

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 ПРАКТИЧЕСКОЙ РАБОТЕ №17
дисциплины «Программирование на Python»
Вариант 23

Выполнил:
Мотовилов Вадим Борисович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Информатика и вычислительная
техника», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Порядок выполнения работы:

1. Создал репозиторий и скопировал его

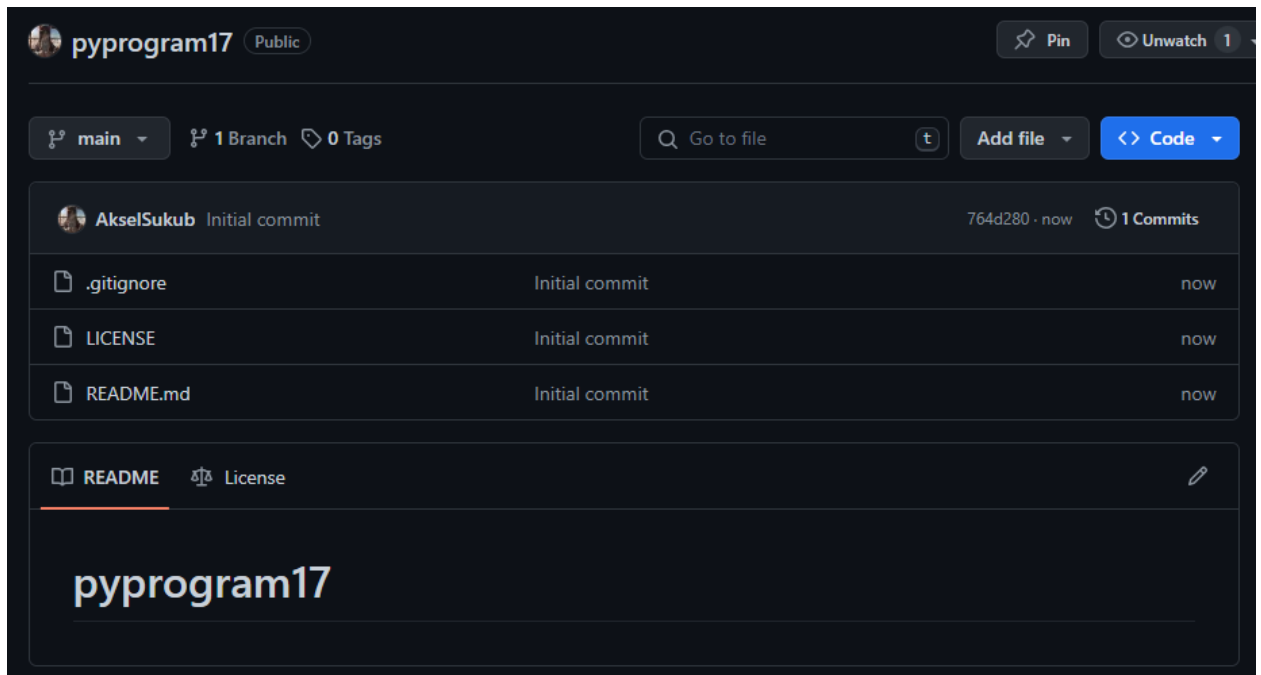


Рисунок 1. Созданный репозиторий

2. Изменил файл .gitignore и README.rm и добавил git flow

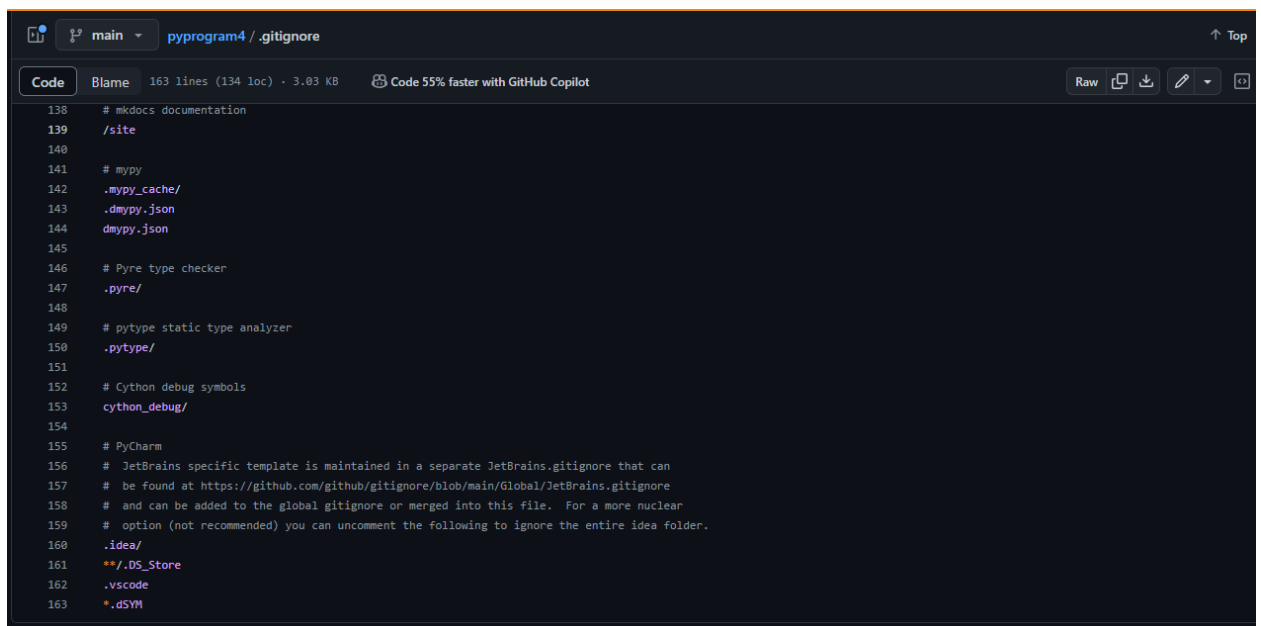


Рисунок 2. Измененный файл .gitignore

3. Создал виртуальное окружение Anaconda и установил необходимые пакеты

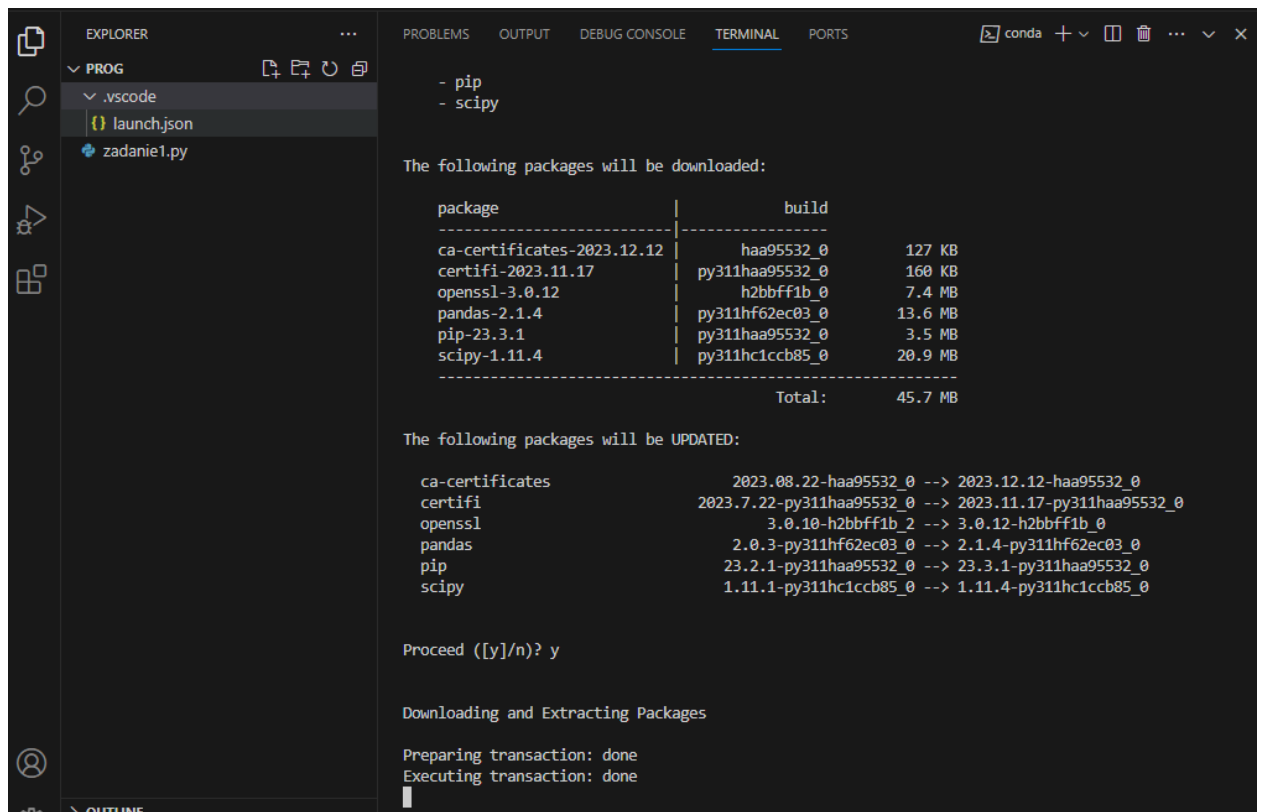
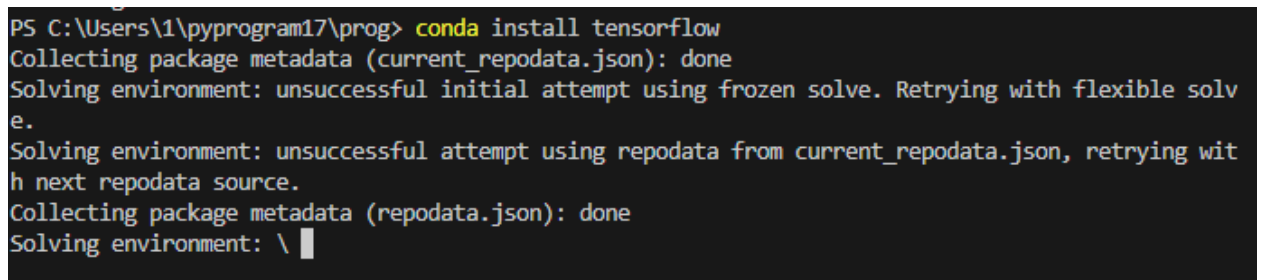


Рисунок 3 . Установка пакетов



```

Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\1\anaconda3\lib\site-packages (from
werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\1\anaconda3\lib\site-packages (fr
om pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.
0->tensorflow) (0.4.8)
Collecting oauthlib>=3.0.0 (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorbo
ard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow)
  Downloading oauthlib-3.2.2-py3-none-any.whl (151 kB)
----- 151.7/151.7 kB 1.1 MB/s eta 0:00:00
Downloading tensorflow-2.15.0-cp311-cp311-win_amd64.whl (2.1 kB)
Downloading tensorflow_intel-2.15.0-cp311-cp311-win_amd64.whl (300.9 MB)
----- 300.9/300.9 MB 2.3 MB/s eta 0:00:00
Downloading absl_py-2.1.0-py3-none-any.whl (133 kB)
----- 133.7/133.7 kB 2.6 MB/s eta 0:00:00
Downloading flatbuffers-23.5.26-py2.py3-none-any.whl (26 kB)
Downloading grpcio-1.60.0-cp311-cp311-win_amd64.whl (3.7 MB)
----- 3.7/3.7 MB 2.9 MB/s eta 0:00:00
Downloading keras-2.15.0-py3-none-any.whl (1.7 MB)
----- 1.7/1.7 MB 2.5 MB/s eta 0:00:00
Downloading libclang-16.0.6-py2.py3-none-win_amd64.whl (24.4 MB)
----- 24.4/24.4 MB 2.7 MB/s eta 0:00:00
Downloading ml_dtypes-0.2.0-cp311-cp311-win_amd64.whl (938 kB)
----- 938.7/938.7 kB 2.4 MB/s eta 0:00:00
Downloading tensorboard-2.15.1-py3-none-any.whl (5.5 MB)
----- 5.5/5.5 MB 2.7 MB/s eta 0:00:00
Downloading protobuf-4.23.4-cp310-abi3-win_amd64.whl (422 kB)
----- 422.5/422.5 kB 2.6 MB/s eta 0:00:00
Downloading tensorflow_estimator-2.15.0-py2.py3-none-any.whl (441 kB)
----- 442.0/442.0 kB 2.8 MB/s eta 0:00:00
Downloading termcolor-2.4.0-py3-none-any.whl (7.7 kB)
Downloading google_auth-2.26.2-py2.py3-none-any.whl (186 kB)
----- 186.5/186.5 kB 1.9 MB/s eta 0:00:00
Downloading google_auth_oauthlib-1.2.0-py2.py3-none-any.whl (24 kB)
Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Downloading cachetools-5.3.2-py3-none-any.whl (9.3 kB)
Installing collected packages: libclang, flatbuffers, termcolor, tensorflow-io-gcs-filesystem, te

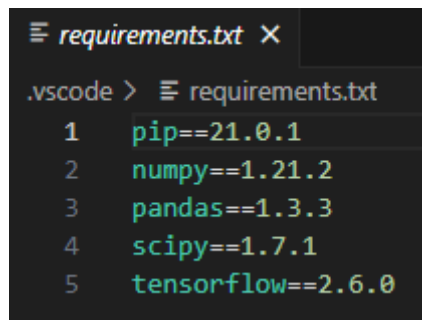
```

Рисунок 4 . Установка TensorFlow

```

! environment.yml X
.vscode > ! environment.yml
1  name: myenv
2  channels:
3  | - defaults
4  dependencies:
5  | - python=3.8
6  | - pip=21.0.1
7  | - numpy=1.21.2
8  | - pandas=1.3.3
9  | - scipy=1.7.1
10 | - tensorflow=2.6.0

```

A screenshot of a code editor window titled 'requirements.txt'. The editor shows five lines of code, each on a new line and numbered 1 through 5 on the left margin. The code is: 1 pip==21.0.1, 2 numpy==1.21.2, 3 pandas==1.3.3, 4 scipy==1.7.1, 5 tensorflow==2.6.0. The text is in a light blue font on a dark background.

```
requirements.txt X
.vscode > requirements.txt
1 pip==21.0.1
2 numpy==1.21.2
3 pandas==1.3.3
4 scipy==1.7.1
5 tensorflow==2.6.0
```

Рисунок 4 . Содержимое файлов requirements.txt и environment.yml

Ответы на контрольные вопросы:

1. Установка пакета Python, не входящего в стандартную библиотеку, обычно осуществляется с помощью инструмента для управления пакетами, такого как `pip` или `conda`. Например, чтобы установить пакет `numpy` с помощью `pip`, вы можете использовать команду `pip install numpy`.

2. Для установки `pip`, если он еще не установлен, вы можете использовать следующую команду в командной строке (для Unix/Linux):

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

```
python get-pip.py
```

Или для Windows, вы можете скачать `get-pip.py` и выполнить его, используя Python.

3. По умолчанию `pip` устанавливает пакеты из Python Package Index (PyPI), который является репозиторием для Python-пакетов.

4. Чтобы установить последнюю версию пакета с помощью `pip`, вы можете использовать команду `pip install имя_пакета`. Например, `pip install numpy` установит последнюю версию `numpy`.

5. Чтобы установить определенную версию пакета с помощью `pip`, вы можете использовать команду `pip install имя_пакета==версия`. Например, `pip install numpy==1.21.2` установит версию `numpy` 1.21.2.

6. Чтобы установить пакет из git репозитория (включая GitHub) с помощью `pip`, вы можете использовать команду `pip install git+https://github.com/username/repository.git#egg=package_name`. Замените URL на URL вашего репозитория.

7. Чтобы установить пакет из локальной директории с помощью `pip`, вы можете перейти в эту директорию и использовать команду `pip install ..`

8. Чтобы удалить установленный пакет с помощью `pip`, вы можете использовать команду `pip uninstall имя_пакета`. Например, `pip uninstall numpy` удалит `numpy`.

9. Чтобы обновить установленный пакет с помощью `pip`, вы можете использовать команду `pip install --upgrade имя_пакета`. Например, `pip install --upgrade numpy` обновит `numpy` до последней версии.

10. Чтобы отобразить список установленных пакетов с помощью `pip`, вы можете использовать команду `pip list`.

11. Виртуальные окружения используются в Python для изоляции зависимостей для конкретного проекта. Они позволяют установить и использовать разные версии пакетов для разных проектов, избегая таким образом конфликтов между ними.

12. Основные этапы работы с виртуальными окружениями включают: создание виртуального окружения, активацию виртуального окружения, установку необходимых пакетов в виртуальное окружение, работу с проектом в рамках виртуального окружения и, в конце концов, деактивацию виртуального окружения.

13. Работа с виртуальными окружениями с помощью `venv` включает следующие шаги:

Создание виртуального окружения: `python3 -m venv /path/to/new/virtual/environment`

Активация виртуального окружения: `source /path/to/new/virtual/environment/bin/activate`

Деактивация виртуального окружения: `deactivate`

14. Работа с виртуальными окружениями с помощью `virtualenv` включает следующие шаги:

Установка `virtualenv`: `pip install virtualenv`

Создание виртуального окружения: `virtualenv /path/to/new/virtual/environment`

Активация виртуального окружения: `source /path/to/new/virtual/environment/bin/activate`

Деактивация виртуального окружения: `deactivate`

15. Работа с виртуальными окружениями с помощью `pipenv` включает следующие шаги:

Установка `pipenv`: `pip install pipenv`

Создание нового проекта: `pipenv --python 3.7`

Установка пакетов: `pipenv install <package>`

Активация виртуального окружения: `pipenv shell`

Деактивация виртуального окружения: `exit`

16. Файл `requirements.txt` используется для указания зависимостей Python для проекта. Это позволяет другим разработчикам легко установить все необходимые пакеты для работы с проектом. Файл `requirements.txt` можно создать с помощью команды `pip freeze > requirements.txt`. Формат файла - это просто список пакетов для установки, возможно, с указанием версий.

17. Conda является как менеджером пакетов, так и системой управления окружением, которая может устанавливать пакеты не только для Python, но и для других языков. Conda также может устанавливать бинарные пакеты, что может быть проще, чем компиляция из исходного кода, как это иногда требуется при использовании `pip`. Кроме того, conda может управлять виртуальными окружениями, что упрощает изоляцию проектов и управление их зависимостями.

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

19. Создание виртуального окружения с помощью conda осуществляется с помощью команды `conda create --name myenv`.

20. Активация виртуального окружения и установка пакетов в нем с помощью conda осуществляется следующим образом:

- Активация: `conda activate myenv`
- Установка пакетов: `conda install package_name`

21. Деактивация и удаление виртуального окружения `conda` осуществляется следующим образом:

- Деактивация: `conda deactivate`
- Удаление: `conda env remove --name myenv`

22. Файл `environment.yml` используется для определения окружения `conda`, включая все зависимости, которые должны быть установлены в нем. Это упрощает воспроизведение окружения на других машинах. Файл `environment.yml` можно создать вручную, он имеет следующий формат:

```
name: myenv
dependencies:
- numpy
- pandas
```

Здесь `myenv` - это имя окружения, а `numpy` и `pandas` - это пакеты, которые должны быть установлены в окружении.

23. Создание виртуального окружения `conda` с помощью файла `environment.yml` осуществляется с помощью команды `conda env create -f environment.yml`.