Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
дисциплины «Программирование на Python»
Вариант 29

Выполнил:
Саенко Андрей Максимович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», заочная
форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. технических
наук, доцент кафедры
инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Основы ветвления Git

Цель: исследование базовых возможностей по работе с локальными и удаленными ветками Git.

Порядок выполнения работы:

Задание 1. Изучить теоретический материал работы.

Задание 2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT.

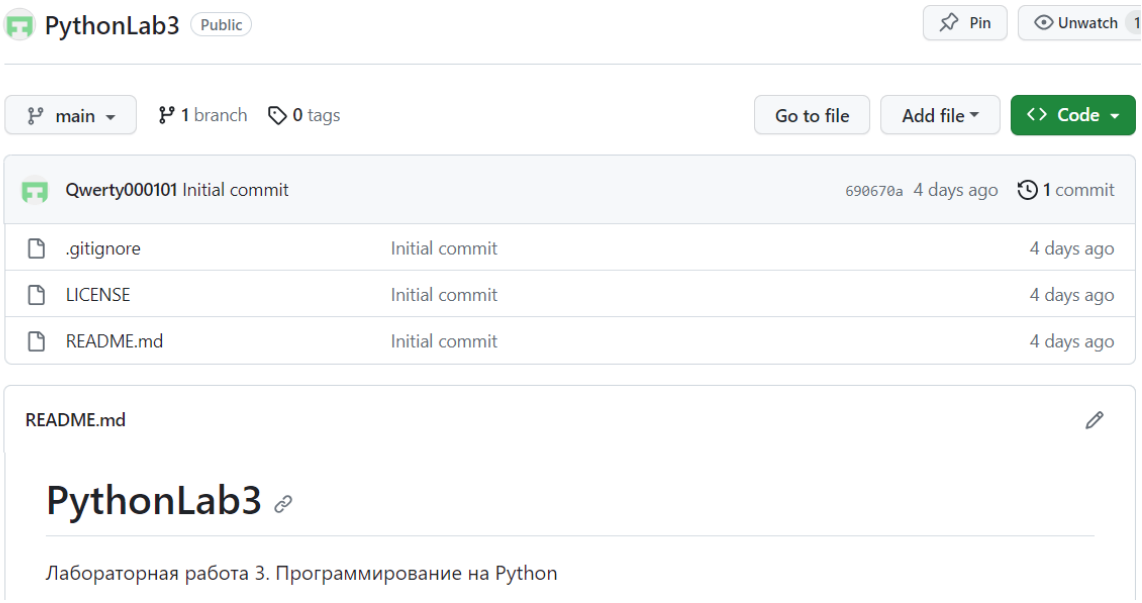

Рисунок 1 – Репозиторий на GitHub

Задание 3. Создать три файла: 1.txt, 2.txt, 3.txt.

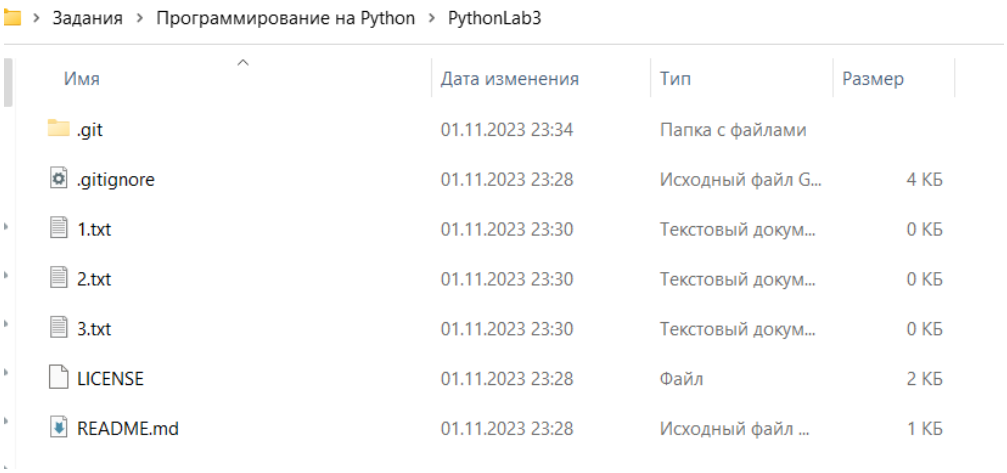

Рисунок 2 – Созданные файлы в локальном репозитории

Задание 4. Проиндексировать первый файл и сделать коммит с комментарием "add 1.txt file".

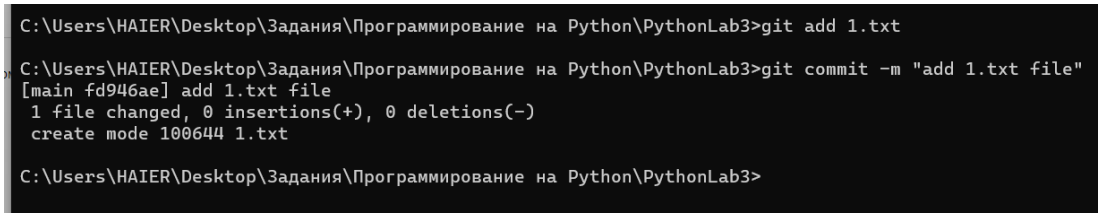

Рисунок 3 – Добавление файла в индекс и создание коммита

Задание 5. Проиндексировать второй и третий файлы.

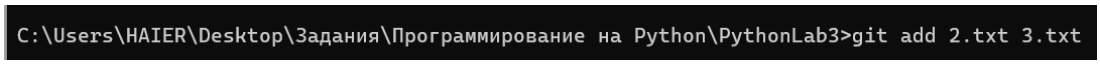

Рисунок 4 – Добавление новых файлов в индекс

Задание 6. Перезаписать уже сделанный коммит с новым комментарием "add 2.txt and 3.txt."

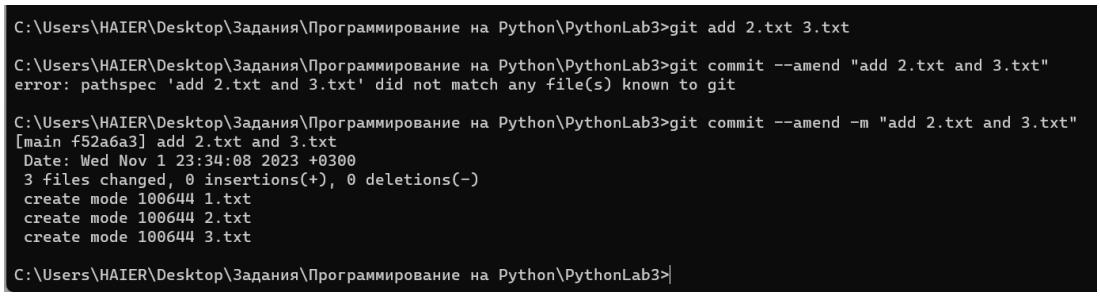

Рисунок 5 – Перезапись коммита

Задание 7. Создать новую ветку my_first_branch.

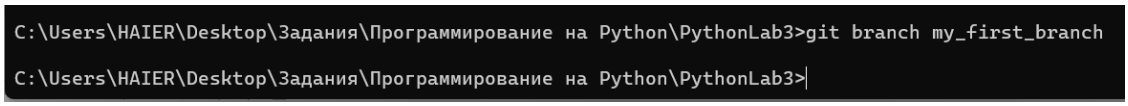

Рисунок 6 – Создание новой ветки

Задание 8. Перейти на ветку и создать новый файл in_branch.txt, закоммитить изменения.

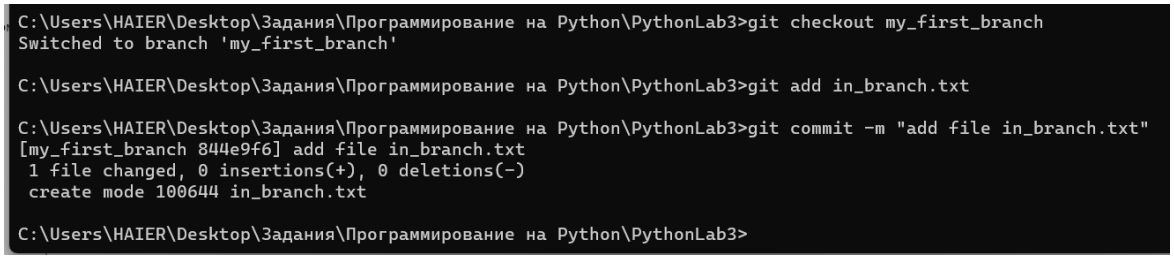

Рисунок 7 – Переход на ветку, добавление созданного файла в индекс и создание коммита

Задание 9. Вернуться на ветку master.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git branch
  main
* my_first_branch

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 8 – Переход на ветку main (master)

Задание 10. Создать и сразу перейти на ветку new_branch.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git branch new_branch

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git checkout new_branch
Switched to branch 'new_branch'

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 9 – Создание ветки new_branch и переход на неё

Задание 11. Сделать изменения в файле 1.txt, добавить строчку “new row in the 1.txt file”, закоммитить изменения.

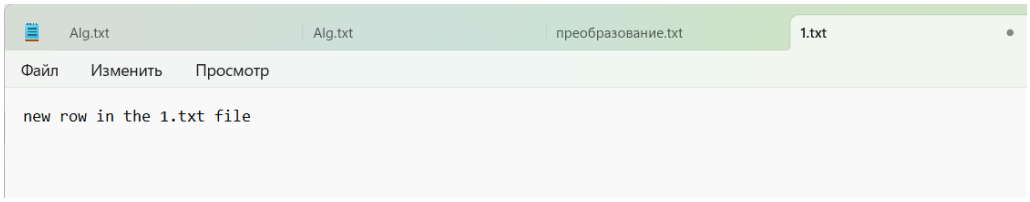

Рисунок 10 – Изменение файла 1.txt

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git add 1.txt

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git commit -m "add new row in file 1.txt"
[new_branch ec6de1f] add new row in file 1.txt
 1 file changed, 1 insertion(+)

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 11 – Внесение изменений в индекс и создание коммита

Задание 12. Перейти на ветку master и слить ветки master и my_first_branch, после чего слить ветки master и new_branch.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git merge my_first_branch
Updating f52a6a3..844e9f6
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 12 – Переход на ветку main, слияние main с ветками my_first_branch и new_branch

Задание 13. Удалить ветки my_first_branch и new_branch.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git branch -d my_first_branch
Deleted branch my_first_branch (was 844e9f6).

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git branch -d new_branch
Deleted branch new_branch (was ec6de1f).

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 13 – Удаление веток

Задание 14. Создать ветки branch_1 и branch_2.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git branch branch_1

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git branch branch_2

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git branch
  branch_1
  branch_2
* main

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 14 – Создание веток

Задание 15. Перейти на ветку branch_1 и изменить файл 1.txt, удалить все содержимое и добавить текст “fix in the 1.txt”, изменить файл 3.txt, удалить все содержимое и добавить текст “fix in the 3.txt”, закоммитить изменения.

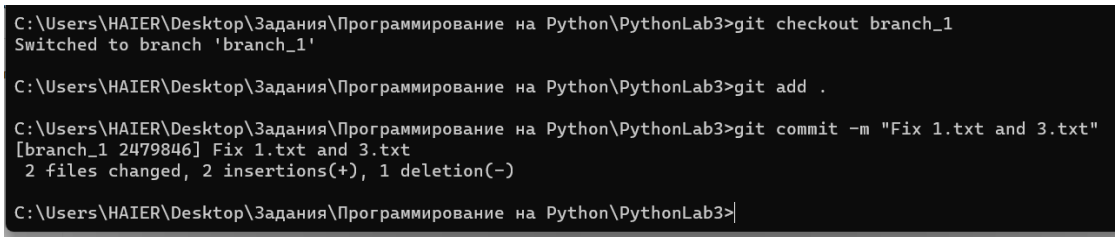

Рисунок 15 – Переход на ветку branch_1, внесение изменений в файлах в индекс и создание коммита

Задание 16. Перейти на ветку branch_2 и также изменить файл 1.txt, удалить все содержимое и добавить текст “My fix in the 1.txt”, изменить файл 3.txt, удалить все содержимое и добавить текст “My fix in the 3.txt”, закоммитить изменения.

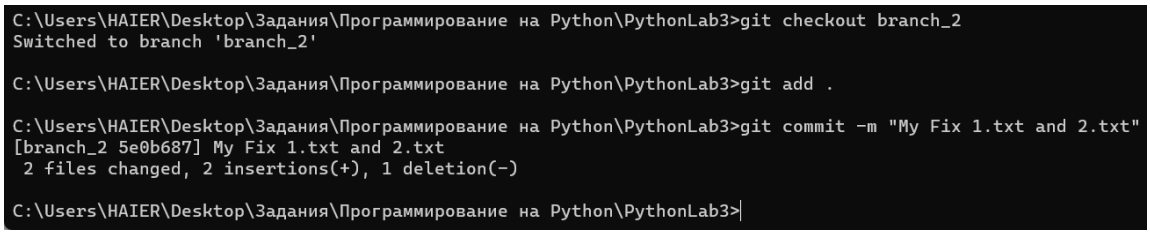

Рисунок 16 – Переход на ветку branch_2, внесение изменений в файлах в индекс и создание коммита

Задание 17. Слить изменения ветки branch_2 в ветку branch_1.

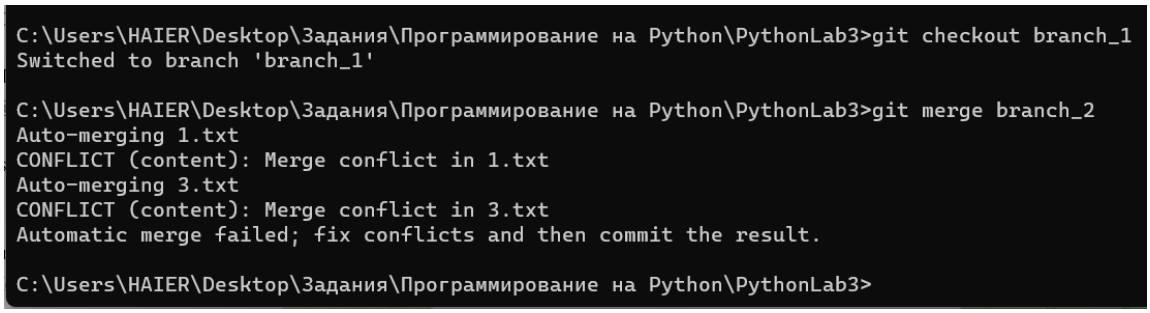

Рисунок 17 – Слияние веток

Задание 18. Решить конфликт файла 1.txt в ручном режиме, а конфликт 3.txt используя команду git mergetool с помощью одной из доступных утилит, например Meld.

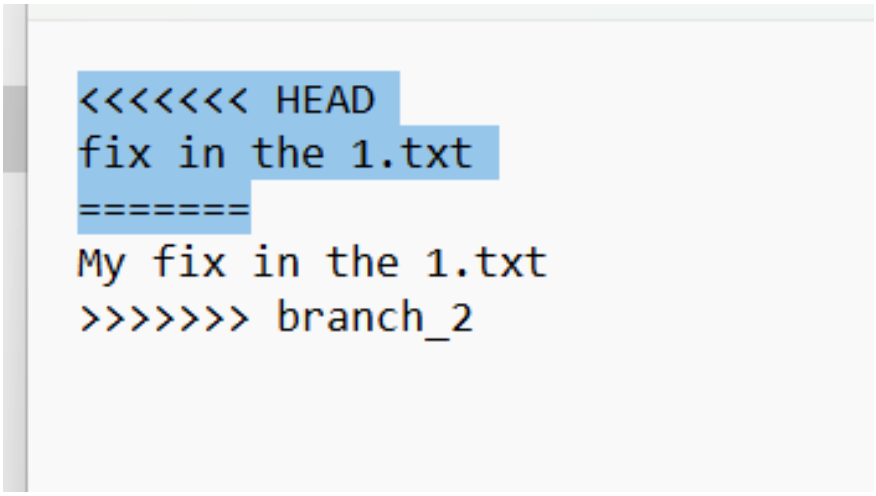

Рисунок 18 – Разрешение конфликта в ручном режиме

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git add 1.txt
```

Рисунок 19 – Добавление файла в индекс

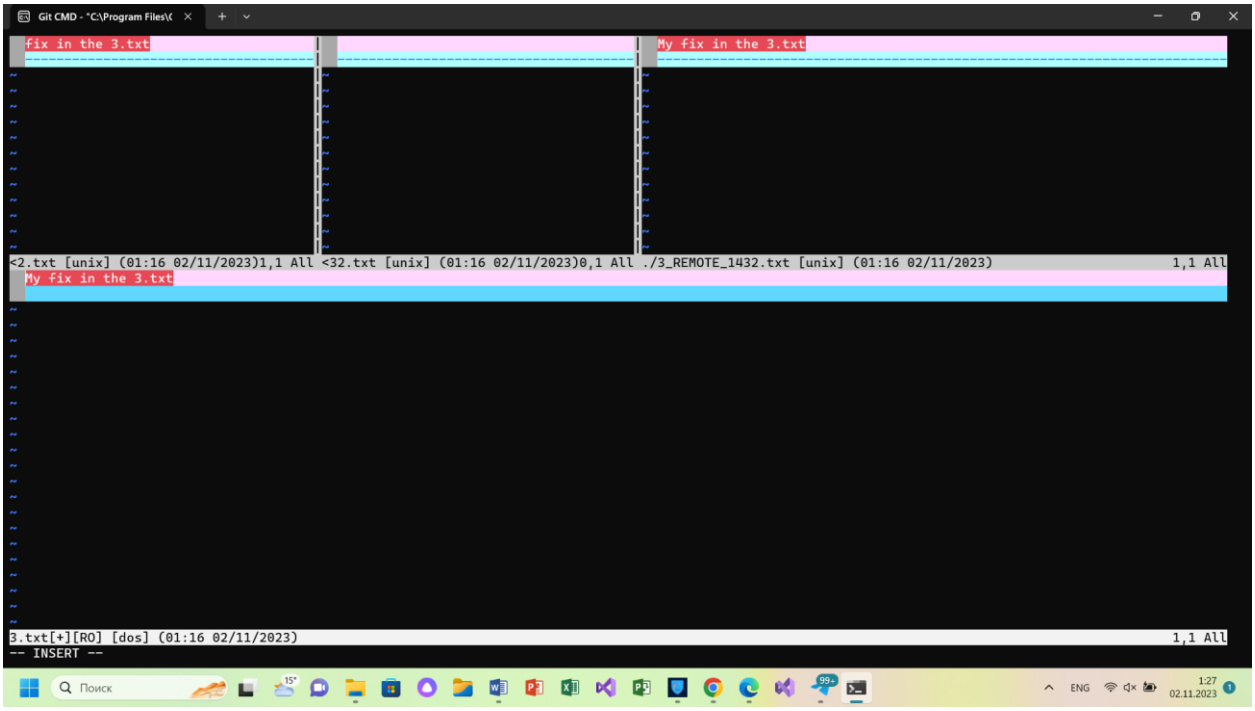

Рисунок 20 – Использование утилиты Meld для разрешения конфликта

Задание 19. Отправить ветку branch_1 на GitHub.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git push origin branch_1
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (22/22), 1.81 KiB | 1.81 MiB/s, done.
Total 22 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (8/8), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:      https://github.com/Qwerty000101/PythonLab3/pull/new/branch_1
remote:
To https://github.com/Qwerty000101/PythonLab3.git
 * [new branch]      branch_1 -> branch_1

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 21 – Отправка ветки branch_1 на GitHub

Задание 20. Создать средствами GitHub удаленную ветку branch_3

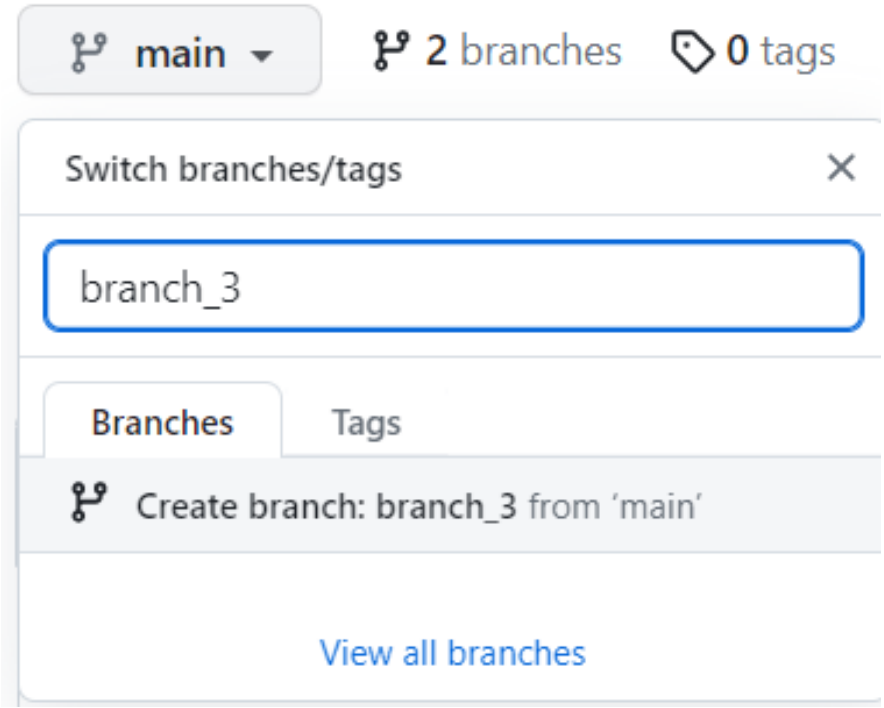

Рисунок 22 – Создание ветки на GitHub

Задание 21. Создать в локальном репозитории ветку отслеживания удаленной ветки branch_3.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git fetch
From https://github.com/Qwerty000101/PythonLab3
 * [new branch]      branch_3   -> origin/branch_3

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git branch
* branch_1
  branch_2
  main

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git checkout -b branch_3_tracking origin/branch_3
Switched to a new branch 'branch_3_tracking'
branch 'branch_3_tracking' set up to track 'origin/branch_3'.

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 23 – Создание ветки отслеживания в локальном репозитории

Задание 22. Перейти на ветку branch_3 и добавить файл файл 2.txt строку "the final fantasy in the 4.txt file"

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git checkout branch_3_tracking
Switched to branch 'branch_3_tracking'
Your branch is up to date with 'origin/branch_3'.

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 24 – Переход на ветку branch_3_tracking

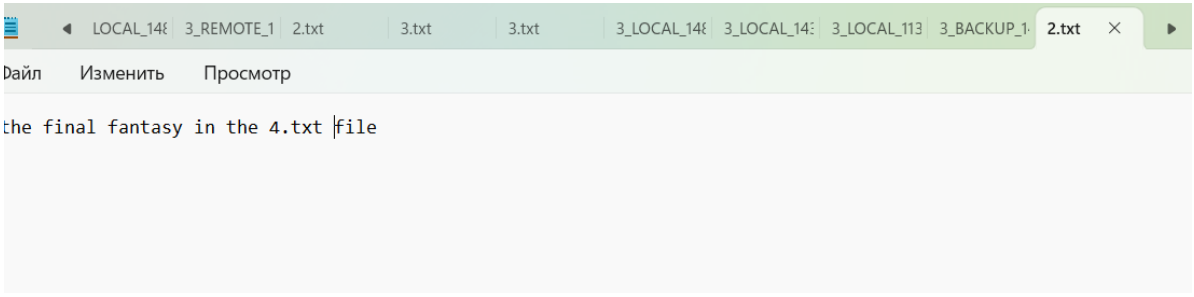

Рисунок 25 – Внесение изменений в файл

Задание 23. Выполнить перемещение ветки master на ветку branch_2.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git rebase main
Current branch branch_2 is up to date.
```

Рисунок 25 – Перемещение ветки

Задание 24. Отправить изменения веток master и branch_2 на GitHub.

```
C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>git push origin branch_2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:      https://github.com/Qwerty000101/PythonLab3/pull/new/branch_2
remote:
To https://github.com/Qwerty000101/PythonLab3.git
 * [new branch]      branch_2 -> branch_2

C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab3>
```

Рисунок 26 – Отправка изменений на сервер

Ответы на контрольные вопросы

1. Что такое ветка?

Ветка - это перемещаемый указатель на один из коммитов.

2. Что такое HEAD?

HEAD – это указатель, задача которого ссылаться на определенный коммит в репозитории.

3. Способы создания веток.

Создать новую ветку можно с помощью команды git branch.

4. Как узнать текущую ветку?

Чтобы узнать текущую ветку, нужно ввести команду git branch, будет ввыведен список всех веток, а текущая будет помечена знаком «*».

5. Как переключаться между ветками?

Чтобы переключиться на другую ветку, нужно ввести команду git checkout <название ветки>.

6. Что такое удаленная ветка?

Удаленные ветки – это ссылки на состояние веток в удаленных репозиториях.

7. Что такое ветка отслеживания?

Ветки слежения - это ссылки на определённое состояние удалённых веток.

8. Как создать ветку отслеживания?

Создать ветку отслеживания можно с помощью команды git checkout -b <branch> <remote>/<branch>.

9. Как отправить изменения из локальной ветки в удаленную ветку?

Для отправки изменений из локальной ветки в удаленную ветку используется команда git push <удаленный сервер> <название ветки>.

10. В чем отличие команд git fetch и git pull ?

Команда git fetch получает с сервера все изменения, которых ещё нет, но не будет изменять состояние рабочей директории. Эта команда просто получает данные и позволяет пользователю самостоятельно сделать слияние. Romанда git pull в большинстве случаев является командой git fetch, за которой непосредственно следует команда git merge .

11. Как удалить локальную и удаленную ветки?

Для удаления локальной ветки используется команда git branch –d <название ветки>. Для удаления удалённой ветки используется команда git push <название сервера> --delete <название ветки>.

12. Какие основные типы веток присуствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

В модели git-flow присутствуют два основных типа веток:

1) master - ветка, содержащая стабильный код, который готов к выпуску;

2) develop - ветка, содержащая последние изменения и новые функции, которые еще не были выпущены в master.

Также в модели git-flow могут использоваться следующие типы веток: feature (ветки, создаваемые для разработки новых функций или исправления ошибок), release (ветки, создаваемые для подготовки к выпуску новой версии программного продукта), hotfix (ветки, создаваемые для исправления критических ошибок в уже выпущенной версии программного продукта).

Работа с ветками в модели git-flow организована следующим образом:

Создается ветка develop, которая является основной для разработки новых функций и исправления ошибок. Для каждой новой функции или исправления ошибки создается отдельная ветка feature, которая ветвится от ветки develop. После завершения работы над функцией или ошибкой, ветка feature сливается обратно в ветку develop. Когда все необходимые функции и исправления были добавлены в ветку develop и прошли тестирование, создается ветка release для подготовки к выпуску новой версии программного продукта. Ветка release проходит тестирование и исправление ошибок (если таковые обнаруживаются), после чего сливается в ветку master и помечается тегом с номером версии. Если в уже выпущенной версии программного продукта обнаруживается критическая ошибка, создается ветка hotfix, которая исправляет ошибку и сливается обратно в ветки master и develop.

По сравнению с моделью магистральной разработки, в Git-flow используется больше веток, каждая из которых существует дольше, а коммиты обычно крупнее. В соответствии с этой моделью разработчики создают функциональную ветку и откладывают ее слияние с главной магистральной веткой до завершения работы над функцией. Такие

долгосрочные функциональные ветки требуют тесного взаимодействия разработчиков при слиянии и создают повышенный риск отклонения от магистральной ветки. В них также могут присутствовать конфликтующие обновления.

13. На прошлой лабораторной работе было задание выбрать одно из программных средств с GUI для работы с Git. Необходимо в рамках этого вопроса привести описание инструментов для работы с ветками Git, предоставляемых этим средством.

Сервис BitBucket предоставляет следующие инструменты для работы с ветками Git:

1. Создание веток: пользователь может создавать новые ветки в своем репозитории с помощью интерфейса BitBucket или командной строки Git.

2. Слияние веток: пользователь может объединять ветки с помощью интерфейса BitBucket или командной строки Git. В BitBucket также доступен инструмент Pull Request, который позволяет создавать запросы на слияние и проводить код-ревью перед объединением веток.

3. Удаление веток: пользователь может удалять ветки с помощью интерфейса BitBucket или командной строки Git.

4. Защита веток: в BitBucket доступна функция защиты веток, которая позволяет ограничить доступ к изменению определенных веток только определенным пользователям или группам пользователей.

Вывод

При выполнении работы было выполнено исследование базовых возможностей по работе с локальными и удаленными ветками Git, получены навыки создания, слияния, удаления веток, а также отправки их на удалённый сервер и создания веток слежения.