## 

## 

**Experiment 1:**

**Using Software Tools and Code Versioning System**

CPE106L (Software Design Laboratory)

**Member 1 (Ice Marcux B. Canda)**

**Member 2 (Eleazar A. Collamat)**

**Member 3 (Elmo Joaquin Estacion)**

**Member 4 (Gian Trinidad)**

Group No.: **10**

Section: **B2**

## **Prelab**



|  |
| --- |
| **Readings, Insights, and Reflection**  [[**What to Include?**  Readings include METIS books, pertinent websites. Provide below your Insights and Reflection.] |
| ***<Insights and Reflections here…>***  Example Readings, Insights, and Reflection  **Professional Git**  **9781119285007**  **<CANDA>**   * Our initiation into GitHub through the course "Introduction to GitHub" provided a structured and hands-on approach to mastering the platform's essential functions. By following a quick reference guide offered by GitHub's own learning platform, we navigated through fundamental actions such as assigning myself to issues, creating branches, committing files, opening pull requests, responding to reviews, and merging pull requests. These steps, meticulously outlined in the course material, demystified GitHub's core functionalities, empowering us to efficiently manage version control and collaborate effectively with peers. Through this guided journey, we grasped the significance of each action, from assigning oneself to an issue to merging pull requests and recognized their collective impact on streamlining development workflows and promoting collaborative software development practices. As we reflect on this introductory experience, we were poised to leverage GitHub as a powerful tool for version control, collaboration, and knowledge sharing, propelling our journey as a developer forward within the dynamic landscape of software development.   **<COLLAMAT>**   * The "Introduction to GitHub" course provided a structured approach to mastering essential platform functionalities, such as issue assignment, branch creation, and pull request initiation. Utilizing GitHub's reference manual, we effectively managed version control and learned to collaborate with peers. The course meticulously detailed each phase, emphasizing the significance of tasks like assigning challenges and aggregating pull requests. Through this journey, we grasped the collective impact on streamlining workflows and fostering collaborative software development approaches. Reflecting on our experience, we now confidently utilize GitHub for version management, knowledge sharing, and collaboration, advancing our skills as software developers.   **<Estacion>**   * The book "Professional Git" is a rundown of the Git program. It taught us about the concepts and commands that are needed for version control mastery, With the terms "Commit," "Histories," "Branching," and "Merging." The Book also discusses the nuances of local and remote repository management; the Book teaches the readers the knowledge graph of complex codebases and collaborative work. Each chapter would methodically break down the various functionalities of Git, emphasizing the importance of rebase, tag creation, and conflict resolution. By the end of the Book, we were able to master the technical skills required for specific version control and enhance collaborative development   **<Trinidad>**   * “Professional Git” provided an effective starting point wherein we learned on how to perform basic operations like allocating tasks, committing modifications, etc. By following a guide (the book), this practical method emphasized git features. In this book, we discovered the importance of each step from allocating roles to combining code, and their combined effect on optimizing processes and promoting teamwork. This experience put us in a position to make use of GitHub’s strength as a medium for knowledge sharing, which is essential to open-source development, as well as for tracking code versions. This book helped us gain a deeper comprehension of the fundamental ideas and commands of version control. It gave us knowledge and the ability to efficiently navigate complex codebases and work together on big projects. Functionalities like rebasing, tagging, and resolving conflicts were efficiently tackled in different chapters. With this, we gained a technical understanding on how to improve and master our capacity with collaborative development.   **Fundamentals of Python: Data Structures, 2nd Edition**  **9780357122754**  **<CANDA>**   * Chapter 1 of "Fundamentals of Python: Data Structures, 2nd Edition" provides a foundational understanding of Python programming, emphasizing its simplicity, versatility, and readability. Through practical examples and exercises, we gained proficiency in essential concepts such as variables, data types, control flow, and functions. One of the key insights we gained from Chapter 1 is Python's versatility and expressiveness, which allows for concise yet powerful code that can be easily adapted to various problem domains. The dynamic typing system of Python enables flexible data manipulation, while its extensive standard library provides a rich set of built-in functions and modules for efficient development. Moreover, as we delved deeper into Python programming constructs like loops, conditionals, and functions, we appreciated their role in writing clear, maintainable, and reusable code. The chapter's focus on abstraction and problem-solving techniques equipped me with the skills needed to tackle real-world programming challenges.   **<COLLAMAT>**   * Programmers must understand Python data structures to organize and manipulate data. Mastering lists, tuples, and dictionaries provides a solid foundation for solving many problems. Learning about complex structures like sets and queues improves one's ability to solve complex programming problems. Practical tasks and examples are crucial to theoretical knowledge reinforcement because they foster experiential learning and skill development. Identifying data structure pros and cons helps choose the best ones for programming tasks, promoting effective problem-solving. Considerations of data structures in practical settings encourage critical thinking and problem-solving. Use and study of Python data structures helps acquire a deeper understanding of programming principles and improve software development skills.   **<ESTACION>**   * "Fundamentals of Python: This book "Data Structures, 2nd Edition" is a comprehensive guide to learning data structures in Python, suitable for both the beginner and the professional programmer who needs to expand their knowledge on the topic. The Book details basic concepts like lists, dictionaries, and sets. It takes you through advanced structures like trees and graphs with a hands-on approach, using examples, exercises, and real-world applications. The Book's current edition has been updated to include the latest Python features. It also ensures that the readers have the latest programming practices, which will help them build a strong base of theoretical knowledge and practical skills in data management and algorithmic problem-solving.   **<TRINIDAD>**   * "Fundamentals of Python: This book "Data Structures, 2nd Edition" provides us knowledge on the essential variables, data types, operators, and the flow that provides an effective framework. The book also provided learnings about data structures. Among the many methods for managing and organizing information, the book presented sets, dictionaries, tuples and lists. Through useful examples and exercises, the book thoroughly explained ideas like indexing for lists, the significance of flexibility, and dealing with complex structures. Lists serving as adaptable toolboxes, tuples providing data security, dictionaries operating as digital address books, and sets offering an additional tool for purposes all these features broadened the reader's vocabulary in programming and enabled them to effectively and methodically represent real-world problems.   **Git, Anaconda and Visual Studio Code**  **<CANDA>**   * Exploring Git, Anaconda, and Visual Studio Code has been instrumental in enhancing my development workflow and productivity. Git, with its powerful version control capabilities, has streamlined collaboration and code management, allowing for efficient tracking of changes and seamless integration with remote repositories. Anaconda's comprehensive package management and environment creation tools have simplified the setup and management of Python environments, enabling smooth execution of data analysis and machine learning tasks. Visual Studio Code, with its intuitive interface and extensive plugin ecosystem, has provided a versatile and feature-rich integrated development environment (IDE) for coding and debugging across various programming languages. By integrating these tools into our workflow, we have experienced improved efficiency, code quality, and collaboration capabilities, laying a strong foundation for effective software development practices.   **<COLLAMAT>**   * The integrated integration of Git, Anaconda, and Visual Studio Code (VS Code) facilitates the optimization of development workflows and enhances overall productivity. The version control feature of Git enables smooth cooperation and effective management of project evolution, whereas Anaconda streamlines Python programming by providing efficient package management and environment setup. The coding experience is enhanced by VS Code through its customized Integrated Development Environment (IDE) features, which encompass integrated Git support and code editing tools. The act of contemplating the utilization of these tools highlights the significance of efficiently harnessing them to produce code that is both clean and efficient, effectively handle dependencies, and optimize the development processes inside software projects.   **<ESTACION>**   * Our development process has been improved vastly by incorporating Git, Anaconda, and Visual Studio Code into it, due to which our productivity has increased, and workflow is smoother. Git's version control system has improved our team's ability to deal with code modifications in a time-efficient manner, allowing for smooth teamwork and simple integration of remote codebases. Anaconda has made our Python-oriented projects, especially those that work with data analysis and machine learning, much easier to create and maintain because it provides customized environments. However, Visual Studio Code is an IDE that supports many programming languages and helps in easy writing and debugging of code. It has a user-friendly interface and many extensions that make our coding practices more efficient.   **<TRINIDAD>**   * The version control system, called Git, has been amazing. The book "Professional Git" has been a great help, going deeper into complex ideas like conflict resolution and branching techniques. My confidence and ability to collaborate have increased because of my newly acquired Git proficiency, which has made it easy for me to work on projects with others. The scientific Python distribution, Anaconda, has changed the field. Its pre-configured scientific libraries have made setting up data-driven projects easier. users no longer must waste time troubleshooting installation; Anaconda offers a ready-made environment that has everything they need to get started right away. This effectiveness frees up to concentrate on the actual work at hand, which is creating code that solves challenging issues. Visual Studio Code was nice to use thanks to its simple interface, adaptable extensions, and integrated terminal. With features like code completion, syntax highlighting, and seamless Git and Anaconda integration, Visual Studio Code offers an environment that promotes efficiency and productivity. It feels equipped to take on ever-more-complex projects with confidence since VS Code streamlines the development process, Anaconda provides pre-built libraries, and Git manages my code versions. |
|  |