

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ
УНІВЕРСИТЕТ

Курсова Робота

По предмету: “Інженерія програмного забезпечення”

Виконав:

студент групи АМ-182

Борщов М. І.

Перевірила:

Шапоріна О.Л.

Одеса 2022

Перелік завдань до курсової роботи

Виконати наступні завдання:

1. Системы контроля версий. Введение в Git
2. Основи роботи в Git. Розгалуження
3. Основи командної роботи в Visual Studio при помощи Git.

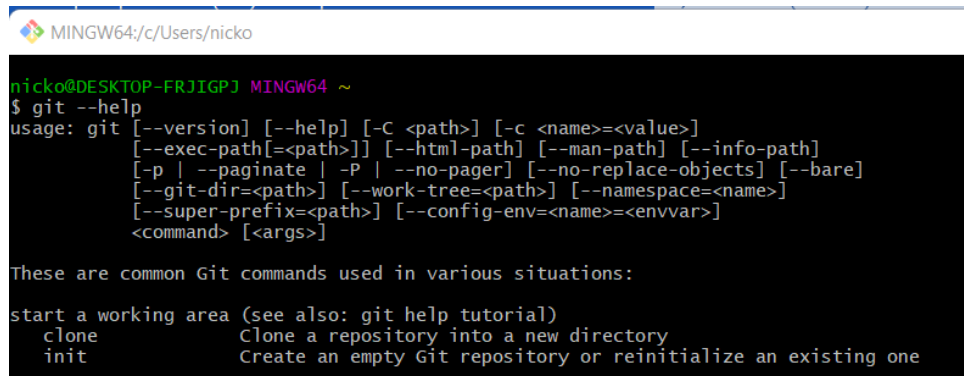
Хід роботи

Завдання 1

1. Вивчити теоретичний матеріал.
2. Встановити Git під Windows і виконати його первинне налаштування.

1. Первинне налаштування Git Bash та GitHub Desktop

Git Bash та GitHub Desktop в мене вже були встановлені.

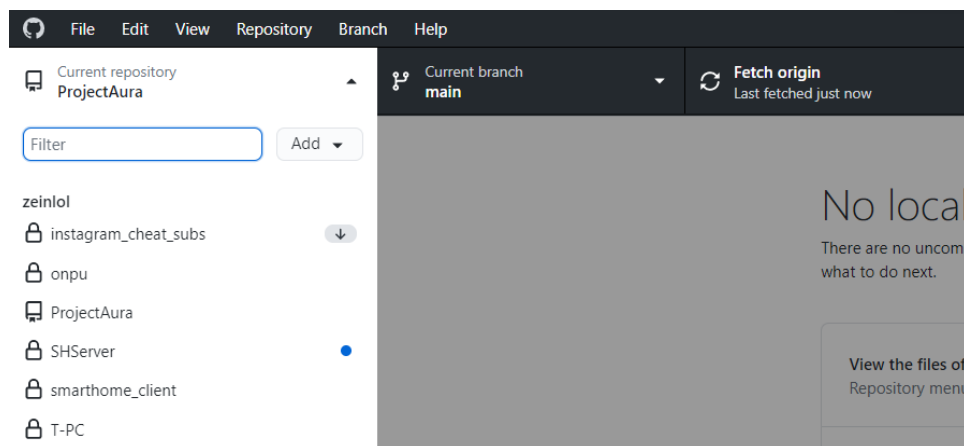


```
nicko@DESKTOP-FRJIGPJ MINGW64 ~
$ git --help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone                Clone a repository into a new directory
  init                 Create an empty Git repository or reinitialize an existing one
```

Команда допомоги в оболонці Git



Лист репозиторіїв у GitHub Desktop

2. Первісна настройка Git (Вивід встановлених раніше параметрів)

```
nicko@DESKTOP-FRJIGPJ MINGW64 ~  
$ git config --list  
diff.astextplain.textconv=astextplain  
filter.lfs.clean=git-lfs clean -- %f  
filter.lfs.smudge=git-lfs smudge -- %f  
filter.lfs.process=git-lfs filter-process  
filter.lfs.required=true  
http.sslbackend=openssl  
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt  
core.autocrlf=true  
core.fscache=true  
core.symlinks=false  
core.editor="C:\\Program Files\\JetBrains\\PyCharm 2021.1\\bin\\pycharm64.exe"  
pull.rebase=false  
credential.helper=manager-core  
credential.https://dev.azure.com.usehttppath=true  
init.defaultbranch=master  
filter.lfs.clean=git-lfs clean -- %f  
filter.lfs.smudge=git-lfs smudge -- %f  
filter.lfs.process=git-lfs filter-process  
filter.lfs.required=true  
user.name=Николай Борщёв  
user.email=56702121+zeinlol@users.noreply.github.com  
nicko@DESKTOP-FRJIGPJ MINGW64 ~
```

Завдання 2

Используя GitBash и GitGUI, выполнить следующие задачи:

1. Создать пустой репозиторий. Добавить в него файл "MyFirstFile.txt". Зафиксировать изменения.
2. Создать ветку "develop" от ветки "master". Создать ветку "feature1" от ветки "develop". Создать ветку "feature2" от ветки "develop".
3. Переключиться на ветку "feature1". Добавить строку "Hello" в файл "MyFirstFile.txt". Зафиксировать изменения. Переключиться на ветку "feature2". Добавить строки "H", "e", "l", "l", "o" в файл "MyFirstFile.txt". Зафиксировать изменения.
4. Создать из ветки "master" ветку "hotfix" и переключиться на нее. Создать файл "HotFixFile" со строкой "HotFix". Зафиксировать изменения. Добавить в файл "HotFixFile.txt" новую строку "HotFix2". Зафиксировать изменения.
5. Сделать перемещение (rebase) ветки "master" на ветку "hotfix".
6. Сделать перемещение (rebase) ветки "develop" на ветку "master". Сделать перемещение (rebase) ветки "feature1" на ветку "develop". Сделать перемещение (rebase) ветки "feature2" на ветку "develop".
7. Переключиться на ветку "feature1". Добавить файл Feature1.txt в репозиторий. Зафиксировать изменения. Переключиться на ветку "feature2". Добавить файл Feature2.txt в репозиторий. Зафиксировать изменения.
8. Разработка функции "feature1" окончена. Необходимо сделать перемещение (rebase) всех изменений из ветки "develop" на ветку "feature1".
9. Сделать перемещение (rebase) ветки "feature2" на ветку "develop". Решить возникший конфликт объединения.
10. Удалить ветку "hotfix".
11. Создать ветку "release01" от ветки "develop". В ней будет готовиться релиз. Переключиться на ветку "release01". Создать файл "ReleaseFix.txt" и поместить его в репозиторий. Зафиксировать изменения.
12. Предположим, что из ветки "release01" был произведен релиз. Сделать перемещение (rebase) ветки "master" на ветку "release01". Сделать перемещение (rebase) ветки "develop" на ветку "master". Сделать перемещение (rebase) ветки "feature2" на ветку "develop". Удалить ветку "feature1".
13. Предположим, что последнее изменение в ветке "feature2" не нужны. Необходимо откатить их назад сделав revert commit. Создать ветку "feature3" из ветки "develop". Добавить в репозиторий 3 файла: "Feature3.txt", "Feature3.1.txt" и "Feature3.2.txt". Зафиксировать изменения.
14. Создать ветку "feature4" от ветки "develop" и переключиться на нее. Добавить в репозиторий папку "f4" с четырьмя файлами внутри: "Feature1.txt", "HotFix.txt", "GettingStarted.txt", "ReleaseFix.txt". Зафиксировать изменения.
15. Сделать перемещение (rebase) ветки "develop" на ветку "feature3". Удалить ветку "feature3".
16. Создать ветку "hotfix2" от ветки "master" и переключиться на нее. Добавить в репозиторий папку "hotfix" с файлом "HotFix2.txt" внутри. Зафиксировать изменения. Переместить (rebase) ветку "develop" на ветку "feature2".
17. Сделать перемещение (rebase) ветки "master" на ветку "hotfix2". Сделать перемещение (rebase) ветки "develop" на ветку "master". Сделать перемещение (rebase) ветки "feature2" на ветку "master". Сделать перемещение (rebase) ветки "feature4" на ветку "develop".
18. Сделать перемещение (rebase) ветки "develop" на ветку "feature4". Удалить ветку "feature2".
19. Удалить ветку "feature4". Сделать перемещение (rebase) ветки "master" на ветку "release-candidate2". Сделать перемещение (rebase) ветки "develop" на ветку "master". Переименовать ветку "release-candidate2" в "release2".

1. Використовуючи GitBash і GitGUI, виконати наступні завдання:
2. Створити порожній репозиторій. Додати в нього файл "MyFirstFile.txt". Зафіксувати зміни.

```
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo
$ git init
Initialized empty Git repository in C:/Users/nicko/Documents/GitHub/test_repo/.git/

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (master)
$ echo '' > MyFirstFile.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (master)
$ git add MyFirstFile.txt
warning: LF will be replaced by CRLF in MyFirstFile.txt.
The file will have its original line endings in your working directory

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (master)
$ git commit -m 'init Repo'
[master (root-commit) 455544c] init Repo
1 file changed, 1 insertion(+)
create mode 100644 MyFirstFile.txt
```

3. Створити гілку "develop" від гілки "master". Створити гілку "feature1" від гілки "develop". Створити гілку "feature2" від гілки "develop".

```
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (master)
$ git checkout -b develop
Switched to a new branch 'develop'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop)
$ git branch feature1

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop)
$ git branch feature2

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop)
$ git branch -a
* develop
  feature1
  feature2
  master
```

4. Переключитись на гілку "feature1". Додати рядок "Hello" в файл "MyFirstFile.txt". Зафіксувати зміни. Переключитись на гілку "feature2". Додати рядки "H", "e", "l", "l", "o" в файл "MyFirstFile.txt". Зафіксувати зміни.

```
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop)
$ git checkout feature1
Switched to branch 'feature1'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ nano MyFirstFile.txt
```

```
MINGW64:/c:/Users/nicko/Documents/GitHub/test_repo
GNU nano 5.6.1 MyFirstFile.txt
Hello
```

```

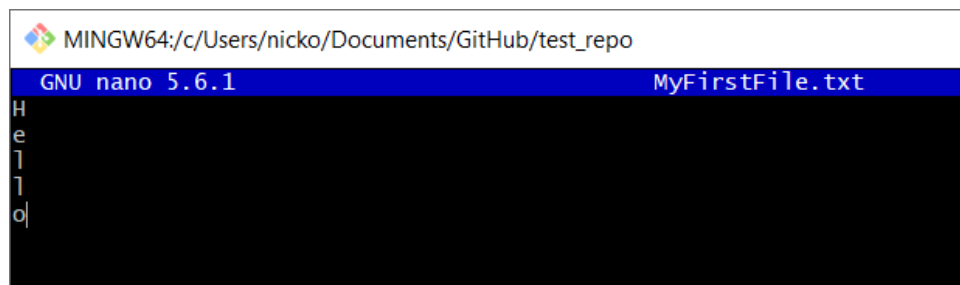
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ git add --all
warning: LF will be replaced by CRLF in MyFirstFile.txt.
The file will have its original line endings in your working directory

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ git commit -m 'update MyFile in feature1 branch'
[feature1 6f717bd] update MyFile in feature1 branch
1 file changed, 1 insertion(+), 1 deletion(-)

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ git checkout feature2
Switched to branch 'feature2'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature2)
$ nano MyFirstFile.txt

```



```

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature2)
$ git add --all

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature2)
$ git commit -m 'update MyFile in feature2 branch'
[feature2 2cee447] update MyFile in feature2 branch
1 file changed, 5 insertions(+), 1 deletion(-)

```

5. Створити з гілки "master" гілку "hotfix" і переключитись на неї. Створити файл "HotFixFile" з рядком "HotFix". Зафіксувати зміни. Додати в файл "HotFixFile.txt" "новий рядок" "HotFix2 ". Зафіксувати зміни.

```

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature2)
$ git checkout master
Switched to branch 'master'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (master)
$ git checkout -b hotfix
Switched to a new branch 'hotfix'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (hotfix)
$ echo 'HotFix' > HotFixFile.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (hotfix)
$ git add --all
warning: LF will be replaced by CRLF in HotFixFile.txt.
The file will have its original line endings in your working directory

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (hotfix)
$ git commit -a -m 'Cheate HotFixFile.txt in hotfix branch'
[hotfix db45910] Cheate HotFixFile.txt in hotfix branch
1 file changed, 1 insertion(+)
create mode 100644 HotFixFile.txt

```

6. Зробити переміщення (rebase) гілки "master" на гілку "hotfix".


```
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (hotfix)
$ git rebase master
Current branch hotfix is up to date.
```

7. Зробити переміщення (rebase) гілки "develop" на гілку "master". Зробити переміщення (rebase) гілки "feature1" на гілку "develop". Зробити переміщення (rebase) гілки "feature2" на гілку "develop".

```
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (hotfix)
$ git checkout master
Switched to branch 'master'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (master)
$ git rebase develop
Current branch master is up to date.

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (master)
$ git checkout develop
Switched to branch 'develop'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop)
$ git rebase feature1
Successfully rebased and updated refs/heads/develop.

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop)
$ git rebase feature2
error: could not apply 6f717bd... update MyFile in feature1 branch
Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply 6f717bd... update MyFile in feature1 branch
Auto-merging MyFirstFile.txt
CONFLICT (content): Merge conflict in MyFirstFile.txt
```

Під час переміщення гілок виник конфлікт файлів. Це сталося бо зміст файлу MyFirstFile.txt різний у гілках feature1 та feature2.

```
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop|REBASE 1/1)
$ git add MyFirstFile.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop|REBASE 1/1)
$ git rebase --continue
```

8. Переключитися на гілку "feature1". Додати файл Feature1.txt в репозиторій. Зафіксувати зміни. Переключитися на гілку "feature2". Додати файл Feature2.txt в репозиторій. Зафіксувати зміни.

```

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop)
$ git checkout feature1
Switched to branch 'feature1'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ touch Feature1.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ git add Feature1.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ git commit -a -m 'New file Reature1.txt in feature1 branch'
[feature1 41780e5] New file Reature1.txt in feature1 branch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Feature1.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ git checkout feature2
Switched to branch 'feature2'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature2)
$ touch Feature2.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature2)
$ git add Feature2.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature2)
$ git commit -a -m 'New file Feature2.txt in feature2 branch'
[feature2 d6802bc] New file Feature2.txt in feature2 branch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Feature2.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature2)
$

```

9. Розробка функції "feature1" закінчена. Необхідно зробити переміщення (rebase) всіх змін з гілки "develop" на гілку "feature1".

```

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature2)
$ git checkout feature1
Switched to branch 'feature1'

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ git rebase develop
error: could not apply 6f717bd... update MyFile in feature1 branc
Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --abort"
.
Could not apply 6f717bd... update MyFile in feature1 branc
Auto-merging MyFirstFile.txt
CONFLICT (content): Merge conflict in MyFirstFile.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1|REBASE 1/2)
$ nano MyFirstFile.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1|REBASE 1/2)
$ git add MyFirstFile.txt

nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1|REBASE 1/2)
$ git rebase --continue

```

- 10.Зробити переміщення (rebase) гілки "feature2" на гілку "develop".
Вирішити конфлікт, що виник у результаті об'єднання.

```
Successfully rebased and updated refs/heads/feature1.
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (feature1)
$ git checkout develop
Switched to branch 'develop'
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop)
$ git rebase feature2
Successfully rebased and updated refs/heads/develop.
nicko@DESKTOP-FRJIGPJ MINGW64 ~/Documents/GitHub/test_repo (develop)
$ ss|
```

Конфлікт переміщення був вирішений в попередніх завданнях

11.Видалити гілку "hotfix".



12. Створити гілку "release01" від гілки "develop". У ній буде готується реліз. Переключитися на гілку "release01". Створити файл "ReleaseFix.txt" і помістити його в репозиторій. Зафіксувати зміни.



13.Припустимо, що з гілки "release01" був проведений реліз. Зробити переміщення (rebase) гілки "master" на гілку "release01". Зробити переміщення (rebase) гілки "develop" на гілку "master". Зробити переміщення (rebase) гілки "feature2" на гілку "develop". Видалити гілку "feature1".

•

14.Припустимо, що останні зміни в гілці "feature2" не потрібні. Необхідно відкинути їх назад зробивши revert commit. Створити гілку "feature3" з гілки "develop". Додати в репозиторій 3 файлу: "Feature3.txt", "Feature3.1.txt" і "Feature3.2.txt". Зафіксувати зміни.



•



15. Створити гілку "feature4" від гілки "develop" і переключитися на неї.
Додати в репозиторій папку "f4" з чотирма файлами всередині:
"Feature1.txt", "HotFix.txt", "GettingStarted.txt", "ReleaseFix.txt".
Зафіксувати зміни.



16. Зробити переміщення (rebase) гілки "develop" на гілку "feature3".
Видалити гілку "feature3".



17. Створити гілку "hotfix2" від гілки "master" і переключитися на неї.
Додати в репозиторій папку "hotfix" з файлом "HotFix2.txt" всередині.
Зафіксувати зміни. Перемістити (rebase) гілку "develop" на гілку "feature2".



.

Вирішення merge конфлікту

18. Зробити переміщення (rebase) гілки "master" на гілку "hotfix2". Зробити переміщення (rebase) гілки "develop" на гілку "master". Зробити переміщення (rebase) гілки "feature2" на гілку "master". Зробити переміщення (rebase) гілки "feature4" на гілку "develop".



19. Зробити переміщення (rebase) гілки "develop" на гілку "feature4".
Видалити гілку "feature2".



20. Видалити гілку "feature4". Зробити переміщення (rebase) гілки "master" на гілку "release-candidate2". Зробити переміщення (rebase) гілки "develop" на гілку "master". Перейменувати гілку "release-candidate2" в "release2".



Далі на github.com зроблено репозиторій, на який було завантажено усі файли та історію змін.



Для завантаження та історії змін було використано IDE PyCharm



Результат

.

Історія змін



Розгалуження гілок не є ідеальним, бо не були закоммітчені рішення merge-конфліктів та частині гілок з конфліктами була видалена. У IDE PyCharm було виконано злиття усіх “робочих” гілок з гілкою `develop`, а потім вона була приєднана до гілки `master`.



Історія гілок на сайті. Гілки можуть бути локальними або завантаженими у мережу, для того щоб їх могли використовувати інші розробники

Завдання 3

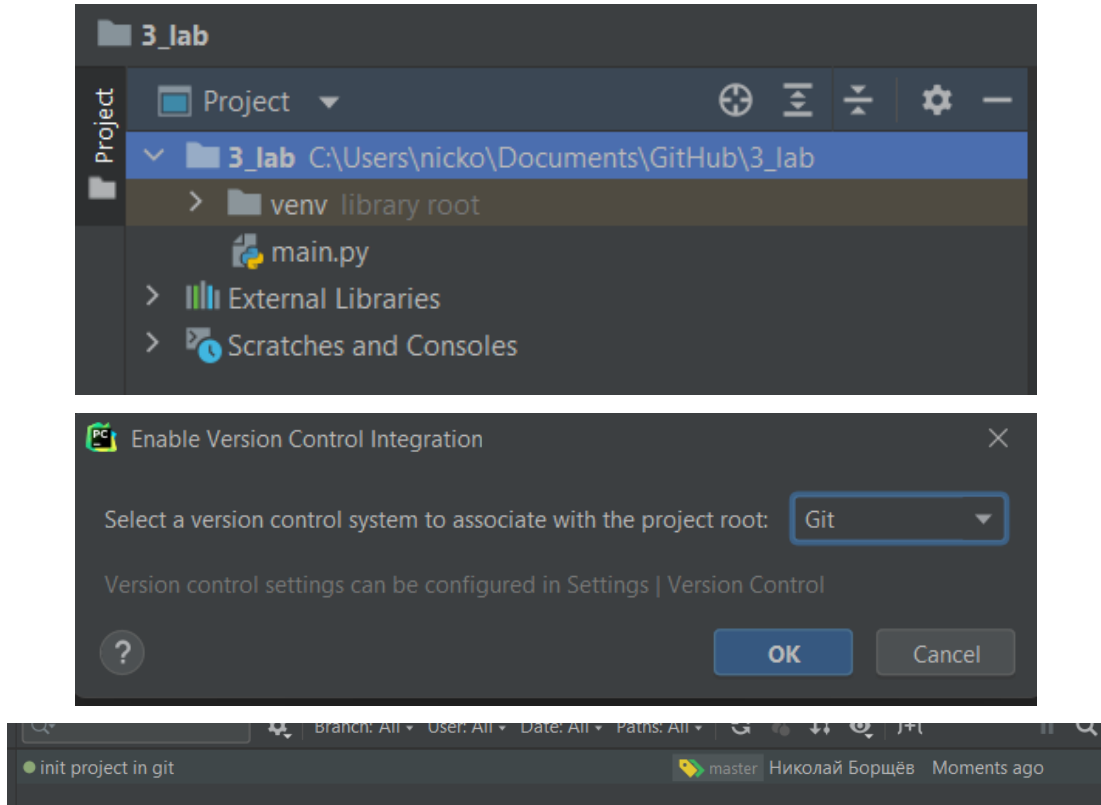
Используя Visual Studio 2015, выполнить следующие задачи:

1. Создать проект консольного приложения на C#, добавив его в систему контроля версий, в качестве которой выбрать Git . Зафиксировать изменения.
2. Создать ветку "develop" от ветки "master". Добавить в файле "Program.cs" в метод "Main" оператор *Console.WriteLine("Hello World!")*. Зафиксировать изменения.
3. Создать ветку "hotfix1" из "develop". Добавить в файле "Program.cs" в метод "Main" оператор *Console.ReadKey()*. Зафиксировать изменения.
4. Создать ветку "feature1" из "develop". Добавить в файле "Program.cs" в метод "Main" оператор *Console.WriteLine("Bye")*. Зафиксировать изменения.
5. Создать ветку "feature2" из "develop". Добавить в файле "Program.cs" в метод "Main" оператор *Console.WriteLine("GoodBye")*. Зафиксировать изменения.
6. Переключиться на ветку "master". Сделать перемещение (merge) ветки "feature1" на ветку "master".
7. Переключиться на ветку "develop". Сделать перемещение (merge) ветки "hotfix1" на ветку "develop". Решить возникший конфликт. В результате слияния в файле "Program.cs" на ветке "develop" должен быть добавлен оператор из файла "Program.cs" "hotfix1".
8. Сделать перемещение (merge) ветки "feature2" на ветку "develop". Решить возникший конфликт. В результате слияния в файле "Program.cs" на ветке "develop" должен быть добавлен оператор из файла "Program.cs" "feature2".
9. Создать ветку "release" из "develop". Сделать перемещение (merge) ветки "release" на ветку "master".
10. Удалить ветки "feature2", "hotfix1" и "release".

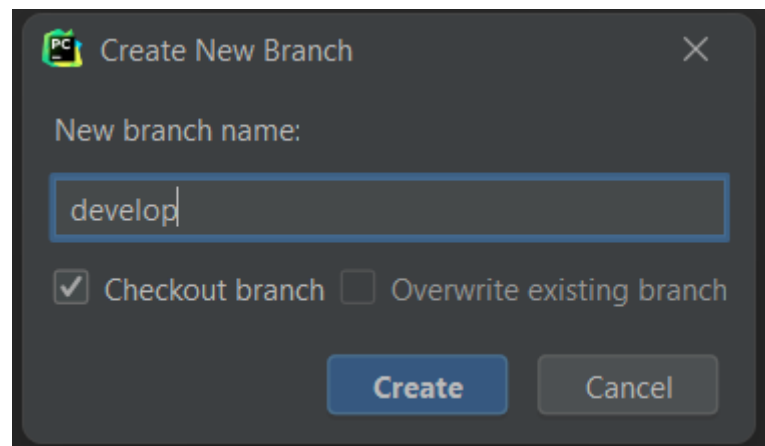
Хід роботи

Для виконання цієї роботи замість Visual Studio буде використана IDE PyCharm

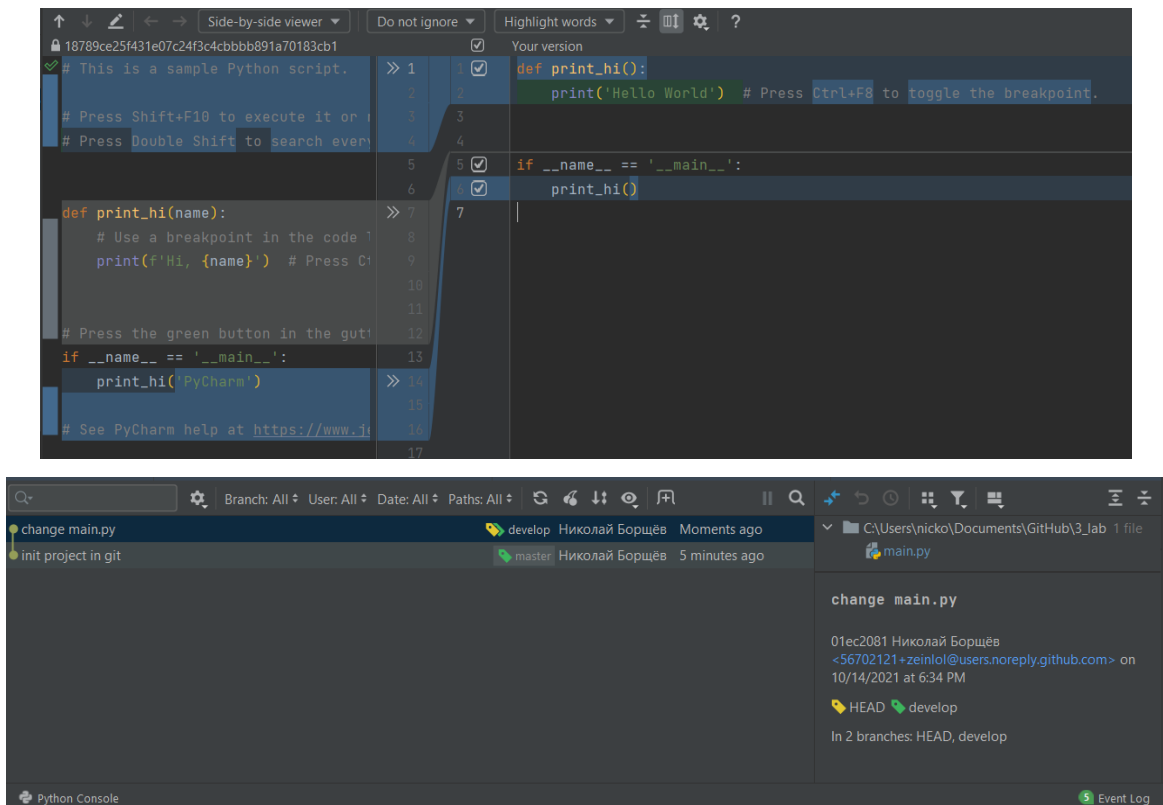
1) Створити проект та підключити Git:



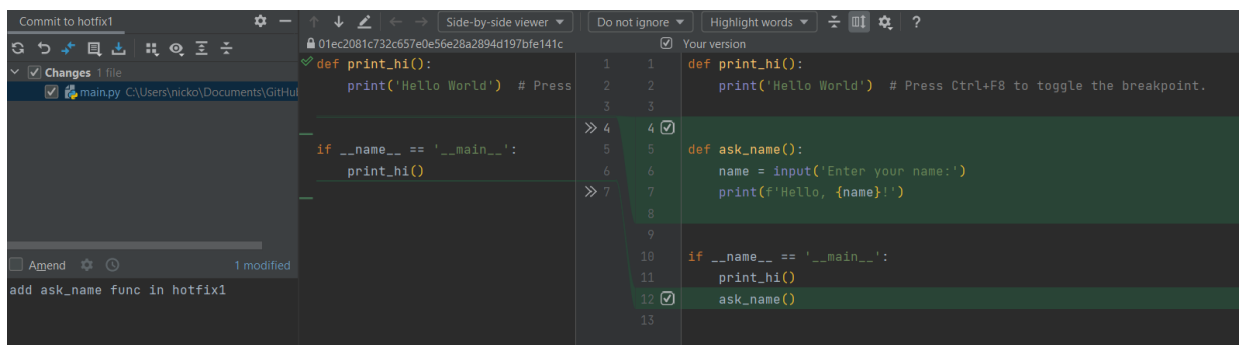
2) Створити нову гілку та додати файли та змінити їх



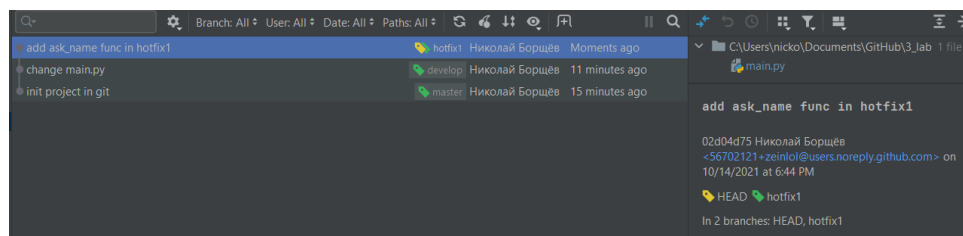
Змінення файлу main.py:



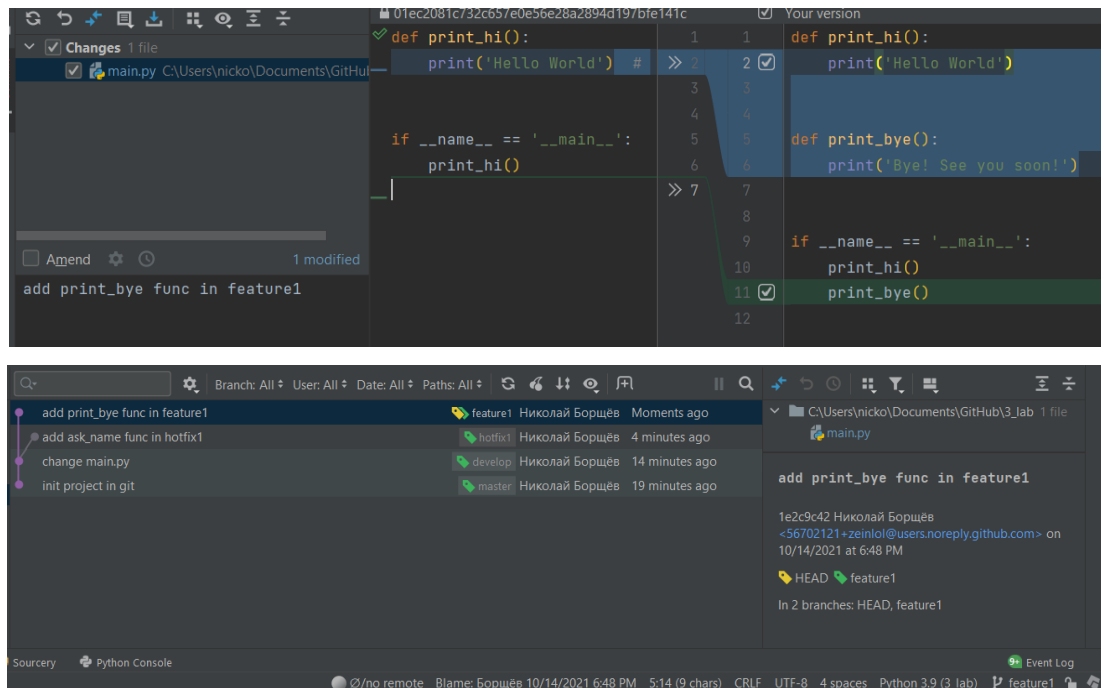
3) Створення нової гілки та внесення змін у файлі:



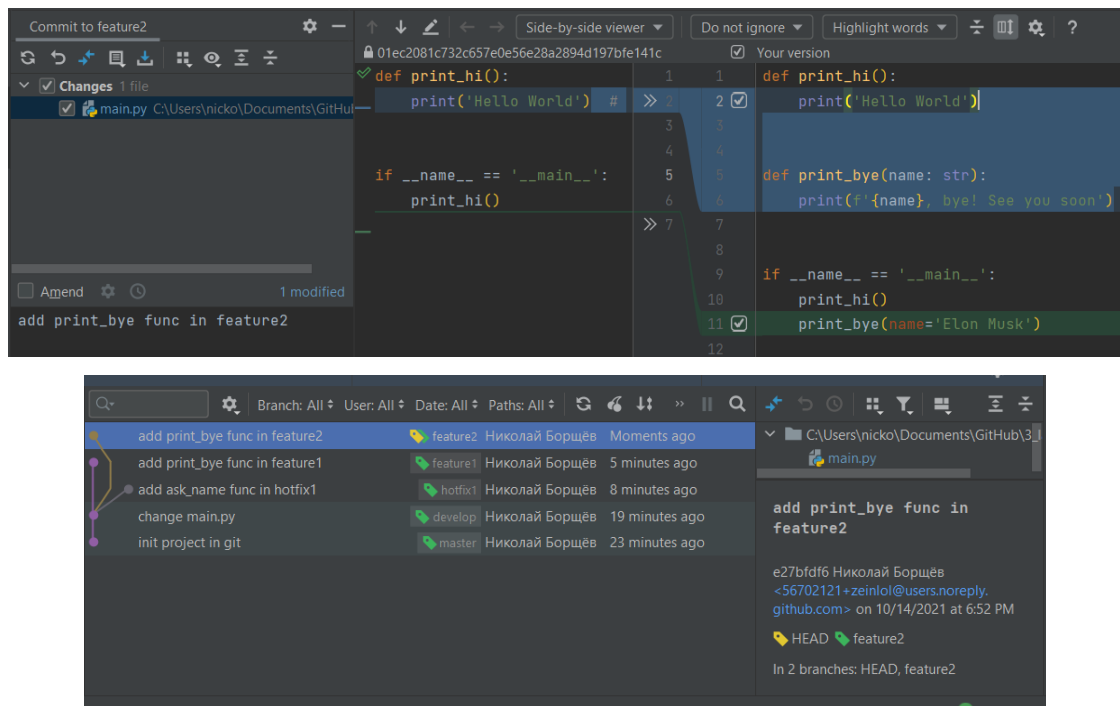
У проєкті я використовую мову Python, тож у цій роботі я використовував аналогічні C# функції.



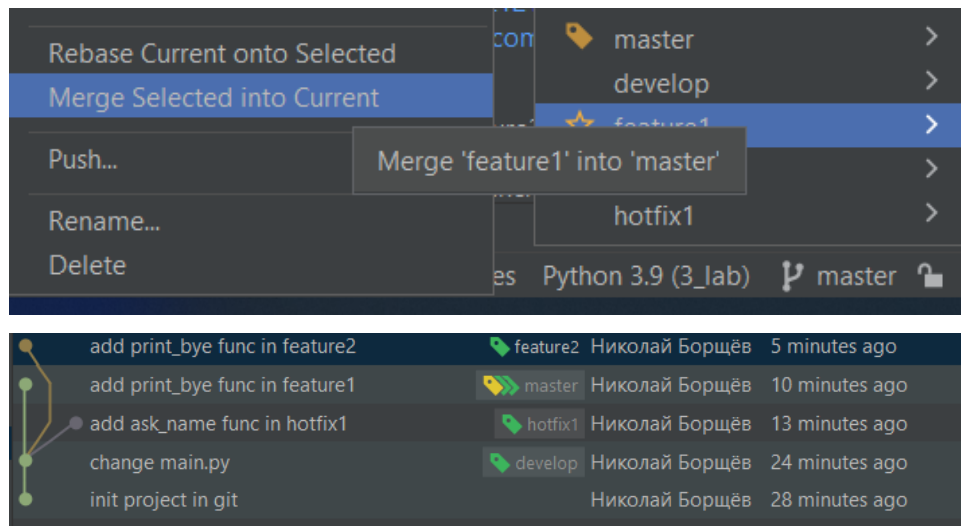
4) Створити гілку feature1 та внести зміни:



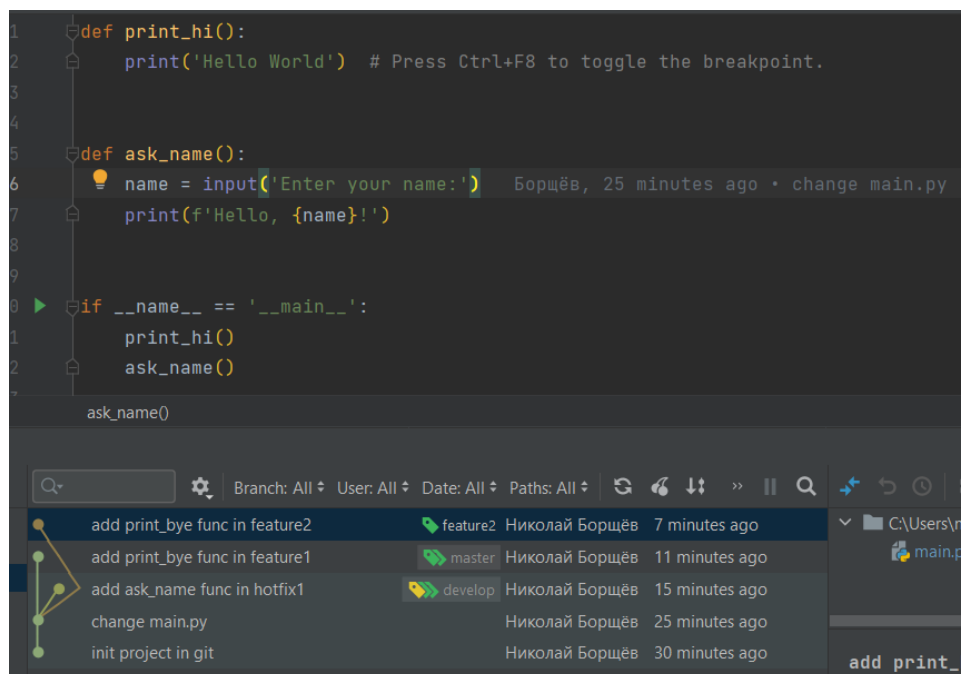
5) Створити гілку feature2 та внести подібні зміни:



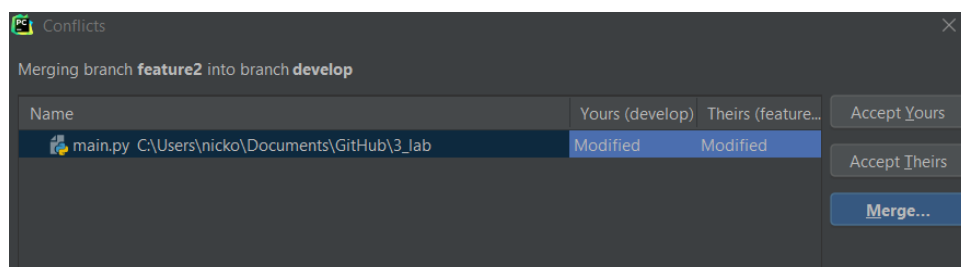
6) Перенести зміни з feature1 у master



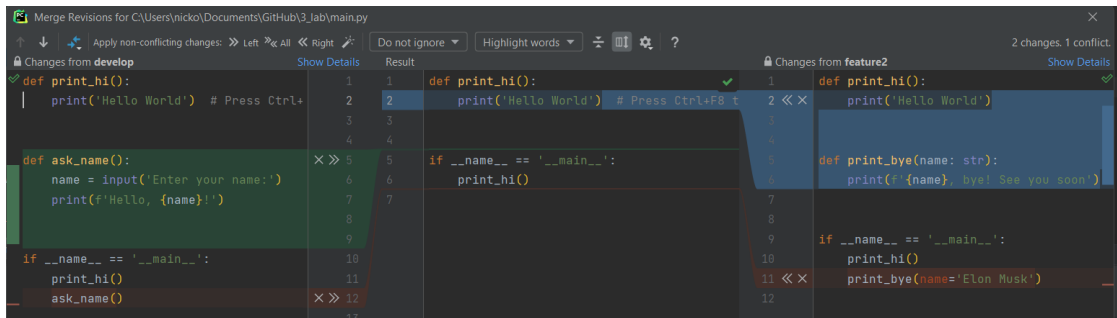
7) Перенесення змін з hotfix1 у develop



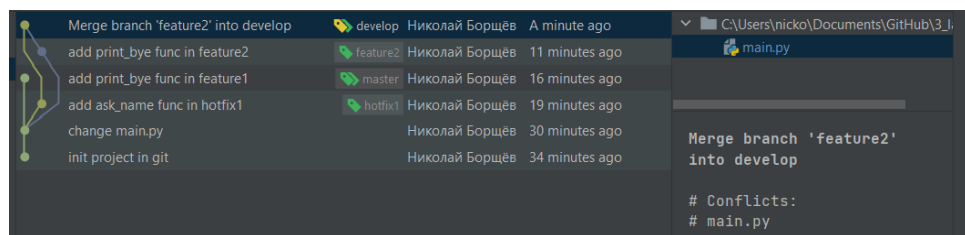
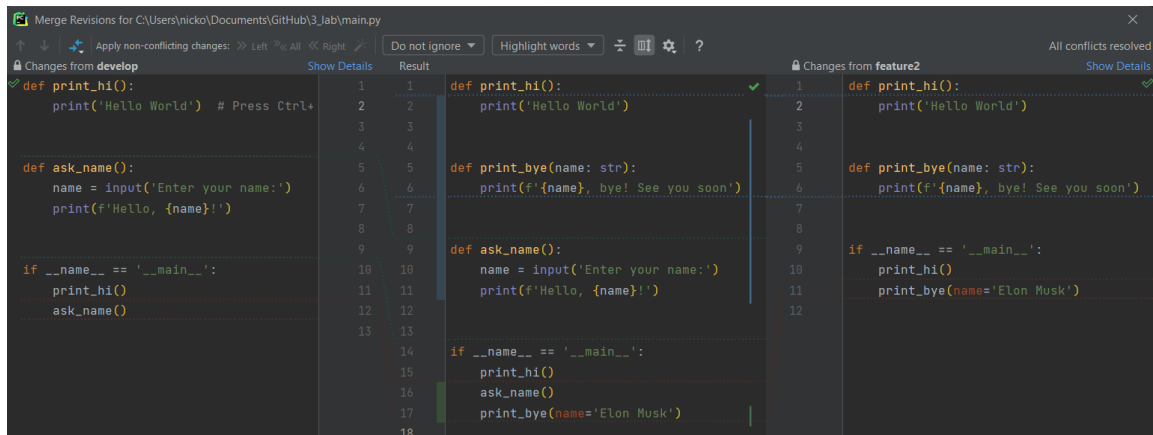
8) Перенесення змін з feature2 у develop



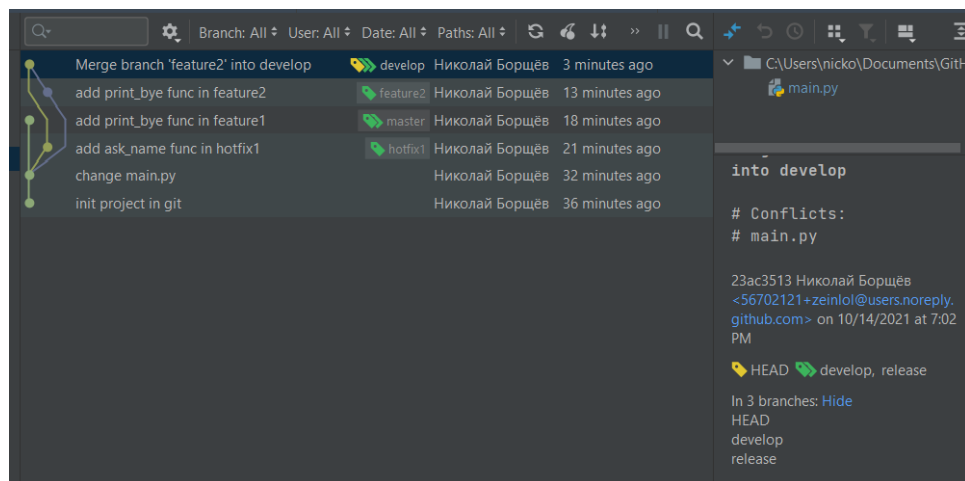
Мереже конфлікт:



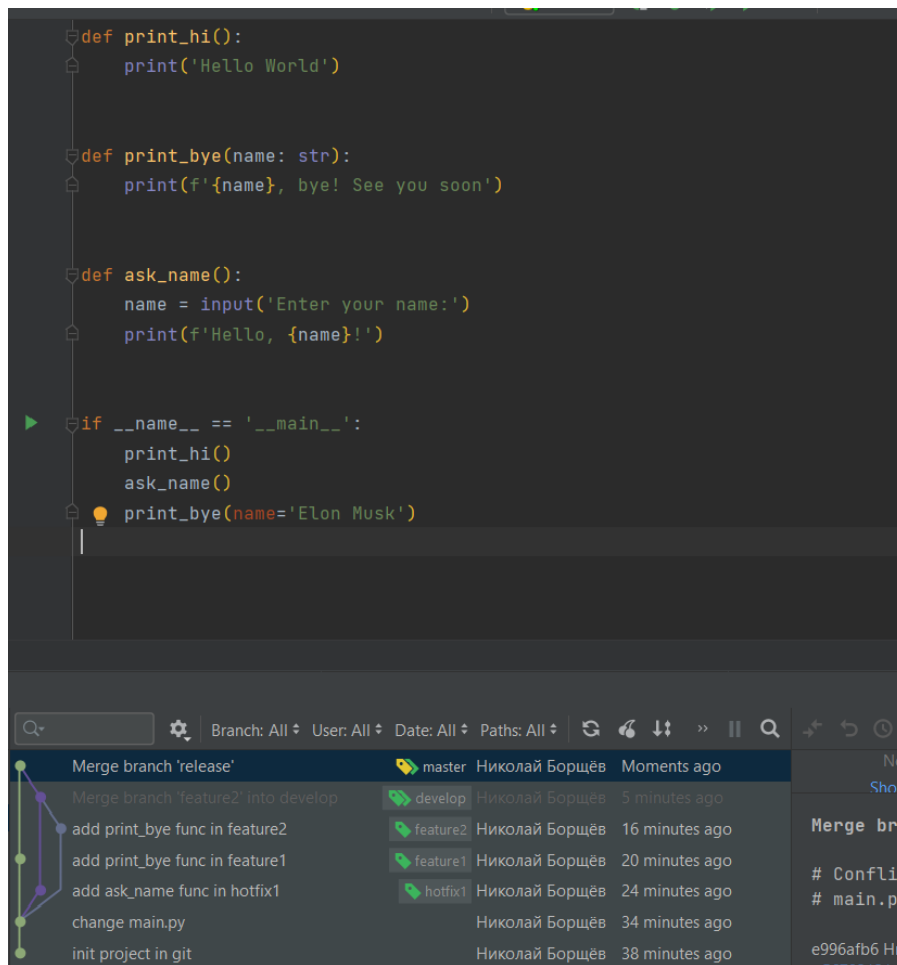
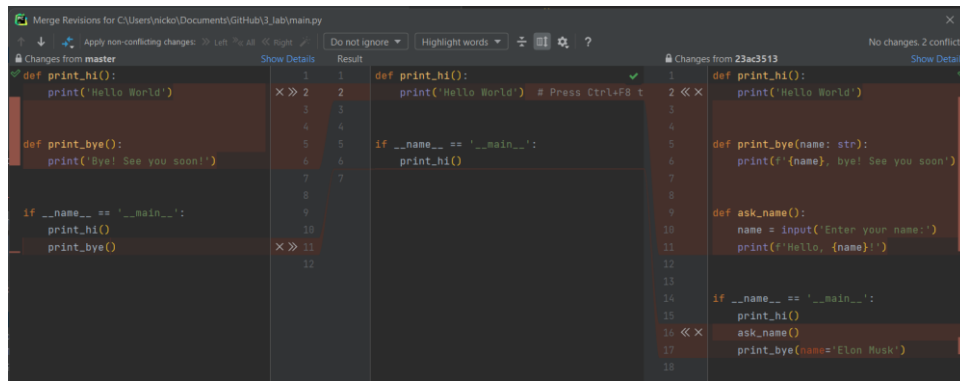
Вирішення конфлікту:



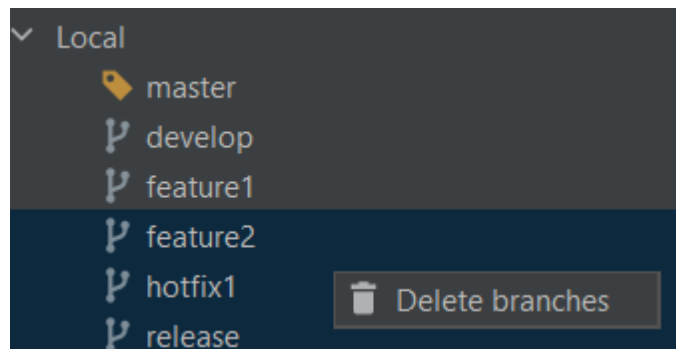
9) Створення нової гілки, зміна даних у гілках



Вирішення конфлікту:



10) Видалення старих гілок



Q

Q

Branch: All User: All Date: All Paths: All

HEAD (Current Branch)

Local

- master
- develop
- feature1

Merge branch 'release'

Merge branch 'feature2' into develop

add print_bye func in feature2

add print_bye func in feature1

add ask_name func in hotfix1

change main.py

init project in git

master

develop

feature1

Николай Борщёв 2 minutes ago

Николай Борщёв 7 minutes ago

Николай Борщёв 18 minutes ago

Николай Борщёв 22 minutes ago

Николай Борщёв 26 minutes ago

Николай Борщёв 36 minutes ago

Николай Борщёв 40 minutes ago

Висновок:

Система Git дозволяє зручно вести контроль версії файлів та проектів на різних пристроях.

В результаті виконання даної лабораторної роботи я ознайомився з основними командами для роботи в Git Bash, навчився механізмам розгалужень, злиття, переміщення гілок, вирішенню merge-конфліктів та створенню файлів. Створив власний тестовий репозиторій де виконував усі завдання, а потім завантажив його та історію змін на сайт github.com.

В результаті виконання даної лабораторної роботи я ознайомився з роботою Git у середовищі PyCharm. Створив простий проект, зробив багато гілок з різними змінами, потім потренирувався вирішувати merge-конфлікти при переміщені гілок.