A Global Survey of Introductory Programming Courses

Raina Mason
Southern Cross University
Gold Coast, Australia
raina.mason@scu.edu.au

Simon
Unaffiliated
Wadalba, Australia
simon.unshod@gmail.com

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

Tom Crick
Swansea University
Swansea, United Kingdom
thomas.crick@swansea.ac.uk

James H. Davenport
University of Bath
Bath, United Kingdom
j.h.davenport@bath.ac.uk

ABSTRACT

We present results of an in-depth survey of nearly 100 introductory programming (CS1) instructors in 18 countries spanning six continents. Although CS1 is well studied, relatively few broadly-scoped studies have been conducted, and none prior have exceeded regional scale. In addition, CS1 is a notoriously fickle and often changing course, and many might find it beneficial to know what other instructors are doing across the globe; perhaps more so as we continue to understand the impact of the COVID-19 pandemic on computing education and as the effects of Generative AI take hold. Expanding upon several surveys conducted in Australasia, the UK, and Ireland, this survey facilitates a direct comparison of global trends in CS1. The survey goes beyond environmental factors such as languages used, and examines why CS1 instructors teach what they do, in the ways they do. In total the survey spans 84 institutions and 91 courses in which a total of over 40,000 students are enrolled.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; **Computer science education**; **CS1**.

KEYWORDS

CS1; CS 1; CS-1; COVID-19; Global; Instructors; Introductory Programming; Novice Programmers; Programming Languages; Teaching Languages; Survey

ACM Reference Format:
Raina Mason, Simon, Brett A. Becker, Tom Crick, and James H. Davenport. 2024. A Global Survey of Introductory Programming Courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626252.3630761

1 INTRODUCTION

The introductory programming sequence is the foundation of most computing degrees and has been studied extensively [7, 35]. The many variants of this course, often called CS1, almost always focus on learning the basics of programming and most often introduce students to one or more specific programming languages [50]. The term 'CS1' was introduced in *Curriculum '78: Recommendations for the Undergraduate Program in Computer Science (A Report of the ACM Curriculum Committee on Computer Science)* [1]. Curriculum '78 specified eight core courses, with short names CS1–CS8. CS1 and CS2 remain in common use as shorthand for the first programming course and the first data structures & algorithms course, respectively. Hertz [32] notes that "while there is wide agreement on the connotation of CS1 and CS2, there is little agreement as to the denotation of these terms". A recent random sampling of 500 US undergraduate degree programs in computer science found that 100% of both ABET-accredited and non-accredited programs had a CS1 course, with perfect inter-rater reliability (using three techniques) [8]. Thus it is safe to say that CS1 courses are ubiquitous in the CS curriculum. It is also well known that students face many struggles in these courses [46]. This gives them a reputation of being "hard to learn" [39] although this notion has been disputed and has led to an unhealthy reputation [3, 55]. It has also been argued that educators' expectations of CS1 students are unrealistic [34], and some research has explored what exactly is expected of CS1 students by analyzing hundreds of syllabi [6]. CS1 has also changed over the decades, and is continuing to evolve [21], in terms of approach and languages [7], changes that have been reflected in model curricula [44, 45, 58], country-specific national policy interventions [9, 22, 63], as well as broader critical analysis of the foundational programming knowledge, skills and experience required for developing sustainable software and systems [61, 62], including an increasing focus on the craft aspect of learning programming (e.g. "software carpentry" and "codemanship" [13, 14]). The increased focus on CS1 might also have been driven in part by extensive developments in K-12 computing education across various countries and jurisdictions – with significant focus on coding and programming – including new statutory national curricula and pre-university level qualifications (e.g. [10, 27, 41, 49, 51]).

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

ACM ISBN 979-8-4007-0423-9/24/03.
https://doi.org/10.1145/3626252.3630761

2 RELATED WORK

CS1 research is often considered particularly challenging, one reason being that it sits in an ecosystem that is undergoing constant change. A very non-exhaustive list of examples include:

- Programming languages change and evolve, as do first programming environments;

- The use of programming languages in the outside world changes, for example the increased visibility of data science and machine learning, which are often programmed in Python in modern times;
- Students' backgrounds change: several educational systems now insist (with varying degrees of success) that programming be taught in schools, which changes what might be expected of CS1 students in universities;
- Curiously, parents (and students) have views about the first programming language in a way that they do not seem to have in other subjects (a professor of Russian would be astounded if a parent asked "what is the first author studied?").

Although much research has been conducted on CS1 [7, 35] until recent years there were relatively few broadly scoped surveys, reviews, and meta-analyses – although two early papers were presented at the ACM SIGCSE Technical Symposium. In 2002 Vasiga presented historical data on introductory programming papers [60], and in 2004 Valentine conducted a review of introductory programming papers, organizing them into somewhat idiosyncratic groups based either on their approach or on their content: experimental, 'John Henry', 'Marco Polo', 'Nifty', philosophy, and tools [59].

A 2018 ITiCSE working group led by Luxton-Reilly and Simon produced a 52-page literature review of introductory programming research, processing thousands of papers and citing 761. Other similar and recent work processed 777 SIGCSE Technical Symposium papers focusing on CS1, identifying several trends in the first 50 years of research on these courses [7].

A longitudinal study of programming languages and environments in CS1 courses in Australasia was conducted by de Raadt et al. in 2001 [23] and continued by various authors in 2003 [24], 2010 [37], and 2013 [36]. This study received scant attention outside Australasia, with one paper in 2014 reporting on a short survey in the USA and noting that "we need to get a better picture of how programming languages are used in academia internationally" [57].

Following a 2016 iteration of the Australasian study [38], the survey was taken up by Murphy, Crick, and Davenport in the UK [43] and by Becker in Ireland [2], with both studies using the same instrument provided by the original authors. For the work reported in this paper, several of these authors joined forces, using the same instrument to survey a global audience with the same questions and to look for commonalities and differences.

2.1 Research Questions

Our research questions are closely related to the survey questions, as taken from the regional studies already mentioned.

RQ1 What programming languages are used in introductory programming courses, with what relative frequencies, and why are they chosen?
RQ2 What programming environments are used with the selected programming languages and why are they chosen?
RQ3 What paradigm is being taught – regardless of what paradigm(s) the language may support?
RQ4 How prevalent is online delivery of courses?
RQ5 What resources are commonly provided to students?

3 METHOD AND CONTEXT

All project members received appropriate IRB/HREC/ethics approval from their institutions prior to collecting data. In geographic terms, we set out to conduct a survey based on the earlier ones [2, 38, 43], but with global reach. In temporal terms, the context of our survey has two features of particular interest, neither of which was part of our research design. We discuss these in Sections 3.1 and 3.2

Invitations to take part in the survey were circulated on a number of mailing lists: principally `SIGCSE-members`, but also lists of computing educators in various countries and regions that might not be directly reached by `SIGCSE-members`, for language or other reasons. Those who were teaching one or more introductory programming courses were invited to complete the survey for one or more of those courses; others were asked to forward the email, if practical, to somebody who was teaching such a course at their institution or in their region. The survey questions can be accessed at dx.doi.org/10.5281/zenodo.10171742.

3.1 Impact of the COVID-19 Pandemic

The survey was conducted in late 2021 and early 2022. This led to the first of our unintended temporal features: many universities and colleges around the world [31, 40, 52, 65] had faced rapid transitions from face-to-face to 'emergency remote teaching' and then to more planned online learning, teaching, and assessment [17, 18, 64]. They were now contemplating when and how, and perhaps even whether, to return to face-to-face teaching [12, 67]. During our survey, some institutions would have still been teaching and assessing online only, while others had already returned to campus, wholly or partly, with widespread impacts on students [19, 20, 42], faculty [15, 16], and institutions [33, 66]. ITiCSE working groups in 2021 and 2022 have looked at these issues specifically for the discipline of computing [53, 54].

Addition to the Survey Instrument. The survey instrument was the same as used in three distinct regional surveys [2, 38, 43], but with one additional question explicitly referring to the COVID-19 pandemic: "How do you think the COVID-19 pandemic has impacted on the teaching of introductory programming?"

3.2 Generative AI

The second unintended temporal feature of our work is that this is the last snapshot of introductory programming as it was before code-generating AI tools became prevalent. Conferences sponsored by or in cooperation with ACM SIGCSE[1] saw their first publications involving generative AI in early 2022. The majority of this work to date has focused on CS1, as noted recently at the SIGCSE Technical Symposium by Becker et al. [5]. Our snapshot is therefore important as generative AI is widely expected to significantly and permanently alter the landscape of CS1 [4, 11, 28] and beyond [29] – and indeed society more broadly [25, 26]. Evidence of the impact of Generative AI on CS1 is apparent in practice [48] and in the emergence of teaching materials such as a new textbook that advocates learning AI-assisted programming in CS1 courses (using GitHub Copilot and ChatGPT) from day one [47].

[1] SIGCSE sponsored conferences are the ACM SIGCSE Technical Symposium, ACM ITiCSE, ACM ICER, and ACM CompEd. SIGCSE in-cooperation conferences can be found at sigcse.org/events/incoop.html.

3.3 Demographics

There were a total of 153 respondents to the survey, but many of these abandoned the survey after the opening questions about country and institution. Respondents were next asked to identify the course about which they were responding, so that we could identify and remove duplicates. When the incomplete responses and duplicates were removed, a total of 91 responses were identified that had at least the programming language question completed. These responses became our data set.

3.3.1 Countries. The responses were dominated by those from the USA, followed by Australia, Ireland, England, and Canada. The number of responses per country can be seen in Table 1. The 'Other' entry represents one response each from Brazil, Denmark, Finland, Italy, Kenya, Portugal, Scotland, and Slovenia.

Table 1: Number of respondent courses per country

Country	Respondents	Percentage
USA	48	54%
Australia	8	8%
Ireland	6	7%
England	5	6%
Canada	4	4%
Germany	3	3%
New Zealand	3	3%
India	2	2%
Jordan	2	2%
Poland	2	2%
Other	8	9%

For further analysis (as in Figure 1), these responses were grouped into 'Americas' (Canada, Brazil, USA: 53 responses), 'British Isles' (UK and Ireland: 12), 'Europe' (10), 'Oceania' (Australia and New Zealand: 11), and 'Other' (5).

3.3.2 Class Sizes. Respondents were asked "approximately how many students are undertaking this course this year (across all cohorts, locations, and modes)?" The 89 responses to this question ranged from 20 to 6,000 students, with a total of 40,362. Table 2 shows the number of courses for each range of class sizes.

Table 2: Number of students by course

Course size	0-50	50-99	100-199	200-499	500-999	1k-<2k	2k-<5k	5k+
Courses	13	14	21	17	14	7	2	1

3.3.3 Years of Experience of Respondents. Guzdial remarked that "On many campuses, teaching introductory courses typically falls to less-experienced instructors" and that "this can have negative consequences" [30]. In contrast 66% of the instructors answering this question have at least 10 years of experience (Table 3). This likely indicates that CS1 instructor experience is appreciated. There is the possibility of survey bias, however. For instance it is possible that less experienced teachers did not complete the survey with the same frequency as those with more experience.

Table 3: Experience teaching programming

Years	0-2	2-5	5-10	10-20	20-30	>30
Respondents	4	6	17	20	20	14

4 RESULTS

Here we summarize our findings to each research question.

4.1 Programming Languages

RQ1 asked what programming languages are used in CS1 courses, with what relative frequencies, and why they are chosen.

4.1.1 Languages Being Used. Table 4 shows the number of institutions and courses using each language.

Table 4: Language use – institutions and courses

Language	Institutions	Courses
Python	39	40
Java	39	39
C++	12	12
C	9	10
JavaScript	3	3
Processing	2	2
Racket	2	2
Scala	2	2
Go*	1	1
Bash*	1	1

* Used only in combination with another language

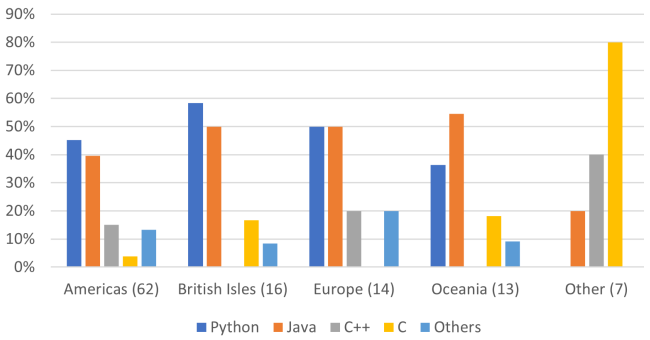

Figure 1: Percentage of language in each region

The total numbers of institutions and courses are greater than the number of respondents, as some courses/institutions use more than one language. Of those using a *single* language, 30 used Java, 29 used Python, six used C, four used C++, two each used Scala and Processing, and one used Racket. Seventeen courses used multiple languages. It makes very little difference whether we count these

languages fractionally or as complete counts – although the choice does affect the relative positions of Python and Java, vs C++ and C, as both pairs are quite close in prevalence. Go and Bash are used only in multi-language courses. While not surprising for Bash, it is perhaps more surprising for Go, which was used only in a course that also used C and Java. Racket occurred once by itself and once jointly with Java.

There are some variations by geographical region (see Figure 1), such as the relative prevalence of C and C++. However, these must be considered with caution. For example, the prevalence of C in 'Other' is remarkably high, but this whole group comprises just seven languages in five courses from India (2), Jordan (2), and Kenya (1), making it unwise to draw conclusions from these results.

4.1.2 Reasons for Choice of Language. As with the most recent Australasian survey [56], the most common reason selected for choice of programming language was pedagogical benefits (Figure 2). In the UK survey [43], this was equal first with relevance to industry, and in the Irish survey [2] the most common reason was relevance to industry, which is ranked fourth in our survey. In view of the increasing industry use of Python in such areas as machine learning and data analysis, it is perhaps surprising that this reason ranks lower than platform independence and availability and cost to students. Several reasons for this may exist, such as a difference between a department's actual reasons and the ones it might give to students and parents.

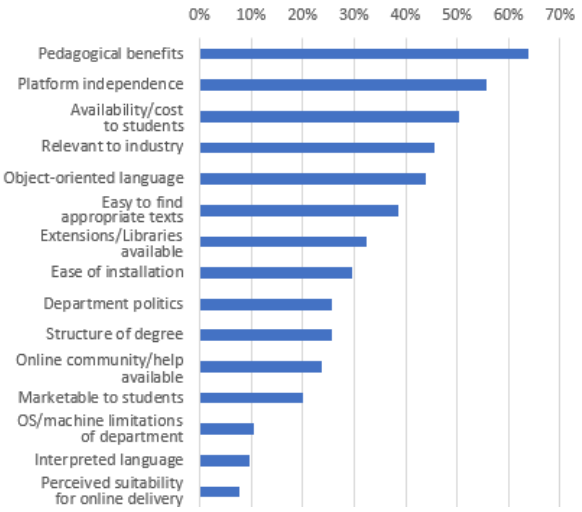

Figure 2: Reasons for choice of language

4.1.3 Reasons to Consider Changing Language. Respondents were asked what reasons might prompt them to consider changing the programming language they were using. Responses are shown in Figure 3. Whilst pedagogical benefits is also the top reason to change, this is followed strongly by industry relevance, possibly indicating a tension between these reasons for choice.

4.2 Programming Environments

RQ2 asked what programming environments are used with the selected programming languages and why they are chosen.

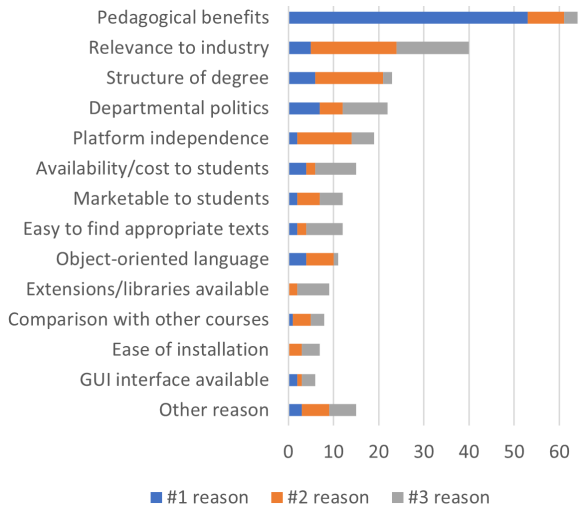

Figure 3: Reasons to consider change of language

4.2.1 Programming Environments Being Used. A quarter of the respondents reported not using a programming environment as such (Table 5), instead relying on text editors and command-line compilers. The table also shows a substantial number of courses using more than one environment, with one using five different environments. Of the environments that are being used, Figure 4 shows a remarkably uniform spread, with just a few, led by Eclipse and Visual Studio, standing clear of the crowd.

Table 5: Number of environments used in a course

Environments	0	1	2	3	4	5
# Courses	22	37	14	5	2	1

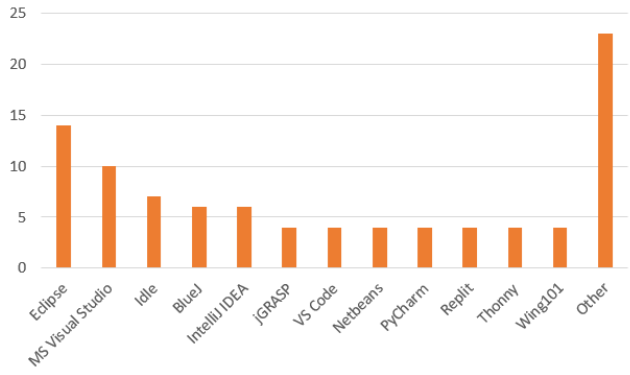

Figure 4: Programming environments used

4.2.2 Reasons for Choice of Environment. In contrast with the reasons for choice of programming language, pedagogical benefits ranks only seventh among the reasons for choice of programming

environment. However, it is clear from Figure 5 that the higher-ranked reasons are all student-centric. Asked whether the same environment is used in other courses in the degree, around two-thirds of respondents said that it was.

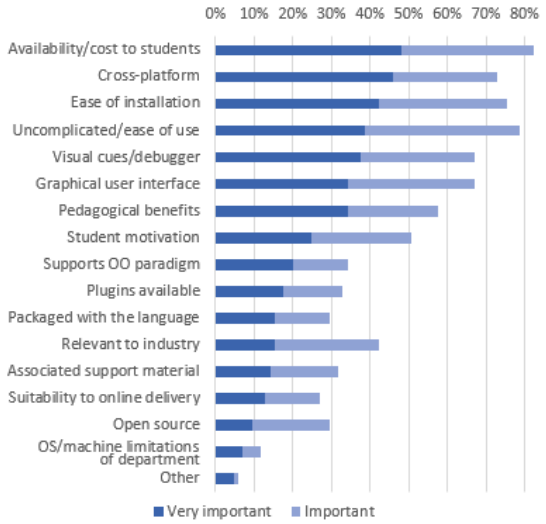

Figure 5: Reason for choice of environment

4.3 Paradigm Taught

RQ3 asked what paradigm is taught – regardless of what paradigm(s) the language may support. Four options were given – procedural, object-oriented, functional, and logical. Table 6 shows that 43 courses taught procedural followed by 32 object-oriented, with only a handful taught using functional (3) or other (4).

Table 6: Paradigm taught

Paradigm	procedural	OO	functional	logical	other
# Courses	43	32	3	0	4

This result contrasts with the 2016 UK survey [43], where object-oriented was a clear leader. In the 2017 Australasian survey, the procedural paradigm had a clear majority [38]. While there are similarities between the latest Australasian data and this survey, note that this data may not be reliable: Mason and Simon [38], noted that some respondents claimed to teach functional programming using C, prompting them to wonder whether they and the respondents had the same understanding of functional programming.

4.4 Online Offerings

RQ4 asked about the prevalence of completely online delivery of courses. When asked "Do you have options for your course where students are not required to attend any regular lectures, workshops, labs, or tutorials, but can complete their study online?" 38 respondents said yes and 43 said no.

Of the 38 'yes' respondents, 17 reported that this was the case prior to the COVID-19 pandemic and 20 said that it was not. When those who offered online options during COVID-19 were asked if they intend to keep offering online delivery post-pandemic, 20 answered affirmatively, six said no, and 12 were still considering whether they would continue online offerings (Table 7).

Table 7: Online delivery before, during, and after COVID-19, of respondents who were currently teaching online. One respondent did not answer the 'before' question. The 43 respondents who did not offer online options during COVID-19 are not reflected in this table (denoted by 'X').

online option	before	during	after (intended)
Yes	17	38	20
No	20	X	6
Maybe	N/A	N/A	12

These results show that of those who did offer an online option during COVID-19 (38) about half were already offering online options prior to the pandemic. It also shows that the number intending to continue online offerings after COVID-19 was only slightly greater than the number who had been using them before the pandemic (17 before, 20 intending to after). Interestingly, only six intended to not offer any online options after COVID-19, while nearly a third (12/33) of those that did offer online options during COVID-19 were undecided about continuing with online offerings. It is likely that the timing of our survey is the reason for this – the COVID-19 pandemic was still ongoing (to various degrees) when respondents completed the survey. Thus, the decision whether to continue offering online options after the worst of the COVID-19 restrictions, had likely not yet been resolved for these 12 respondents. It is also possible that the advent of Generative AI (discussed in Section 3.2) may have influenced survey responses for those responding late in the survey timeframe.

4.5 Resources Provided to Students

Respondents provided a wide range of resources to students (Figure 6). Over 50% of instructors provided lecture slides or notes, worked examples, discussion boards or forums, recorded lectures (possibly a COVID-19 artifact), assignment hints, and/or specified a textbook. Least popular were glossaries of keywords, topic summaries, and publisher-supplied slides or notes. Respondents also provided other resources, including automatic feedback systems, peer mentors, additional workshops for specific skills development, extensive screencasts (or other videos including animations or YouTube videos), a repository of code developed in class, and/or peer code presentations. We did not include 'office hours' as one of the choices provided in this question, but perhaps we should have done so, considering that not all courses may provide this resource.

5 LIMITATIONS

One significant limitation of this study is that responses were very US-dominated, and even beyond the US, most respondents were from English-speaking regions. We also did not ask any gender-related questions. A well-defined question would be "what is the gender of the respondent?"; however to ask the "gender of the

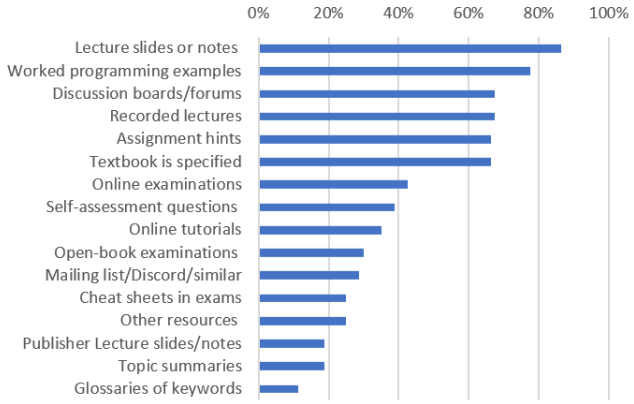

Figure 6: Percentage of courses using each resource.

instructor" is less well-defined. In a large class (and most CS1 classes are large: see Table 2) there will be many teaching assistants, and the student will have much more interaction with those than with the primary professor(s). So in terms of the influence of the gender of the instructors, the teaching assistant is possibly more important. Indeed, we asked nothing about teaching assistants, a vital part of large course delivery. This could well be a survey in its own right.

6 CONCLUSION AND FUTURE WORK

Ten years since Stefik and Hanenberg [57] noted the absence of an international picture of the programming languages used in introductory programming courses, we have provided that picture and more besides. We have established that Java and Python are in a clear joint lead; that the difference is not so stark among programming environments; and that most instructors consider pedagogical benefits to be a prime factor in the choice of programming language.

We have collected a wealth of data, and future work will include deeper analysis of that, including qualitative analysis of the free-response questions in the survey. It is instructive to compare our global (but US-dominated) findings with those of the earlier surveys in Australasia, the UK, and Ireland. For example, Figure 1 shows that Python just beats Java in the British Isles, whereas the 2016 UK survey [43, Figure 2] had Java with a 4:1 lead over Python. This could be a sign of the times.

Most importantly, this or a similar survey should be conducted a few years from now, when the world of (computing) education has hopefully come to terms with the end of the pandemic, and the impacts of Generative AI.

ACKNOWLEDGMENTS

We are grateful to the creators of the survey for permission to use it, and to the instructors who completed the survey.

REFERENCES

[1] Richard H. Austing, Bruce H. Barnes, Della T. Bonnette, Gerald L. Engel, et al. 1979. Curriculum '78: Recommendations for the Undergraduate Program in Computer Science – A Report of the ACM Curriculum Committee on Computer Science. *Communications of the ACM* 22, 3 (1979), 147–166. https://doi.org/10.1145/359080.359083

[2] Brett A. Becker. 2019. A Survey of Introductory Programming Courses in Ireland. In *Proc. of ITiCSE'19*. 58–64. https://doi.org/10.1145/3304221.3319752

[3] Brett A. Becker. 2021. What Does Saying That 'Programming is Hard' Really Say, and about Whom? *Communications of the ACM* 64, 8 (2021), 27–29. https://doi.org/10.1145/3469115

[4] Brett A. Becker, Michelle Craig, Hieke Keuning, Natalie Kiesler, et al. 2023. Generative AI in Introductory Programming. In *ACM/IEEE-CS/AAAI Compuitng Curricula 2023 Curricular Practices Volumes*. https://csed.acm.org/large-language-models-in-introductory-programming/

[5] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming Is Hard – Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proc. of SIGCSE'23*. 500–506. https://doi.org/10.1145/3545945.3569759

[6] Brett A. Becker and Thomas Fitzpatrick. 2019. What Do CS1 Syllabi Reveal About Our Expectations of Introductory Programming Students?. In *Proc. of SIGCSE'19*. 1011–1017. https://doi.org/10.1145/3287324.3287485

[7] Brett A. Becker and Keith Quille. 2019. 50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research. In *Proc. of SIGCSE'19*. 338–344. https://doi.org/10.1145/3287324.3287432

[8] Richard Blumenthal. 2022. Alignment among Normative, Prescriptive, and Descriptive Models of Computer Science Curriculum: The Effect of ABET Accreditation on CS Education. *ACM Transactions on Computer Science Education* 22, 3 (2022). https://doi.org/10.1145/3513141

[9] David Bowers, Alan Hayes, Tom Prickett, Tom Crick, et al. 2023. The Institute of Coding Accreditation Standard: Exploring the Use of a Professional Skills Framework to Address the UK Skills Gap. In *Proc. of UKICER'23*. https://doi.org/10.1145/3610969.3611121

[10] Neil C. C. Brown, Sue Sentance, Tom Crick, and Simon Humphreys. 2014. Restart: The Resurgence of Computer Science in UK Schools. *ACM Transactions on Computer Science Education* 14, 2 (2014), 1–22. https://doi.org/10.1145/2602484

[11] Peter Brusilovsky, Barbara J. Ericson, Cay S. Horstmann, Christian Servin, et al. 2023. Significant Trends in CS Educational Material: Current and Future. In *Proc. of SIGCSE'23*. https://doi.org/10.1145/3545947.3573353

[12] Tom Crick. 2021. COVID-19 and Digital Education: A Catalyst for Change? *ITNOW* 63, 1 (2021). https://doi.org/10.1093/itnow/bwab005

[13] Tom Crick, James H. Davenport, and Alan Hayes. 2015. Innovative Pedagogical Practices in the Craft of Computing. Advance HE. https://www.advance-he.ac.uk/knowledge-hub/innovative-pedagogical-practices-craft-computing.

[14] Tom Crick, James H. Davenport, Alan Hayes, and Tom Prickett. 2023. Teaching Programming Competencies: A Role for Craft Computing?. In *Proc. of UKICER'23*. https://doi.org/10.1145/3610969.3611140

[15] Tom Crick, Cathryn Knight, and Richard Watermeyer. 2022. Measuring the Impact of COVID-19 on the Health and Wellbeing of Computer Science practitioners. In *Proc. of SIGCSE'22*. https://doi.org/10.1145/3478432.3499129

[16] Tom Crick, Cathryn Knight, and Richard Watermeyer. 2022. Reflections on a Global Pandemic: Capturing the Impact of COVID-19 on the UK Computer Science Education Community. In *Proc. UKICER'22*. https://doi.org/10.1145/3555009.3555027

[17] Tom Crick, Cathryn Knight, Richard Watermeyer, and Janet Goodall. 2020. The Impact of COVID-19 and "Emergency Remote Teaching" on the UK Computer Science Education Community. In *Proc. of UKICER'20*. https://doi.org/10.1145/3416465.3416472

[18] Tom Crick, Cathryn Knight, Richard Watermeyer, and Janet Goodall. 2021. The International Impact of COVID-19 and "Emergency Remote Teaching" on Computer Science Education Practitioners. In *Proc. of EDUCON'21*. 1048–1055. https://doi.org/10.1109/EDUCON46332.2021.9453846

[19] Tom Crick, Tom Prickett, and Jill Bradnum. 2022. Exploring Learner Resilience and Performance of First-Year Computer Science Undergraduate Students during the COVID-19 Pandemic. In *Proc. of ITiCSE'22*. 519–525. https://doi.org/10.1145/3502718.3524764

[20] Tom Crick, Tom Prickett, Christina Vasiliou, Neeranjan Chitare, et al. 2023. Exploring Computing Students Post-Pandemic Learning Preferences with Workshops: A UK Institutional Case Study. In *Proc. of ITiCSE'23*. 173–179. https://doi.org/10.1145/3587102.3588807

[21] Quintin Cutts, Maria Kallia, Ruth Anderson, Tom Crick, et al. 2023. Considering Computing Education in Undergraduate Computer Science Programmes. In *Proc. of ITiCSE'23*. https://doi.org/10.1145/3587103.3594210

[22] James H. Davenport, Tom Crick, and Rachid Hourizi. 2020. The Institute of Coding: A University-Industry Collaboration to Address the UK's Digital Skills Crisis. In *Proc. of EDUCON'20*. 1400–1408. https://doi.org/10.1109/EDUCON45650.2020.9125272

[23] Michael de Raadt, Richard Watson, and Mark Toleman. 2002. Language Trends in Introductory Programming Courses. In *Proc. of InSITE 2002*. 229–337.

[24] Michael de Raadt, Richard Watson, and Mark Toleman. 2004. Introductory Programming: What's Happening Today and Will There Be Any Students to Teach Tomorrow?. In *Proc. of ACE'04*. 277–282.

[25] Yogesh K. Dwivedi et al. 2021. Artificial Intelligence (AI): Multidisciplinary Perspectives on Emerging Challenges, Opportunities, and Agenda for Research, Practice and Policy. *International Journal of Information Management* 53, 101994 (2021). https://doi.org/10.1016/j.ijinfomgt.2019.08.002

[26] Yogesh K. Dwivedi et al. 2023. "So What If ChatGPT Wrote It?" Multidisciplinary Perspectives on Opportunities, Challenges and Implications of Generative Conversational AI for Research, Practice and Policy. *International Journal of Information Management* 71, 102642 (2023). https://doi.org/10.1016/j.ijinfomgt.2023.102642

[27] Katrina Falkner, Sue Sentance, Rebecca Vivian, Sarah Barksdale, et al. 2019. An International Comparison of K-12 Computer Science Education Intended and Enacted Curricula. In *Proc. of Koli Calling'19*. 1–10. https://doi.org/10.1145/3364510.3364517

[28] James Finnie-Ansley, Paul Denny, Brett A. Becker, et al. 2022. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Proc. of ACE'22* (Virtual Event, Australia) *(ACE '22)*. ACM, NY, NY, USA, 10—-19. https://doi.org/10.1145/3511861.3511863

[29] James Finnie-Ansley, Paul Denny, Andrew Luxton-Reilly, Eddie Antonio Santos, et al. 2023. My AI Wants to Know If This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. In *Proc. of ACE'23* (Melbourne, VIC, Australia) *(ACE '23)*. ACM, NY, NY, USA, 97—-104. https://doi.org/10.1145/3576123.3576134

[30] Mark Guzdial. 2018. It Matters a Lot Who Teaches Introductory Courses if We Want Students to Continue. https://computinged.wordpress.com/2018/06/22/it-matters-a-lot-who-teaches-introductory-courses-if-we-want-student-to-continue/.

[31] Joanne Hardman, Richard Watermeyer, Kalpana Shankar, Venkata Ratnadeep Suri, et al. 2022. "Does Anyone Even Notice Us?" COVID-19's Impact on Academics' Well-being in a Developing Country. *South African Journal of Higher Education* 36, 1 (2022), 1–19. https://doi.org/10.20853/36-1-4844

[32] Matthew Hertz. 2010. What Do "CS1" and "CS2" Mean?: Investigating Differences in the Early Courses. In *Proc. of SIGCSE'10*. 199–203. https://doi.org/10.1145/1734263.1734335

[33] Alastair Irons and Tom Crick. 2022. Cybersecurity in the Digital Classroom: Implications for Emerging Policy, Pedagogy and Practice. In *Higher Education in a Post-COVID World: New Approaches and Technologies for Teaching and Learning*. 231–244. https://doi.org/10.1108/978-1-80382-193-120221011

[34] Andrew Luxton-Reilly. 2016. Learning to Program is Easy. In *Proc. of ITiCSE'16*. 284–289. https://doi.org/10.1145/2899415.2899432

[35] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, et al. 2018. Introductory Programming: A Systematic Literature Review. In *Proc. of ITiCSE'18*. https://doi.org/10.1145/3293881.3295779

[36] Raina Mason and Graham Cooper. 2014. Introductory Programming Courses in Australia and New Zealand in 2013 - Trends and Reasons. In *Proc. of ACE'14*.

[37] Raina Mason, Graham Cooper, and Michael de Raadt. 2012. Trends in Introductory Programming Courses in Australian Universities: Languages, Environments and Pedagogy. In *Proc. of ACE'12*. 33–42.

[38] Raina Mason and Simon. 2017. Introductory Programming Courses in Australasia in 2016. In *Proc. of ACE'17*. 81–89. https://doi.org/10.1145/3013499.3013512

[39] Michael McCracken, Vicki Almstrum, Danny Diaz, Mark Guzdial, et al. 2001. A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students. *SIGCSE Bulletin* 33, 4 (2001), 125–180. https://doi.org/10.1145/572139.572181

[40] Fiona McGaughey, Richard Watermeyer, Kalpana Shankar, Venkata Ratnadeep Suri, et al. 2022. 'This Can't Be the New Norm': Academics' Perspectives on the COVID-19 Crisis for the Australian University Sector. *Higher Education Research & Development* 41, 7 (2022). https://doi.org/10.1080/07294360.2021.1973384

[41] Faron Moller and Tom Crick. 2018. A University-Based Model for Supporting Computer Science Curriculum Reform. *Journal of Computers in Education* 5, 4 (2018), 415–434. https://doi.org/10.1007/s40692-018-0117-x

[42] Catherine Mooney and Brett A. Becker. 2021. Investigating the Impact of the COVID-19 Pandemic on Computing Students' Sense of Belonging. In *Proc. of ITiCSE'21* (Virtual Event, USA) *(SIGCSE '21)*. ACM, NY, NY, USA, 612–618. https://doi.org/10.1145/3408877.3432407

[43] Ellen Murphy, Tom Crick, and James H Davenport. 2017. An Analysis of Introductory Programming Courses at UK Universities. *The Art, Science, and Engineering of Programming* 1, 2 (2017). https://doi.org/10.22152/programming-journal.org/2017/1/18

[44] ACM/IEEE CS Joint Task Force on Computing Curricula. 2013. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. https://www.acm.org/education/curricula-recommendations.

[45] The Joint ACM/IEEE Task CS Force on Computing Curricula. 2001. Computing Curricula 2001. https://www.acm.org/education/curricula-recommendations.

[46] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, et al. 2007. A Survey of Literature on the Teaching of Introductory Programming. In *Proc. of ITiCSE-WGR'07*. 204–223. https://doi.org/10.1145/1345443.1345441

[47] Leo Porter and Daniel Zingaro. 2023. *Learn AI-Assisted Python Programming with GitHub Copilot and ChatGPT*. Manning Publications.

[48] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, et al. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proc. of ITiCSE-WGR'23* (Turku, Finland) *(ITiCSE-WGR '23)*. ACM, NY, NY, USA, 52 pages. https://doi.org/10.1145/3623762.3633499

[49] Keith Quille, Roisin Faherty, and Brett A. Becker. 2022. Building K-12 Teacher Capacity to Expand Uptake in a National CS Curriculum. In *Proc. of SIGCSE'22*. 1086. https://doi.org/10.1145/3478432.3499063

[50] Anthony Robins. 2010. Learning Edge Momentum: A New Account of Outcomes in CS1. *Computer Science Education* 20, 1 (2010), 37–71. https://doi.org/10.1080/08993401003612167

[51] Sue Sentance, Diana Kirby, Keith Quille, Elizabeth Cole, et al. 2022. Computing in School in the UK & Ireland: A Comparative Study. In *Proc. of UKICER'22*. 1–7. https://doi.org/10.1145/3555009.3555015

[52] Kalpana Shankar, Dean Phelan, Venkata Ratnadeep Suri, Richard Watermeyer, et al. 2021. "The COVID-19 Crisis is Not the Core Problem": Experiences, Challenges, and Concerns of Irish Academia in the Pandemic. *Irish Educational Studies* 40, 2 (2021), 169–175. https://doi.org/10.1080/03323315.2021.1932550

[53] Angela A. Siegel, Mark Zarb, Bedour Alshaigy, Jeremiah Blanchard, et al. 2021. Teaching through a Global Pandemic: Educational Landscapes Before, During and After COVID-19. In *Proc. of ITiCSE-WGR'21*. 1–25. https://doi.org/10.1145/3502870.3506565

[54] Angela A. Siegel, Mark Zarb, Emma Anderson, Brett Crane, et al. 2022. The Impact of COVID-19 on the CS Student Learning Experience: How the Pandemic has Shaped the Educational Landscape. In *Proc. of ITiCSE-WGR'22*. 165–190. https://doi.org/10.1145/3571785.3574126

[55] Simon, Andrew Luxton-Reilly, Vangel V. Ajanovski, Eric Fouh, et al. 2019. Pass Rates in Introductory Programming and in Other STEM Disciplines. In *Proc. of ITiCSE-WGR'19*. 53–71. https://doi.org/10.1145/3344429.3372502

[56] Simon, Raina Mason, Tom Crick, James H. Davenport, et al. 2018. Language Choice in Introductory Programming Courses at Australasian and UK Universities. In *Proc. of SIGCSE'18*. https://doi.org/10.1145/3159450.3159547

[57] Andreas Stefik and Stefan Hanenberg. 2014. The Programming Language Wars: Questions and Responsibilities for the Programming Language Community. In *Proc. of Onward! 2014*. 283–299. https://doi.org/10.1145/2661136.2661156

[58] ACM/IEEE CS CS2008 Review Taskforce. 2008. Computer Science Curriculum 2008: An Interim Revision of CS 2001. https://www.acm.org/education/curricula-recommendations.

[59] David W. Valentine. 2004. CS Educational Research: A Meta-analysis of SIGCSE Technical Symposium Proceedings. In *Proc. of SIGCSE'04*. 255–259. https://doi.org/10.1145/971300.971391

[60] Troy Vasiga. 2002. What Comes after CS 1 + 2: A Deep Breadth before Specializing. *SIGCSE Bulletin* 34, 1 (2002), 28–32. https://doi.org/10.1145/563517.563350

[61] Colin C. Venters, Rafael Capilla, Stefanie Betz, Birgit Penzenstadler, et al. 2018. Software Sustainability: Research and Practice from a Software Architecture Viewpoint. *Journal of Systems and Software* 138 (2018), 174–188. https://doi.org/10.1016/j.jss.2017.12.026

[62] Colin C. Venters, Rafael Capilla, Elisa Yumi Nakagawa, Stefanie Betz, et al. 2023. Sustainable Software Engineering: Reflections on Advances in Research and Practice. *Information and Software Technology* (2023). https://doi.org/10.1016/j.infsof.2023.107316

[63] Rupert Ward, Oliver Phillips, David Bowers, Tom Crick, et al. 2021. Towards a 21st Century Personalised Learning Skills Taxonomy. In *Proc. of EDUCON'21*. 344–354. https://doi.org/10.1109/EDUCON46332.2021.9453883

[64] Richard Watermeyer, Tom Crick, and Cathryn Knight. 2022. Digital Disruption in the Time of COVID-19: Learning Technologists' Accounts of Institutional Barriers to Online Learning, Teaching and Assessment in UK Universities. *International Journal for Academic Development* 27, 2 (2022), 148–162. https://doi.org/10.1080/1360144X.2021.1990064

[65] Richard Watermeyer, Tom Crick, Cathryn Knight, and Janet Goodall. 2021. COVID-19 and Digital Disruption in UK Universities: Afflictions and Affordances of Emergency Online Migration. *Higher Education* 81 (2021), 623–641. https://doi.org/10.1007/s10734-020-00561-y

[66] Richard Watermeyer, Cathryn Knight, Tom Crick, and Margarida Borras Batalla. 2023. 'Living at Work': COVID-19, Remote-working and the Spatio-relational Reorganisation of Professional Services in UK Universities. *Higher Education* 85 (2023), 1317–1336. https://doi.org/10.1007/s10734-022-00892-y

[67] Richard Watermeyer, Kalpana Shankar, Tom Crick, Cathryn Knight, et al. 2021. 'Pandemia': A Reckoning of UK Universities' Corporate Response to COVID-19 and its Academic Fallout. *British Journal of Sociology of Education* 42, 5-6 (2021), 651–666. https://doi.org/10.1080/01425692.2021.1937058