

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 «Программирование на Python»**  
**Вариант 29**

Выполнил:  
Саенко Андрей Максимович  
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 Р.А., канд. технических  
наук, доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_  
Ставрополь, 2023 г.

Тема: Установка пакетов в Python. Виртуальные окружения

Цель: приобретение навыков по работе с менеджером пакетов `pip` и виртуальными окружениями с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

Выполнены задания:

1) Создайте виртуальное окружение Anaconda с именем репозитория.

```
(base) C:\Users\HAIER>cd C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17
(base) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17>mkdir PythonLab17
(base) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17>cd PythonLab17
(base) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>copy NUL > main.py
```

Рисунок 1 – Создание директории с именем репозитория

```
(base) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>conda create -n PythonLab17 python=3.11
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 23.11.0

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=23.11.0
```

Рисунок 2 – Создание окружения conda

```
(base) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>conda activate PythonLab17
(PythonLab17) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>|
```

Рисунок 3 – Активация окружения conda

2) Установите в виртуальное окружение следующие пакеты: `pip`, `NumPy`, `Pandas`, `SciPy`

```
(PythonLab17) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>conda list
# packages in environment at D:\anaconda3\envs\PythonLab17:
#
# Name                        Version      Build      Channel
blas                          1.0          mkl
bottleneck                    1.3.5        py311h5bb9823_0
bzip2                         1.0.8        he774522_0
ca-certificates               2023.12.12   haa95532_0
icc_rt                         2022.1.0     h6049295_2
intel-openmp                  2023.1.0     h59b6b97_46320
libffi                        3.4.4        hd77b12b_0
mkl                           2023.1.0     h6b88ed4_46358
mkl-service                   2.4.0        py311h2bbff1b_1
mkl_fft                       1.3.8        py311h2bbff1b_0
mkl_random                    1.2.4        py311h59b6b97_0
numexpr                        2.8.7        py311h1fcbade_0
numpy                          1.26.2       py311hdab7c0b_0
numpy-base                    1.26.2       py311hd01c5d8_0
openssl                       3.0.12       h2bbff1b_0
pandas                        2.1.4        py311hf62ec03_0
pip                            23.3.1       py311haa95532_0
python                         3.11.5       he1021f5_0
python-dateutil               2.8.2        pyhd3eb1b0_0
python-tzdata                 2023.3       pyhd3eb1b0_0
pytz                          2023.3.post1 py311haa95532_0
scipy                          1.11.4       py311hc1ccb85_0
setuptools                     68.2.2       py311haa95532_0
six                            1.16.0       pyhd3eb1b0_1
sqlite                         3.41.2       h2bbff1b_0
tbb                            2021.8.0     h59b6b97_0
tk                             8.6.12       h2bbff1b_0
tzdata                        2023c        h04d1e81_0
vc                             14.2         h21ff451_1
vs2015_runtime                14.27.29016  h5e58377_2
wheel                          0.41.2       py311haa95532_0
xz                             5.4.5        h8cc25b3_0
zlib                           1.2.13       h8cc25b3_0
```

Рисунок 4 – Список установленных пакетов

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

```
(PythonLab17) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>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: unsuccessful initial attempt using frozen solve. Retrying with flexible solve.
Solving environment: \
Found conflicts! Looking for incompatible packages.
This can take several minutes. Press CTRL-C to abort.
failed

UnsatisfiableError: The following specifications were found
to be incompatible with the existing python installation in your environment:

Specifications:

- tensorflow -> python[version='3.10.*|3.9.*|3.8.*|3.7.*|3.6.*|3.5.*']

Your python: python=3.11

If python is on the left-most side of the chain, that's the version you've asked for.
When python appears to the right, that indicates that the thing on the left is somehow
not available for the python version you are constrained to. Note that conda will not
change your python version to a different minor version unless you explicitly specify
that.
```

Рисунок 5 – Неудачная попытка установки TensorFlow

Ошибка возникла из-за слишком новой версии Python.

4) Попробуйте установить пакет TensorFlow с помощью менеджера пакетов `pip`.

```
(PythonLab17) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>pip install TensorFlow
Collecting TensorFlow
  Downloading tensorflow-2.15.0-cp311-cp311-win_amd64.whl.metadata (3.6 kB)
Collecting tensorflow-intel==2.15.0 (from TensorFlow)
  Downloading tensorflow-intel-2.15.0-cp311-cp311-win_amd64.whl.metadata (5.1 kB)
Collecting absl-py==1.0.0 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading absl_py-2.0.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse==1.6.0 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting flatbuffers==23.5.26 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading flatbuffers-23.5.26-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast==0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading gast-0.5.4-py3-none-any.whl (19 kB)
Collecting google-pasta==0.1.1 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
 57.5/57.5 kB 761.8 kB/s eta 0:00:00
Collecting h5py==2.9.0 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading h5py-3.10.0-cp311-cp311-win_amd64.whl.metadata (2.5 kB)
Collecting libclang==13.0.0 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading libclang-16.0.6-py2.py3-none-win_amd64.whl.metadata (5.3 kB)
Collecting ml-dtypes==0.2.0 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading ml_dtypes-0.2.0-cp311-cp311-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in d:\anaconda3\envs\pythonlab17\lib\site-packages (from tensorflow-intel==2.15.0->TensorFlow) (1.26.2)
Collecting opt-einsum==2.3.2 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
 65.5/65.5 kB 3.7 MB/s eta 0:00:00
Collecting packaging (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading packaging-23.2-py3-none-any.whl.metadata (3.2 kB)
Collecting protobuf!=4.21.0,!>4.21.1,!>4.21.2,!>4.21.3,!>4.21.4,!>4.21.5,<5.0.0dev,>=3.20.3 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading protobuf-4.25.1-cp310-abi3-win_amd64.whl.metadata (541 bytes)
Requirement already satisfied: setuptools in d:\anaconda3\envs\pythonlab17\lib\site-packages (from tensorflow-intel==2.15.0->TensorFlow) (68.2.2)
Requirement already satisfied: six>=1.12.0 in d:\anaconda3\envs\pythonlab17\lib\site-packages (from tensorflow-intel==2.15.0->TensorFlow) (1.16.0)
Collecting termcolor==1.1.0 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading termcolor-2.4.0-py3-none-any.whl.metadata (6.1 kB)
Collecting typing-extensions==3.6.6 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading typing_extensions-4.9.0-py3-none-any.whl.metadata (3.0 kB)
Collecting wrapt<1.15,>=1.11.0 (from tensorflow-intel==2.15.0->TensorFlow)
  Downloading wrapt-1.14.1-cp311-cp311-win_amd64.whl.metadata (6.9 kB)
```

Рисунок 6 – Установка TensorFlow с помощью менеджера пакетов `pip`

```
Installing collected packages: libclang, flatbuffers, wrapt, urllib3, typing-extensions, termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator, tensorboard-data-server, pyasn1, protobuf, packaging, opt-einsum, oauthlib, ml-dtypes, MarkupSafe, markdown, keras, idna, h5py, grpcio, google-pasta, gast, charset-normalizer, certifi, cachetools, astunparse, absl-py, werkzeug, rsa, requests, pyasn1-modules, requests-oauthlib, google-auth, google-auth-oauthlib, tensorboard, tensorflow-intel, TensorFlow
Successfully installed MarkupSafe-2.1.3 TensorFlow-2.15.0 absl-py-2.0.0 astunparse-1.6.3 cachetools-5.3.2 certifi-2023.11.17 charset-normalizer-3.3.2 flatbuffers-23.5.26 gast-0.5.4 google-auth-2.25.2 google-auth-oauthlib-1.2.0 google-pasta-0.2.0 grpcio-1.60.0 h5py-3.10.0 idna-3.6 keras-2.15.0 libclang-16.0.6 markdown-3.5.1 ml-dtypes-0.2.0 oauthlib-3.2.2 opt-einsum-3.3.0 packaging-23.2 protobuf-4.23.4 pyasn1-0.5.1 pyasn1-modules-0.3.0 requests-2.31.0 requests-oauthlib-1.3.1 rsa-4.9 tensorboard-2.15.1 tensorboard-data-server-0.7.2 tensorflow-estimator-2.15.0 tensorflow-intel-2.15.0 tensorflow-io-gcs-filesystem-0.31.0 termcolor-2.4.0 typing-extensions-4.9.0 urllib3-2.1.0 werkzeug-3.0.1 wrapt-1.14.1

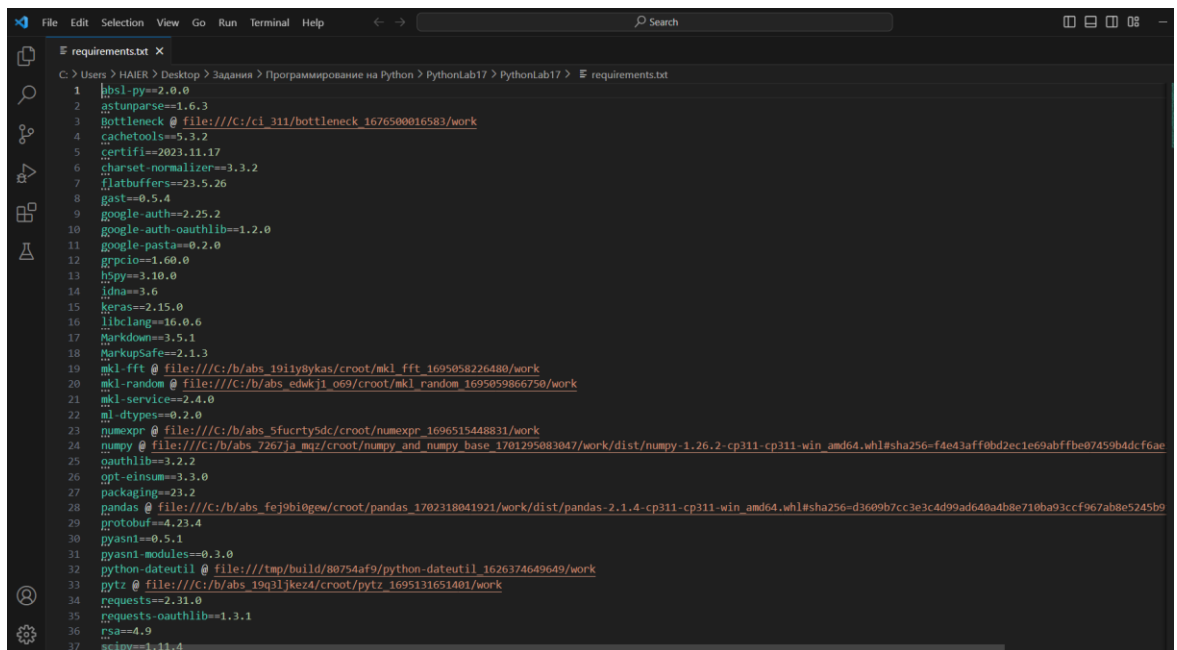
(PythonLab17) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>
```

Рисунок 7 – Успешная установка TensorFlow с помощью менеджера пакетов `pip`

5) Сформируйте файлы `requirements.txt` и `environment.yml`. Проанализируйте содержимое этих файлов.

```
(PythonLab17) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>conda env export > environment.yml
(PythonLab17) C:\Users\HAIER\Desktop\Задания\Программирование на Python\PythonLab17\PythonLab17>pip freeze > requirements.txt
```

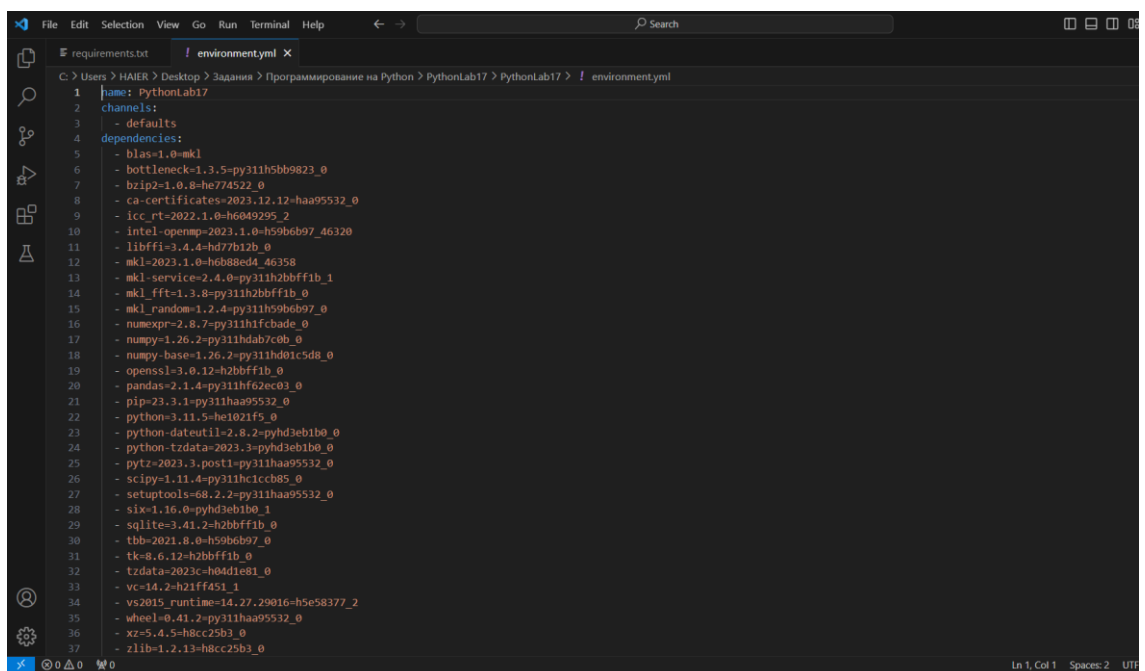
Рисунок 8 – Формирование файлов `requirements.txt` и `environment.yml`



```
1 hbsl-py==2.0.0
2 astunparse==1.6.3
3 bottleneck @ file:///C:/ci_311/bottleneck_1676508016583/work
4 cachetools==5.3.2
5 certifi==2023.11.17
6 charset-normalizer==3.3.2
7 flatbuffers==23.5.26
8 gast==0.5.4
9 google-auth==2.25.2
10 google-auth-oauthlib==1.2.0
11 google-pasta==0.2.0
12 grpcio==1.60.0
13 h5py==3.10.0
14 idna==3.6
15 keras==2.15.0
16 libclang==16.0.6
17 markdown==3.5.1
18 MarkupSafe==2.1.3
19 mkl_fft @ file:///C:/b/abs_1911y8ykas/croot/mkl_fft_1695058226480/work
20 mkl_random @ file:///C:/b/abs_edukj1_069/croot/mkl_random_1695059866750/work
21 mkl-service==2.4.0
22 ml-dtypes==0.2.0
23 numexpr @ file:///C:/b/abs_5fucrt5dc/croot/numexpr_1696515448831/work
24 numpy @ file:///C:/b/abs_7267ja_mqz/croot/numpy_and_numpy_base_1701295083047/work/dist/numpy-1.26.2-cp311-cp311-win_amd64.whl#sha256=f4e43aff0bd2ec1e69abffbe07459b4dcf6ae
25 oauthlib==3.2.2
26 opt-einsum==3.3.0
27 packaging==23.2
28 pandas @ file:///C:/b/abs_fej9bi0gew/croot/pandas_1702318041921/work/dist/pandas-2.1.4-cp311-cp311-win_amd64.whl#sha256=d3609b7cc3e3cd99ad648a4b8e710ba93ccf967ab8e5245b9
29 protobuf==4.23.4
30 pyasn1==0.5.1
31 pyasn1-modules==0.3.0
32 python-dateutil @ file:///tmp/build/80754af9/python-dateutil_1626374649649/work
33 pytz @ file:///C:/b/abs_19q3ljkez4/croot/pytz_1695131651401/work
34 requests==2.31.0
35 requests-oauthlib==1.3.1
36 rsa==4.9
37 scipy==1.11.4
```

Рисунок 9 – Содержимое файла requirements.txt

Файл requirements.txt содержит точные версии всех установленных пакетов.



```
1 name: PythonLab17
2 channels:
3   - defaults
4 dependencies:
5   - blas=1.0-mkl
6   - bottleneck=1.3.5=py311h5bb9823_0
7   - bzip2=1.0.8=he774522_0
8   - ca-certificates=2023.12.12=haa95532_0
9   - icc_rt=2022.1.0=h6049295_2
10  - intel-openmp=2023.1.0=h59b6b97_46320
11  - libffi=3.4.4=hd77b12b_0
12  - mkl=2023.1.0=h6b88ed4_46358
13  - mkl-service=2.4.0=py311h2bbff1b_1
14  - mkl_fft=1.3.8=py311h2bbff1b_0
15  - mkl_random=1.2.4=py311h59b6b97_0
16  - numexpr=2.8.7=py311h1fcbae_0
17  - numpy=1.26.2=py311hdab7c0b_0
18  - numpy-base=1.26.2=py311hd01c5d8_0
19  - openssl=3.0.12=h2bbff1b_0
20  - pandas=2.1.4=py311hf62ec03_0
21  - pip=23.3.1=py311haa95532_0
22  - python=3.11.5=he1021f5_0
23  - python-dateutil=2.8.2=pyhd3eb1b0_0
24  - python-tzdata=2023.3=pyhd3eb1b0_0
25  - pytz=2023.3.post1=py311haa95532_0
26  - scipy=1.11.4=py311hc1ccb85_0
27  - setuptools=68.2.2=py311haa95532_0
28  - six=1.16.0=pyhd3eb1b0_1
29  - sqlite=3.41.2=h2bbff1b_0
30  - tbb=2021.8.0=h59b6b97_0
31  - tk=8.6.12=h2bbff1b_0
32  - tzdata=2023c=h04d1e81_0
33  - vc=14.2=h21ffa51_1
34  - vs2015_runtime=14.27.29016=h5e58377_2
35  - wheel=0.41.2=py311haa95532_0
36  - xz=5.4.5=h8cc25b3_0
37  - zlib=1.2.13=h8cc25b3_0
```

Рисунок 10 – Содержимое файла environment.yml

Файл environment.yml выступает в качестве манифеста, описывая используемые ресурсы и расположение шаблона для определения среды.

Ответы на контрольные вопросы:

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

Для установки пакетов, не входящих в стандартную библиотеку используется менеджер пакетов `pip`. Для установки пакета вводится команда `pip install <название_пакета>`.

2. Как осуществить установку менеджера пакетов `pip`?

Если используется Python 2.7.9 (и выше) или Python 3.4 (и выше), PIP устанавливается вместе с Python по умолчанию. Для установки `pip` на Windows необходимо скачать установочный пакет, открыть командную строку, перейти в каталог со скачанным файлом и ввести команду «`python get-pip.py`». Для установки `pip` на другие ОС необходимо следовать соответствующим инструкциям по установке.

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

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

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

Для установки последней версии пакета с помощью `pip` используется следующая команда:

```
pip install --upgrade <имя_пакета>
```

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

Для установки определённой версии пакета с помощью `pip` используется следующая команда:

```
pip install <имя_пакета>==<версия_пакета>
```

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

Для установки пакета из git репозитория с помощью `pip` используется следующая команда:

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

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

Для установки пакета из git репозитория с помощью pip используется следующая команда:

```
pip install ./dist/ProjectName.tar.gz
```

8. Как удалить установленный пакет с помощью pip?

Для удаления установленного пакета с помощью pip используется следующая команда:

```
pip uninstall ProjectName
```

9. Как обновить установленный пакет с помощью pip?

Для обновления установленного пакета с помощью pip используется следующая команда:

```
pip install --upgrade ProjectName
```

10. Как отобразить список установленных пакетов с помощью pip?

Для отображения списка установленных пакетов с помощью pip используется следующая команда:

```
pip list
```

11. Каковы причины появления виртуальных окружений в языке Python?

Виртуальные окружения в Python нужны для изоляции проектов и их зависимостей.

12. Каковы основные этапы работы с виртуальными окружениями?

Основные этапы работы с виртуальными окружениями: создание, активация, деактивация и удаление

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

Для работы с виртуальными окружениями с помощью venv используется стандартная библиотека Python, с помощью которой создаются виртуальные окружения и осуществляется управление ими.

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

`Virtualenv` предоставляет инструменты для работы с виртуальными окружениями.

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

`Pipenv` – менеджер зависимостей для Python-проектов. С его помощью можно создавать виртуальные среды и управлять зависимостями приложений. `Pipenv` решает ряд проблем, которые возникали при использовании `pip`, `virtualenv` и `requirements.txt`. Данный инструмент аналогичен `npm` у NodeJS, или `composer` у PHP, и является официально рекомендуемым менеджером зависимостей для Python.

Работа с виртуальными окружениями в `pipenv` происходит с помощью командной строки и позволяет создавать отдельные изолированные окружения для каждого проекта, что позволяет избежать конфликтов между зависимостями разных проектов.

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

Файл `requirements.txt` содержит точные версии всех установленных пакетов. Создаётся этот файл с помощью команды `pip freeze > requirements.txt`.

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

`Conda` позволяет работать с пакетами, написанными не только на языке Python.

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

`Conda` входит в дистрибутивы `Anaconda` и `Miniconda`.

19. Как создать виртуальное окружение `conda`?

Для создания виртуального окружения `conda` используется следующая команда:



```
conda create -n %PROJ_NAME% python=<версия_Python>
```

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

Для активации окружения conda используется команда `conda activate <Название>`. Для установки пакетов используется команда `conda install <название пакета>`.

21. Как деактивировать и удалить виртуальное окружение conda?

Для деактивации виртуального окружения conda используется команда `conda deactivate`.

Для удаления виртуального окружения conda используется команда `conda remove -n $PROJ_NAME`.

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

Файл `environment.yml` позволит воссоздать окружение в любой нужный момент. Для его создания используется команда `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` используется следующая команда:

```
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.

Создавать и активировать окружения в PyCharm можно через специальный пользовательский интерфейс. Для этого должен быть установлен плагин Conda.

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

Данные файлы должны храниться в репозитории `git`, чтобы другие разработчики могли быстро воссоздать виртуальное окружение проекта.

## Вывод

В ходе выполнения работы приобретены навыки по работе с менеджером пакетов `pip` и виртуальными окружениями с помощью языка программирования Python версии 3.x.