NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Data Science and Business Analytics"

Software Project Report
on the topic
Development of Materials for Teaching Programming in Python

(interim, the first stage)

Fulfilled by the Student:
group #БПАД__212__ Kulakov Denis
Surname, First name, Patronymic, if any

Checked by the Project Supervisor:
Voznesenskaya Tamara Vasilievna
Surname, First name, Patronymic (if any), Academic title (if any)
Associate Professor, Department of Big data and Information search
Job
Department of Big Data and Information search
Place of Work (Company or HSE Department)

Moscow 2024

Contents

Basic terms and definitions

- **Python** - A high-level, interpreted programming language known for its readability and broad applicability in various programming tasks.

- **Educational Materials** - Resources created to support learning objectives, including texts, videos, and interactive exercises.

- **Git Repository** - A version-controlled system used to store and manage project files, allowing for collaborative contributions and modifications.

- **Docker** - A set of platform-as-a-service products that use OS-level virtualization to deliver software in packages called containers.

- **School student guide** - An advanced class for students, especially in a particular discipline, conducted by an expert or professional.

Introduction

The project titled “Development of Materials for Teaching Programming in Python” seeks to develop a set of pedagogical materials for teachers to use to teach Python programming to the secondary school audience. Python is selected as the introductory language in computer science education with increasing frequency and for good reason. Its syntactic clarity and flexibility make it an excellent tool both for the teaching of foundational programming concepts and for building complex applications. The goal of this project is to carefully construct a set of instructional materials that is both comprehensive and pedagogically rigorous, in line with the academic needs of modern students.

This effort is underpinned by the use of modern software development tools, Docker and Git, that serve to expose students to current industry standards and practices. Docker is a tool for crafting lightweight, self-contained computing environments that remove disparities between computational settings that might arise from differential local development environments. Git is a distributed version control system that exposes students to workflows of collaborative and iterative development that are in widespread use throughout the software industry today.
The project aspires to deliver advanced educational experiences through the construction of master classes. These are designed to provide exposure to expert knowledge and sophisticated programming concepts, and will build on the foundational curriculum to enrich the learning experiences of the students.

The ultimate goal is to construct an educational guide that not only transmits technical knowledge, but also equips students with the skills they will need to manage the intricacies of modern computational problem solving. By addressing this goal in a systematic, evidence-based fashion, the hope is that the project will make a meaningful contribution both to the broader field of computer science education and the pipeline of future technologists who will enter the technology workforce.

Plan for Further Work

The plan for further work is structured into several key areas, each with specific steps to ensure the project's objectives are met effectively and efficiently. The areas of focus include the development and execution of master classes for students, the incorporation of Docker for creating consistent development environments, and the use of Git for version control and collaboration. Additionally, there will be an ongoing effort to develop the curriculum, create documentation, and ensure the accessibility of materials.

Master Classes for School Students:

Curriculum Development:

- Identify key Python programming concepts suitable for master class treatment.
- Design a series of master classes that build upon each other, allowing students to progress from basic to advanced topics.
- Develop lesson plans that include objectives, materials, activities, and assessment methods for each master class.

Resource Creation:

- Produce supplementary materials, such as slide decks, code examples, and reference sheets.
- Create hands-on exercises that encourage active learning and problem-solving.
- Develop evaluation tools to assess student understanding and the effectiveness of the master classes.

Working with Docker:

Containerization Concepts:

- Introduce the concept of containers, explaining their advantages over traditional virtualization.
- Develop tutorials that demonstrate the setup and management of Docker containers.

Practical Exercises:

- Design lab exercises that guide students through the process of building, running, and managing Docker containers.
- Create project-based assignments that require the use of Docker to set up development environments for Python programming.

Integration with Curriculum:

- Align Docker learning modules with the existing Python curriculum to reinforce the application of containerization in software development.

- Plan checkpoints where students must demonstrate their proficiency in using Docker as part of their programming assignments.

Working with Git:

Git Workflows:

- Create a module that covers basic Git commands and workflows, including cloning repositories, committing changes, and pushing to remote repositories.
- Teach branching strategies and how to resolve merge conflicts effectively.

Collaborative Projects:

- Design group projects that require students to collaborate using Git, mimicking real-world development workflows.
- Introduce code review practices and the use of pull requests to improve code quality and collaboration.

Assessment of Git Skills:

- Develop assessments that require students to demonstrate their ability to use Git in various scenarios.
- Collect feedback from students on the usability of Git and their confidence in using version control.

The project's advancement strategy is to develop and roll-out master class lessons for Python programming, integrate Docker to create and manage uniform development environments, and use Git for version control and collaborative development. The curriculum will expand through a tiered approach, building on basic concepts and moving into more advanced topics. Interactive exercises and evaluative tools will be provided to support this evolution in educational resources.

In parallel, Docker will be introduced to students through tutorials and lab exercises to ensure that they also gain practical exposure to the related core concepts such as container management, which will also be integrated into the programming curriculum. Git will teach students the core commands and collaborative workflows. Students will practice real world development scenarios as they work on group projects, while the effectiveness of these educational components will be measured through targeted evaluations and feedback mechanisms to ensure improved student proficiency in Python programming, as well as essential software development practices.

Roadmap

Master Classes for School Students:

Curriculum Development:

Identify Python concepts: Deadline - March 5, 2024

Design master class series: Deadline - March 15, 2024

Develop lesson plans: Deadline - March 25, 2024

Resource Creation:

Supplementary materials production: Deadline - April 5, 2024

Creation of hands-on exercises: Deadline - April 15, 2024

Development of evaluation tools: Deadline - April 25, 2024

Working with Docker:

Containerization Concepts:

Introduction to containers: Deadline - March 10, 2024

Tutorials on Docker setup and management: Deadline - March 20, 2024

Practical Exercises:

Lab exercise design: Deadline - April 10, 2024

Project-based assignments: Deadline - April 20, 2024

Integration with Curriculum:

Docker module alignment: Deadline - April 30, 2024

Proficiency checkpoints: Deadline - May 10, 2024

Working with Git:

Git Workflows:

Basic Git module creation: Deadline - March 15, 2024

Branching strategies and merge conflicts: Deadline - March 25, 2024

Collaborative Projects:

Group project design: Deadline - April 15, 2024

Code review and pull request integration: Deadline - April 25, 2024

Assessment of Git Skills:

Development of Git assessments: Deadline - May 5, 2024

Collection of feedback: Deadline - May 15, 2024