**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая Кибернетика и Информационные технологии»

Лабораторная работа №10

Практическая работа с Git

Выполнила: Студентка группы

БВТ2402

Шилинцева Татьяна

Москва

2025

**Цель работы:**

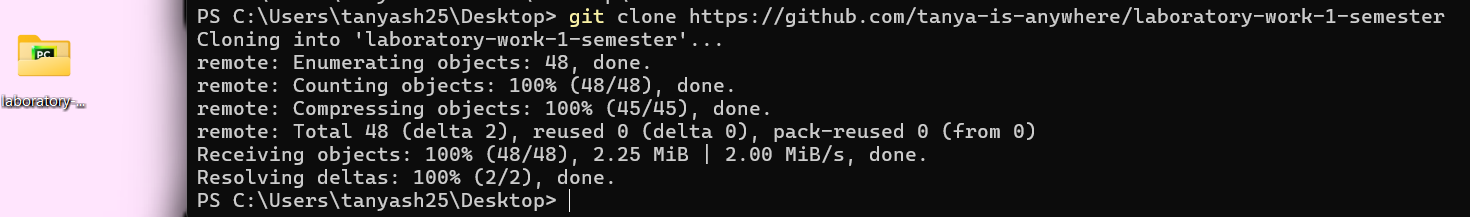
Познакомиться с основными операциями работы с удалёнными репозиториями в Git, включая клонирование, добавление удалённых репозиториев, отправку изменений и работу с ветками на удалённом репозитории.

***Задание 1 Клонировать удалённый репозиторий***

Инструкция:

1. Найти удалённый репозиторий на GitHub, GitLab или другой платформе.
2. На локальном компьютере выполнить команду git clone <URL удалённого репозитория>.
3. Убедиться, что репозиторий успешно клонирован на локальный компьютер.

Выполнение:



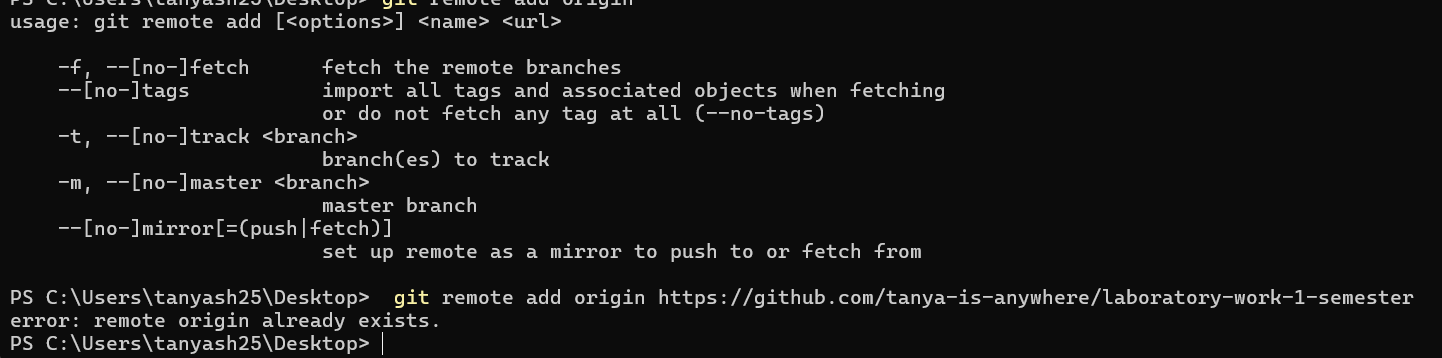
Можно заметить, что после выполнения команды на рабочем столе появилась соответствующая папка с названием репозитория.

***Задание 2 Добавить удалённый репозиторий***

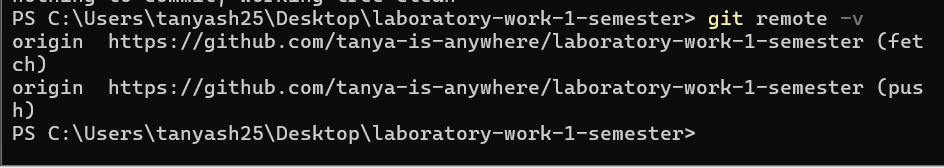
Инструкция:

В командной строке выполнить команду git remote add origin <URL удаленного репозитория> для добавления удалённого репозитория в локальный.

Выполнение:



Проверка добавления с помощью команды git remote -v



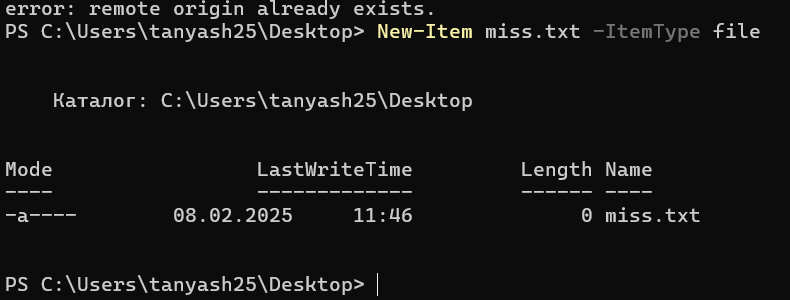
***Задание 3 Отправить изменения на удалённый репозиторий***

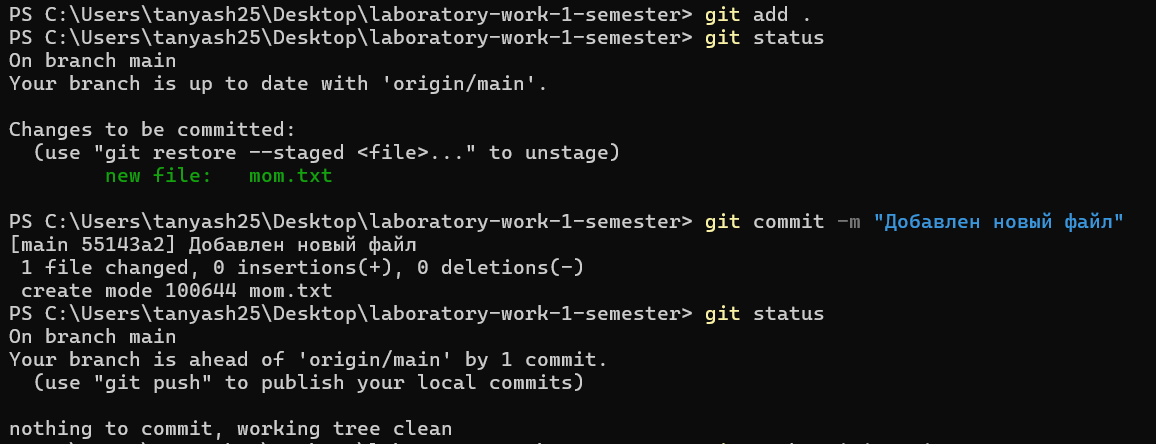
Инструкция:

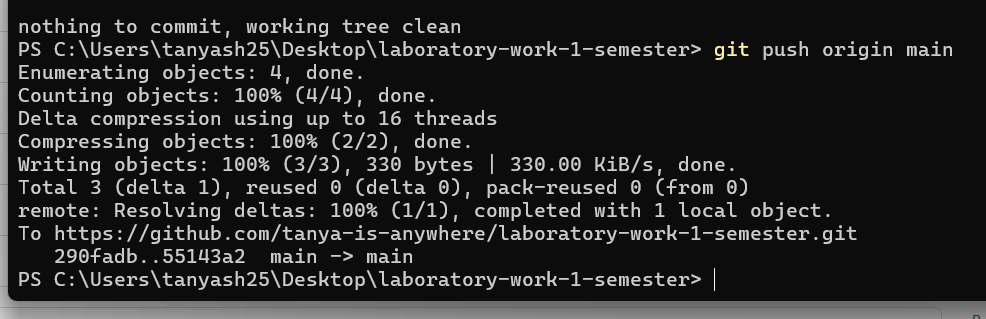
1. Выполнить команду git push origin <branch\_name> для отправки изменений из локального репозитория на удаленный.
2. Проверить, что изменения успешно отправлены на удаленный репозиторий.

Выполнение:

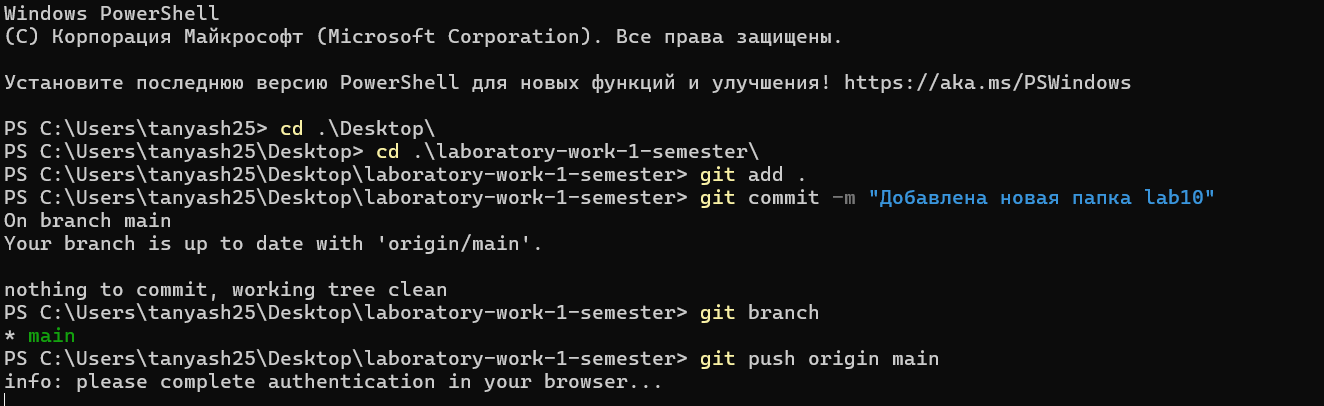
Внесение изменений:

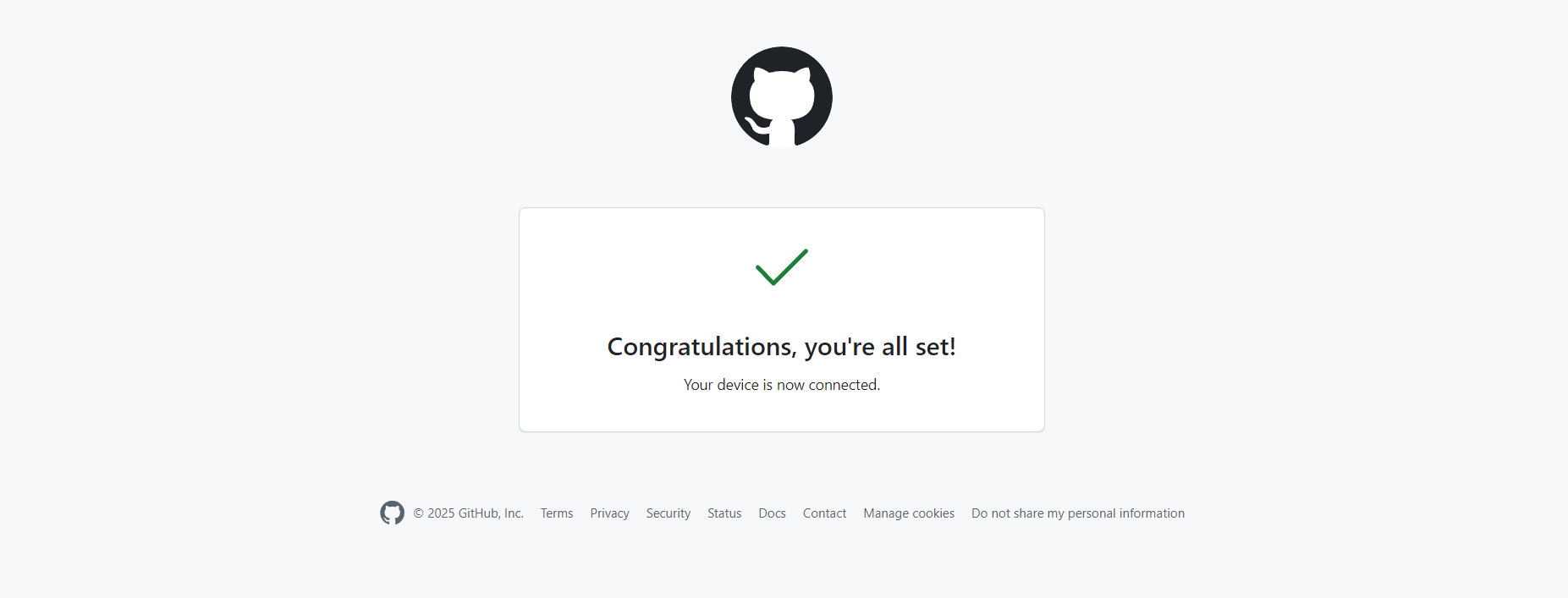


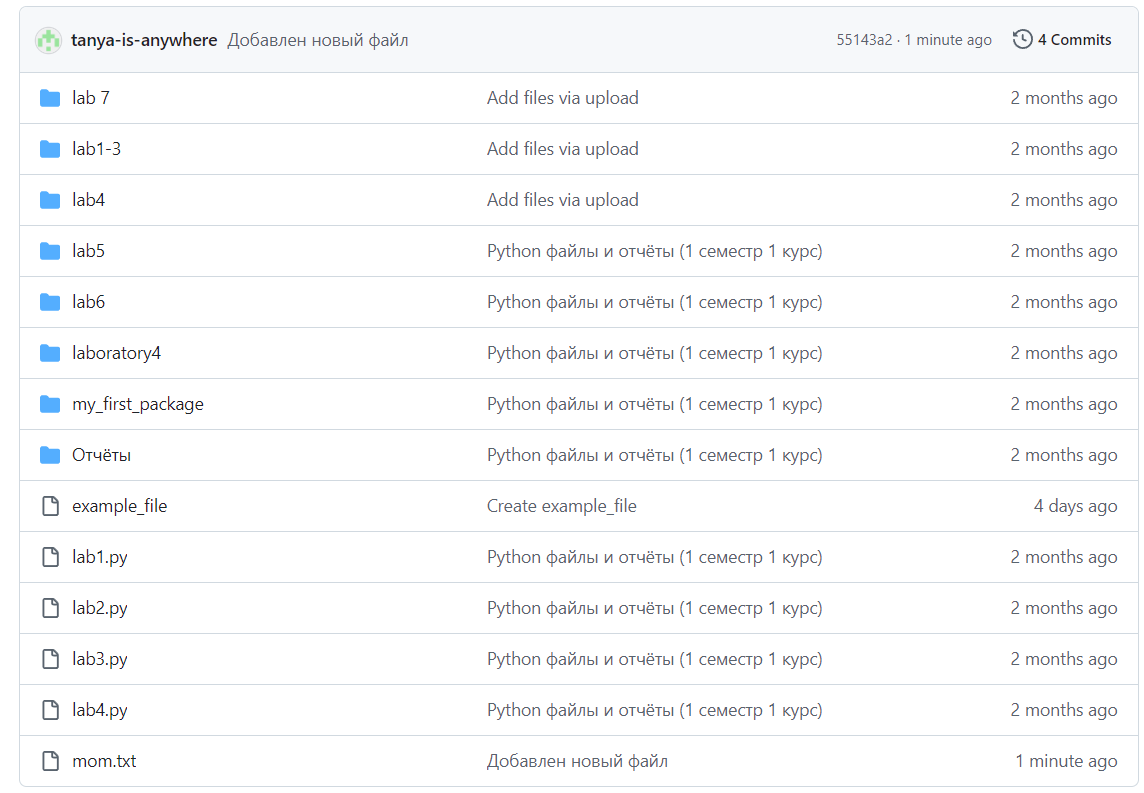
Добавим в индекс, закоммитим и запушим:



Сделаем привязку через браузер:





Проверим изменения в удалённом репозитории:

Видно, что появился новый файл.

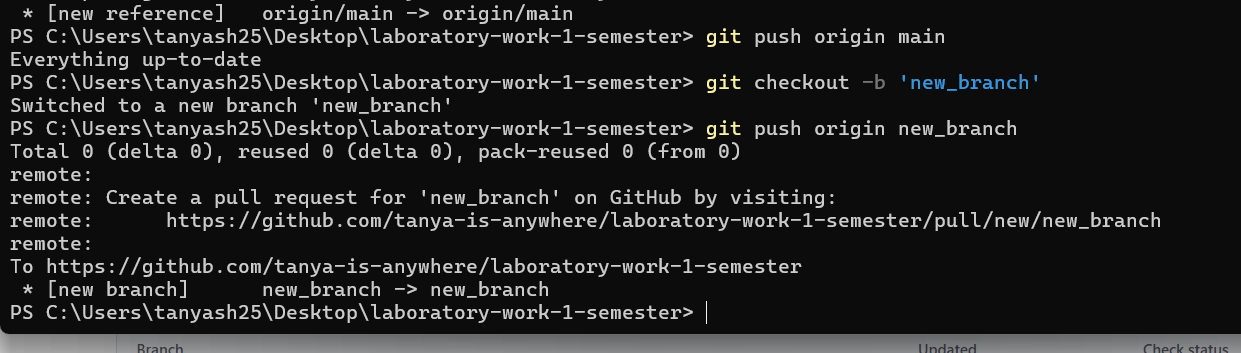
***Задание 4 Работа с ветками на удалённом репозитории***

Инструкция:

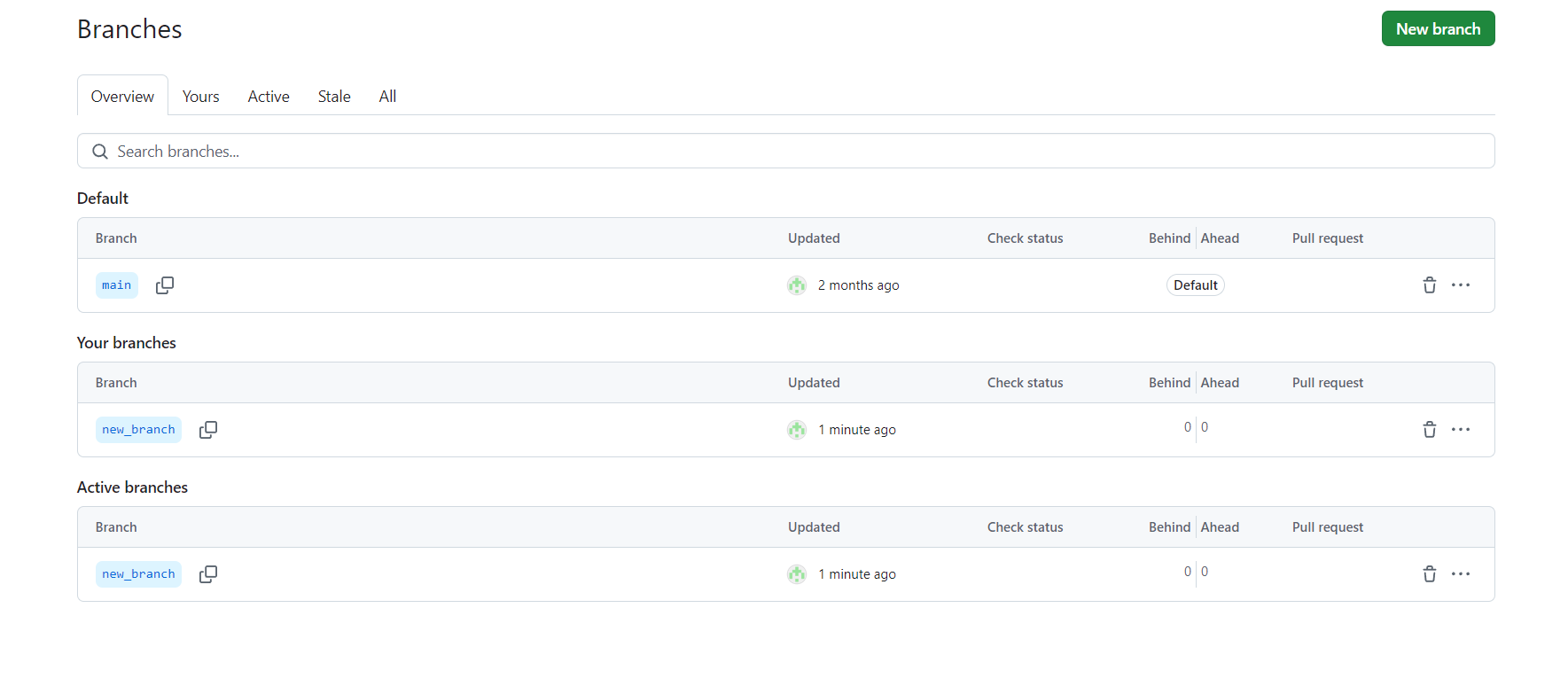
1. Создать новую ветку локально с помощью команды git checkout -b <branch\_name>.
2. Запушить новую ветку на удалённый репозиторий с помощью команды git push origin <branch\_name>.
3. Удалить ветку из удалённого репозитория с помощью команды git push origin --delete <branch\_name>.

Выполнение:

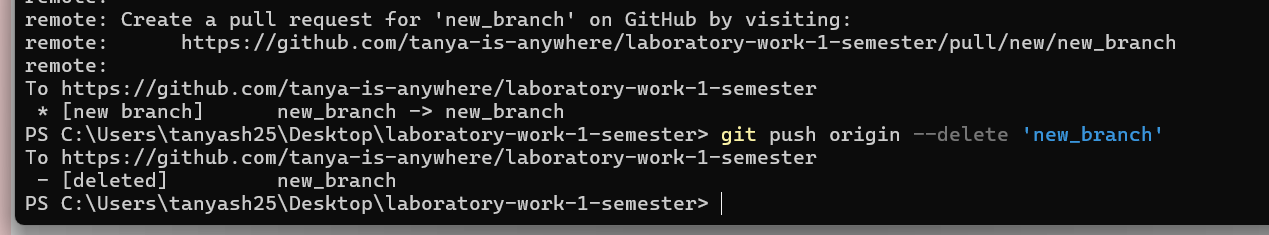
Добавление новой ветки:



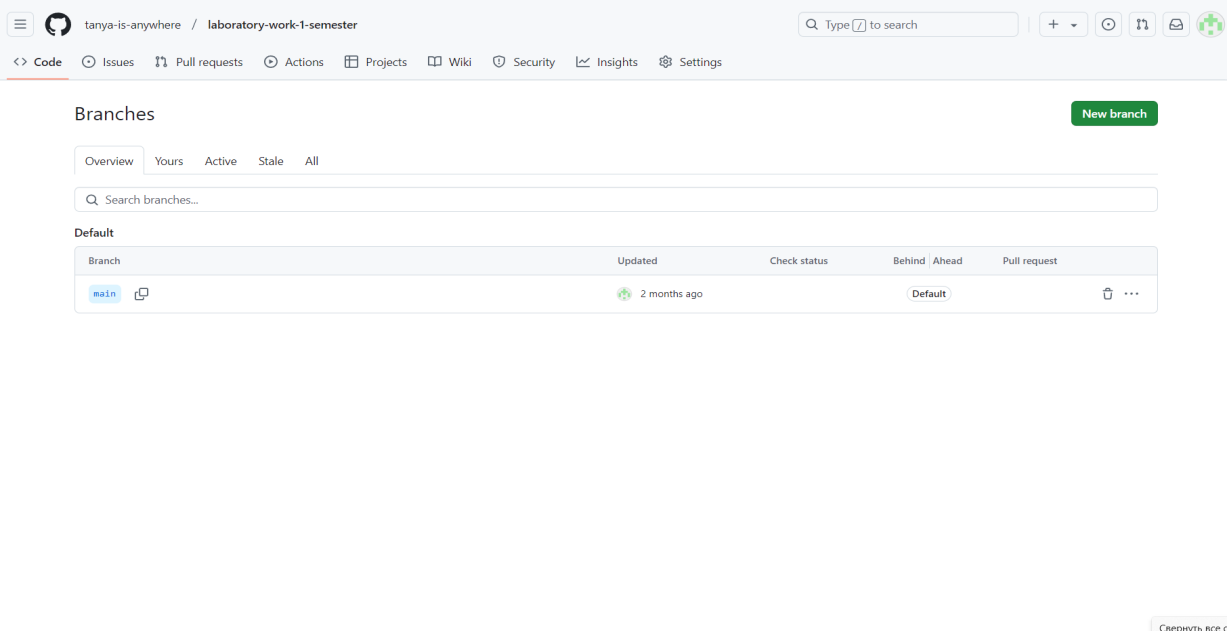
Новая ветка появилась:



Удаление ветки:



Теперь ветка исчезла:

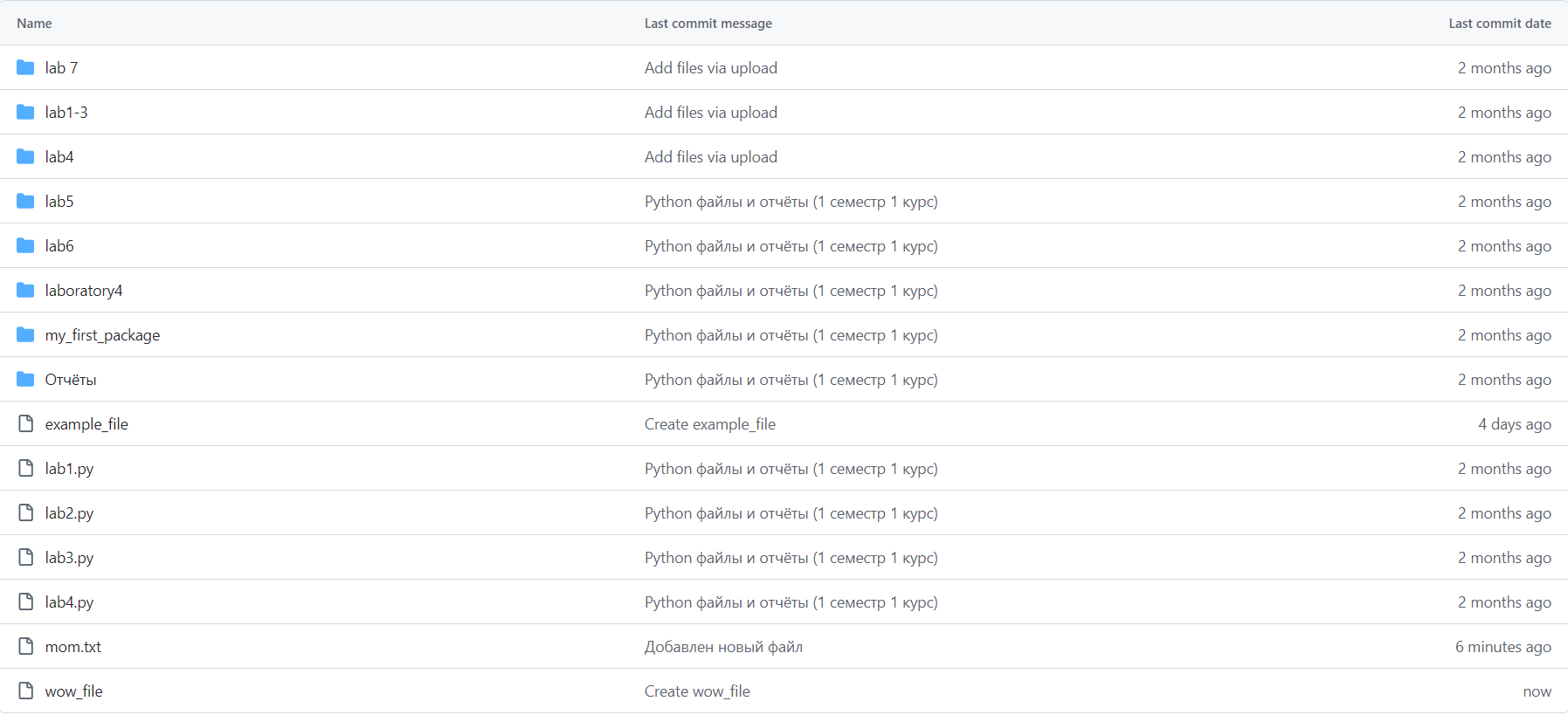


***Задание 5 Получение изменений из удалённого репозитория***

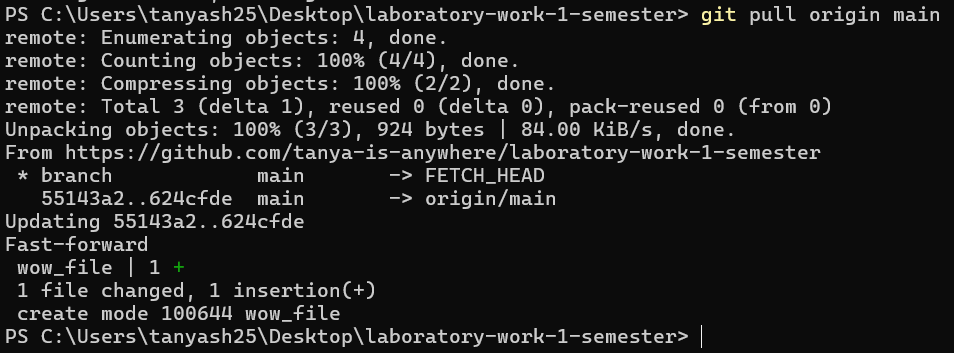
Инструкция:

1. Выполнить команду git pull origin <branch\_name> для получения изменений из удалённого репозитория на локальный.
2. Убедиться, что изменения успешно применены к локальному репозиторию.

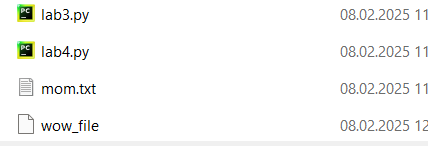
Выполнение:

Добавим изменения в удалённый репозиторий с помощью браузера (создадим новый файл):

Теперь получим изменения из удалённого репозитория с помощью указанной команды:



Убедимся, что файл был добавлен в локальный репозиторий:



**Вывод**: В ходе выполнения данной работы были изучены основы взаимодействия локального и удалённого репозитория git. Алгоритм внесения изменений из локального в глобальный: . Клонирование удалённого репозитория на локальный компьютер и получение изменений из удалённого репозитория проще и выполнима с помощью двух разных команд, без необходимости выполнять несколько действий. Взаимодействия с ветками также просты и требуют только единичных действий.

## Ответы на вопросы:

1. Что такое система контроля версий и для чего она используется?

**Система контроля версий (Version Control System, VCS)**, также известная как система управления версиями или система отслеживания версий, — это инструмент, который позволяет отслеживать и управлять изменениями в файлах и каталогах проекта во времени. VCS записывает историю изменений, позволяя вернуться к любой предыдущей версии проекта, сравнивать различные версии, объединять изменения, внесённые разными людьми, и отслеживать, кто и когда вносил изменения.

**Для чего используется:**

Отслеживание истории изменений: VCS позволяет видеть, как проект развивался со временем, кто вносил изменения и когда. Это полезно для понимания причин изменений и выявления ошибок.

Восстановление предыдущих версий: Если в проект была внесена ошибка или необходимо вернуться к более старой версии, VCS позволяет легко восстановить любую предыдущую версию проекта.

Совместная работа: VCS позволяет нескольким людям работать над одним и тем же проектом одновременно, без риска конфликтов и потери данных. VCS управляет слиянием изменений, внесённых разными разработчиками.

Ветвление и экспериментирование: VCS позволяет создавать ветки разработки, в которых можно экспериментировать с новыми функциями или исправлениями ошибок, не затрагивая основную кодовую базу.

Резервное копирование и восстановление: VCS обеспечивает резервное копирование файлов проекта, что защищает от потери данных в случае аппаратных сбоев или других проблем.

Аудит и соответствие нормативным требованиям: VCS предоставляет аудит изменений, показывая кто, что и когда изменил. Это важно для проектов, где требуется соответствие нормативным требованиям.

1. Какие основные преимущества использования Git перед другими системами контроля версий?

**Git** — это распределенная система контроля версий (Distributed Version Control System, DVCS), что даёт ей ряд преимуществ перед централизованными VCS, такими как SVN или CVS:

Распределенная архитектура: Каждый разработчик имеет полную копию репозитория, включая всю историю изменений. Это позволяет работать в автономном режиме, без подключения к центральному серверу. Изменения можно коммитить локально и отправлять на удалённый репозиторий позже.

Скорость: Большинство операций в Git выполняются локально, что делает их очень быстрыми.

Ветвление и слияние: Git имеет мощные и гибкие возможности для ветвления и слияния. Создание, переключение и слияние веток происходят очень быстро и просто.

Поддержка нелинейной разработки: Git хорошо поддерживает нелинейные процессы разработки, позволяя создавать сложные ветки и объединять их разными способами.

Целостность данных: Git использует криптографическую хеш-функцию SHA-1 для идентификации каждого объекта в репозитории, что гарантирует целостность данных и предотвращает случайную порчу.

Open Source: Git — это проект с открытым исходным кодом, что означает, что он бесплатен для использования и может быть изменён и распространяем.

Широкое распространение: Git является самой популярной VCS в мире, это означает, что существует большое сообщество пользователей и разработчиков, множество инструментов и ресурсов для обучения и поддержки.

1. Что такое команда "git clone" и как она используется?

Команда git clone используется для создания локальной копии удалённого репозитория Git. Она скачивает все файлы, историю коммитов и ветки из удалённого репозитория на ваш компьютер.

git clone <https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git>

Можно выбирать разные расширения для репозитория, например: HTTPS или SSH.

1. Какая команда используется для добавления файлов в индекс в Git?

Команда git add используется для добавления файлов в индекс (staging area) в Git.

git add <имя\_файла1> <имя\_файла2> …

1. Как создать новую ветку в Git и что означает "ветвление" в контексте системы контроля версий?

**Создание новой ветки:**

git branch <имя\_новой\_ветки>

Эта команда создаёт новую ветку с указанным именем, но не переключает вас на неё.

Для создания и переключения на новую ветку в одной команде используется:

git checkout -b <имя\_новой\_ветки>

**Что означает “ветвление”:**

Ветвление в Git — это создание отдельной линии разработки, которая отходит от основной ветки (обычно main или master). Ветви позволяют разрабатывать новые функции, исправлять ошибки или проводить эксперименты, не затрагивая основную кодовую базу.

Ветви создаются как указатели на определённый коммит в истории репозитория. Когда вы создаёте ветку, вы создаёте новый указатель, который указывает на тот же коммит, что и текущая ветка. После этого вы можете вносить изменения в новой ветке, не затрагивая основную ветку.

1. Что означает команда "git push" и как она используется для отправки изменений на удалённый репозиторий?

**Команда git push используется** для отправки локальных коммитов и веток на удалённый репозиторий. Она позволяет синхронизировать локальный репозиторий с удалённым репозиторием, делая ваши изменения доступными для других участников проекта.

**Как используется:**

git push <имя\_удалённого\_репозитория> <имя\_локальной\_ветки>:<имя\_удалённой\_ветки>

<имя\_удалённого\_репозитория>: Имя удалённого репозитория (обычно origin).

<имя\_локальной\_ветки>: Имя локальной ветки, которую вы хотите отправить.

<имя\_удаленной\_ветки>: Имя удалённой ветки, в которую вы хотите отправить изменения. Если имя удалённой ветки не указано, Git будет использовать имя локальной ветки.

Если локальная ветка настроена для отслеживания удаленной ветки, вы можете использовать git push без указания имени удаленного репозитория и имени ветки. Git автоматически определит, куда нужно отправить изменения.

Отправка новой ветки с установкой отслеживания:

git push -u <имя\_удаленного\_репозитория> <имя\_локальной\_ветки>

Или

git push --set-upstream <имя\_удалённого\_репозитория> <имя\_локальной\_ветки>

Эта команда отправит локальную ветку на удалённый репозиторий и установит отслеживание между локальной и удалённой веткой.

Пример:

git push origin main

1. Как можно получить изменения из удалённого репозитория на локальный с помощью Git?

Существует несколько способов получить изменения из удаленного репозитория в Git:

**git fetch**: Загружает коммиты, ветки и теги из удаленного репозитория в ваш локальный репозиторий, но не объединяет их с вашими локальными ветками.

git fetch <имя\_удаленного\_репозитория>

После выполнения git fetch вы можете просмотреть изменения с помощью git log origin/<имя\_ветки> и решить, как их интегрировать (с помощью git merge или git rebase).

**git pull**: Загружает коммиты и ветки из удаленного репозитория и автоматически объединяет их с вашей текущей локальной веткой.

git pull <имя\_удаленного\_репозитория> <имя\_ветки>

Например:

git pull origin main

git pull — это сокращение для выполнения git fetch и git merge в одной команде.

**Выбор способа получения изменений:**

git fetch рекомендуется, когда вы хотите просмотреть изменения перед их интеграцией.

git pull рекомендуется, когда вы уверены, что хотите интегрировать все изменения из удаленного репозитория.

1. Что такое merge (слияние) в Git и какое его предназначение?

**Слияние (merge) в Git** — это процесс объединения изменений из одной ветки в другую. Обычно используется для интеграции изменений из feature branch в основную ветку (например, main или master) после завершения работы над новой функцией или исправления ошибки.

**Предназначение**:

Интеграция изменений: Слияние позволяет объединить изменения, внесенные в разных ветках разработки, в единую кодовую базу.

Разрешение конфликтов: Во время слияния могут возникать конфликты, если в разных ветках были внесены изменения в одни и те же строки кода. Git предоставляет инструменты для разрешения этих конфликтов, позволяя выбрать, какие изменения нужно сохранить.

Сохранение истории: Слияние создаёт новый коммит, который связывает историю обеих веток, сохраняя информацию о том, когда и как были объединены изменения.

**Как выполнить слияние:**

Переключитесь на ветку, в которую вы хотите слить изменения:

git checkout <целевая\_ветка>

Выполните команду git merge:

git merge <ветка\_источник>

Например:

git checkout main

git merge feature/new-feature

9. Какое предназначение у команды "git pull"?

Команда git pull используется для получения изменений из удалённого репозитория и автоматического слияния их с текущей локальной веткой. Она объединяет функциональность команд git fetch и git merge в одной команде.

10. Какая команда используется для удаления ветки из удалённого репозитория в Git?

Для удаления ветки из удаленного репозитория в Git используется команда git push origin --delete <имя\_ветки>

Альтернативный синтаксис:

git push origin :<имя\_ветки>

Что аналогично git pull:

Три команды, которые аналогичны действиям команды git pull в Git:

**git fetch**. Связывается с удалённым репозиторием и забирает из него все изменения, которых пока нет, и сохраняет их локально.

**~~git clone~~**~~. Позволяет скачать удалённый код с удалённого URL-адреса репозитория на локальный компьютер.~~

**git merge**. Используется для слияния одной или нескольких веток в текущую.

**git branch**

Команда git pull работает как комбинация команд git fetch и git merge: сначала забирает изменения из указанного удалённого репозитория, а затем пытается слить их с текущей веткой.

Подробно о том, что делает git pull

Аналог git: gitverse - российский аналог git hub

Некоторые другие системы контроля версий (VCS) помимо Git:

**SVN** (Subversion). Подходит для небольших команд, которым нужна простая централизованная система.

**Mercurial**. Альтернатива Git с более простым интерфейсом, но менее популярная.

**Perforce** (Helix Core). Рекомендуется для крупных организаций, работающих с большими файлами (например, в геймдеве или медиа).

**ClearCase**. Выбор для сложных корпоративных проектов, но с высокой стоимостью и сложностью настройки.

**Fossil**. Подходит для небольших проектов, требующих интеграции баг-трекинга, wiki и веб-интерфейса прямо в систему контроля версий.

Что такое origin и удалённый репозиторий?

origin — это удобное сокращение для URL-адреса удалённого репозитория, с которого вы клонировали проект. Удалённый репозиторий — это репозиторий, который находится не на вашем компьютере, а в сети, и используется для резервного копирования, совместной работы и развертывания. Понимание этих терминов необходимо для эффективной работы с Git.

Задание

Windows PowerShell

(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! https://aka.ms/PSWindows

PS C:\Users\tanyash25> git clone https://github.com/mr-dxdy/calculator-kivy.git

Cloning into 'calculator-kivy'...

remote: Enumerating objects: 13, done.

remote: Counting objects: 100% (4/4), done.

remote: Compressing objects: 100% (4/4), done.

remote: Total 13 (delta 0), reused 3 (delta 0), pack-reused 9 (from 1)

Receiving objects: 100% (13/13), done.

PS C:\Users\tanyash25> git push https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git main

To https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git

! [rejected] main -> main (fetch first)

error: failed to push some refs to 'https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git'

hint: Updates were rejected because the remote contains work that you do not

hint: have locally. This is usually caused by another repository pushing to

hint: the same ref. If you want to integrate the remote changes, use

hint: 'git pull' before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

PS C:\Users\tanyash25> git push origin main

To https://gitlab.com/bvt2402/LabWork9.git

! [rejected] main -> main (non-fast-forward)

error: failed to push some refs to 'https://gitlab.com/bvt2402/LabWork9.git'

hint: Updates were rejected because the tip of your current branch is behind

hint: its remote counterpart. If you want to integrate the remote changes,

hint: use 'git pull' before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

PS C:\Users\tanyash25> git pull

There is no tracking information for the current branch.

Please specify which branch you want to merge with.

See git-pull(1) for details.

git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

git branch --set-upstream-to=origin/<branch> main

PS C:\Users\tanyash25> git pull origin

You asked to pull from the remote 'origin', but did not specify

a branch. Because this is not the default configured remote

for your current branch, you must specify a branch on the command line.

PS C:\Users\tanyash25> git pull origin main

From https://gitlab.com/bvt2402/LabWork9

\* branch main -> FETCH\_HEAD

fatal: refusing to merge unrelated histories

PS C:\Users\tanyash25> cd .\calculator-kivy\

PS C:\Users\tanyash25\calculator-kivy> git status

On branch master

Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

PS C:\Users\tanyash25\calculator-kivy> git add .

PS C:\Users\tanyash25\calculator-kivy> New-Item 'rar.txt' ItemType file

New-Item : Не удается найти позиционный параметр, принимающий аргумент "ItemType".

строка:1 знак:1

+ New-Item 'rar.txt' ItemType file

+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

+ CategoryInfo : InvalidArgument: (:) [New-Item], ParameterBindingException

+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.NewItemCommand

PS C:\Users\tanyash25\calculator-kivy> New-Item 'rar.txt' -ItemType file

Каталог: C:\Users\tanyash25\calculator-kivy

Mode LastWriteTime Length Name

---- ------------- ------ ----

-a---- 10.02.2025 12:23 0 rar.txt

PS C:\Users\tanyash25\calculator-kivy> git add .

PS C:\Users\tanyash25\calculator-kivy> ls

Каталог: C:\Users\tanyash25\calculator-kivy

Mode LastWriteTime Length Name

---- ------------- ------ ----

-a---- 10.02.2025 12:16 60 android.txt

-a---- 10.02.2025 12:16 2094 calculator.kv

-a---- 10.02.2025 12:16 576 icon.png

-a---- 10.02.2025 12:16 656 main.py

-a---- 10.02.2025 12:23 0 rar.txt

-a---- 10.02.2025 12:16 106 README.md

PS C:\Users\tanyash25\calculator-kivy> git status

On branch master

Your branch is up to date with 'origin/master'.

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: rar.txt

PS C:\Users\tanyash25\calculator-kivy> git commit -m 'Добавлен новый файл'

[master 4be8734] Добавлен новый файл

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 rar.txt

PS C:\Users\tanyash25\calculator-kivy> git push https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git main

error: src refspec main does not match any

error: failed to push some refs to 'https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git'

PS C:\Users\tanyash25\calculator-kivy> git push origin main

error: src refspec main does not match any

error: failed to push some refs to 'https://github.com/mr-dxdy/calculator-kivy.git'

PS C:\Users\tanyash25\calculator-kivy> git push origin https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git

fatal: invalid refspec 'https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git'

PS C:\Users\tanyash25\calculator-kivy> git push https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git

Enumerating objects: 16, done.

Counting objects: 100% (16/16), done.

Delta compression using up to 16 threads

Compressing objects: 100% (14/14), done.

Writing objects: 100% (16/16), 3.42 KiB | 1.71 MiB/s, done.

Total 16 (delta 2), reused 11 (delta 0), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (2/2), done.

remote:

remote: Create a pull request for 'master' on GitHub by visiting:

remote: https://github.com/tanya-is-anywhere/laboratory-work-1-semester/pull/new/master

remote:

To https://github.com/tanya-is-anywhere/laboratory-work-1-semester.git

\* [new branch] master -> master

PS C:\Users\tanyash25\calculator-kivy>