Inquiry-Based Python Programming at Secondary Schools

J. Guniš*, L. Šnaijder*, Z. Tkáčová* and V. Gunišová**
* Pavol Jozef Šafárik University in Košice, Košice, Slovakia
** Secondary grammar school of Ján Adam Rayman, Prešov, Slovakia
jan.gunis@upjs.sk

Abstract – Programming is a traditional part of informatics education from Primary to Upper Secondary Education in Slovakia. We have seen a growing interest of using Python in programming at our secondary schools during the past six years. However, the transition from using the still prevalent Pascal to Python is often only formal, regardless of the language specifics of Python. In addition, teaching programming is oriented primarily to mastering language constructions. In the frame of the National project IT Academy we have focused on inquiry-based programming teaching. In our teaching and learning materials we exploit the 5E learning cycle. Firstly, pupils are engaged in a topic, then they explore a topic and give explanation for their findings, then elaborate on their learning and finally are evaluated. Their learning is not focused just on acquiring skills and experience using suitable Python commands and data structures, but also on using various problem solving strategies. The paper presents examples of teaching materials and interesting findings and comments from pilot teachers.

Keywords – Inquiry-based education; programming; Python; computational thinking; secondary school

I. INTRODUCTION

There are various programming paradigms, programming languages, various orders of teaching topics, as well as different teaching approaches used in teaching secondary school programming.

In Slovakia, teaching of programming has a long tradition. Algorithmic problem solving is one of five main areas of teaching informatics that is described in the National Education Programme (NEP) [1].

Under the topic Algorithmic problem solving, pupils from the 3rd grade of primary school to the last grade of secondary school solve algorithmic problems in various programming environments, starting with interactive iconic environments [2], through block environments (e.g. Scratch) to text programming languages (e.g. Python, Pascal, Logo, C, Java). Programming environments and languages are merely a means, not a learning objective; it is to develop the ability to solve problems using suitable tools of a programming language. The emphasis is gradually shifting from acquiring concrete experience and skills at primary school to mastering the basics of informatics at secondary schools, where this goal is already dominant.

Innovation of informatics is considered not only as content innovation (e.g. selection of suitable programming language or environment [3]), but also as innovation of learning methods. These methods according to [4] should make it possible to develop pupils’ competences – creativity and innovation; communication and collaboration; research and information fluency; critical thinking, problem solving, and decision making; digital citizenship; technology operations and concepts.

These competences can be achieved by Inquiry-based learning (IBL) [5], which has no tradition in informatics teaching. IBL is naturally implemented in science education (e.g. Physics, Biology, Chemistry). In the implementation of IBL in programming teaching we have based on experience in implementing IBL in physics, chemistry and biology [6], research results in teaching programming [7, 8, 9] and our own experience with IBL in computer science [10, 11].

The results of researches aimed at teaching novice-programming were the most inspiring for our research. Kolouri et al. report in [12] as the main results of their research - the positive use of the syntactically simple Python language, teaching problem solving before teaching programming, the use of formative assessment tools. In their meta-analysis [13], Piteria and Costa mention programming topics from the highest to the lowest level of comprehension.

In the framework of the National IT Academy project, the authors of this article (two of them are didactics experts in informatics and two others are secondary school teachers) proposed the conception of teaching the subject and developed teaching and learning materials for the basic programming course with the IBL implementation. These results, achieved in cooperation with pilot teachers from participating secondary schools, are presented and discussed in the following parts of the article.

II. IMPLEMENTATION OF IBL IN TEACHING OF PROGRAMMING

A. Learning objectives

In specifying the content of education for the proposed teaching materials, we followed the NEP for secondary grammar schools with a four-year and five-year educational program for informatics subject [14]. Provided teaching materials for teachers cover the entire
content of the algorithmic problem solving area defined in the NEP. In each teaching material we specify more detailed objectives defined in the NEP, e.g., in the teaching material where pupils learn about the Python console, we stated:

Learning objectives defined in the NEP:
- To use mathematical expressions to express relationships.
- To solve problems where values need to be stored in memory, and then to use those stored values in expressions.

In our more detailed specification of learning objectives in the teaching material we add extra:
- To create and evaluate arithmetic expressions.
- To create and use a variable.
- To use appropriate, descriptive variable names.

In the informatics subject, especially in the area of algorithmic problem solving, we see great potential for the pupil development beyond the cognitive objectives defined in the NEP.

Computational thinking is a key competence for the 21st century [15]. It is the ability to solve problems, think and propose solutions of problems so that we can use computers efficiently to implement them. We tried to specify this thinking more precisely and to develop it through teaching in accordance with our teaching materials. For the purposes of our research, we used the framework defined in Barefoot [16]. Within this framework, 6 concepts of computational are distinguished:

- Logic – predicting and analyzing,
- Algorithms – making steps and rules,
- Decomposition – breaking down into parts,
- Patterns – recognizing and using similarities,
- Abstraction – removing unnecessary details,
- Evaluation – making judgements.

The concept of computational thinking is new to informatics teachers in Slovakia, they need to learn more about it and have also a set of practical examples how to develop it. Therefore, in our teaching materials we specifically describe the tasks (parts of their solutions) in which we intend to develop particular concepts of computational thinking, e.g.:

- Decomposition – linear decomposition - linearly divide problems into smaller parts so that they can be used to achieve the goal (splitting the calculation into smaller steps).
- Abstraction – using essential elements of problems (solving verbal problems).

Pupils develop these computational thinking concepts in a particular lesson by solving a problem where they should suggest to the fast food retailer a way to calculate as quickly as possible the price of the goods sold, e.g.:

```python
>>>hotdog = 0.55
>>>hamburger = 1.2
>>>fries = 0.7
>>>2 * hotdog + 5 * hamburger + 3 * fries
```

B. Content of innovative teaching materials

Python itself is not our goal. The dominant goal that we want to develop is pupils’ ability to solve problems algorithmically. In our teaching materials we purposefully develop computational thinking of pupils. We achieve this by appropriate formulating problems (see Fig. 1.) and also by order of topics. E.g. we included topic of functions (defining and calling) among beginning lessons, because we want to develop ability of problem decomposition as an important concept of computational thinking. This ability allows pupils to solve more complex problems by programming smaller functions solving particular sub-problems.

![Figure 1. Task: Create the stamp by which can be filled the entire shape of the pawn in the image](image)

We assume that approximately one third of the total hourly subsidy (3 x 33 hours) is devoted to algorithmic problem solving at secondary grammar school. We have designed a total of 27 teaching materials covering 32 lessons (Table I)

<table>
<thead>
<tr>
<th>TABLE I. DESIGN OF LESSONS EXPLOITING INNOVATIVE TEACHING MATERIALS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Visualization of the calculation using turtle graphics (turtle module)</strong></td>
</tr>
<tr>
<td>01. Introduction to programming, variables, calculations in console, errors</td>
</tr>
<tr>
<td>02. Sequence programming, turtle graphics, drawing simple geometric shapes</td>
</tr>
<tr>
<td>03. Definition of functions without parameters and without return value</td>
</tr>
<tr>
<td>04. FOR statement</td>
</tr>
<tr>
<td>05. Recapitulation of the thematic area and pupils testing</td>
</tr>
<tr>
<td>06. Definition of functions with parameters – drawing tasks</td>
</tr>
<tr>
<td><strong>Console applications, problem solving</strong></td>
</tr>
<tr>
<td>07. Definition of functions with parameters and with return value – numerical tasks</td>
</tr>
<tr>
<td>08. Errors in programs, debugging</td>
</tr>
<tr>
<td>09. IF statement</td>
</tr>
<tr>
<td>10. Recapitulation of the thematic area and pupils testing</td>
</tr>
<tr>
<td>11. Strings and manipulation with part of strings (indices, slices)</td>
</tr>
<tr>
<td>12. Strings and their methods, nested IF statements</td>
</tr>
<tr>
<td>13. Algorithms on strings</td>
</tr>
<tr>
<td>14. Exception handling</td>
</tr>
<tr>
<td>15. Determine problem parts of program, raising exceptions</td>
</tr>
<tr>
<td>16. Recapitulation of the thematic area and pupils testing</td>
</tr>
<tr>
<td>17. Lists and their methods</td>
</tr>
<tr>
<td>18. Lists – creating and modifications, random values</td>
</tr>
<tr>
<td>19. Algorithms on lists</td>
</tr>
<tr>
<td>20. WHILE loop statement</td>
</tr>
<tr>
<td>21. Nested statements (loops and ifs)</td>
</tr>
<tr>
<td>22. Recapitulation of the thematic area and pupils testing</td>
</tr>
</tbody>
</table>
C. Approaches and practices of teaching

Informatics education is based not only on facts and knowledge, but also on methods and ways of acquiring this knowledge. We consider it important for pupils to develop their inquiry skills (e.g. experimenting, predicting, verifying their predictions, critical reasoning, making conclusions, etc.) as well as a positive relationship to the scientific way of getting to know the world. We apply the 5E instructional model [17] in most of our teaching materials. The learning cycle consists of five phases (examples from the 4th lesson in which pupils encounter the loop concept):

**Engage** – motivation of pupils, finding out the pupils' initial knowledge about the studied topic. This phase can take the form of a discussion, a video presentation or a short story, solving an interesting problem, etc. The aim of this phase is to attract pupils to the topic of study and to increase their interest in the topic. For efficient use of cycles, it is important to recognize repeating parts/patterns of the problem, e.g. when drawing pictures (see Fig. 2.).

![Figure 2. Task for recognizing repeating patterns in the images. (engage phase)](image)

**Explore** – activities in which pupils come up with various ideas and hypotheses about how the studied system works. It is an independent work of pupils in which the teacher does not interfere or intervenes only minimally, e.g. in the case of guidance or clarification of the task. In this phase we often use Python console, which enables quick exploring and getting feedback from running parts of code without the need of writing complete code of program. Reading, comparing and comprehension of programming codes is the essential part of learning both programming language and programming as a process (see Fig. 3.)

![Figure 3. First touch with a new command (FOR statement) by comparing two program codes and further exploring the new command (changing the parameter in function range()). (explore phase)](image)

**Elaborate** – at this phase, pupils practice and extend new knowledge (e.g. FOR statement). They solve various types of tasks (e.g. analytical, evaluation, synthetic), suggest possible improvements of solutions or other solutions of tasks. Typical task is to create a complete program for solving problem, e.g. to draw image with repeating pattern. (see Fig. 4.).

![Figure 4. Solving tasks of varying complexity to practice FOR statement. (elaborate phase)](image)

**Evaluate** – the aim of this phase is to obtain information on the level of pupils' acquired knowledge and misconceptions. This phase is often carried out by various forms of formative assessment, e.g. self-assessment test, self-assessment card, checklist. Typical misconception of control flow statement is that statement evaluates knowledge. These two lessons also include a didactic test, its authorial solution and phenomena analysis. We have included project-oriented teaching materials in the final part of our Python course. The final lessons are focused on creating a product creation that is the result of a unique pupils' solution. Pupils take more responsibility for the

![Figure 5. Self-assessment test to obtain potential pupils’ misconceptions about FOR statement. (evaluate phase)](image)
solution and in the case of group projects they choose which particular part of the project they will solve. An important part of the project solution is also the presentation of results and their evaluation.

D. Support for teachers and pupils

Teachers play an important role in implementing innovative teaching of programming, so we provide them with a number of supporting teaching materials. In the teaching materials we draw the teacher’ attention to the key parts and also to the supposed problematic parts of the lessons. The final versions of the teaching materials will include teachers' experience in testing of the teaching materials and, where appropriate, alternative teaching approaches.

In addition to the teaching materials, we prepared two conceptual documents for teachers. One is more methodologically oriented and describes the whole concept of innovative teaching materials and teaching methods used (e.g. 5E learning cycle, formative assessment and project method). It also includes the concepts of computational thinking, which are described in more details. The second conceptual material is focused on the concept of teaching programming in Python where the specific content of the education and the reasons for the proposed lesson order are described.

We prepared worksheets, task collections and work files for pupils. The worksheets completed by pupils themselves can serve as notes from Python lessons. Our intention is that the worksheets do not restrict pupils; not to force them to strictly follow a defined sequence of steps, but to lead them to think about problems, use different problem-solving strategies, thoroughly analyze problems, and not just rely on trial-and-error approaches.

III. FINDINGS FROM TESTING OF THE TEACHING AND LEARNING MATERIALS

The teaching and learning materials have been verified at partner secondary schools in two runs. During the school year 2018/2019, there were 24 teachers involved in the verification and 20 teachers during the following school year. At least 8 teachers verified all 27 teaching and learning materials in each run. Based on their comments and recommendations, each year the original version of the materials was adapted.

The testing of the proposed teaching materials is currently still running at more than 25 secondary schools across Slovakia. We communicate with teachers via LMS Moodle. We receive feedback by:

- discussion forum posts,
- questionnaire for evaluation of the teaching materials after the lesson,
- quantitative and qualitative analysis of problems’ solutions.

Based on feedback from teachers, here are the initial findings:

- Teachers appreciate comprehensive support for teaching (teaching materials, worksheets, teacher training, etc.),
- the level of pupils in the groups varies, that complicate teachers' pedagogical interventions,
- lessons are time consuming in cases where pupils do not have the necessary basic programming knowledge and skills,
- in the beginning pupils are not accustomed to IBL, they gradually adopt this approach and learn to use it effectively,
- pupils have difficulty using knowledge from other school subjects to solve problems with programming tools,
- pupils gradually learn to work together (by discussions, work in pairs),
- some teachers required additional tasks for their pupils,
- some teachers have proactively extended the content of the lesson, because of providing the complete learning content according to their opinions,
- only some teachers have identified themselves with inquiry-based teaching materials, especially those who have already had some knowledge and experience with inquiry-based teaching or have attended our training,
- teachers who have experience in IBL, as a student in our lessons, implement IBL better,
- pupils are not accustomed to using worksheets on informatics lessons,
- we initially assumed that teachers print worksheets for pupils, but because of the high printing costs, teachers asked for worksheets in editable electronic form, which some teachers find problematic because it is difficult for pupils to program and read and complete the worksheet on one screen at the same time,
- teachers tend to inappropriately transfer approaches from other programming languages to Python.

These and other findings from the testing we exploit in final versions of teaching materials.

IV. CONCLUSION

We consider solving problems with programming tools as a key area of school informatics. In this paper, we presented the proposal for an innovative way of teaching programming, using modern teaching approaches and practices developing inquiry skills and problem solving skills of pupils. We also respect specifics and approaches of Python programming.

Based on our experience [18] we recommend for better implementation of IBL in informatics, that the
teacher progress through the following roles: a pupil who explore, a teacher with theoretical bases of IBL using IBL lessons, a teacher adopting existing IBL lessons, a teacher creating own IBL lessons.

At the end of the project, efficacy of our methodological approach will be evaluated based on pre-testing and post-testing pupils' computational thinking level and feedback from teachers after their lessons. Final version of teaching materials will be released by the end of September 2020 under the CC BY SA license for use in all secondary schools.

Developed teaching and learning materials are unique in Slovakia and they provide comprehensive support for teachers and pupils. Ultimately, it is up to secondary school teachers to decide whether and to what extent they will use the materials in their programming lessons.

ACKNOWLEDGEMENT

This article was created in the framework of the National project IT Academy – Education for the 21st Century [19], ITMS: 312011F057, which is supported by the European Social Fund and the European Regional Development Fund in the framework of the Operation Programme Human Resources and in the frame of project KEGA 029UKF-4/2018 Innovative Methods in Programming Education in the University Education of Teachers and IT Professionals.

REFERENCES


