

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 7

Выполнил:
Зармухамбетов Булат Эльдарович
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.

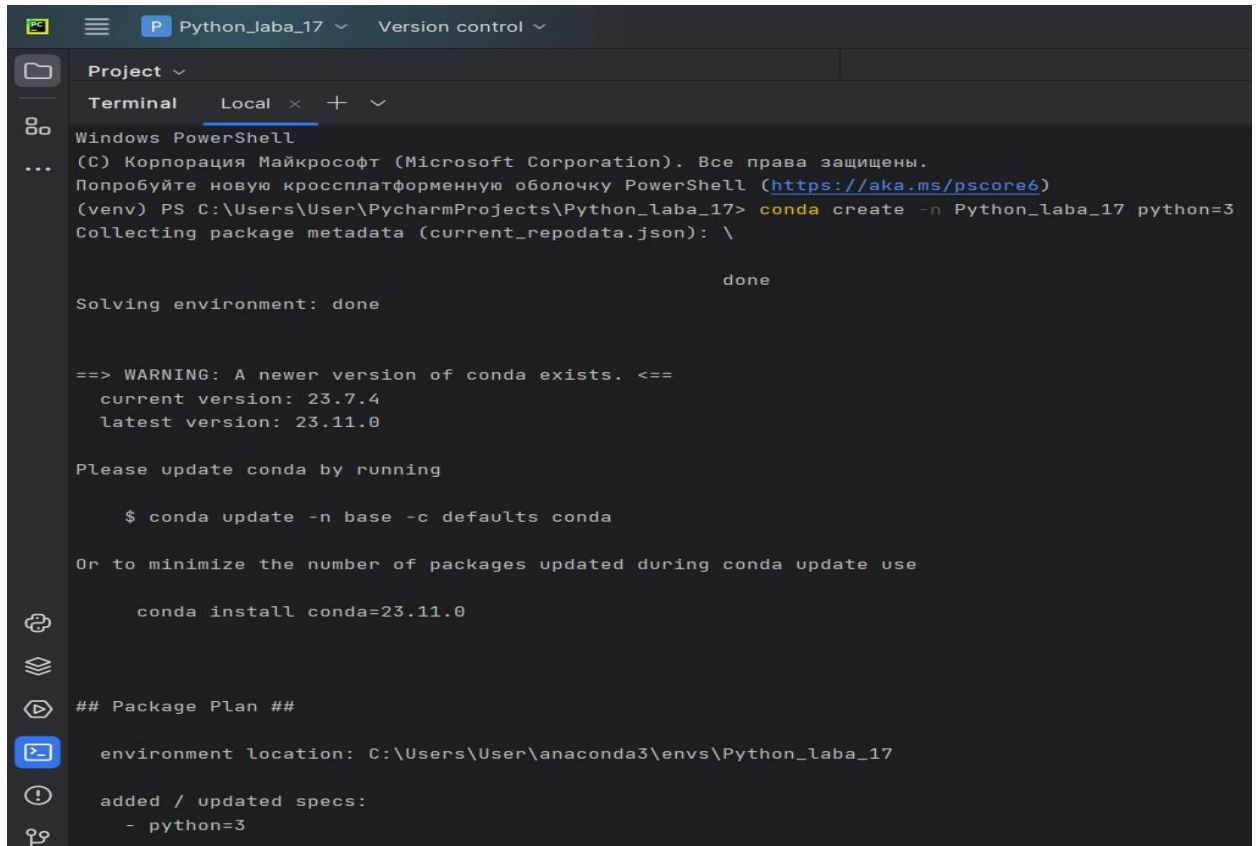
(подпись)

Отчет защищен с оценкой_____ Дата защиты_____

Ставрополь, 2023

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.



```
Python_laba_17  Version control
Project
Terminal Local x + v
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/powershell)
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> conda create -n Python_laba_17 python=3
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

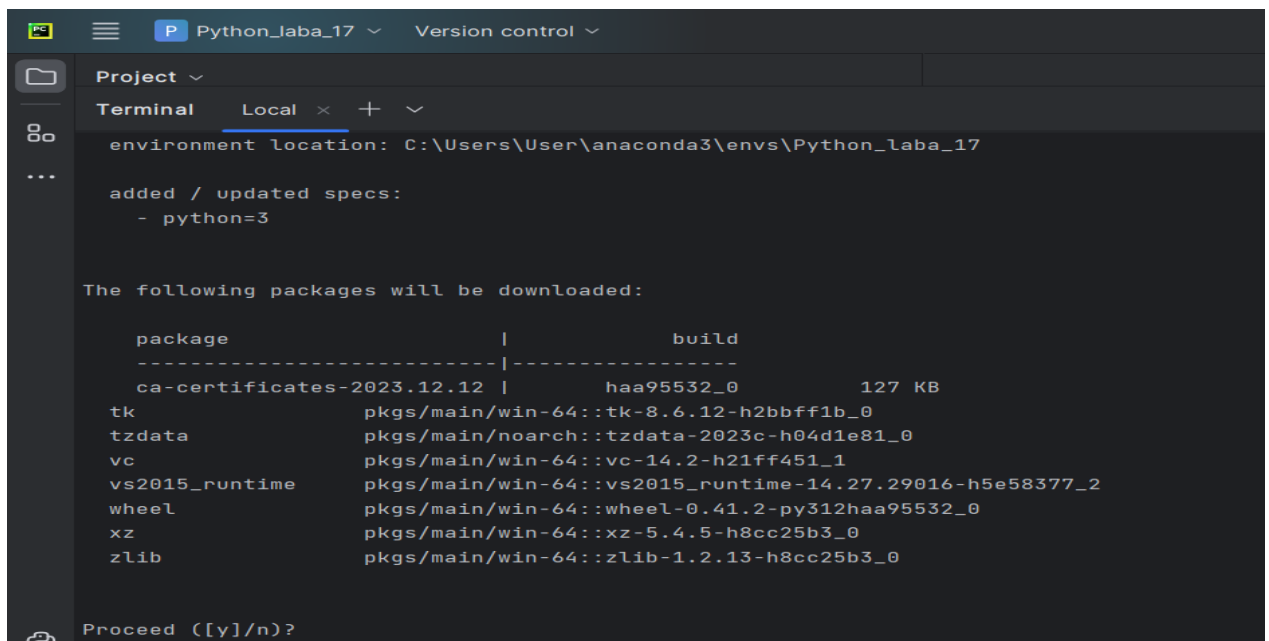
conda install conda=23.11.0

## Package Plan ##

environment location: C:\Users\User\anaconda3\envs\Python_laba_17

added / updated specs:
- python=3
```

Рисунок 1. Создание виртуального окружения Anaconda



```
Python_laba_17  Version control
Project
Terminal Local x + v
environment location: C:\Users\User\anaconda3\envs\Python_laba_17

added / updated specs:
- python=3

The following packages will be downloaded:

package | build
-----|-----
ca-certificates-2023.12.12 | haa95532_0 127 KB
tk | pkgs/main/win-64::tk-8.6.12-h2bbff1b_0
tzdata | pkgs/main/noarch::tzdata-2023c-h04d1e81_0
vc | pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime | pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel | pkgs/main/win-64::wheel-0.41.2-py312haa95532_0
xz | pkgs/main/win-64::xz-5.4.5-h8cc25b3_0
zlib | pkgs/main/win-64::zlib-1.2.13-h8cc25b3_0

Proceed ([y]/n)?
```

Рисунок 2. Список того, что будет установлено

```
Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate Python_laba_17
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

Рисунок 3. Успешная установка

```
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> 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

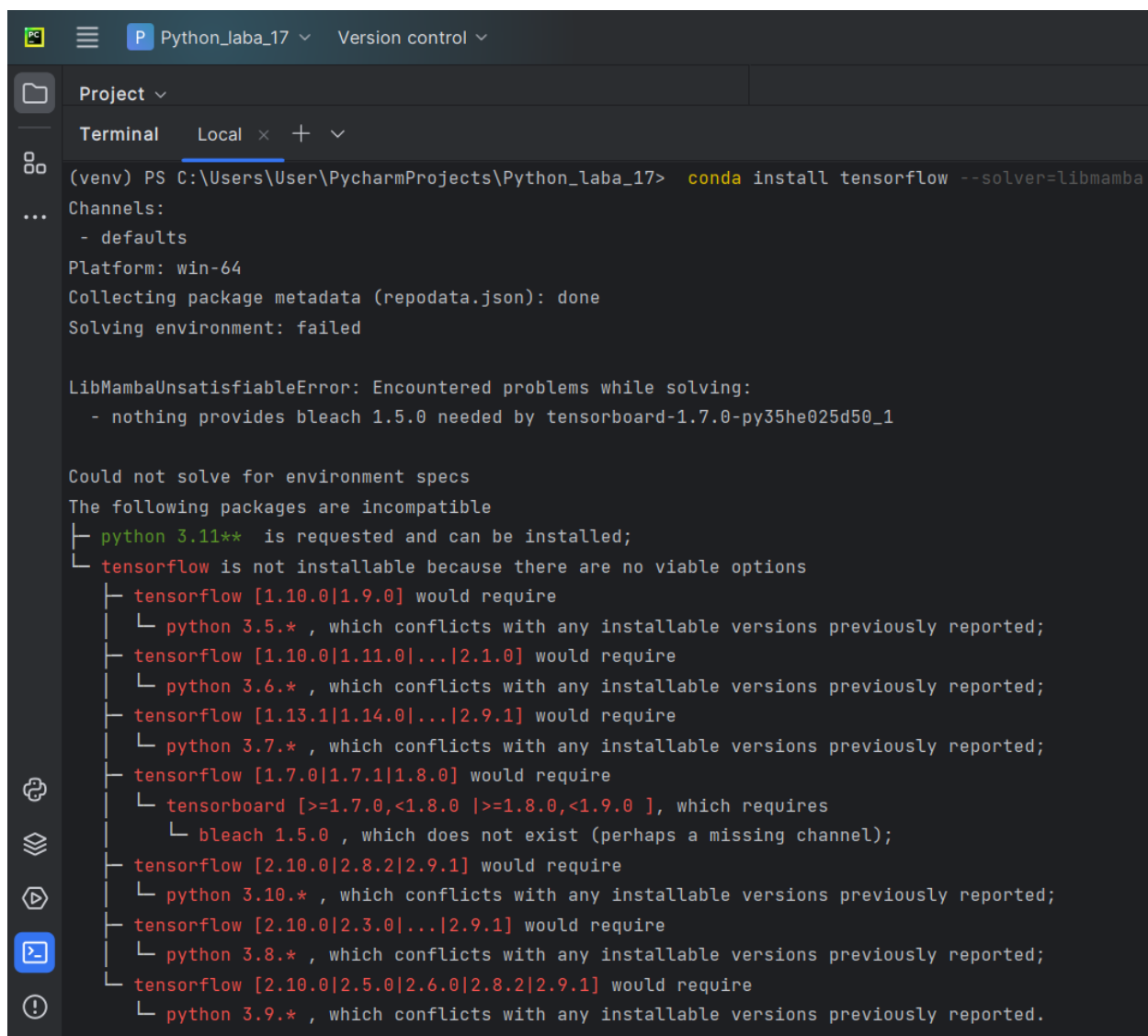
If python is on the left-most side of the chain, that's the version you've asked for.
When python appears to the right, that indicates that the thing on the left is somehow
not available for the python version you are constrained to. Note that conda will not
change your python version to a different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

Рисунок 4. Неудачная установка TensorFlow

После неудачи я решил почитать документацию Anaconda: <https://conda.github.io/conda-libmamba-solver/user-guide/>

Вкратце, есть conda—libmamba-solver — плагин, позволяющий использовать libmamba, тот же libsolv “решатель”, который используется mamba и micromamba, непосредственно в conda.

Я установил его (\$ conda install -n base conda-libmamba-solver)



```
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> conda install tensorflow --solver=libmamba
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: failed

LibMambaUnsatisfiableError: Encountered problems while solving:
 - nothing provides bleach 1.5.0 needed by tensorboard-1.7.0-py35he025d50_1

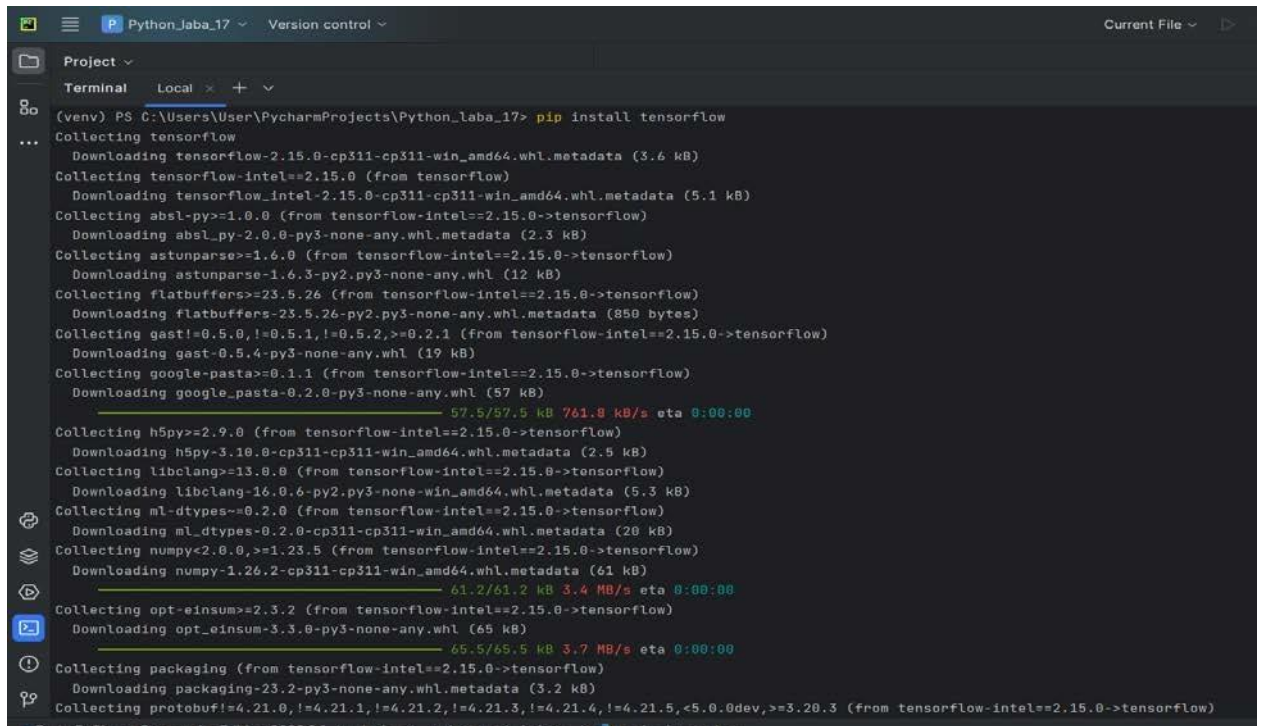
Could not solve for environment specs
The following packages are incompatible
├─ python 3.11** is requested and can be installed;
└─ tensorflow is not installable because there are no viable options
    ├─ tensorflow [1.10.0|1.9.0] would require
    │   └─ python 3.5.* , which conflicts with any installable versions previously reported;
    ├─ tensorflow [1.10.0|1.11.0|...|2.1.0] would require
    │   └─ python 3.6.* , which conflicts with any installable versions previously reported;
    ├─ tensorflow [1.13.1|1.14.0|...|2.9.1] would require
    │   └─ python 3.7.* , which conflicts with any installable versions previously reported;
    ├─ tensorflow [1.7.0|1.7.1|1.8.0] would require
    │   └─ tensorboard [>=1.7.0,<1.8.0 |>=1.8.0,<1.9.0 ], which requires
    │       └─ bleach 1.5.0 , which does not exist (perhaps a missing channel);
    ├─ tensorflow [2.10.0|2.8.2|2.9.1] would require
    │   └─ python 3.10.* , which conflicts with any installable versions previously reported;
    ├─ tensorflow [2.10.0|2.3.0|...|2.9.1] would require
    │   └─ python 3.8.* , which conflicts with any installable versions previously reported;
    └─ tensorflow [2.10.0|2.5.0|2.6.0|2.8.2|2.9.1] would require
        └─ python 3.9.* , which conflicts with any installable versions previously reported.
```

Рисунок 5. Выявление причины неудачной установки TensorFlow

Дело в том, что TensorFlow не устанавливается при взаимодействии с conda при версии языка Python 3.11 (я этой версией и пользуюсь). Есть вариант установить более старую версию и переключиться на неё при работе с TensorFlow.

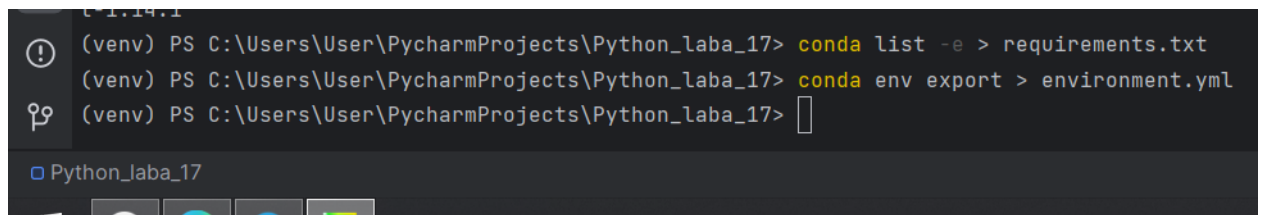
Также стоит учитывать, что пакет TensorFlow по умолчанию пытается использовать GPU для распараллеливания вычислений, то есть пытается всё перенести на графическую карту ПК. Если графический процессор не используется, то TensorFlow вычисления будет проводить на ЦП. Возможно, conda СТРОГО учитывает все характеристики при установке, а потому не позволяет эту установку осуществить.

Поэтому стоит попробовать установить TensorFlow при помощи pip.



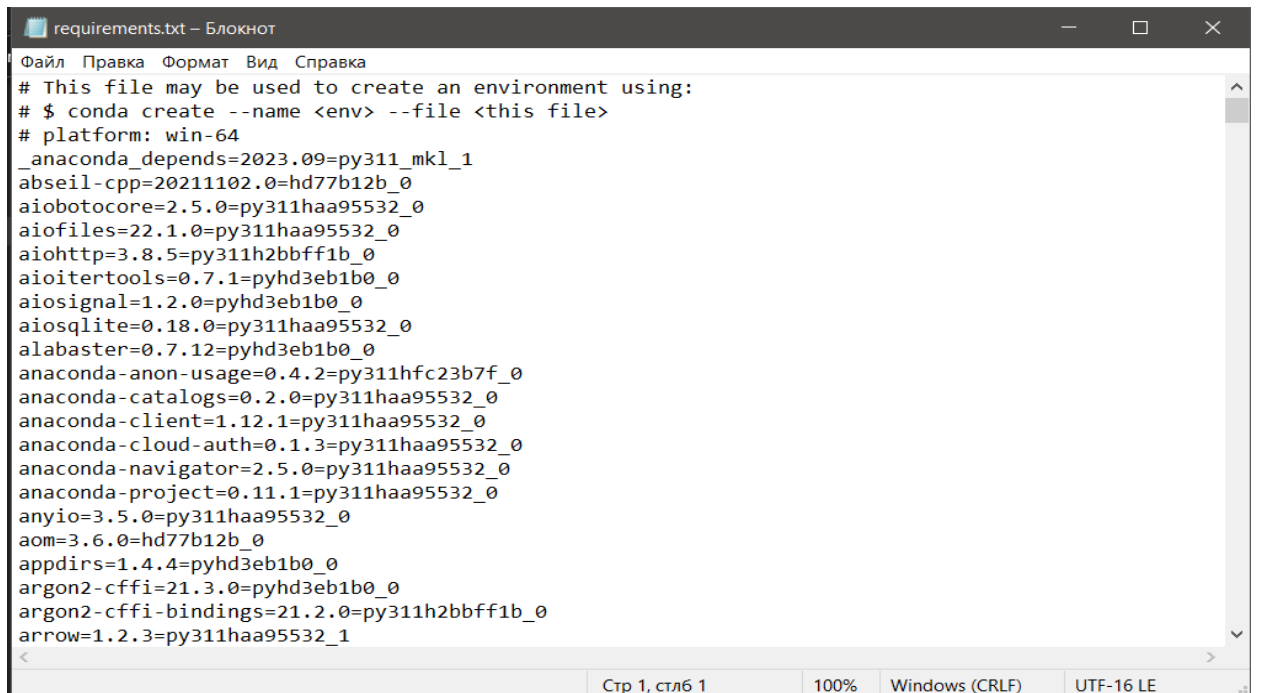
```
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-2.15.0-cp311-cp311-win_amd64.whl.metadata (3.6 kB)
Collecting tensorflow-intel==2.15.0 (from tensorflow)
  Downloading tensorflow_intel-2.15.0-cp311-cp311-win_amd64.whl.metadata (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)
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==13.0.8 (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)
Collecting mpmc==2.0.0,>=1.23.5 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading numpy-1.26.2-cp311-cp311-win_amd64.whl.metadata (61 kB)
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)
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)
```

Рисунок 6. Установка TensorFlow при помощи pip



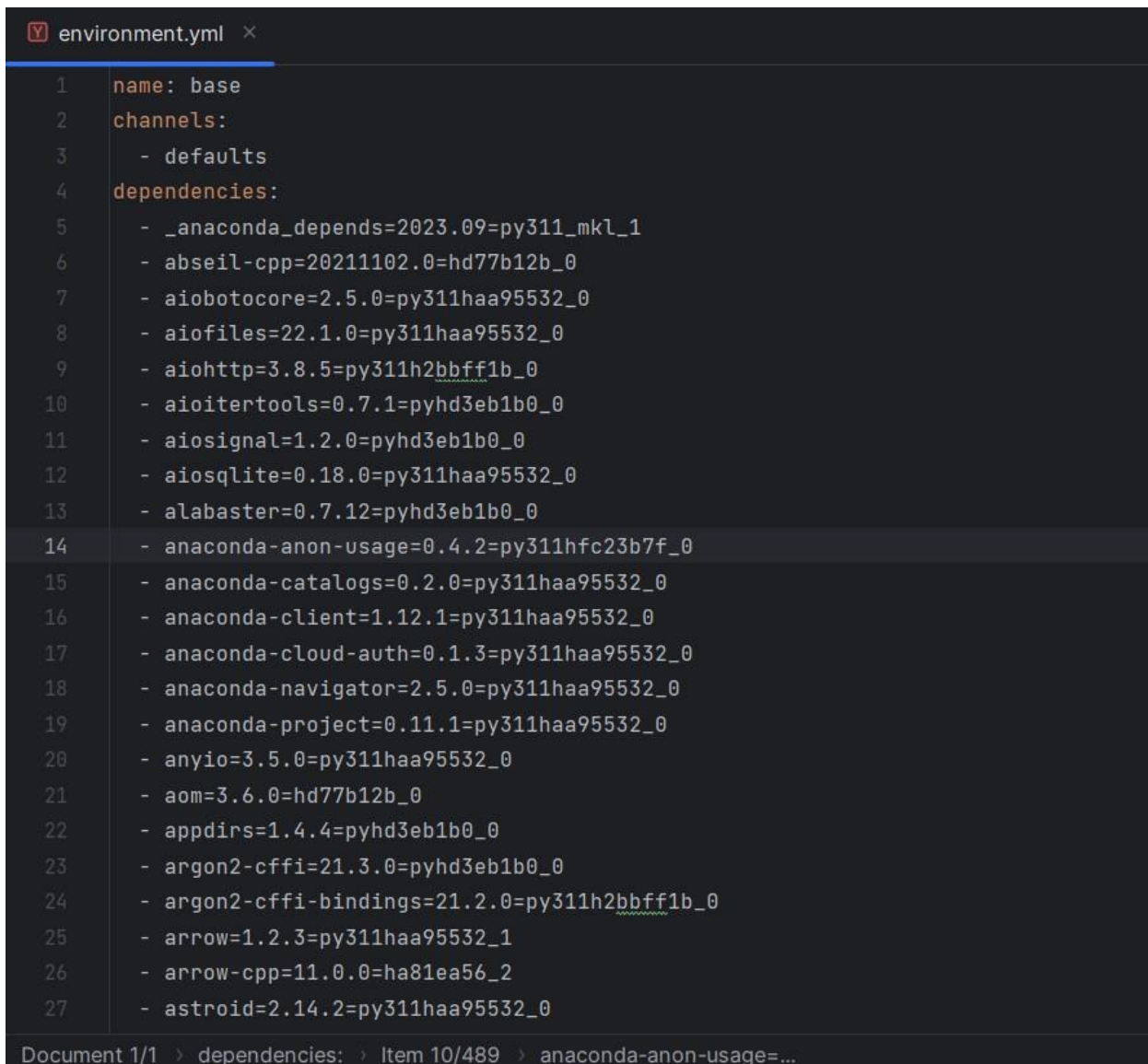
```
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> conda list -e > requirements.txt
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> conda env export > environment.yml
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17>
```

Рисунок 7. Формирование файлов requirements.txt и environments.yml



```
requirements.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
# This file may be used to create an environment using:
# $ conda create --name <env> --file <this file>
# platform: win-64
_anaconda_depends=2023.09=py311_mkl_1
abseil-cpp=20211102.0=hd77b12b_0
aioboto3=2.5.0=py311h9a95532_0
aiofiles=22.1.0=py311h9a95532_0
aiohttp=3.8.5=py311h2bbff1b_0
aioitertools=0.7.1=pyhd3eb1b0_0
aiosignal=1.2.0=pyhd3eb1b0_0
aiosqlite=0.18.0=py311h9a95532_0
alabaster=0.7.12=pyhd3eb1b0_0
anaconda-anon-usage=0.4.2=py311hfc23b7f_0
anaconda-catalogs=0.2.0=py311h9a95532_0
anaconda-client=1.12.1=py311h9a95532_0
anaconda-cloud-auth=0.1.3=py311h9a95532_0
anaconda-navigator=2.5.0=py311h9a95532_0
anaconda-project=0.11.1=py311h9a95532_0
anyio=3.5.0=py311h9a95532_0
aom=3.6.0=hd77b12b_0
appdirs=1.4.4=pyhd3eb1b0_0
argon2-cffi=21.3.0=pyhd3eb1b0_0
argon2-cffi-bindings=21.2.0=py311h2bbff1b_0
arrow=1.2.3=py311h9a95532_1
```

Рисунок 8. Файл requirements.txt



```
environment.yml
1 name: base
2 channels:
3   - defaults
4 dependencies:
5   - _anaconda_depends=2023.09=py311_mkl_1
6   - abseil-cpp=20211102.0=hd77b12b_0
7   - aiobotocore=2.5.0=py311h9a95532_0
8   - aiofiles=22.1.0=py311h9a95532_0
9   - aiohttp=3.8.5=py311h2bbff1b_0
10  - aioitertools=0.7.1=pyhd3eb1b0_0
11  - aiosignal=1.2.0=pyhd3eb1b0_0
12  - aiosqlite=0.18.0=py311h9a95532_0
13  - alabaster=0.7.12=pyhd3eb1b0_0
14  - anaconda-anon-usage=0.4.2=py311hfc23b7f_0
15  - anaconda-catalogs=0.2.0=py311h9a95532_0
16  - anaconda-client=1.12.1=py311h9a95532_0
17  - anaconda-cloud-auth=0.1.3=py311h9a95532_0
18  - anaconda-navigator=2.5.0=py311h9a95532_0
19  - anaconda-project=0.11.1=py311h9a95532_0
20  - anyio=3.5.0=py311h9a95532_0
21  - aom=3.6.0=hd77b12b_0
22  - appdirs=1.4.4=pyhd3eb1b0_0
23  - argon2-cffi=21.3.0=pyhd3eb1b0_0
24  - argon2-cffi-bindings=21.2.0=py311h2bbff1b_0
25  - arrow=1.2.3=py311h9a95532_1
26  - arrow-cpp=11.0.0=ha81ea56_2
27  - astroid=2.14.2=py311h9a95532_0
```

Document 1/1 > dependencies: > Item 10/489 > anaconda-anon-usage=...

Рисунок 9. Файл environments.yml

Вывод: в ходе выполнения данной лабораторной работы были приобретены навыки взаимодействия с менеджером пакетов `pip` и виртуальными окружениями с помощью языка программирования Python версии 3.x.

Ответы на контрольные вопросы

1. Для установки пакета Python, не входящего в стандартную библиотеку, можно использовать менеджер пакетов `pip`. Например, для установки пакета `requests`, можно выполнить команду `pip install requests`.

2. Установка менеджера пакетов `pip` осуществляется вместе с установкой Python. Если `pip` не установлен, его можно установить вручную, выполнив команду `python get-pip.py` из командной строки.

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

4. Для установки последней версии пакета с помощью `pip`, можно выполнить команду `pip install package_name`. Если не указать версию, будет установлена последняя доступная версия.

5. Для установки заданной версии пакета с помощью `pip`, можно выполнить команду `pip install package_name==version_number`.

6. Для установки пакета из `git` репозитория с помощью `pip`, можно выполнить команду `pip install git+https://github.com/username/repository.git`.

7. Для установки пакета из локальной директории с помощью `pip`, можно выполнить команду `pip install /path/to/package`.

8. Для удаления установленного пакета с помощью `pip`, можно выполнить команду `pip uninstall package_name`.

9. Для обновления установленного пакета с помощью `pip`, можно выполнить команду `pip install --upgrade package_name`.

10. Для отображения списка установленных пакетов с помощью `pip`, можно выполнить команду `pip list`.

11. Виртуальные окружения в Python используются для изоляции проектов и их зависимостей друг от друга.

12. Основные этапы работы с виртуальными окружениями включают создание нового окружения, активацию окружения, установку необходимых пакетов и деактивацию окружения.

13. Работа с виртуальными окружениями с помощью `venv` осуществляется через командную строку. Для создания нового окружения используется команда `python -m venv myenv`, а для активации окружения - `source myenv/bin/activate` (для Unix) или `myenv\Scripts\activate` (для Windows).

14. Работа с виртуальными окружениями с помощью `virtualenv` аналогична работе с `venv`.

15. `Pipenv` - это инструмент для управления зависимостями и виртуальными окружениями в Python. Работа с виртуальными окружениями `pipenv` осуществляется через командную строку. Создание нового окружения - `pipenv --python 3.8`, активация - `pipenv shell`, установка зависимостей - `pipenv install package_name`, деактивация - `exit`.

16. Файл `requirements.txt` используется для хранения списка зависимостей проекта. Создать этот файл можно вручную или автоматически с помощью команды `pip freeze > requirements.txt`. Формат файла: каждая строка содержит имя пакета и его версию (например, `package_name==1.0.0`).

17. `Conda` предоставляет дополнительные возможности по управлению зависимостями, такие как установка библиотек, написанных на C/C++, и создание изолированных сред для разработки научных вычислений.

18. Conda входит в дистрибутивы Anaconda и Miniconda.

19. Для создания виртуального окружения conda используется команда `conda create --name myenv`.

20. Активация и установка пакетов в виртуальное окружение conda осуществляется командами `conda activate myenv` и `conda install package_name`.

21. Деактивация и удаление виртуального окружения conda производится командами `conda deactivate` и `conda remove --name myenv --all`.

22. Файл `environment.yml` используется для описания всех зависимостей проекта и его окружения conda. Создать этот файл можно вручную или автоматически с помощью команды `conda env export > environment.yml`. Формат файла: YAML-структура, описывающая список пакетов и их версий.

23. Создать виртуальное окружение conda с помощью файла `environment.yml` можно командой `conda env create -f environment.yml`.

24. В IDE PyCharm работа с виртуальными окружениями conda осуществляется через настройки проекта: `File -> Settings -> Project -> Python Interpreter`. Здесь можно создавать и управлять виртуальными окружениями conda.

25. Файлы `requirements.txt` и `environment.yml` должны храниться в репозитории git, чтобы другие разработчики могли легко создать такое же окружение и установить все необходимые зависимости для работы с проектом.