IMI Python: Upgraded CS Circles web-based Python course

Marija Đokić[1], Miloš Ivanović[1], David Pritchard[2] and Vladimir Cvjetković[1]
[1] *Department of Mathematics and Informatics, Faculty of science, University of Kragujevac, Serbia*
[2]*Centre for Education in Mathematics and Computing, University of Waterloo, Canada*[*]
E-mail: m.djokic@kg.ac.rs, mivanovic@kg.ac.rs, dagpritchard@uwaterloo.ca, vladimir@kg.ac.rs

Abstract

The rapid growth of student demand for flexible education and learning alternatives has caused a significant increase in web-based programming course offerings. In order to ensure easy and enjoyable ways of acquiring knowledge, many web-based solutions have customized the design and content to student needs. This paper introduces a project of the Institute for Mathematics and Informatics (IMI) called *IMI Python*, an interactive online course. It is based on the open-source Computer Science Circles (CS Circles) project. IMI Python aims to assist the target audience, primarily students, learn a spectrum of Python knowledge. The benefits of this enhanced system are multiple, both for students and their teachers. The course content is structured and divided by levels: basic, medium and advanced. Flexible navigation through the different levels of difficulty and lesson units allows students to easily review any forgotten material and adopt new knowledge. Teachers have the ability to follow the progress of individual students or all students in a level, and communicate with them about their work. Teachers and students can communicate within the system to discuss individual exercises through a simple user interface. The system is enhanced with the possibility of testing students' knowledge through quizzes. Quizzes are visible at assigned time intervals and are worth a certain number of points. By tracking students' results, teachers can determine whether the site has enough quality material and what can contribute to its improvement.

1. Introduction

In the last decade, web-based programming courses became a major trend in many institutions of higher education [1]. Faced with the rapid growth of demand, most courses are traditional frameworks, either just static information on a webpage, or exercises without feedback. Many online approaches try to innovate in the educational sense; they are improving their demonstrations of programming techniques by including examples, exercises with feedback, and environments that adapt to users' needs. But, some problems are still common [2]. Students can benefit from a richer approach that includes interaction between students and teachers. Following the course is difficult when there is a separation between lesson content and exercises; following all the details is quite hard and laborious work and requires constant student attention in the flood of information. Re-reading prior content before writing solutions to exercises also diverts student concentration, and is an obstacle to wholly understanding the programming language.

In order to overcome these deficiencies, and minimize the difficulties in the learning process, we created an improved web-based programming course called IMI Python (named after our Institute for Mathematics and Informatics). It is publicly available for free[1], and we also use it in our classes. The system software is based on Computer Science Circles (CS Circles) [3], an open-source interactive platform for learning Python. We have

[*] Work performed while located at U. Southern California. Currently located at Google Los Angeles.
[1] http://147.91.205.71/wordpress

extended and built upon its infrastructure. Previously, in the IMI, Python was taught in a workshop[2] aimed at Python beginners..The workshop activities were based on presentations and solving tasks using Python console. In this environment students generally have to choose between capturing notes from and listening to the teacher. We noticed that students lose focus when the teacher is explaining the material and they take notes at the same time. Also, the learning was rather isolated because the students were taking notes instead of participating in lessons activities. During course, we concluded that the level of students' understanding was seriously affected by the quality of their notes. If notes contain transliteration mistakes, the student is cramped from correct learning and achieving effectiveness in the lesson's topic. As well, the teaching units have to be finished in the prescribed period, which makes it difficult for us to present more complex subjects. Often, a lack of time prevents students from providing feedback on what they learned and what they do not understand.. As a result, students are not inspired and motivated to participate in the learning activities and deepen their understanding.

However, we wanted to promote Python in a more sophisticated way. Based on the issues mentioned above we decided to experiment with a new teaching approach, and we introduced IMI Python as the mechanism for doing so. Our primary target group was students from the academic years 2014–2015. From three degree programs (MCSI: Module Computer Science and Informatics, MPM: Module Professor of Mathematics, MPI: Module Professor of Informatics), 66 students were enrolled in our Python course. Before the course starting, 82% of the students were total beginners, while the others had experience with Python through extracurricular activities.

Bearing in mind that CS Circles has high traffic and good user reviews of content and interactivity, IMI Python adopted the CS Circles source code as a starting point. CS Circles is based on WordPress, a widely-used content management system with a robust plug-in system and a large user community. CS Circles allows in-browser code editing, automated programming assessment, and other features. A number of CS Circles features have been modified, upgraded and customized in order to make a better interactive learning system designed for Python.

One of the challenges that we faced was to adjust the content of CS Circles to the Serbian language. The main motivation for this step was the deficiency of Python web-based courses in Serbian. To the best of our knowledge, the only web-based Python material in Serbian language is offered by Faculty of Mathematics, University of Belgrade. Its Python course material is a freely available[3] and intended for beginners. Although the content is well-constructed, this system has one substantial shortcoming, namely its lack of interactivity. We try to compensate for this deficiency and provide the devotees of Python in Serbian-speaking regions with a high-quality and interactive system.

1.1 Contributions of this work

Adhering to the philosophy of the CS Circles source code, we have sought to expand and enable the original website with new features. These features represent very interesting improvements which aim to alleviate the monotony of the course and make it more attractive for users. IMI Python is enriched with the following features:

- Creating lessons within different difficulty levels: basic, medium, and advanced.
- Navigating through the levels.
- Creating exams (each associated with one difficulty level). The exams are visible to students only during a specified time interval. Each exam problem is worth a certain number of points.
- Monitoring one or all students' learning progress for a level.
- Monitoring one or all students' exam progress for a level.
- Communication with students.

---

[2] http://imi.pmf.kg.ac.rs/moodle/course/view.php?id=288

[3] http://www.edusoft.matf.bg.ac.rs/python/

- Filtering the progress of students according to level and date of account creation.
- A menu item that shows, on the basis of tasks completed, active and uncompleted lessons for each level.
- The possibility for users to choose independent learning or supervision of a mentor.

To begin, the paper gives the overview of the system architecture, in Section 3. It is similar to the CS Circles architecture, with a bigger focus on teachers as a separate class of users, in order to support collaborative course development and administration. In Section 4, we explain the material that is contained in our course. The material that we wrote consisted of both the narrative parts where we introduced and explained features of Python, and also the exercises that are embedded into the lessons and help students solidify their understanding and practice.

In Section 5, we go into detail about the new features added to our course. These infrastructural improvements took a large fraction of the overall effort of this project, but we think are worthwhile, since every new feature only needs to be implemented once and then can be re-used over and over in our site. General goals for these improvements included (a) keeping an integrated look and feel to reduce distraction, (b) allowing for evaluation of students rather than just facultative exercises, and (c) making the tracking of students more feasible in order to support our instructors.

We supplemented our project by sending our students a survey with a variety of questions about their experiences. We provide this along with some demographic and academic statistics in Section 6. We close in Section 7 with a discussion of possible directions for future work.

2. Related work

Web-based programming courses have become a popular and important issue over the past years. Several critical features, such as improvement of students' stimulation and motivation, methods for enriching learning, and facilities to assist stuck students remain an important focus.

Unlike a textbook, an online course can provide a student with feedback on their work, which is crucially beneficial to their development [4]. In classically-taught courses, providing personalized feedback for all students is a time-consuming and error-prone process for teachers, even though it is very essential for helping improve student achievement. Further, *instant* feedback can motivate students to perfect their work more quickly. Development of automated feedback has also helped unburden teachers and reduce their workload. There are many web-based tools and courses that utilize automated feedback such as OTO [5], CourseMarker [6], PASS [7], BOSS [8], etc. Like these, IMI Python allows students to submit their problem solutions and receive instant and detailed feedback from the compiler and auto-grader.

Although it is a notable feature, automatic grading is not a substitution for expert feedback from a real teacher. Monitoring of students' work plays a significant pedagogical role [9]. Many courses resort to communication system and forums [10, 11, 12]. A messaging system can allow teachers to monitor students' progress in order to provide more comprehensive one-on-one feedback on problems that are generating difficulties. IMI Python offers such a messaging system and alerts student to communicate through the "Help" button. Messages sent in this way automatically include the code from students' editor of the stuck problem; then the message is sent to the selected mentor.

The process of learning programming is a demanding task, especially for beginners. Due to this fact, students need to be adequately motivated in order to learn programming in a successful and effective manner. One great characteristic for improving the learning process is providing a set of problems with varying levels of difficulty [7]. Different groups of students with different knowledge levels should be able to easily focus on appropriate learning materials. This possibility can ameliorate student performance and satisfaction. Students with less programming

experience can approach easier problems and become motivated to solve harder problems later [12]. In accordance with this detail, we decided to offer 3 different levels of Python course: basic, medium and advanced.

One popular feature of modern web-based programming learning is quizzes. They are intended for assessment, including self-assessment. This is the case with Mooshak [13]. In order to improve competitiveness, the system demonstrates ranking of students according to the number of problems solved. IMI Python uses a similar approach while trying to assess the knowledge of students. For authors in [14] self-assessment is an innovative method. Their approach consists of black-box quizzes, through students may test their programs simultaneously against the instructor's code, comparing both outputs. In our case students do not receive immediate information on whether they passed quizzes. We believe that this option has a pedagogical role. That could influence to students to discuss their assignments, share solutions and ideas before teacher responds with their results. If they would have known the results of the test immediately, their level of interest would have declined. An interesting possibility is to organize team competitions; an environment with a mix of lessons and quizzes is more constructive for learning programming and acquisition of teamwork skills [15]. The EduJudge system [12] is one system offering this feature.

The rapid explosion of technology has had influences on the evolution of learning. A web environment with an effective user interface (UI) is one of the critical issues for courses' success. A well-designed UI can help students increase their self-confidence and produce better problem solutions. Many open educational websites have proposed great frameworks that pay close attention to offering better learning environments. With MOOCs (Mass Online Open Courses) [16] the student's complete interaction with the course takes place on the web. MOOCs that are very similar to IMI Python include Khan Academy CS[4] and Codecademy[5]. Khan Academy CS stands out for using a non-traditional coding environment. In place of real teachers it has video contents with live editable code replays. Narration allows students to follow the material at a speed that matches their ability. Using automatically-generated suggestions for every syntax error, Khan Academy CS attempts to help students learn programming. Like our system, code can be re-compiled and re-executed, though it is continuous on Khan Academy. Like IMI Python, the most recently submitted version of the code is automatically loaded into the text editor. Harder levels for some programming languages in Khan Academy CS also exist, but you cannot navigate freely between levels like in IMI Python.

Codecademy, with embedded exercises in lessons, gives students a sense of continuity in progressing. Two main Codecademy defects are a lack of hints and occasional inappropriate grader feedback for failing student code. Given these shortcomings, IMI Python is trying to effectively interconnect students' needs and technology preferences. It offers a much richer environment through a patchwork of different Python levels and diverse processes of student monitoring.

Today's dominant e-learning systems use Learning Management Systems (LMS) or Content Management Systems (CMS) to improve educational process and to create interactive environments. One of the most popular CMS is WordPress[6]. It aimed at building websites and blogs, without a focus on education per se. There are a lot of different options emerging in the WordPress space for e-learning. For example, WordPress has been used to help develop a virtual campus for Biomedical Engineering [17]. IMI Python builds a small framework for a Learning Management System (LMS) on top of WordPress (as well as other CS-specific features). In comparison, there are other systems designed from the start as an LMS. Some of them are not open-source, such as Blackboard and Desire2Learn, and therefore are not suitable for us to develop on top of. Moodle is a well-known open-source LMS. It is certainly possible that for deployment across an entire university, Moodle might be better, since it has powerful administration features. In fact there was an investigation [18] into extending Moodle for CS education. However,

---

[4] https://www.khanacademy.org/computing

[5] http://www.codecademy.com/

[6] wordpress.org

for our limited setting where the administration of the site was done by computer science instructors, the relatively lightweight WordPress setup suited us well, as it supports experimentation and feature development without the need for a large rigorous framework.

The above-mentioned solutions emphasize that it is very difficult to find a system that successfully incorporates all mentioned features and satisfies all different needs. With a combination of automated feedback, mentor feedback, different levels of programming and monitoring of students' progress, IMI Python has potential as a versatile and robust learning environment. Through the contents of the lessons, embedded exercises, as well as the communication system, IMI Python has the ability to meet students' and teachers' requirements to a higher degree.

3. IMI Python architecture

The principal objective of IMI Python is to help students acquire basic knowledge about Python. As mentioned earlier, lessons and quizzes can be either intended for absolute beginners, or for those who want to expand their Python knowledge. This helps students achieve their learning goals efficiently and effectively. Teachers also have privileges to easily create lessons as well as quizzes, and monitor the progress of students. A short behavioural specification of IMI Python is given below.

- Registered students have access to all course material. They have freedom in a selecting the lesson unit and course level (basic, medium, or advanced). They can interact with the system through the help of their mentor (teacher). They can send messages about any problem where they are stuck. Registered students can access quizzes. Students can follow their own progress on the current level. The details related to sessions and the student's interactions with learning objects are stored in a relational database.

- Unregistered students have significant limitations compared to registered ones. First of all, they don't have a mentor, therefore they are not able to ask questions through the communication system and the "Help" button. This lack of interaction can be unpleasant if a student has difficulty in solving problems. They also don't have access to quizzes. Also, they are not able to keep track of their progress or review old submissions. There is no recording of their interactions in the relational database.

- Teachers (mentors) have all privileges of the system. They can create lesson units and quizzes. As well, they can monitor the progress of their registered students. In communication with them, they can offer possible help in solving problems. They have access to the database and therefore the absolute control over all functionalities of the system.

The implementation was performed by the first, second and fourth authors, supported by the third author in troubleshooting. We used a three-layer architecture to implement IMI Python. Recall that IMI Python is built on CS Circles, which is in turn built on top of the WordPress content management system. Since WordPress has an easy web-based interface for content editing and management, a built-in secure login system, a well-maintained and easy-to-modify code base as well as a great plug-in system, we inherited the entirety of the CS Circles system. We expanded it and adapted IMI Python to the new needs. Figure 1 shows the IMI Python architecture. The first layer is the user interface, through which the students and teachers interact with the system.

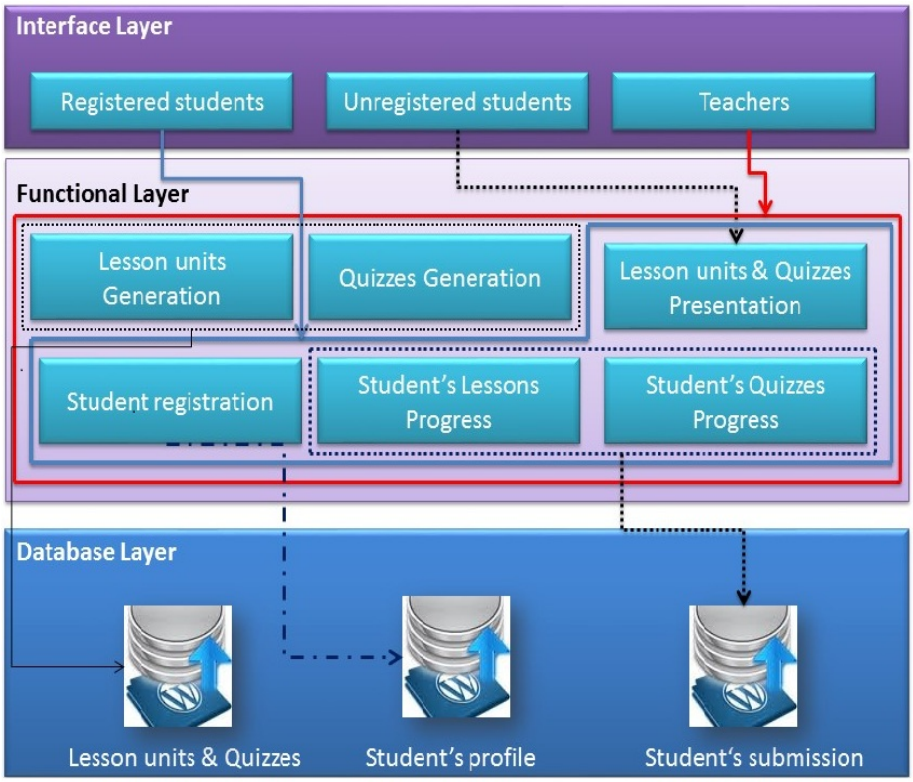


Figure 1: The IMI Python architecture

The functional modules mainly constitute the second layer. The various functions, like student registration, generation of lesson units and quizzes, presentation of content material, and saving student progress are taken care of by this level of architecture.

All data, such as information about students, lessons, quizzes and students submissions, are recorded into the relational database. It constitutes the third architectural layer. The most important part of the database is the information related to the course material – lesson units and quizzes, which is elaborated in the next section.

4. Course material

Recall that the course material and quizzes are divided in three levels – basic, medium and advanced. The lessons of the basic level help students become familiar with the fundamentals of the Python programming language. This course level has 20 lesson units. About 70% of the lessons were inspired by the content of CS Circles, which is Creative Commons-licensed. Our content begins with the archetypical "Hello, world!" program, including a working click-to-run program and as well as a crashing one. We continue through variables, function calls, comments, string literals and types of data. Some lessons explain errors, design and debugging. We have lessons connected to mathematical functions and geometry. The final lesson discusses basic object oriented Python programming. The main objective of our lessons is that the student will have enough knowledge to write or understand an elementary Python program. Also, we are trying to develop student self-confidence and problem-solving skills.

The content of the lessons and exercises must be at an appropriate level of challenge. Problems must not be too easy, because students need to stay motivated to learn and expand their knowledge; and they must not be too hard, lest a student become frustrated. Allowing students to select their own difficulty level aids students in this way.

Figure 2 shows an example program about mathematical operations: exponentiation, division and modulus. The program prints out a message showing the results of these operations. When the student submits program code, it is compiled, executed and graded server-side. Next, without a page reload, the grading box is updated according to the result of grading. The bottom of Figure 2 shows the result of a successful run for this example.

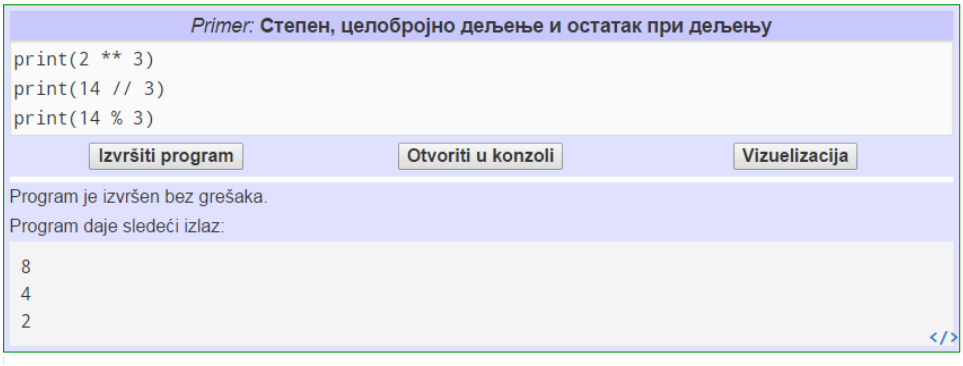


Figure 2: Basic Python example. The leftmost button, "Execute program," has been pressed; the results, in the bottom half, read "The program executed with no errors. It produced the following output:"

The course materials (infrastructure, lessons, exercises, and quizzes) were written by the first, second, and fourth authors and a student from the MSCI degree program who satisfied certain requirements (they had knowledge of Python and SQL, and previously took the Methodology of Teaching course). The evaluation of the system was then performed in two main phases. In the first phase, authors performed module and link testing in order to find possible errors and correct them. The second phase included additional students from the MSCI profile to try out the system. They gave valuable comments and suggestions, which helped to improve the system and make the user interface more effective and user-friendly.

5. Upgraded features

The content of IMI Python was built with the tools described in the following section. Starting from the base of CS Circles, we created, customized and upgraded features that we felt would make a better interactive learning system. By analysing and considering the students' requirements, we developed the following features, which we believe can help student engagement and reduce monotony. The features are described in the following sub-sections, organized according to tools for students, teachers, and authors.

5.1 Student tools

One of the major barriers to an effective programming course is closely connected with various students' background and ability level. In order to meet the needs of students with different backgrounds, we designed lessons and exercises with various levels of difficulty: basic, medium and advanced. The less capable students may try out the basic level. More talented students can directly access the medium or advanced levels. In addition, flexible design allows students to scroll through lessons of the same level, but also transition to an easier or harder level. In this way, students can pick lessons or exercises that best match their needs.

In our system, students can follow their own progress using the "My progress" webpage. It keeps a history of exercises, the most recently submitted code, and a hyperlinked list of completed and uncompleted exercises for the current level (see Figure 3). With these links students can visit new problems or previously-skipped ones.

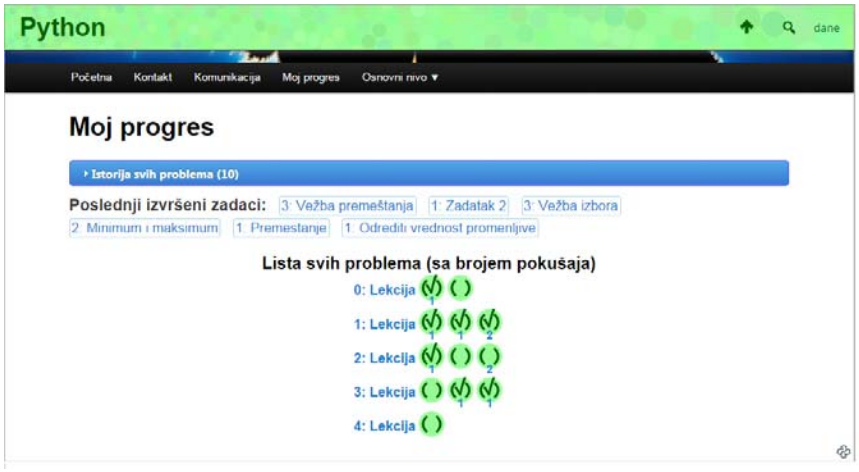


Figure 3: Progress page of IMI Python student. The two bold subsections show "Recently completed problems" and "List of all problems (with number of attempts)."

We did find one usability issue in the "My Progress" page from CS Circles. Visiting this page means abandoning the current page, which can be exhausting. To avoid this shortcoming, we propose a new idea. We added an item positioned in the main menu that represents a summary of the current level's lesson units, showing which ones are fully complete or not. For example, Figure 4 shows that the student is currently located on a basic level, at Lesson 2, which is denoted with . Lessons with all exercises completed are indicated with , while not-yet-complete lessons have a label. A similar solution is offered by Codecademy, but it doesn't allow free navigability back and forth between lessons and levels. And, this is crucial! Though most exercises focus on notions developed in the current lesson, it is inevitable that some lessons and exercises build on content taught previously. Our experience is that not only students benefit from this flexible navigation, but so do teachers: they are pressed for time and benefit from students more easily accessing earlier concepts for review.

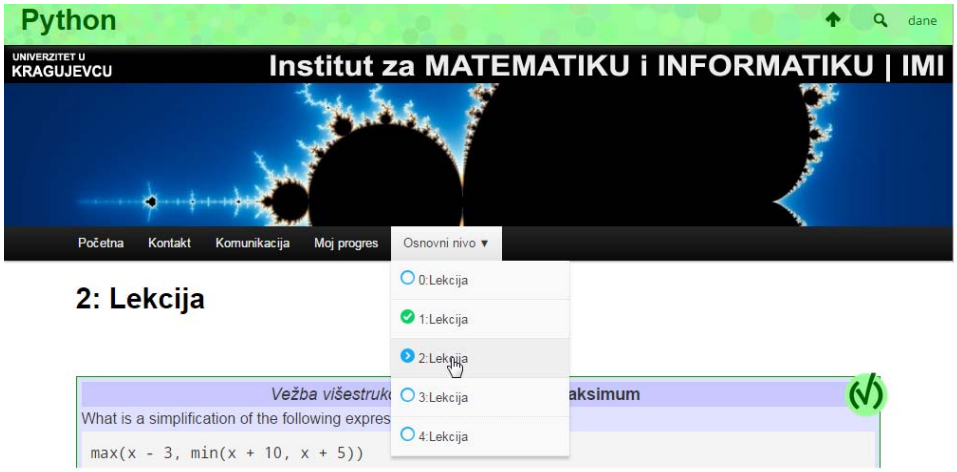


Figure 4: Students' progress in the current level is visible in the main menu. The menu label "Osnovni nivo" means "Basic level" and the student is currently viewing lesson 2.

In web-based programming courses, it's not simple to support students' cognitive skills if learning actions and learning through assistance are not well integrated. In CS Circles, registered users have the possibility to select another user from the system as their "Guru," or assistant. However, not all students may know someone with strong

Python skills who can act in this capacity. Therefore, we decided to foster a traditional student-teacher relationship: the teachers (administrators) are given the role of mentor, by default. A related IMI Python feature that is useful is the ability to change the mentor at any time, which is important if there is a change in the teaching staff or other unforeseen circumstances. Alternatively, as in CS Circles, registered users can choose to learn independently without any guru. This independent way of learning can be extremely important for a wide audience, such as users from the general public who stumble upon the site from outside of the university.

As in many web-based courses, the messaging communication system plays an important role. It offers a unique opportunity to promote a reflective, collaborative and interactive way of learning. IMI Python retains the messaging system from CS Circles. The "Help" button allows registered students to request assistance from their mentor (see Figure 5).

Figure 5: In this exercise the Help button, labelled "Pomoć," has been pressed, revealing a message box where the student can explain where they are stuck. Once done they can click "Pošaljite poruku," meaning "Send message."

The students use this feature when they are stuck. A message is sent to the mentor, together with the latest version of the code and the student's short description of the issue. This option is disabled when students are solving quizzes, so that we can try to verify the students' knowledge without the help of a mentor. All past messages can be found on the "Communication" page (see Figure 9).

One of the debugging tools available in IMI Python is the "console". It allows the student to run arbitrary code without any grader interference. In CS Circles, there is a button on each example, which allows the code to be copied to the console. The console is opened within a new browser tab, with a single click. In CS Circles, the console's history of submissions is automatically stored, but bearing in a mind that students can arbitrarily and continuously test their code, we found that the console's history was too large and of little importance, often representing attempts by students to test their code. Aiming to reduce the size of the IMI Python database, the option for saving console history was abolished. Additionally, we changed the console UI so that it now opens in a pop-up window, reducing distraction (see Figure 6). We implemented this feature using the Anything Popup[7] plug-in. We retained the usage of CodeMirror[8] in CS Circles, which highlights syntax, enumerates lines of code, matches parentheses as the user types and performs smart indentation.

[7] https://wordpress.org/plugins/anything-popup/
[8] http://codemirror.net/

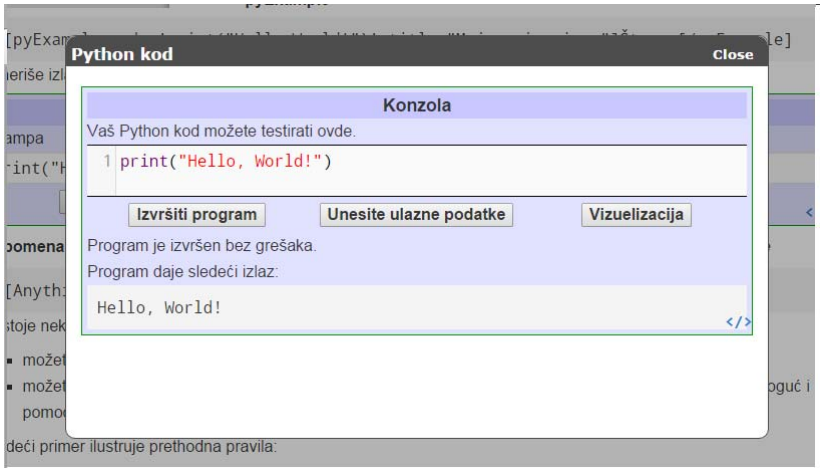


Figure 6: The console, in a pop-up window. Any code can be typed in here. The three buttons say "Execute program," "Enter test input," and "Visualize."

A main challenge in web-based courses is getting students to understand how source code maps to highly dynamic program execution. Like CS Circles, we use the Python Visualizer from Online Python Tutor [19] in order to help students understand the details of a running program. However, CS Circles put the visualizer on a page with a totally different look and feel; so to unify and simplify the user interface, we moved the "Visualization" into a page formatted like the main site.

5.2 Teacher tools

Maximizing the efficiency of teaching is one of the primary goals of IMI Python. In order to achieve our chief educational goals, one major teaching strategy is the introduction of multiple levels of difficulty. But implementing this required changes to tutoring, assessment, and the process of monitoring and communication with students. This section also mentions changes made to the CS Circles infrastructure in order to accommodate scored quizzes.

Teachers can actively monitor the progress of their students, using the “Students' progress” page. This web page was already a part of CS Circles, but introducing multiple levels of difficulty required some infrastructure modifications. Several factors affect the user interface: the modified page is affected by the choice of difficulty levels; the level determines the selection of exercises; and a teacher can be a mentor to a large number of students. Bearing this in mind, the ability to select one out of a large number of registered students was absolutely necessary. Filtering by date of account creation was the most practical way to select a student from the current semester. In Table 1 we show how the teacher choices affect the user interface. For example, Figure 7 represents the second combination from Table 1.

Combination	Page content
• All students in a given level • All exercises	• History of submissions • Hyperlinked lists of individual students’ progress • The list of exercises with the number of solved exercises
• All students in a given level • One exercise	• History of submissions and solutions of students for selected exercise • Hyperlinked lists of individual students’ submissions
• One student in a given level • All exercises	• History of submissions • Last executed exercises of selected level • The list of problems with the number of submissions
• One student in a given level • One exercise	• History of submissions of exercise • Last executed exercises of selected level • The list of problems with the number of submissions

Table 1: All possible user interface options for the “Progress” page.

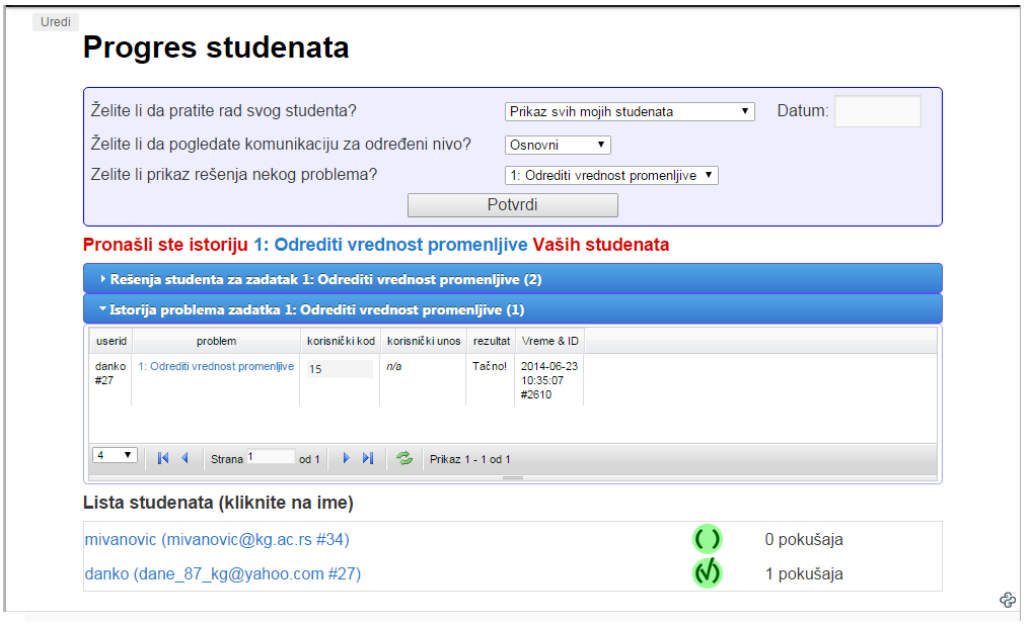


Figure 7: This is an instructor’s view of the “Progress” page. They have selected to view all attempts by students in the basic level for an exercise in lesson 1. The page shows all past solutions, and all past attempts, and at the bottom, the number of attempts (pokušaja) per student.

Students who have difficulty in solving exercises can communicate with their mentor using the communication system. Messages which students send using the “Help” button are stored in the relational database. Teachers can manage messages through a simple user interface on the “Communication” page. This web page is similar in nature to the “Students' progress” page. While the “Students' progress” page enables monitoring of student groups, the “Communication” page supports the review of only one student. In Table 2 we show the possible options affecting the user interface. For example, Figure 8 represents the first combination from Table 2.

Combination	Page content
• One student in a given level • All exercises	• All messages from selected students and selected level • All messages for selected level
• One student in a given level • One exercise	• All messages from selected students and selected exercise • All messages for selected exercise

Table 2: All possible combinations of UI selections for the “Mail” page.

Figure8: The “Mail” page, as seen by an instructor view. They can filter by student, level or problem. The open grid panel in the bottom half lists “All past messages about this student’s work.”

The next aspect of our teaching philosophy is the quiz feature. Online quizzes are a primary tool for assessment and self-assessment of students’ knowledge; this is the main reason for their place in IMI Python. Quizzes are time-limited and they are executed during one teaching class. Each exercise in a quiz is worth a certain number of points. This is achieved by a small modification to the CS Circles shortcode infrastructure for defining exercises: a new attribute called “points” is added. See Figure 9(b).

```
[pyBox
placeholder="Ovde otkucajte Vaš kod"
grader="@file:graders/warmup.py"
answer=REDACTED
slug="0.zadatak1"
title="Zadatak 1"
rows="2"
]
Napisati program koji ispisuje
<pre>Ovo je prva lekcija!</pre>
[/pyBox]
```

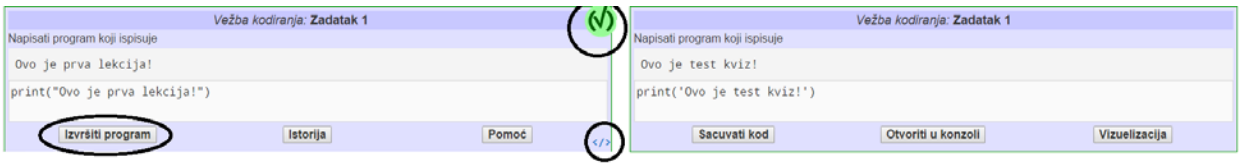
(a) Lesson exercise shortcode (definition)

```
[pyBox
points="5"
placeholder="Ovde otkucajte Vaš kod"
grader="@file:graders/warmup.py"
answer=REDACTED
slug="0.test1zadatak1"
title="Zadatak 1"
rows="2"
]
Napisati program koji ispisuje
<pre>Ovo je test kviz!</pre>
[/pyBox]
```

(b) Quiz exercise shortcode

Figure 9: Shortcodes to define exercises, showing the new field needed for quizzes.

Every exercise format that is available in normal lessons (programming, code scramble, short answer, multiple choice) can be used in a quiz. However, there are differences in their design. Exercises in quizzes don't have a “Run code” or “Checked” button; this means that the student doesn't know if he has done the exercise correctly during the running quiz. If they wish to test it, they must use the “Open in console” and “Visualization” tools, entering their own test cases. Once done, students use the “Save exercise code” button. They can make an unlimited number of submissions. After termination of the quiz, the teacher informs each students about the quality of their work.



a) Lesson design exercise b) Quiz design exercise

Unlike registered students, unregistered students cannot access quizzes; they are only intended for IMI students.

When the time allotted for the execution of the quiz has been finished, teachers can check who passed the quiz based on the number of points, which we implemented with a new "Tests" page, where monitoring can be performed in several ways (see Table 3 for a list of user interface possibilities). Figure 10 demonstrates the second combination from Table 3.

Combination	Page content
• All students in a given level • All tests and all exercises	• List of hyperlinked students • Number of points for each quizzes
• All students in a given level • Selected test • All exercises	• List of hyperlinked students • Number of points for selected quiz
• All students in a given level • Selected test • One exercise	• History and submissions of exercise • List of hyperlinked students with number of submissions for selected exercise
• One student in a given level • All tests and all exercises	• Last executed exercises for selected tests • List of all exercises with number of submissions
• One student in a given level • Selected test • All exercises	• Last executed exercises for selected tests • List of all exercises with number of submissions
• One student in a given level • Selected test • One exercise	• History of selected exercise • Last executed exercises for selected tests • List of all exercises with number of submissions

Table 3: All six combinations of user interface options for the "Tests" page

Figure10: The "Tests" page, showing the score for all students in the basic level.

Finding the right combination of online and traditional teaching tools to meet the wide variety of educational demands is one of the greatest challenges. Our implementation seeks to establish a balance between the students' requirements and site organization; we have tried to overcome the deficiencies present in previous web-based systems.

5.3 Lessons & quiz tools

Like CS Circles, IMI Python primarily retains the form of an exercise-centric web site, which means that exercises are embedded through the lessons. Exercise formats include short answer, multiple choice, coding exercises and "code scramble." Like CS Circles, these are stored in a database table; however, we had to add two new fields.

In our database, WordPress pages represent lessons and quizzes. The level indicator, saved in the "level_id" column, uses 1 to denote the basic level, 2 for medium, and 3 for advanced. We also record whether each page is a lesson or a quiz, using 0 to represent a lesson and 1 to represent a quiz, in the new column "is_test.".

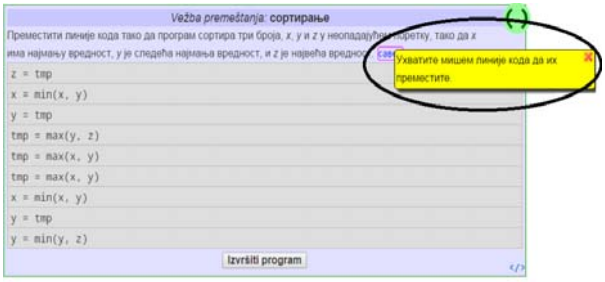
	major	minor	ordering	title	id	number	lang	level_id	is_test
Edit Copy Delete	1		1	Увод	1301	1	sr	1	0
Edit Copy Delete	2		2	Променљиве	1315	2	sr	1	0
Edit Copy Delete	3		3	Грешке	1337	3	sr	1	0
Edit Copy Delete	4		4	Функције	1328	4	sr	1	0
Edit Copy Delete	5		5	Вежбе	1378	5	sr	1	0
Edit Copy Delete	6		6	Коментари и наводници	1394	6	sr	1	0
Edit Copy Delete	7		7	Типови података	1416	7	sr	1	0

.

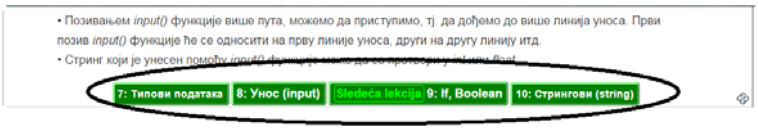
Note that the authors, editors and administrators of the system can create lessons and quizzes, but only administrators can publish them. The WordPress plug-in User Role Editor[9] has facilitated this approach.

IMI Python lessons include several features from CS Circles that we did not modify substantially: user feedback from the auto-grader, clickable hints, and "previous/next lesson" buttons on the bottom of each page. These are illustrated in Figure11.

a) Auto-grader feedback



b) Clickable hints



c) "Previous/next lesson" buttons

Figure11: User interface elements of lesson expositions.

[9] https://wordpress.org/plugins/user-role-editor/

CS Circles includes a feature where logged-in users can retrieve the definition of any exercise. This was not in the original site, but it was added when the site became open-source [20]. However, this was a counterproductive feature for our purposes, at best distracting for the students. This option is now visible only for the teachers. Additionally, instead of opening the Python code in a new web page, it is opened in a form of dropdown box, by clicking an icon in the bottom right corner of the exercise (see Figure12).

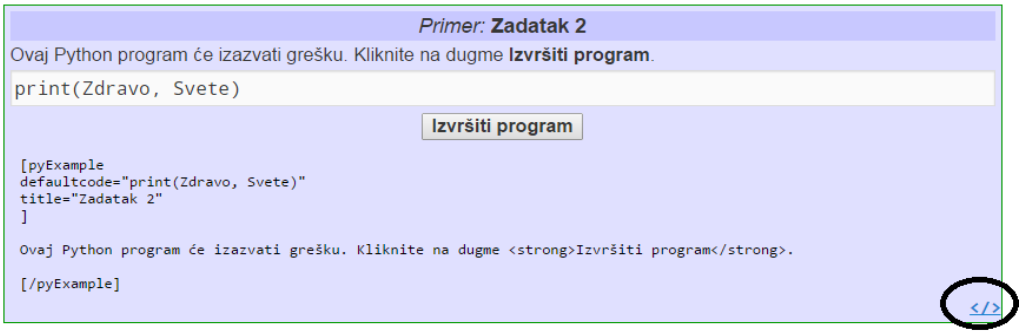


Figure12: Exercise source code accessor.

All mentioned IMI Python characteristics are intended to contribute to the quality of the course and meet student's requirements. In the next section the students' and authors' feedback about upgraded features will be analyzed.

6. Results

Satisfaction and motivation are useful ways to measure the success of web-based courses. Aiming to estimate pedagogical results of the IMI Python system, the relationship between students' outcomes and their utilization of the system was analysed. The main objective of our analysis was to evaluate the usefulness of IMI Python and its different components for meeting students' needs. The analysis is based on data collected from on-line surveys when the pilot test courses were completed. After all courses were completed, students received an email invitation to fill out the on-line survey. In the invitations, we explained that the purpose of the survey was to gather feedback about the courses, ensuring anonymity of the responses. We also pointed out that information was important in deciding how to improve the program and website. During the academic year 2014–2015, a total of 66 students, were enrolled in three Python courses (). The basic level was aimed at the 2nd-year students, of which there were 43; the middle level was aimed at the 3rd-year students, of which there were 13; and the advanced level was aimed at the 4th-year students, of which there were 10. However, many students also tried out the other levels, since all were available to them on the website. We elaborate in Section 6.2.

The survey was completed by 33 students (50%). In our opinion, the reason for the small sample is that the survey became available when the courses were already been finished.

6.1. Feedback from students about IMI Python

The information we collected during the on-line survey includes:

- Fourteen Likert-scale items for measuring student contentment with the IMI Python system;
- Five Questionnaire items about the student's overall satisfaction.

Likert-scale items. The Likert-scale items are generally organized in following groups: course organization (CO items 1, 2 and 3), exercises (EX items 4, 5 and 6), feedback (FB items 7, 8, 9 and 10), quizzes (QU items 11 and 12), communication (COM item 13) and technological tools (TT item 14).

In order to numerically analyse the survey results, we translated the Likert scale responses to numbers using the following five point scale: 1 = strongly disagree; 2 = disagree, 3 = neutral; 4 = agree; 5 = strongly agree. Table 4 shows the mean obtained by students for three courses and for three different academic year profiles. Means ranged from 2.76 to 4.67. In order to validate the reliability, we applied Cronbach's alpha [21] to the results, using the MedCalc[10] application to perform the calculation. The alpha value was 0.8609, which indicates a very high level of reliability. Most item's scores were above the average of 3, suggesting a positive perception of IMI Python.

Likert-scale item	Category	Mean
1. The courses are effectively organized.	CO	2.79
2. Lessons are clearly linked to the course objective.	CO	3.42
3. Navigation through levels is easy to follow and well implemented.	CO	3.58
4. Exercises are useful.	EX	2.88
5. Exercises are eye-catching.	EX	2.97
6. Exercises are easy to understand.	EX	3.27
7. Feedback is clearly defined.	FB	3
8. Feedback is timely.	FB	4.67
9. Feedback is significant in solving problems.	FB	2.91
10. Hints are helpful in solving problems.	FB	3.45
11. The quizzes are a good way to check my own knowledge.	QU	3.33
12. The quiz weights are appropriate.	QU	3.03
13. The communication with mentor is timely.	COM	2.76
14. Technological tools have an important role in the improvement of programming skills.	TT	3.09

Table 4: Students' perceptions of IMI Python system, measured by the survey means

The main results of the surveys based on the academic year profiles are represented in Figure13. It is interesting to note that the students who indicated the most satisfaction were from the MPM profile. We think that these students wanted to peek in more detail at the world of programming and escape from the framework of mathematics. MSCI students showed less satisfaction, possibly because they have more IT expertise than the previous group. The students from the MPI profile showed the lowest level of satisfaction, perhaps due to their professional orientation.

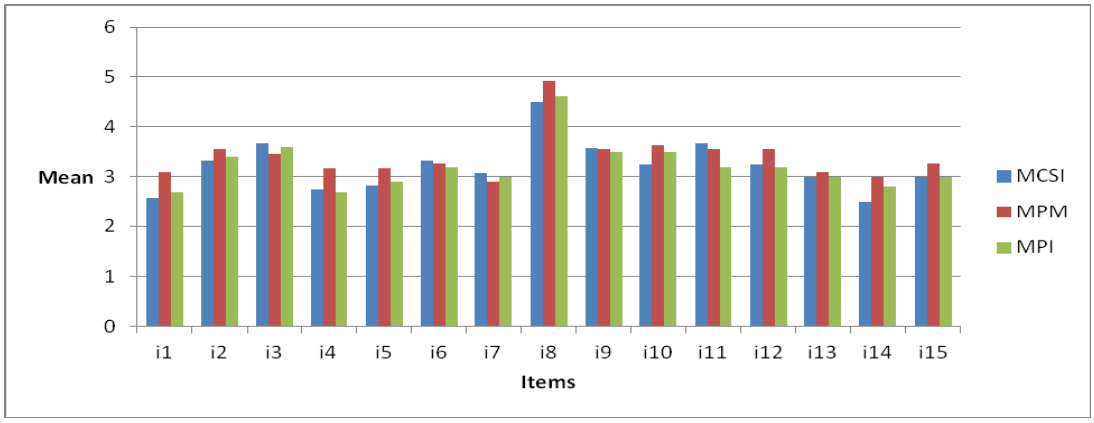


Figure13: The result of students' surveys for IMI Python system

[10] http://www.medcalc.org/

Questionnaire true/false items. We asked five questionnaire items were used to represent the general student's satisfaction. The results, shown in Table 5, give the students' responses. They indicate overall that the students have a very positive opinion about IMI Python.

Questionnaire item	YES	NO
I have enjoyed studying with the IMI python	92%	8%
I wish that the course has a more difficult exercises	34%	66%
I have enjoyed running online quizzes	81%	19%
I wish that the course offers the option of group competition	79%	21%
I will recommend the IMI Python to other students	89%	11%

Table 5: Overall student satisfaction with the course.

Discussion of Survey Results. To simplify the discussion below, we use only percentages below to discuss both the true/false and Likert questions. To make the discussion format uniform we translate the means to percentages, dividing by 5 and multiplying by 100%, e.g., instead of a mean of 4.5, we write 90%.

The most interesting items were related to the organization of the course. Students generally thought that the courses were clearly linked to the course objective (68.4%), indicating that the courses were tailored to the students' needs. Navigating through courses was one of the questions that had the most positive responses from students (71.6%). Students had mixed opinions (55.8%) on the organization of the course.

About exercises, students thought they were easy to understand (65.4%), but had more neutral opinions on whether they were eye-catching (59.4%) and useful (57.6%). In fact our discussions with students said that a significant number wanted to have exercises that were even harder! So while we did stimulate students well, there is a necessity to go further to make IMI Python more flexible and enjoyable.

According to the students' opinions, feedback was clearly defined (60%), timely delivered (93.4%) and significant in solving problems (69%). This indicates that the system was supportive in searching for errors when the submissions were failing. Also, most of the students (69.6%) thought that hints were very helpful and motivational for solving problems. This points that the IMI Python fulfilled students' expectations.

Another item for the evaluation is quizzes. The students positively rated the integration of quizzes into the system (66.6%). Through comments, we concluded that most students considered the lack of feedback and hints during solving quizzes as major problems. Although this issue has a pedagogical role, it should not be rejected, because it can have a negative effect on the students' motivation. The majority of students expressed a desire to add group competitions in the future. A healthy and competitive environment thus could significantly improve IMI Python.

The mentors' behaviour, through the communication system, was also analysed and rated. Mentors have an obligation to provide feedback to help students to fix their mistakes. But this feedback, according to students' opinion (55.2%), usually is not fast enough. The pleasure is surely much higher through automated feedback, but certainly we should not ignore the fact that real live feedback can have a much better effect on student motivation since it will give more accurate and actionable information.

Technological tools, such as consoles and visualization, have positive effects on the students. Most students (61.8%) believe that they have an important role in the improvement of the programming skills. In addition to the exercises, the tools allow smooth development of software skills.

Generally, the overall impression was positive and encouraging, and we found the IMI Python to be very useful. Successful user interaction is a major cornerstone in the development of successful web-based courses. So, we will continue to listen to the demands of students and try to tailor the system to their needs.

6.2 Qualitative Data

During the course, all the interactions with the IMI Python were registered into the database, in order to obtain information about the usage of the site. The next two figures show charts that describing usage of data for IMI Python. Figure 14shows the number of students that registered for each level. As mentioned at the start of Section 6, the different difficulty levels were intended to match the three different class years of students (2^{nd}=beginner, 3^{rd}=medium, 4^{th}=advanced), but students were free to use materials from any level they wanted. We observed:

- More than half of the 2^{nd}-year students tried out the middle and advanced levels.
- All 3^{rd}-year students tried out the advanced level at least once, but a majority of 3^{rd}-year students also wanted to remember of some material from the basic level.
- About half of the 4^{th}-year students utilized either of the easiest two levels.

The most level with the most registrations overall is the basic level (39%). Its popularity can be justified by the fact that most of the students who attended the course had little knowledge of Python and wanted to learn gradually, as well as the large size of the 2^{nd}-year cohort. The basic level was also interesting for those students who wanted to be reminded of forgotten materials, or upper-year students who chose the Python course as elective one (This made sense to us since a few 3rd-year students chose the Python course as an elective, and some did not already have the essential knowledge they needed to pass through the basic level.) The advanced level was the second-most popular in terms of total registrations (32%), while the medium level (29%) was third. The advanced level was quite interesting for the students who had prior experience with Python and for those who wanted to learn more.

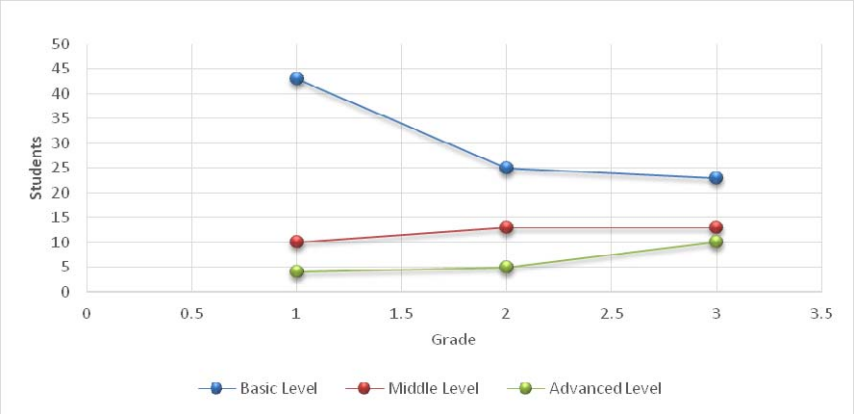


Figure 14: Students per level

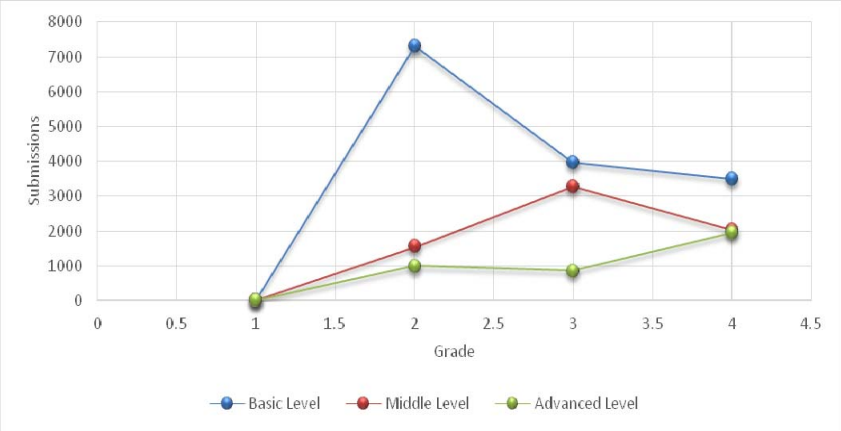


Figure 15: Submissions per level

The Figure 15 shows the relationship between students and the number of submissions. The most submissions were at basic level (39%), which is not surprising, given that it was attended by students who had the least experience in Python programming, and that the majority of students were 2nd-year ones. The advanced level has the smallest number of submissions (29%) which can be explained by the fact that the only the most experienced students tried it out.

We also noticed that the number of submissions, until the exercise solution becomes correct, depends on the exercise type. The programming exercises are leading in this sense (63% of submissions were for programming exercises). This is understandable, because code typing can make unintentional errors. They are followed by short answer exercises (21%), scramble exercises (9%) and multiple choice exercises (7%).

We observe , during the teaching activities, that the most amusing types of exercises were multiple choice and programming exercises. The first type of exercise is the most popular among students who expressed that they have little knowledge of Python, while the coding exercise was popular among students who have a solid knowledge of Python. Thus, using a variety of different types of exercises is important for our course materials to reach the broadest possible audience.

Considering the data related to the students/teachers communication, we discovered that the largest number of messages had been exchanged on the basic level (49%). This is evident because the students on basic level are beginners. Message exchange is considerably smaller at advanced level (33%) and (18%) at the middle level. This can be verified by the fact that students at advanced level address with harder exercises, while students at middle level are now more confident in their Python knowledge.

The information relevant to the quizzes, indicate that the main number of successfully resolved quizzes are at middle level (42%). This is probably the effect of the students' maturity in Python programming. Following are the basic (35%) and the advanced level (23%). Low percentage of success at an advanced level may be justified by the weight of tasks.

6.3..Implications for Teachers

The development of IMI Python was a real challenge for authors. Upgrading and adapting existing features from CS Circles was a pretty demanding job. However, all of its original features were very useful in creating the course content. The authors found the quizzes feature of IMI Python was the most interesting during the course design

phase. Also, creating lessons of different levels was interesting and often required inspiration in order to create a sufficient variety of material.

While the course ran, the teachers saw that students were quite interested. We also noted that the students' motivation grew over time. In fact, they eventually started making suggestions on how to improve the site; several ideas were born! For example, some proposed adding video tutorials to increase the effectiveness of the site. They also suggested adding additional topics. This was quite surprising and pointed out that they fully mastered the basics of Python and even more. As mentioned earlier, the most experienced students wanted even more difficult exercises. The comments and suggestions from the students were collected and, whenever possible, used to improve the course.

IMI Python could be suitable for a variety of training environments. Though designed for a programming course, there are a number of possible uses for other computer science courses, math courses, or electrical engineering courses. For example, our user interface could be used for students to develop incompletely-provided SQL queries, running the results in real-time; or allow a student to give an exact solution to an algebra problem that a Computer Algebra System could simplify and check for correctness.

An orthogonal avenue for development would be to enhance our system with new features. Some features we plan for future offerings of our introductory programming courses are daily and weekly quizzes, and team competitions. Regular practice and a competitive environment can be a healthy way to reach our main objective of enriching student programming and problem-solving skills.

The results of the survey allowed us to conclude that IMI Python is a useful framework. It can be employed as a web-based educational environment and integrated with web-based learning platforms to support students as well as teachers in a better way. The IMI Python system will contribute significantly to the learning options available for Serbian-language students; and that the design choices more generally lead to the system promoting inquiry, problem-solving skills, accessibility and competitiveness. We now believe that Serbian-speaking devotees of Python have reasons to be satisfied.

7. Conclusion and future work

The fast changes and enlarged complexity of students' needs introduced new challenges and is putting new requirements on the educational system. In confronting the challenges it is crucial to consider difficulties that must be addressed and create a system that will be able to, at least partially, meet the complex desires of users. IMI Python has the potential to satisfy these needs. It is a very useful system that facilitates the learning process and promotes creativity and competitiveness. By designing exercises at different levels of difficulty and providing flexible navigation, it makes progressive learning experiences available to the students, so they can solve exercises matching their ability level. In addition to meeting the students' high-level requirements, IMI Python provides a number of benefits for teachers. It offers the possibility to manage the execution of exercises in the form of lessons and quizzes.

IMI Python could be suitable for a variety of training environments. Although it makes conducive surroundings for learning programming, the space for the potential manoeuvre is still possible. We plan to enhance our system with daily and weekly quizzes and form of team competitions. A competitive environment can be healthy with the main objective to enrich student programming and other mental skills. In our future studies, we expect to collect more representative user information. We could measure and collect additional information for data analysis and intelligent reasoning, including data from non-university users that find our site freely available on the web. We are particularly interested in using Semantic Web technologies [22] to model and measure user interactions, so that we could create Semantic Rules [23] for intelligent reasoning, to make suggestions on which levels, lessons, and exercises users should do next.

Acknowledgments

This paper was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, III41010 - Preclinical testing of bioactive substances.

We are grateful to the student Milorad Obrenovic who participated in process of creating lessons and quizzes.

We thank the SIGCSE Special Projects Committee, whose grant allowed CS Circles to be open-sourced [20].

References

[1] Oblinger, D., Oblinger, J. L., & Lippincott, J. K. (2005). Educating the net generation. Online e-book. Educause .

[2] Capra, T. (2011). Online education: Promise and problems. Journal of Online Learning and Teaching, 7(2), 299-293.

[3] Pritchard, D., & Vasiga, T. (2013, March). CS circles: an in-browser Python course for beginners. In Proceeding of the 44th ACM Technical Symposium on Computer Science Education (pp. 591-596). ACM.

[4] Mory, E. H. (2004). Feedback research revisited. Handbook of research on educational communications and technology, 2, 745-783.

[5] Tremblay, G., Guérin, F., Pons, A., & Salah, A. (2008). Oto, a generic and extensible tool for marking programming assignments. Software: Practice and Experience, 38(3), 307-333.

[6] Higgins, C., Hegazy, T., Symeonidis, P., & Tsintsifas, A. (2003). The coursemarker cba system: Improvements over ceilidh. Education and Information Technologies, 8(3), 287-304.

[7] Wang, F. L., & Wong, T. L. (2008). Designing programming exercises with computer assisted instruction. In Hybrid Learning and Education (pp. 283-293). Springer Berlin Heidelberg.

[8] Heng, P., Joy, M. S., Boyatt, R. C., & Griffiths, N. (2005). Evaluation of the BOSS online submission and assessment system. Technical Report. Coventry: University of Warwick.

[9] Darling-Hammond, L., Amrein-Beardsley, A., Haertel, E., & Rothstein, J. (2012). Evaluating teacher evaluation. Phi Delta Kappan, 8-15.

[10] Stahl, G., Koschmann, T., & Suthers, D. (2006). Computer-supported collaborative learning: An historical perspective. In R. K. Sawyer (Ed.), Cambridge handbook of the learning sciences (pp. 409–426). Cambridge: Cambridge University Press.

[11] Su, B., Bonk, C. J., Magjuka, R. J., Liu, X., & Lee, S. H. (2005). The importance of interaction in web-based education: A program-level case study of online MBA courses. Journal of Interactive Online Learning, 4(1), 1-19.

[12] Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P., & Queirós, R. (2012). A distributed system for learning programming on-line. Computers & Education, 58(1), 1-10.

[13] Leal, J. P., & Silva, F. (2003). Mooshak: a Web-based multi-site programming contest system. Software: Practice and Experience, 33(6), 567-581.

[14] Cedazo, R., Garcia Cena, C. E., & Al-Hadithi, B. M. (2015). A friendly online C compiler to improve programming skills based on student self-assessment. Computer Applications in Engineering Education, 23(6), 887-896.

[15] Manne, F. (2000). Competing in computing. SIGCSE BULLETIN, 32(3), 190-190.

[16] Christensen, G., Steinmetz, A., Alcorn, B., Bennett, A., Woods, D., & Emanuel, E. J. (2013). The MOOC phenomenon: who takes massive open online courses and why?. Available at SSRN 2350964.

[17] Kybartaite, A., Nousiainen, J., & Malmivuo, J. (2013). Technologies and methods in virtual campus for improving learning process. Computer Applications in Engineering Education, 21(1), 185-192.

[18] Rößling, G., McNally, M., Crescenzi, P., Radenski, A., Ihantola, P., & Sánchez-Torrubia, M. G. (2010, June). Adapting moodle to better support CS education. In Proceedings of the 2010 ITiCSE working group reports (pp. 15-27). ACM.

[19] Guo, P. J. (2013, March). Online python tutor: embeddable web-based program visualization for cs education. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 579-584). ACM.

[20] Pritchard, D., Graham, S. & Vasiga, T. (2015) The State of CS Circles: Open Source & Outreach using an Introductory Python Website (Poster). 46th ACM Technical Symposium on Computer Science Education (pg. 688) ACM.

[21] Ritter, N. L. (2010). Understanding a Widely Misunderstood Statistic: Cronbach's “Alpha”. Paper presented at the Southwest Educational Research Association, New Orleans, LA. Retrieved from www.eric.ed.gov/ED526237[22] Mika, P., & Greaves, M. (2012). Editorial: Semantic Web & Web 2.0. Web Semantics: Science, Services and Agents on the World Wide Web, 6(1).

[23] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 21, 1-31.