nearly all remaining schools continuing with Python into CS2. The schools using C in CS1 split almost evenly between C++ and Java, with the remainder continuing in C.

VI. CONCLUSIONS

In this paper, we continued the work initiated by Richard Reid in the early 1990s. Reid's lists, later continued by Van Scoy and then by Siegfried et. al. provides a longitudinal overview of the programming language of choice for CS1 classes taught at post-secondary institutions. This edition shows clearly that the adoption of Python continues and predominantly takes place at the expense of Java and C++.

Based on our survey, we can now answer our first question: What is the most common programming language of choice

in post-secondary CS1 courses? The answer is that 88% of the Reid List schools use one of only four languages: Java, Python, C++, C in their CS1 courses.

In this edition of the survey, we extended our data collection efforts by also capturing the language-of-choice for the second programming language course. In this case, the field is much narrower than in CS1: Java is the predominant language for CS2 courses. This finding provides an answer to our second question: What is the most common programming language of choice in post-secondary CS2 courses? Java's dominance is indisputable in CS2 courses. C++ takes second place, having been adopted by approximately 22% of programs.

Our third question was: Is there a relationship between the languages used for CS1 courses and CS2 courses taught in post-secondary programs of study? If so, what is the relationship?

We found that 195 schools (54%) chose the same programming language for CS1 and CS2. However, that also means that 46% of the colleges prefer a different language for the two courses. Of this group, the most common switch is from Python in CS1 to Java in CS2. Almost all schools indicated that they introduce additional languages later in their curricula.

Future work for this research includes further exploring this data set to better understand the observed trends and differences. Additionally, we plan to look at including further, publicly available data to help understand the student's experience in programming languages in their first two years of computing study.

ACKNOWLEDGEMENTS

This work has been through an IRB process and was found exempt. A data repository has been created with first person interviews either conducted via phone or email of representatives from the participating universities for reproducibility purposes.

REFERENCES

[1] A. Pears, S. Seidman, L. M. L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson, "A Survey of Literature on the Teaching of Introductory Programming," in *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*, ser. ITiCSE-WGR '07. New York, NY, USA: ACM, 2007, p. 204–223. [Online]. Available: https://doi.org/10.1145/1345443.1345441

[2] P. W. Shantz, R. A. German, J. G. Mitchell, R. S. K. Shirley, and C. R. Zernike, "WATFOR – The University of Waterloo FORTRAN IV Compiler," *Communications of the ACM*, vol. 10, no. 1, January 1967.

[3] R. C. Holt, "Teaching the Fatal Disease or Introductory Computer Programming Using PL/I," *ACM SIGPLAN Notices*, vol. 8, no. 5, pp. 8–23, May 1973.

[4] R. W. Conway and T. R. Wilcox, "Design and Implementation of a Diagnostic Compiler for PL/I," *Communications of the ACM*, vol. 16, no. 3, pp. 169–179, March 1973.

[5] B. W. Kernighan, "Why Pascal Is Not My Favorite Programming Language," AT&T Bell Laboratories, Computing Science Technical Report 100, April 1981.

[6] A. N. Haberman, "Critical Comments on the Programming Language Pascal," *Acta Informatica*, vol. 3, pp. 47–57, 1973.

[7] S. S. Brilliant and T. Wiseman, "The first programming paradigm and language dilemma," *ACM SIGCSE Bulletin*, vol. 28, no. 1, pp. 338–342, 1996.

[8] L. F. Johnson, "C In The First Course Considered Harmful," *Communications of the ACM*, vol. 38, no. 5, pp. 99–101, May 1995.

[9] R. Cartwight, R. Kick, C. Horstmann, F. Trees, G. Chapman, D. Gries, H. Walkers, U. Wolz, and O. Astrachan, "Recommendations for changes in advanced placement computer science (panel session)," in *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education (SIGCSE '00)*. ACM, 2000, p. 416.

[10] C. L. Fletcher and J. B. Owen, "WeTeach_CS Support for AP Computer Science; "A", Test and CS Principles," in *Proceedings of the 2018 Texas Computer Education Association (TCEA 2018)*, February 2018.

[11] S. Hadjerroult, "Java As First Programming Language: A Critical Evaluation," *ACM SIGCSE Bulletin*, vol. 30, no. 2, pp. 43–47, June 1998.

[12] M. Madden and D. Chambers, "Evaluation of Student Attitudes to Learning the Java Language," in *Proceedings of the Conference on the Principles and Practice of Programming in Java*, June 2002, pp. 125–130.

[13] M. Felleisen, R. B. Findler, M. Flatt, and S. Krishnamurthi, "The TeachScheme! Project: Computing and Programming for Every Student." *Computer Science Education*, vol. 14, no. 1, pp. 55–57, 2004.

[14] R. Mason and G. Cooper, "Introductory Programming Courses in Australia and New Zealand in 2013 - trends and reasons," in *Proceedings of the Sixteenth Australasian Computing Education Conference (ACE2014)*, 2014, pp. 139–147.

[15] R. J. Reid, "First Course Language for Computer Science Majors," West Virginia University, Tech. Rep., 2011. [Online]. Available: http://www.csee.wvu.edu/~vanscoy/REID06.html

[16] F. V. Scoy, "Reid List 25," West Virginia University, Tech. Rep., 2011. [Online]. Available: {http://groups.google.com/group/comp.edu/browse_thread/thread/4f00b5f437ce261a/3267514419052033?q=Reid+List#3267514419052033}

[17] S. P. Levy, "Computer Language Usage in CS1: Survey Results," *3C ON-LINE*, vol. 3, no. 1, pp. 13–17, Jan. 1996. [Online]. Available: https://doi.org/10.1145/218806.218812

[18] M. de Raadt, R. Watson, and M. Toleman, "Language trends in introductory programming courses," in *Informing Science + Information Technology Education Joint Conference (InSITE 2002)*, June 2002.

[19] ——, "Introductory programming: what's happening today and will there be any students to teach tomorrow?" in *Proceedings of the 6th Australasian Computing Education Conference (ACE 2004)*, February 2004, pp. 18–24.

[20] R. Mason, G. Cooper, and M. de Raadt, "Trends in Introductory Programming Courses in Australian Universities – Languages, Environments and Pedagogy," in *Proceedings of the Fourteenth Australasian Computing Education Conference (ACE2012)*, January 2012, pp. 33–42.

[21] R. Mason and Simon, "Introductory programming courses in australasia in 2016," in *Proceedings of the Nineteenth Australasian Computing Education Conference*. ACM, 2017, p. 81–89. [Online]. Available: https://doi.org/10.1145/3013499.3013512

[22] E. Murphy, T. Crick, and J. H. Davenport, "An Analysis of Introductory Programming Courses at UK Universities," *The Art, Science, and Engineering of Programming*, vol. 1, no. 2, 2017.

[23] B. A. Becker, "A Survey of Introductory Programming Courses in Ireland," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19)*, July 2019, pp. 58–64.

[24] N. Avouris, "Introduction to computing: a survey of courses in Greek higher education institutions," in *Proceedings of the 22nd Pan-Hellenic Conference on Informatics (PCI '18*. ACM, 2018, pp. 64–69.

[25] S. Davies, J. A. Polack-Wahl, and K. Anewalt, "A snapshot of current practices in teaching the introductory programming sequence," in *Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11)*, March 2011, pp. 625–630.

[26] D. Nevins, "CS 1 language adoption at California Community Colleges: 2012," *ACM Inroads*, vol. 4, no. 1, pp. 34–37, March 2013.

[27] D. B. Silva, R. d. L. Aguiar, D. S. Dvconlo, and C. N. Silla, "Recent studies about teaching algorithms (cs1) and data structures (cs2) for computer science students," in *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1–8.

[28] B. A. Becker and T. Fitzpatrick, "What do cs1 syllabi reveal about our expectations of introductory programming students?" in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. New York, NY, USA: ACM, 2019, p. 1011–1017. [Online]. Available: https://doi.org/10.1145/3287324.3287485