Revista EIA
ISSN: 1794-1237
revista@eia.edu.co
Escuela de Ingeniería de Antioquia
Colombia

Fostering Motivation and Improving Student Performance in an introductory programming course: An Integrated Teaching Approach

Solarte Pabón, Oswaldo; Machuca Villegas, Liliana
Fostering Motivation and Improving Student Performance in an introductory programming course: An Integrated Teaching Approach
Revista EIA, vol. 16, no. 31, 2019
Escuela de Ingeniería de Antioquia, Colombia
Available in: https://www.redalyc.org/articulo.oa?id=149258931005
DOI: https://doi.org/doi.org/10.24050/reia.v16i31.1230

Fostering Motivation and Improving Student Performance in an introductory programming course: An Integrated Teaching Approach

Fortaleciendo la motivación y mejorando el rendimiento de estudiantes de un curso introductorio de programación: Un enfoque de enseñanza integrado

Fomentar a motivação e melhorar o desempenho dos alunos em um curso de programação introdutória: uma abordagem de ensino integrado

Oswaldo Solarte Pabón
oswaldo.solarte@correounivalle.edu.co
Universidad del Valle, Colombia
Liliana Machuca Villegas
liliana.machuca@correounivalle.edu.co
Universidad del Valle, Colombia

Revista EIA, vol. 16, no. 31, 2019

Escuela de Ingeniería de Antioquia, Colombia

Received: 01 April 2018
Accepted: 25 July 2018

DOI: https://doi.org/doi.org/10.24050/reia.v16i31.1230

Redalyc: https://www.redalyc.org/articulo.oa?id=149258931005

Resumen: Este artículo es una extensión de una propuesta de enseñanza presentada en Innovation and Technology in Computer Science Education Conference, en el año 2016. La propuesta representa un enfoque de enseñanza integrado para mejorar el rendimiento de los estudiantes en un primer curso de programación. El enfoque se basa en cuatro componentes principales: el uso de Python como primer lenguaje de programación, aprendizaje orientado a proyectos y basado en problemas, recursos multimedia y rúbricas de evaluación. Para el desarrollo del curso estuvieron disponibles materiales y recursos de aprendizaje en plataformas virtuales. Los hallazgos sugieren que el enfoque mejoró el rendimiento académico de los estudiantes, evidenciado en sus calificaciones, así como en una disminución en las tasas de deserción.

Palabras clave: Curso introductorio de programación, Enfoque de enseñanza, Python, Aprendizaje orientado por proyectos y basado en problemas.

Abstract: This paper expands a teaching proposal presented at the Innovation and Technology in Computer Science Education Conference, in 2016. The proposal provides an integrated teaching approach for improving students' performance in a first programming course. The approach is based on four main components: the use of Python as first programming language, project-oriented and problem-based learning, multimedia resources, and assessment rubrics. Material and learning resources for the course development are available on virtual platforms. Our findings suggest that the approach enhanced students' academic performance, as can be seen in their grades, as well as a decrease in dropout rates.

Keywords: Introductory programming course, Teaching approach, Python, Project-oriented and problem-based learning.

Resumo: Este artigo expande uma proposta de ensino apresentada na Conferência de Inovação e Tecnologia em Ciência da Computação, em 2016. A proposta explica uma abordagem de ensino integrado para melhorar o desempenho dos alunos em um primeiro curso de programação. A abordagem baseia-se em quatro componentes principais: o uso de Python como primeira linguagem de programação, aprendizagem orientada a projetos e baseada em problemas, recursos multimídia e rubricas de avaliação. Materiais e recursos de aprendizagem para o desenvolvimento do curso estavam disponíveis o tempo todo em plataformas virtuais. Os achados sugerem que a abordagem melhorou o desempenho acadêmico dos alunos, evidenciado em suas notas, bem como em uma redução nas taxas de abandono escolar.

Palavras-chave: Curso de programação introdutória, Abordagem de ensino, Python, Aprendizado orientado para projetos e com base em problemas.

INTRODUCTION

Attending a computer programming course for the first time might be a challenging task for many students. In fact, programming courses often have considerable amounts of students who either fail or dropout (Bennedsen and Caspersen, 2007), (Mason, Cooper and Raadt, 2012). Moreover, this problem is not restricted to computer science students, since students in other engineering majors must also take programming courses. Students generally consider introductory programming courses are difficult and low motivating subjects, as stated by (Chan Mow, 2008), (Ali and Smith, 2014), (Koulouri et al, Lauria and Macredie, 2014), (Alturki R. A. ,2016).

These students' perceptions influence their performance, since they enroll on the courses with false preconceptions. This may derive from a lack of motivation to learn programming, low final marks, and high dropout rates. Other factors that feed such perceptions are related to a shortage of successful methodologies for programming teaching (Salcedo and Idrobo, 2011), (Gomes and Mendes, 2015), (Alturki R. A. ,2016). Another reason, is the complexity of some programming languages that are chosen for introductory courses (Koulouri, Lauria and Macredie, 2014). For these reasons, students find that introductory programming courses are neither interesting nor relevant for their academic needs.

Nevertheless, learning to program is a required essential skill in all fields of knowledge, as it may be applied to solve a vast array of problems through the use of computers and algorithms. For instance, algorithms and programming have been reported to be a great help in the fields of biomedical science (Chapman et al., 2015), Civil, Electrical engineering (Hoffbeck et al, 2016) and Mechanical engineering (Furman B & Wertz E, 2010). Moreover, according to Van Roy et al, (2003), everyone should learn programming. Programming is not just a specialised discipline limited to computer science majors, it is a form of thinking that is useful to everyone.

The Universidad del Valle, Cali, Colombia, offers a course in Algorithms and Programming that is part of the curriculum for all Engineering majors, and has to be attended by all first-year students. During the last 8 years, the professors from this specific faculty have constantly observed low academic performance from these students, as well as high dropout rates, not just from the course but also from the majors themselves.

Considering this problem, this paper proposes a promising integrated teaching approach for introductory programming courses. This initiative seeks, on the one hand, to reduce students' failure and dropout rates, and on the other hand, to improve students' motivation towards programming. This will help them to perceive the courses' contents as meaningful and providing them with useful knowledge that may be applied in their daily life. This approach is based on four main components: the use of Python as first programming language, project-oriented and problem-based learning, multimedia resources, and assessment rubrics. The initial implementation of this approach has yielded partial results, suggesting a positive impact on the academic performance of engineering students.

The rest of this article is structured as follows: Section 2 presents related works. Section 3 describes the proposed teaching approach. Section 4, shows preliminary results. Finally, the paper ends with some conclusions and further research issues.

2. RELATED WORKS

Teaching an introductory programming course is a considerable challenge for any teacher, especially as many students regard this course as a difficult subject and their motivation towards it is very low. For this reason, several proposals have been developed (Yadin, 2011), (Chien-An et al., 2015), all of which aim at improving students' performance in introductory programming courses. In this article the proposals are divided into two groups: The first group takes into account the importance of programming language (Van Roy et al, 2003), (Enbody & Punch, 2009), (Zelle, 1999). The second group contains proposals, that in addition to the programming language, consider other aspects of the teaching process such as pedagogical strategies or teaching aids (Yadin, 2011), (Koulouri, Lauria and Macredie, 2014), (Salcedo and Idrobo, 2011).

The proposal reported by Van Roy et al. (2003), describes the role of different programming paradigms and languages in teaching programming. There are many programming paradigms: imperative programming, object-oriented programming, logic programming, and functional programming. They all have their advantages and disadvantages. Choosing an appropriate programming paradigm is a hard decision. One way to solve it is to focus programming courses on concepts, and the design process: how problem statements lead to well-organized solutions. According to Vujošević-Janičić, M. and Tošić, D. (2008), tools such as C language (Imperative) or Java language (Object-oriented) are very difficult to learn on a first programming course because they have complex syntax for novice programmer students. Therefore, students spend most of their time trying to learn the syntax of the programming language instead of the most important concepts.

Zelle (1999) states that high-level scripting languages such as Python, Perl, Tcl, and Rexx are better candidates for a first programming course than traditional systems programming languages such as C, C ++, and Java. Scripting languages are simpler, safer and more flexible than system languages. A first programming course should be designed to provide an introduction to the field of computer science and focus on problem solving. Considering these facts, scripting languages may improve these goals because these languages offer simple syntax and semantics. Moreover, in a first programming course students tackle simple problems which should be solved simply.

Under this perspective, Stajano, F. (2000) explains how Python is an excellent choice for introducing fundamental ideas about programming. Python has a high level of abstraction, simplicity, conciseness, and versatility. It is widely recognised as being easy to learn and to use for beginners. Another Python strength is its community around the world, which has encouraged the development of large number of modules and packages for a wide variety of applications. For instance, a great tool for educators is Jupyter Notebook. This is a web-based programming environment for Python and facilitates code writing and execution.

According to Chien-An et al. (2015), Python should be taught as the first programming language because it has simpler syntax and high-level data structures that facilitate writing code for learners. Although languages such as Java or C++ are effective for designing real applications and therefore are popular in industry, these tools are not the ideal as a first programming language. This is because some concepts, such as; classes, methods, types, and complex syntax can be a challenge for novice programmers and can make the learning process difficult. In order to reduce learning difficulties and failures, the criteria for choosing a first programming language should include: simple input/output statements, readable and consistent code, and clear syntax. Bearing these criteria in mind, Python could be a good option for novice programmers.

Enbody & Punch (2009) describe the experience and the impact of replacing C++ language with Python in the first programing course at Michigan State University. The impact of this change was measured in two ways: First, they assessed students' performance in the first programming course using Python. Second, they assessed students' performance in the second programming course which is taught in C ++. Their conclusions show that Python has useful features, such as, readability and practicality which facilitate the learning process. It is also considered as a viable alternative for a first programming course, even for curriculums whose subsequent courses are based on a different language, such as, C++ or Java.

The previously mentioned proposals are very important because they analyze the importance of the programming language in a first programming course. However, in the teaching process other aspects such as pedagogical strategies that support the learning process must be considered. That is to say, choosing an appropriate programming language is not the only thing that can improve programming learning. For instance, Yadin (2011), proposes a teaching strategy based on three elements: the use of Python as programming language, the use of visualization microworlds, and the assignment of individual tasks. This strategy was applied over four semesters and students' performance was monitored in order to help them to face issues related to introductory programming courses. This strategy allowed them to reduce their failure rates by 77.4%.

Salcedo and Idrobo (2011) propose tools and methodologies for programming languages learning using the Scribbler Robot and Alice. Using these tools, the students have a friendly interface that allows them to learn programming concepts in a more friendly and didactic way. The authors express the need to create new alternatives for improving pedagogical methods in the teaching of programming. The goal is to motivate and encourage students' performance using visual tools. This proposal has been implemented at ICESI University, Colombia. The results show a strengthening in the learning of programming concepts and the development of algorithmic thinking.

Aris (2015) explores four approaches for improving students' performance in an introductory programming course: attendance monitoring, personalised attention during lab session, restructuring of the content, and quantifiable distribution of examination questions. These changes have been implemented over four semesters, obtaining positive results, which show that students marks have improved. In the case of attendance monitoring, the author considers that this strategy is viable if it is recorded. The personalized attention has allowed teachers to discover that some students cannot perform the exercises proposed and if they are not guided by a teacher, they will not ask for help either. Following modifications to the content of the course, it was possible to organize some topics at the end of the subject, since they are considered more difficult for the students, specifically those related to modular programming. Lastly, the final exam questions were assigned a percentage according to the topics, for example, 40% for fundamental topics.

Echeverría et al. (2017) describe an approach to teach programming to non-Computer Science majors based on collaborative method. This approach is supported by the TASystem platform through which the different joint-working scenarios can be configured. A collaborative scenario is a learning scenario that has collaborative learning activities and evaluation strategies. The Instructors can design collaborative learning and assessment tasks. In the case of students, they can submit tasks, write comments on classmates' tasks, and rate other classmates' tasks. The results of the implementation of the approach show that students' performance has improved and that social interactions also had a positive effect on the process.

Additionally, a survey of literature on teaching in introductory programming courses is presented by (Pears et al., 2007). This survey focuses on searching for literature about curriculum, pedagogy, choice of language and tools for teaching in this field. It presents a wide range of research and works to be used as approaches to support the teaching of a first course of program. (Gomes and Mendes, 2015) describe a study related with the educational and motivational strategies used to teach programming. In order to achieve this aim, interviews were conducted with different teachers to collect their experiences in teaching programming.

In the same way, (Koulouri, Lauria and Macredie, 2014) describe a quantitative evaluation of different approaches which studied the effects of three factors related with teaching introductory programming. It suggests the combinations of different elements: choice of programming language, problem-solving training and the use of formative assessment. Their findings suggest that by using Python, teaching problem-solving and formative feedback may facilitate students' learning of; programming concepts, improving students' performance and developing programming skills.

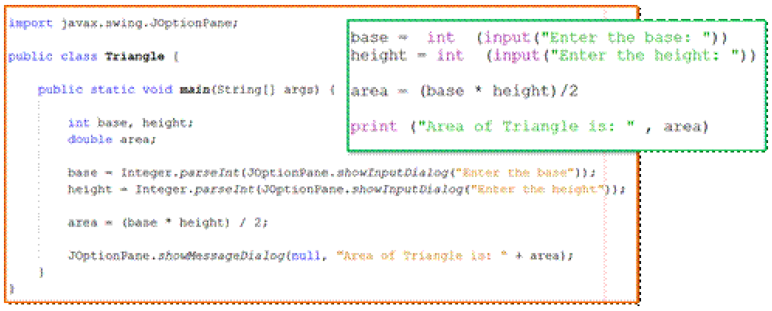
3. TEACHING APPROACH

In this section, an integrated teaching approach for a first programming course is presented. It seeks to improve students motivation and performance. This approach was structured on the basis of four main components covering different perspectives: the use of Python as first programming language (technological perspective), project-oriented and problem-based learning (didactic perspective), multimedia resources (technology enhanced learning perspective), and assessment rubrics (evaluation perspective). According to our experience, teaching programming for the first time, can be considered as a complex process that involves the aforementioned perspectives, which can help to improve the learning process of the students further.

3.1. Python as a first programming language

Choosing an appropriate programming language for an introductory programming course is a challenging task. A simple syntax and a friendly programming environment are desirable pre-requisites in order for students to understand basic concepts and develop problem solving skills (Guo, 2014). At Universidad del Valle, however, the use of Java for more than ten years in introductory course showed that students perceived this programming language as difficult. This translated into low motivation towards learning, low academic performance, high rates of failure and dropouts. With this in mind, Java was replaced by Python, which complies with the desired characteristics. For example, Figure 1, shows a simple algorithm to calculate the area of a triangle given its base and its height. Python code is easier and more legible than Java code. Moreover, Java code has complex syntax that demotivated students because they do not understand many terms that Java is using to solve a simple problem. Some Java terms such as "public class", "public static void", can turn a simple problem into a complex problem and programming is perceived as difficult task.

On the contrary, Python is more appropriate than Java for learning how to program. It offers a high level of abstraction of programming concepts. This makes learning easier and reduces students' anxiety towards other aspects that may not be relevant in this level, such as, memory management and data types. In other words, the main advantage of Python as a programming language is its high level of abstraction, which is appropriate to introduce the fundamental concepts of algorithms. According to Figure 1, Python only uses two simple functions (input, print), in order to solve this problem. Python code allow students to concentrate on problem-solving, instead of worrying about understanding the syntax, as happened when the course was taught with Java.



```
import javax.swing.JOptionPane;

public class Triangle {

    public static void main(String[] args) {

        int base, height;
        double area;

        base = Integer.parseInt(JOptionPane.showInputDialog("Enter the base"));
        height = Integer.parseInt(JOptionPane.showInputDialog("Enter the height"));

        area = (base * height) / 2;

        JOptionPane.showMessageDialog(null, "Area of Triangle is: " + area);
    }
}
```

```
base =  int  (input("Enter the base: "))
height = int  (input("Enter the height: "))

area = (base * height)/2

print ("Area of Triangle is: " , area)
```

Figure 1

Figure 1. A simple algorithm in Java and Python code.

In addition to using Python, students follow a methodology (Figure 2), which facilitates the understanding of the problem they need to solve. This methodology consists of a set of steps to guide the development of the class: problem analysis, pseudocode design, coding, and testing. Through this methodology, the professor can guide the work of the students to develop proposed programming exercises in class. In the same way, student can use this methodology in their lab sessions and class project. Using this methodology is very important, because it helps students to understand the problem and facilitates solution implementation step by step.

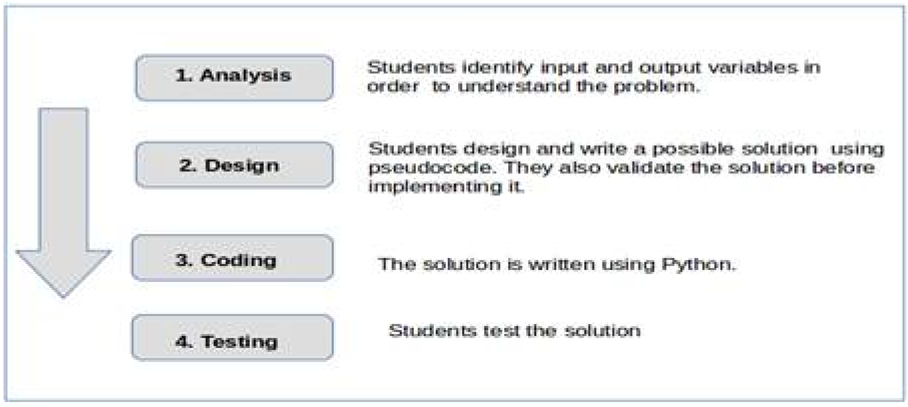

1. Analysis
Students identify input and output variables in order to understand the problem.
2. Design
Students design and write a possible solution using pseudocode. They also validate the solution before implementing it.
3. Coding
The solution is written using Python.
4. Testing
Students test the solution

Figure 2

Figure 2. A problem-solving methodology

3.2. Project-oriented and problem-based learning

The approach is grounded on a constructivist theory of learning, specifically project-based and problem-based learning (Konecki and

Petrlic, 2014), (Soares, 2011). Through these approaches, students can achieve meaningful learning and critical thinking while developing computational skills. The learning activities are designed in order for students to build a collaborative project, on one hand, and solve real-life problems related to students' academic needs on the other. Through the design of the collaborative project, students enhance research abilities related to the process of suggesting an idea and structuring a relevant proposal in their academic field.

Similarly, some lab sessions have been integrated throughout the course as a strategy to provide students with opportunities to develop problem-solving skills. The goal of the project is for students to achieve more motivation towards algorithms, so that they become acquainted with their applicability in their Engineering majors. The lab sessions consist of a set of exercises that directly relate to the course content that is being taught at that moment. For each algorithm exercise, students are expected to submit the analysis, design, implementation and algorithm test. Figure 3 illustrates some of the project and lab session activities.

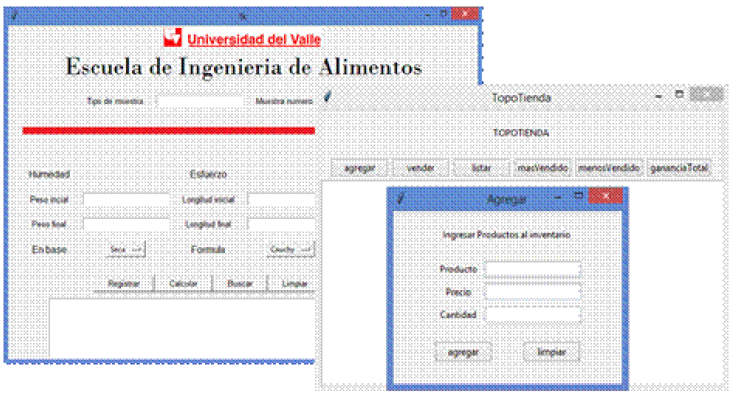


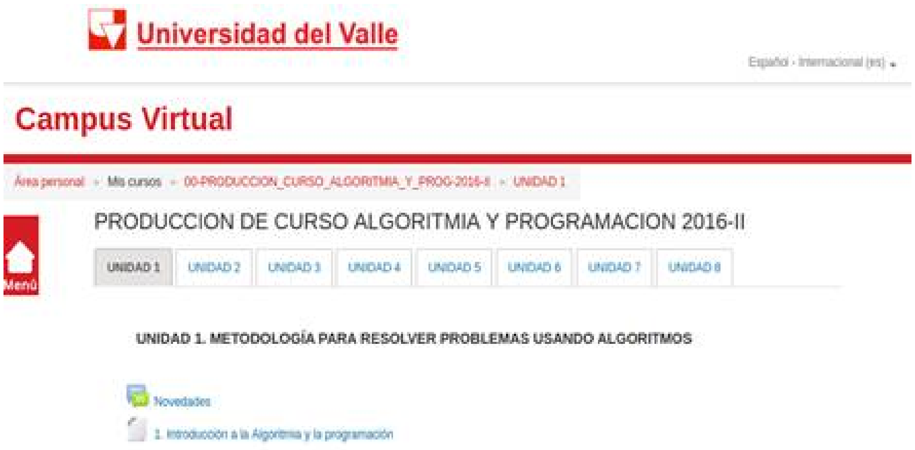
Figure 3.

Figure 3. Final Project Examples

3.3. Multimedia Resources

In order to provide a high-quality teaching process, some strategies of technology-enhanced-learning were promoted through didactic multimedia resources. These multimedia resources support learning activities on our courses. They comprise of pictures, slideshows, videos, animation and tutorials, all of which were utilised to further strengthen concepts and experiences in the learning of programming skills.

In addition, multimedia resources are very useful for students to understand abstract concepts or ideas in a much easier way. Accordingly, the School of Computer Sciences and Systems, at Universidad del Valle, is currently designing new material that consists of recording the introductory programming course classes, which will be shown on virtual platforms. Using these resources, the material will be available all the time, and students will be able to take different classes in a virtually. Some of these videos, animation and multimedia resources are illustrated in Figures 4 to 6.



Universidad del Valle

Español - Internacional (es)

Campus Virtual

Área personal » Mis cursos » 00-PRODUCCION_CURSO_ALGORITMIA_Y_PROG-2016-II » UNIDAD 1

Menú

PRODUCCION DE CURSO ALGORITMIA Y PROGRAMACION 2016-II

UNIDAD 1 | UNIDAD 2 | UNIDAD 3 | UNIDAD 4 | UNIDAD 5 | UNIDAD 6 | UNIDAD 7 | UNIDAD 8

UNIDAD 1. METODOLOGÍA PARA RESOLVER PROBLEMAS USANDO ALGORITMOS

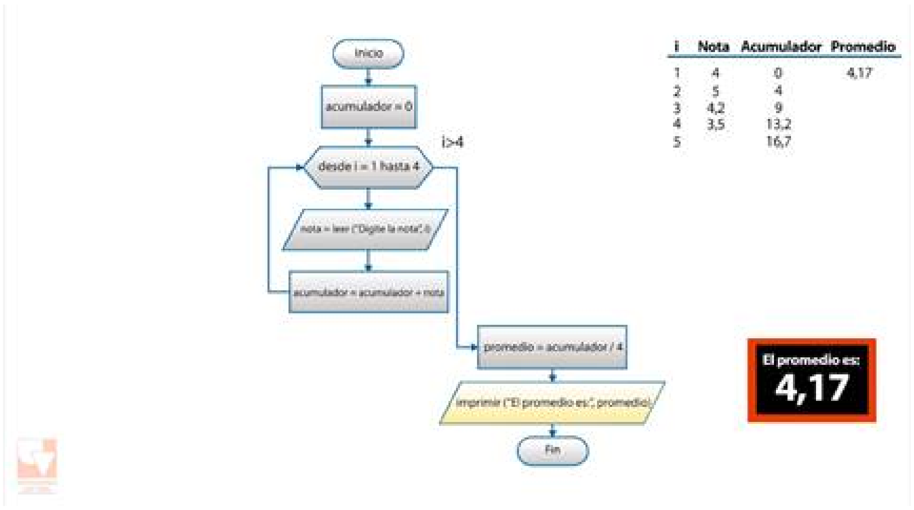
Novedades

1. Introducción a la Algoritmia y la programación

Figure 4.
Figure 4. Virtual Campus for First Programming Course at Univalle

Algoritmia
y Programación
CURSO DE ALGORITMIA Y PROGRAMACIÓN EISC
PÁGINA PRINCIPAL
VIDEOS
LISTAS DE REPRODUCCIÓN
CANALES
ACERCA DE
Unidad 8. Diseño de Interfaces Gráficas de Us...
Unidad 7.5-9 Matrices, Ciclos y Condicionales (operaciones
Unidad 7.4 Operaciones de matrices con NumPy
Unidad 7.1-3 Matrices (arreglos bidimensionales,

Figure 5
Figure 5. YouTube Channel for First Programming Course
https://www.youtube.com/channel/UCgok2gslPgwNzWBcnMMEblA


Inicio
acumulador = 0
i>4
desde i = 1 hasta 4
nota = leer ("Digite la nota",0)
acumulador = acumulador + nota
promedio = acumulador / 4
imprimir ("El promedio es:", promedio)
Fin
El promedio es:
4,17

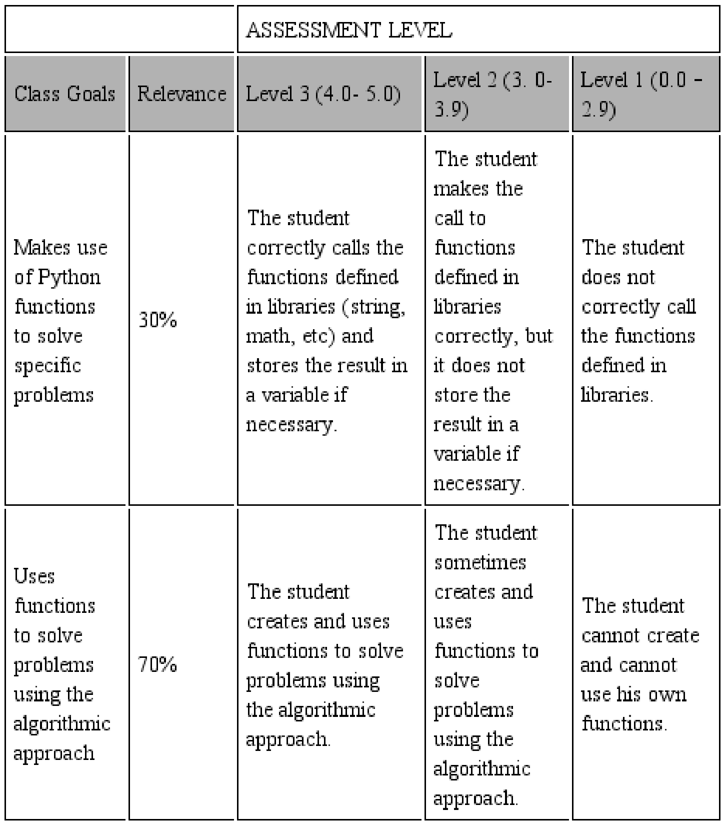
i	Nota	Acumulador	Promedio
1	4	0	4,17
2	5	4	
3	4,2	9	
4	3,5	13,2	
5		16,7	

Figure 6
Figure 6. Screenshot of an animation

3.4. Rubrics for assessment

Assessment also is an important part of a teaching programming course because it helps students and professors to review the process to enhance students' learning. Rubric is a scoring tool that lists the criteria for a specific assignment and it describes the levels of quality for each criterion. Using rubrics with detailed explanations of an assignment and its assessment, can assist students in improving their performance. Rubrics provide students with a much clearer picture of the kind of performance that is expected from them and the requirements under which they will be assessed. Therefore, in our approach, assessment rubrics for each activity were presented and explained to the students in advance. An example is shown in Table 1.

Table 1
Table 1. Rubric example



		ASSESSMENT LEVEL		
Class Goals	Relevance	Level 3 (4.0- 5.0)	Level 2 (3. 0- 3.9)	Level 1 (0.0 - 2.9)
Makes use of Python functions to solve specific problems	30%	The student correctly calls the functions defined in libraries (string, math, etc) and stores the result in a variable if necessary.	The student makes the call to functions defined in libraries correctly, but it does not store the result in a variable if necessary.	The student does not correctly call the functions defined in libraries.
Uses functions to solve problems using the algorithmic approach	70%	The student creates and uses functions to solve problems using the algorithmic approach.	The student sometimes creates and uses functions to solve problems using the algorithmic approach.	The student cannot create and cannot use his own functions.

Table 1 Table 1. Rubric example

This rubric portrays three main parts: class goals, relevance and assessment levels. Class goals show students the skills that they are expected to develop in a given activity. Relevance indicates the percentage for each goal in the final mark. Finally, there are three assessment levels (Level 1, Level 2, Level 3) that indicate the progress made by the student in the development of a given activity. In this teaching approach, students know previously the rubric for an assignment, therefore they know how it will be scored. This represents a positive learning factor because it motivates students to achieve the best grade because rubric describes how to reach it

4. RESULTS

The proposed approach has been implemented by all the majors in the Engineering Faculty at Universidad del Valle since 2015. Some initial results are reported in (Machuca and Solarte Pabón, 2016) showing an analysis of the final grades in programming courses from 2011 to 2015. In the last couple of years (2016 and 2017) the percentage of students passing the first programming course has increased in around 13% (Figure 7). These findings suggest a considerable improvement in student performance. For instance, while in 2011 the percentage of students who passed the course was 70% and those who failed was around 30%, in 2017 the percentage was 90% and 10% respectively. This is an important achievement since it was possible to reduce the rate of failure in the course.

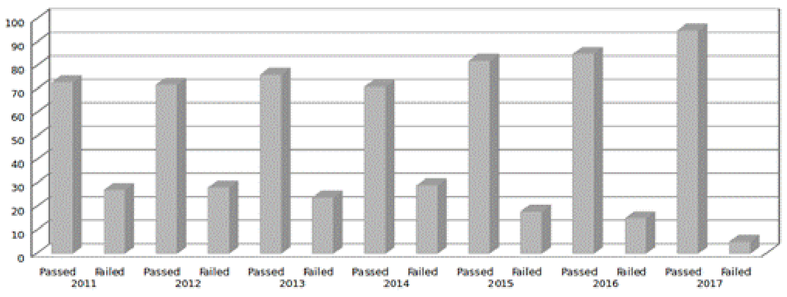


Figure 7

Figure 7. Final programming course marks

Additionally, we carried out a survey, which was given to 100 students from different engineering majors. Some students had previously failed the course and some had previously studied Java or other programming languages. With this in mind, the use of Java was compared with Python. Figure 8 describes the perception towards the use of Python in comparison to Java, in terms of level of difficulty. The results show that most students considered Python as an easier programming language than Java

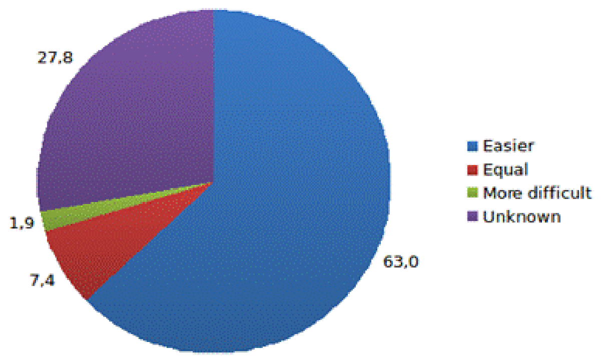


Figure 8
Figure 8. Python vs Java

5. CONCLUSIONS AND FUTURE WORKS

This article presented a teaching proposal for an introductory programming course. The proposal design aimed at improving students' academic performance and motivating them in the use and application of algorithms in different branches of engineering. The structure of the approach facilitates the integration of different perspectives from the pedagogical, technological, assessment and didactic points of view.

Teaching programming during the first year of a major can be a difficult and challenging task because most of the students generally find it difficult to learn the subject, which results in low academic performance and high dropout rates. The teaching approach proposed in this article was applied at Universidad del Valle, Colombia, obtaining positive results in recent years. After applying this approach, we succeeded in reducing dropout rates and increased student motivation. This was reflected in an improvement in students' grades and their perceptions of the course.

The teaching of computer programming depends not only on the programming language but also on other strategies that support the learning process, such as, the use of multimedia resources, project-oriented and problem-based learning, and assessment rubrics. Making use of these four perspectives can help students and professors to improve the learning process.

The results obtained suggest that Python is a more suitable programming language to teach an introductory programming course. This is due to its simple syntax, simplicity in code debugging, as well as its easy integration with other teaching tools.

Future work is planned to extend this teaching approach to other universities and academic centers in Colombia through the creation of a free-access platform in which all the material produced for this project will be available. We also propos to carry out an experiment with a first programming course using a blended learning methodology.

REFERENCES

Ali, A. and Smith, D. (2014) Teaching an Introductory Programming Language in a General Education Course, Proceedings of the 2008 International Conference on Frontiers in Education: Computer Science and Computer Engineering, FECS 2008, 13, pp. 236–240. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-62649122836&partnerID=40&md5=ed1302320017b959ae9f4c7e10972080.

Alturki, R. (2016). Measuring and improving student performance in an introductory programming course. Informatics in Education, 15(2), 183.

Aris, H. (2015, July). Improving students performance in introductory programming subject: A case study. In Computer Science & Education (ICCSE), 2015 10th International Conference on (pp. 657-662). IEEE.

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

Chan Mow, I. T. (2008). Issues and difficulties in teaching novice computer programming, Innovative Techniques in Instruction Technology, E-Learning, E-Assessment, and Education, pp. 199–204. doi: 10.1007/978-1-4020-8739-4-36.

Chapman, B. E. et al. (2015). Python as a First Programming Language for Biomedical Scientists, (Scipy), pp. 12–17.

Chien-An L. YU-Tzu L., and Cheng-Chih Wu (2015). ¿Which programming language should students learn first?. International Conference on Learning and Teaching in Computing Engineering, Taiwan.

Echeverría, L., Cobos, R., Machuca, L., and Claros, I. (2017). Using collaborative learning scenarios to teach programming to non-CS majors, Computer Applications in Engineering Education, (November 2016), pp. 719–731. doi: 10.1002/cae.21832.

Enbody R.J., Punch W. (2009) Python CS1 as preparation for C++ CS2. Conference Paper in ACM SIGCSE Bulletin·Chattanooga, Tennessee, USA

Furman B., Wertz E., (2010). A First Course in Computer Programming for Mechanical Engineers. Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications Pages 70-75.

Gomes, A. and Mendes, A. (2015). A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations, Proceedings - Frontiers in Education Conference, FIE, 2015–February(February). doi: 10.1109/FIE.2014.7044086.

Guo, P. (2014). Python is now the most popular introductory teaching language at top us universities, BLOG@ CACM, July, p. 47.

Hoffbeck P., Dilon H., Albright., Lu W., Doughty T. (2016) Teaching programming in the context of solving engineering problems. Frontiers in Education Conference (FIE), IEEE. Pennsilvania, USA.

Konecki, M. and Petrlic, M. (2014). Main problems of programming novices and the right course of action, in Central European Conference on Information and Intelligent Systems, p. 116.

Koulouri, T., Lauria, S. and Macredie, R. D. (2014). Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches, Trans. Comput. Educ., 14(4), p. 26:1–26:28. doi: 10.1145/2662412.

Machuca, L. and Solarte Pabón, O. (2016). Improving Student Performance in a First Programming Course, in Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, p. 367.

Mason, R., Cooper, G. and Raadt, M. De (2012). Trends in Introductory Programming Courses in Australian Universities – Languages, Environments and Pedagogy, 14th Australasian Computing Education Conference, pp. 33–42.

Pears, a et al. (2007). A survey of literature on the teaching of introductory programming, SIGCSE Bulletin, 39(4), pp. 204–223. doi: 10.1080/08993400500150747.

Salcedo, S. L. and Idrobo, A. M. O. (2011). New tools and methodologies for programming languages learning using the scribbler robot and Alice, Proceedings - Frontiers in Education Conference, FIE, pp. 1–6. doi: 10.1109/FIE.2011.6142923.

Soares, A. (2011). Problem based learning in introduction to programming courses, Journal of Computing Sciences in Colleges. Consortium for Computing Sciences in Colleges, 27(1), p. 36.

Stajano, F. (2000). Python in Education: Raising a Generation of Native Speakers, in 'Proceedings of the 8th International Python Conference'

Van Roy P., Armstrong J., Flatt M., and Magnusson (2003). The role of Language Paradigms in Teaching programming. SIGCSE February 19-23, 2003, Reno, Nevada, USA. ACM 1-58113-648-X/03/0002.

Vujošević-Janičić, M. and Tošić, D. (2008). The role of programming paradigms in the first programming courses, The Teaching of Mathematics. Društvo matematičara Srbije, (21), pp. 63–83.

Yadin, A. (2011). Reducing the dropout rate in an introductory programming course, ACM inroads. ACM, 2(4), pp. 71–76.

Zelle, J. M. (1999). Python as a First Programming Language, J. Comput. Sci. Coll., 29(6), pp. 153–154. Available at: http://dl.acm.org/citation.cfm?id=2602724.2602754.