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 №2.14
дисциплины «Основы программной инженерии»

Выполнил:
Соколов Михаил Романович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Руководитель практики:
Богданов С.С., ассистент кафедры
инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Установка пакетов в Python. Виртуальные окружения.

Цель работы: приобретение навыков по работе с менеджером пакетов pip и виртуальными окружениями с помощью языка программирования Python 3.x.

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и .gitignore файл для языка программирования Python:

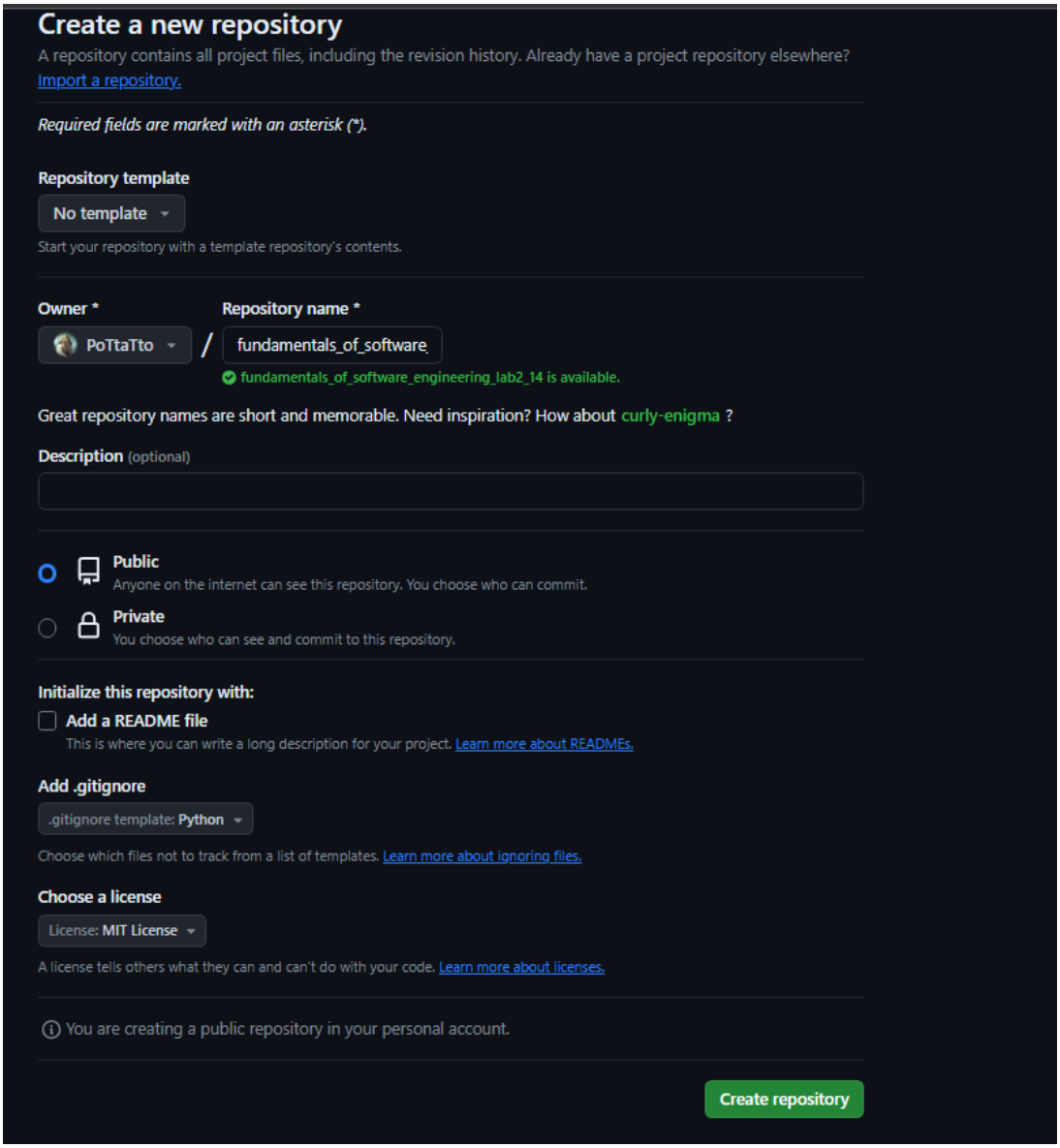

Рисунок 1 – Создание репозитория с заданными настройками

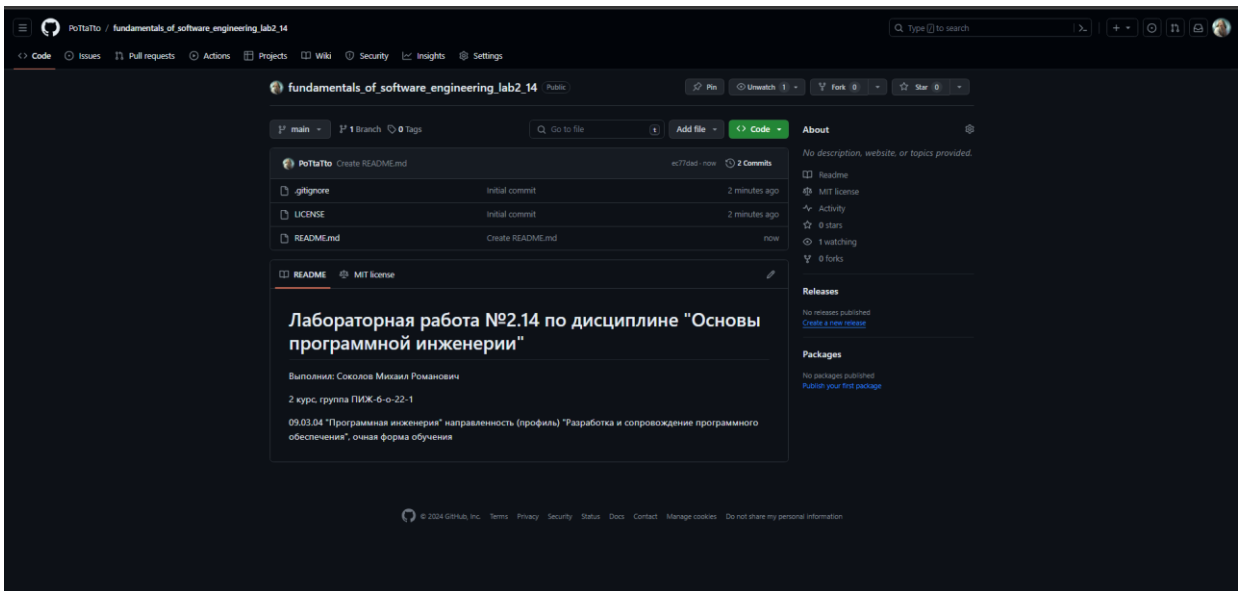

Рисунок 2 – Созданный репозиторий

Рисунок 3 – Клонирование репозитория

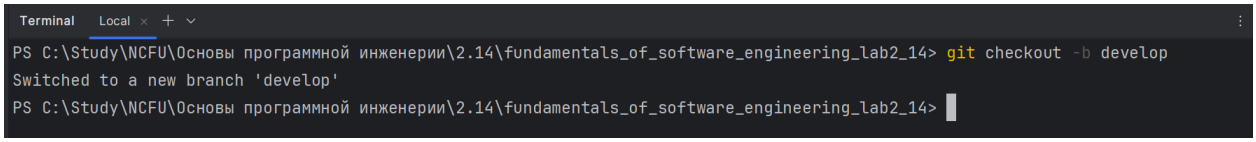

Рисунок 4 – Создание ветки develop, где будут происходить изменения проекта до его полного релиза

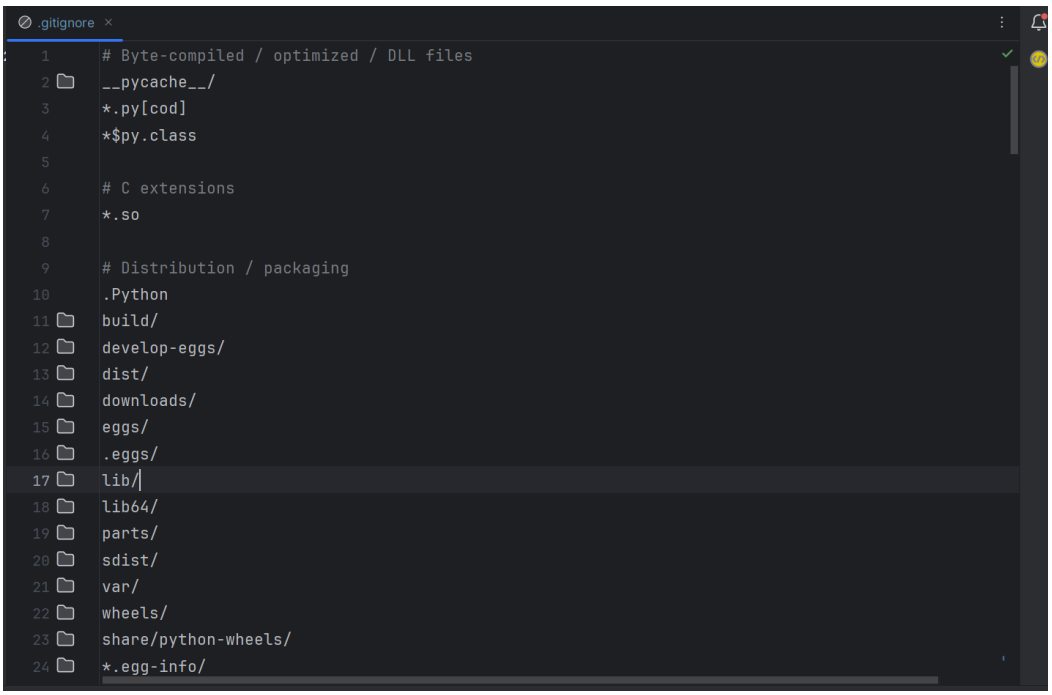

Рисунок 5 – Часть .gitignore, созданного GitHub

2. Создадим и активируем виртуальное окружение Anaconda с именем репозитория:

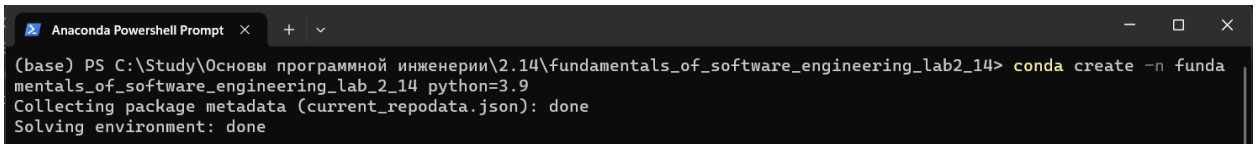

Рисунок 6 – Создание виртуального окружения

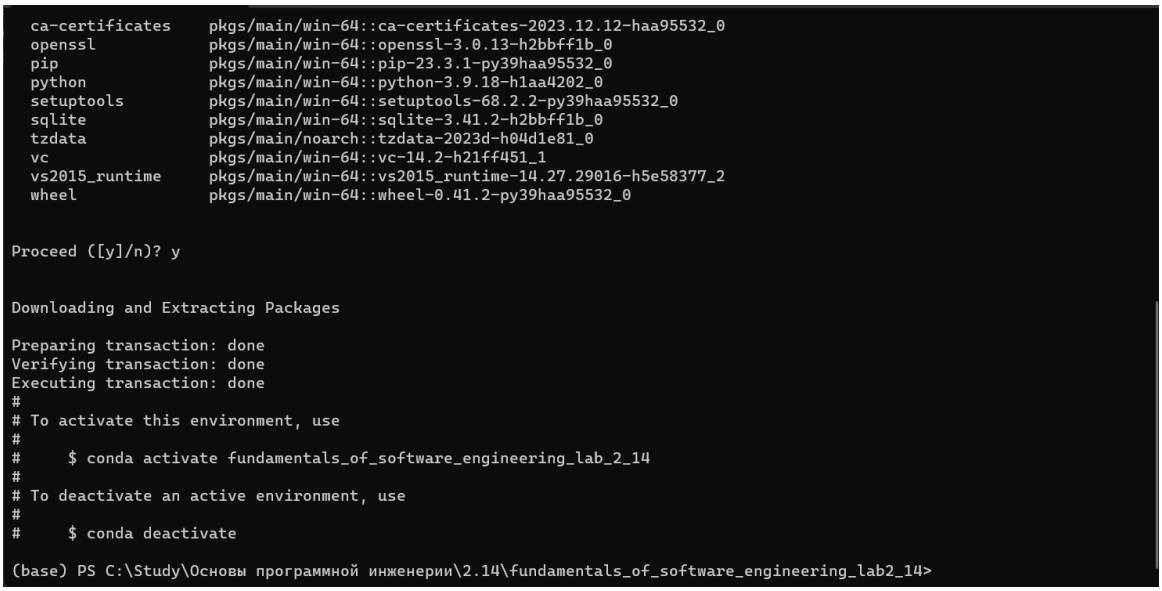

Рисунок 7 – Успешная установка

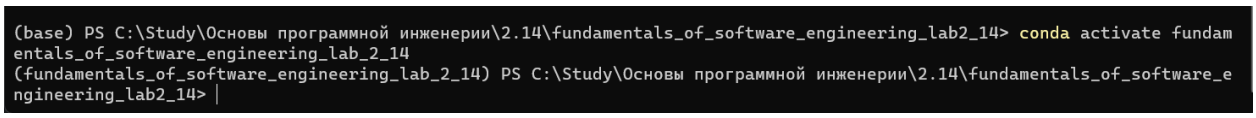

Рисунок 8 – Активация виртуального окружения

3. Установим пакеты pip, NumPy, Pandas и SciPy:

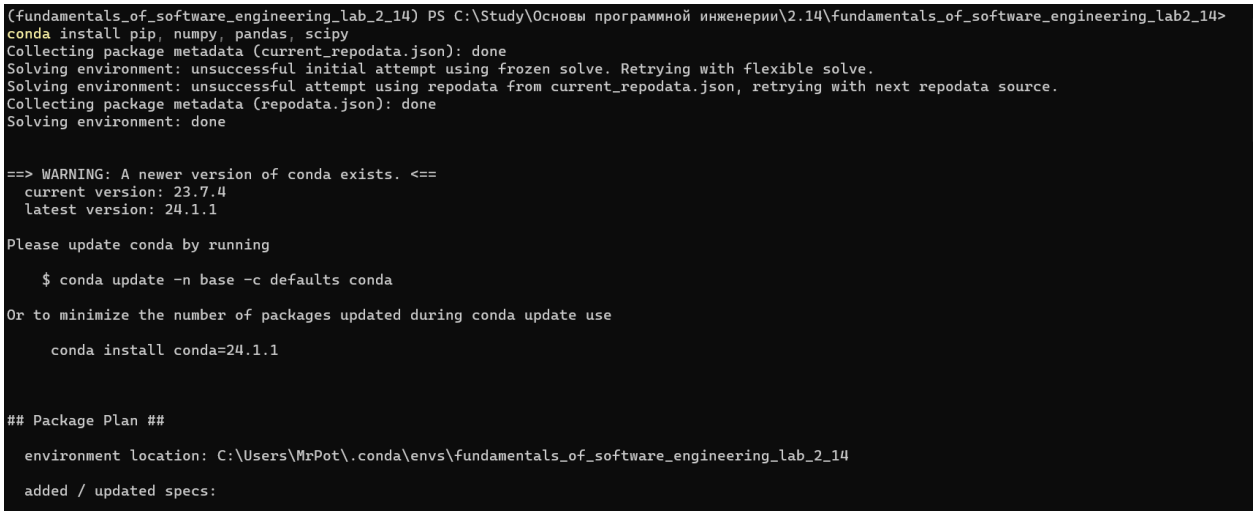

Рисунок 9 – Начало установки нужных пакетов

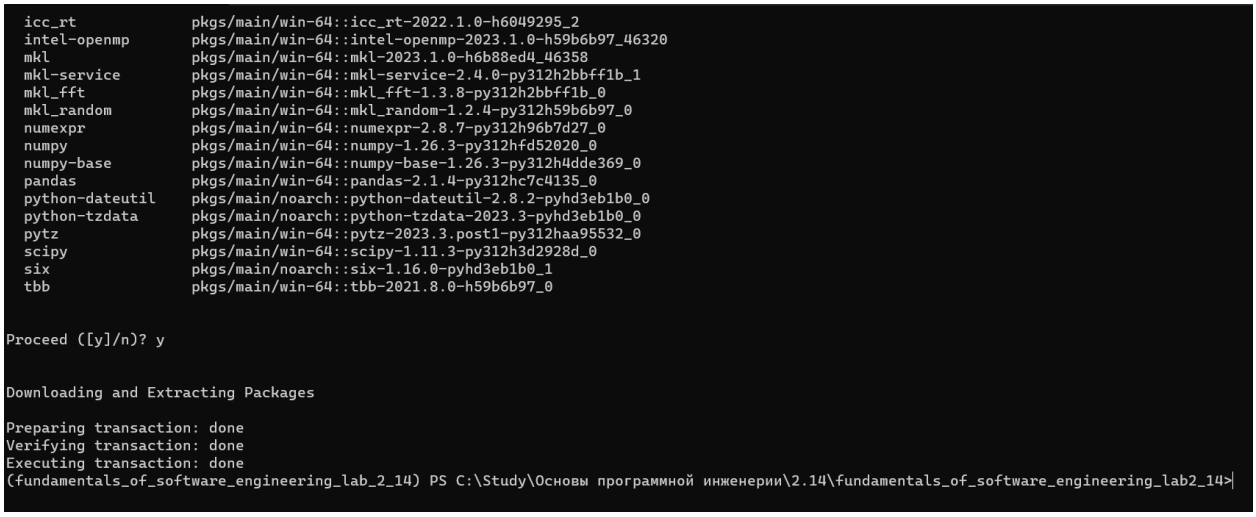

Рисунок 10 – Завершение установки пакетов

4. Попробуем установить пакет TensorFlow:

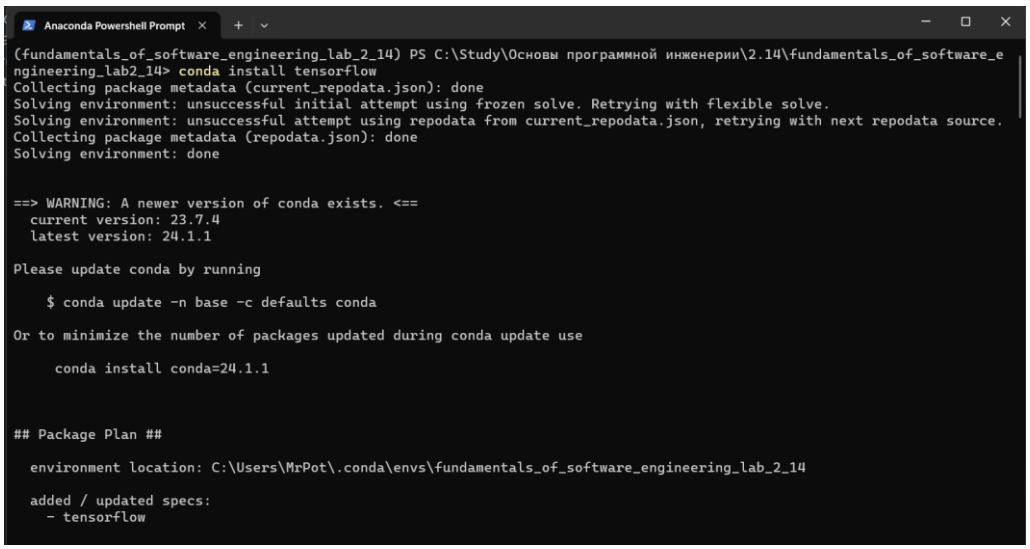

Рисунок 11 – Попытка установки с помощью утилиты conda

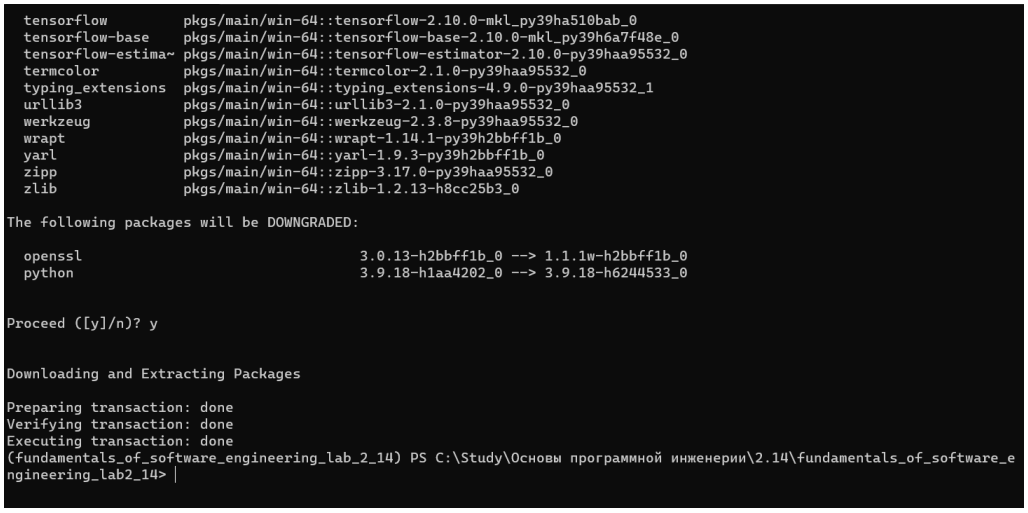

Рисунок 12 – Успешная установка пакета

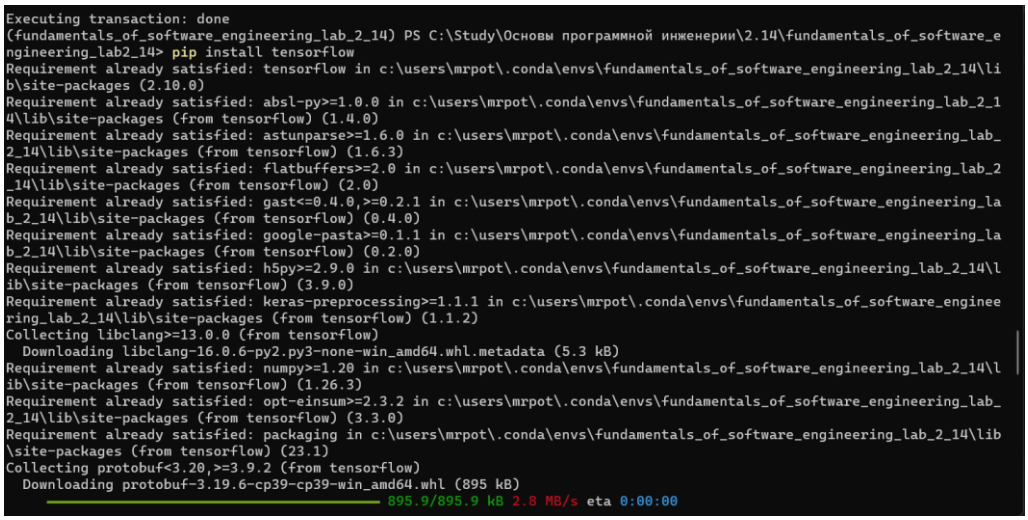

Рисунок 13 – Переустановка пакета с помощью pip

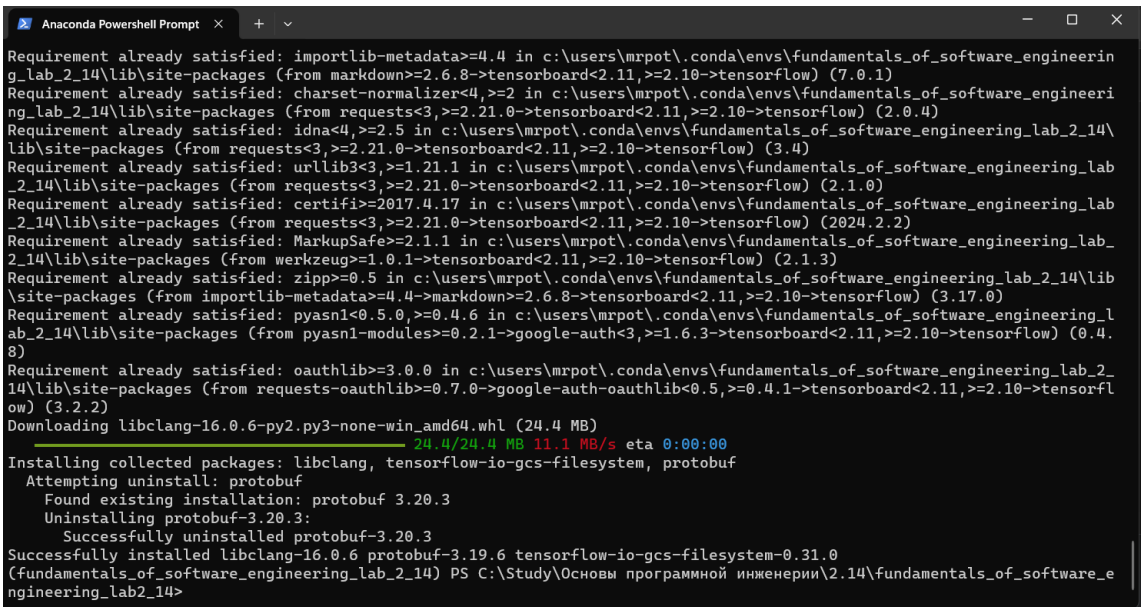

Рисунок 14 – Завершение переустановки с помощью pip

5. Сформируем файлы requirements.txt и environment.yml:

```
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> pip freeze > requirements.txt
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> conda env export > environment.yml
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14>
```

Рисунок 15 – Создание файлов

```
pycparser @ file:///tmp/build/80754af9/pycparser_1636541352034/work
PyJWT @ file:///C:/ci/pyjwt_1657511236979/work
pyOpenSSL @ file:///C:/b/abs_08f38zyck4/croot/pyopenssl_1690225407403/work
python-dateutil @ file:///tmp/build/80754af9/python-dateutil_1626374649649/work
pytz @ file:///C:/b/abs_19q3ljkez4/croot/pytz_1695131651401/work
requests @ file:///C:/b/abs_474vaa3x9e/croot/requests_1707355619957/work
requests-oauthlib==1.3.0
rsa @ file:///tmp/build/80754af9/rsa_1614366226499/work
scipy==1.11.4
six @ file:///tmp/build/80754af9/six_1644875935023/work
tensorboard @ file:///C:/Users/builder/adipietro/mc3/tf210/conda-bld/tensorboard_1669760968711/work/tensorboard-2.10.0-py3-none-any.whl
tensorboard-data-server @ file:///C:/b/abs_2fhvpo862s/croot/tensorboard-data-server_1670853600144/work/tensorboard_data_server-0.6.1-py3-none-any.whl
tensorboard-plugin-wit @ file:///C:/tf/b/tensorboard-plugin-wit_1660162132996/work/tensorboard_plugin_wit-1.8.1-py3-none-any.whl
tensorflow==2.10.0
tensorflow-estimator @ file:///C:/Users/builder/adipietro/mc3/tf210/conda-bld/tensorflow-estimator_1669761460695/work/tensorflow_estimator-2.10.0-py2.py3-none-any.whl
tensorflow-io-gcs-filesystem==0.31.0
termcolor @ file:///C:/b/abs_16qe7jhw7n/croot/termcolor_1668084642458/work
typing_extensions @ file:///C:/b/abs_72cdotwc_6/croot/typing_extensions_1705599364138/work
tzdata @ file:///croot/python-tzdata_1690578112552/work
urllib3 @ file:///C:/b/abs_a9n398cvep/croot/urllib3_1700840517182/work
Werkzeug @ file:///C:/b/abs_3901hwqcw0/croot/werkzeug_1706210134718/work
wrapt @ file:///C:/Windows/Temp/abs_7c3dd407-1390-477a-b542-fd15df6a24085_diwiza/croots/recipe/wrapt_1657814452175/work
yarl @ file:///C:/b/abs_8bxwdyhjvp/croot/yarl_1701105248152/work
zipp @ file:///C:/b/abs_b0beoc27oa/croot/zipp_1704206963359/work
```

Рисунок 16 – Часть файла requirements.txt

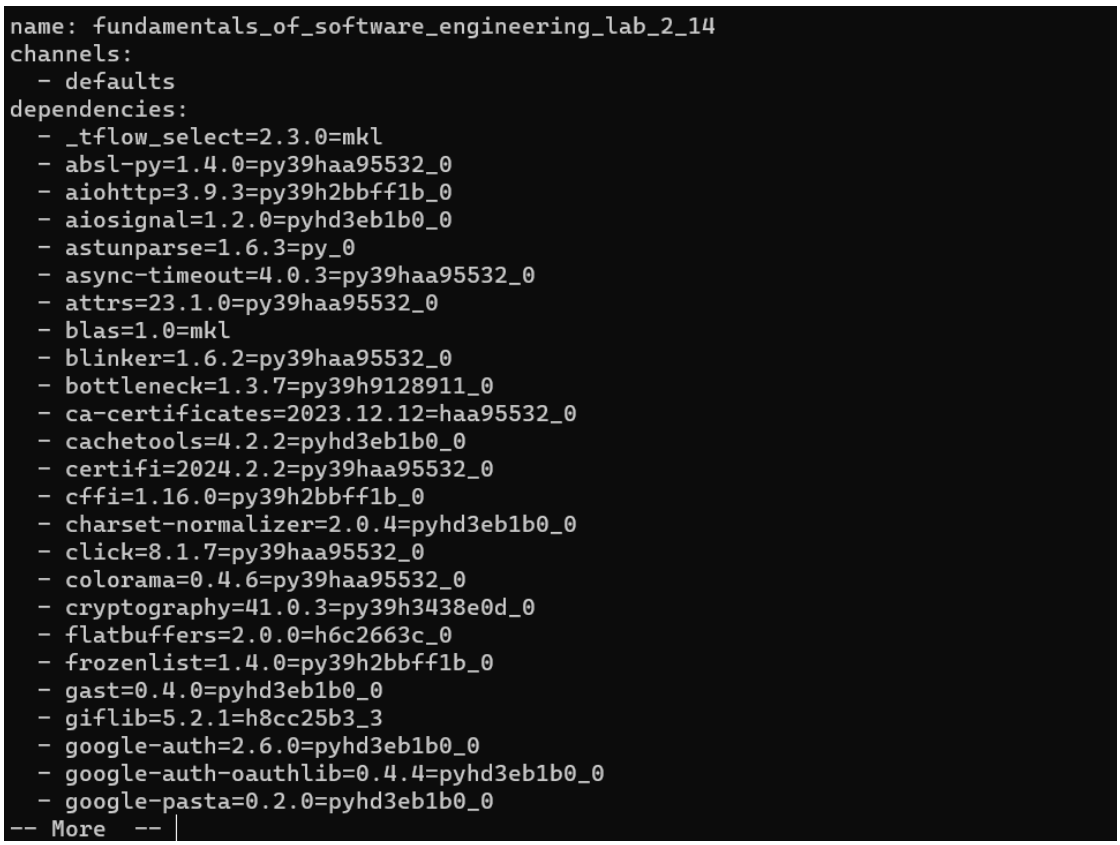

Рисунок 17 – Часть файла environments.yml

Содержимое файлов requirements.txt и environment.yml представляют собой список зависимостей для проекта, но в различных форматах, соответствующих разным менеджерам пакетов.

requirements.txt содержит список Python-пакетов с указанием их версий, кроме того некоторые пакеты указаны с URL-адресами, указывающими на локальные файлы, что означает установку из локальных источников.

environment.yml представляет собой файл YAML, определяющий среду conda. Он включает зависимости с конкретными версиями для пакетов, таких как TensorFlow, PyJWT и других.

Оба файла определяют необходимые зависимости для проекта, но в разных форматах, подходящих для различных менеджеров пакетов.

6. Сольем ветки develop и main/master и отправим изменения на удаленный репозиторий:

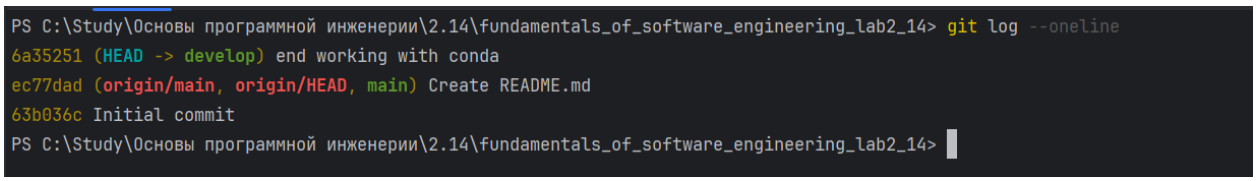

Рисунок 18 – История коммитов

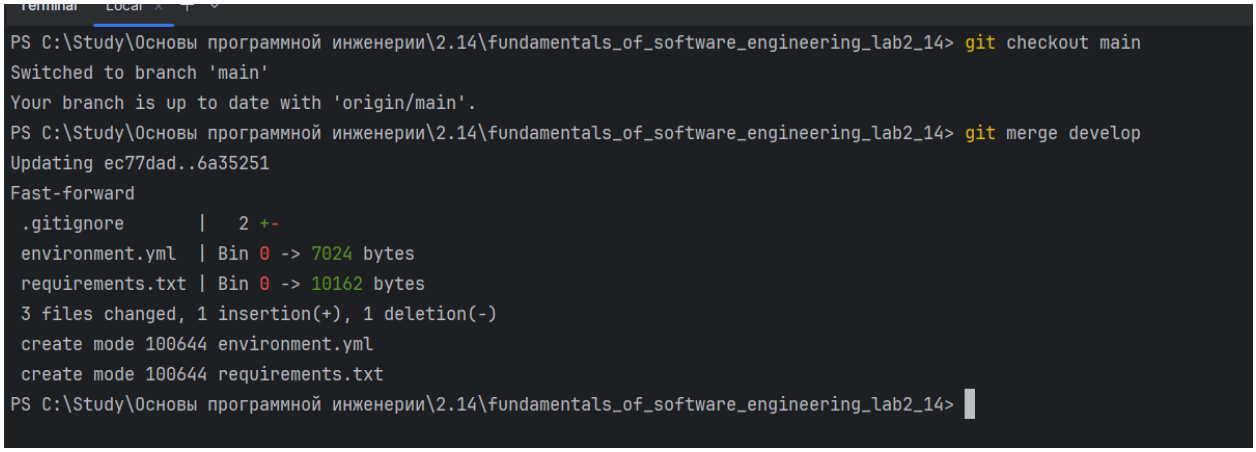

Рисунок 19 – Слияние веток

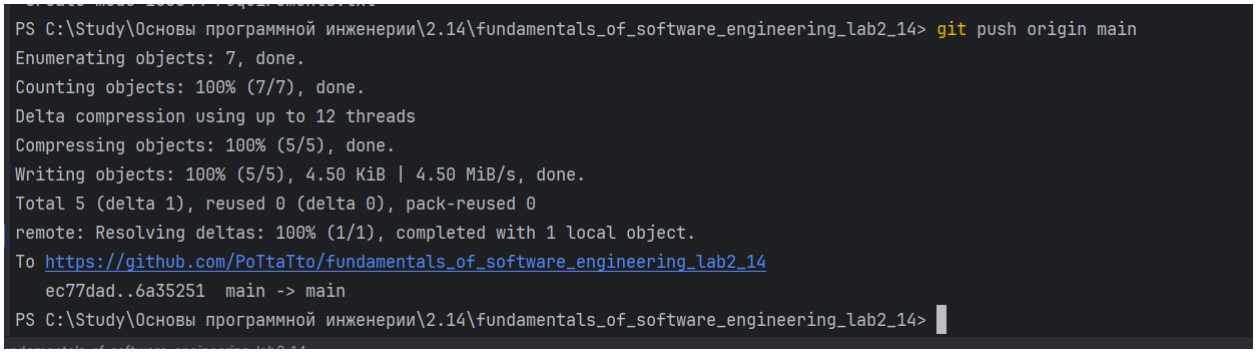

Рисунок 20 – Отправка изменений на удаленный репозиторий

Ответы на контрольные вопросы:

1. Каким способом можно установить пакет Python, не входящий в стандартную библиотеку?

Для установки пакета Python, не входящего в стандартную библиотеку, можно использовать менеджер пакетов pip, указав имя пакета после команды установки, например: pip install package_name.

2. Как осуществить установку менеджера пакетов pip?

Установку менеджера пакетов pip можно осуществить вместе с установкой Python. В более поздних версиях Python pip устанавливается автоматически. В старых версиях можно установить pip, загрузив и запустив скрипт get-pip.py: python get-pip.py.

3. Откуда менеджер пакетов pip по умолчанию устанавливает пакеты?

Менеджер пакетов pip по умолчанию устанавливает пакеты из Python Package Index (PyPI), который является крупнейшим репозиторием Python-пакетов.

4. Как установить последнюю версию пакета с помощью pip?

Для установки последней версии пакета с помощью pip следует использовать флаг --upgrade вместе с именем пакета: pip install --upgrade package_name.

5. Как установить заданную версию пакета с помощью pip?

Для установки заданной версии пакета с помощью pip следует указать конкретную версию пакета после его имени, например: pip install package_name==1.2.3.

6. Как установить пакет из git репозитория (в том числе GitHub) с помощью pip?

Для установки пакета из git репозитория с помощью pip можно использовать команду pip install git+URL, где URL - это адрес репозитория на GitHub или другом git-хостинге.

7. Как установить пакет из локальной директории с помощью pip?

Для установки пакета из локальной директории с помощью pip следует указать путь к директории после команды установки, например: pip install /path/to/package.

8. Как удалить установленный пакет с помощью pip?

Для удаления установленного пакета с помощью pip можно использовать команду pip uninstall package_name.

9. Как обновить установленный пакет с помощью pip?

Для обновления установленного пакета с помощью pip можно использовать команду pip install --upgrade package_name.

10. Как отобразить список установленных пакетов с помощью pip?

Для отображения списка установленных пакетов с помощью pip следует использовать команду pip list.

11. Каковы причины появления виртуальных окружений в языке Python?

Виртуальные окружения в языке Python используются для изоляции проекта от других проектов и системных установок, что позволяет иметь разные версии пакетов для разных проектов и избежать конфликтов зависимостей.

12. Каковы основные этапы работы с виртуальными окружениями?

Основные этапы работы с виртуальными окружениями включают создание, активацию, установку и использование пакетов в виртуальном окружении, а также деактивацию и удаление виртуального окружения.

13. Как осуществляется работа с виртуальными окружениями с помощью venv?

Работа с виртуальными окружениями с помощью venv осуществляется с помощью стандартного модуля Python venv, который позволяет создавать и управлять виртуальными окружениями.

14. Как осуществляется работа с виртуальными окружениями с помощью virtualenv?

Работа с виртуальными окружениями с помощью virtualenv осуществляется с помощью утилиты virtualenv, которая создает изолированные окружения и позволяет управлять ими.

15. Изучите работу с виртуальными окружениями pipenv. Как осуществляется работа с виртуальными окружениями pipenv?

pipenv – это инструмент для управления зависимостями и виртуальными окружениями в Python. Работа с виртуальными окружениями pipenv включает создание, активацию, установку и использование пакетов в виртуальном окружении, а также деактивацию и удаление виртуального окружения.

16. Каково назначение файла requirements.txt ? Как создать этот файл? Какой он имеет формат?

Файл requirements.txt используется для указания списка зависимостей проекта. Создать этот файл можно вручную, просто перечислив имена пакетов с версиями или используя команду pip freeze > requirements.txt. Формат файла - каждый пакет с указанием имени и версии на отдельной строке.

17. В чем преимущества пакетного менеджера conda по сравнению с пакетным менеджером pip?

Основные преимущества пакетного менеджера conda по сравнению с pip включают управление зависимостями более сложных пакетов, включая не только Python, но и бинарные зависимости, а также возможность создания изолированных сред с помощью виртуальных окружений.

18. В какие дистрибутивы Python входит пакетный менеджер conda?

Пакетный менеджер conda входит в дистрибутив Anaconda и Miniconda.

19. Как создать виртуальное окружение conda?

Для создания виртуального окружения conda следует использовать команду conda create --n myenv.

20. Как активировать и установить пакеты в виртуальное окружение conda?

Для активации и установки пакетов в виртуальное окружение conda следует использовать команду conda activate myenv для активации окружения, а затем устанавливать пакеты с помощью conda install package_name.

21. Как деактивировать и удалить виртуальное окружение conda?

Для деактивации и удаления виртуального окружения conda следует использовать команды conda deactivate для деактивации окружения и conda env remove --name myenv для удаления окружения.

22. Каково назначение файла environment.yml ? Как создать этот файл?

Файл environment.yml используется для определения среды conda. Создать этот файл можно вручную, перечислив зависимости, или сгенерировать автоматически с помощью команды conda env export > environment.yml. Формат файла - YAML секция dependencies с указанием пакетов и их версий.

23. Как создать виртуальное окружение conda с помощью файла environment.yml?

Для создания виртуального окружения conda с помощью файла environment.yml следует использовать команду conda env create -f environment.yml.

24. Самостоятельно изучите средства IDE PyCharm для работы с виртуальными окружениями conda. Опишите порядок работы с виртуальными окружениями conda в IDE PyCharm.

В IDE PyCharm работа с виртуальными окружениями conda осуществляется с помощью встроенных инструментов. Для этого следует перейти в настройки проекта, выбрать интерпретатор Python и указать путь к интерпретатору из виртуального окружения conda.

25. Почему файлы requirements.txt и environment.yml должны храниться в репозитории git?

Файлы requirements.txt и environment.yml должны храниться в репозитории git, чтобы другие разработчики могли легко создать ту же среду и установить необходимые зависимости для работы с проектом. Это также обеспечивает воспроизводимость среды и управление зависимостями.