

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»**

Выполнила:  
Панюкова Ксения Юрьевна  
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:  
Воронкин Р. А., доцент кафедры  
инфокоммуникаций

---

(подпись)

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

Ставрополь, 2023 г.

# Ход работы

## 1. Изучаем теоретический материал работы

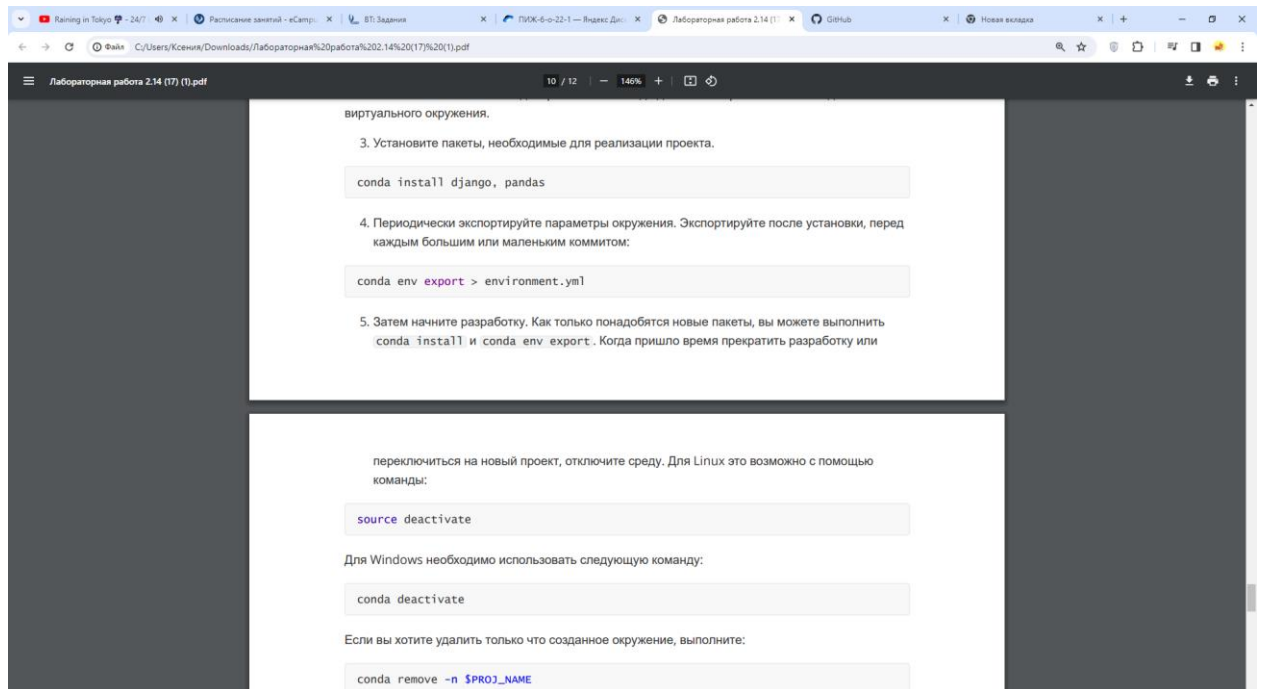


Рисунок 1.1 – Изучение материала для лабораторной работы


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

## 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 \*

 MrFincheck

Repository name \*

VirtualEnvironment

 Your new repository will be created as Virtual-nvironment.  
The repository name can only contain ASCII letters, digits, and the characters -, ., and \_.

Great repository names are short and memorable. Need inspiration? How about **fictional-octo-bassoon** ?

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


.gitignore template: Python

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

Choose a license

License: MIT License

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

 You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Настройка репозитория

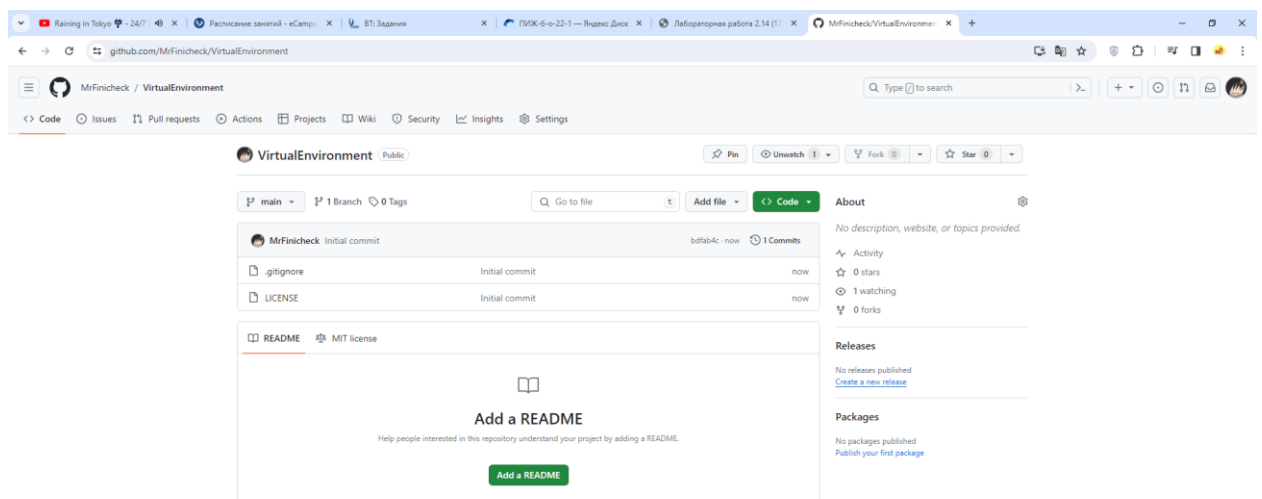


Рисунок 2.2 – Готовый репозиторий

### 3. Выполняем клонирование созданного репозитория

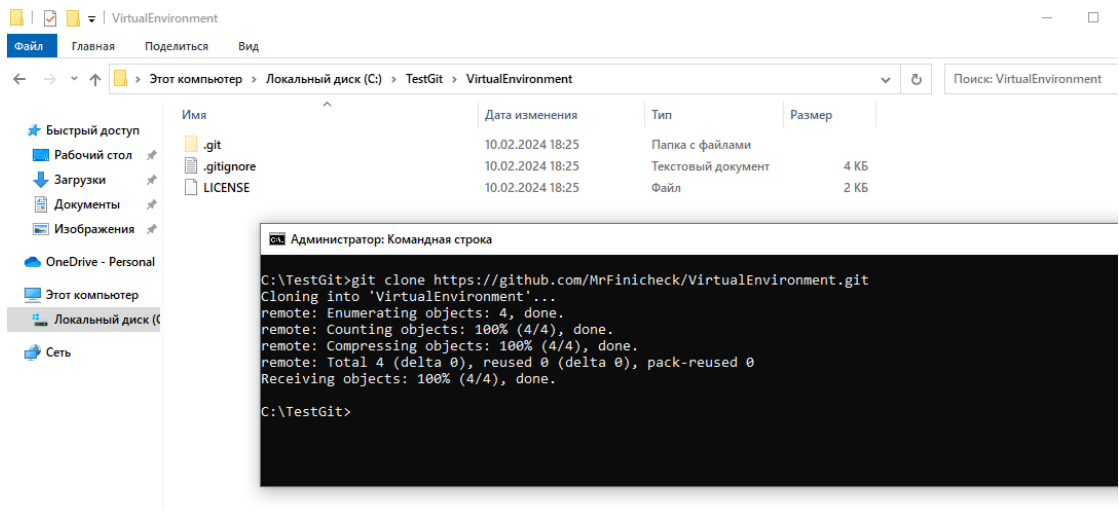


Рисунок 3.1 – Клонирование репозитория на локальный диск

#### 4. Организовываем свой репозиторий в соответствии с моделью ветвления git-flow

```

C:\TestGit\VirtualEnvironment>git branch develop
fatal: a branch named 'develop' already exists

C:\TestGit\VirtualEnvironment>git checkout develop
Already on 'develop'

C:\TestGit\VirtualEnvironment>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/MrFinich/VirtualEnvironment/pull/new/develop
remote:
To https://github.com/MrFinich/VirtualEnvironment.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.
  
```

Рисунок 4.1 – Создание ветки develop от ветки main

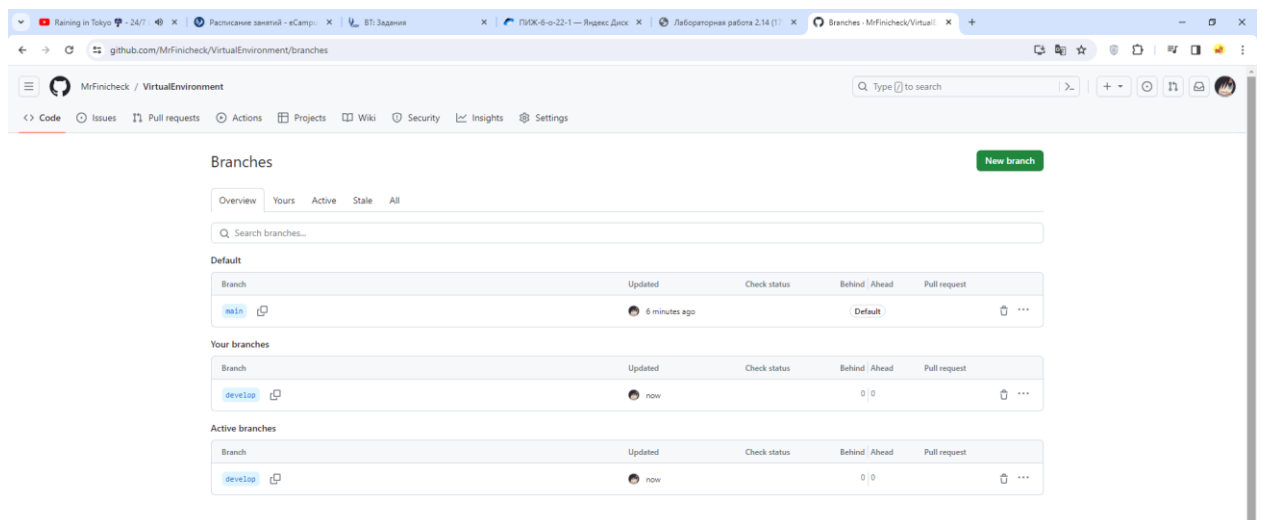


Рисунок 4.2 – Ветка develop на GitHub

5. Создаем виртуальное окружение Anaconda с именем репозитория.

```
Anaconda Prompt

(base) C:\Users\User>cd ..

(base) C:\Users>cd ..

(base) C:\>cd Sunduk

(base) C:\Sunduk>mkdir LAB17

(base) C:\Sunduk>cd LAB17
```

Рисунок 5.1 – Запуск Anaconda Powershell Prompt

```
(base) C:\Sunduk\LAB17>conda create -n %VirtualEnvironment% python=3.10
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.1.2

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

    conda install conda=24.1.2

## Package Plan ##

environment location: C:\Users\User\anaconda3\envs\python=3.10
```

Рисунок 5.2 – Создание виртуального окружения

```
The following packages will be downloaded:

package                                     build                                     127 KB
-----
ca-certificates-2023.12.12                haa95532_0
openssl-3.0.13                            h2bbff1b_0                             7.4 MB
pip-23.3.1                                py310haa95532_0                         2.9 MB
python-3.10.13                             he1021f5_0                             15.8 MB
setuptools-68.2.2                          py310haa95532_0                         942 KB
tzdata-2023d                              h04d1e81_0                             117 KB
wheel-0.41.2                               py310haa95532_0                         127 KB
xz-5.4.5                                   h8cc25b3_0                             593 KB
-----
Total:                                     28.0 MB

The following NEW packages will be INSTALLED:

bz2pkgs/main/win-64::bz2-1.0.8-he774522_0
ca-certificatespkgs/main/win-64::ca-certificates-2023.12.12-haa95532_0
libffi pkgs/main/win-64::libffi-3.4.4-hd77b12b_0
opensslpkgs/main/win-64::openssl-3.0.13-h2bbff1b_0
pippkgs/main/win-64::pip-23.3.1-py310haa95532_0
pythonpkgs/main/win-64::python-3.10.13-he1021f5_0
setuptoolspkgs/main/win-64::setuptools-68.2.2-py310haa95532_0
sqlitepkgs/main/win-64::sqlite-3.41.2-h2bbff1b_0
tkpkgs/main/win-64::tk-8.6.12-h2bbff1b_0
tzdatapkgs/main/noarch::tzdata-2023d-h04d1e81_0
vcpkgs/main/win-64::vc-14.2-h21ff451_1
```

Рисунок 5.3 – Загрузка пакетов

```
(base) C:\Sunduk\LAB17\VirtualEnvironment>conda activate VirtualEnvironment  
(VirtualEnvironment) C:\Sunduk\LAB17\VirtualEnvironment>
```

Рисунок 6.1 – Активация виртуального окружения

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

```
(base) C:\Sunduk\LAB17\VirtualEnvironment>conda activate VirtualEnvironment  
(VirtualEnvironment) C:\Sunduk\LAB17\VirtualEnvironment>conda install NumPy  
Collecting package metadata (current_repodata.json): done  
Solving environment: done  
  
==> WARNING: A newer version of conda exists. <==  
  current version: 23.7.4  
  latest version: 24.1.2  
  
Please update conda by running  
  
    $ conda update -n base -c defaults conda  
  
Or to minimize the number of packages updated during conda update use  
  
    conda install conda=24.1.2  
  
## Package Plan ##  
  
environment location: C:\Users\User\anaconda3\envs\VirtualEnvironment
```

Рисунок 6.1 – Установка пакета NumPy

```
(VirtualEnvironment) C:\Sunduk\LAB17\VirtualEnvironment>conda install Pandas  
Collecting package metadata (current_repodata.json): done  
Solving environment: done  
  
==> WARNING: A newer version of conda exists. <==  
  current version: 23.7.4  
  latest version: 24.1.2  
  
Please update conda by running  
  
    $ conda update -n base -c defaults conda  
  
Or to minimize the number of packages updated during conda update use  
  
    conda install conda=24.1.2  
  
## Package Plan ##  
  
environment location: C:\Users\User\anaconda3\envs\VirtualEnvironment
```

Рисунок 6.2 – Установка пакета Pandas

```

(VirtualEnvironment) C:\Sunduk\LAB17\VirtualEnvironment>conda install SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.1.2

Please update conda by running

  $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

  conda install conda=24.1.2

## Package Plan ##

environment location: C:\Users\User\anaconda3\envs\VirtualEnvironment

```

Рисунок 6.3 – Установка пакета SciPy

7. Попробуем установить менеджером пакетов conda пакет TensorFlow. Выявляем и указываем причину этой ошибки.

```

(VirtualEnvironment) C:\Sunduk\LAB17\VirtualEnvironment>conda install TensorFlow
Collecting package metadata (current_repodata.json): done
Solving environment: unsuccessful initial attempt using frozen solve. Retrying with flexible solve.
Solving environment: unsuccessful attempt using repodata from current_repodata.json, retrying with next repodata source
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.1.2

Please update conda by running

  $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

  conda install conda=24.1.2

## Package Plan ##

environment location: C:\Users\User\anaconda3\envs\VirtualEnvironment

```

Рисунок 7.1 – Установка пакета TensorFlow менеджером conda

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

```
(VirtualEnvironment) C:\Sunduk\LAB17\VirtualEnvironment>pip install TensorFlow
Requirement already satisfied: TensorFlow in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (2.10.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (2.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (3.9.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (1.1.2)
Collecting libclang>=13.0.0 (from TensorFlow)
  Downloading libclang-16.0.6-py2.py3-none-win_amd64.whl.metadata (5.3 kB)
Requirement already satisfied: numpy>=1.20 in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (1.26.4)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (3.3.0)
Requirement already satisfied: packaging in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (23.1)
Collecting protobuf>=3.20,>=3.9.2 (from TensorFlow)
  Downloading protobuf-3.19.6-cp310-cp310-win_amd64.whl.metadata (806 bytes)
Requirement already satisfied: setuptools in c:\users\user\anaconda3\envs\virtualenvironment\lib\site-packages (from TensorFlow) (65.5.0)
```

Рисунок 8.1 – Установка пакета TensorFlow менеджером pip

- Сформировываем файлы requirements.txt и environment.yml. Анализируем содержимое этих файлов.

```
(VirtualEnvironment) C:\Sunduk\LAB17\VirtualEnvironment>pip freeze > requirements.txt
(VirtualEnvironment) C:\Sunduk\LAB17\VirtualEnvironment>conda env export > environment.yml
```

Рисунок 9.1 – Формирование файлов

```
C:\Sunduk\LAB17\VirtualEnvironment\requirements.txt - Notepad++
1  absl-py @ file:///C:/b/abs/430v8xrl1/crook/absl-py/1.4.0/work
2  aiohttp @ file:///C:/b/abs/27h/1tqag2/crook/aiohttp/1.70.2/work
3  aiohttp @ file:///C:/b/abs/27h/1tqag2/crook/aiohttp/1.70.2/work
4  astunparse==1.6.3
5  async-timeout @ file:///C:/b/abs/08f9uixkq/crook/async-timeout/1.0.0/work
6  attrs @ file:///C:/b/abs/35d/0uac6/crook/attrs/16.0.0/work
7  blinker @ file:///C:/b/abs/08g2m7ow/crook/blinker/1.4.0/work
8  Bottleneck @ file:///C:/b/abs/05kqh7yvj/crook/bottleneck/1.0.0/work
9  cachetools @ file:///C:/b/abs/05kqh7yvj/crook/bottleneck/1.0.0/work
10 cecify @ file:///C:/b/abs/35d/0uac6/crook/cecify/1.0.0/work
11 cffi @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
12 charset-normalizer @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
13 click @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
14 colorama @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
15 cryptography @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
16 flatbuffers @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
17 frozendict @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
18 gast @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
19 google-auth @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
20 google-auth-oauthlib @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
21 google-pasta @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
22 h5py @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
23 idna @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
24 keras @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
25 keras-preprocessing @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
26 libclang==16.0.6
27 markdown @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
28 MarkupSafe @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
29 mkl-fft @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
30 mkl-random @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
31 mkl-service @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
32 multidict @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
33 numexpr @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
34 numpy @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
35 oauthlib @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
36 opt-einsum @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
37 packaging @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
38 pandas @ file:///C:/b/abs/02d9lkm3/crook/cffi/1.0.0/work
```

Рисунок 9.2 – Файл requirements.txt





Рисунок 9.3 – Файл environment.yml

## 10. Фиксируем сделанные изменения в репозитории.

```

C:\Sunduk\LAB17\VirtualEnvironment>git add .

C:\Sunduk\LAB17\VirtualEnvironment>git commit -m "add files"
[main d60fbad] add files
2 files changed, 169 insertions(+)
create mode 100644 environment.yml
create mode 100644 requirements.txt

```

Рисунок 10.1 – Коммит файлов в репозитории git

## Контрольные вопросы

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

Для скачивания и установки используется специальная утилита, которая называется `pip`.

2. Как осуществить установку менеджера пакетов `pip`?

```
pip install ProjectName
```

3. Откуда менеджер пакетов `pip` по умолчанию устанавливает пакеты?

Пакеты, установленные с помощью `pip`, обычно устанавливаются в директорию, где установлен Python.

4. Как установить последнюю версию пакета с помощью `pip`?

```
pip install ProjectName
```

5. Как установить заданную версию пакета с помощью `pip`?

```
pip install ProjectName==3.2
```

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?

Виртуальное окружение позволяет создавать изолированные среды для каждого проекта, в которых можно устанавливать и управлять зависимостями и пакетами, не вмешиваясь в другие проекты или системные установки Python.

12. Каковы основные этапы работы с виртуальными окружениями?

Основные этапы:

1. Установка инструментов;
2. Создание виртуального окружения;
3. Активация виртуального окружения;
4. Установка пакетов и зависимостей;
5. Деактивация виртуального окружения

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>`. Чтобы активировать окружение под нам нужно дать команду: `> env\\Scripts\\activate`. Чтобы переключиться с одного окружения на другое нам нужно выполнить команду деактивации и команду активации другого виртуального окружения:

`deactivate`

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

Для начала пакет нужно установить. Установку можно выполнить командой: `python3 -m pip install virtualenv`. Активация и деактивация такая же, как у стандартной утилиты Python

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

Работа с виртуальными окружениями `pipenv`:

1. Установка `pipenv`, если его нет, используя команду: `pip install pipenv`
2. Создание новое виртуальное окружение, в котором будут установлены все зависимости проекта, с помощью команды: `pipenv install`
3. Активация виртуального окружение, используя команду: `pipenv shell`
4. Установка необходимых пакетов в виртуальное окружение, например: `pipenv install requests`

5. После завершения работы с виртуальным окружением деактивация с помощью команды: `exit`

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

Файл `requirements.txt` используется для хранения списка всех зависимостей вашего проекта. Этот файл позволяет другим разработчикам или системам легко установить все необходимые зависимости для вашего проекта. Что бы его сохранить, нужно перенаправить вывод команды в файл: `pip freeze > requirements.txt`. Данный формат является обычным текстовым файлом, где указано название пакета `python`, его версия и условие, больше, меньше, равно.

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`).

18. В какие дистрибутивы `Python` входит пакетный менеджер `conda`?

Пакетный менеджер `conda` входит в дистрибутив `Anaconda`, `Miniconda` и другие связанные с ними дистрибутивы `Python`

19. Как создать виртуальное окружение `conda`?

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

```
mkdir %PROJ_NAME%  
cd %PROJ_NAME%  
copy NUL > main.py
```

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

```
conda create -n %PROJ_NAME% python=3.7 conda
```

```
activate %PROJ_NAME%
```

```
conda install django, pandas
```

21. Как деактивировать и удалить виртуальное окружение conda?

```
conda deactivate
```

```
conda remove -n $PROJ_NAME
```

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

Файл environment.yml в среде conda используется для описания зависимостей вашего проекта, включая список пакетов и их версии. Этот файл может быть использован для воссоздания точной среды с необходимыми пакетами.

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

Файл environment.yml позволит воссоздать окружение в любой нужный момент. Достаточно набрать:

```
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.

Для работы с виртуальными окружениями conda в IDE PyCharm, следуем этим шагам:

1. Открытие проекта: Откройте свой проект в IDE PyCharm.
2. Настройка интерпретатора
3. Выбор интерпретатора
4. Создание виртуального окружения: В появившемся окне выберите "Conda Environment" слева и затем "Existing environment". Укажите путь к каталогу, где хранится ваше виртуальное окружение

conda, или создайте новое виртуальное окружение, указав путь к каталогу Miniconda или Anaconda.

5. Применение изменений
6. Выбор виртуального окружения
7. Работа с виртуальным окружением

Теперь вы можете легко работать с виртуальными окружениями conda прямо из IDE PyCharm, что поможет вам эффективно управлять зависимостями и пакетами в вашем проекте.

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

Хранение файлов requirements.txt и environment.yml в репозитории Git является хорошей практикой, которая облегчает совместную работу над проектом, обеспечивает воспроизводимость окружения и упрощает управление зависимостями.