Introductory Programming Courses in Australia and New Zealand in 2013 - trends and reasons

Raina Mason
Southern Cross Business School
Southern Cross University
Coffs Harbour NSW Australia
raina.mason@scu.edu.au

Graham Cooper
Southern Cross Business School
Southern Cross University
Coffs Harbour NSW Australia
graham.cooper@scu.edu.au

Abstract

This paper reports the results of a survey of 38 introductory programming courses in Australian and New Zealand universities, conducted in the first half of 2013. Results of this survey are compared with a survey conducted in 2010 on Australian universities and two other previous studies conducted in 2001 and 2003. 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 the adoption of Python for Introductory Computer Programming courses and that this language is being used in a structured approach for programming. Introductory computer programming courses that focus upon an Object Orientated approach predominantly use Java.

Keywords: introductory programming, programming languages, programming environments, Australian university courses, New Zealand university courses, pedagogy, trends.

1 Introduction

Most Computer Science and Information Technology degree programs include at least one compulsory introductory programming course. Programming is generally perceived to be complex and difficult and these courses can suffer from high attrition rates and low levels of competency (McCracken *et al.* 2001). Debate continues on which languages, environments and paradigms should be used in a first programming course to maximise student success and motivation (Bloch 2000, Jenkins 2002, Pears *et al.* 2007, Dale 2005, 2006).

To establish the (then) current state-of-play in Australian and New Zealand universities, censuses were conducted in 2001 and 2003 (de Raadt *et al.* 2002, 2004) which reported on the languages, paradigms and environments/tools being used, the reasons for choice of language, student numbers (and the downwards trend) in each course, texts employed, instructor experience and the teaching of problem solving strategies.

In the latter months of 2010 a phone interview survey which repeated the previous two surveys with minor changes was performed with a large sample of 44 programming courses, across 28 Australian universities (Mason *et al.* 2012). Longitudinal trends in languages, tools and paradigms were identified, as well as reported reasons for such changes over the 10 year period since the survey was initially conducted. The 2010 survey showed Java as the most popular language, followed by Python and then C. "Pedagogical benefits" was the most common reason for the choice of language, followed by "Relevance to industry/marketability to students". The procedural paradigm was most often used for teaching, and fewer participants (20% compared to 43% in 2003) were choosing to use only text editors and command-line compilers rather than IDEs or other tools.

In early 2013 the survey was repeated in an online survey format, with Australian and New Zealand universities invited to participate. Details about the interview questions and the methodology of the study are described in the next section, followed by results and discussion of the implications for teaching introductory programming.

2 Methodology

2.1 Recruitment of participants

The list of participants from the 2010 study was used as a starting point for contacting potential participants. An email was sent to each previous participant inviting participation in the 2013 study. As the survey was to be conducted online, rather than by telephone interview which imposed cost and time-zone difference issues, New Zealand universities were included in this study. University websites were used to identify potential participants from New Zealand and invitations were sent either directly to potential participants, or to administrative staff responsible for those programs. A general invitation to participate was sent to the SIGCSE-Australasian mailing list and the SIGCSE list for the attention of the Australian and New Zealand members.

The online survey was open from mid-April to mid-July 2013, when it was closed and the results were downloaded and analysed.

2.2 Questions

For all questions, the terminology "course" was used for the basic unit of study that is completed by students towards a degree, usually studied over the period of a semester or session in conjunction with other units of study ("courses"). This terminology was used to maintain consistency with the previous studies.

Large portions of the 2013 survey questionnaire were drawn from the previous studies including questions about language and paradigm choice, programming environment/development tools, instructor experience, reasons for choice of language, and perceived difficulty of the language and environment (if one is used).

Additional questions were added to ascertain the relative importance of each reason given for language and environment choice, It was anticipated that there may be a relationship between the choice of language and environments and the reasons for these choices. Instructors were also asked how useful the language was for teaching the fundamental concepts of programming.

A final section asked instructors to identify what he or she considered to be the 3 most important aims of the introductory programming course. Other general interest questions were asked regarding whether the course was offered in external mode, and what resources were provided to students.

3 Results and Discussion

The results of this study are reported below, with comparison to the previous three studies where applicable.

3.1 Universities and Courses

The number of courses covered in the 2013 study was fewer than each of the other three studies. Forty-eight courses from twenty-nine Australian and New Zealand universities participated, however eight participants failed to progress through the study and these surveys were not analysed. A further two participants gave details of the course and student numbers but did not answer questions on programming languages or environments. Some participants answered most but not all questions. This has been indicated in the results and discussion where necessary. Participants were asked for their course codes and universities, so matching could be performed with previous surveys, where necessary.. This also eliminated possible duplication.

3.2 Student Numbers

Comparison of the 2001, 2003, 2010 and 2013 course participation and reported numbers of students are given in Table 1.

	2001	2003	2010	2013
Courses in study	57	71	44	38
Total students in study	19900	16300	7743	10454
students/course	349	229	176	264

Table 1: Course and Students summary

The decline in the numbers of students studying programming was a serious concern in 2003 and 2010. Average enrolments halved from 2001 to 2010, following a general trend in declining student enrolments in all areas of ICT education, as reported by the Australian Computer Society (Australian Computer Society 2011).

From 2010 to 2013 there appeared to have been a 50% increase in the average number of students per course, bouncing back to pre-2003 levels. In case this was an institution effect (i.e. larger institutions participating in this survey than in 2010), where possible, courses that participated in the 2010 study were directly compared with the same courses in the 2013 study. Comparing courses that participated in both studies gave an increase from a 2010 mean of 198 students per course to a 2013 mean of 253 students per course - a 27.8% increase in students over 4 years. The apparent increase in student numbers is consistent with the trends in ACS data to 2010 (latest figures) which show a 4.5% increase in enrolments across the sector from 2009 to 2010 (ACS, 2012).

This is good news for the ICT industry which is predicting a significant shortfall of suitably educated and skilled ICT professionals in the near future (DEEWR 2011).

3.3 Languages

3.3.1 Choice of Language(s)

One of the main areas of interest to this study was the language(s) being used in these introductory programming courses. Instructors were presented with a choice of languages used in the previous three studies and asked to indicate which they used in their courses, as well as offered a space to indicate other languages.

In the 2013 study, a total of 12 languages were used in first programming courses. The majority (33) of courses used one language throughout the first programming course (Table 2). When more than one language was used, the generic approach adopted was for one language to be used initially and then another language added (while keeping the first). In only one case the course was segmented into learning different languages consecutively.

# of languages	2010 courses	2013 courses
1	37	33
2	4	4
3-6	3	1

Table 2: Comparison of number of languages/course

Language	Courses	%age	Weighted by students
Java	12	27.3%	26.9%
Python	12	27.3%	33.7%
C#	4	9.1%	4.8%
C	3	6.8%	8.6%
Javascript	3	6.8%	10.3%
Visual Basic	3	6.8%	1.4%
C++	2	4.5%	3.0%
Ada	1	2.3%	1.7%
Haskell	1	2.3%	1.7%
Matlab	1	2.3%	1.7%
Scribble	1	2.3%	5.6%
Alice	1	2.3%	0.5%

Table 3: 2013 Languages

The programming languages used by the participant courses are shown in Table 3. Languages are presented by number of courses, percentage of courses, and weighted by student numbers. Note that the "courses" column will add to more than 38 courses, as some courses used more than one language.

The top three languages in the first half of 2013 (in order) were Java, Python and C# (by courses) and Python, Java and Javascript weighted by students (Figure 1).

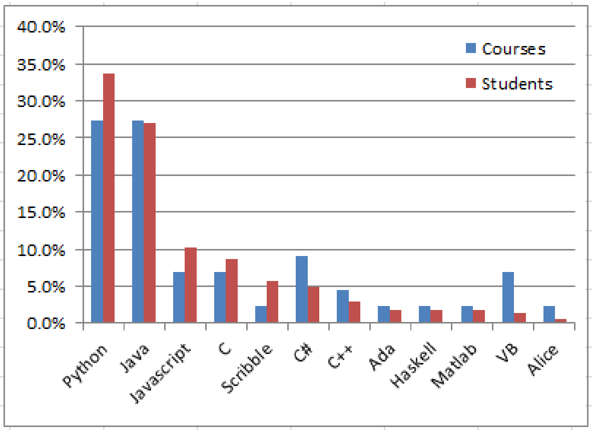

Figure 1: Programming Languages in 2013 by courses and students.

	2001	2003	2010	2013	change
Java	40.4%	40.8%	36.4%	27.3%	-9.1%
Python	0%	0%	13.6%	27.3%	13.7%
C#	0%	0%	9.1%	9.1%	0%
C	7%	12.7%	11.4%	6.8%	-4.6%
VB/VB.NET	24.6%	26.8%	9.1%	6.8%	-2.3%
Javascript	0%	0%	2.3%	6.8%	4.5%
C++	14%	11.3%	7%	4.5%	-2.5%
Matlab	0%	1.4%	2.3%	2.3%	0%
Alice	0%	0%	2.3%	2.3%	0%
Haskell	5.3%	4.2%	0%	2.3%	2.3%
Ada	1.8%	0%	0%	2.3%	2.3%
Processing	0%	0%	4.5%	0%	-4.5%
Fortran	0%	1.4%	2.3%	0%	-2.3%

Table 4: Longitudinal language comparison – courses

The percentages of introductory programming courses exposed to various languages across all four studies is shown in Table 4, with the percentage change in 2013 from 2010.

Similarly, the percentages of students exposed to various languages in introductory programming courses across all four studies is shown in Table 5, with the percentage change in 2013 from 2010.

For the first time in (at least) 13 years, Java has lost the top language crown. In 2010 Java was used by nearly 40% of students, with Python trailing as second most popular language at nearly 20%. In 2013 the positions have reversed, with Python being used by nearly 34% of students, and Java 27%. Note that these two languages represent more than 54% of courses in this study, and more than 60% of the students.

	2001	2003	2010	2013	change
Python	0%	0%	19.5%	33.7%	14.2%
Java	43.9%	44.4%	39%	26.9%	-12.1%
Javascript	0%	0%	1.5%	10.3%	8.8%
C	5.5%	10.6%	11.9%	8.6%	-3.3%
C#	0%	0%	8.2%	4.8%	-3.4%
C++	15.2%	18.7%	4.9%	3%	-1.9%
Matlab	0%	1%	1.3%	1.7%	0.4%
Haskell	8.8%	6%	0%	1.7%	1.7%
Ada	1.7%	0%	0%	1.7%	1.7%
VB/VB.NET	18.9%	16.4%	5.2%	1.4%	-3.8%
Alice	0%	0%	0.9%	0.5%	-0.4%
Processing	0%	0%	5.3%	0%	-5.3%
Fortran	0%	0.7%	3.9%	0%	-3.9%

Table 5: Longitudinal language comparison – students

Javascript is now the third most popular language, displacing C. Visual Basic has continued its downwards slide to just 1.4% of students. The trends in popularity of the top 3 languages of each year are visually depicted in Figure 2 (by courses) and Figure 3 (by students).

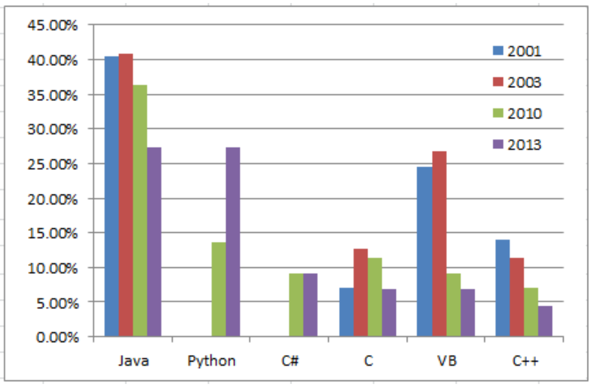

Figure 2: Longitudinal trends – top 3 languages of each year by courses.

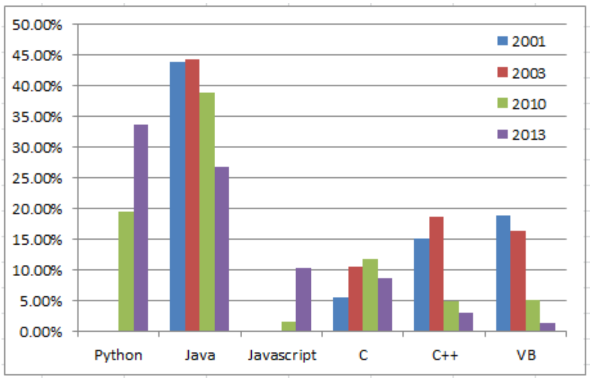

Figure 3: Longitudinal trends – top 3 languages of each year by students.

3.3.2 Reasons for choice of language

In the previous studies instructors were asked about the reasons for their choice of language. The two most common reasons given in both the 2001 and 2010 studies were "industry relevance/marketability to students" and "pedagogical benefits". The 2010 study saw shifts in the frequency of some of the reasons given, with, for example, industry relevance/marketability declined from 56.1% to 48.8% and pedagogical benefits increased from 33.3% to 53.5%.

The 2001 and 2010 surveys identified the reasons for the choice of language, but did not distinguish between the importance of these reasons. For example, an instructor may have indicated that "structure of degree/department politics" and "platform independence" were two reasons for their choice of language. One of these reasons may have been very important in their choice, and the other only slightly important. The 2001 and 2010 studies did not distinguish between the importance of these reasons and only counted frequencies of given reasons.

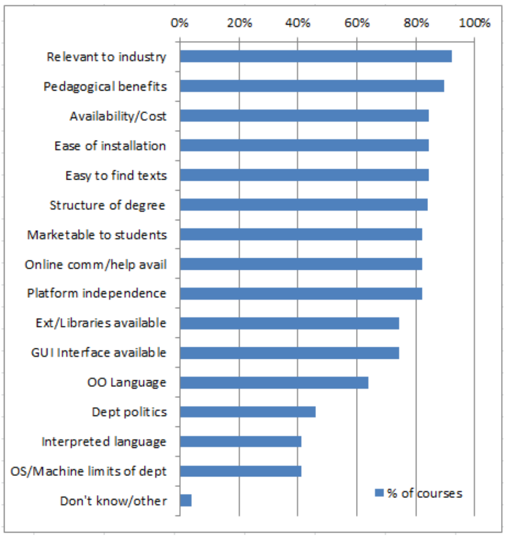

Figure 4: Frequency of reasons given for choice of languages.

To address this issue, the 2013 survey asked participants to rate each reason as not applicable, slightly important, important or very important.

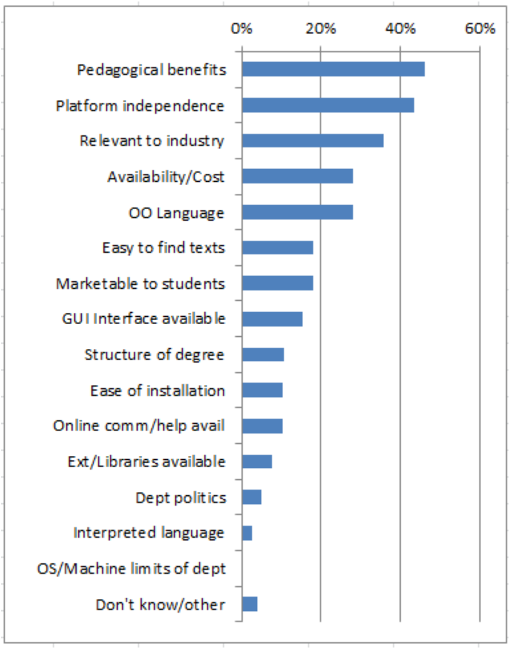

Figure 5: "Very important" reasons for language choice in 2013.

The reasons offered for choices were those offered by the participants in the 2010 survey as well as a space for 'Other'. Figure 4 shows the frequency for reasons given for choice of programming language - not weighted by importance of the reason.

Figure 5 presents the frequencies for identifying a reason for choice of language as "very important".

The first three ranks for "very important" reasons are: 46% for "Pedagogical benefits of the language", 44% for "Platform independence", and 36% for "Relevant to industry". A second analysis was conducted whereby the frequencies for identifying a reason as either "important or very important" was considered. The rank order of reasons between these two methods of analysing importance are not the same. In this case the first three ranks (noting that there was a tied first rank and tied third rank) for "important or very important" reasons are: 79% for "Pedagogical benefits of the language", 79% for "Relevant to industry", 67% for "Platform independence and 67% for "Availability/ cost to students".

It should be noted that both methods of analysis return "Pedagogical benefits of the language" as a first rank.

Comparison of Python and Java

Given that Python has had a large increase in popularity, with a corresponding drop in popularity for Java, and given that these two languages represent over 60% of students in the survey, it was decided to make direct comparisons between the reasons for choice of Python and Java. Note that not all participants who use Python and Java have given reasons for their choice.

The first method of analysis for this purpose was to identify the reasons which all participants identified as a reason for the choice of language (varying importance being either slightly important, important or very important).

Python: All of the Python-using participants gave the following reasons for their choice (varying importance):

- Availability/Cost to students
- Easy to find texts
- Extensions/Libraries available
- Platform independence

Java: In contrast, all of the Java-using participants gave the following reasons for their choice (varying importance):

- Object-Oriented Language
- Online community/Help available
- Relevant to industry

It is interesting to note that there is an absence of overlap between these two sets of reasons. That is, the set of reasons which all instructors using Python offered for their choosing of Python is mutually exclusive to the set of reasons which all instructors using Java offered for their choosing of Java. Although Java is free for students and platform independent, these reasons appear to be more important to those choosing Python. Although Python is an object-oriented language, those looking for an object-oriented language are tending to choose Java. See Section 3.4 for more information about paradigm choices and the relation to language choice.

Note, however, that this analysis includes identification of reasons that are 'slightly important'. Excluding the 'slightly important' reasons to focus upon the combined set of important / very important reasons yields the data presented in Figure 6 showing the set of important/very important reasons given for choice of either Java or Python.

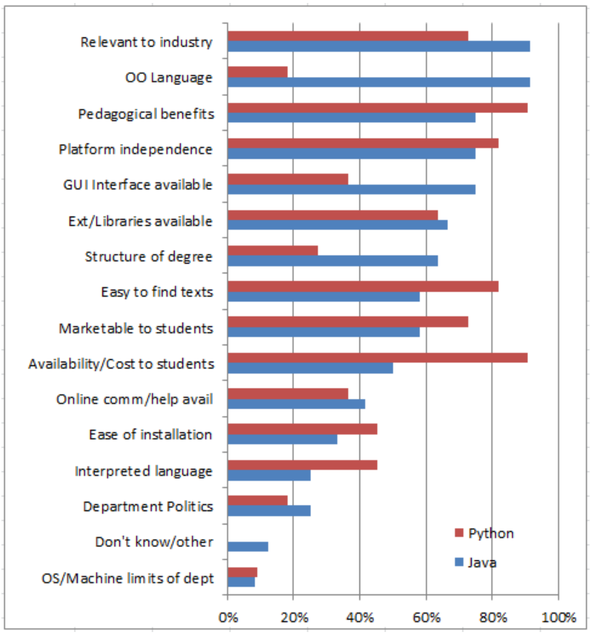

Figure 6: Important/V.Important reasons for choice of Python or Java

From Figure 6, the important/very important reasons that return at least an 80% selection rate for choice of Python are:

- 91% Availability / Cost to students [Java 50%]
- 91% Pedagogical benefits [Java 75%]
- 82% Platform Independence [Java 75%]
- 82% Easy to find texts [Java 58%]

The important/very important reasons that return at least an 80% selection rate for choice of Java are:

- 92% Object-oriented language [Python 18%]
- 92% Relevant to industry [Python 73%]

It should be noted that Python is an object oriented language but can be used in a structured way with no necessity to discuss objects (at an introductory level). In comparison, Java is difficult to teach without providing some class structure, either by using an environment such as BlueJ or Greenfoot or by providing students with skeleton code and getting them to fill in the blanks.

The choice of language appears to have not been done at a mere surface level, but rather, with deep consideration as to how the language is to be used strategically with respect to presenting programming activities to students.

3.3.3 Perceived difficulty and usefulness to teach fundamental concepts

Participants were asked to indicate how difficult they believed their chosen language was for novice students, on a Likert scale of 1 - 7 where 1 was 'very easy' and 7 was 'very difficult'. The medians of the results are given below in Figure 7. Note only languages where answers have been given by at least 2 participants have been included.

From these results, Java is perceived as more difficult for novices than Python. C is considered the most difficult for novices.

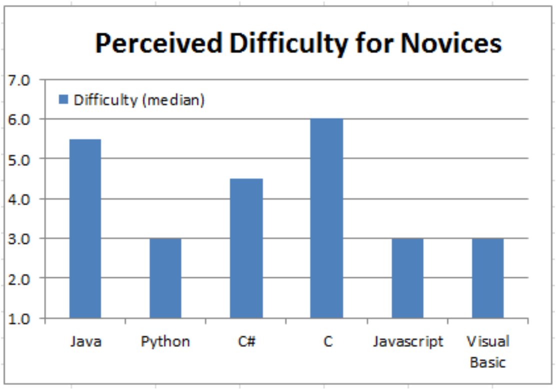

Figure 7: Perceived difficulty of language for novices

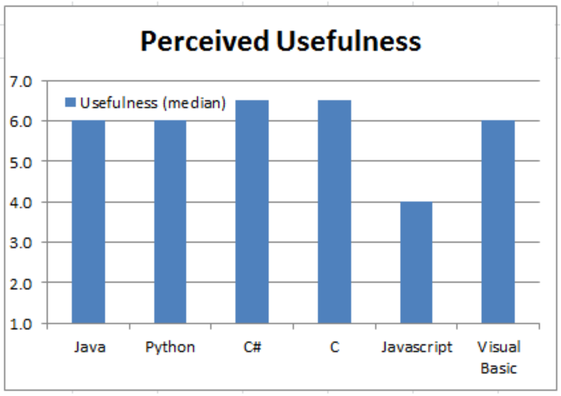

Figure 8: Perceived usefulness of language for teaching fundamental concepts of programming.

Regardless of whether or not the various languages really do exhibit these relative levels of difficulty, instructors are indicating that they perceive these relative levels of difficulty to exist, and this may be a factor of consideration in their choice of language.

Participants were also asked about the perceived usefulness of their language for teaching the fundamental concepts of programming, on a 7-point Likert scale where 1 was 'very useless' and 7 was 'very useful'. The medians of their answers are given below in Figure 8. All languages, other than Javascript, are reported at about '6' on the 7 point Likert scale. Javascript is reported at '4'.

3.3.4 Reasons for changing language

Respondents were also asked to rank reasons for which they might consider changing language in their course. Figure 9 presents the frequencies for identifying the first rank reason for which participants might consider changing language. 'Pedagogical benefits' accounts for close to half of all first rank preferences (47%) and attracts about 3 times as many nominations as the next most common factor, which is 'Relevant to industry' (15%).

A second analysis was conducted whereby the frequencies for identifying a reason in any of the top three reasons for considering a change of language was considered.

While there were some slight variations to the rank order listings of some of the less common reasons, the first rank remained as 'Pedagogical benefits' (68%) and the next most common factor was again 'Relevant to Industry' (44%).

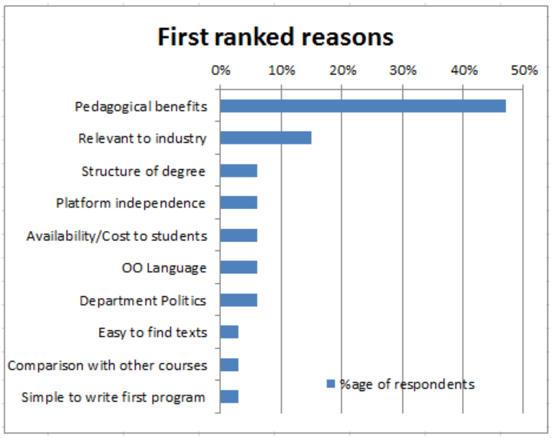

Figure 9: Reasons ranked in top 3 for considering change of language in their course.

3.4 Paradigm taught

Figure 10 presents trends for use of each paradigm over the set of four studies from 2001 to the current. Three aspects are apparent. The continuing dominance, and increasing rise of a procedural approach, the moderately low use of an object orientated approach, and the very low frequency use of a functional approach.

Although not reflected in Figure 6, in 2013, as in 2010, some instructors commented that they had chosen 'procedural' but introduced some aspects of object-oriented programming at the end of the course.

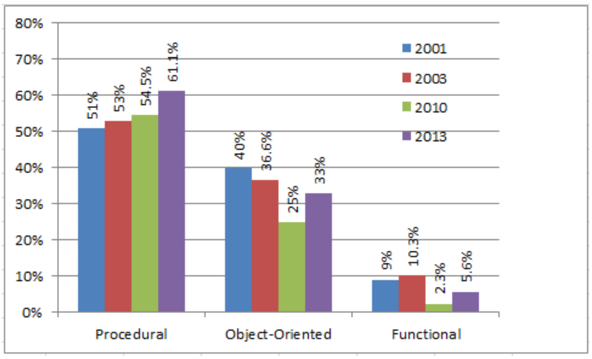

Figure 10: Trends in paradigms taught (4 studies)

Java vs Python: what paradigm is being used? Table 6 shows language (Java or Python) vs. paradigm chosen.

Language	Procedural	Object-Oriented
Java	2	10
Python	10	1

Table 6: Paradigm by Language – Java and Python

An analysis was conducted comparing the language chosen (Python versus Java) by the preferred paradigm used for teaching (Procedural versus Object Oriented). This returned a statistically significant difference (Fisher exact test: $p < 0.001$). Instructors who reported object oriented approaches to their introductory programming courses were predominantly using Java. Conversely, instructors who reported procedural approaches to their introductory programming courses were predominantly using Python.

Despite Python being an object orientated language, instructors are choosing it and then using it in a procedural/ structured way. Java can also be used in this way but it is more difficult unless some additional strategies are included (such as using the BlueJ environment). It appears that the objects-first instructors, also influenced by industry-relevance, are drawn towards Java and the procedural-first instructors (who may also wish to introduce objects later in the course, in the same programming language) are selecting Python.

3.5 Instructor Experience

Participants were asked to indicate their level of experience in teaching introductory programming, and as with the other studies in 2003 and 2010, there was a large range of experience. Four participants had less than 2 years, while two others reported over 30 years of experience. The majority had between 10 and 20 years of experience. This is consistent with the 2010 survey, where participants had a mean of 12.3 years of experience with standard deviation of 7.3 years.

3.6 IDEs and Tools

3.6.1 Choice of IDE/tools

An environment is used in most courses (77.8%) and by most students (69.7%). A significant proportion (22.2%) of the courses surveyed did not use any environment apart from text editors and command-line compilers. This is a similar figure to the 2010 results, and much fewer courses with no environment than in 2001 and 2003.

Of the courses that did use environments, Visual Studio was the most popular IDE at 15.6% of courses. Eclipse was used with 11.1% of courses. Idle and BlueJ followed with 8.9% each, and Netbeans at 6.7%. The remainder of the courses used Alice, Greenfoot and Quincy (all at 2.2% of courses), one 'in house web-based environment", and various other tools. Figure 11 indicates the percentage of students exposed to each of the major environments.

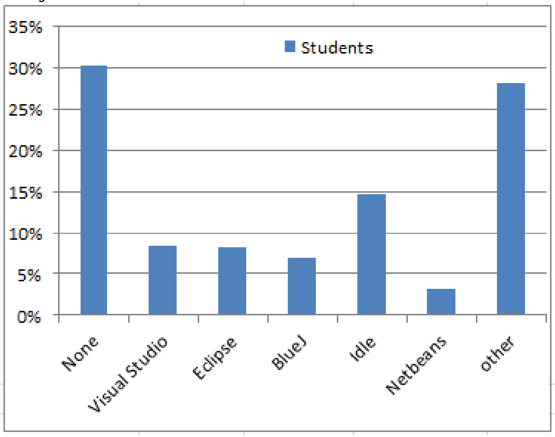

Figure 11: Environments by percentage of students

There are specific relationships between languages and environments, so comparison between different environments is awkward. For example Idle is an IDE with Python which comes bundled with the language, while Netbeans and BlueJ are used with Java. Several environments, such as Visual Studio can be used with multiple languages. Nevertheless, an approach that focuses upon why any specific environment was selected may provide insight into the dynamics and attributes of an environment that motivate their selection and use.

3.6.2 Reasons for choice of environment

The details of which IDE has been used in which language and for which reason is omitted due to space restrictions, but the primary reasons (and motivations) for selection and use of an IDE are presented.

The five most frequent reasons provided for selecting an IDE (not weighted by importance) which each scored at least 80% were: 88% pedagogical reasons, 88% visual cues/debugger, 85% uncomplicated/ease of use, 82% availability/cost to students, 82% student motivation (see Figure 12).

Analysing on the basis of reasons that have been identified as ‘very important’ yields the four most frequent responses where each scored at least 30%: 33% graphical user interface, 30% visual cues/debugger, 30% supports OO paradigm, and 30% pedagogical benefits.

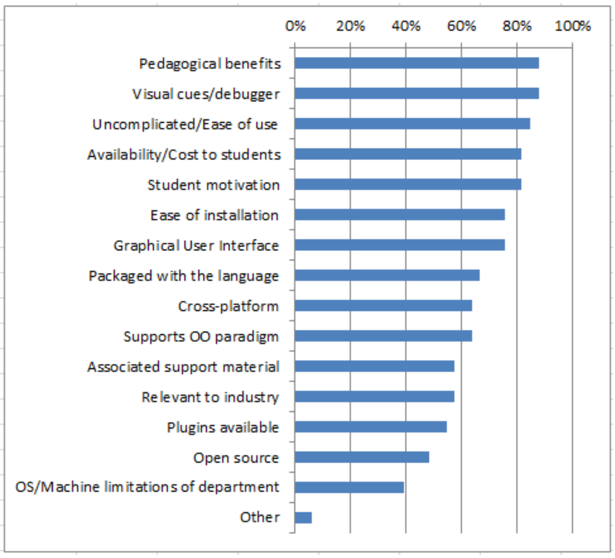

Figure 12: Reasons for choosing environments

3.6.3 Difficulty of environment

Instructors were asked to indicate how difficult they believed the environment was to use for themselves, and for novice students. The results indicate explicitly that instructors perceive students to have more difficulty with an environment than the instructors. This is consistent with the 2010 study indicating the same effect for language. Comparative difficulty is shown below in Figure 13, where 1 is “very easy” and 7 is “very difficult”.

Note that if a student is finding the use of an environment “somewhat difficult” and the language “somewhat difficult”, they may not have the cognitive resources available to problem solve, or develop algorithmic thinking (see Section 3.8)

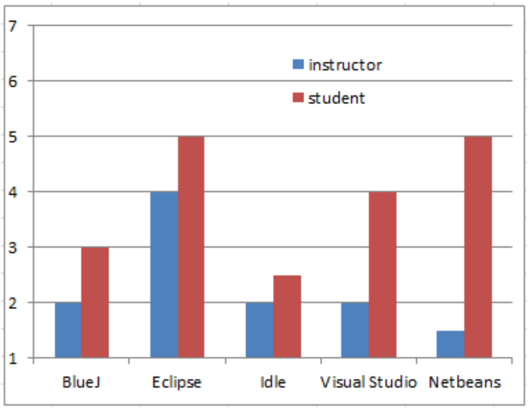

Figure 13: Difficulty of environment (medians)

3.7 Other Aspects of the course

3.7.1 External delivery

Of the 34 courses that answered this part of the survey, the majority (65%) indicated that they do not offer their course via distance or external mode, i.e. a mode where students are not required to attend regular lectures, workshops, labs or tutorials.

3.7.2 Resources given to students

Courses, whether offered externally or not, have various resources provided to students. Below in Figure 14 are the frequencies of resources reported by participants:

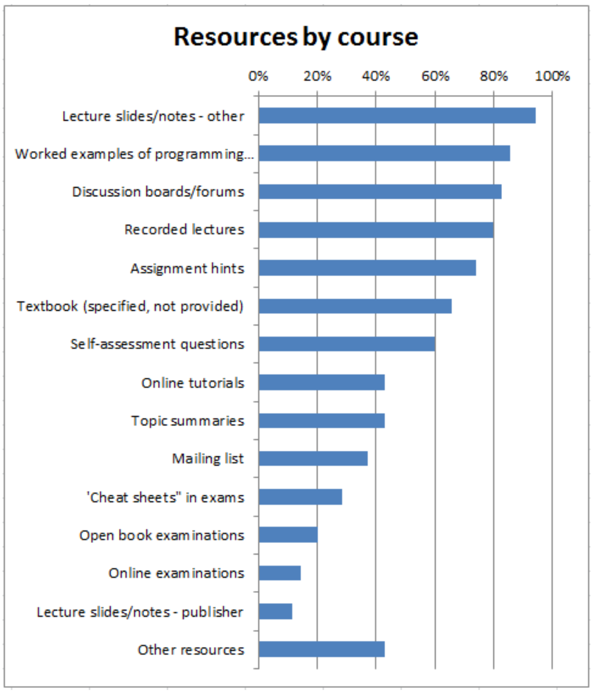

Figure 14: Resources offered to students

3.8 Aims of an introductory programming course

3.8.1 Aim of the course – all languages

Participants were asked what they considered were the three most important aims of an introductory programming course. Answers varied but some themes became apparent. Around half of the participants indicated that nurturing algorithmic thinking was one of the main aims of the introductory course, closely followed by giving student an introductory experience of what it was like to program, and 'learning fundamental concepts'. Interestingly, problem-solving and learning syntax did not appear in the top 3 aims. The themes for which at least two instructors agree are given below in Figure 15.

There were numerous other themes with only one instructor identifying each: basic writing skills, programming proficiency, teaching students to program, basic tools of programming, breadth of paradigms, computing literacy, conceptual models, differentiation of students, real industry-type experience, planning skills, see results, attention to detail, clarity of expression, modification of code, and programming achievement.

3.8.2 Aims of the course – Java vs Python

The reasons given for the choice of environment were compared for courses using Java and courses using Python and the results are presented in Figure 16. Visual inspection indicates that 'algorithmic thinking' was the most important reason for selection of an environment for use with Python, but this had relatively little influence in the selection of an environment for use with Java.

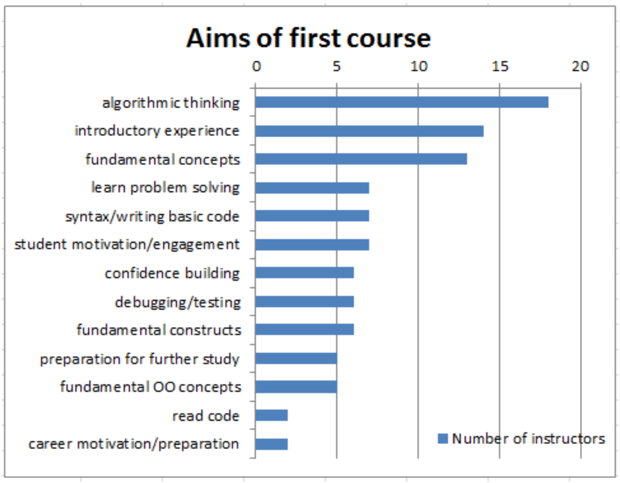

Figure 15: Aims of the introductory course

There were several factors that contributed relatively higher for the selection of an environment for use with Java compared to Python, including:

- fundamental OO concepts,
- fundamental concepts,
- fundamental constructs and
- confidence building.

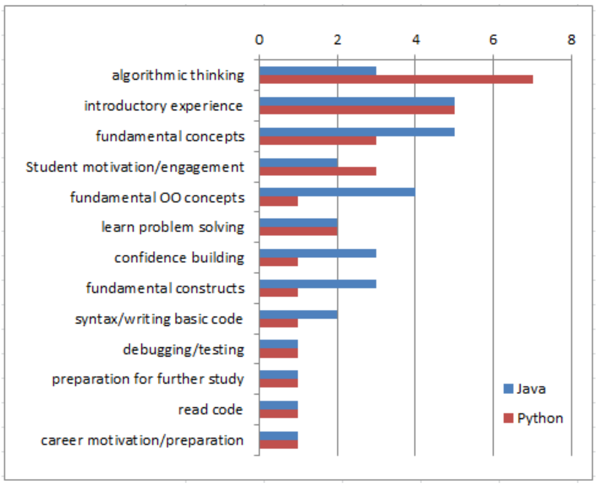

Figure 16: Aims – Java vs Python

4 General Discussion

Two languages currently dominate use in introductory programming courses in Australia and New Zealand. Java has been the most popular language for this purpose since at least 2001 until the present, where it has now fallen to second most frequently used language (as measured by number of students receiving the language). The majority of instructors who use Java have indicated that the primary reasons for their choice of Java have been for its industry relevance and object oriented paradigm.

The language that is now presented to the highest number of students in Australasia (based upon this study) is Python. Python is a relatively new language, and did not even appear in the 2001 and 2003 censuses of introductory programming courses in Australasia, which the current study seeks to broadly repeat. Python has delivered a substantial and sustained impact upon university delivered courses in introductory programming, with Python rising from nothing to top rank in ten years.

The majority of instructors who use Python have indicated that the primary reasons for their choice have been student focussed. This includes pedagogical benefit to facilitate student learning but also other aspects to make life easy for students, through minimising cost and maximising platform independence and access to learning support in the form of textbooks.

The two factors that have been of primary importance for language selection since the 2001 study (de Raadt *et al.* 2002) have been industry relevance and pedagogical benefits. This is still the case, but whereas in 2001 the reason 'pedagogical benefits' was second to industry relevance, it has now risen in relative importance to be the most common reason for language selection. Instructors, when queried about what would motivate them in the future to change language, have indicated a weighting towards pedagogical benefits 3 times more commonly than the second most important factor...industry relevance.

The two factors of pedagogical benefit and industry relevance do not necessarily work together in harmony. A language that is "ideal to industry" will not necessarily be a language that also offers "pedagogical benefits". The vice versa is also true; a language that offers high "pedagogical benefits" will not necessarily be highly relevant to industry. While it may be logically possible for a language to score highly on both of these attributes, it does not appear to be reflected in the reasons currently offered for selection of a language for introductory programming. There is an apparent tension in a dichotomy of Java being selected for OOP and industry relevance, while Python is being selected for ease of student learning and overall uncomplicated experience.

The heightened emphasis given to pedagogical benefits is also demonstrated in instructors' responses regarding selection and use of environments. Although there is a relatively wide range of environments, with sometimes complex relations to a range of languages, the motivations and reasons for selection align broadly to those identified for language selection.

Some of the primary reasons for adopting an IDE are again associated with pedagogical benefits. Indeed, several of the reasons that were highly rated, such as 'GUI' and 'uncomplicated/ease of use' have, for theoretical reasons, been identified through Cognitive Load Theory (Sweller, 1999) to be likely mechanisms to reduce a student's cognitive load, and thus facilitate learning.

There is a clear and continuing trend for instructors of introductory programming courses to be mindful of aspects of their student's experiences in the context of learning computer programming. This always involves aspects of sitting at a computer, using an interface to navigate and operate upon elements of code, syntax and structure.

As a more complete understand of the dynamics of student learning, thinking and program construction is obtained, and as these feed into future computer program interfaces and architectures, it is anticipated that instructors may continue to enhance their focus upon consideration of student (learner) focussed aspects of introductory programming. These may continue to play an important role in the selection of languages and environments for introductory programming and represent areas for further research.

5 Acknowledgements

The authors would like to thank the participants in this study for their involvement.

6 References

Australian Computer Society. (2011). *Australian ICT Statistical Compendium 2011*. [Online]. Available at: http://www.acs.org.au/2011compendium/ [Accessed: 16 February 2012].

Bloch, S. A. (2000). Scheme and Java in the first year. *Journal of Computing Sciences in Colleges*, 15 (5), p.157–165. [Accessed: 9 May 2011].

Dale, N. (2005). Content and emphasis in CS1. *ACM SIGCSE Bulletin*, 37 (4), p.69–73.

Dale, N. B. (2006). Most difficult topics in CS1: results of an online survey of educators. *ACM SIGCSE Bulletin*, 38 (2), p.49–53. [Online]. Available at: doi:Reviewed paper.

DEEWR. (2011). *Australian Jobs 2011*. [Online]. Available at: http://www.deewr.gov.au/Employment/ResearchStatistics/Pages/AustralianJobs.aspx [Accessed: 22 August 2011].

Jenkins, T. (2002). On the difficulty of learning to program. In: *Proceedings of the 3rd annual conference of the LTSN-ICS*, 2002, Loughborough, Ireland, p.53–58.

Mason, R., Cooper, G. and de Raadt, M. (2012). Trends in Introductory Programming Courses in Australian Universities – Languages, Environments and Pedagogy. In: de Raadt, M. and Carbone, A. (eds.), *Proceedings of the Fourteenth Australasian Computing Education Conference (ACE2012)*, 123, January 2012, Melbourne, Australia: Australian Computer Society, Inc., p.33–42. [Online]. Available at: http://crpit.com/confpapers/CRPITV123Mason.pdf.

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

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M. and Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, 39 (4), p.204–223. [Online]. Available at: doi:10.1145/1345375.1345441.

de Raadt, M., Watson, R. and Toleman, M. (2002). Language trends in introductory programming courses. *Informing Science + IT Education Conference*. [Online]. Available at: http://proceedings.informingscience.org/IS2002Proceedings/papers/deRaa136Langu.pdf.

de Raadt, M., Watson, R. and Toleman, M. (2004). Introductory programming: what's happening today and will there be any students to teach tomorrow? In: *ACE'04 Proceedings of the Sixth Conference on Australasian Computing Education*, 30, 2004, Australian Computer Society, Inc., p.277–282.

Sweller, J. (1999). *Instructional design in technical areas*. Camberwell, VIC: The Australian Council for Educational Research Ltd.