Abstract

This paper presents a study aimed at examining the novice student answers in an introductory programming final e-exam, to identify misconceptions and types of errors. Our study used the Delphi Concept Inventory (DCI) to identify student misconceptions and Skill, Rule, and Knowledge (SRK) based errors approach to identify the types of errors made by novices in Python programming. The students' responses to each question were scrutinized by using the DCI, heuristic-analytic theory and Neo-Piagetian theory of cognitive development for qualitative data analysis. Moreover, the motivation for this exploratory study was to also address the misconceptions that students held in programming, and help educators to redefine the teaching methods to correct those alternative conceptions. Student misconceptions were spotted in list referencing and inbuilt functions in Python. In a further quantitative analysis the study found that students who had misconceptions, made knowledge errors and failed to complete the coding tasks. Surprisingly, and coincidentally, it was identified that only a few students were able to write code related to mathematical problems.

Keywords: Delphi's CI, Programming courses, ViLLE, E-Final exam, Student misconceptions, Taxonomy of errors

Introduction

This paper presents a study to identify taxonomy of misconceptions and errors from analyzing university novice students' programming e-final exam answers. A misconception is an erroneous belief, which is not true or valid. Misconceptions arise due to various reasons such as preconceived notions, nonscientific beliefs, and vernacular alternative conceptions (Moore;Abella;& Boggs, 1997). For example, some students have misconceptions about learning computer programming that, "only math experts can program a computer", which is not completely true (Perry & Miller, 2013). Misconceptions in learning mathematics, science and arts are quite common and a normal part of the learning process. Student misconceptions in learning occur due to teachers, textbooks that contain incorrect information and errors, student's prior learning and self-developed ideas that are scientifically inaccurate, linguistic transfer, and lack of knowledge of the subject (Moore;Abella;& Boggs, 1997; Nakiboglu & Tekin, 2006; Black & Lucas, 1993; Simanek, 2008). Student misconceptions affect learning and impede students from acquiring new concepts in computer programming. To date, several studies have been conducted to identify student misconceptions of programming. Notably, novice misconceptions of programming are one of the biggest concerns to programming educators and students (Kaczmarczyk;Petrick;East;& Herman, 2010; Özdener, 2008; Bringula;Manabat;Tolentino;& Torres, 2012).

In addition, novice programming students often make different kinds of errors due to various reasons such as, lack of attention, misjudgment, misconception and strong habit intrusions (Kaczmarczyk;Petrick;East;& Herman, 2010; Bringula;Manabat;Tolentino;& Torres, 2012). Notably, study of taxonomy of novice programming errors has long been of interest to researchers and educators and there have been many studies done on novice programmer errors to identify the type and cause of those errors (Bringula;Manabat;Tolentino;& Torres, 2012; McCall & Kölling, 2014). However, learning to program or to write code without misconceptions and errors is often believed to be difficult for novice programming students at university level. Educators are always looking for effective teaching methods to reduce novice misconceptions of programming and errors. Notably, novice programming misconceptions and errors are frequently analyzed to improve their code writing and debugging skills (Kaczmarczyk;Petrick;East;& Herman, 2010; Özdener, 2008; Bringula;Manabat;Tolentino;& Torres, 2012; Spohrer & Soloway, 1986; Teague;Corney;Colin Fidge;Ahadi;& Lister, 2012; Lahtinen;Ala-Mutka;& Järvinen, 2005).

Moreover, many teaching techniques and educational technologies have been developed to alleviate some of the problems in computer programming learning (Muller M. , 2006; Salleh;Mendes;& Grundy, 2011; Tiwari;Lai;So;& Yuen, 2006; Uysal, 2014; Lee, 2005) errors identified in practical sessions are frequently examined, and for which solutions are often proposed in the literature, we observed that misconceptions and specific types of errors still occur when students answer questions, particularly when writing program solutions in the final programming e-exam.

It is important for educators to identify programming related misconceptions and errors, devising ways to address them, and to enhance their teaching methods and quality. Moreover, identifying the novice programming errors would help educators to improve the teaching and learning of computer programming (Marceau;Fisler;& Krishnamurthi, 2011; Ebrahimi;Kopec;& Schweikert, 2006). Goldman et al. developed the Delphi Concept Inventory (DCI), which contains a list of topics that are important and difficult for students to learn in programming fundamentals subject (Goldman, ym., 2008). This CI was defined based on the decisions supplied by panel of experts who had taught computing courses frequently and published textbooks or pedagogical articles (Goldman, ym., 2008). There have been studies that used DCI to identify the programming misconceptions of programming (Kaczmarczyk;Petrick;East;& Herman, 2010). However, none have yet examined the student answers in programming final exams (summative assessments), which reflect what students have actually learned and misunderstood, as well as the skills gained throughout the entire study period. Summative assessment at the end of a study period serves an important formative purpose for further study as misconceptions in the former period will have a compounding impact in future study periods.

Thus, the goal of this research is to refine our teaching methods from learning novice programming misconceptions and identifying skill, rule and knowledge based errors. Towards this aim, this paper addresses the following research questions.

- What types of programming misconceptions might be identified within solutions presented to coding questions in e-exams?
- What types of programming errors are made in the final e-exam and why are such errors made?

In order to find answers to these research questions, we collected and analyzed introductory programming final e-exam data.

This paper is organized as follows. Section II presents a literature review of studies conducted around misconceptions, and errors and their significance in relation to final exam results. Section III, describes the research methodologies used in this study to find the answers for our research questions. Section IV describes the findings of the study. Finally, Section V presents conclusions, future work intentions, and limitation of the study to proceed for the next stage of the research, which is to produce an enhanced and innovative approach to teaching introductory programming course.

Literature review

Student Misconceptions

There has been much research done on student misconceptions in learning science, mathematics, arts, and concluded that student misconceptions act as barrier to learning (Goldman, ym., 2008; Almstrum, ym., 2006; E.Byrd;McNeil;Chesney;& G.Matthews, 2015; Hestenes;Wells;& Swackhamer, 1992). Sirkiä quoted "the less you know the subject, the more severe the misconception can be" (Sirkiä, 2012). For example, students who interpret equal sign only as arithmetic specific may suffer learning the preliminary level algebra where the equal sign has different interpretations [3].Technology-supported learning fosters students' knowledge acquisition. On the other hand, it increases student misconceptions. Wendt et al. tested the effect of on-line collaborative learning in alignment with students' science misconceptions and concluded that "students who participated in collaborative activities in the on-line environment had more science misconceptions than students who participated in collaborative activities in the traditional classroom" (L.;Wendt;& Rockinson-Szapkiw, 2014). Students come to class with various experiences and knowledge. Students' prior understanding about the topic plays a vital role in learning. However, educators at the university level often ignore the engagement of student's prior understanding which leads them as procedural based problem solvers (Vigeant;Prince;& Nottis, 2014). Moreover, once the misconception has been formed it is very hard to change (Eggen & Kauchak, 2004). So, it is important for educators to know students' prior understanding about the topic to identify if students have alternative conceptions (Kaczmarczyk;Petrick;East;& Herman, 2010).

Misconceptions in Learning Computer Programming Languages

There has been considerable research done on student misconceptions in learning computer programming languages such as BASIC, C, Java, and concluded that "bugs are likely to arise" when students have misconceptions about programming language constructs (Kaczmarczyk;Petrick;East;& Herman, 2010; Özdener, 2008; Bringula;Manabat;Tolentino;& Torres, 2012). Sorva prepared a catalogue of novice misconceptions about introductory programming courses extracted from various research articles (Sorva, 2012). It seems novice programming related misconceptions occur due to confusion about the computational model. That is, how the computer program is executed; and how the variables and control structures are linked (Sorva, 2012). Notably, students who fail to see the difference between syntax and semantics make syntax errors and fail to trace the code linearly (Kaczmarczyk;Petrick;East;& Herman, 2010; Spohrer & Soloway, 1986). Besides, Özdener found that both high school and university students have similar kinds of misconceptions about time-efficiency in algorithms. They believe code which contains less syntax, fewer variables, and commands require less execution time. Furthermore, these kinds of misconceptions would affect students' programming skills (Özdener, 2008). Hence, knowledge of students' conceptual misunderstandings in programming would help educators to improve student learning by addressing such misconceptions.

Concept Inventory in Learning Computing Courses

Student misconceptions can be identified by educators through testing their prior understanding by developed concept inventories and defined assessment tasks (Almstrum, ym., 2006; E.Byrd;McNeil;Chesney;& G.Matthews, 2015). Concept Inventories (CI) are a form of diagnostic tests widely used in education to examine students' misconceptions in learning arts, mathematics, science, and computing courses (Goldman, ym., 2008; Almstrum, ym., 2006; Hestenes;Wells;& Swackhamer, 1992; Treagust, 1988). These CI-based test results are used by educators to redefine teaching strategies and to help students to reconstruct correct conceptions. The first concept inventory, the Force Concept Inventory was a test designed by physics educators, to identify student misconceptions of Newtonian physics in physics (Hestenes;Wells;& Swackhamer, 1992). Goldman et al. developed the Delphi Concept Inventory (DCI) and addressed important and difficult concepts of introductory computing courses. Moreover, DCI is a significant assessment approach to measure the programming concepts that students are struggling with and what specific programming misconceptions they hold (Goldman, ym., 2008). Kaczmarczyk et al. used a Delphi CI to reveal programming misconceptions held by novices in Java (Kaczmarczyk;Petrick;East;& Herman, 2010). They analyzed the *verbatim* transcripts of *audio* & video recorded *interviews* and found that students commit mistakes due to programing misconceptions. However, they did not conduct any summative assessments to identify if students have programming misconceptions when writing program solutions.

Taxonomy of Errors

People make mistakes in daily life for various reasons. Errors are said to be made due to lack of knowledge, carelessness, and misjudgment (Meister, 1989; Sutcliffe & Rugg, 1998). Knowledge errors occur when a person fails to achieve the task due to lack of knowledge. Skill based errors or "slips and lapses" occur due to strong habit intrusions. This kind of error happens when a person misses a step in isolation sequence or presses the wrong key. Moreover, this kind of mistake often even made by experienced, highly trained and skilled persons (James, 1990). For example, use of "/" instead of "%" to get the remainder value from integer division in a program, by experienced programmers, can be considered a skill-based error. Rule-based errors occur due to misapplication of a rule (James, 1990; http://patientsafetyed.duhs.duke.edu/module_e/types_errors.html, 2014; Embrey, 2005). Furthermore, these types of errors occur when people make unwarranted assumptions to solve a problem (Davidson & Sternberg, 2003). For example, the application of string handling functions to numerical values in the program can be considered a rule-based error (Bringula;Manabat;Tolentino;& Torres, 2012).

Novice Errors in Computer Programming

Several research studies have been conducted on novice programming errors and have concluded that many novices make knowledge type errors (Bringula;Manabat;Tolentino;& Torres, 2012; Lahtinen;Ala-Mutka;& Järvinen, 2005; Butler & Morgan, 2007). Bringula et al. conducted a laboratory study to predict the errors committed by novice Java programmers and concluded that "a knowledge type error is one of the consistent predictors of novice Java programmers" (Bringula;Manabat;Tolentino;& Torres, 2012). Lahtinen et al. reported that novices make knowledge errors in defining loops, and passing parameters and semantics due to lack of understanding in language construction, use of semantics and poor understanding of programming concepts (Lahtinen;Ala-Mutka;& Järvinen, 2005). Mathew et al. concluded that for beginners it is hard to define algorithms and methods, understand the difference between syntax and semantics, and the scope of variables inside nested loops (Butler & Morgan, 2007).

Final Exam

A final exam is a summative assessment instrument, which typically takes place at the end of a course of study. (Summative testing may also be conducted at various intervals in the study period as a way to monitor progress of students at strategic milestones.) A final exam is one of the core assessment tasks at educational institutions to assess student academic performance. Final exam data are frequently used at higher education to assess student learning and performance, and to evaluate course learning outcome (Reeves, 2006; Richard O. Mines, 2014). Moreover, the results of final exams indicate what has actually been learnt, and what skills have been attained, by the end of the study period. Besides, final results give great insight to teachers (Santiago & Benavides, 2009). There has been research that used exams as a tool of measurement to identify students' errors and misconceptions (Nakiboglu & Tekin, 2006; Olsen, 1999; Movshovitz-Hadar;Zaslavsky;& Inbar, 1987). Olsen used an error analysis approach and examined Norwegian students' English final exam papers to identify the grammar and vocabulary errors committed by lower secondary school students (Olsen, 1999). Hadar et al. examined high school students' written answers of mathematics matriculation exam and generated a system of six error categories (Movshovitz-Hadar;Zaslavsky;& Inbar, 1987). Nakiboglu et al. examined the Turkish high school students' multiple choice test answers, and concluded that pre-university students have a series of misconceptions in learning nuclear chemistry (Nakiboglu & Tekin, 2006).

Moreover, Simanek pointed that exam questions allow students to use certain student misconceptions to get "the right answer" in the exam (Simanek, 2008). So, the cited studies endorse that it is possible to identify the taxonomy of misconceptions and errors by scrutinizing students' written work. Moreover, to the best of the authors' knowledge, although novice programming misconceptions are frequently investigated through formative tests, none has attempted to identify beginners programming misconceptions through summative tests notably by examining student answers of tests.

Research Method

This study analyses the answers written for final programming exam in 2014 by students from different disciplines at university. There were 69 students enrolled in the course. However, only 39 students attended the final exam. The final exam was conducted online at the end of a course of study. This exam was a closed book and three hours in duration. The final exam contained two multiple choice questions, two code tracing questions, and six code writing questions, covering the syllabus topics and reflecting the Delphi CI given in the Table A1. The list of final programming exam questions is provided in the Appendix.

Qualitative Analysis

The primary purpose of this analysis was to reveal misconceptions held by novice students in their introductory programming (Python) course. The analysis protocol list for this study was derived via the

Delphi process. A Delphi Concept Inventory has thirty-two concepts that are important and difficult in programming fundamentals courses. These key concepts were identified through the Delphi process (Goldman, ym., 2008). We derived 10 topics from those as key themes based on the topics covered in the programming language final exam. Table A1 shows the key topics derived from Delphi CI for qualitative analysis. The Python programming language does not support array data structures but it has a higher order list data structure instead.

Table A1

We collected Python programming final exam data via the ViLLE collaborative tutorial software tool. ViLLE is software used in introductory programming course to support technology enhanced programming classes (ViLLE). ViLLE is mainly used to provide lecture materials, homework, lab exercises, and class work for programming courses. All novice programming exercises were made available via ViLLE for students to practice and grade their submitted answers automatically (ViLLE). We analyzed the student answers for multiple choice questions, code tracing questions, and code writing questions. Table A2 shows the list of questions with details that were examined for qualitative analysis.

Question 1a was a multiple choice question. Its purpose was to check if students understood the basic syntax and semantics of Python programming. Questions 1c and 1d were prepared to measure if student understood the loop and decision statements and able to trace the code in Python programming. The final six coding questions numbered from 2 to 7 tested code writing skills covering the syllabus topics taught in the introductory programming course. Question 1b was omitted due to its irrelevance for this study.

Table A2

We analyzed 39 students’ answers in the final exam. The students’ responses to each question were scrutinized by using the DCI, heuristic-analytic theory and Neo-Piagetian theory of cognitive development for qualitative data analysis as described in the Kaczmarczyk et al.

(Kaczmarczyk;Petrick;East;& Herman, 2010), Kryjevskaia et al. (Kryjevskaia;Stetzer;& Grosz, 2014), and Teague et al., (Teague;Corney;Colin Fidge;Ahadi;& Lister, 2012), respectively.

1. All of the student answers were compared with solutions to check if the student committed any mistakes due to lack of knowledge, confusion, assumption, and or used certain misconception to get right answers for exam questions.
2. We classified the findings based on the CI- key topics listed in the table A1 to seek for answers to the questions given here.
   a) Does the code contain skill/rule/knowledge based errors?
   b) Does the student have misconception about the coding questions and or topics that he/she learned?
3. The coding mistakes and our findings were grouped to identify the cognitive stage of novices based on Neo-Piagetian theory to develop the types of errors, and if any other key themes emerged from those.

For example, Figure B1 shows a screenshot of a student answer for Question 5.

Figure B1 Student’s answer versus sample solution

As shown in Figure B1, each student’s answer was verified against a sample solution and the details given in the question to identify whether or not the student had any misconceptions or had made any errors. In this example, the student has answered the question correctly but has failed to follow the details given in the question, that is, “print the largest fraction”. So, we checked the student’s answers to the other questions that had similar details to identify if student committed similar error, however not. So, we confirmed that the student had committed the skill based error due to “slips and lapses”.

Quantitative analysis

The secondary purpose of this study was to identify the types of errors made by students. In quantitative analysis, we counted the number of mistakes that were identified in the qualitative analysis by statistical methods. That is, all types of errors were counted and the average calculated based on the number of students. Table A3 shows the types of errors that were prepared based on the literature review and the errors detected from students' answers.

Table A3

Results and Discussion

Results of Qualitative Analysis

Table A4 shows the results that were generated from our qualitative analysis.

Table A4

Table A5

From the results it was identified that students had confusion about the application of inbuilt functions, and failed to see a type of data that was required. Moreover, students did not understand the difference between string and numeric data type values syntactically and semantically. For example, Figure B2 shows a screenshot of a student answer for Question 3.

Figure B2 Student's answer versus misapplication of comparison operator

In Question 3, students were asked to find the longest string from procedure *outputLongest(s1, s2, s3),* which receives three strings as arguments. Seven students out of the 39 students applied the comparison operators on string type variables to find the longest string assumed those operators can be applied on strings. This kind of misconception occurs when students attempt to integrate the new and previous understandings. Moreover, this confusion occurs if the new concept is much more similar to previously learned concept (Alhalifa, 2006).

Hristova et al. identified that novices had misconceptions in defining methods and calling parameters (Hristova;Misra;Rutter;& Mercuri, 2003). However, our findings did not support (refer table A5) their statements fully. 67% of students defined the methods correctly though a few had problems in using the return statement in functions. Moreover, 68% of the students understood the application of functions, although 8% of the students passed the wrong argument/value as the return value, which may be considered as semantics error (refer table A5). For example, Figure B3 shows a screenshot of a student answer for Question 5.

Figure B3 Student’s answer versus skill based error

In Question 5, the incomplete code was given including comments as specifications to write the missing part of the code as answer. Nearly one third of students did not read or did not follow or ignored the comments preceded with # as tips.

Tracing loop execution needs cognitive skills and students should have that ability to trace the code linearly. Our results reflected that a few students failed to trace the code linearly due to the lack of knowledge in “how the looping technique works”. For example, Figure B4 shows a screenshot of a student answer for code tracing Question 1c.

Figure B4 Student’s answer versus code tracing skill

Surprisingly, 83% of the students (refer table A5) who answered incorrectly for tracing the code Question 1c were able to write program solutions that require looping statements. It seems students used mental shortcut approach called availability heuristic technique to get the right answer (Kryjevskaia;Stetzer;& Grosz, 2014). That is, solving the problem based on individuals prior knowledge, experience and belief and therefore more likely to be correct.

According to Delphi CI, reference to arrays versus array elements, identifying off by one index errors which occur when a student using less than or equal to where is less than, and declaring and manipulating arrays as important and difficult topic for students (Goldman, ym., 2008). Notably, Boulay identified that students had misconceptions with array subscripts and dimensions (Boulay, 1986). We also had similar results with students confused about lists and list indices. Moreover, 13% of the students used negative numbers as indices to read or display the values from lists.

Apart from those listed in the Table A4, there were also a few other misconceptions identified in our study that are similar to Bruce's list of Python programming misconceptions (Bruce, 2015). They are: confusion in variable declaration, index out of bound errors in lists, inappropriate use of comparison operators, confusion between a key and its corresponding associated value in the list.

Results of Quantitative Analysis

The same set of exam question data was used for our quantitative analysis to enumerate the types of errors made by students in the final exam. The results of quantitative analysis are shown in Figure B5.

Figure B5 Misconceptions and type of errors committed in the final exam

As Bringula et al. identified (Bringula;Manabat;Tolentino;& Torres, 2012) that it is common for learners who are inexperienced in coding to most often make knowledge type errors. Statistical results of our study support Bringula et al.'s conclusion with, on average (see Figure B2), 69.2% of students failing to completely answer the code writing questions, due to lack of knowledge in the topics reflected in these questions. Remarkably, a few students made rule based errors due to "assumption based confusion" around the use of library functions in coding their answers. Notably, these rule based error findings strongly support "The psychology of problem solving - Education Psychology literature - assumption based attitude or this kind of cognitive behavior is ubiquitous" (Davidson & Sternberg, 2003). That is, it is a common tendency of humans to automatically bring their prior knowledge as a tool, assuming that it would solve the problem and this kind of attitude cannot be turned off easily. In addition, due to "slips and lapses" only 5.1% of students committed skill-based errors. For example, a few students ignored the instructions given in the questions, and included unnecessary commands in the program (refer Figure B6).

Figure B6 Student's answer versus ignorance of directives

It was accepted that some knowledge of mathematics would help to learn "how to program". Moreover, programming is often concerned with mathematical concepts of logic. So, math based coding questions were frequently asked in the introductory programming courses to test students' problem solving skills.

This study analysis also explored that novices of programming struggled in writing code for math related questions 6 and 7 (refer Appendix). Nearly 66% of students did not do well in the mathematical problems based questions though explained and allowed to surf the Internet to seek for more details during the exam hours. A Neo-Piagetian theory of cognitive development stated that students who are at the concrete operational stage struggle to write large programs with partial specifications, although they can write small programs from well-defined specifications (Teague;Corney;Colin Fidge;Ahadi;& Lister, 2012).

Conclusion and Future work

This study analyzed the student answers of E-final exam questions to reveal misconceptions of novice programming learners. The qualitative data analysis based on the Delphi Concept Inventory of programming misconceptions and applied heuristic-analytic theory and Neo-Piagetian theory of cognitive development research was conducted to identify students' alternative conceptions of programming. In a further quantitative analysis it was found that students who had misconceptions, made knowledge errors and failed to complete the coding tasks. However, relatively few students made rule and skill based errors. Besides, it was accidently identified that more than two-thirds of all students failed to answer the mathematical task-based coding questions in the exam, which might be considered by educators in refining teaching methods to foster students' problem solving skills. This issue can also be connected with math anxiety for future research work.

The motivation for this explorative study was to address the misconceptions that novice students held in programming, and help educators to redefine the teaching methods to correct those misconceptions. Further research could be conducted to measure the correlation between type of errors committed by student and student misconceptions. Another way to explore is to study the gender difficulties and misconceptions in programming courses. That is, "Are there gender effects on student misconceptions of programming?" Finally, this study will also be implemented to different courses, institutions, countries and with different student populations to explore student misconceptions.

Like all research, this study has several limitations and not free from its weakness. First, the sample size of this study is not adequate enough to generalize our findings. Second, this research did not focus on students’ prerequisite skills, test anxiety, how student think during exam hours, and influence of other assessment tasks’ such as homework, project work results. Finally, students’ answers to the questions may be out of researcher’s control to confirm the findings strongly.

References

Abu-Oda, G. S., & El-Halees, A. M. (2015). Data Mining in Higher Education: University student dropout case study. *International Journal of Data Mining & Knowledge Management process, 5*(1), 15-27.

Afzaal H. Seyal, Y. S., Matusin, M. H., Siau, H. N., & Rahman, A. A. (2015). Understanding Students Learning Style and Their Performance in Computer Programming Course:Evidence from Bruneian Technical Institution of Higher Learning. *International Journal of Computer Theory and Engineering , 7*(3), 243-247.

Alastair, I. (2010). An Investigation into the Impact of Formative Feedback on the Student Learning Experience. Durham University.

Alavi, M. (1994, June). Computer-Mediated Collaborative Learning: An Empirical Evaluation. *MIS Quarterly, 18*(2), 159-174.

Alexandre, R., Queiros, P., & Leal, J. P. (2012). PETCHA: a programming exercises teaching assistant. *ITiCSE'12-17th ACM annual conference on Innovation and technology in computer science education*, (pp. 192-197).

Alhalifa, E. M. (2006). Effects of Learner Misconceptions on Learning. *IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2006)*, (pp. 123-128). CELDA.

Ali, A., & Smith, D. (2014). Teaching an Introductory Programming Language. *Journal of Information Technology Education: Innovations in Practice, 13*, 57-67.

Ali, A., & Smith, D. (2014). Teaching an introductory programming language in a general education course. *Journal of Information Technology Education: Innovations in Practice, 13*, 57-67.

Ali, N., Jusoff, K., Ali, S., Mokhtar, N., & Salamat, A. S. (2009). The Factors influencing Students' Performance at Universiti Teknologi MARA Kedah, Malaysia. *Management Science and Engineering, 3*(4), 81-90.

Almstrum, V. L., Henderson, P. B., Harvey, V., Heeren, C., Marion, W., Riedesel, C., . . . Tew, A. E. (2006). Concept Inventories in Computer Science for the Topic Discrete Mathematics. *ITiCSE '06* (pp. 26-28). Bologna, Italy: ACM.

AlQahtani, D. A., & Al-Gahtani, S. M. (2014). Assessing Learning Styles of Saudi Dental Students Using Kolb's Learning Style Inventory. *Journal of Dental Education, 78*(6), 927-933.

Anjali, G. (2014). Alignment of Teaching Style to Learning Preference: Impact of Student Learning. *Training and Development Journal, 5*(2), 119-131.

Bennett, S., Bishop, A., Dalgarno, B., Waycott, J., & Kennedy, G. (2012). Implementing Web 2.0 technologies in higher education: A collective case study. *Computers & Education, 59*(2), 524-534.

Biggs, J. (1998). What the student does: Teaching for enhanced learning in the '90s. *The higher Education research and development society of Australiasia annual International conference.* Auckland, New Zealand.

Black, P. J., & Lucas, A. M. (1993). *Children's Informal Ideas in Science.* Routledge.

Blerkom, M. L. (1992). Class Attendance in Undergraduate Courses. *The Journal of Psychology:Interdisciplinary and Applied, 126*(5), 487-494.

Boulay, B. D. (1986). Some Difficulties of Learning to Program. *Journal of Educational Computing Research, 2*(1), 57-73.

Brijesh Kumar Baradwaj, S. P. (2011). Mining Educational Data to Analyse Students' Performance. *International Journal of Advanced Computer Science and Applications, 2*(6), 63-69.

Bringula, R. P., Manabat, G. M., Tolentino, M. A., & Torres, E. L. (2012). Predictors of Errors of Novice Java Programmers. *World Journal of Education, 2*(1), 3-15.

Bruce, E. (2015, July 8). *https://edwinbruce.wordpress.com*. (wordpress.com) Retrieved September 29, 2015, from https://edwinbruce.wordpress.com/2015/07/08/programming-misconceptions-2/

Butler, M., & Morgan, M. (2007). Learning challenges faced by novice programming students studying high level and low feedback concepts. *Proceedings of the 24th ascilite Conference*, (pp. 2-5). Singapore.

Cakiroglu, U. (2014). *http://www.irrodl.org*. (Athabasca University) Retrieved 07 27, 2015, from http://www.irrodl.org/index.php/irrodl/article/view/1840/3004

Carrillo-de-la-Pena, M. T., Bailles, E., Caseras, X., Martinez, A., Ortet, G., & Perez, J. (2009). Formative assessment and academic achievement in pre-graduate students of health sciences. *Adv in Healh Science Education, 14*, 61-67.

Castano-Munoz, J., Duart, J. M., & Sancho-Vinuesa, T. (2014). The Internet in face-to-face higher education: Can interactive learning improve academic achievement? *British Journal of Educational Technology, 45*(1), 149-159.

ChanMin Kim, S. W., & Cozart, J. (2014). Affective and motivational factors of learning in online mathematics courses. *British journal of Educational Technology, 45*(1), 171-185.

Chatti, M., Dyckhoff, A., Schroeder, U., & Thus, H. (2012). A Reference Model for Learning Analytics. *International Journal of Technology Enhanced Learning*, *4*(5-6), 318-331.

Chen, J., & Lin, T.-F. (2008). Class Attendance and Exam Performance: A Randomized Experiment. *he Journal of Economic Education, 39*(3), 213-227.

Cohn, E., & Johnson, E. (2006). Class attendance and performance in Principles of Economics. *Education Economics, 14*(2), 211-233.

Corbett, A. T., & Anderson, J. R. (2001). Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 245-252). New York: ACM.

Credé, M., Roch, S. G., & Kieszczynka, U. M. (2010). Class Attendance in College: A Meta-Analytic Review of the Relationship of Class Attendance With Grades and Student Characteristics. *Review of Educational Research , 80*(2), 272-295.

Davidson, J. E., & Sternberg, R. J. (2003). The Fundamental Computational Biases of Human cognition. In *The Psychology of problem Solving* (pp. 291-310). Cambridge: Cambridge University Press.

de-la-Fuente-Valentín, L., Pardo, A., & Kloos, C. D. (2013). Addressing drop-out and sustained effort issues with large practical groups using an automated delivery and assessment system. *Computers & Education, 61*(February), 33-42.

Dinh, T. H. (1987). *Introduction to Vietnamese Culture.* ERIC.

E.Byrd, C., McNeil, N. M., Chesney, D. L., & G.Matthews, P. (2015). A specific misconception of the equal sign acts as a barrier to children's learning of early algebra. *Learning and Individual differences, 38*, 61-67.

Ebrahimi, A., Kopec, D., & Schweikert, C. (2006). Taxonomy of Novice Programming Error Patterns with Plan, Web, and Object Solutions. *ACM Computing Surveys*.

Eggen, P., & Kauchak, D. (2004). Windows on classrooms. *Educational Psychology, 6*.

Embrey, D. (2005). *www.humanreliability.com.* Retrieved August 04, 2015, from http://www.humanreliability.com/articles/Understanding%20Human%20Behaviour%20and%20Error.pdf

Eren, O., & Henderson, D. J. (2008). The impact of homework on student achievement. *The Econometrics Journal, 11*(2), 326-348.

Farooq, M., Chaudhry, A., Shafiq, M., & Berhanu, G. (2011). Factors affecting students' quality of Academic performance: A case of Secondary school level. *Journal of Quality and Technology Management, VII*(II), 1-14.

Fischer, F., Bruhn, J., Gräsel, C., & Mandl, H. (2002). Fostering collaborative knowledge construction with visualization tools. *Learning and Instruction, 12*(2), 213-232.

Fleming, N. (2001). Teaching and Learning styles: VARK strategies. Honolulu Community College.

Gaal, F. V., & Ridder, A. D. (2013). The Impact of assessment tasks on subsequent examination performance. *Active Learning in Higher Education, 14*(3), 213-225.

Goldman, K., Gross, P., Heeren, C., Herman, G., Kaczmarczyk, L., C.Loui, M., & Zilles, C. (2008). Identifying Important and difficult concepts in introductory computing courses using a Delphi process. *SIGCSE '08 Proceedings of the 39th SIGCSE technical symposium on Computer science education* (pp. 256-260). Portland: ACM.

Gratchev, I., & Balasubramaniam, A. (2012). Developing assessment tasks to improve the performance of engineering students. *AAEE 2012 conference.* Melbourne.

Gries, D. (1974). What should we teach in an introductory programming course? *SIGCSE '74 Proceedings of the fourth SIGCSE technical symposium on Computer science education*, *6*, pp. 81-89. New York, USA.

Gruba, P., & Sondergaard, H. (2001). A Constructivist Approach to Communication Skills Instruction in Computer Science. *Computer Science Education, 11*(3), 203-219.

Gump, S. E. (2005). The Cost of Cutting Class: Attendance As A Predictor of Success. *College Teaching, 53*(1), 21-26.

Handelsman, M. M., Briggs, W. L., Sullivan, N., & Towler, A. (2005). A Measure of College Student Course Engagement. *The Journal of Educational Research, 98*(3), 184-192.

Hestenes, D., Wells, M., & Swackhamer, G. (1992, March). Force Concept Inventory. *The Physics Teacher, 30*(3), 141-158.

Higgins, C. A., Gray, G., Symeonidis, P., & Tsintsifas, A. (2005). Automated assessment and experiences of teaching programming. *Journal on Educational Resources in Computing, 5*(3).

Hofstede, G., Hofstede, G. J., & Minov, M. (2010). *Culturers and Organizations: Software of the Mind.* New York: McGraw-Hill.

Holvitie, J., Haavisto, R., Kaila, E., Teemu Rajala, M.-J. L., & Salakoski, T. (2012). Electronic exams with automatically assessed exercises. *ICEE 2012 - International Conference on Engineering Education.* Turku, Finland.

Horton, D. M., Wiederman, S. D., & Saint, D. A. (2012). Assessment outcome is weakly correlated with lecture attendance: influence of learning style and use of alternative materials. *Advances in Physiology Education, 36*(2), 108-115.

housand, J. S. (2002). *The practical guide to empowering students, teachers, and families.* Baltimore: Paul H. Brookes Publishing Co.

Hristova, M., Misra, A., Rutter, M., & Mercuri, R. (2003). Identifying and correcting Java Programming errors for introductory computer science students. *SIGCSE '03 Proceedings of the 34th SIGCSE technical symposium on Computer science education, 35*, pp. 153-156. New York.

http://patientsafetyed.duhs.duke.edu/module_e/types_errors.html. (2014). Retrieved June 30, 2015

HU, K., AM, K., IU, M., A, M., S, A., MH, K., . . . SH, S. (2002). Impact of class attendance upon examination results of students in basic medical sciences. *Journal of Ayub Medical College, Abbottabad, 15*(2), 56-58.

Huet, I., Pacheco, O. R., Tavares, J., & Weir, G. (2004). New Challenges in Teaching Introductory Programming courses: A Case Study. *ASEE/IEEE Frontiers in Education Conference.* Savannah, GA.

James, R. (1990). *Human error.* Cambridge university press.

Judy Kay, M. B., Fekete, A., Greening, T., Hollands, O., & Crawford, J. H. (2000). Problem-Based Learning for Foundation Computer Science Courses. *Computer Science Education, 10*(2), 109-128.

Kaczmarczyk, L. C., Petrick, E. R., East, J. P., & Herman, G. L. (2010). Identifying Student Misconceptions of Programming. *SIGCSE '10 Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 107-111). Milwaukee, Wisconsin: ACM.

Kaila, E., Rajala, T., & Mikko-Jussi Laakso, S. (2008). Automatic Assessment of Program Visualization Exercises. *8th koli Calling International Conference on Computing Education Research.* Joensuu, Finland.

Kasuya, M. (2008). *Classroom Interaction Affected by Power Distance.* Language Teaching Methodology and Classroom Research and Research Methods.

Kennelly, B., & Flannery, J. C. (2011). Online Assignments in Economics: A Test of Their Effectiveness. *The Journal of Economic Education, 42*(2), 136-146.

Kim, V., & Seung Won Park, J. C. (2014). Affective and motivational factors of learning in online mathematics courses. *British Journal of Educational Technology, 45*(1), 171-185.

Knowles, M. S., III, E. F., & Swanson, R. A. (2014). *The Adult Learner: The definitive classic in adult education and human resource development.* Routledge.

Kolb, D. A. (1984). *Experiential Learning: Experience as a source of learning and development'.* New Jersey: Prentice Hall.

konan, P. N., Chatard, A., & Leila Selimbegovic, G. M. (2010). Cultural Diversity in the Classroom and its Effects on Academic Performance: A Cross National Perspective. *Social Psychology, 41*(4), 230-237.

Koulouri, T., Lauria, S., & Macredie, R. D. (2014). Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches. *ACM Transactions on Computing Education (TOCE), 14*(4), 26.1-26.27.

Krpan, D. &. (2011). Introductory programming languages in higher education. *2011 Proceedings of the 34th International Convention - IEEE.*

Kryjevskaia, M., Stetzer, M. R., & Grosz, N. (2014). Answer first: Applying the heuristic-analytic theory of reasoning to examine intuitive thinking in the context of physics. *Physical Review Special Topics-Physics Education Research, 10*(2), 020109-1- 020109-12.

L., J., Wendt, & Rockinson-Szapkiw, A. (2014). The Effect of Online Collaboration on Middle School Student Science Misconceptions as an Aspect of Science Literacy. *Journal of Research in Science Teaching, 51*(9), 1103-1118.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ITiCSE '05 Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education. 37*, pp. 14-18. iTiCSE '05.

Latif, E., & Miles, S. (2013). Class Attendance and Academic Performance: A Panel Data Analysis. *Economic Papers, 32*(4), 470-476.

Lee, L. (2005). Using Web-based Instruction to Promote Active Learning: Learners' Perspectives. *CALICO Journal, 23*(1), 139-156.

Lenox, T. L., & Woratschek, C. R. (2008). Exploring Declining CS/IS/IT Enrollments. *Information System Education Journal, 6*(44), 1-11.

lin, T.-F., & Chen, J. (2006). Cumulative class attendance and exam performance. *Applied Economics Letters, 13*(14), 937-942.

Lister, R., & Leaney, J. (2003). Introductory Programming, criterion-referencing and bloom. *ACM SIGCE BULLETIN, 35*(1), 143-147.

M.A. Chatti, A. D. (2012). A reference model for Learning Analytics. *International Journal of Technology Learning, 4*(5-6), 318-331.

Maguire, p., Maguire, R., Hyland, P., & Marshall, P. (2014). Enhancing collaborative Learning Using Pair Programming: Who Benefits? *All Ireland Journal of Teaching and Learning in Higher Education, 6*(2), 14111-14124.

Mallik, G. (2011). *Lecture and Tutorial Attendance and Student Performance in the First year economics course: A quantile regression approach.* Denver, United States: American Economic Association Annual Meeting.

Marburger, D. R. (2001). Absenteeism and Undergraduate Exam Performance. *The Journal of Economic Education, 32*(2), 99-109.

Marburger, D. R. (2006). Does Mandatory Attendance improve Student Performance? *The Journal of Economic Education, 37*(2), 148-155.

Marceau, G., Fisler, K., & Krishnamurthi, S. (2011). Measuring the Effectiveness of Error Messages Designed for Novice Programmers. *Proceedings of the 42nd ACM technical symposium on Computer science education.*

McCall, D., & Kölling, M. (2014). Meaningful Categorisation of Novice Programmer Errors. *Frontiers in Education Conference (FIE), 2014 IEEE. IEEE, 2014.*

Meister, D. (1989). The nature of human error. *Global Telecommunications Conference and Exhibition 'Communications Technology for the 1990s and Beyond' (GLOBECOM), 1989, IEEE. 2*, pp. 783-786. Dallas, TX: IEEE.

Miliszewska, I., & Tan, G. (2007). Information and Beyond Issues in Informing Science & Information Technology. In *Befriending Computer Programming: A proposed Approach to Teaching Introductory Programming* (pp. 277-289). Santa Rosa: Informing Science Press.

Mlambo, V. (2011). An analysis of some factors affecting student academic performance in an introductory biochemistry course at the University of West Indies. *Caribbean Teaching Scholar, 1*(2), 79-92.

Moore, C. B., Abella, I. D., & Boggs, G. (1997). Science Teaching Reconsidered: A Handbook. In *Chapter 4: Misconceptions as Barriers to Understanding Science* (pp. 27-30). Washington D.C.: National Academy Press.

Movshovitz-Hadar, N., Zaslavsky, O., & Inbar, S. (1987). An Empirical Classification Model for Errors in High School Mathematics. *Journal for Research in Mathematics Education, 18*(1), 3-14.

Muller, M. (2006). A Preliminary Study on the Impact of a Pair Design Phase on Pair Programming and Solo Programming. *Information and Software Technology, 48*(5), 335-344.

Muller, M. M. (2006). A preliminary study on the impact of a pair design phase on pair programming and solo programming. *Information and software Technology, 48*, 335-344.

Muller, M. M., & Padberg, F. (2004). An Empirical Study about the Feelgood Factor in Pair Programming. *Proceedings of the 10th International Symposium on Software Metrics.*

Nakiboglu, C., & Tekin, B. B. (2006). Identifying Students' Misconceptions about Nuclear Chemistry. *Journal of Chemical Education, 83*(11), 1712-1718.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., . . . Velázquez-Iturbide, J. Á. (2002). Exploring the role of visulaization and engagement in computer science education. *ITiCSE-WGR '02 Working group reports from ITiCSE on Innovation and technology in computer science education.* New York.

Narula, M., & Nagar, P. (2013). Relationship Between Students' Performance and Class Attendance in a Programming Language Subject in a Computer Course. *International Journal of Computer Science and Mobile Computing, 2*(8), 206-210.

O'Grady, & J, M. (2012). Practical Proble-Based Learning in Computing Education. *BACM Transactions on Computing Education, 2*(3), A1-A14.

Olsen, S. (1999). Errors and compensatory strategies: a study of grammar and vocabulary in texts written by Norwegian learners of English. *ScienceDirect-System, 27*(2), 191-205.

Orion, N., & Hofstein, A. (1994). Factors that influence Learning during a Scientific Field Trip in a Natural Environment. *Journal of Research in Science Teaching, 31*(10), 1097-1119.

Osmanbegović, E., & Suljic, M. (2012). Data mining approach for predicting student performance. *Journal of Economics and Business, X*(1), 3-12.

Ota, A. (2013, January 1). *Factors Influencing Social, Cultural, and Academic Transitions of Chinease Interntional ESL Students in U.S. Higher Education.* Port´land State University. PDXScholar. Retrieved August 4, 2015

Palincsar, A. S., Stevens, D. D., & Gavelek, J. R. (1989). Collaborating with teachers in the interest of student collaboration. *International Journal of Educational Research, 13*(1), 41-53.

Payne, L. (1989). A step towards group learning in computer programming. *Computer Education* , 23-25.

Perry, G., & Miller, D. (2013). *Sams Teach yourself Beginning Programming in 24 Hours.* Indianapolis, Indiana: Pearson Education.

Pudaruth, S., Nagowah, L., Sungkur, R., Moloo, R., & Chiniah, A. (2013). The Effect of Class Attendance on the Performance of Computer Science Students. *2nd International Conference on Machine Learning and Computer Science(IMLCS'2013).* Kuala Lumpur (Malaysia) .

Purcell, P. (2007). Engineering Student Attendance at Lectures: Effect on Examination Performance. *International Conference on Engineering Education – ICEE 2007* . Coimbra, Portugal.

Rajala, T., & Kaila, E. (2010). How does Collaboration Affect Algorithm Learning- A Case Study Using TRAKLA2. *International Conference on Education Technology and Computer(ICETC 2010)Shanghai*, (pp. 22-24).

Rajala, T., Kaila, E., Laakso, M.-J., & Salakoski, T. (2009). Effects of collaboration in program visualization. *Technology Enhanced Learning Conference 2009 (TELearn 2009).*

Rajala, T., Kaila, E., Laakso, M.-J., & Salakoski, T. (2010). How Does Collaboration Affect Algorithm Learning? - A case study using TRAKLA2. *International Conference on Education Technology and Computer (ICETC 2010)*, (pp. 22-24). Shanghai.

Rajala, T., Kaila, E., Laakso, M.-J., & Salakoski, T. (2011). Comparing the collaborative and independent viewing of program visualizations. *ASEE-IEEE Frontiers in Education Conference.* Rapid City, SD.

Rajala, T., Kaila, E., Laakso, M.-J., & Salakoski, T. (2011). Comparing the collaborative and independent viewing of program visulaizations. *41st ASEE-IEEE Frontiers in Education Conference.* Rapid City: IEEE.

Redecker, C. (2013). *The Use of ICT for the Assessemnt of Key Competences.* Luxembourg: European Commission, Joint Research Centre, Institute for Prospective Technological Studies. Retrieved from r http://europa.eu/

Reeves, T. C. (2006). How do you know they are learning?: the importance of alignment in higher education. *International Journal of Learning Technology, 2*(4), 294-309.

Reza Kormi-Nouri, S. M., Farahani, M.-N., Trost, K., & Shokri, O. (2015). Academic Stress as A Health Measure and Its Relationship to Patterns of Emotion in Collectivist and Individualist Cultures: Similarities and Differences. *International Journal of Higher Education, 4*(2), 92.

Richard O. Mines, J. (2014). The Impact of Testing Frequency and Final Exams on Student Performance. *ASEE Southeast Section Conference : American Society for Engineering Education, 2014.* Macon, GA.

Roberts, T. S. (2005). *Computer supported collaborative learning in higher education.* Australia: IGI Global.

Rocca, C. L., Margottini, M., & Capobianco, R. (2014). Collaborative Learning in Higher Education. *Open Journal of Social Sciences*, 61-66.

Romero, C., Ventura, S., Espejo, P. G., & Hervás, C. (2008). Data Mining Algorithms to Classify Students. *Educational Data Mining 2008.*

Salleh, N., Mendes, E., & Grundy, J. C. (2011). Emperical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. *IEEE Transactions on Software Engineering, 37*(4), 509-525.

Santiago, P., & Benavides, F. (2009). Teacher evaluation: A conceptual framework and examples of country practices. *OECD-Mexico Workshop 'Towards a Teacher Evaluation Framework in Mexico: International Practices, Criteria and Mechanisms.* Mexico City.

Sherman, M., Bassil, S., Lipman, D., Tuck, N., & Martin, F. (2013). Impact of auto-grading on an introductory computing course. *Journal of Computing Sciences in Colleges, 28*(6), 69-75.

Shuhidan, S., Hamilton, M., & D'Souza, D. (2010). Instructor perspectives of multiple-choice questions in summative assessment for novice programmers. *Computer Science Education*, *20*(3), 229-259.

Shute, V. J. (2008). *Focus on Formative Feedback.* Educational Testing Service.

Simanek, D. E. (2008, November). *Didaktikogenic Physics Misconceptions: Student misconceptions induced by teachers and textbooks*. Retrieved October 6, 2015, from https://www.lhup.edu/~dsimanek/scenario/miscon.htm

Sirkiä, T. (2012). *Recognizing Programming Misconceptions.* Aalto University.

Sorva, J. (2012). Misconception Catalogue - Appendix A. In *Visual Program Simulation in Introductory Programming Education* (pp. 358-368). Espoo, Finland: Aalto University publication series.

Spohrer, J. C., & Soloway, E. (1986). Alternatives to construct-based programming misconceptions. *CHI '86 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 17,* pp. 183-191. Boston.

Stanca, L. (2006). The Effects of Attendance on Academic Performance: Panel Data Evidence for Introductory Microeconomics. *The Journal of Economic Education, 37*(3), 251-266.

Sun, P.-C., Tsai, R. J., Finge, G., Chen, Y.-Y., & Yeh, D. (2008). What dirves a successful e-Learning? An empirical investigation of the critical factors influencing learner satisfaction. *Computers & Education, 50*(4), 1183-1202.

Sutcliffe, A., & Rugg, G. (1998). A Taxonomy of Error Types for Failure Analysis and Risk Assessment. *International Journal of Human-Computer Interaction, 10*(4), 381-405.

Teague, D., Corney, M., Colin Fidge, M. R., Ahadi, A., & Lister, R. (2012). Using Neo-Piagetian Theory, Formative In-Class Tests and Think Alouds to Better Understand Student Thinking: A Preliminary Report on Computer Programming. *AAEE2012 Conference.* Melbourne, Australia.

Thanh, P. T. (2011). An Investigation of Perceptions of Vietnamese Teachers and Students toward Cooperative Learning (CL). *International Education Studies, 4*(1), 3-12.

Tiwari, A., Lai, P., So, M., & Yuen, K. (2006). A comparison of the effects of problem-based learning and lecturing on the development of students' critical thinking. *Medical Education, 40*(6), 547-554.

Tran, V. D. (2014). The Effects of Cooperative Learning on the Academic Achievement and Knowledge Retention. *International Journal of Education, 3*(2), 131-140.

Treagust, D. F. (1988). Development and use of diagnostic tests to evaluate students' misconceptions in science. *International Journal of Science Education, 10*(2), 159-169.

Trumbull, E., & Lash, A. (2013). Understanding Formative Assessment: Insights from Learning Theory and Measurement Theory. San Francisco: WestEd.

Uysal, M. P. (2014). Improving First Computer Programming Experiences: The Case of Adapting a Web-Supported and Well- Structured problem-Solving Method to a Traditional Course. *Contemporary Educational Technology, 5*(3), 198-217.

Weinberger, A., Fischer, F., & Stegmann, K. (2005). Computer-supported collaborative learning in higher education:scripts for argumentative knowledge construction in distributed groups. *CSCL '05 Proceedings of th 2005 conference on Computer support for collaborative learning: learning 2005: the next 10 years!*

Vigeant, M., Prince, M., & Nottis, K. (2014). Repairing Engineering Students' Misconceptions About Energy and Thermodynamics. In *Teaching and Learning of Energy in K–12 Education* (pp. 223-236). Springer International Publishing.

ViLLE. (n.d.). (University of Turku) Retrieved 10 20, 2015, from http://villeteam.fi/index.php/en/

Willging, P. A., & Johnson, S. D. (2009). Factors that Influence Students’ Decision to Dropout of Online Courses. *Journal of Asynchronous Learning Networks, 13*(3), 115-127.

Wilson, B. C., & Shrock, S. (2001). Contributing to Success in an Introductory Computer Science Course: A Study of Twleve Factors. *ACM SIGCSE Bulletin, 33*(1), 184-188.

Virtanen, A., Tynjälä, P., & Eteläpelto, A. (2014). Factors promoting vocational students. *Journal of Education and Work, 27*(1), 43-70.

Vygotsky, L. (1992). *Educational Psychology.* Florida: St. Lucie Press.

Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes.* Harvard university press.

Yoo, J. Y., Swann, W. B., & Kim, K. O. (2014). The Influence of Identity Fusion on Patriotic Consumption: A Cross-Cultural Comparison of Korea and the US. *The Korean Journal of Advertising , 25*(5), 81-106.

Zualkernan, I. A., & Qadah, G. Z. (2006). Learning Styles of Computer Programming Students: A Middle Eastern and American Comparison. *IEEE Transactions on 49.4 (2006): 443-450.*

Özdener, N. (2008). A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer-programming students. *Computers & Education*, *51*(3), 1094-1102.

Appendix A

Table A1. Delphi's CI

No.	Key topics derived from Delphi's CI (Goldman, ym., 2008)
1	Syntax and semantics
2	Writing expressions for conditionals
3	Tracing control flow through execution
4	Tracing loop execution correctly
5	Understanding loop variable scope
6	Issues of scope, local vs global & understanding loop variable scope
7	Parameters scope and use in design
8	call by ref vs call by value
9	Writing expressions for conditionals
10	*Declaring and manipulating lists, referencing list elements (*Python does not have array data structure)

Table A2. E-final exam questions - Analyzed

Question	Topic covered	Concept - Delphi CI
1a	Variable declaration and assignment	Syntax and semantics
1c	Loop execution	Tracing loop execution correctly
1d	Loop and if statements in lists	Tracing control flow through execution
2	String handling functions and nested if statements	Writing expressions for conditionals
3	Using loop and functions in String handling	Understanding loop variable scope
4	Functions	Parameters scope and use in design, and call by ref vs call by value
5	Lists	Declaring and manipulating lists, and referencing list elements
6	Mathematical problem	Writing expressions for conditionals Understanding loop variable scope
7	Lists (kind of multi- dimensional array)	Declaring and manipulating lists, and referencing list elements

Table A3. Variables of Quantitative Analysis

Error type	Meaning (Meister, 1989)
Knowledge error	due to insufficient knowledge
Skill based error	due to slips and lapses
Rule based error	due to incorrect implementation of rule/method

Table A4. Results of Qualitative analysis

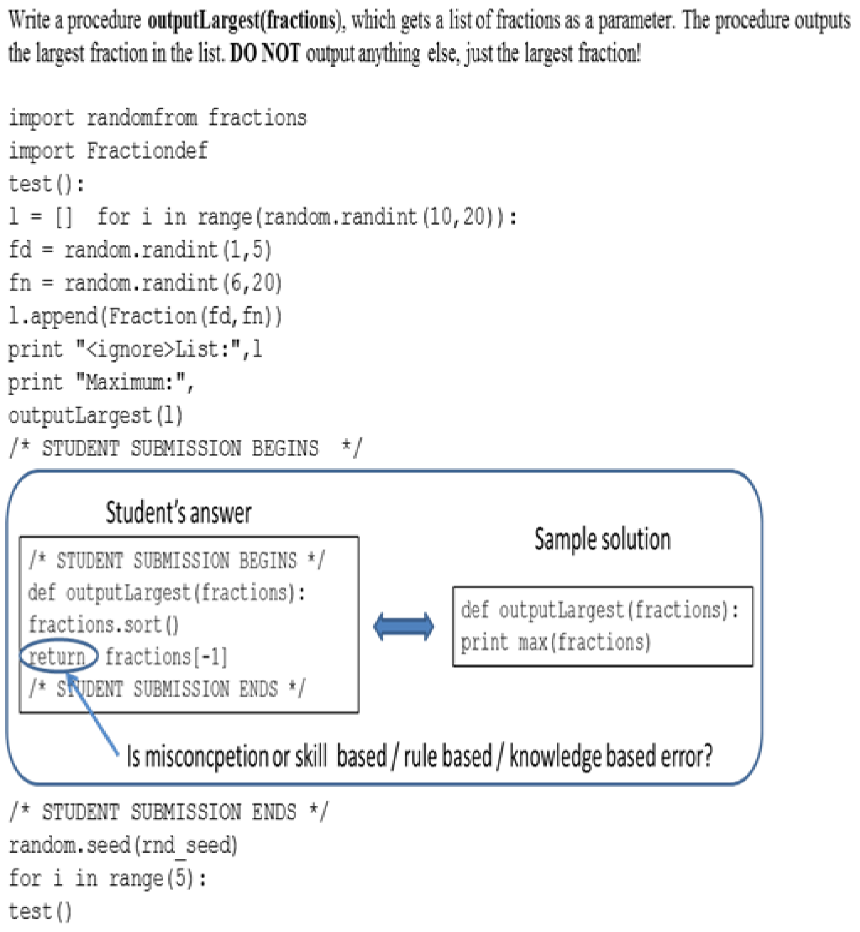
Key topics	Misconception
Syntax and semantics	student misunderstood the meaning of inbuilt functions and its application
parameter scope	student confused about the use of return statement and data type of the parameter passing
Writing expressions for conditionals	student misunderstood the process of control flow and especially nested if
Tracing loop execution	student misunderstand the process of *for* loop operation
Defining and referencing list elements	Student was not clear with index position and referencing *list* elements

Table A5. Key themes versus number of students

Details	No. of students	in %
Defined the methods correctly	26	67
Failed to define methods	4	10
Failed to trace the code	6	15
Failed to trace the code (6 students) but written program solutions that require looping statements	5	83
Had confusion in return statement of methods	3	8
Applied negative numbers as index positions in lists	5	13
Failed to complete mathematical problems based questions	26	67
Held programming misconception	14	36

Appendix B

Figure B1 Student's answer versus sample solution



Write a procedure **outputLargest(fractions)**, which gets a list of fractions as a parameter. The procedure outputs the largest fraction in the list. **DO NOT** output anything else, just the largest fraction!

```
import randomfrom fractions
import Fractiondef
test():
l = []  for i in range(random.randint(10,20)):
fd = random.randint(1,5)
fn = random.randint(6,20)
l.append(Fraction(fd,fn))
print "<ignore>List:",l
print "Maximum:",
outputLargest(l)
/* STUDENT SUBMISSION BEGINS  */
```

Student's answer

```
/* STUDENT SUBMISSION BEGINS */
def outputLargest(fractions):
fractions.sort()
return  fractions[-1]
/* STUDENT SUBMISSION ENDS */
```

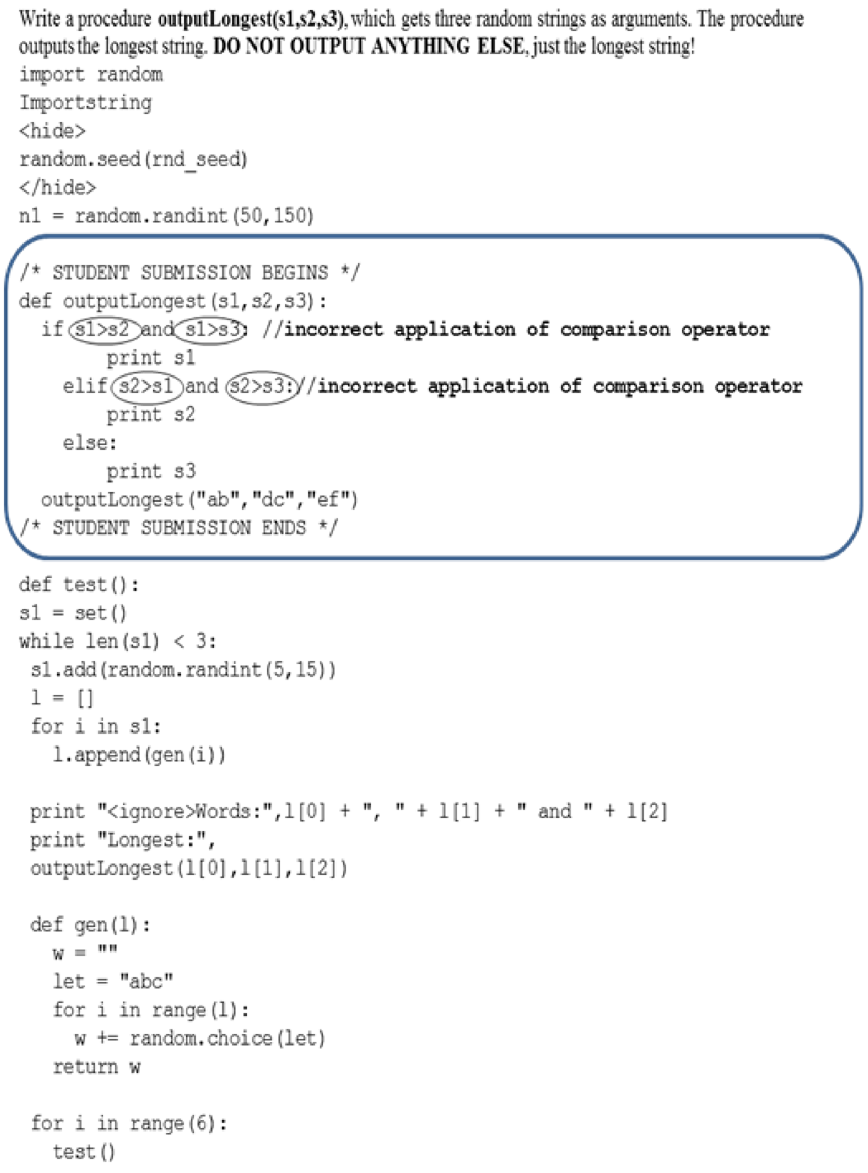
Sample solution

```
def outputLargest(fractions):
print max(fractions)
```

Is misconcpetion or skill based / rule based / knowledge based error?

```
/* STUDENT SUBMISSION ENDS */
random.seed(rnd_seed)
for i in range(5):
test()
```

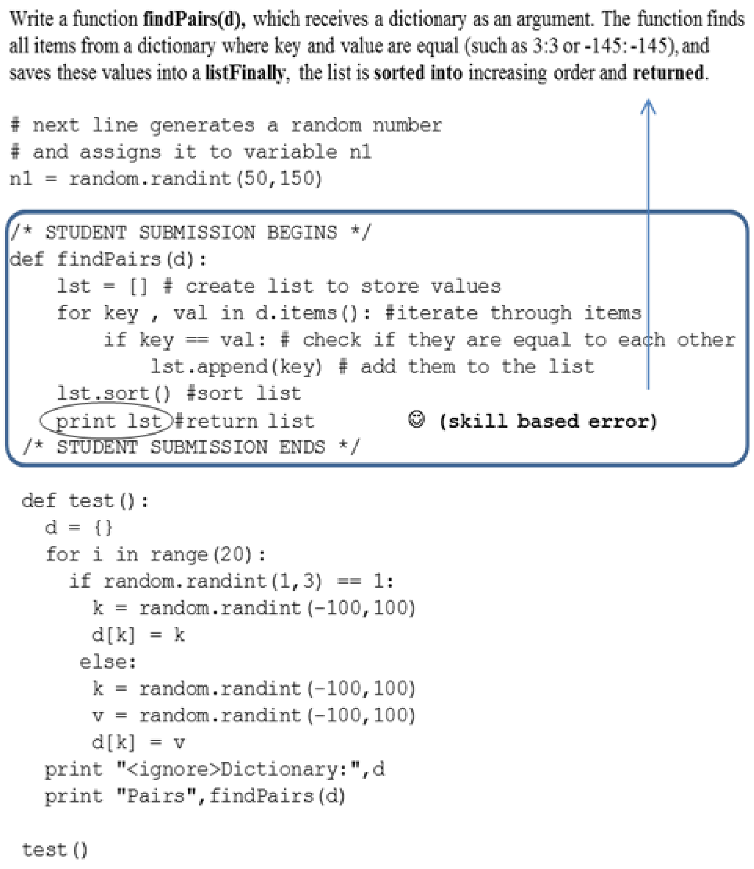
Figure B2 shows a screenshot of a student answer for Question 3.



Write a procedure **outputLongest(s1,s2,s3)**, which gets three random strings as arguments. The procedure outputs the longest string. **DO NOT OUTPUT ANYTHING ELSE**, just the longest string!

```
import random
Importstring
<hide>
random.seed(rnd_seed)
</hide>
n1 = random.randint(50,150)

/* STUDENT SUBMISSION BEGINS */
def outputLongest(s1,s2,s3):
  if s1>s2 and s1>s3: //incorrect application of comparison operator
        print s1
    elif s2>s1 and s2>s3://incorrect application of comparison operator
        print s2
    else:
        print s3
  outputLongest("ab","dc","ef")
/* STUDENT SUBMISSION ENDS */

def test():
s1 = set()
while len(s1) < 3:
 s1.add(random.randint(5,15))
 l = []
 for i in s1:
   l.append(gen(i))

 print "<ignore>Words:",l[0] + ", " + l[1] + " and " + l[2]
 print "Longest:",
 outputLongest(l[0],l[1],l[2])

 def gen(l):
   w = ""
   let = "abc"
   for i in range(l):
     w += random.choice(let)
   return w

 for i in range(6):
   test()
```

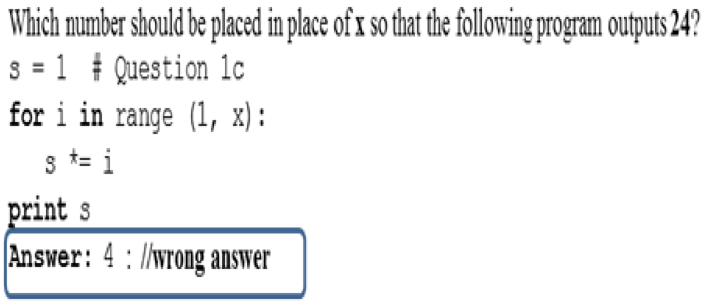
Figure B3 shows a screenshot of a student answer for Question 5.



Write a function **findPairs(d),** which receives a dictionary as an argument. The function finds all items from a dictionary where key and value are equal (such as 3:3 or -145:-145), and saves these values into a **listFinally**, the list is **sorted into** increasing order and **returned**.

```
# next line generates a random number
# and assigns it to variable n1
n1 = random.randint(50,150)

/* STUDENT SUBMISSION BEGINS */
def findPairs(d):
    lst = [] # create list to store values
    for key , val in d.items(): #iterate through items
        if key == val: # check if they are equal to each other
            lst.append(key) # add them to the list
    lst.sort() #sort list
   print lst #return list          ☺ (skill based error)
 /* STUDENT SUBMISSION ENDS */

 def test():
   d = {}
   for i in range(20):
     if random.randint(1,3) == 1:
       k = random.randint(-100,100)
       d[k] = k
      else:
       k = random.randint(-100,100)
       v = random.randint(-100,100)
       d[k] = v
   print "<ignore>Dictionary:",d
   print "Pairs",findPairs(d)

 test()
```

Figure B4 shows a screenshot of a student answer for code tracing Question 1c.



Which number should be placed in place of **x** so that the following program outputs **24**?

```
s = 1  # Question 1c
for i in range (1, x):
   s *= i
print s
Answer: 4 : //wrong answer
```

The results of quantitative analysis are shown in **Figure B5.**

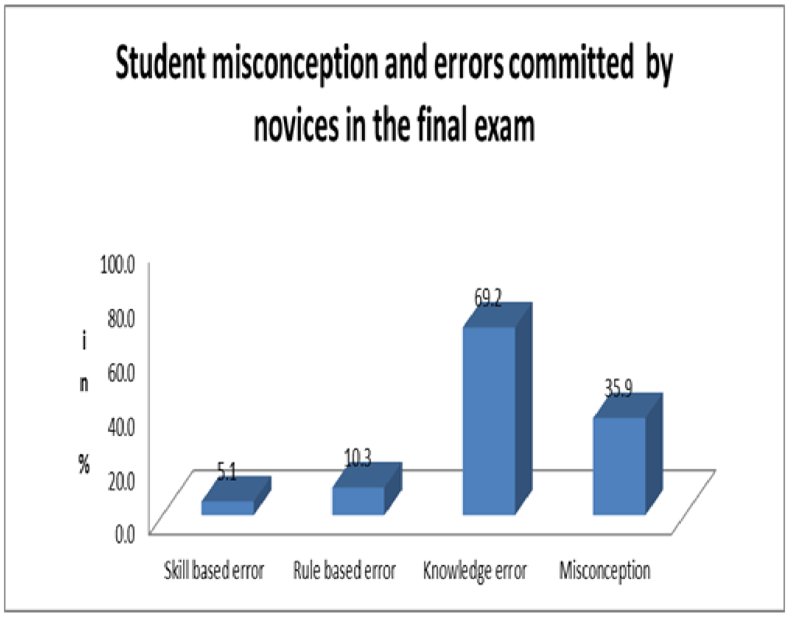
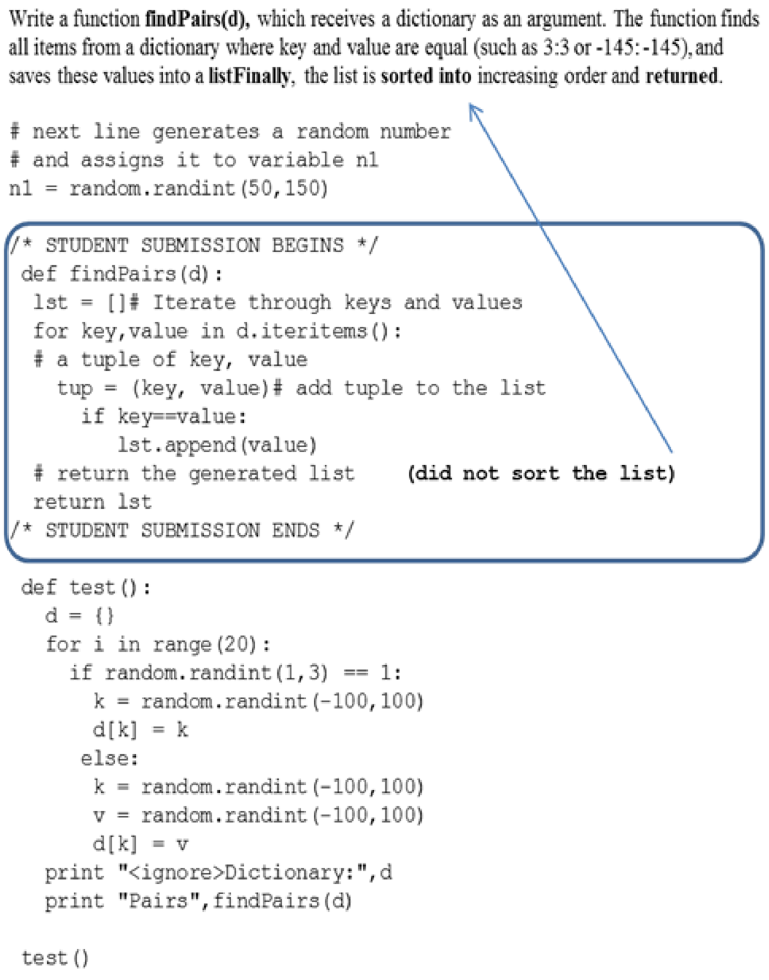


Figure B6 Student's answer versus ignorance of directives



Write a function **findPairs(d),** which receives a dictionary as an argument. The function finds all items from a dictionary where key and value are equal (such as 3:3 or -145: -145), and saves these values into a **listFinally**, the list is **sorted into** increasing order and **returned**.

```
# next line generates a random number
# and assigns it to variable n1
n1 = random.randint(50,150)

/* STUDENT SUBMISSION BEGINS */
 def findPairs(d):
  lst = []# Iterate through keys and values
  for key,value in d.iteritems():
  # a tuple of key, value
    tup = (key, value)# add tuple to the list
      if key==value:
         lst.append(value)
  # return the generated list     (did not sort the list)
  return lst
/* STUDENT SUBMISSION ENDS */
 def test():
   d = {}
   for i in range(20):
     if random.randint(1,3) == 1:
       k = random.randint(-100,100)
       d[k] = k
      else:
       k = random.randint(-100,100)
       v = random.randint(-100,100)
       d[k] = v
   print "<ignore>Dictionary:",d
   print "Pairs",findPairs(d)

 test()
```

Appendix C

Question	Details
1a	Which one of the following expressions evaluates into an object of type **string**?
	“abc”.count(“a”)
	“abc” is “ab” + “c”
	“abc”.find(“a”)
	“abc”[1:2]
	len(“abc”)
	”abc” == ”a” + ”bc”
1c	Which number should be placed **in place of x** so that the following program outputs **24**?
	s=1
	for i **in** range (1, x):
	s *= i
	print s
1d	The following function...
	def abc(lst):
	lst2 = []
	for i **in** lst:
	if i % 2 == 0:
	lst2.append(i)
	return lst2
	..is equivalent to which expression given below?
	abc = lambda x : [y for y in x if y % 2 == 0]
	abc = if y % 2 == 0: lambda x : [y for y in x]
	abc = lambda x : [y for y in x] % 2
	abc = [y for y in x if y % 2 == 0]
2	Write a procedure ***outputLongest(s1,s2,s3),*** which gets three random strings as arguments.
	The procedure outputs the longest string. **DO NOT OUTPUT ANYTHING ELSE**, just the

	longest string!
3	Write a function ***removeVowels,*** which gets a random string s as an argument. The function creates and returns a new string, which contains all character in s except for vowels (a, e, i, o, u, y). For example, if the function was called with a string "python", it would return a string "pthn".
4	Write a function ***minMidMax(numberList)***, which gets a list of integers as an argument. The function returns a tuple containing minimum, median and maximum items from the list. The median item is an item which has an equal amount of smaller and larger items in the list. For example, if the function was called with list [5, 3, 1, 2, 4], it would return a tuple (1, 3, 5).Note, that **you can NOT change** the list given as an argument in any way!
5	Write a function ***findPairs(d)***, which receives a dictionary as an argument. The function finds all items from a dictionary where key and value are equal (such as 3:3 or -145: -145), and saves these values into a **listFinally**, the list is **sorted** into increasing order and returned.
6*	Write a function ***isPermutation(s1, s2)***, which returns true, if string s1 is a permutation of string s2. For s1 to be a permutation of s2, the number of times each character [a...z] occurs in the string should be equal for s1 and s2. The order of characters does not matter.
7*	Write a procedure ***flip(matrix)***, which receives a matrix with random items as an argument. The matrix is a **square matrix** (i.e. there is an equal amount of rows and columns).The procedure is supposed to flip the matrix, i.e. convert the rows into columns and vice versa. For example, if called with matrix like this [[1,2], [3,4]], the procedure transforms it into [[1,3],[2,4]].
*students were allowed to use internet facilities to know more about Permutation and square matrix during exam hours.	