

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»**

Выполнил:  
Гайчук Дарья Дмитриевна  
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», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А.-доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

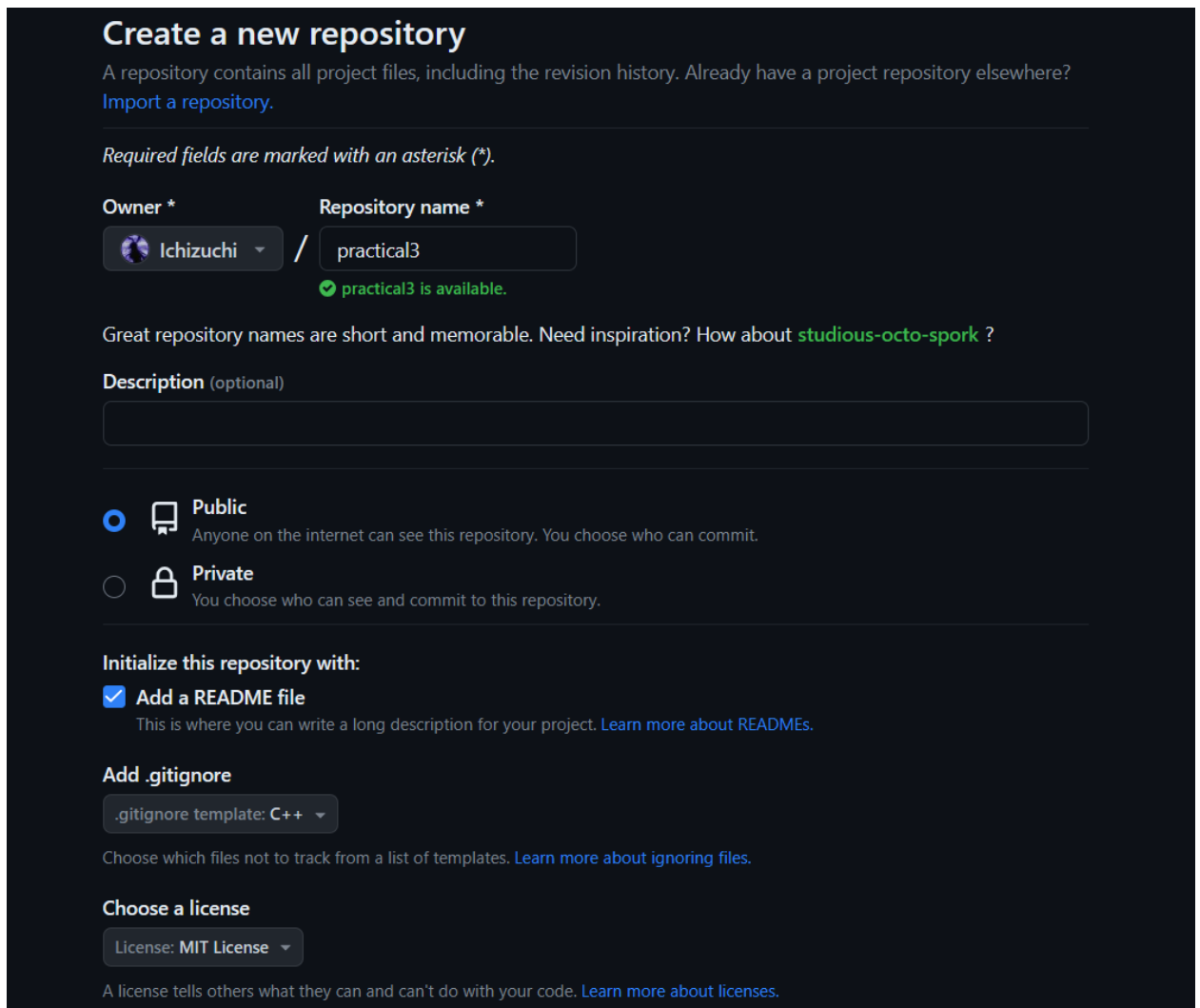
Ставрополь, 2023 г.

**Тема:** «Основы ветвления Git»

**Цель работы:** исследовать базовые возможности по работе с локальными и удаленными ветками Git.

**Порядок выполнения работы:**

1. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT.




**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*


Owner \* Repository name \*


 Ichizuchi / practical3

✔ practical3 is available.

Great repository names are short and memorable. Need inspiration? How about [studious-octo-spork](#) ?

**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**

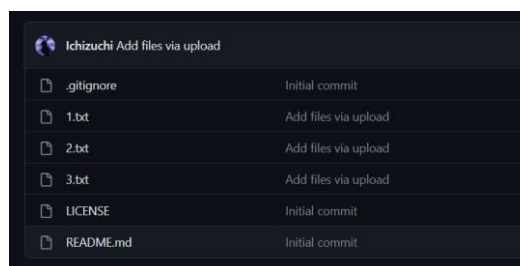
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

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

Рисунок 1. Новый репозиторий.

2. Добавила 3 новых текстовых файла.









Ichizuchi Add files via upload	
 .gitignore	Initial commit
 1.txt	Add files via upload
 2.txt	Add files via upload
 3.txt	Add files via upload
 LICENSE	Initial commit
 README.md	Initial commit

Рисунок 2. Новые текстовые файлы.

3. Проиндексировала первый файл и сделать коммит.

```
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
```

Рисунок 3. Добавление изменений и их фиксация

4. Перезаписал уже сделанный коммит.

```
@Ichizuchi →/workspaces/practical3 (main) $ git commit --amend
hint: Waiting for your editor to close the file...
[main cc65844] Add files via upload
Date: Thu Sep 21 13:40:48 2023 +0300
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt
```

Рисунок 4. Редактирование существующего коммита

5. Создание новой ветки и переход на нее и создать новый файл in\_branch.txt.

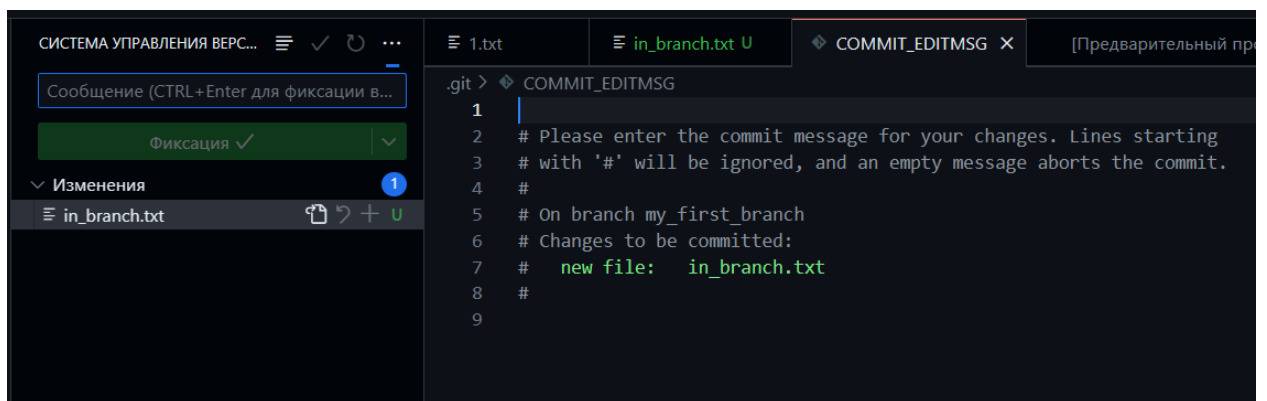


Рисунок 5. Создание файла и фиксация.

6. Перешла на новую ветку main, создала и сразу перешла на ветку new\_branch

```
@Ichizuchi →/workspaces/practical3 (my_first_branch) $ git checkout main
A      in_branch.txt
Switched to branch 'main'
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" if you want to integrate the remote branch with yours)
@Ichizuchi →/workspaces/practical3 (main) $ git checkout -b new_branch
Switched to a new branch 'new_branch'
```

Рисунок 6. Создание и сразу переход на ветку.

7. Сделала изменения в файле 1.txt, добавил строчку “new row in the 1.txt file”, закоммитила изменения.

```

Switched to a new branch 'new_branch'
● @Ichizuchi →/workspaces/practical3 (new_branch) $ git add 1.txt
● @Ichizuchi →/workspaces/practical3 (new_branch) $ git commit -m "new row in the 1.txt file"
[new branch 56466ff] new row in the 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

```

Рисунок 7. Изменение в 1.txt фиксация этих изменений

8. Перешла на ветку main и слил ветки main и my\_first\_branch, после слила ветки main и new\_branch.

```

● @Ichizuchi →/workspaces/practical3 (new_branch) $ git checkout main
Switched to branch 'main'
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" if you want to integrate the remote branch with yours)
● @Ichizuchi →/workspaces/practical3 (main) $ git merge my_first_branch
Already up to date.

```

Рисунок 8. Слияние веток.

9. Удалила ветки my\_first\_branch и new\_branch.

```

● @Ichizuchi →/workspaces/practical3 (main) $ git branch -d my_first_branch
Deleted branch my_first_branch (was cc9293c).
● @Ichizuchi →/workspaces/practical3 (main) $ git branch -d new_branch
Deleted branch new_branch (was cc9293c).

```

Рисунок 9. Удаление веток

10. Создала ветки branch\_1 и branch\_2.

```

● @Ichizuchi →/workspaces/practical3 (main) $ git branch branch_1
● @Ichizuchi →/workspaces/practical3 (main) $ git branch branch_2

```

Рисунок 10. Создание новых веток.

11. Перешла на ветку branch\_1 и изменила файл 1.txt, удалила все содержимое и добавила текст “fix in the 1.txt”, изменила файл 3.txt, удалила все содержимое и добавила текст “fix in the 3.txt”, закоммитила изменения.

```

● @Ichizuchi →/workspaces/practical3 (branch_1) $ git commit -m "fix files 1 and 3"
[branch_1 6062e54] fix files 1 and 3
1 file changed, 1 insertion(+)
create mode 160000 practical3

```

Рисунок 11. Изменения в первой и во второй ветке, фиксация изменений

12. Перешла на ветку branch\_2 и изменила файл 1.txt, удалила все содержимое и добавила текст “My fix in the 1.txt”, изменила файл 3.txt, удалила все содержимое и добавила текст “My fix in the 3.txt”, закоммитила изменения.

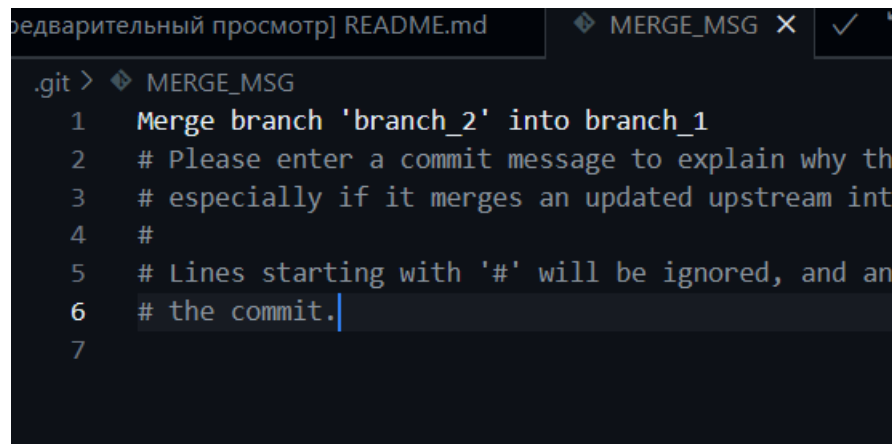
```

● @Ichizuchi → /workspaces/practical3 (branch_2) $ git commit -m "My
  fix files 1 and 3"
[branch_2 46ec928] My fix files 1 and 3
1 file changed, 1 insertion(+)
create mode 160000 practical3

```

Рисунок 12. Переход на вторую ветку.

16. Слила изменения ветки branch\_2 в ветку branch\_1.



```

.git > MERGE_MSG
1 Merge branch 'branch_2' into branch_1
2 # Please enter a commit message to explain why th
3 # especially if it merges an updated upstream into
4 #
5 # Lines starting with '#' will be ignored, and an
6 # the commit.
7

```

Рис. 16. Слияние веток.

17. Решила конфликт файла 1.txt и 2.txt

```

● @Ichizuchi → /workspaces/practical3 (branch_1) $ git status
On branch branch_1
nothing to commit, working tree clean

```

Рис. 17. Решение конфликта

18. Отправила branch\_1 на удаленный git push origin branch\_1.

```

nothing to commit, working tree clean
● @Ichizuchi → /workspaces/practical3 (branch_1) $ git push --set-upstr
eam origin branch_1
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 469 bytes | 469.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/Ichizuchi/practical3
cc9293c..05246db branch_1 -> branch_1
branch 'branch_1' set up to track 'origin/branch_1'.

```

Рис. 18. Отправление ветки на удаленный сервер

18. Создала средствами GitHub удаленную ветку branch\_3.

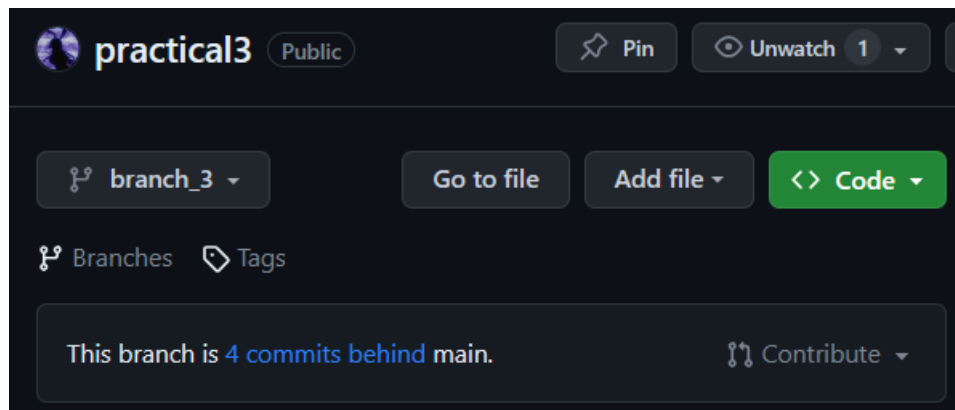


Рис. 18. Создание ветки на GitHub.

19. Создала в локальном репозитории ветку отслеживания удаленной ветки branch\_3.

```
* [new branch] my_first_branch -> origin/my_first_branch
* [new branch] new_branch -> origin/new_branch
● @Ichizuchi →/workspaces/practical3 (branch_1) $ git checkout -b branch_3 origin/branch_3
warning: unable to rmdir 'practical3': Directory not empty
branch 'branch_3' set up to track 'origin/branch_3'.
Switched to a new branch 'branch_3'
```

Рис. 19. Создание ветки отслеживания

20. Перешла на ветку branch\_3 и добавил файл файл 2.txt строку "the final fantasy in the 4.txt file".

```
● @Ichizuchi →/workspaces/practical3 (branch_3) $ git commit -m "the final fantasy in the 4.txt. file"
[branch_3 740147f] the final fantasy in the 4.txt. file
1 file changed, 1 insertion(+)
create mode 160000 practical3
```

21. Выполнила перемещение ветки main на ветку branch\_2.

```
Successfully rebased and updated refs/heads/main.
● @Ichizuchi →/workspaces/practical3 (main) $ git checkout branch_2
Switched to branch 'branch_2'
● @Ichizuchi →/workspaces/practical3 (branch_2) $ git merge main
Already up to date.
```

Рис. 25. Перемещение веток.

22. Отправила изменения веток master и branch\_2 на удаленный сервер.

```
● @Ichizuchi →/workspaces/practical3 (branch_2) $ git push origin branch_2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0

Создать запрос на вытягивание (CTRL + щелчок) on GitHub by visiting:
remote:      https://github.com/Ichizuchi/practical3/pull/new/branch_2
remote:
To https://github.com/Ichizuchi/practical3
* [new branch]      branch_2 -> branch_2
```

Рис. 22. Отправка изменения веток на GitHub.

Ответы на контрольные вопросы:

1. Что такое ветка?

Почти каждая система контроля версий (СКВ) в какой-то форме поддерживает ветвление. Используя ветвление, Вы отклоняетесь от основной линии разработки и продолжаете работу независимо от неё, не вмешиваясь в основную линию.

2. Что такое HEAD?

HEAD в Git – это указатель на текущую ссылку ветви, которая, в свою очередь, является указателем на последний сделанный вами коммит или последний коммит, который был извлечен из вашего рабочего каталога.

3. Способы создания веток.

Создать ветку можно с помощью двух команд. Команда, которая просто создает ветку: `git branch "name_branch"`. Команда, которая создает ветку и сразу же к ней переходит: `git checkout -b "name_branch"`.

4. Как узнать текущую ветку?

Текущую ветку можно узнать с помощью команды: `git branch`.

5. Как переключаться между ветками?

Между ветками можно переключаться с помощью команды: `git checkout "name_branch"`.

6. Что такое удаленная ветка?

Удаленные ветки — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать.

7. Что такое ветка отслеживания?

Отслеживаемые ветки — это локальные ветки, которые напрямую связаны с удалённой веткой. Если, находясь на отслеживаемой ветке, вы наберёте `git push`, Git уже будет знать, на какой сервер и в какую ветку отправлять изменения.

8. Как создать ветку отслеживания? Ветку отслеживания можно создать с помощью команды: `git checkout -- track origin/`.

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

Отправить изменения из локальной ветки в удаленную можно с помощью команды: `git push origin`.

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

Команда `git fetch` получает с сервера все изменения, которых у вас ещё нет, но не будет изменять состояние вашей рабочей директории. Эта команда просто получает данные и позволяет вам самостоятельно сделать слияние. Тем не менее, существует команда `git pull`, которая в большинстве случаев является командой `git fetch`, за которой непосредственно следует команда `git merge`.

11. Как удалить локальную и удаленную ветки?

Для удаление локальной ветки используется команда: `git branch -d` Для удаления удаленной ветки используется команда: `git push --delete origin/`

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

Существуют следующие типы ветвей: 1) ветви функциональности; 2) ветви релизов; 3) ветви исправлений.

Ветви функциональности (`feature branches`), также называемые иногда тематическими ветвями (`topic branches`), используются для разработки новых функций, которые должны появиться в текущем или будущем релизах.

Ветви релизов (`release branches`) используются для подготовки к выпуску новых версий продукта. Они позволяют расставить финальные точки над *i* перед выпуском новой версии.



Ветви для исправлений (hotfix branches) весьма похожи на ветви релизов (release branches), так как они тоже используются для подготовки новых выпусков продукта, разве лишь незапланированных.

Недостатки git flow: авторам приходится использовать ветку develop вместо master, поскольку master зарезервирован для кода. Вторая проблема процесса git flow – сложности, возникающие из-за веток для патчей и для релиза. Подобная структура может подойти некоторым организациям, но для абсолютного большинства она просто убийственно излишняя.

Вывод: исследовала базовые возможности по работе с локальными и удаленными ветками Git.