ISSN:2828-6235
Impact factor:13.77

International Journal of Literature and Language Studies(IJLLS)

# METHODOLOGY OF USING MODERN PEDAGOGICAL TECHNOLOGIES IN TEACHING THE PYTHON PROGRAMMING LANGUAGE

**Khodjaeva Dilnoza Mutalibovna**
Academic Lyceum of Andijan State Technical Institute.
Doi: https://doi.org/10.5281/zenodo.22840273

**Abstract:**This article examines the use of modern pedagogical technologies in teaching the Python programming language. It focuses on project-based learning, problem-based learning, flipped classroom, gamification, collaborative learning, and interactive coding activities. The proposed methodology combines theoretical knowledge with practical tasks, problem solving, project development, and formative assessment. These approaches can increase students' motivation, programming skills, computational thinking, and ability to solve problems independently.

**Keywords:** Python, programming education, pedagogical technologies, project-based learning, computational thinking, interactive learning.

## INTRODUCTION

The rapid development of information and communication technologies has increased the importance of programming education. Programming helps students develop logical thinking, creativity, computational thinking, and problem-solving skills. Among modern programming languages, Python is widely used in education because of its simple syntax, readability, and broad application in data analysis, artificial intelligence, web development, and scientific computing.

However, effective Python education requires more than memorizing programming commands. Students need opportunities to apply theoretical knowledge, solve practical problems, develop projects, and analyze their own mistakes. Traditional lecture-based instruction may not provide sufficient opportunities for active learning.

Therefore, the use of modern pedagogical technologies is important for improving the quality of programming education. Project-based learning, problem-based learning, flipped classroom, gamification, and collaborative learning can make Python lessons more interactive and practice-oriented.

The purpose of this article is to examine effective modern pedagogical technologies for teaching Python and to propose a practical methodology for their application.

## MATERIALS AND METHODS

The proposed methodology is based on learner-centered, competency-based, activity-based, and problem-based approaches. The learning process consists of several stages:

1. Motivation – demonstrating the practical applications of Python.
2. Theory – explaining basic programming concepts.
3. Practice – completing short coding exercises.
4. Problem solving – developing algorithms and solving programming tasks.
5. Project work – creating individual or group programs.
6. Assessment and reflection – evaluating results and correcting mistakes.

This approach allows students to gradually move from theoretical knowledge to independent programming.

Modern Pedagogical Technologies in Python Teaching

Project-Based Learning

Project-based learning connects theoretical concepts with practical results. After studying variables, conditions, loops, lists, and functions, students can develop simple applications such as a calculator, quiz program, grade calculator, expense tracker, or text-based game.

Projects should be divided into smaller stages: identifying the problem, designing an algorithm, writing code, testing, debugging, and presenting the final product. This method develops programming, creativity, planning, communication, and teamwork skills.

Problem-Based Learning

Problem-based learning encourages students to find solutions independently. Instead of providing a ready-made algorithm, the teacher presents a practical problem.

For example, students may be asked to create a Python program for calculating the average grade of several students. They must determine what data are required, how to store them, and how to calculate and display the result.

Such tasks help students understand the practical purpose of programming constructions rather than simply memorizing their syntax.

Flipped Classroom

The flipped classroom allows students to study basic theoretical material before the lesson using videos, presentations, or electronic resources. Classroom time is then devoted mainly to coding, problem solving, and teacher-supported practice.

For example, students can study if, elif, and else structures before the lesson and then solve practical tasks during class. This approach gives the teacher more time to provide individual assistance and correct programming errors.

Gamification

Gamification introduces game elements into the learning process. Students can receive points, badges, or achievements for completing programming tasks.

A Python course can be organized into levels:

- Level 1 – Variables and Data Types;
- Level 2 – Conditional Statements;
- Level 3 – Loops;
- Level 4 – Functions;
- Level 5 – Data Structures;
- Level 6 – Mini Project.

Gamification can increase students' interest and motivation while maintaining the main educational objectives.

Collaborative Learning

Programming can also be effectively taught through pair and group activities. In pair programming, one student writes the code while another analyzes the solution and identifies possible errors. Students can then exchange roles.

Collaborative learning develops communication, teamwork, critical thinking, and the ability to explain programming solutions.

Interactive and Practical Learning

One of the most important principles of Python teaching is the immediate connection between theory and practice. After explaining a new concept, students should complete a related programming task.

For example:

```
name = input("Enter your name: ")
age = int(input("Enter your age: "))

print("Hello,", name)
print("Your age is", age)
```

After studying loops, students can solve practical problems such as calculating the sum of numbers:

```
total = 0

for i in range(1, 101):
total += i

print(total)
```

The learning process should gradually develop according to the following sequence:

Understand → Reproduce → Modify → Apply → Create

This progression helps students move from simple exercises to independent programming.

Differentiated Instruction and Assessment

Students have different levels of previous programming experience. Therefore, tasks should be differentiated according to their abilities.

Basic level: variables, input/output, and arithmetic operations.

Intermediate level: conditions, loops, lists, and functions.

Advanced level: complex algorithms and independent projects.

Assessment should measure both theoretical knowledge and practical programming skills. Teachers can use coding exercises, quizzes, debugging tasks, project presentations, peer assessment, and self-assessment.

An important assessment principle is to evaluate not only whether the final program works, but also whether students understand the algorithm and can explain their solution.

**RESULTS AND DISCUSSION**

The integration of modern pedagogical technologies can improve several aspects of Python education. Practical projects increase the connection between programming theory and real-world applications. Problem-based learning develops computational thinking and logical reasoning. Flipped classroom provides more time for practical activities, while collaborative learning strengthens communication and teamwork.

The role of the teacher also changes. Instead of only explaining programming concepts, the teacher becomes a facilitator and mentor who organizes activities, provides feedback, and helps students overcome difficulties.

An effective methodological model can therefore be presented as:

Motivation → Theory → Practice → Problem Solving → Collaboration → Project → Reflection → Assessment

This model provides a gradual transition from theoretical knowledge to practical programming competence.

**CONCLUSION**

Teaching Python should not be limited to memorizing syntax and programming commands. The main goal should be to develop students' ability to analyze problems, design algorithms, write and debug programs, and create independent digital solutions.

Modern pedagogical technologies provide effective opportunities to achieve these goals. Project-based learning develops practical skills, problem-based learning strengthens logical and computational thinking, flipped classroom increases practical classroom time, gamification supports motivation, and collaborative learning develops communication and teamwork.

The proposed methodology can be used in schools, academic lyceums, vocational institutions, and higher education. Combining theoretical explanations with practical programming tasks, projects, problem solving, and formative assessment can contribute to the development of students' programming competence and prepare them for further education and professional activity in the digital economy.

**References**

1. Python Software Foundation. *Python Documentation*. Python.org.
2. Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.
3. Bell, S. (2010). Project-based learning for the 21st century: Skills for the future. *The Clearing House, 83*(2), 39–43.
4. Bishop, J. L., & Verleger, M. A. (2013). The flipped classroom: A survey of the research. *2013 ASEE Annual Conference & Exposition Proceedings*.
5. Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.
6. Freeman, S., et al. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences, 111*(23), 8410–8415.