12Министерство 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 «Программирование на python»

Выполнил:
Кожуховский Виктор Андреевич
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 Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 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.

Методика и порядок выполнения работы

1. Изучил теоретический материал работы.

2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

4. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

5. Создал виртуальное окружение Anaconda с именем репозитория.

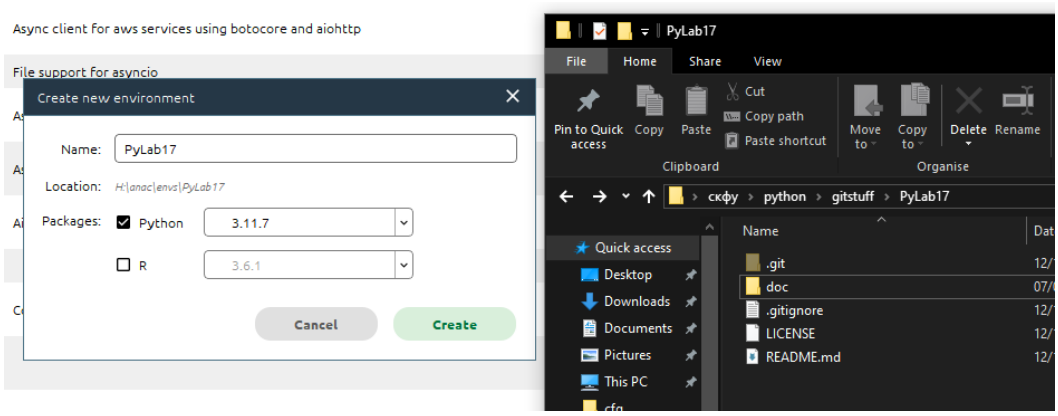

Рисунок 1. Создание виртуального окружения с именем репозитория

6. Установил в виртуальное окружение следующие пакеты: pip, NumPy, Pandas, SciPy.

```
(PyLab17) C:\Users\viktor>pip install numpy
Requirement already satisfied: numpy in c:\users\viktor\appdata\roaming\python\python311\site-packages (1.25.2)

(PyLab17) C:\Users\viktor>pip install pandas
Collecting pandas
  Downloading pandas-2.1.4-cp311-cp311-win_amd64.whl.metadata (18 kB)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\viktor\appdata\roaming\python\python311\site-packages (from pandas) (1.25.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\viktor\appdata\roaming\python\python311\site-packages (from pandas) (2.8.2)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2023.3.post1-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.1 (from pandas)
  Downloading tzdata-2023.4-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in c:\users\viktor\appdata\roaming\python\python311\site-packages (from python-dateutil>=2.8.2->pandas
) (1.16.0)
Downloading pandas-2.1.4-cp311-cp311-win_amd64.whl (10.6 MB)
   ---------------------------------------- 10.6/10.6 MB 1.1 MB/s eta 0:00:00
Downloading pytz-2023.3.post1-py2.py3-none-any.whl (502 kB)
   ---------------------------------------- 502.5/502.5 kB 1.2 MB/s eta 0:00:00
Downloading tzdata-2023.4-py2.py3-none-any.whl (346 kB)
   ---------------------------------------- 346.6/346.6 kB 1.2 MB/s eta 0:00:00
Installing collected packages: pytz, tzdata, pandas
Successfully installed pandas-2.1.4 pytz-2023.3.post1 tzdata-2023.4

(PyLab17) C:\Users\viktor>pip install scipy
Requirement already satisfied: scipy in c:\users\viktor\appdata\roaming\python\python311\site-packages (1.11.3)
Requirement already satisfied: numpy<1.28.0,>=1.21.6 in c:\users\viktor\appdata\roaming\python\python311\site-packages (from scipy) (1.25.2)
```

Рисунок 2. Установка пакетов NumPy, Pandas, SciPy

7. Попробовал установить менеджером пакетов conda пакет TensorFlow. Возникает ли при этом ошибка? Попробуйте выявить и укажите причину этой ошибки.

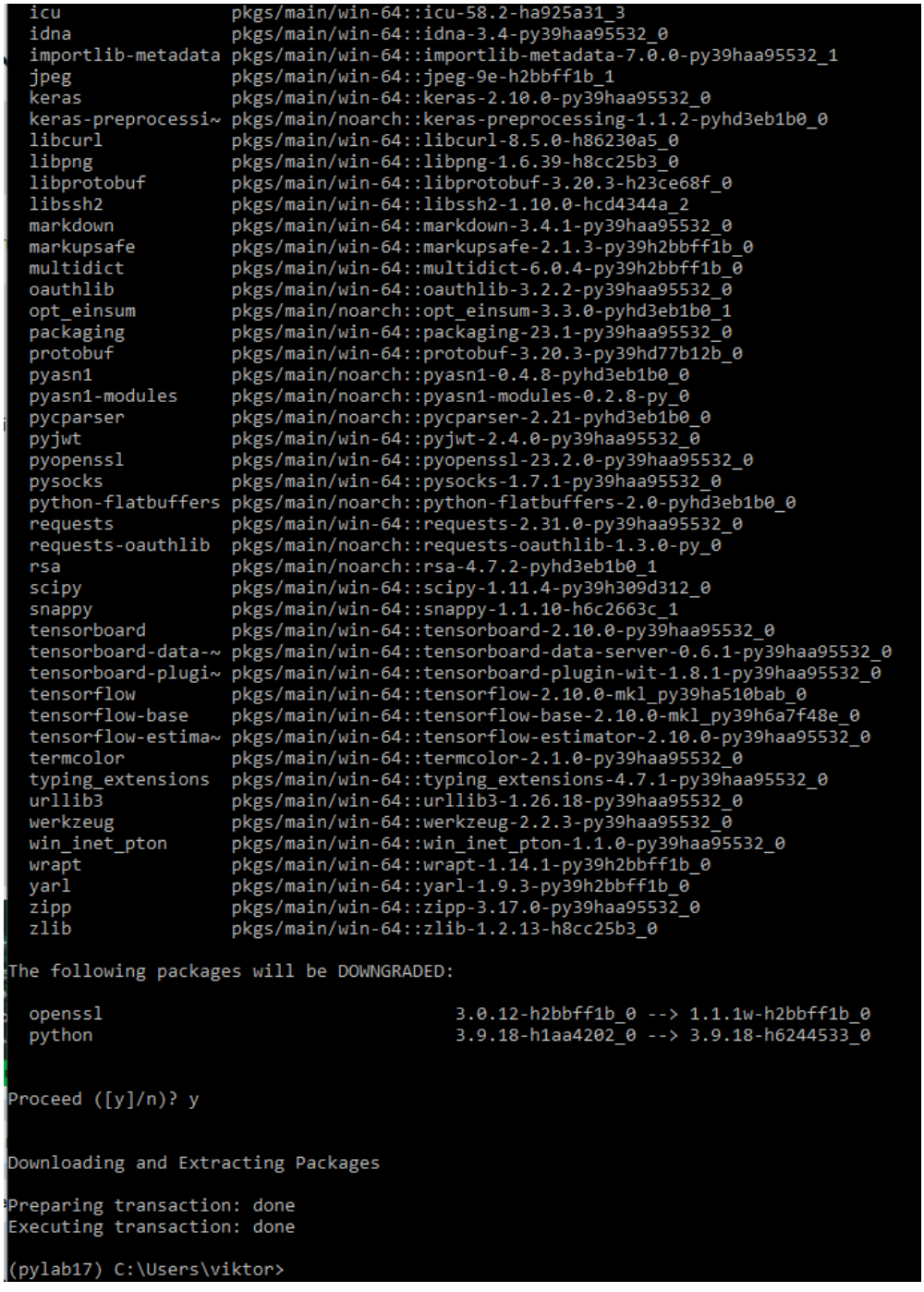

Рисунок 3. Tensorflow был успешно установлен

8. Попробовал установить пакет TensorFlow с помощью менеджера пакетов pip.

```
   ---------------------------------------- 422.5/422.5 kB 732.2 kB/s eta 0:00:00
Downloading tensorflow_estimator-2.15.0-py2.py3-none-any.whl (441 kB)
   ---------------------------------------- 442.0/442.0 kB 789.3 kB/s eta 0:00:00
Downloading termcolor-2.4.0-py3-none-any.whl (7.7 kB)
Downloading typing_extensions-4.9.0-py3-none-any.whl (32 kB)
Downloading wrapt-1.14.1-cp311-cp311-win_amd64.whl (35 kB)
Downloading google_auth-2.26.1-py2.py3-none-any.whl (186 kB)
   ---------------------------------------- 186.4/186.4 kB 863.7 kB/s eta 0:00:00
Downloading google_auth_oauthlib-1.2.0-py2.py3-none-any.whl (24 kB)
Downloading Markdown-3.5.1-py3-none-any.whl (102 kB)
   ---------------------------------------- 102.2/102.2 kB 843.2 kB/s eta 0:00:00
Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Downloading werkzeug-3.0.1-py3-none-any.whl (226 kB)
   ---------------------------------------- 226.7/226.7 kB 921.0 kB/s eta 0:00:00
Downloading cachetools-5.3.2-py3-none-any.whl (9.3 kB)
Downloading MarkupSafe-2.1.3-cp311-cp311-win_amd64.whl (17 kB)
Downloading pyasn1-0.5.1-py2.py3-none-any.whl (84 kB)
   ---------------------------------------- 84.9/84.9 kB 957.6 kB/s eta 0:00:00
Installing collected packages: libclang, flatbuffers, wrapt, typing-extensions, termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator
 tensorboard-data-server, pyasn1, protobuf, opt-einsum, oauthlib, ml-dtypes, MarkupSafe, markdown, keras, h5py, grpcio, google-pasta, gast, c
achetools, astunparse, absl-py, werkzeug, rsa, requests-oauthlib, pyasn1-modules, google-auth, google-auth-oauthlib, tensorboard, tensorflow-i
ntel, tensorflow
Successfully installed MarkupSafe-2.1.3 absl-py-2.0.0 astunparse-1.6.3 cachetools-5.3.2 flatbuffers-23.5.26 gast-0.5.4 google-auth-2.26.1 goo
gle-auth-oauthlib-1.2.0 google-pasta-0.2.0 grpcio-1.60.0 h5py-3.10.0 keras-2.15.0 libclang-16.0.6 markdown-3.5.1 ml-dtypes-0.2.0 oauthlib-3.2.
 opt-einsum-3.3.0 protobuf-4.23.4 pyasn1-0.5.1 pyasn1-modules-0.3.0 requests-oauthlib-1.3.1 rsa-4.9 tensorboard-2.15.1 tensorboard-data-serve
r-0.7.2 tensorflow-2.15.0 tensorflow-estimator-2.15.0 tensorflow-intel-2.15.0 tensorflow-io-gcs-filesystem-0.31.0 termcolor-2.4.0 typing-exten
sions-4.9.0 werkzeug-3.0.1 wrapt-1.14.1

(PyLab17) C:\Users\viktor>
```

Рисунок 4. Успешная установка TensorFlow

9. Сформировал файлы requirements.txt и environment.yml. Проанализировал содержимое этих файлов.

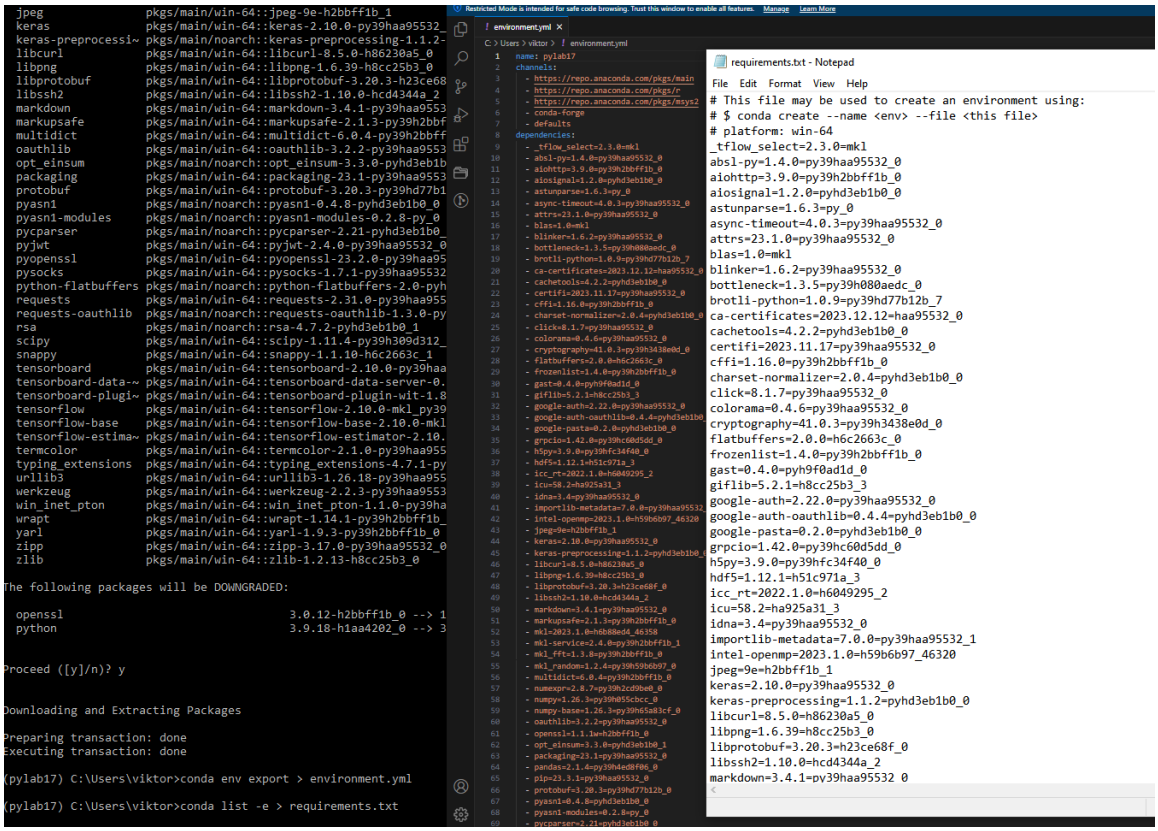

Рисунок 5. Экспортированные requirements.txt и environment.yml

10. Зафиксировал сделанные изменения в репозитории.

11. Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксировал изменения.

12. Выполнил слияние ветки для разработки с веткой master/main.

13. Отправил сделанные изменения на сервер GitHub.

Вопросы для защиты работы

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

При помощи Python Package Index (PyPI) – репозитория, открытого для всех Python разработчиков, в котором можно найти пакеты для решения практически любых задач.

2. Как осуществить установку менеджера пакетов pip?

При развертывании современной версии Python (начиная с Python 2.7.9 и Python 3.4), pip устанавливается автоматически. Но если, по какой-то причине, pip не установлен на вашем ПК, то сделать это можно вручную.

Будем считать, что Python у вас уже установлен, теперь необходимо установить pip. Для того, чтобы это сделать, скачайте скрипт get-pip.py

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

и выполните его.

$ python get-pip.py

При этом, вместе с pip будут установлены setuptools и wheels. Setuptools – это набор инструментов для построения пакетов Python. Wheels – это формат дистрибутива для пакета Python.

3. Откуда менеджер пакетов pip по умолчанию устанавливает пакеты?

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

4. Как установить последнюю версию пакета с помощью pip?

pip install ProjectName

5. Как установить заданную версию пакета с помощью pip?

pip install ProjectName==*

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

pip install -e git+https://gitrepo.com/ProjectName.git

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

pip install ./dist/ProjectName.tar.gz

8. Как удалить установленный пакет с помощью pip?

pip uninstall ProjectName

9. Как обновить установленный пакет с помощью pip?

pip install --upgrade ProjectName

10. Как отобразить список установленных пакетов с помощью pip?

pip list

11. Каковы причины появления виртуальных окружений в языке Python?

В системе для интерпретатора Python может быть установлена глобально только одна версия пакета. Это порождает ряд проблем: проблема обратной совместимости и проблема коллективной разработки. Получается, что для каждого проекта нужна своя "песочница", которая изолирует зависимости. Такая "песочница" придумана и называется "виртуальным окружением" или "виртуальной средой".

12. Каковы основные этапы работы с виртуальными окружениями?

Создаём через утилиту новое виртуальное окружение в отдельной папке для выбранной версии интерпретатора Python.

Активируем ранее созданное виртуального окружения для работы.

Работаем в виртуальном окружении, а именно управляем пакетами используя pip и запускаем выполнение кода.

Деактивируем после окончания работы виртуальное окружение.

Удаляем папку с виртуальным окружением, если оно нам больше не нужно.

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

Для создания виртуального окружения достаточно дать команду в формате: python3 -m venv <путь к папке виртуального окружения>

Обычно папку для виртуального окружения называют env или venv. В описании команды выше явно указан интерпретатор версии 3.x. Под Windows и некоторыми другими операционными системами это будет просто python.

Чтобы активировать виртуальное окружение нужно:

$ source env/bin/activate

В Windows мы вызываем скрипт активации напрямую.

> env\\Scripts\\activate

Чтобы переключиться с одного окружения на другое нам нужно выполнить команду деактивации и команду активации другого виртуального окружения, например, так:

$ deactivate

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

Для начала пакет нужно установить. Установку можно выполнить командой: # Для python 3 python3 -m pip install virtualenv # Для единственного python python -m pip install virtualenv Создание виртуального окружения с утилитой virtualenv отличается от стандартного. Например, создание в текущей папке виртуального окружения для интерпретатора доступного через команду python3 с названием папки окружения env: virtualenv -p python3 env Активация и деактивация такая же, как у стандартной утилиты Python.

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

pipenv install — Создание виртуального окружения pipenv

install — Установка определённого пакета и добавление его в Pipfile.

pipenv uninstall — Удаление установленного пакета и его исключение из Pipfile зависимостей.

pipenv shell — Активация виртуального окружения.

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

Просмотреть список зависимостей мы можем командой: pip freeze Что бы его сохранить, нужно перенаправить вывод команды в файл:

pip freeze > requirements.txt

Имя файла хранения зависимостей requirements.txt выбрано не зря. Оно является стандартной договоренностью и используется некоторыми утилитами автоматически.

Установка пакетов из файла зависимостей в новом виртуальном окружении так же выполняется одной командой: pip install -r requirements.txt

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

Основная проблема заключается в том, что pip , easy_install и virtualenv ориентированы на Python. Эти инструменты игнорируют библиотеки зависимостей, реализованные с использованием других языков. Например, XSLT , HDF5 , MKL и другие, которые не имеют setup.py в исходном коде и не устанавливают файлы в директорию site-packages. Conda же способна управлять пакетами как для Python, так и для C/ C++, R, Ruby, Lua, Scala и других. Conda устанавливает двоичные файлы, поэтому работу по компиляции пакета самостоятельно выполнять не требуется (по сравнению с pip).

Существуют также некоторые различия, если вы заинтересованы в создании собственных пакетов. Например, pip создан на основе setuptools, тогда как conda использует свой собственный формат, который имеет некоторые преимущества (например, статическая компиляция пакета).

18. В какие дистрибутивы Python входит пакетный менеджер conda?

Anaconda и Miniconda.

19. Как создать виртуальное окружение conda?

Начиная проект, создайте чистую директорию и дайте ей понятное короткое имя. Для Linux это будет соответствовать набору команд:

mkdir $PROJ_NAME

cd $PROJ_NAME

touch README.md main.py

Создайте чистое conda-окружение с таким же именем:

conda create -n $PROJ_NAME python=3.7

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

conda activate $PROJ_NAME

conda install $PACKAGE_NAME

21. Как деактивировать и удалить виртуальное окружение conda?

conda deactivate

conda remove -n $PACKAGE_NAME

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

Файл environment.yml позволит воссоздать окружение в любой нужный момент.

conda env export > environment.yml

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

Для создания виртуального окружения conda с помощью файла `environment.yml` вы должны выполнить следующие шаги:

Создайте новый файл с именем `environment.yml` или использовать существующий файл `environment.yml`, который содержит описание вашего окружения.

Откройте командную строку или терминал и перейдите в папку, где находится файл `environment.yml`.

Запустите следующую команду для создания виртуального окружения на основе файла `environment.yml`:

conda env create -f environment.yml

Эта команда попытается создать новое виртуальное окружение с именем, указанным в файле `environment.yml`.

После завершения создания виртуального окружения можно активировать его, используя команду:

conda activate <имя_окружения>

где `<имя_окружения>` - это имя виртуального окружения, которое вы указали в файле `environment.yml`.

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

Необходимо установить Anaconda или Miniconda.

В Pycharm необходимо настроить интерпретатор Python:

Нужно перейти в File > Settings (для Windows/Linux) или PyCharm > Preferences (для macOS).

В левой части окна настроек выбрать Project: ваш_проект > Python Interpreter.

Нажать на шестерёнку справа от списка интерпретаторов и выбрать Add.

В открывшемся окне добавления интерпретатора выбрать Conda Environment.

Можно либо создать новое окружение, выбрав New environment, либо использовать существующее, выбрав Existing environment.

Создание нового окружения Conda:

Необходимо указать имя окружения, версию Python и нажать кнопку OK.

PyCharm автоматически создаст новое окружение Conda и установит в него выбранную версию Python.

Использование существующего окружения Conda: Нужно нажать на кнопку с тремя точками и найти путь к существующему окружению Conda.

Активация окружения Conda:

При использовании терминала в PyCharm окружение Conda должно активироваться автоматически. Если этого не произошло, его можно

активировать вручную, введя команду conda activate имя_окружения в терминале.

Работа с проектом: После настройки окружения Conda можно работать с проектом в PyCharm, как обычно.

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

Чтобы пользователи, которые скачивают какие-либо программы, скрипты, модули могли без проблем посмотреть, какие пакеты им нужно установить дополнительно для корректной работы. За описание о наличии каких-либо пакетов в среде как раз и отвечают файлы requirements.txt и environment.yml.