Teaching Programming Basics for First Year non-IT Students

Olga Mironova, Irina Amitan, Jelena Vendelin, Jüri Vilipõld, Merike Saar
Faculty of Information Technology, Department of Informatics, Chair of Software Engineering,
Tallinn University of Technology
Tallinn, Estonia
{olga.mironova, irina.amitan, jelena.vendelin, juri.vilipold, merike.saar}@ttu.ee

***Abstract*—The purpose of this study is to demonstrate a teaching methodology in a general programming course for the first-year non-IT students at the Department of Informatics of Tallinn University of Technology, Estonia. The authors suggest some solutions for achieving better results in programming issues, which are usually sophisticated for the beginners in this field.**

Keywords—object-oriented programming, Pyhton, Scratch, VBA

I. INTRODUCTION

In recent years information technology is playing an increasingly greater role in human life, both at home and at work or study. Accordingly, "Computational thinking" is a skill that a learner must grasp to feel confident and be ready for the future workplace and able to participate effectively in the digital world. Thereby the objective of computer education for the non-IT students today is to educate them to be knowledgeable in technical fundamentals and be able to find common ground with IT-specialists.

In the present paper the authors try to find answers to questions about how to achieve better results in programming basics teaching for the first-year non-IT students.

II. THE BRIEF REVIEW OR THE CURRENT SITUATION IN THE COMPUTING TEACHING

Recently several countries have carried out thorough investigations of the use of information technology and courses on computer science in different schools. Analyses have shown that most of the courses do not meet the needs. As a result, several new curricula have been proposed to improve the situation [1] – [5].

In the present teaching approach to the programming course the authors use the principles introduced there and try to implement them in the best possible ways.

The current situation of teaching computer sciences at Estonian schools is contradictory. Surveys show that some Estonian secondary schools do not have informatics lessons at all. In the majority of the schools it is taught only for two or three years, which is a very short period to prepare learners for the university level.

This drawback is associated with two main reasons. The first one being that there is no nationwide Informatics curriculum in Estonia. The second one is that Informatics subjects are not mandatory in our schools.

A logical consequence of these reasons is the situation where each teacher introduces learners to the material at his own discretion: some learners draw in a graphics editor, others learn the computer hardware, etc. In connection with this, the level of PC skills among non-IT learners falls every year and reduces to commonplace social networks usage. In the Informatics course we have to take these facts into account and build the curricula accordingly.

However, it should be mentioned that quite many school pupils attend additional courses and learn IT and especially programming there. Unfortunately, they are not our students in future as usually they choose IT-specialities at universities.

Thus, the present course is aimed at students, which are not interested in using PC in their study, excluding finding the required information in any search system.

III. THE INFORMATICS COURSE DESCRIPTION

Basic computer education topics have been included into all the curricula at Tallinn University of Technology, and for non-IT specialities have been consolidated into a course named “Informatics”. This course lasts for two semesters and students have two academic hours weekly. Usually the group size is 20-30 students, which annually depends on the total number of students at the university. During the course lecturers apply classic face-to-face classroom methods, pair or group works and independent learning in the Moodle e-environment. The last one gives a lot of opportunities to make the course more attractive and dynamic to raise students’ interest and motivation.

The main learning outcomes of the Informatics course are listed below. SA student who completes the course:

- Acquires the foundations of problem analysis and system modelling.
- Can analyse relations between objects and provide rationale for the algorithms and methods applied.
- Is familiar with the nature of data and objects and can specify them and use them in programs.
- Is familiar with and can describe, using VBA/Python and UML activity diagrams, the main activities occurring in programs and algorithms.

- Is familiar with the nature and main concepts of object-oriented programming.
- Can compose programs consisting of multiple procedures and organize data flow between them.

The course aims at reaching the results in two different but tightly linked ways: learning to understand the object-oriented approach and getting necessary skills in building algorithms. Both skills have to be implemented in simple applications. After years of the experiments two programming languages were chosen for the Informatics course. These are Python and Visual Basic for Applications (VBA).

It should be mentioned that the Informatics course seems to be rather complicated for most of the non-IT students. The main problems in teaching programming at present have been clearly identified and systemized in [6], [7]. However, the course authors and teachers still face some problems. The biggest of them is lack of preparation and lack of prior knowledge for the Informatics course among the first-year non-IT students. As a result, they have lack of motivation in the learning process, which, in its turn, leads to poor knowledge and poor academic results. Consequently, instructors try to improve the course program and content from year to year with the aim of finding the best solutions to achieve the goals and get the outcomes.

As practice shows, the main programming concepts that are complicated for the non-IT learners' understanding are:

- Conditional statements and iterations
- Parallel and sequential processing
- Data
- Subroutines

At the same time, they are the most important concepts in programming and there is no opportunity to learn and teach without them.

The authors suggest ways to help non-IT learners to grasp the programming basics - the implementation of the visual programming to the Informatics course before any serious coding.

IV. VISUAL PROGRAMMING

As practise shows, for better comprehension, it is good to graphically demonstrate and provide an opportunity to try out things that are hard to understand.

A new and rapidly upcoming way in teaching programming basics is visual programming using an environment which has been created especially for learning. The most popular are Scratch [8], Snap! [9], Blockly [10] graphical tools, which make the programming process much easier for the beginners, especially for non-IT, who have not any experience in building algorithms, programming and coding.

In the presented Informatics course the graphical programming environment Scratch is used as a supporting tool before VBA or Python. After a few years of practice, the authors came to the conclusion that it is an effective introductory tool to understand both the object-oriented approach and the functionality of a program. This conclusion is reinforced by some facts:

- Syntax errors are impossible in Scratch.
- Scratch works as an interpreter.
- Graphical command blocks give an excellent visual picture of the different controls, used in the program.
- Scratch is simple and expressive, it makes understanding the behaviour of created objects easier.

With regard to object-oriented programming, creation of objects in Scratch is provided by drawing or importing graphics. Scratch objects are named sprite and each one has its own properties and methods. Combining the blocks for each object creates the methods. Some of the blocks are used to show the reaction of the object to some events. Thus, we see here the main aspects of object-oriented programming resulting in an attractive animation.

Let us see how Scratch solves the problem areas in programming basics, which were mentioned above:

A. *Conditional statements and iterations*

Using Scratch blocks makes it easy to show students how iterations work – after its implementation learners immediately see the result: multiple iterations of chosen blocks (Fig. 1).

Fig. 1. The iterations in Scartch

Using branching blocks in Scratch helps students to compose if-sentences. For example, it becomes clear why they have not written the second condition in them (Fig. 2).

Fig. 2. The branching in Scartch

This construction of the iteration and branching blocks gives a clear picture about the actions inside these blocks.

B. *Parallel and sequential processing*

Running each script is considered a separate process. Scripts that handle the same event will be running in parallel after the event occurs. For example, the left scripts in Fig. 3 are performing at the same time and they implement the reaction on the "click" event. The right script is the same actions but they are executed sequentially. It should be noted that it is easy to follow and explain the difference due to animation: the right side figure – the object first jumps and then moves; the left side – the object is jumping and moving at the same time.

Fig. 3. Realization of the parallel and sequental processes

C. Data

Variable is one of the basic concepts in all programming languages. Its meaning in programming differs from its use in mathematics, which students know from secondary school. As to lists, and especially indexation of the list elements, these are absolutely new concepts for the first-year non-IT students and usually cause learning difficulties and mistakes in usage.

The graphical environment Scratch provides clarity in understanding the meaning of a variable and list concepts. All variables and lists have to be created manually before using them in a script. Using the command "Make a Variable" or "Make a List" in the Data group of the blocks students try out, see and this way understand it better as a named place in the computer memory (Fig. 4).

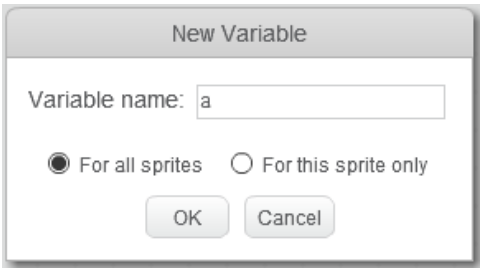

Fig. 4. A variable declaration in Scratch

A variable and list images on Scratch stage (Fig. 5) give students an overview of their values and some properties (length), which can be changed manually and/or in a program. Also, due to Scratch animation, learners can follow how program processes list elements – during the process indexes of the elements are blinking. Moreover, there are two ways to fill a list with elements or clear a list: manually and in the program. This greatly helps students in using lists in other programing languages.

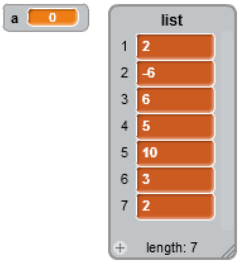

Fig. 5. A variable and list images on Scratch stage

Moreover, the learners have to define the scope of the created variable or list, which leads them to better understanding the meanings of global and local variables and demonstrates the difference between these two. If an object (sprite) is active, only global and local variables and lists can be used - it can be seen immediately. Thereby students obtain the concept of the data scope faster and better.

D. Subroutines

The ability to split a big task into smaller pieces plays an important role in structured programming. This is the most preferred approach in building programs.

The majority of algorithmic languages support the definition of subroutines and functions, used in creating the code for the pieces of the project. One of the main methods of transferring data to subroutines is using the parameters. The authors' experience shows that this is the most confusing topic for a beginner.

Scratch 2.0, the new version of Scratch, provides teachers and learners with a perfect opportunity to make this issue easier. Learners can create and use their own Scratch blocks, where the definition of parameters is included (Fig. 6).

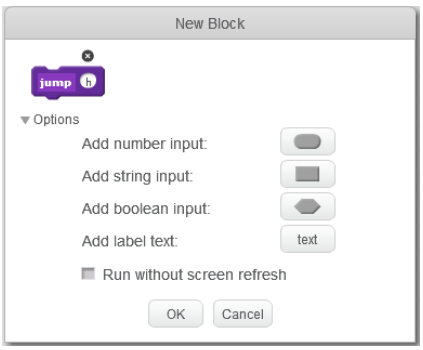

Fig. 6. The user's block creation

Students learn to create a clear structure in their project. They divide their task into logical parts and create necessary user blocks, providing them with parameters. Now the main script can use standard and user-defined blocks, transferring the necessary data by means of parameters. Fig. 7 shows the definition and calling of the user-defined blocks.

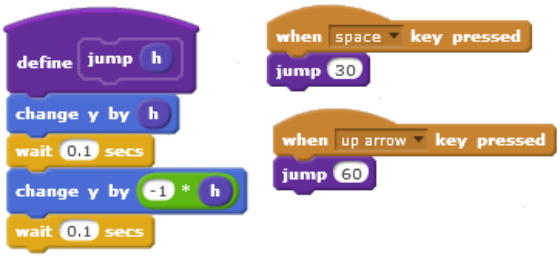

Fig. 7. The user-defined block and its use in the main script

The next Informatics course step is to proceed with more complicated tasks in other programming systems. Some lessons of practicing with Scratch tools make this transition easier.

It should be mentioned that according the annual students' feedback Scratch is the most popular module in the course. Students emphasize that it is Scratch that gives them an overview of the issue model and an algorithm for its solution.

The authors, one more time, would like to draw the reader's attention to the fact that Scratch is a tool, which was not created to solve complicated tasks but it is an excellent instrument for introducing programming, especially for non-IT learners.

V. Python and Visual Basic for Applications Based on Scratch

As was mentioned earlier, Scratch and Visual Basic for Applications (VBA) have a lot in common. For example, a number of graphic objects can be placed into MS Excel applications. There are a number of VBA methods, which have equivalents in the form of Scratch blocks (e.g. set colour). Therefore, the coding part of the course starts with graphical objects animation in Excel. After that students solve tasks related to their speciality.

During the Informatics course, the authors provide beginners with some ready-made procedures, to be used as "black-boxes". For example, procedures are used that correspond to Scratch blocks such as "wait" (Fig. 8), "move", "glide", "touching" etc.

```
Sub pause(pp)
  Dim pl
  pl = Timer() + pp
  Do
    DoEvents
  Loop While Timer() < pl
End Sub
```

Fig. 8. VBA procedure and analogous Scratch block

This enables a faster transition to more sophisticated tasks in VBA. If students build and understand an algorithm (verbally, on paper and/or using Scratch), they can compose the textual solution using the existing procedures and their own commands.

The Python programming language supports structured programming and procedural styles. Python does not require any declaration of simple variables, which makes work with it easier for the beginners. It has a large standard library. It is an open source and is available to all students. The language is a high-level language and has syntax, which allows programmers to express concepts in fewer lines of code than would be possible in some other languages.

As practice shows, during programming in Python beginners usually face problems with indentations in their code. These indentations play a great role in Python and their misapplication can ruin the entire program. And again, Scratch helps to solve this problem: command blocks, which are situated inside iteration blocks and if-blocks, have the same indentation level as commands in Python (Fig. 9).

```
x=int(input("Give me a number => "))
if x>0:
    print("Positive")
else:
    if x<0:
        print("Negative")
    else:
        print("Zero")
```

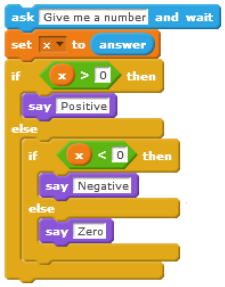

Fig. 9. The indentations in Python and Scratch

In addition, it should be noted that writing a program in VBA using indentations is very useful and effective for beginners to better understand the program structure and better represent the final result (Fig. 10).

```
x = InputBox("Give me a number")
If x > 0 Then
   MsgBox "Positive"
Else
    If x < 0 Then
      MsgBox "Negative"
    Else
      MsgBox "Zero"
    End If
End If
```

Fig. 10. The indentations in VBA

Thus, it can be seen that the visual programming environment Scratch really helps beginners in their starting in a programming field. Firstly, it motivates students due to its graphics and animation and, at the same time, helps them to follow the created objects´ behaviour. Secondly, it has not any syntax errors. Finally, Scratch is a remarkable tool for introducing a problem solution algorithm.

VI. The Teaching Metodology

For complicated tasks in text programming (Python, VBA), included in the training program of the Informatics course, teachers and students together try to build UML (Unified Modelling Language) activity diagrams to describe algorithms. It is the first step in any task solution. In addition, a verbal description and pseudo code are typically used.

Now, when students create Scratch projects as an introduction to programming, teachers can also use its scripts to visualize, formulate and describe the problem.

At the beginning of the programming module of the course (after Scratch), the teacher provides the students with a prepared model (UML, pseudo code or Scratch script), which is analysed in a group. The analysis is followed by writing the program code according to the given algorithm. Later on, students have to create the models themselves.

In addition, the course instructors always try to give students tasks with a similar content to solve. If a student has created a model in UML and after that the same application in Scratch was done, it is easier for him to "translate" it into the VBA or Python.

Thus, it should be emphasized once again that if a learner understands the content of the model and the algorithm, it is easier for him to understand the syntax of any language and, finally, to grasp it faster.

It should to be mentioned that according to the test [11] most of present students are visual learners [12], [13], who want to see "how it works". Based on this fact, the Informatics teachers try to maximally visualize all the course content. In addition to learning materials in Khan Academy style [14], instructors teach students to use visualizing tools such as the built-in Locals Window in VBA and the Online Python Tutor [16].

These instruments help students to follow the program execution similarly to Scratch, but on a higher difficulty level. Although Online Python Tutor has some drawbacks (it does not support work with graphics, time and files), it gives an opportunity to check each step of the program code with its results – values. More similar opportunities are provided by the Locals Window in VBA.

In addition, during the course teachers give students tasks where they should find and correct some errors in an already ready-made program. These tasks raise students' interest and motivate them, especially if it is presented in a competition form and give an opportunity to get some bonus points for future exam.

As a group-work assignment during contact lessons it is also possible to offer students to play a game: they have the algorithm of a program and each student in class should write one line in the code. This method is quite controversial, but it is useful at the beginning of coding, when students just learn the basics. Afterwards, when each student has his/her own programming style, it is not so useful. However, it teaches to understand others' manners and proves and demonstrates why one solution can be better and more logical than another. It should be noted that this is important knowledge in any subject, not only in programming.

The course authors have also worked out a test system, which uses tasks similar to the pre-prepared program tasks. However, in tests students have to fill in gaps in the program blocks. Doing this, students learn the syntax and learn to understand the algorithm.

CONCLUSIONS

The issues reviewed in this paper are very useful in the process of understanding modern concepts in building applications and, hopefully, help students in their future study and professional work.

Based on the above said, it should be concluded that the authors of the Informatics course for the first-year non-IT students focus mostly on the model, algorithm and their visualization, rather than teaching syntax and coding techniques.

Visualized tools like Scratch are a good way to introduce and, afterwards, better and faster understand the main concepts of object-oriented approach. In addition, Scratch makes it easier to write a programming code in any language when these concepts are clear. After "the first level understanding", any programming process for non-IT students and, especially for the beginners, should be visualized in any case using suitable instruments.

In the future development of the described Informatics course, the authors try to keep up with times and main trends in computer education as [16], [17].

REFERENCES

[1] CSTA K–12 Computer Science Standards. 2011. Available at: http://csta.acm.org/Curriculum/sub/K12Standards.html

[2] AP Computer Science Principles, 2011-2016. Available at: https://advancesinap.collegeboard.org/stem/computer-science-principles

[3] CSTA Computational Thinking Task Force. Available at: http://csta.acm.org/Curriculum/sub/CompThinking.html

[4] UK. The Royal Society. „ Shut down or restart? " The way forward for computing in UK schools. Retrieved from: https://royalsociety.org/~/media/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf

[5] UK. Computing in the national curriculum: a guide for secondary teachers. Retrieved from: http://www.computingatschool.org.uk/data/uploads/cas_secondary.pdf

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

[7] A. Kak, 2014. Teaching Programming. Available at: https://engineering.purdue.edu/kak/TeachingProgramming.pdf

[8] MIT Media Lab, 2013. Scratch. Available at: http://scratch.mit.edu/

[9] Snap! Available at: https://snap.berkeley.edu/

[10] Blocky. Google Developers. Available at: https://developers.google.com/blockly/

[11] B. A. Soloman, and R. M. Felder, (n. d.). Index of Learning Styles Questionnaire. Retrieved from: http://www.engr.ncsu.edu/learningstyles/ilsweb.html

[12] O. Mironova, T. Rüütmann, I. Amitan, J. Vilipõld, M. Saar, "Computer Science E-Courses for Students with Different Learning Styles", in: Annals of Computer Science and Information Systems: Federated Conference on Computer Science and Information System, Kraków, 2013, pp. 735 - 738.

[13] O. Mironova, I. Amitan, J. Vendelin, M. Saar, T. Rüütmann, "Strategies for the Individualization of an Informatics Course", in: Annals of Computer Science and Information Systems: Federated Conference on Computer Science and Information Systems, Warsaw, 2014, pp. 835 - 840.

[14] Khan Academy. Available at: https://www.khanacademy.org

[15] Ph. Guo. Online Python Tutor. Available at: http://www.pythontutor.com/

[16] Exploring Computer Science. Retrieved from: http://www.exploringcs.org/

[17] National Science Foundation. Retrieved from: http://www.nsf.gov/