Звіт з лабораторної роботи №1

Студентки групи МІТ-31  
Баш Анастасії

Тема. Основи GIT. Робота з різними гілками програми в Git. Конфлікти в Git.

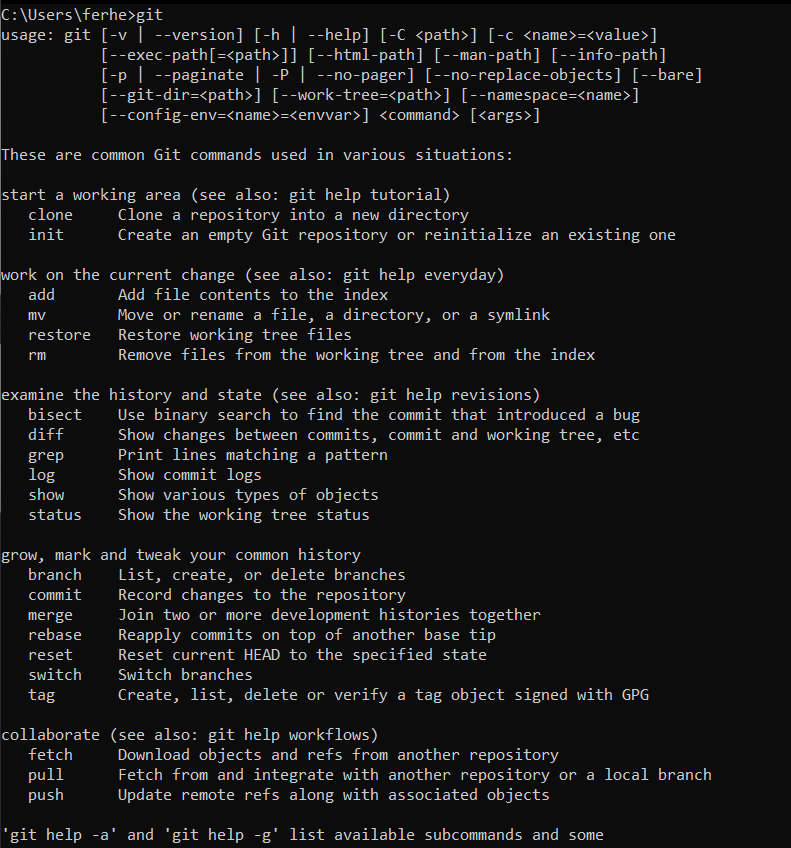
Мета: розглянути призначення та можливості систем контролю версій програмного забезпечення. Ознайомитися з Git та навчитися виконувати базові операції в Git. Навчитися працювати з різними гілками програми в Git, розглянути причини виникнення конфліктів зливання у Git та засоби боротьби з ними.

Хід роботи

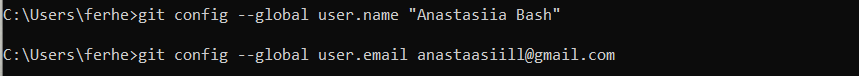
1.2.3 Початкове налаштування Git

Виконую команду git в командному рядку після інсталяції

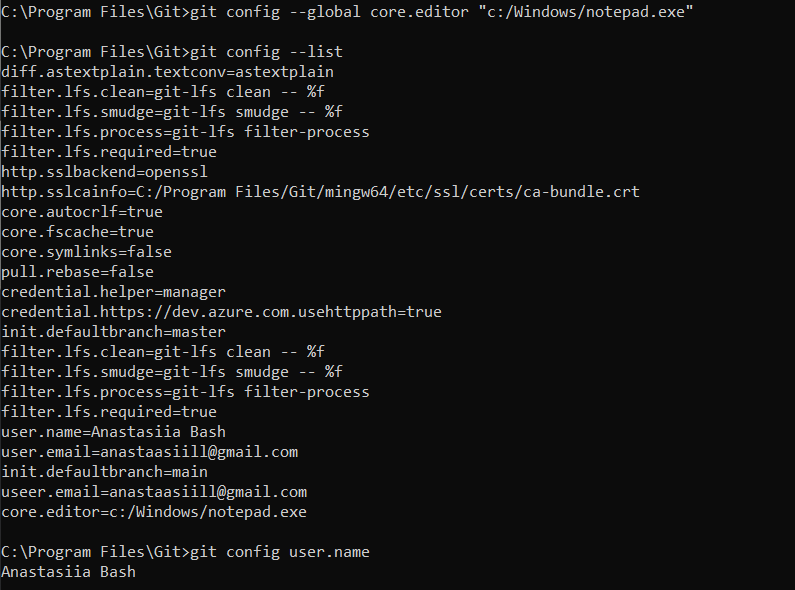
Виводиться список усіх форматів запису команд



Вводимо ім’я користувача та адресу електронної пошти за допомогою команди git config



Тепер налаштовуєм параметри редактора, який буде використовувати Git



1.2.4 Створення нового репозиторію, перегляд статусу файлів

Для того щоб стоворити репозиторій, переходимо в папку нашого проекту (як проект я використовувала свою лабораторну роботу з Інтелектуального аналізу даних). Створюємо там також файл readme.txt з текстом “Цей проект створено для вивчення та тестування роботи СКВ Git студентом Баш Анастасією Василівною”

Створюємо локальний репозиторій з допомогою команди git init.

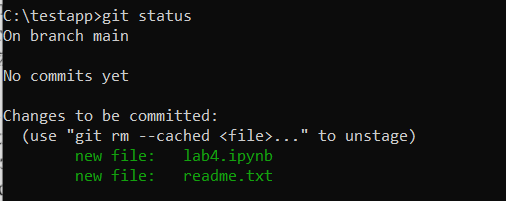


1.2.5 Додавання файлів до репозиторію

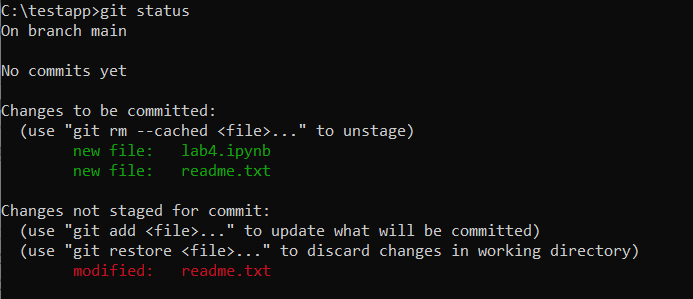
За допомогою команди git add додаємо усі файли нашого проекту.



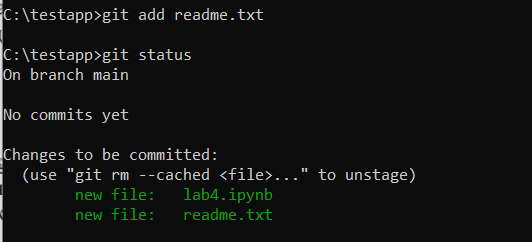
Перевіряємо статус наших файлів за допомогою команди git status.



Редагуємо файл readme.txt.



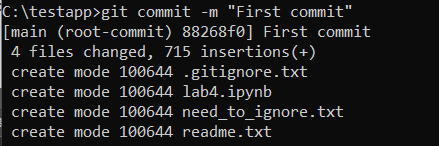
Потрібно знову виконати команду git add для файлу readme.txt, щоб відтрекати його зніми.



Створюємо файли .gitignore та вписуєм туди те, що потрібно ігнорувати файл nedd\_to\_ignore.txt.

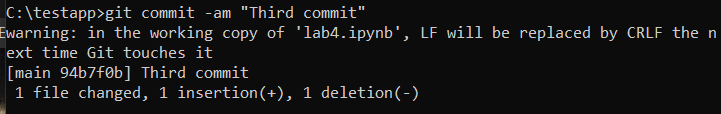
1.2.6 Комміти

Після того, як ми виконилали останній раз комманду git add . і всі наші файли та зміни в них відстежуються, ми робимо перший коміт за допомогою команди git commit –m “First commit”.

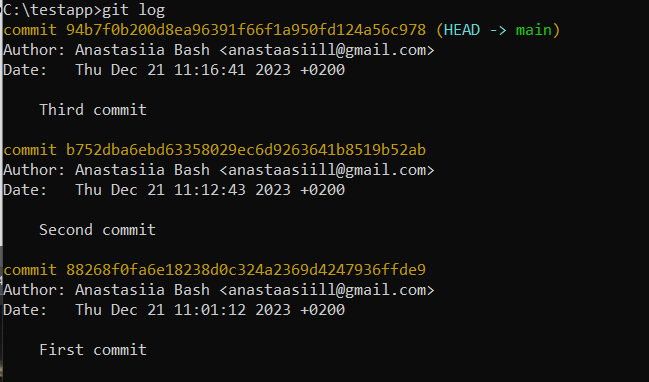


Змінюємо файл readme.txt та робимо другий коміт.

Тепер зміюнюємо файл lab4.ipynb та робимо коміт з параметром –а, що змушує git автоматично додати кожен файл, що вже контролюється системою до комміту.



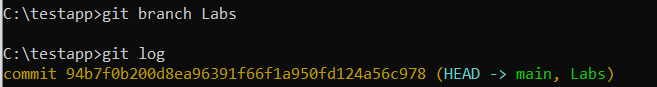
Перевіряємо наші комміти за допомогою команди git log.



Продовження  
Друга частина лабораторної роботи

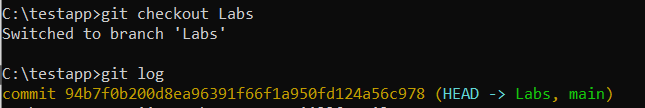
1.6.3 Створення нової гілки

Створюємо нову гілку з іменем Labs і перевіряємо, чи не змістився вказівник HEAD при створенні нової гілки.

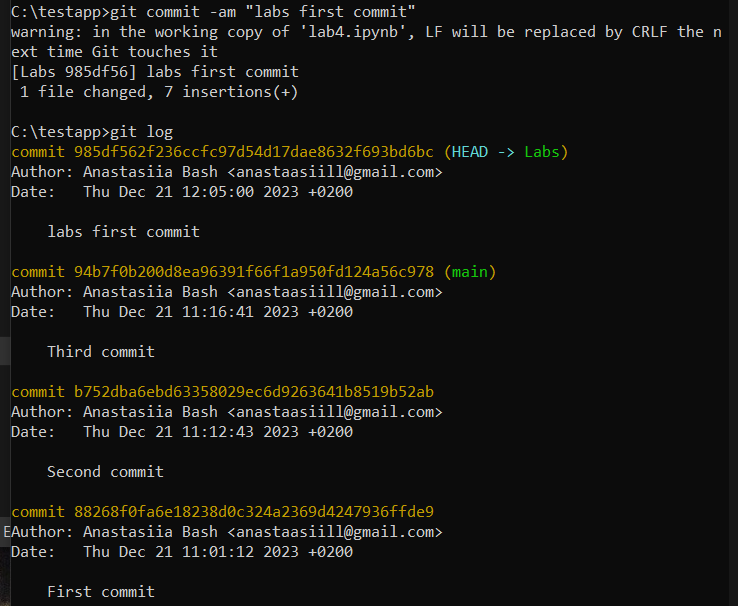


1.6.4 Перехід між гілками

Переходимо на гілку Labs за допомогою команди git checkout



Вносимо зміни в проект, робимо комміт та перевіряємо положення вказівника

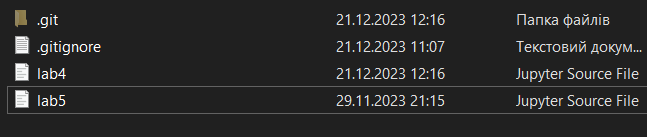


Я додала ще одну лабораторну роботу та видалила файли need\_to\_ignore.txt та readme.txt. Після цих змін зробила комміт для наглядної зміни гілок у папці

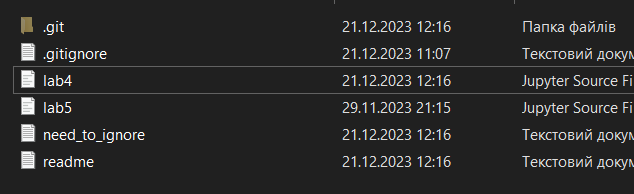
Тепер переключаюсь на гілку main.



Файли у папці до зміни гілок

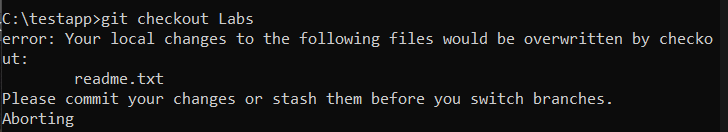


Після:

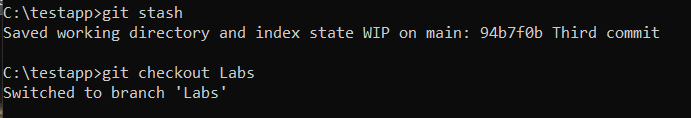


Зробила зміни у файлі readme.txt та тепер намагюсь перейти на гілку Labs.

В резцльтат в нас виникає конфлікт з тим, що зміни в нашому файлі не були закомічені і гіт пропонує або їх закомітити, або ж приберегти за допомогою команди stash

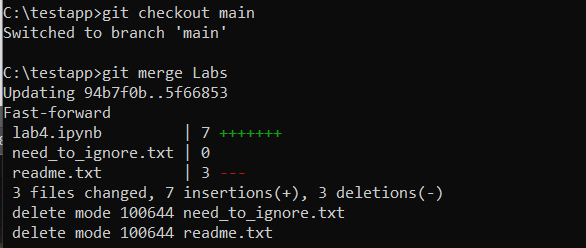


За допомогою команди stash зберігаємо зміни файлу без коміту та переходимо на гілку Labs.

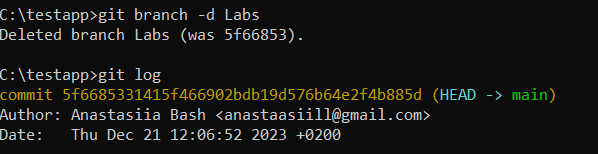


1.6.5 Злиття гілок

За допомогою команди git merge проводимо злиття гілки Labs в гілку main

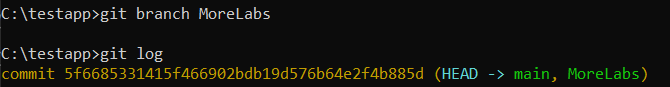


Тепер гілка Labs виглядає ідентично гілці main, тому ми видаляємо гілку Labs

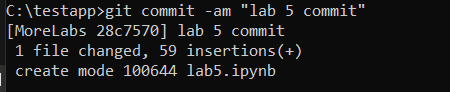


1.6.6 Конфлікти при злитті

Створюємо нову гілку MoreLabs

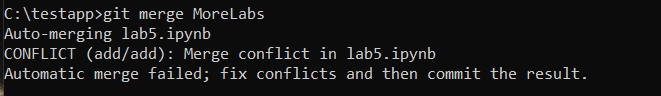


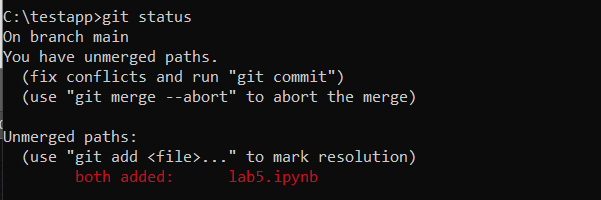
Дописала початок лабораторної роботи у lab5.ipynb та закомітила зміну на гілці MoreLabs.



Тепер переходжу на гілку main та змінію файл lab5.ipynb. Закомічую це.

Тепер історія змін цих двох гілок відрізняється. При злитті цих двох гілок трапиться так званий конфлікт так як зміни в одному й тому самому міці в двох різних гілках.





Бачимо, що конфлікт при злитті в нас відбувається у файлі lab5.ipynb. Тепер вручну аналізуємо на розвязуємо несуміності в кожному блоці