SETTING UP A MODEL FOR STUDYING THE BASICS OF PROGRAMMING AT UNIVERSITIES USING THE PYTHON PROGRAMMING LANGUAGE

Dejan VIDUKA
Luka ILIC
Aleksandar ŠIJAN

***Abstract**: In this paper, we present a simple model that should lead to easier mastering of programming among students. The model is based on the study of one programming language, which should directly develop all the necessary knowledge of students and thus enable easier continuation of independent learning and integration into other programming languages. Through, we have processed the behavior of the Python programming language with algorithms but also with other popular programming languages such as Java and C ++. We only looked at the problem of programming through procedural and object-oriented programming using the Python programming language. The ultimate goal of the paper is to offer a new model that provides an effective approach to teaching students the basics of programming using a single programming language. The application of one programming language should lead to easier and deeper mastering of programming knowledge. In this paper, we recommend Python as an excellent choice for teaching object-oriented programming. Although often viewed as a "scripting" language, Python is a completely object-oriented language with a consistent object model and a rich set of built-in classes. Based on our experiences in teaching and working with students, we propose this model as a potential solution for faster and easier development of new skills in the field of programming.*
***Keywords:** basics of programming, Python, programming languages, C++i Java*

Introduction

Computer sciences integrate several different knowledge required in the field of data structures, algorithms, numerical analysis, programming methodology, simulations, etc. At the universities where experts of this

profile study, a number of programming languages are often used for education, including: C, C ++, C #, Java, PHP and Python. This model of programming learning can spend a lot of time constantly learning basic syntax instead of using every subject that is currently learning a new programming language to deepen and gain new knowledge and experience in the field of programming. If you compare this with learning foreign languages, which are usually one or two at most universities, then it is clear that this can be done with programming languages as well.

The professional public has accepted the Python programming language as a platform for learning programming since primary education because it provides a wide range of possibilities. The main advantage of this programming language is its simplicity in writing, and therefore in learning. Considering that most users have already encountered this programming language, then the knowledge acquired in this field can be improved at universities without a special need for new learning of syntax. New learning of syntax in various programming languages only helps the user to see new possibilities but not to raise them to a higher level when learning similar things throughout the semester as in other programming languages such as: variables, numbers, lists, loops, etc.

Python, on the other hand, has built-in data types that allow students to quickly create functional programs on their own. (Myers & Sethna, 2018)

Choosing a primary programming language would facilitate students' accelerated development, true in only one programming language but when one considers that logic, not just syntax knowledge, is primarily developed. It is characteristic of every programmer that he constantly develops and improves his knowledge on his own, for which he only initially needs help to penetrate as easily and deeply as possible into the core of the problem.

Many studies have highlighted the difficulties experienced by University students in learning programming. (Grandell et al., 2006) Some of these problems are encountered by the authors themselves in their work with students at the university and based on this knowledge they propose the following model that can facilitate learning and give greater competencies to students in further work after graduation.

About the Python programming language

Python is a powerful dynamic programming language used in a wide range of applications. (Python, 2022) Its high-level data structures and clear syntax make it an ideal first programming language (Downey, 2012) or a

language for easier merging of tools from different domains to solve complex problems. (Langtangen, 2006)

Python has a large number of standard libraries that can speed up the development of programs for various applications, which is why it is used by large global companies such as Google and Youtube. The programming language itself was developed as Open Source Software (OSS) and its source code is available to users for free use as well as for commercial use. It is also very well documented which makes it easier for users to further learn and improve. In addition to all of the above, Python is an interpreted programming language and as such allows students to analyze each individual command. (Kapanowski, 2010)

Python in education

In the last few years, there has been an increase in the use of (Radenski, 2006) Python in academic circles, which has led to a large number of textbooks in this field. (Gaddis, 2009)

The main culprit for this progress is its clean and simple syntax, which allows students to devote more time to learning the concept of program development, rather than the syntax itself.

In the meantime, in addition to educational institutions, Python has gained wide application in the industry itself, which separates it from programming languages such as: Pascal, Delhi, Visula Basic, which were previously used in education. Python programming language is a powerful and object-oriented programming (OOP) language, which makes it suitable for the development of serious programming solutions, but also for learning object-oriented paradigms. Some of the industries in which python is used are: robotics, multimedia, science, etc., which also have their use in education.

Influence of Python programming language on learning OOP paradigm

Our general reason for using Python is quite similar to those that use Python to teach procedural access: it allows for a greater emphasis on basic principles with less unwanted focusing on syntax. Our goal is steady progress in which each lesson is consistent with previous lessons. More effort is needed to make object awareness clear from the outset, rather than an unpleasant paradigm shift later in learning. With this we want to be able to use the same programming language for learning initial procedural programming as object-oriented in a simple and fast way. True, this is very

difficult to do in 15 weeks, as long as one semester lasts, so it would be good to separate these two paradigms, to learn the procedure first, and then object-oriented programming in two special subjects. Similar to the case of computer graphics, where it is recommended to study vector and raster graphics separately.

The first subject can be called the basics of programming and learn the dream elements of programming using the procedural paradigm. Another subject can be called Object Oriented Programming and it should deal with the object paradigm but using the same programming language in our case it is Python.

Python programming language and algorithms

Algorithms are one of the most important knowledge that an engineer should possess. They are used not only in writing care programs in all cases of solving engineering tasks. Precisely because of its use, this discipline is studied in all engineering and other fields at universities around the world.

Algorithms can be studied only by reading the literature and doing sets of tasks, this approach is not so popular among students, but the knowledge itself is smaller and less applicable. For that reason, it is common to learn algorithms with programming, which implements the acquired knowledge faster and easier into real projects.

In practice, there is a large amount of literature that uses algorithms to study programming languages such as Java, C ++ and C. It is often argued that these are programming languages that are familiar to students from work, so it is easier to master new materials. Unfortunately, experience has shown that algorithms are rarely studied with programming, but only the syntax of programming languages.

Algorithms themselves have several lines and are very easy to draw, while on the other hand in programming languages such as C or C ++ they are syntactically defined in many lines and thus difficult to follow. In the case of Python, which is very optimal with syntax code, it is easier to define in less lines and more complex algorithms, and thus easier to follow and learn. This ability makes it suitable and desirable for learning algorithms in the educational process. Second, Python provides basic data structures such as lists, torques, and dictionaries that algorithms can use directly. Even more complex data structures such as trees and graphs can also be expressed in Python in a concise, human-readable form, without the need to reinvent these data structures. (Chou, 2002)

Comparison of Python with Java and C ++ programming languages

In practice, programming languages such as Java and C ++ are very common in education. They are used from the ground up, advanced programming and as a testing ground for mastering the OOP paradigm. All that can be learned in education as a material for mastering programming can be done with the Python programming language in an easier and faster way. What makes the python even more attractive for the basics is that it enables transition to other languages due to small differences such as: marking block structures, primitive names, use of I / O, etc.

The different object models of Java and C ++ are new to students migrating from Python, while migrating from Python to Java could be easier. Python's models for identification, information transfer, and symmetry are assigned in accordance with Java's reference model for object types. Both languages rely on the collection of surpluses in terms of protecting programs from poor memory management. On the other hand, the transition from Python to C ++ is much more extreme due to the complexity of the C ++ programming language. In addition, the difference between static and dynamic memory allocation places more responsibility on C ++ programmers. (Goldwasser & Letscher, 2008)

Model proposal

This model is very simple and does not require major changes in administrative form. For this model, it is necessary to define two subjects, and if desired, the third.

- The first subject would be called Fundamentals of Programming and would study the basic building blocks of a programming language. In our model, that programming language for all three subjects would be Python. This course would be studied in the first semester of the first year and in this way students would be introduced to the basics of programming and pre-procedural programming.
- The second subject would be called Object Oriented Programming (OOP) and would deal with, as its name suggests, object programming also in the Python programming language, which would upgrade the acquired knowledge from the first year. This subject would be studied in the first semester of the second year or in the second semester of the first year, depending on the duration of studies (6 or 8 semesters)
- The third and recommended subject could be called the Practicum in Applied Programming, which would combine the acquired knowledge

of students from both previously mentioned subjects that they would apply in independent (mentoring) work. This course would be studied in the second year in the second semester or in the case of a study length of 6 semesters this course would be taught in the first semester of the second semester.

This would complete the learning of programming with practical work. This model avoids constant learning of new syntax and possibilities of various programming languages, but time would be wisely used to acquire new and deeper knowledge in programming.

Discussion

As you can already see from the paper, Python has many advantages for acquiring the basics. Applying this model gives you time to learn and better master programming, and not just a great variety of knowledge of programming languages. The model takes only the first two years, which leaves enough space for the third and possibly the fourth year to then do advanced programming in programming languages such as C ++ and Java. The third subject is recommended because it achieves greater involvement of students in software development, and thus their satisfaction with learning. Other models require a lot of work from students to constantly master the basics, and it is enough to learn everything in one programming language and then only upgrade knowledge from other programming languages if and when students need it. Our task in education is to teach students to think and encourage the use of their logic in solving certain problems, not learning syntax and solving math problems as standard batteries of tests to test their skills. It often happens that students master the material, and when it is necessary to apply it, they do not know how to apply that knowledge. Just as mathematicians practice tasks, and chess players various chess combinations, so programmers need to work on real projects that will make their knowledge more applicable. This approach would be more appreciated by the students themselves and by future employers.

Conclusion

The model based on the previous analysis has a great possibility of success if it is accepted by educational institutions. The task is to teach students to program and all the time that is available to professors in the

curriculum should be used for new learning and deepening knowledge. This model can be done with other mentioned programming languages, but Python has more advantages over other languages, and the learning effects are equal or greater when you consider the simplicity of Python.

Applied courses are rarely done at academic levels, so the third proposed course is a great opportunity for students to finally do an independent task in a real environment with acquired knowledge, and it is certain that during the process they will upgrade their knowledge depending on project needs. This achieves exactly what we talked about at the beginning of the paper, and that is that students need to learn to learn independently and upgrade their knowledge. For independent learning, students should use available literature, Internet sources as well as the commune gathered around the Python programming language, but also exchange knowledge with other students, assistants and professors. From the offered model, it is clear that students do not need ten programming languages to learn programming, but let's say in our model, one through three subjects. This is quite enough to learn programming and to get a job in the economy later, which is the ultimate goal of the educational process. The application of other programming languages is possible in accordance with the study program such as: internet programming, mobile programming, etc. These subjects study specific branches of programming application, and our model talks about acquiring knowledge about the basics of programming regardless of its application.

LITERATURE

1. Chou P. H. (2002) Algorithm Education in Python, Proceedings of the Python 10 Conference, Alexandria, VA.
2. Downey A. B. (2012) Think Python, An Introduction to Software Design, http://www.thinkpython.com/.
3. Kapanowski A. (2010) Python for education: the exact cover problem, Online source: https://arxiv.org/abs/1010.5890v1, (Access 14.03.2022.).
4. Gaddis T. (2009) Starting Out with Python. Addison-Wesley.
5. Goldwasser M. H. and Letscher D. (2008) Teaching an Object-Oriented CS1 — with Python, ITiCSE'08, Madrid, Spain.
6. Grandell L., Peltomäki M., Back R. J. and Salakoski T. (2006) Why Complicate Things? Introducing Programming in High School Using Python, Academia.
7. Langtangen H. P. (2006) Python Scripting for Computational Science, Series: Text in Computational Science and Engineering, Vol. 3, 2nd Edition, Springer-Verlag Berlin Heidelberg.
8. Myers C. R. and Sethna J. P. (2018) Python for Education: Computational Methods for Nonlinear Systems, Online source: https://arxiv.org/abs/0704.3182v1, (Access 15.03.2022.)

9. Python Programming Language, http://www.python.org/. (Access: 03.03.2022.)
10. Radenski A. (2006) "Python first": A lab-based digital introduction to computer science. In Proc. 11th Annual Conf. on Innovation and Technology in Computer Science (ITiCSE), pages 197–201, Bologna, Italy.

NOTES ON THE AUTHORS

Dejan VIDUKA was born in Osijek, Croatia in July 30, 1980. Currently he is a Assistant professor of Computer Science Department at the Faculty for Applied Management, Economics and Finance, Belgrade, University Business Academy, Novi Sad, Serbia. Having 23 years of experience at IT sector (hardware, software, SEO, Internet, e-business, e-marketing, operating systems, CMS systems and Open Source systems) and extensive experience in the development of Open Source projects, and is also a member of Elxis Community and provides expert assistance to customers from Croatia and Serbia. As an IT expert, he worked on many projects for well-known clients from Serbia and from abroad. *dejan.viduka@mef.edu.rs.*

Luka ILIĆ was born in 1998 in Bijeljina, Republika Srpska. Since 2014 he has been working as freelancer in information technology. Graduated from the Faculty of applied management, economics and finance (MEF), department of Information technology. Currenlty he is pursuing PhD title at Faculty of Electronic Engineering, University of Nis. Works as a teaching assistant at MEF faculty and as a freelancer in web technologies. He has an interest in technology, sports and adventure. *luka.ilic@mef.edu.rs.*

Aleksandar ŠIJAN was born on November 29th, 1990 in Rijeka. He graduated from the ICT College in Belgrade summa cum laude and got the title Engineer of Electrical and Computer Engineering – BSc (appl.). After that he graduated from the Faculty of Applied Management, Economics and Finance (University of Business Academy Novi Sad) and earned the title of Bsc (Hons) in information technology. He completed his master's thesis in 2019 at the same University and acquired the title Msc in information technology. Currenlty he is pursuing PhD title at Faculty of Electronic Engineering, University of Nis. He is a former scholarship holder of the Ministry of Education, Science and Technological Development of the Republic of Serbia. He is a member of the DevTeam and a Teaching Fellow on the Faculty of Applied Management, Economics and Finance. As a freelancer, he has been worked on numerous projects since 2011. In addition to English, he understands Norwegian language. *aleksandar@mef.edu.rs.*