An Analysis of Introductory University Programming Courses in the UK

Ellen Murphy
Institute for
Mathematical Innovation
University of Bath
e.murphy@bath.ac.uk

Tom Crick
Dept. of Computing
Cardiff Metropolitan University
tcrick@cardiffmet.ac.uk

James H. Davenport
Dept. of Computer Science
University of Bath
j.h.davenport@bath.ac.uk

ABSTRACT

This paper reports the results of a survey of xx introductory programming courses delivered at UK universities as part of their first year computer science (or similar) degree programmes, conducted in the first half of 2016. Results of this survey are compared with a related survey conducted since 2010 (as well as earlier surveys from 2001 and 2003) on universities in Australia and New Zealand. Trends in student numbers, programming paradigm, programming languages and environment/tools used, as well as the reasons for choice of such are reported. Other aspects of first programming courses such as instructor experience, external delivery of courses and resources given to students are also examined.

The results indicate a trend towards...

Categories and Subject Descriptors

K.3.2 [**Computers & Education**]: Computer and Information Science Education—*Computer Science Education*; K.4.1 [**Computers And Society**]: Public Policy Issues

Keywords

Introductory Programming, Programming Languages, Programming Environments, Computer Science Education, Higher Education, Tertiary Education, UK

1. INTRODUCTION

[8] is the latest in a long line [12, 9] of papers surveying the teaching of introductory programming courses in Australasia. However, such surveys are not the norm elsewhere, and this paper reports the authors' findings from running the first such survey in the United Kingdom. We remind the reader that in the United Kingdom (2011 cenus population 63.2 million), Higher Education is the responsibility of the Westminster Goverment in England (53 million) and of the devolved goverments in Scotland (5.3 million), Wales (3.1 million) and Northern Ireland (1.8 million).

SIGCSE 2017 Seattle, Washington, USA

ACM ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

UK policy context e.g. schools [1, 2], Shadbolt/Wakeham, The Shadbolt enquiry split generalist universities in England in three bands, based on their average (across all subjects) entrance tariff (qualifications of entrants), and we have followed that banding when we split the English results, so our data should be comparable with [**?**]

TEF, graduate employability, etc.

Other work in this space [10, 7, 4, 11, 6]

Also, our previous work [3, 5].

2. METHODOLOGY

2.1 Recruitment of Participants

To recruit participants for the survey, a general invitation email was sent to the Council of Professors and Heads of Computing (CPHC) mailing list. CPHC have members from over 100 UK universities, and are the representational body for this group in the UK[1].The invitation asked for the survey to be passed on to the most appropriate person to fill it out; Director of Studies, Chair of Teaching Committee or the best fit for the individual institution. A couple of reminder circular e-mails were also issued.

The survey was hosted online, and was open from mid-May until the end of June 2016, at which point it was closed and the results were downloaded and analysed. Due to the recruitment method, there were duplicate responses from some departments, and these were reconciled by direct enquiry.

2.2 Questions

The questions used in the survey were generously provided by the authors of [8], so as to allow direct comparison between the results of this survey and that of the Australian/New Zealand 2014 survey. Where possible, questions were left unchanged, although a small minority were edited to reflect the UK target audience.

As written in [8], text in the survey made clear that the terminology "course" was used for "the basic unit of study that is completed by students towards a degree, usually studied over a period pf a semester or session, in conjunction with other units of study".

The first section of the survey asked about the programming language(s) in use, the reasons for their choice, and the perceived difficulty of usefulness of the programming language(s). Following this were questions regarding the use of environments or development tools; which ones were

[1] `https://cphc.ac.uk/who-we-are/`

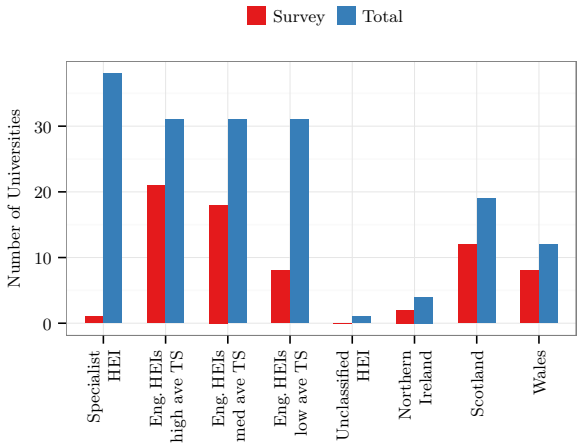

Figure 1: The number of responding universities per Nation/ Tariff Group.

used, the reasons for their choice and the perceived difficulty. General questions about paradigm, instructor experience and external delivery were asked, along with questions regarding students receiving unauthorised assistance, and the resources provided to students. Finally, participants were asked to identify their top three main aims when teaching introductory programming, and were also allowed to provide further comments.

In [8], participants were asked to rank the importance of the given (up to three (@James: not sure if this is true; I will check)) reasons for choosing a programming language, environment or tool. Due to technical limitations in online survey tool used, it was not possible to do so in this survey, so our Figurefig:reasons just reports counts.

3. RESULTS AND DISCUSSION

3.1 Universities and Courses

We had responses from 79 universities. Excluding the Open University's 3200 students, this was 13462 students, with a mean of 173 (but a standard deviation of 88). Looking at Figure 1 we see good response rates, apart from the specialist HEIs (most of whom do not teach computing) and the "low tariff" English ones. Fewer of these teach computing, but we are not convinced that this factor alone explains the response rate.

3.2 Languages

This is the immediate major difference with [8]. Their survey showed a dead heat (27.3% of courses) between Java and Python, with Python winning (33.7% to 26.9%) when weighted by the number of students on the course. Our findings (Figure 2) show that Java is a clear winner by any metric, being used in just under half the courses, while the runner-up, Python, is in use in 13.2%. The C family (C, C++ and C#) together scores 23.6% by courses and 19.5% by students. Figure 2 shows the effect of student-number weighting *but* we have excluded the Open University from this weighting, as its 3200 students learning Python (and Sense, a variant of Scratch) would have distorted the comparison.

Table 1: The number of programming languages used in first programming courses.

Languages	1	2	3	4
Courses	59	17	3	1

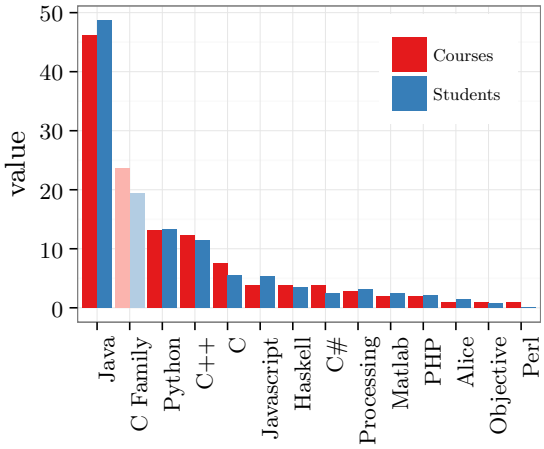

Figure 2: Language popularity by percentage of courses and students (excl. OU).

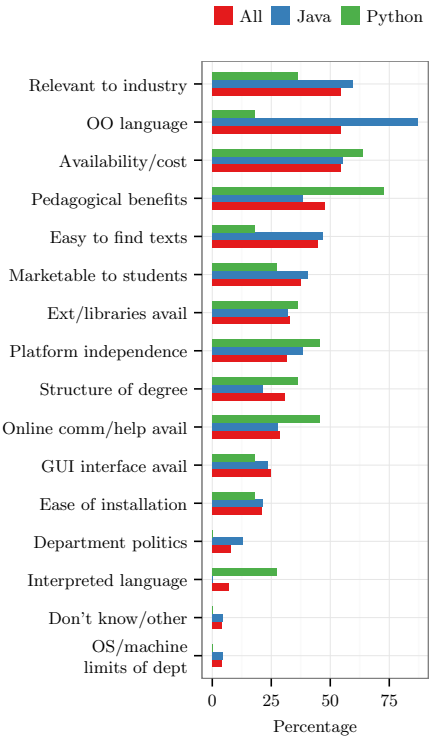

Figure 3: Reasons given for choosing a programming language by percentage for: all languages; Java; and Python.

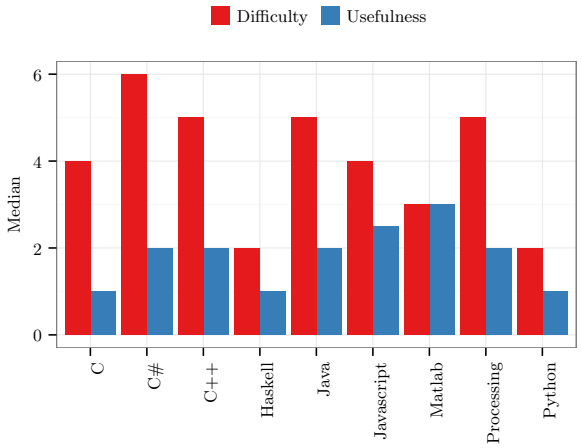

Figure 4: The median of the perceived difficulty and usefulness of language, where 1 is 'extremely easy' and 7 is 'extremely difficult' for difficulty and 1 is 'extremely useful' and 7 is 'extremely useless' for usefulness. Answers must have been given by at least two instructors.

Table 2: The main paradigm in use in the first programming course.

Paradigm	Object-Oriented	Procedural	Functional
Courses	40	27	7

Figure 3 shows some of the reasons for this: Java scores higher on "relevance to industry" and, perhaps somewhat surprisingly, much higher on "ObjectOriented language". @Tom: do you agree with "somewhat surprisingly"?

Figure 5 breaks down the choice of language by nation and tariff group. It is noticeable that the three English tariff groups differ significantly, with Python outnumbering Java in the low tariff universities, and C being almost exclusively in the high tariff universities.

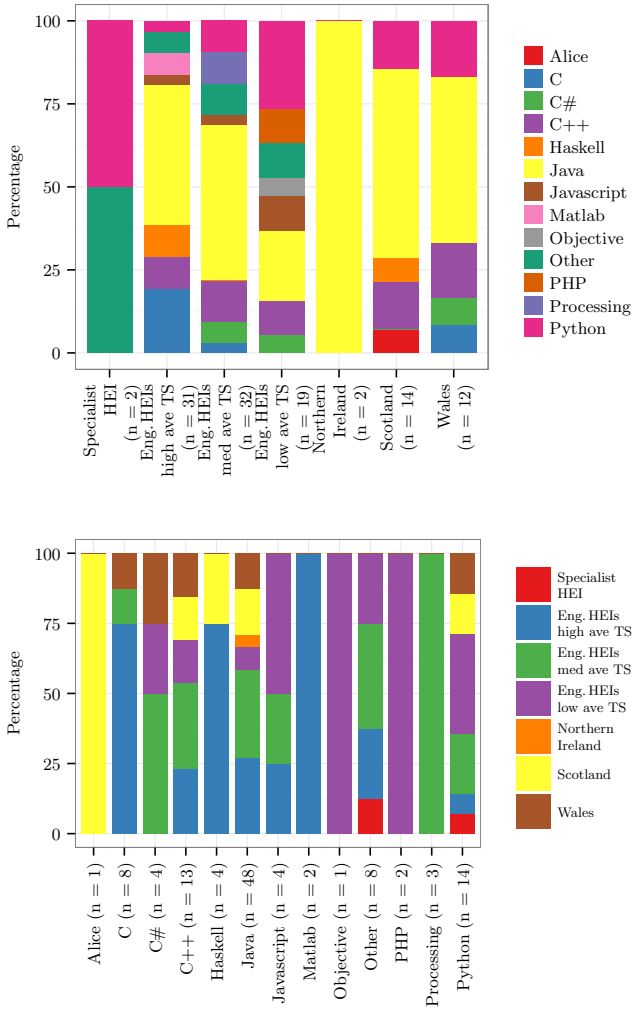

Figure 5: The breakdown of programming languages by Nation and Tariff Groups.

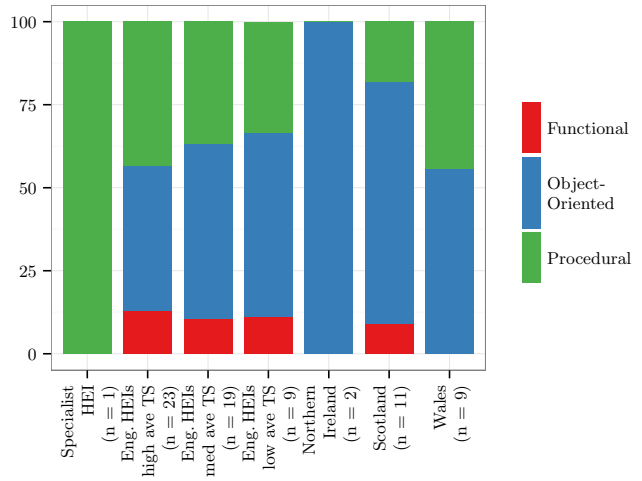

Figure 6: The breakdown of the main paradigm in use for every Tariff Group.

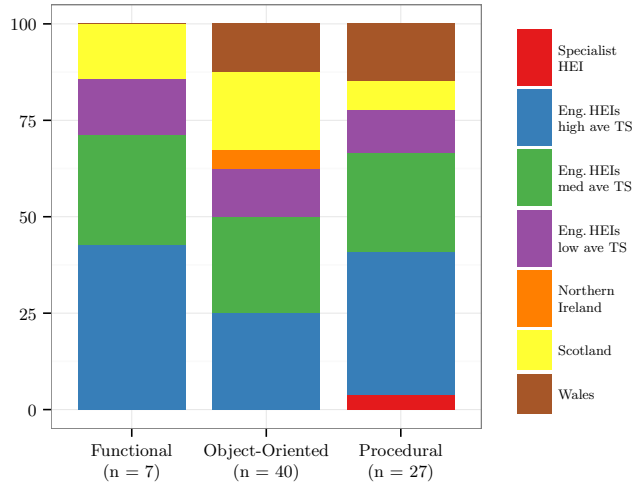

Figure 7: The breakdown of Tariff Groups for each paradigm.

3.3 Instructor Experience

Data about the experience of the main teacher (@Ellen: can we quote the question here) are given in Table 3.

3.4 IDEs and Tools

Various data about IDEs and tools are collected in Figures 10–13. We note that, while Eclipse is the most popular tool by some way (Figure 10), it is also deemed to be most difficult (Figure 13). This, apparently perverse, practice might be explained by the extent of re-use of Eclipse in other courses (Figure 12).

3.5 Other Aspects of the Course

3.6 Aims of an Introductory Programming Course

[8] asked their respondents for the aims of their introduc-

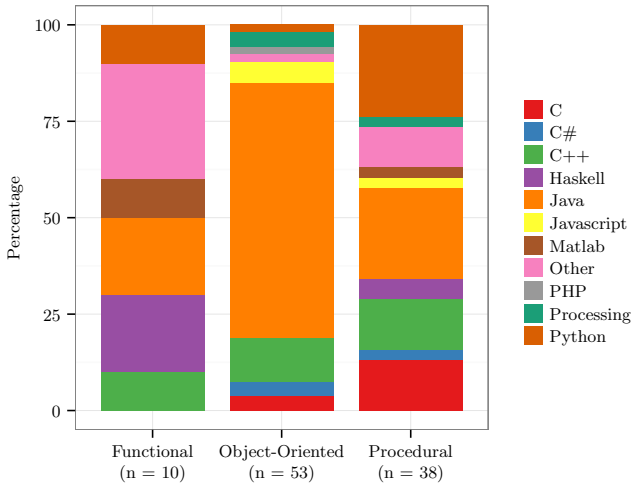

Figure 8: The breakdown of programming languages in use for each paradigm.

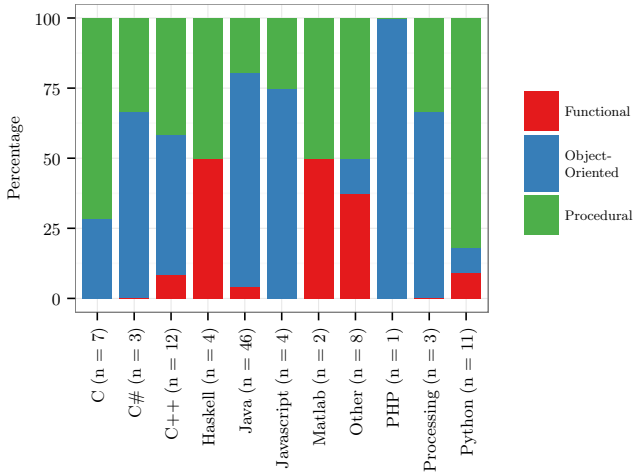

Figure 9: The breakdown of the main paradigm in use for each programming language.

Table 3: The number of years the instructor has been involved in teaching introductory programming.

Years	<2	2 - 5	5 - 10	10 - 20	20 - 30	>30
Instructors	3	9	9	27	19	7

Table 4: The number of tools/environments used in first programming courses.

Tools	1	2	3	4	8
Courses	34	15	6	2	1

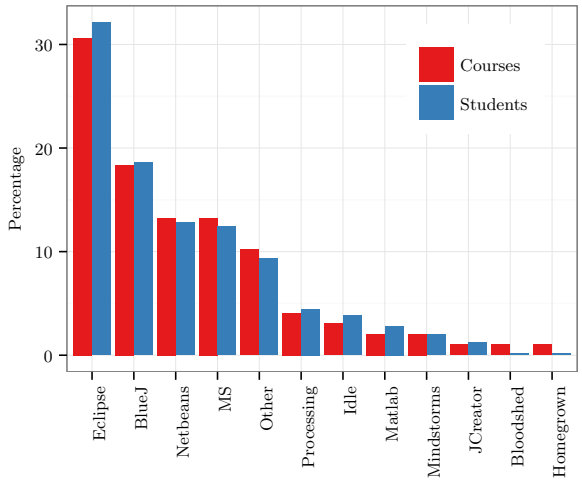

Figure 10: Tool or environment popularity by percentage of courses and students.

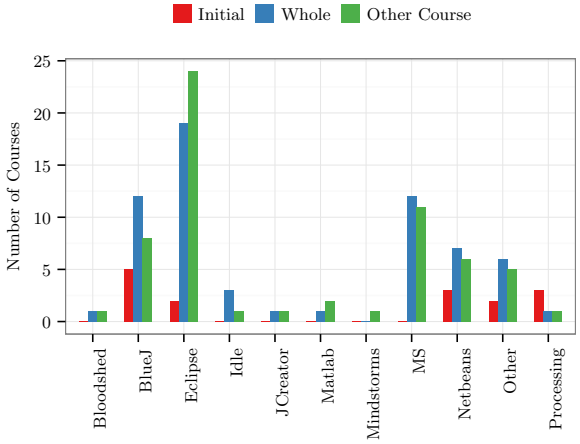

Figure 12: For each tool or environment, whether it is used: for an initial part of the first programming course; throughout the whole of the first programming course; in any other course in the degree.

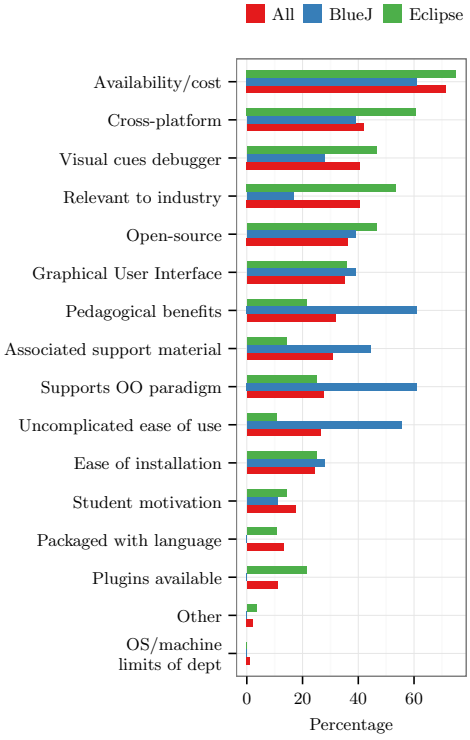

Figure 11: Reasons given for choosing a tool or environment by percentage for: all tools and environments; BlueJ; and Eclipse.

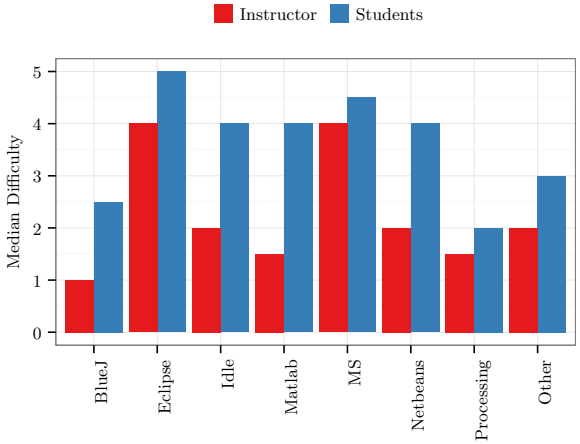

Figure 13: The median difficulty rating of tool/environment for the instructor and students to use, where 1 is 'extremely easy' and 7 is 'extremely difficult'. Answers must have been given by at least two instructors.

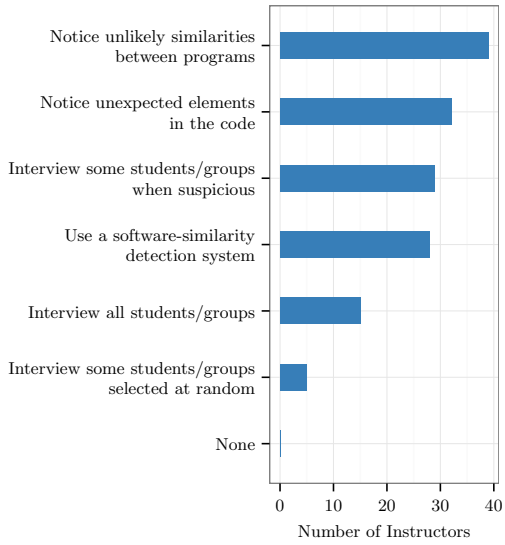

Figure 14: Steps taken to determine whether students have received unauthorised assistance on assignments.

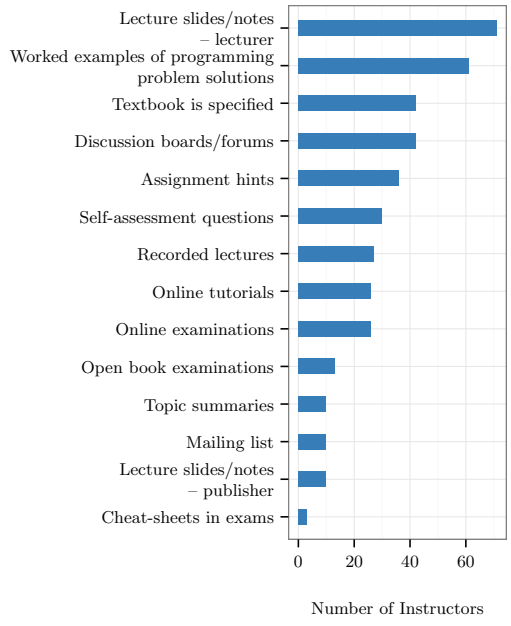

Figure 15: Resources provided to students.

tory programming course. They, and we, asked for (up to) three aims. The authors then attempted to classify the free-text answers into the same categorisation as [8] used. While it is trivial to map the written aim "Thinking algorithmically" to [8]'s "Algorithmic thinking" and so on, many were not so clear: for example, we mapped "To learn a specific language" to "syntax/writing basic code". There were also a class of aims, such as "Establish professional software development practices", that seemed very coherent, but didn't map clearly to the [8] aims. These we have categorised as "Software Engineering".

4. GENERAL DISCUSSION

4.1 The U.K. context

4.2 Comparison with Australasia

Here we compare with [8], the latest Australasian survey. We have already commented on the major difference in language choice, which colours many of the other comparisons. Another difference shows up in Figure 10 versus [8]'s Figure 11. There, "None" and "Other" were the top two categories, with Idle, at 15%, the most popular named product. In the UK, "Other" is fifth and Idle seventh.

5. ACKNOWLEDGEMENTS

The authors would like to thank the participants for their engagement with the survey, as well as the authors of [8] for providing us with their survey and permission to use it. We are grateful to the GW4 Alliance (Universities of Bath, Bristol, Cardiff and Exeter) for funding the underpinning survey.

6. REFERENCES

[1] N. Brown, M. Kölling, T. Crick, S. Peyton Jones, S. Humphreys, and S. Sentance. Bringing Computer Science Back Into Schools: Lessons from the UK. In *Proc. of SIGCSE 2013*, pages 269–274, 2013.

[2] N. Brown, S. Sentance, T. Crick, and S. Humphreys. Restart: The Resurgence of Computer Science in UK Schools. *ACM TOCE*, 14(2):1–22, 2014.

[3] T. Crick, J. H. Davenport, and A. Hayes. Innovative Pedagogical Practices in the Craft of Computing. Innovative Pedagogies in the Disciplines. Higher Education Academy, August 2015. https://www.heacademy.ac.uk/innovative-pedagogical-practices-craft-computing.

[4] N. B. Dale. Most difficult topics in CS1: results of an online survey of educators. *ACM SIGCSE Bulletin*, 38(2):49–53, 2006.

[5] J. H. Davenport, A. Hayes, R. Hourizi, and T. Crick. Innovative Pedagogical Practices in the Craft of Computing. In *Proceedings of 4th International Conference on Learning and Teaching in Computing and Engineering (LaTiCE)*, 2016.

[6] P. Guo. Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities. http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities, July 2014.

[7] D. Gupta. What is a good first programming language? *Crossroads*, 10(4), 2004.

[8] R. Mason and G. Cooper. Introductory programming courses in Australia and New Zealand in 2013 – trends and reasons. In *Proceedings of the 16th Australasian Computing Education Conference (ACE'14)*, pages 139–147, 2014.

[9] R. Mason, G. Cooper, and M. Raadt. Trends in introductory programming courses in Australian universities: languages, environments and pedagogy. In *Proceedings of the 14th Australasian Computing Education Conference (ACE'12)*, pages 33–42, 2012.

[10] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, 33(4):125–180, 2001.

[11] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson. A Survey of Literature on the Teaching of Introductory Programming. *ACM SIGCSE Bulletin*, 39(4):204–223, 2007.

[12] M. Raadt, R. Watson, and M. Toleman. Trends in introductory programming courses in Australian universities: languages, environments and pedagogy. In *Proceedings of the 6th Australasian Computing Education Conference (ACE'04)*, pages 277–282, 2004.