

Lab №1. Introduction to the tools

Introduction

Main topics of the laboratory classes in the course:

- data processing in the power industry;
- data visualization and analysis;
- code examples on Python – a simple and multi-purpose programming language most often used for big data processing and analysis;
- tools for collaborative code development.

F.A.Q.

1. Why do non-IT students need to learn programming, why can't they just use pre-existing software?

Answer:

- There are no programs for every possible situation.
- Basic understanding of software development allows for better use of other software products.
- Even basic programming skills can boost one's productivity by task automation, numerical modeling, and automatic generation of charts and diagrams.
- Data processing is the core of many IT applications. Fast and accurate data processing is not possible without programming.
- Learning programming skills gives non-IT students an opportunity to start a career in IT.

2. Why can't we just use MS Excel for data processing?

Answer:

- Excel and its alternatives can and should be used. However, not every data processing task can be effectively solved in Excel.
- There are some cases when spreadsheet editors are not suitable for the task.
 1. Larger amount of data can negatively affect the performance of spreadsheet editors and cause freezes/crashes.
 2. Any non-typical data manipulation requires programming by using either formulas or scripts, and there are better tools to create and apply scripts.
 3. The task is usually not limited to data processing. Sometimes, the task involves something from the list:

- applying a complex mathematical tool to data, such as wavelet transform, neural networks etc.;
- integrating the solution into a data system like a webstore;
- downloading data from certain sources based on specific criteria;
- looking for certain e-mails in a database and sending messages;
- processing digital data from and sending control commands to electrical devices.

3. Why should we learn Python if we learned Matlab/Delphi/C/VBA/etc. before?

Answer:

- Learning programming skill at an early stage should not be tied to a particular programming language. It is more important to understand algorithmization in general.
- Learning more than one programming language is beneficial for beginners because it teaches them to see the algorithm separately (not tied to a specific language) and the implementation of the algorithm in code.
- The authors of the course consider Python the most suitable for beginners based on a combination of criteria: simplicity, versatility, and high demand in tasks related to data processing and analysis.

1. The objective

The objective of the assignment is to get acquainted with the tools necessary for completing the course and useful for future work in the following areas:

- development of algorithmic skills;
- development of programming skills;
- data processing, analysis, and visualization;
- machine learning;
- organization of individual and collective project work.

It is recommended to complete the assignment in teams of two.

2. GitHub

GitHub (<https://github.com/>) is a cloud platform for hosting IT projects. It is called "cloud" because it runs on remote servers, not a program that can simply be installed on your computer. It is called a "platform" because it is designed to solve multiple tasks. The symbol of GitHub is an octocat (a cat-octopus), as shown in Figure 1.



Figure 1. GitHub logotype

2.1. Relevance of GitHub for Students

For IT students, even a basic understanding of version control systems and platforms for collaborative project work is very important, as this skill is required in any professional IT company. The habit of uploading your projects, even academic ones, to GitHub can help create a portfolio over several years of study, which can be shown during interviews to demonstrate that a career-starting specialist possesses certain skills.

For students in technical non-IT fields, this is less critical, but firstly, using such platforms helps better organize work on projects during studies. Secondly, a minimal proficiency in GitHub is often required for completing online courses. For example, on Coursera, many practical assignments need to be uploaded to GitHub so they can be reviewed.

2.2. GitHub Capabilities

GitHub is used for solving many tasks, and the main ones are listed below.

1. Version Control, File Storage, and Change History Tracking. This ensures that no coursework disappears without a trace after "accidental" disk formatting, and users don't have to worry about ruining a document by reworking a particular section.
2. Collaborative Project Work. Primarily for working on code in IT projects. In a team of four, GitHub prevents situations where Vasya does something, sends it to everyone, Petya changes half of it, and Fedya and Kolya also make changes, resulting in four different versions of the same file with no one understanding who changed what and how to merge it all. Some GitHub projects are worked on by hundreds of people simultaneously.
3. Project Distribution. After creating something useful or interesting, it can be uploaded to GitHub for public use. This can lead to recognition, constructive feedback, or finding like-minded contributors. Many companies publish their solutions as open-source (where the entire code is made public) to attract clients, partners, and employees. On GitHub, you can find projects from Google, Facebook, Microsoft, Apple, and other IT giants.

GitHub's functionality continues to expand. For example, since 2019, it has become possible to create private projects, accessible only to those working on them. You can also host your own website for free on GitHub (<https://pages.github.com/>), which will be available via a link like <https://user.github.io>, where "user" is replaced with your account name.

It's important to distinguish between GitHub and Git. The former is an IT company and the cloud platform it created. The latter is a version control system that sets the rules for project work. GitHub uses Git, but Git exists independently, just as websites often use HTML, but HTML itself is simply a set of rules or a markup language.

2.3. Instructions and Task 1

Objective: Learn to create GitHub repositories, work with files both individually and as part of a team.

1. **Create an Individual GitHub Account** on the website <https://github.com/>. Choose a meaningful and appropriate username, and provide an email that you regularly use and are unlikely to lose access to.

Each member of the team should create their own account. If the team performs the assignment on the same computer, one team member can complete steps 2-8, then log out, allowing another member to repeat these steps.

2. **Create a Public Repository.** Each person creates their own repository. A repository is a place to store and manage project files. To create a repository, click the plus sign in the upper right corner on github.com and select "New repository" (Figure 2).

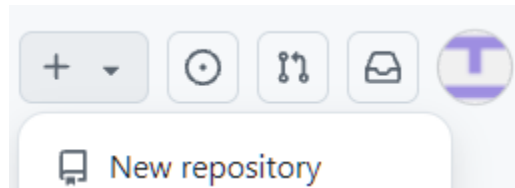


Figure 2. Repository creation tab

Next, enter the repository name in the "Repository name" field and a description in the "Description" field. For example: "My_First_Test_Repo" and "This is my first repository." Select the repository type as "Public" and check the box for "Initialize this repository with a README," as shown in Figure 3. After this, your repository will be accessible at <https://github.com/your-username/your-repository-name>.

Required fields are marked with an asterisk (*).

Owner * / Repository name *

☒ Test_Guide is available.

Great repository names are short and memorable. Need inspiration? How about [crispy-bassoon](#) ?

Description (optional)

☒ ☐ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ ☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set [main](#) as the default branch. Change the default name in your [settings](#).

☐ You are creating a public repository in your personal account.

[Create repository](#)

Figure 3. Repository creation menu

3. **Edit the README.md File.** Enter the group number and last names (without first names) of all team members. Each person should do this in their own repository.

Experienced developers use special tools for working with repositories, typically through a command prompt. However, for beginners, this may be unfamiliar, so the entire assignment will be done through the web interface.

In your repository, click on the pencil icon, as shown in Figure 4.

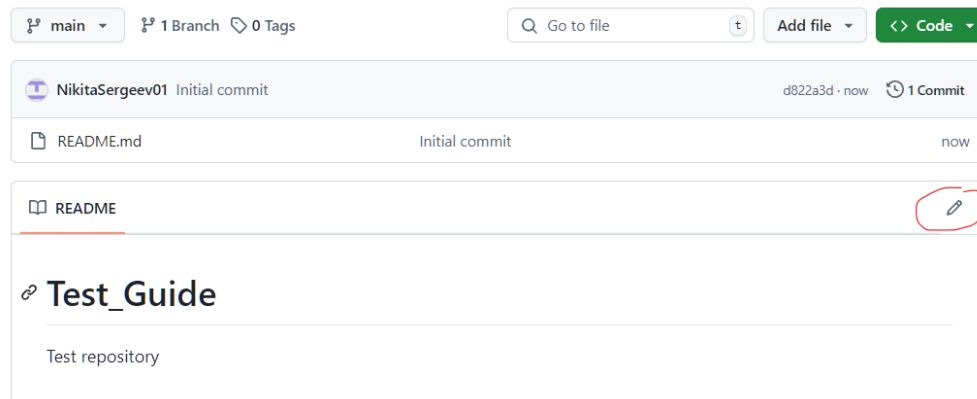


Figure 4. Editing a file in a repository

A window will open for editing the file. README files are usually in MD format—Markdown. Markdown is a special language for simplified text formatting. It will be covered in more detail later. For now, it's enough to know that headings are marked with hash symbols, and regular text is written just like in a text editor. To start a new line, you need to insert a blank line. Your edited file should look something like what is shown in Figure 5.

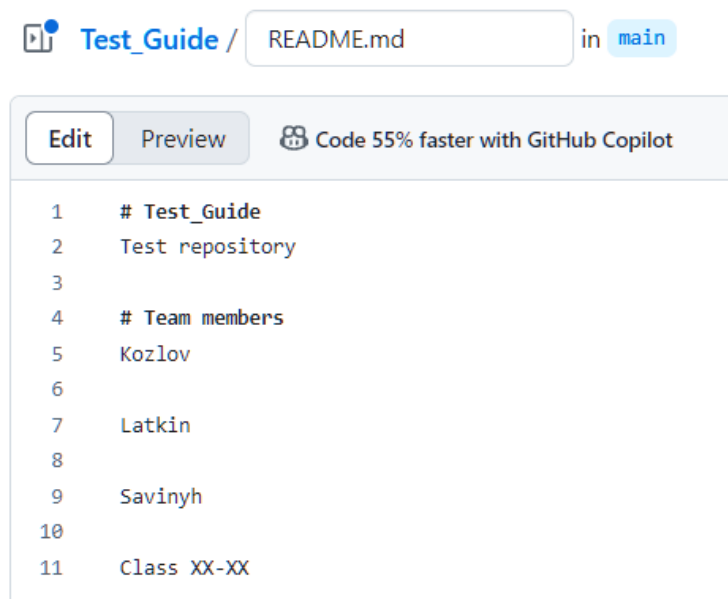


Figure 5. README.md editing

You can view the result by clicking on “Preview changes.”

4. **Make a Commit.** Each person should do this in their own repository. To save the changes to the README file, you need to make a commit. A commit is a snapshot of changes made to the files in a repository, and developers often say "commit" the changes. Each commit should have a comment that helps quickly understand what was done. An example is shown in Figure 6. After entering the comment, click "Commit changes."

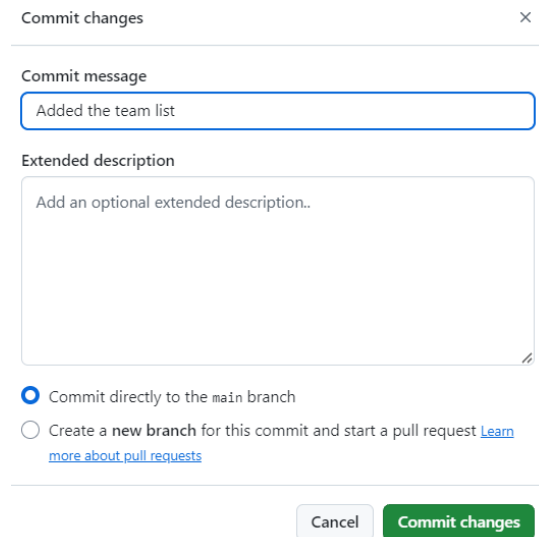


Figure 6. A commit

You will then be redirected to the main page of the repository, where you will see the applied changes.

5. **Create a New Branch.** Each person should do this in their own repository. In IT projects, development often occurs not in a linear step-by-step process but with parallel processes and branches, as shown in Figure 7.

Adding new features as a separate subproject

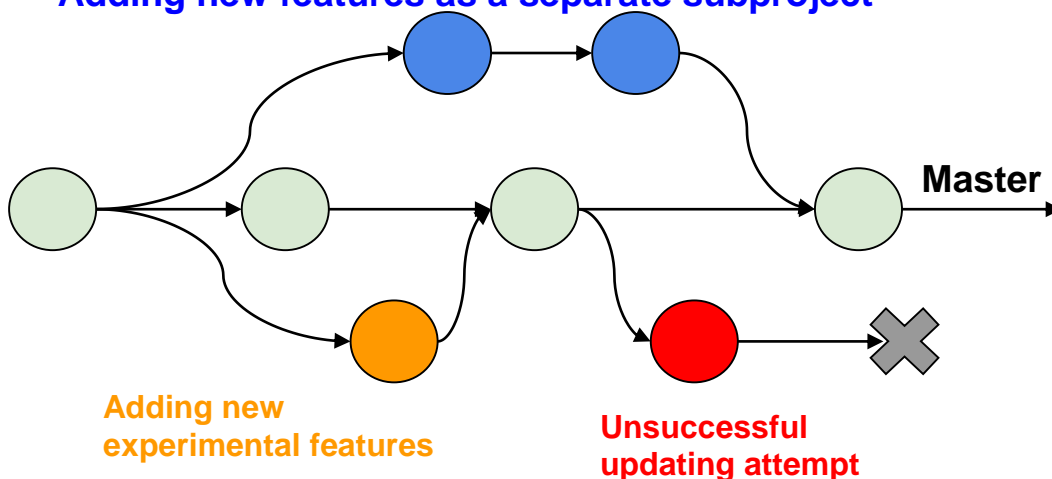


Figure 7. Branching

Master is the main branch. To work on a separate part of the project without interfering with other team members, it is common to create new branches, work in them, and then merge them with the main branch, as shown in the blue branch in Figure 7.

Even if only one person works on a project, it is convenient to create branches to experiment with changes. These changes might be useful or might not work out at all. If everything goes well, you can add the new features to the main branch (the orange branch in Figure 7). If not, you can simply leave it as a dead-end branch (the red branch in Figure 7).

This approach significantly reduces the risk of introducing changes that could damage the main branch. Of course, you could revert to a version before the damage occurred, but if useful changes were made, they would also be lost. Therefore, it is better to use separate branches.

An example of creating a branch is shown in Figure 8.

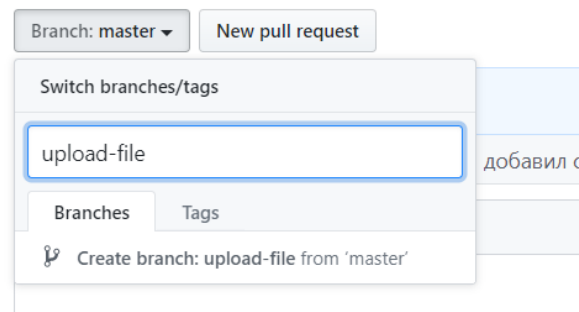


Figure 8. Creating a branch

Now, using the "Branch" dropdown menu, you can choose which branch to work on. You need to select the newly created branch.

6. **Upload a Directory with a File.** Each person should do this in their own repository.

If you had to create and edit all files directly on the GitHub website, it would be quite inconvenient. GitHub allows you to work with files on your computer and, when needed, upload them to the repository or download them to your computer. With some experience using Git version control, this can be done easily without even opening the GitHub website. However, for beginners, it's better to use the more manual method to understand the process.

To upload files, you can click the "Upload files" button on the main page of the repository. To access the main page, simply go to the Code tab at the top of the page, as shown in Figure 9 at the top, or if that tab is already open, click on the repository name link at the bottom of Figure 9.

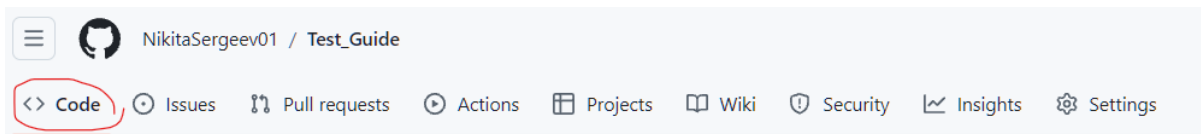


Figure 9. Navigating to the main page of the repository

After navigating to the main page, you can simply drag and drop files from your file explorer into the browser window. You can upload an entire directory with all its contents this way. After that, enter a commit message in the field and make the commit (Figure 10).

This way, while working on a project on your computer, you can periodically upload changes to GitHub.

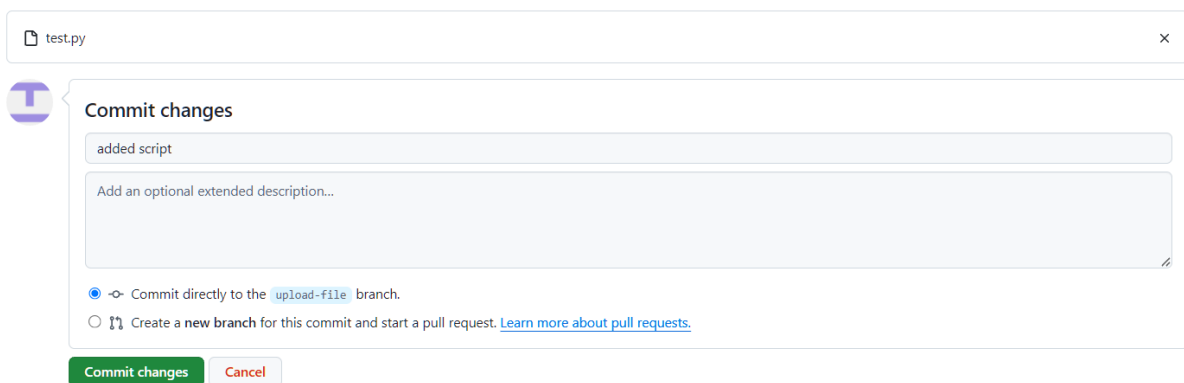


Figure 10. Uploading files to GitHub

Create a directory on your computer with a file and upload it to your repository.

7. Merge the New Branch into Master. Each person should do this in their own repository.

Navigate to the main page of your repository, click on “branches” to open the page with branches, then click “New pull request” for your branch (Figure 11 shows these two actions together; they are done sequentially).

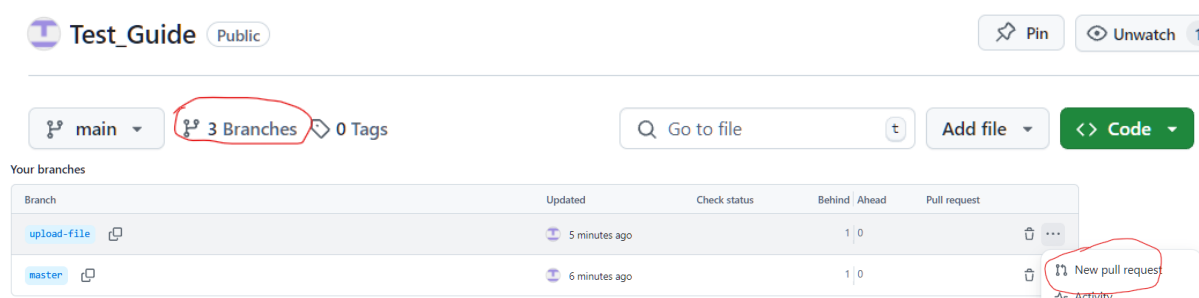


Figure 11. Merging branches

Next, select the branch into which you want to merge the changes. In this case, it's main. Then click "Create pull request." You will be redirected to a new page where you need to confirm the merge by clicking "Merge pull request," followed by "Confirm merge."

If you now go to the main page of your repository (<https://github.com/your-username/your-repository-name>) and select the master branch, you will see that the necessary files have been added.

8. **Grant Access to Your Repository to All Team Members.** Each person should do this in their own repository.

The repository you created is public, so anyone can view its contents. However, only those who are granted permission by the repository owner can make changes. To grant access:

1. Open the repository settings by clicking on "Settings."
2. In the menu, select "Manage access."
3. Click on the "Invite a collaborator" button.
4. Enter the email addresses or usernames of your team members.

They will receive an invitation via email and will be able to make changes to your repository.

9. **Add Your Name to the README.md Files.** Each person should add only their own name to their last name in each team member's repository.

In your own repository, you can add your name by simply editing the README.md file in the master branch. In other people's repositories, you need to create a new branch named *add-name-[your-name]*. For example, if the team members are Andrei Kozlov and Kirill Latkin:

- Andrei creates a branch *add-name-andrei* in Kirill's repository and adds his name to the README file.
- Kirill creates a branch *add-name-kirill* in Andrei's repository and adds his name to the README file.

10. **Merge Changes into the Master Branch.** Each person should merge the branches used to add names into the master branch in their own repository. As a result, each team member's repository should have a README file listing the names and surnames.

11. **View the Change History. Take a screenshot for your report.**

To view the history of all changes in the repository, go to the main page and click on “commits,” as shown in Figure 12.

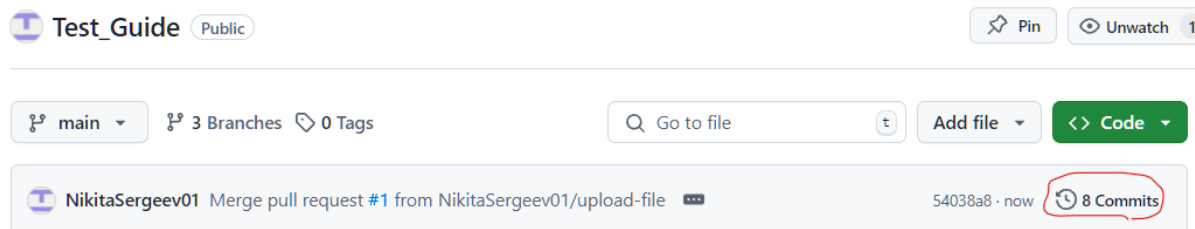


Figure 12. Commits

An example is shown in Figure 13. Each commit has a unique number—a hash, highlighted in red in the figure. Clicking on the hash allows you to see the changes made by that commit.

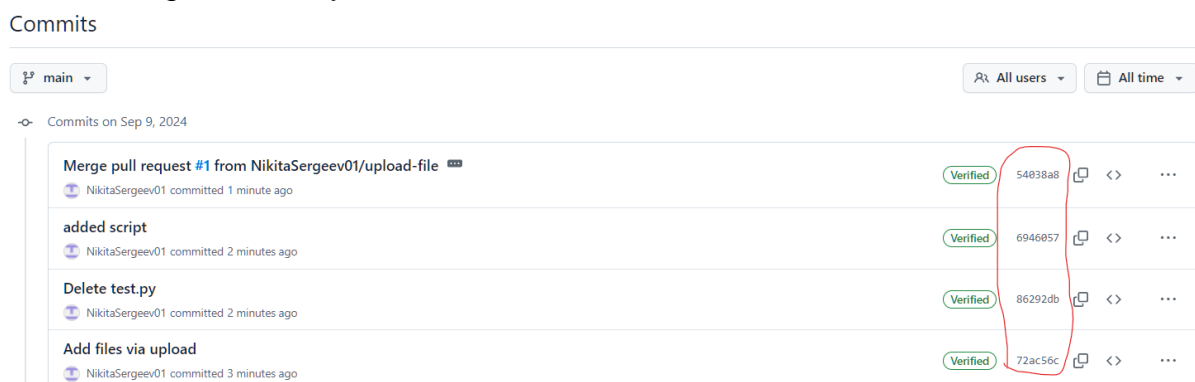


Figure 13. Commit list

Initially, working with GitHub might seem like it takes a lot of time. Over time, using Git becomes more familiar, and tasks can be completed quicker. Additionally, there are tools to simplify the process, such as GitHub Desktop, Git GUI, and Git Bash.

3. Google Colab + Jupyter

Google Colab is a cloud platform for creating and sharing solutions in data processing and visualization, machine learning, and programming in Python.

3.1. Relevance of Google Colab for Students

Firstly, Google Colab is convenient for students to start programming in Python and solving data-related tasks, as it requires no installation or setup on their computer. It allows students to work on projects both at home on their personal computers and on various university computers, as well as on classmates' computers, without the need to copy or send files.

Secondly, Google Colab offers several advantages that attract both beginners and professionals in mathematical modeling and data science.

3.2. Features of Google Colab

An overview of Google Colab can be obtained from the video “Get started with Google Colaboratory” (<https://www.youtube.com/watch?v=inN8seMm7UI>). On the Google Colab website, there are beginner examples available at https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=5fCEDCU_qrC0. A Google account is required to work with these examples.

Work in Google Colab is based on the Jupyter Notebook principle. This principle involves placing code, explanations, and results in one space divided into cells, as shown in Figure 14. This approach is especially convenient for programming tasks related to modeling, data analysis, and scientific research because everything is displayed on one screen. In contrast, the traditional approach to software development involves writing code in one set of files, documentation in another, and running the created program to obtain results.

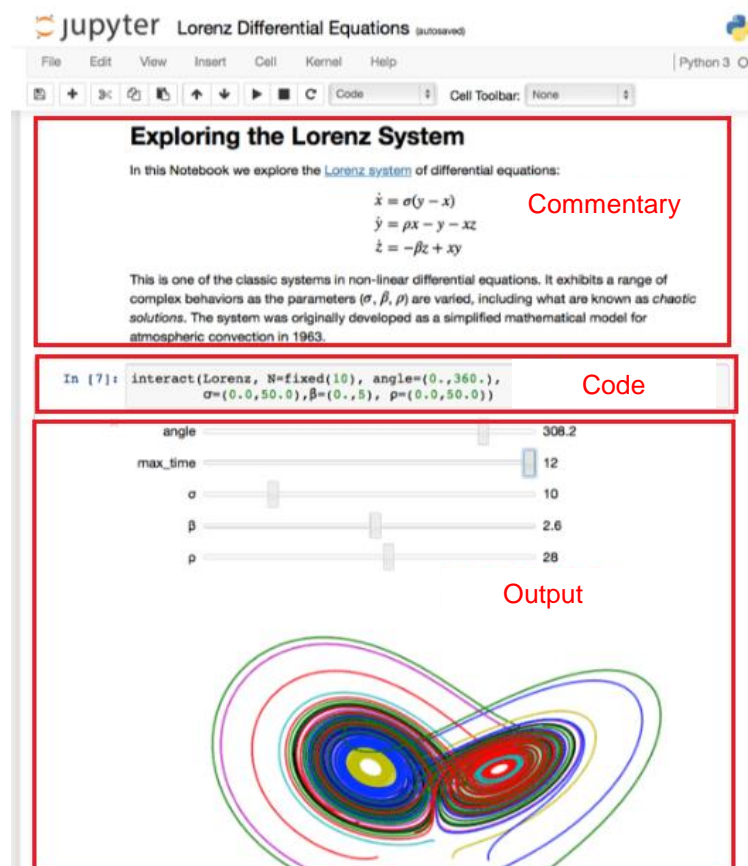


Figure 14. An example of Jupyter notebook from jupyter.org

Features and Advantages of Google Colab:

- It does not require installation on your computer and provides immediate access to a vast number of software libraries for solving tasks ranging from mathematical calculations to video recognition.
- Collaboration features for working on projects together.

- A project (notebook) created in Google Colab can be shared with others simply by sharing a link.
- The ability to accelerate calculations by using the provided GPU.
- Computationally intensive tasks are performed in the cloud, not on the user's computer.
- Integration with Google Drive for working with personal files.

Disadvantages:

- Requires an Internet connection.
- Not possible to work with files and programs on your computer directly, as everything is executed in the cloud.

3.4. Instructions and Task 2

Objective: Learn how to create projects (notebooks) in Google Colab, create descriptions using the Markdown markup language, and make commits to GitHub from Google Colab.

1. **Create a Google account** if you don't have one. One account is sufficient for the whole team.
2. **Watch the video and examples** provided in section 3.1.
3. **Create one notebook in Google Colab.** In it, create a Markdown cell with a heading, a bulleted list, an image, a table, and a link to a website. The content can be arbitrary. An example is shown in Figure 15.

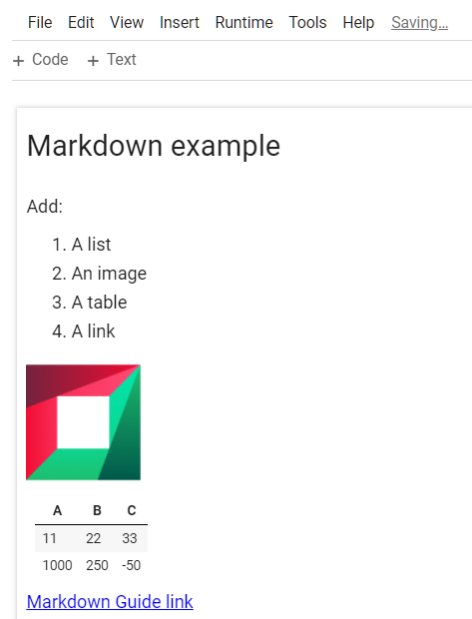


Figure 15. Markdown cell example

4. Pin the current version and give it a name.

Google Colab has a simple version control system. In the menu, select "File," then in the dropdown menu, choose "Save and pin revision." This action will save the current version of the notebook with a tag. Try making some changes and then save the new version again. After that, if you select "File" and then "Revision history," a list of versions will appear, as shown in Figure 16. In this list, you can name versions (e.g., "v.1.0") and compare two versions to see what has changed between them.

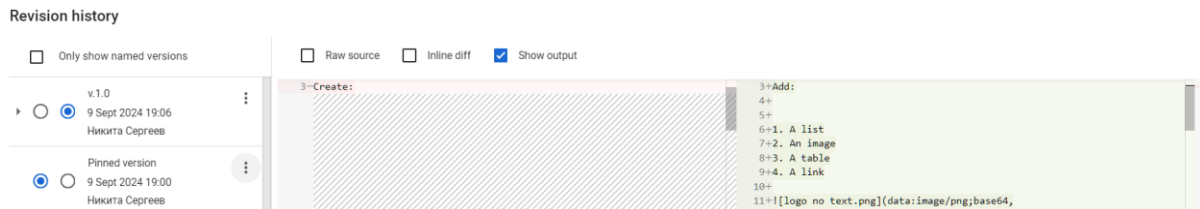


Figure 16. Google Colab version control

5. Grant the teacher access to the notebook (you will need to find out their email address associated with their Google account).

6. Add the notebook to the previously created GitHub account.

Although Google Colab has its own version control system, its creators understand that developers are fond of GitHub. That's why they added the ability to quickly send the notebook to a GitHub repository and vice versa. In the "File" menu, select "Save a copy in GitHub...", enter your GitHub account in the window that appears, and agree to grant access to it. After that, you can select your GitHub repository, choose a branch (master can be selected), and commit from Google Colab to GitHub, as shown in Figure 17.

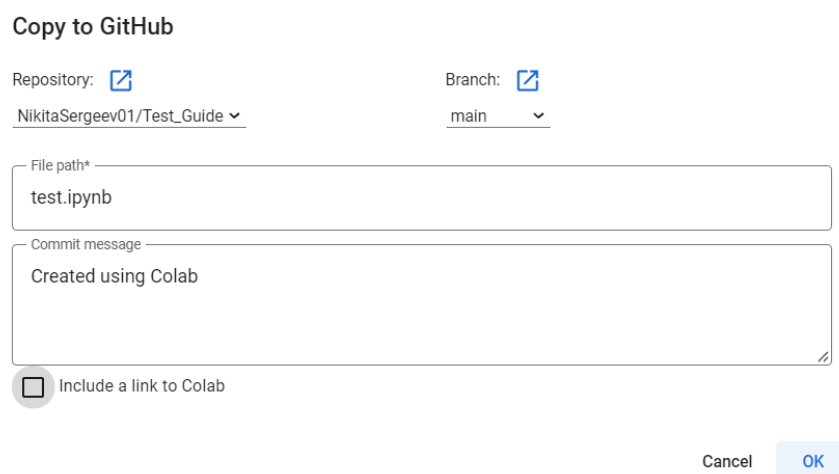


Figure 17. Commit from Google Colab to GitHub

Afterward, open your GitHub repository and ensure that your notebook has been added.

4. Report

Prepare the report in Google Docs, MS Word, or similar applications.

- Title page.
- Objective of the assignment.
- Names of the created accounts and repositories, indicating which belongs to whom. Example shown in Table 1.

Table 1. GitHub accounts and repositories.

Account	Repository	Student
pyotr-ivanov	My_test_repo (github repository link)	Pyotr Ivanov

- Screenshot of the repository's history (from any repository of the team).
- Screenshot of the final version of the README.md file (from any repository of the team).
- The code from the Markdown cell created in Google Colab.
- Screenshot of the notebook created in Google Colab, opened via GitHub.
- Conclusion.