

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:  
Матвеев Александр Иванович  
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.

**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 \*** SashkaHacker / **Repository name \*** laba10

✔ laba10 is available.

Great repository names are short and memorable. Need inspiration? How about [fluffy-guide](#) ?

**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](#)

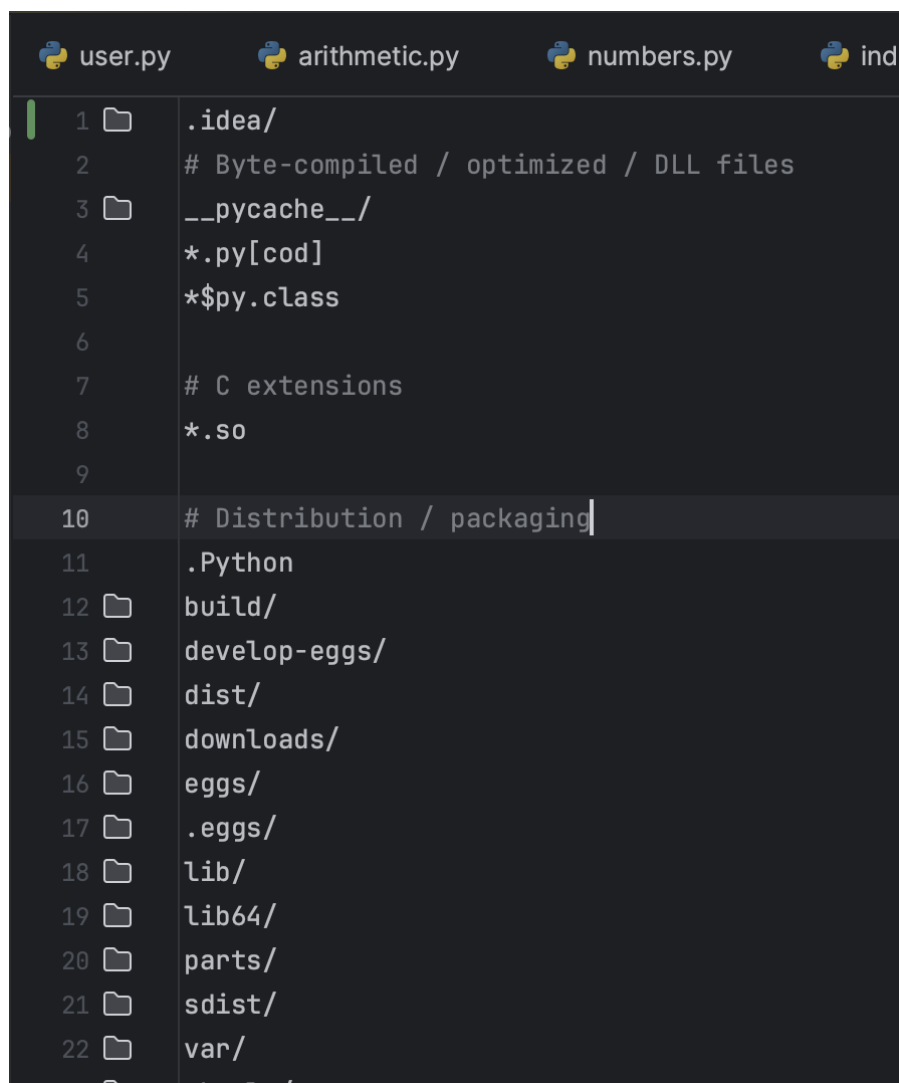
Рисунок 1 – Создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```
[→ GitHub git clone https://github.com/SashkaHacker/laba16.git
Cloning into 'laba16'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 2 – Клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.



The image shows a code editor with a dark theme. At the top, there are four tabs: 'user.py', 'arithmetic.py', 'numbers.py', and 'ind'. The active tab is 'user.py'. The editor displays the content of a '.gitignore' file. The text is as follows:

```
1  .idea/
2  # Byte-compiled / optimized / DLL files
3  __pycache__/
4  *.py[cod]
5  *$py.class
6
7  # C extensions
8  *.so
9
10 # Distribution / packaging
11 .Python
12 build/
13 develop-eggs/
14 dist/
15 downloads/
16 eggs/
17 .eggs/
18 lib/
19 lib64/
20 parts/
21 sdist/
22 var/
```

Рисунок 3 – Файл .gitignore

4. Установка miniconda.

## Quick command line install

These quick command line instructions will get you set up quickly with the latest Miniconda installer. For graphical installer (.exe and .pkg) and hash checking instructions, see [Installing Miniconda](#).

Windows

macOS

Linux

These four commands quickly and quietly install the latest M1 macOS version of the installer and then clean up after themselves. To install a different version or architecture of Miniconda for macOS, change the name of the `.sh` installer in the `curl` command.

```
mkdir -p ~/miniconda3
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-arm64.sh -o ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
```

After installing, initialize your newly-installed Miniconda. The following commands initialize for bash and zsh shells:

```
~/miniconda3/bin/conda init bash
~/miniconda3/bin/conda init zsh
```

↑ Back to top

Рисунок 4 – Команды для установки

5. Создание виртуального окружения Anaconda с именем репозитория.

```
→ laba17 git:(main) ✕ conda create -n "laba17"
Channels:
 - defaults
Platform: osx-arm64
Collecting package metadata (repodata.json): done
Solving environment: done
```

Рисунок 6 – Командная строка

6. Активация виртуального окружения, установка пакетов: `pip`, `NumPy`, `Pandas`, `SciPy`.

```
→ laba17 git:(main) ✖ conda activate laba17
```

Рисунок 7 – Активация

```
(laba17) → laba17 git:(main) ✖ conda install pip numpy pandas scipy
Channels:
  - defaults
Platform: osx-arm64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/aleksandrmatveev/miniconda3/envs/laba17

  added / updated specs:
    - numpy
    - pandas
    - pip
    - scipy
```

Рисунок 8 – Установка пакетов

#### 7. Установка tensorflow при помощи conda и при помощи pip.

```
(laba17) → laba17 git:(main) ✖ conda install tensorflow
Channels:
  - defaults
Platform: osx-arm64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/aleksandrmatveev/miniconda3/envs/laba17

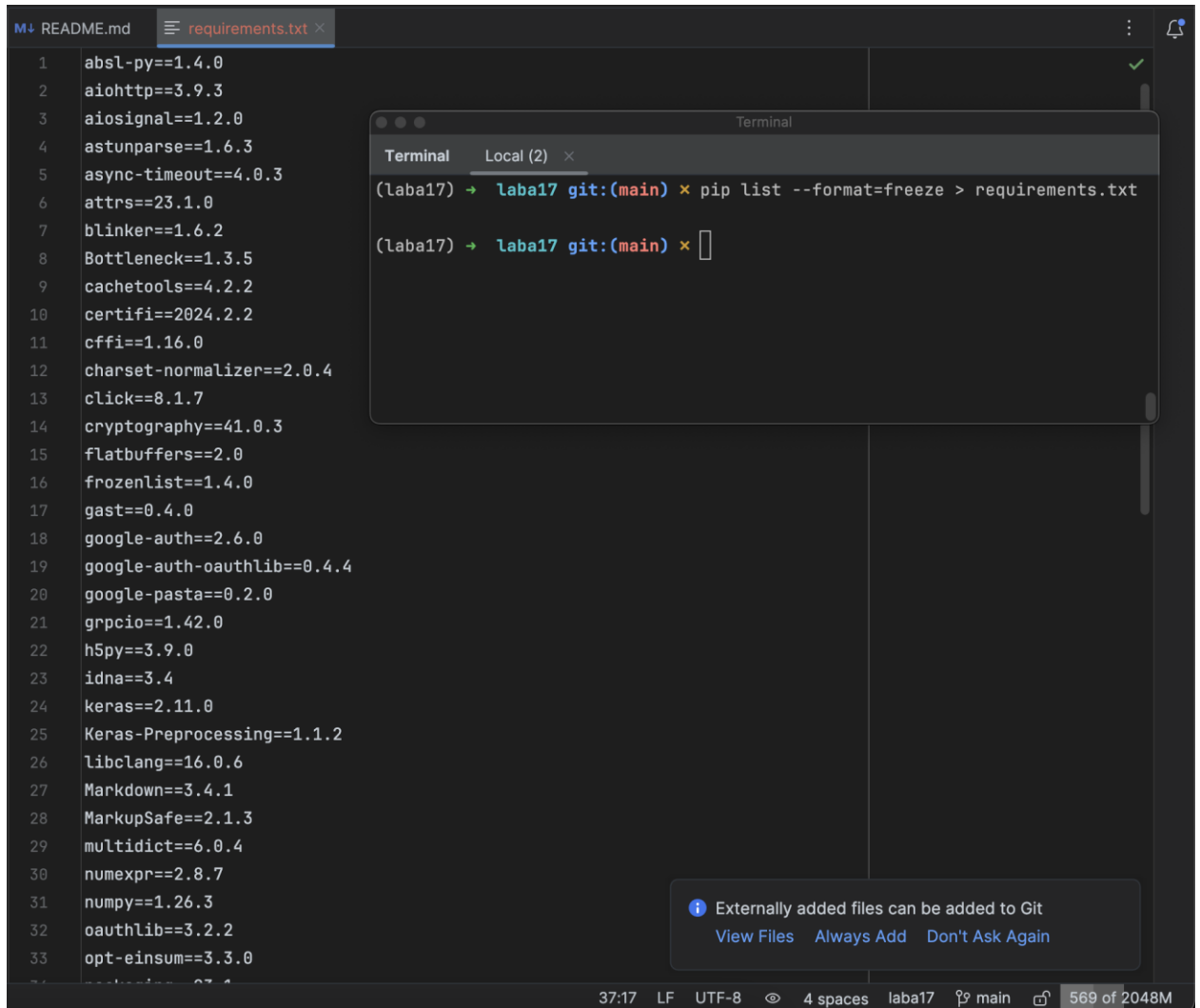
  added / updated specs:
    - tensorflow
```

Рисунок 9 – Установка при помощи conda

```
(laba17) → laba17 git:(main) × pip install tensorflow
```

Рисунок 10 – Установка при помощи pip

## 8. Формирование файлов requirement.txt и environment.yml.



The screenshot shows a code editor with a file named `requirements.txt` open. The file contains a list of Python dependencies with version constraints. A terminal window is overlaid on the editor, showing the command `pip list --format=freeze > requirements.txt` being executed. The terminal output shows the command prompt `(laba17) → laba17 git:(main) ×` followed by a blank line. A notification bubble at the bottom right of the editor states: "Externally added files can be added to Git" with buttons for "View Files", "Always Add", and "Don't Ask Again". The status bar at the bottom of the editor shows "37:17 LF UTF-8 4 spaces laba17 main 569 of 2048M".

```
1  absl-py==1.4.0
2  aiohttp==3.9.3
3  aiosignal==1.2.0
4  astunparse==1.6.3
5  async-timeout==4.0.3
6  attrs==23.1.0
7  blinker==1.6.2
8  Bottleneck==1.3.5
9  cachetools==4.2.2
10 certifi==2024.2.2
11 cffi==1.16.0
12 charset-normalizer==2.0.4
13 click==8.1.7
14 cryptography==41.0.3
15 flatbuffers==2.0
16 frozenlist==1.4.0
17 gast==0.4.0
18 google-auth==2.6.0
19 google-auth-oauthlib==0.4.4
20 google-pasta==0.2.0
21 grpcio==1.42.0
22 h5py==3.9.0
23 idna==3.4
24 keras==2.11.0
25 Keras-Preprocessing==1.1.2
26 libclang==16.0.6
27 Markdown==3.4.1
28 MarkupSafe==2.1.3
29 multidict==6.0.4
30 numexpr==2.8.7
31 numpy==1.26.3
32 oauthlib==3.2.2
33 opt-einsum==3.3.0
34 packaging==23.1
```

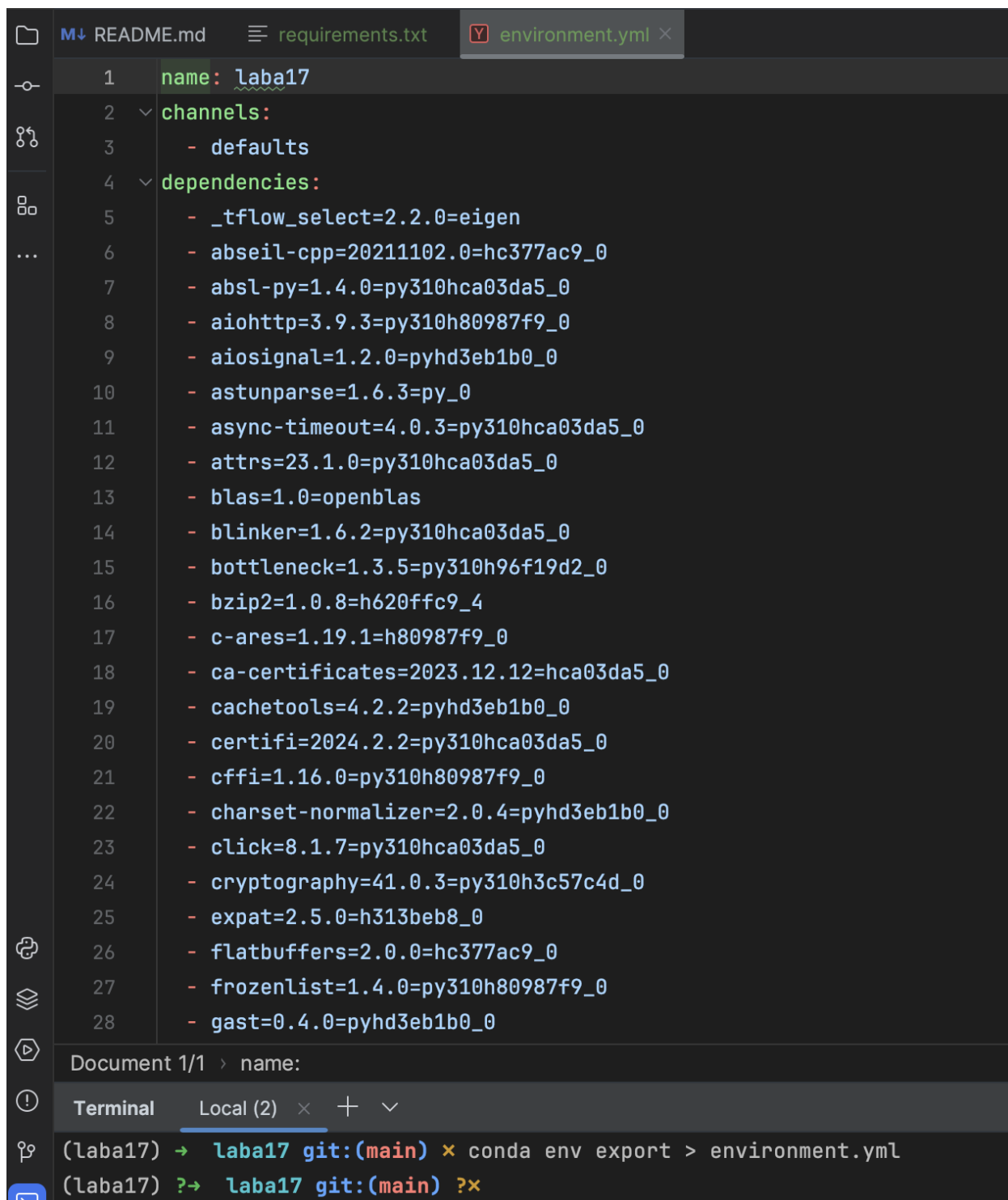
```
Terminal Local (2) ×
(laba17) → laba17 git:(main) × pip list --format=freeze > requirements.txt

(laba17) → laba17 git:(main) ×
```

Externally added files can be added to Git  
[View Files](#) [Always Add](#) [Don't Ask Again](#)

37:17 LF UTF-8 4 spaces laba17 main 569 of 2048M

Рисунок 11 – requirements.txt



The image shows a code editor with three tabs: README.md, requirements.txt, and environment.yml. The environment.yml file is open and displays a list of dependencies for a Conda environment named 'laba17'. The dependencies include various scientific and data processing libraries like tensorflow, numpy, pandas, and matplotlib. Below the code editor, a terminal window is open, showing the command 'conda env export > environment.yml' being executed in the 'laba17' environment.

```
1 name: laba17
2 channels:
3   - defaults
4 dependencies:
5   - _tfflow_select=2.2.0=eigen
6   - abseil-cpp=202111102.0=hc377ac9_0
7   - absl-py=1.4.0=py310hca03da5_0
8   - aiohttp=3.9.3=py310h80987f9_0
9   - aiosignal=1.2.0=pyhd3eb1b0_0
10  - astunparse=1.6.3=py_0
11  - async-timeout=4.0.3=py310hca03da5_0
12  - attrs=23.1.0=py310hca03da5_0
13  - blas=1.0=openblas
14  - blinker=1.6.2=py310hca03da5_0
15  - bottleneck=1.3.5=py310h96f19d2_0
16  - bzip2=1.0.8=h620ffc9_4
17  - c-ares=1.19.1=h80987f9_0
18  - ca-certificates=2023.12.12=hca03da5_0
19  - cachetools=4.2.2=pyhd3eb1b0_0
20  - certifi=2024.2.2=py310hca03da5_0
21  - cffi=1.16.0=py310h80987f9_0
22  - charset-normalizer=2.0.4=pyhd3eb1b0_0
23  - click=8.1.7=py310hca03da5_0
24  - cryptography=41.0.3=py310h3c57c4d_0
25  - expat=2.5.0=h313beb8_0
26  - flatbuffers=2.0.0=hc377ac9_0
27  - frozenlist=1.4.0=py310h80987f9_0
28  - gast=0.4.0=pyhd3eb1b0_0
```

Document 1/1 > name:

Terminal Local (2) × + ▾

(laba17) → laba17 git:(main) × conda env export > environment.yml

(laba17) ?→ laba17 git:(main) ?×

Рисунок 12 – environment.yml

### 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`: создайте окружение командой `python -m venv имя_окружения`, активируйте его (`source имя_окружения/bin/activate` на Linux/macOS или `«имя_окружения\Scripts\activate»` на Windows), устанавливайте пакеты и деактивируйте (`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 создание окружения и установку зависимостей через `pipenv install`, активацию окружения командой `pipenv shell` и установку зависимостей из `Pipfile`.

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.