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 №17
дисциплины «Основы программной инженерии»

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

_____
(подпись)

Проверил Воронкин Роман Александрович
_____
(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 версии 3.x.

Ход работы.

1. Создание нового репозитория с лицензией MIT.

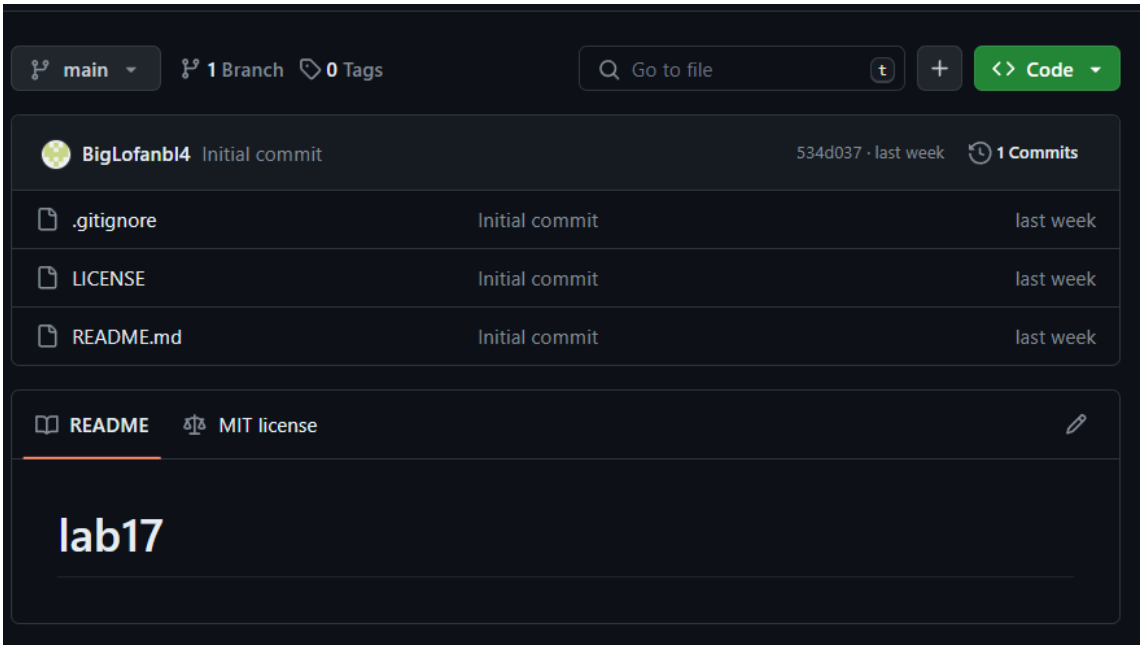

Рисунок 1.1 – Созданный репозиторий

2. Создание и активация виртуального окружения Anaconda с именем репозитория.

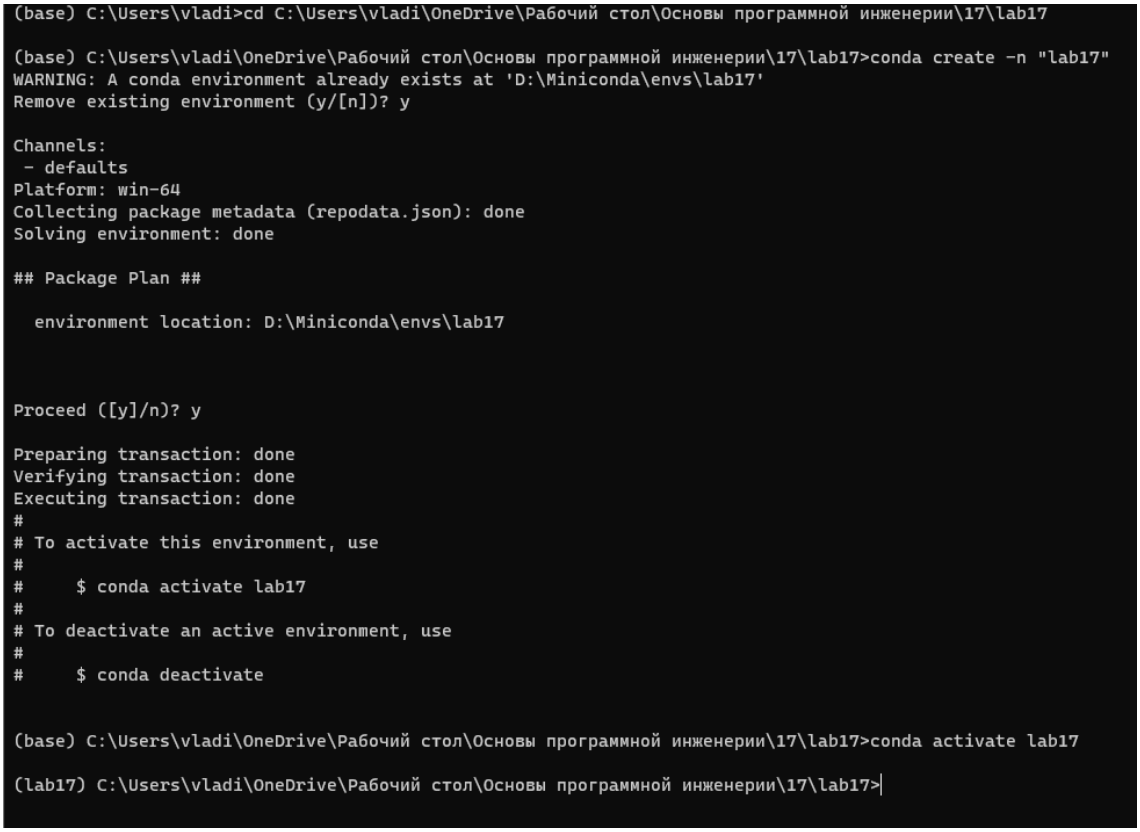

Рисунок 2.1 – Создание виртуального окружения

3. Установка в виртуальное окружение пакетов: pip, NumPy, Pandas, SciPy.

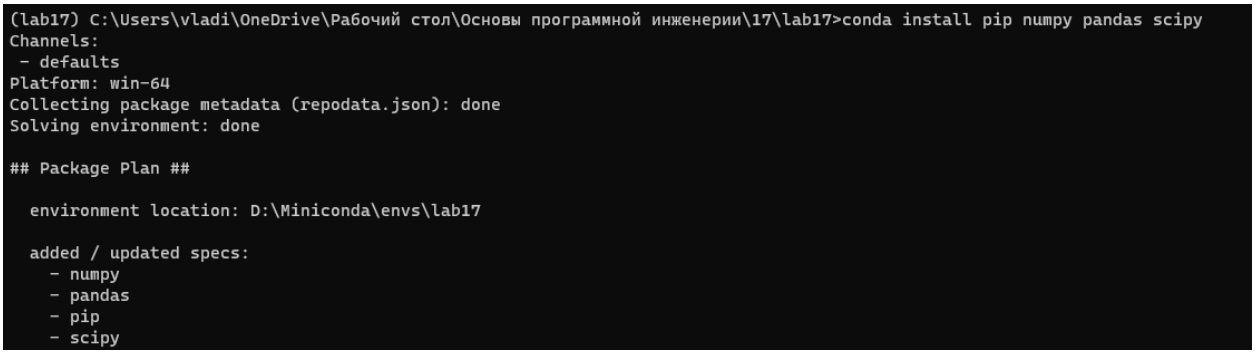

Рисунок 3.1 – Установка требуемых пакетов

4. Установка пакета TensorFlow с помощью conda и pip.

```
(lab17) C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\17\lab17>conda install tensorflow
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: D:\Miniconda\envs\lab17

  added / updated specs:
    - tensorflow


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    _tflow_select-2.2.0        |              eigen           3 KB
    absl-py-1.3.0              |   py37haa95532_0         170 KB
    aiohttp-3.8.3              |   py37h2bbff1b_0         411 KB
```

Рисунок 4.1 – Установка TensorFlow с помощью conda

```
(lab17) C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\17\lab17>pip install tensorflow
Requirement already satisfied: tensorflow in d:\miniconda\envs\lab17\lib\site-packages (2.10.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in d:\miniconda\envs\lab17\lib\site-packages (from tensorflow) (1.1.2)
Collecting protobuf<3.20,>=3.9.2
  Downloading protobuf-3.19.6-cp37-cp37m-win_amd64.whl (896 kB)
     ---------------------------------------- 896.6/896.6 kB 3.3 MB/s eta 0:00:00
```

Рисунок 4.2 – Установка TensorFlow с помощью pip

5. Формировка файлов requirements.txt и environment.yml.

```
(lab17) C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\17\lab17>conda env export > environment.yml
```

Рисунок 5.1 – Формирование environment.yml

```yaml
name: lab17
channels:
  - defaults
dependencies:
  - _tflow_select=2.2.0=eigen
  - absl-py=1.3.0=py37haa95532_0
  - aiohttp=3.8.3=py37h2bbff1b_0
  - aiosignal=1.2.0=pyhd3eb1b0_0
  - astunparse=1.6.3=py_0
  - async-timeout=4.0.2=py37haa95532_0
  - asynctest=0.13.0=py_0
  - attrs=22.1.0=py37haa95532_0
  - blas=1.0=mkl
  - blinker=1.4=py37haa95532_0
  - bottleneck=1.3.5=py37h080aedc_0
  - brotlipy=0.7.0=py37h2bbff1b_1003
  - ca-certificates=2023.12.12=haa95532_0
  - cachetools=4.2.2=pyhd3eb1b0_0
  - certifi=2022.12.7=py37haa95532_0
  - cffi=1.15.1=py37h2bbff1b_3
  - charset-normalizer=2.0.4=pyhd3eb1b0_0
  - click=8.0.4=py37haa95532_0
  - colorama=0.4.6=py37haa95532_0
  - cryptography=39.0.1=py37h21b164f_0
  - fftw=3.3.9=h2bbff1b_1
  - flatbuffers=2.0.0=h6c2663c_0
  - flit-core=3.6.0=pyhd3eb1b0_0
  - frozenlist=1.3.3=py37h2bbff1b_0
  - gast=0.4.0=pyhd3eb1b0_0
  - giflib=5.2.1=h8cc25b3_3
```

Рисунок 5.2 – Полученный файл

```
>pip list --format=freeze > requirements.txt
```

Рисунок 5.3 – Формирование requirements.txt

```
absl-py==1.3.0
aiohttp==3.8.3
aiosignal==1.2.0
astunparse==1.6.3
async-timeout==4.0.2
asynctest==0.13.0
attrs==22.1.0
blinker==1.4
Bottleneck==1.3.5
brotlipy==0.7.0
cachetools==4.2.2
certifi==2022.12.7
cffi==1.15.1
charset-normalizer==2.0.4
click==8.0.4
colorama==0.4.6
```

Рисунок 5.4 – Полученный файл

Ответы на контрольные вопросы

Контрольные вопросы:

1. Для установки пакета Python, не входящего в стандартную библиотеку, можно использовать менеджер пакетов pip, выполнив команду «pip install имя_пакета».

2. Установить менеджер пакетов pip можно, скачав get-pip.py и выполнив его с помощью Python: «python get-pip.py».

3. По умолчанию менеджер пакетов pip устанавливает пакеты из Python Package Index (PyPI).

4. Для установки последней версии пакета с помощью pip можно использовать команду «pip install имя_пакета».

5. Для установки заданной версии пакета с помощью pip можно использовать команду «pip install имя_пакета==версия».

6. Для установки пакета из git репозитория с помощью pip можно использовать команду «pip install git+https://github.com/пользователь/репозиторий.git».

7. Для установки пакета из локальной директории с помощью pip можно использовать команду «pip install ./директория».

8. Для удаления установленного пакета с помощью pip можно использовать команду «pip uninstall имя_пакета».

9. Для обновления установленного пакета с помощью pip можно использовать команду «pip install --upgrade имя_пакета».

10. Для отображения списка установленных пакетов с помощью pip можно использовать команду «pip list».

11. Виртуальные окружения в Python позволяют изолировать зависимости проекта, предотвращая конфликты между различными проектами и версиями пакетов.

12. Основные этапы работы с виртуальными окружениями: создание виртуального окружения, его активация, установка зависимостей внутри окружения, деактивация.

13. Для работы с виртуальными окружениями с помощью venv: необходимо создать окружение командой «python -m venv имя_окружения», активировать его («source имя_окружения/bin/activate» на Linux/MacOS или «имя_окружения\Scripts\activate» на Windows), установить пакеты и деактивировать («deactivate»).

14. Работа с виртуальными окружениями с помощью virtualenv аналогична venv, но требует предварительной установки virtualenv («pip install virtualenv»).

15. Работа с виртуальными окружениями pipenv включает в себя:

- создание виртуальной среды с помощью команды «pipenv shell»;
- установки необходимых пакетов с помощью команды «pipenv install имя_пакета», после установки создаются два файла «Pipfile» «Pipfile.lock»;
- с помощью команды «pipenv lock» обновляем файл «Pipfile.lock» («Pipenv» берет информацию о зависимостях, указанных в «Pipfile», и создает

«закрепленный» список этих зависимостей), файл «Pipfile.lock» позволяет иметь одну и ту же конфигурацию зависимостей;

– деактивируем виртуальное окружение с помощью команды «exit»l;

– для создание виртуального окружения в текущей директории, копируем файл «Pipfile» и используем команду «pipenv install».

16. Файл requirements.txt используется для указания зависимостей проекта. Создать его можно командой «pip freeze > requirements.txt». Формат: одна зависимость на строку, с указанием версии («пакет==версия»).

17. Преимущества conda перед pip включают управление не только Python-пакетами, но и бинарными зависимостями и окружениями, а также поддержку пакетов для разных языков.

18. Пакетный менеджер conda входит в дистрибутивы Python Anaconda и Miniconda.

19. Создать виртуальное окружение conda можно командой «conda create --name имя_окружения пакеты».

20. Активировать виртуальное окружение conda можно командой «conda activate имя_окружения», установить пакеты - «conda install пакеты».

21. Деактивировать виртуальное окружение conda можно командой «conda deactivate», удалить - «conda remove --name имя_окружения --all».

22. Файл environment.yml используется для определения окружения conda, включая зависимости. Создать его можно вручную, указав имя окружения и зависимости.

23. Создать виртуальное окружение conda с помощью файла environment.yml можно командой «conda env create -f environment.yml».

24. Работа с виртуальными окружениями conda в IDE PyCharm включает создание или выбор существующего окружения conda при настройке проекта, а также управление зависимостями через PyCharm.

25. Файлы requirements.txt и environment.yml должны храниться в репозитории git для обеспечения воспроизводимости среды разработки и

зависимостей проекта среди разработчиков и в различных средах развертывания.