METHODOLOGY OF FORMING STUDENTS' COMPETENCE IN CREATING CUSTOM MODULES USING PYTHON

Khudoyberdieva Shoira, Navoi State University, Navoi, Uzbekistan
shoira0594@gmail.com

Abstract: This article discusses the pedagogical and methodological framework for developing student's competencies in designing and implementing reusable custom modules using the Python programming language. In modern software development, code reusability and modular architecture are critical skills for efficient programming. However, students often face challenges when structuring code outside of standard libraries. The proposed methodology integrates project-based learning and practical module development tasks to enhance students' coding standards, maintainability, and problem-solving skills in Python.

Keywords: custom modules, code reusability, competence, methodology, programming education.

The rapid evolution of information technology and software engineering demands highly skilled developers who can write clean, scalable, and maintainable code. One of the core principles of modern programming is code reusability, which significantly reduces development time and minimizes errors. In academic settings, while students quickly learn basic syntax, control structures, and built-in functions in Python, they often struggle with creating independent, reusable custom modules (such as custom .py files containing functions, classes, and utilities) and integrating them into larger software projects.

Traditionally, programming education focuses heavily on basic syntax and small scripts, while modular architecture is often ignored. Consequently, learners face difficulties when moving to team projects or large applications. The main goal of this

study is to create a modern pedagogical approach for building custom module creation skills in python. This helps students connect theoretical concepts with real software development tasks.

To effectively build students' competence in creating custom modules, the learning process is divided into three consecutive methodological stages.

First, the conceptual stage focuses on teaching namespace management, variable scope, and the mechanics of the *import* mechanism in Python. Student learn how Python locates *.py* files and how to structure individual script files properly.

Second, the practical integration stage involves teaching encapsulation principles. Students practice writing independent functions and classes that perform single, well-defined tasks, ensuring their code can be easily reused across different files.

Finally, the project-based stage requires learners to apply these skills within collaborative mini-projects, where they build and share custom utility packages, bridging the gap between theoretical syntax and professional software development.

In conclusion, the integration of a targeted pedagogical methodology for developing custom module creation skills in Python significantly enhances students' programming competence. By focusing on modular architecture, clean code structure, and practical code reusability, learners successfully transition from writing isolated scripts to designing professional-grade software components. Future research will explore the implementation of automated assessment tools to evaluate the efficiency and quality of student-generated modules in diverse academic environments.

References

1. Lutz, M. (2013). Learning Python: powerful object oriented programming. O'Reilly Media.
2. McKinney, W. (2022). Python for data Analysis: Data wrangling with Pandas,

NumPy, and IPython, O'Reilly Media.

3. Mirsanov U., Ruzieva D., Khudoyberdieva Sh., Kilichev Kh., Esanbaev B. 2025. Effective methods for organizing laboratory sessions in programming language courses. AIP conf.Proc.3268,070020. https://doi.org/10.1063/5.0257142