PYTHON FOR TEACHING INTRODUCTORY PROGRAMMING: A QUANTITATIVE EVALUATION

Ambikesh Jayal
Department of Computing,
University of Gloucestershire, UK
ajayal@glos.ac.uk

Stasha Lauria
SCISM,
Brunel University, UK
Stasha.Lauria@brunel.ac.uk

Allan Tucker
Stephen Swift
SCISM,
Brunel University, UK
Allan.Tucker@brunel.ac.uk
Stephen.Swift@brunel.ac.uk

ABSTRACT

This paper compares two different approaches of teaching introductory programming by quantitatively analysing the student assessments in a real classroom. The first approach is to emphasise the principles of object-oriented programming and design using Java from the very beginning. The second approach is to first teach the basic programming concepts (loops, branch, and use of libraries) using Python and then move on to oriented programming using Java. Each approach was adopted for one academic year (2008-09 and 2009-10) with first year undergraduate students. Quantitative analysis of the student assessments from the first semester of each year was then carried out. The results of this analysis are presented in this paper. These results suggest that the later approach leads to enhanced learning of introductory programming concepts by students.

Keywords
Teaching programming, object oriented approach, procedural approach, java, python.

1. INTRODUCTION

Teaching programming is a complex task. The task is even more challenging for introductory modules. Failure rates are not marginal (Bennedsen & Caspersen, 2007). One of the common problems shared by Computer Science Departments is the lack of basic programming skills reported by module leaders of courses following first programming courses and how to equip students with better programming skills after the introductory courses. First programming courses typically emphasise the principles of object-oriented programming and design from the very beginning. An alternative approach is based on starting with a more traditional procedural approach first. The evidence from past research seems to suggest that Object Oriented Programming is generally more complex than procedural (Robins et al., 2003). McCane (2009) has demonstrated the effectiveness of Python to teach introductory programming by using qualitative analysis.

In this paper it is therefore suggested to introduce Python (Python Software Foundation, 2010) for the basic (procedural) aspects of programming and then introduce Java (Java, 2010) to focus on object-oriented aspects. The use of Python is expected to both reduce the overhead attached to Java syntax and allow immediate feedback while the students practice with basic instructions (due to the interpreted nature of Python). In other words, at the beginning of the module, novice student are able to devote particular attention to procedural concepts, flow of control, flow of data, etc.

The aim of this paper is to evaluate the success of using Python to introduce basic concepts (loops, branch, and use of libraries). Such evaluation is based on the analysis of student assessments. To measure the success of the proposed method, quantitative analysis of the assessments outcome are presented and discussed. Ehlert (2009) highlights the importance to develop an experimental setting for a fair comparison and the importance to control the variables carefully. The experimental setting used in this paper controls and defines the variables carefully and measures them using clear indicators.

2. LITERATURE REVIEW

Various approaches to teach programming have been summarised in (Lemos, 1979) and (Engel et al., 2001). But there is an ongoing debate in the teaching community over the best approach to teach introductory programming (Lister et al., 2006; Pears et al., 2007). We have found four experimental studies that compare object oriented approach with the traditional procedural approach. The first study by Decker (2003) has found no difference in student performance between the two approaches. The second study by Reges (2006) has found significant gains in student satisfaction and enrolment after replacing the object oriented programming first curriculum with a procedural approach. The third study by Vilner (2007) has found no significant gains in student performance between the two approaches. The parameters used in this research to compare the two teaching approaches are the pass rate of students and grades in questions related to recursion, efficiency of algorithms and designing of classes. The fourth study by Ehlert (2009) has found no significant gains by changing the object-oriented programming first approach with object oriented programming later approach.

According to Ehlert (2009), different pegagogical dimensions complicate the analysis of different approaches to teach programming. Bruce (2005) argues that there is a need for more experimental studies to examine the different approaches to teach programming.

3. METHODOLOGY

Within this paper Python assessment submissions are compared with Java ones. The Python and Java assessments are very similar and therefore comparisons should give a reliable measure. The aim of the comparison is to quantitatively measure a student’s ability to master basic programming concepts when Python is used instead of object-oriented Java. Both assessments consisted of students having to implement a program (See below Section 4 for further details).

The following three comparisons have been carried out: grades, programs with bugs and frequency of keywords. The grades of the submitted program are a measure of success of the student ability to implement programs. Bugs are interpreted as a measure of their overall understanding of programming. Frequency values of keywords are interpreted as a measure of the familiarity with basic concepts of programming. The four keywords used in the frequency measure are “`if`”, “`for`”, “`while`” and “`import`” (in combination with the use of “`random`” class). They are indicators of student’s use of conditional, loops and import (the random library) statements. Occurrences of these keywords in commented sections of the code have been discarded. No distinction has been made between correct and incorrect use of the statements in the frequency measurement.

Two different academic years have been used to collect data. During the 2008-09 academic year, the approach to emphasise the principles of object-oriented programming and design using Java from the very beginning is followed. During the 2009-10 academic year, the approach to first teach the basic programming concepts (loops, branch, and use of libraries) using Python and then move on to oriented programming using Java is followed. In both cases, students were assessed after approximately ten weeks of teaching. Java assessments are from 2008-09 academic year, whereas Python assessments are from the 2009-10 academic year. Also all students in a cohort have (some) similar background since they have to fulfil similar requirements to be accepted. Some may have a better proficiency than other but this is true in equal measure for both cohorts. The research is based on the assumption that students cohort for the academic year 2008-09 and 2009-10 are of similar academic level. This assumption is supported by the fact that the admission criteria for both the years were the same. No student was in both cohorts. Table 1 summarises the student’s cohort.

Academic Year	Language	Assessment point	Number of students
2008-09	Java	After ten weeks of teaching	157
2009-10	Python	After ten weeks of teaching	185

Table 1: Summary of student’s cohort

4. MODULE OUTLINE

Both 2008-09 and 2009-10 modules during the first ten weeks emphasize mastery of basic skills. In particular, lectures and lab tutorials/exercises have focused on loops, conditionals, arrays/lists and use of libraries/packages (in particular the ones to generate random numbers). BlueJ (BlueJ, 2010) was used in the 2008-09 module as a tool to practice with Java, whereas in 2009-10 the standard Python tools have been, used (i.e. both the graphical user interface and the command line applications). Approximately, the same amount of time has been dedicated to each topic during both years.

Students are assessed for their ability to implement programs. Criteria for the assessments include: generalizability, complexity, coverage of loops, conditionals, etc. Moreover, the ability to produce a program that could compile and run without errors has been considered a requirement for a pass. However, for both Java and Python data, small typos (a missing semicolon, etc.) have not been considered bugs (even if the submitted program did not run correctly). That is, they have not been graded as a fail grade, although they have been graded as just above a pass. To assess the style of the submitted program the source code was inspected.

Students have an hour to complete the task and they can use the same environment as the one they used during their lab tutorials (BlueJ, Python, etc.). That is, during the test they can implement, run and debug their programs. They have access to both the standard Java/Python documentation and to the module resources (lecture notes, lab examples, etc.). Tasks are kept as similar to one another as possible in both assessments.

5. RESULTS AND DISCUSSION

Tables 2 and 3 summarise the results of the quantitative analysis.

Columns 3 to 6 in Table 2 show the number of occurrences of programs where respectively at least once a conditional, a `for` loop, a `while` loop and the `import` of the random package has occurred. The second column (Total Students) indicates the total number of assessments and the last column indicates the occurrences of a program that could not run without performing some debugging first.

	Total Students	If	%	for	%	While	%	random	%	bugs	%
Python	185	67	**36**	81	**44**	27	**15**	97	**52**	51	**28**
Java	157	36	**23**	1	**1**	8	**5**	42	**27**	90	**57**

Table 2: Key Indicators. Occurrences and percentages for each key indicator. The Total column indicates the total number of assessments.

	Total Students	A	%	B	%	C	%	D	%	E	%	D(40)	%
Python	185	6	**3**	15	**8**	22	**12**	121	**65**	21	**11**	36	**19**
Java	157	6	**3**	10	**6**	22	**14**	42	**27**	78	50	19	**12**

Table 3: Grades. Occurrences and percentages for each key indicator. The Total column indicates the total number of assessments.

Table 3 shows the grade distribution. The use of loops, conditional, etc. to achieve the given task was part of the marking criteria. For example, a grade A requires the correct use of conditional and the correct use of loops and the correct use of libraries in the program implemented. The last column (D(40)) shows the number of occurrences of programs that did not run without debugging but where bugs have been nevertheless considered of minor entity(and therefore graded as just a pass).

The results presented in Table 2 show that there is evidence of a significantly increased use of all the four markers when Python is used as programming language. Moreover, the occurrence of bugs is lower with Python than with Java. The results presented in Table 3 show that grades distribution shows a higher percentage of programs able to meet the minimum requirements for a pass when Python has been used.

In order to statistically validate our results, Two sample t-test was conducted on the four markers (for "`if`", "`for`", "`while`" and "`random`") shown in Table 2. R (The R Foundation, 2010) was used to carry out this statistical test. The confidence interval was taken as 0.95. The p-value when variance was assumed to be equal was 0.043 and when the variance was assumed to be unequal was 0.048. As the p-value is less than 0.05, the difference between the samples is statistically significant. So the number of key indicators used by students statistically differs when Python was introduced as compared to the Java only approach. This implies that the Python first approach had a positive effect on the grades.

These results imply that our approach to introduce Python for the basic (procedural) aspects of programming and then switch to Java to focus on object-oriented aspects enhanced students understanding of the introductory programming.

6. CONCLUSIONS

This paper presents experimental results from real classroom comparing two different approaches to teach introductory programming. The first approach is to emphasise the principles of object-oriented programming and design using Java from the very beginning. The second approach is to first teach the basic programming concepts (loops, branch, and use of libraries) using Python and then move on to oriented programming using Java.

The first teaching approach using Java has been followed for the academic year 2008-09 and the second teaching approach using Python has been followed for the academic year 2009-10. The student assessments towards the end of first semester in each academic year have been used to carry out the experiments described in this paper. Three indicators namely grades, programs with bugs and frequency of keywords have been used for comparison. The grades of the submitted program are a measure of success of the student ability to implement programs. Bugs are interpreted as a measure of their overall understanding of programming. Frequency values of keywords are interpreted as a measure of the familiarity with basic concepts of programming. The four keywords used in the frequency measure are "`if`", "`for`", "`while`" and "`import`" (in combination with the use of "`random`" class).

The experimental results discussed in this paper show a measured positive (increase) in all the three indicators in favour of the second approach which is to first teach the basic programming concepts (loops, branch, and use of libraries) using Python and then move on to oriented programming using Java. The use of Python could facilitate the mastering of basic concepts such as loops, branch, and use of libraries for novice students. A possible explanation could be that with Python students can focus on the crucial basic issues without being distracted by the overheads. Moreover, the complexity of the programs used during their practice could be tailored in a more accurate way to their level of proficiency.

One limitation of this research is that it considers only one case study. Such case studies require considerable amount of time as they have to fit in with the academic calendar and hence we could only consider one case study. We hope that similar case studies will be replicated by other institutions as well. The results from our case study are positive and statistically significant in favour of our proposed approach which is to first teach the basic programming concepts (loops, branch, and use of libraries) using Python and then move on to oriented programming using Java. We hope that our research will encourage debate over different approaches to teaching introductory programming and in turn will lead to adoption of better teaching methodologies.

Also this research focuses only on the quantitative analysis of the student assessments and does not elicit the views of the students about their learning experience. So an interesting direction for future work would be to collect and analyse the qualitative data from the students about their perspective.

This research takes into account the performance of students up until the first semester. We are analysing the performance of students over the whole academic year and will be publishing it at a later stage.

7. References

Bennedsen, J., Caspersen, M.E. (2007), Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39(2), pp. 32–36.

BlueJ, (2010), BlueJ - Teaching Java - Learning Java, http://www.bluej.org/ (01 Dec 2010)

Bruce, K.B. (2005), Controversy on how to teach CS 1: a discussion on the SIGCSE-members mailing list, *ACM SIGCSE Bulletin*, 37(2), pp. 111-117

Decker, A. (2003), A tale of two paradigms, *Journal of Computing Sciences in Colleges*, 19(2), pp. 238-246

Engel, G., Roberts, E. (2001), Computing Curricula 2001 Computer Science: final report, *IEEE Computer Society and Association for Computing Machinery*, 28, pp. 2004

Ehlert, A., Schulte, C. (2009), Empirical comparison of objects-first and objects-later, *ACM Fifth international workshop on Computing education research workshop*, pp. 15-26

Java, (2010), Oracle Java Technologies, http://www.oracle.com/us/technologies/java/index.html (01 Dec 2010)

Lemos, R.S. (1979), Teaching programming languages: A survey of approaches, *ACM SIGCSE Bulletin*, 11(1), pp. 174-181

Lister, R., Berglund, A., Clear, T., Bergin, J., Garvin- Doxas, K., Hanks, B., Hitchner, L., Luxton-Reilly, A.,Sanders, K., Schulte, C., and Whalley, J. L. (2006), Research perspectives on the objects-early debate. *ACM SIGCSE Bulletin*, 38(4), pp. 146-165

McCane, B. (2009), Introductory programming with python, *The Python Papers Monograph*, 1, pp. 1-18.

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), pp. 204-223

Python Software Foundation, (2010), Python Programming Language, http://www.python.org/ (01 Dec 2010)

Robins, A., Rountree, J., Rountree, N. (2003), Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13(2), pp. 137-172

Reges, S. (2006), Back to basics in CS 1 and CS 2, *ACM SIGCSE Bulletin*, 38(1), pp. 293-297

The R Foundation, (2010), The R Project for Statistical Computing, http://www.r-project.org/ (01 Dec 2010)

Vilner, T., Zur, E., Gal-Ezer, J. (2007), Fundamental concepts of CS1: procedural vs. object oriented paradigm-a case study, *ACM SIGCSE Bulletin*, 39(3), pp. 171-175