

и 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 ____

Выполнил:

Репкин Александр Павлович
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. техн. наук,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____
Ставрополь, 2023 г.

Тема: Установка пакетов в Python. Виртуальные окружения.

Цель: приобрести навыки по работе с менеджером пакетов `pip` и виртуальными окружениями при помощи языка программирования Python версии 3.x.

Порядок выполнения работы:

1. Создано виртуальное окружение Anaconda с именем репозитория.

По окончании создания, окружение было активировано.

```
(base) C:\Users\yabuz\GitHub\Python17\Программы>conda create -n Python17
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: J:\Anaconda\envs\Python17

Proceed ([y]/n)? y
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate Python17
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\yabuz\GitHub\Python17\Программы>conda activate Python17

(Python17) C:\Users\yabuz\GitHub\Python17\Программы>
```

Рисунок 1. Создание виртуального окружения Anaconda.

2. Установлены требуемые пакеты: `pip`, `NumPy`, `Pandas`, `SciPy`. Для установки `pip` использована команда “`conda install pip`”, а для остальных “`pip install пакет`”.

```
pip-23.3.1      | py312haa095532_0      | 2.9 MB
python-3.12.0   | h1d029f7_0            | 16.2 MB
setuptools-68.2.2 | py312haa095532_0      | 1.2 MB
sqlite-3.41.2   | h2bbff1b_0            | 894 KB
tk-8.6.12       | h2bbff1b_0            | 3.1 MB
tzdata-2023c    | h04d1e81_0            | 116 KB
vc-14.2         | h21ff451_1            | 8 KB
vs2015_runtime-14.27.29016 | h5e58377_2          | 1007 KB
wheel-0.41.2    | py312haa95532_0       | 150 KB
xz-5.4.5        | h8cc25b3_0            | 593 KB
zlib-1.2.13     | h8cc25b3_0            | 113 KB
-----
Total:          | 34.2 MB

The following NEW packages will be INSTALLED:

bzip2             pkgs/main/win-64::bzip2-1.0.8-he774522_0
ca-certificates  pkgs/main/win-64::ca-certificates-2023.12.12-haa95532_0
expat             pkgs/main/win-64::expat-2.5.0-hd77b12b_0
libffi           pkgs/main/win-64::libffi-3.4.4-hd77b12b_0
openssl          pkgs/main/win-64::openssl-3.0.12-h2bbff1b_0
pip             pkgs/main/win-64::pip-23.3.1-py312haa95532_0
python          pkgs/main/win-64::python-3.12.0-h1d929f7_0
setuptools       pkgs/main/win-64::setuptools-68.2.2-py312haa95532_0
sqlite          pkgs/main/win-64::sqlite-3.41.2-h2bbff1b_0
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)? y

Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(Python17) C:\Users\yabuz\GitHub\Python17\Программы>pip list
Package Version
-----
pip      23.3.1
setuptools 68.2.2
wheel    0.41.2
```

Рисунок 2. Успешная установка pip.

```
(Python17) C:\Users\yabuz\GitHub\Python17\Программы>pip install NumPy
Collecting NumPy
  Downloading numpy-1.26.2-cp312-cp312-win_amd64.whl.metadata (61 kB)
-----
61.2/61.2 kB 217.8 kB/s eta 0:00:00
  Downloading numpy-1.26.2-cp312-cp312-win_amd64.whl (15.5 MB)
-----
15.5/15.5 MB 6.3 MB/s eta 0:00:00
Installing collected packages: NumPy
Successfully installed NumPy-1.26.2

(Python17) C:\Users\yabuz\GitHub\Python17\Программы>pip list
Package Version
-----
numpy      1.26.2
pip        23.3.1
setuptools 68.2.2
wheel      0.41.2
```

Рисунок 3. Успешная установка NumPy.

```
(Python17) C:\Users\yabuz\GitHub\Python17\Программы>pip install Pandas
Collecting Pandas
  Downloading pandas-2.1.4-cp312-cp312-win_amd64.whl.metadata (18 kB)
Requirement already satisfied: numpy<2, >=1.26.0 in j:\anaconda\envs\python17\lib\site-packages (from Pandas) (1.26.2)
Collecting python-dateutil<=2.8.2 (from Pandas)
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
-----
247.7/247.7 kB 842.1 kB/s eta 0:00:00
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-2023.3-py2.py3-none-any.whl (341 kB)
-----
341.8/341.8 kB 3.0 MB/s eta 0:00:00
Collecting six>=1.5 (from python-dateutil<=2.8.2->Pandas)
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Downloaded pandas-2.1.4-cp312-cp312-win_amd64.whl (10.5 MB)
-----
10.5/10.5 MB 5.2 MB/s eta 0:00:00
Downloaded pytz-2023.3.post1-py2.py3-none-any.whl (502 kB)
-----
502.5/502.5 kB 3.9 MB/s eta 0:00:00
Installing collected packages: pytz, tzdata, six, python-dateutil, Pandas
Successfully installed Pandas-2.1.4 python-dateutil-2.8.2 pytz-2023.3.post1 six-1.16.0 tzdata-2023.3

(Python17) C:\Users\yabuz\GitHub\Python17\Программы>pip list
Package Version
-----
numpy      1.26.2
pandas     2.1.4
pip        23.3.1
python-dateutil 2.8.2
pytz       2023.3.post1
setuptools 68.2.2
six        1.16.0
tzdata     2023.3
wheel      0.41.2
```

Рисунок 4. Успешная установка Pandas.

```
(Python17) C:\Users\yabuz\GitHub\Python17\Программы>pip install SciPy
Collecting SciPy
  Downloading scipy-1.11.4-cp312-cp312-win_amd64.whl.metadata (60 kB)
----- 60.4/60.4 kB 230.1 kB/s eta 0:00:00
Requirement already satisfied: numpy<1.28.0,>=1.21.6 in j:\anaconda\envs\python17\lib\site-packages (from SciPy) (1.26.2)
Downloading scipy-1.11.4-cp312-cp312-win_amd64.whl (43.7 MB)
----- 43.7/43.7 MB 5.3 MB/s eta 0:00:00
Installing collected packages: SciPy
Successfully installed SciPy-1.11.4

(Python17) C:\Users\yabuz\GitHub\Python17\Программы>pip list
Package            Version
-----
numpy              1.26.2
pandas             2.1.4
pip               23.3.1
python-dateutil    2.8.2
pytz              2023.3.post1
scipy              1.11.4
setuptools         68.2.2
six               1.16.0
tzdata            2023.3
wheel              0.41.2
```

Рисунок 5. Успешная установка SciPy.

3. Произведена попытка установки пакета TensorFlow при помощи `conda install`, однако получена ошибка. Согласно ей, причина заключается в допустимых версиях. Последняя поддерживаемая версия – 3.10, в то время как в персональном пользовании – 3.12. После выявления причины ошибки, произведена успешная попытка установки при помощи `pip install`.

```
(Python17) C:\Users\yabuz\GitHub\Python17\Программы>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=3.12

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 6. Безуспешная попытка установки TensorFlow через conda.

```
(Python17) C:\Users\yabuz\GitHub\Python17\Программы>pip list
```

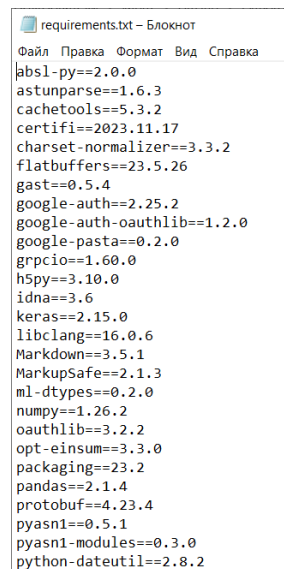
Package	Version
absl-py	2.0.0
astunparse	1.6.3
cachetools	5.3.2
certifi	2023.11.17
charset-normalizer	3.3.2
flatbuffers	23.5.26
gast	0.5.4
google-auth	2.25.2
google-auth-oauthlib	1.2.0
google-pasta	0.2.0
grpcio	1.60.0
h5py	3.10.0
idna	3.6
keras	2.15.0
libclang	16.0.6
Markdown	3.5.1
MarkupSafe	2.1.3
ml-dtypes	0.2.0
numpy	1.26.2
oauthlib	3.2.2
opt-einsum	3.3.0
packaging	23.2
pandas	2.1.4
pip	23.3.1
protobuf	4.23.4
pyasn1	0.5.1
pyasn1-modules	0.3.0
python-dateutil	2.8.2
pytz	2023.3.post1
requests	2.31.0
requests-oauthlib	1.3.1
rsa	4.9
scipy	1.11.4
setuptools	68.2.2
six	1.16.0
tensorboard	2.15.1
tensorboard-data-server	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_extensions	4.9.0
tzdata	2023.3

Рисунок 7. Успешная установка TensorFlow через pip.

4. Созданы файлы requirements.txt (При помощи команды `pip freeze > requirements.txt`) и environment.yml (При помощи команды `conda env export > environment.yml`). Исходя из хранящихся в них данных, requirements.txt хранит в себе зависимости (С его помощью можно быстро установить все требуемые пакеты), а environment.yml – параметры текущего окружения (Благодаря нему можно легко и быстро воссоздать окружение)

```
(Python17) C:\Users\yabuz\GitHub\Python17\Программы>pip freeze > requirements.txt
(Python17) C:\Users\yabuz\GitHub\Python17\Программы>conda env export > environment.yml
```

Рисунок 8. Создание файлов requirements и environment.



```
requirements.txt – Блокнот
Файл Правка Формат Вид Справка
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
MarkupSafe==2.1.3
ml-dtypes==0.2.0
numpy==1.26.2
oauthlib==3.2.2
opt-einsum==3.3.0
packaging==23.2
pandas==2.1.4
protobuf==4.23.4
pyasn1==0.5.1
pyasn1-modules==0.3.0
python-dateutil==2.8.2
```

Рисунок 9. Содержимое файла requirements.txt.



```
environment.yml – Блокнот
Файл Правка Формат Вид Справка
name: Python17
channels:
- defaults
dependencies:
- bzip2=1.0.8=he774522_0
- ca-certificates=2023.12.12=h5555522_0
- expat=2.5.0=hd77b12b_0
- libffi=3.4.4=hd77b12b_0
- openssl=3.0.12=h2bbff1b_0
- python=3.10.13=he1021f5_0
- setuptools=68.2.2=py310h555522_0
- sqlite=3.41.2=h2bbff1b_0
- tk=8.6.12=h2bbff1b_0
- vc=14.2=h21ffa451_1
- vs2015_runtime=14.27.29016=h5e58377_2
- wheel=0.41.2=py310h555522_0
- xz=5.4.5=h8cc25b3_0
- zlib=1.2.13=h8cc25b3_0
- pip:
- 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
- markupsafe==2.1.3
- ml-dtypes==0.2.0
- numpy==1.26.2
- oauthlib==3.2.2
- opt-einsum==3.3.0
- packaging==23.2
- pandas==2.1.4
```

Рисунок 10. Содержимое файла environment.yml.

5. Ответы на вопросы.

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

Ответ: менеджер пакетов `pip` позволяет устанавливать пакеты, не входящие в стандартную библиотеку.

2) Как осуществить установку менеджера пакетов `pip`?

Ответ: зачастую менеджер пакетов `pip` устанавливается по умолчанию, однако если этого не произошло, то можно использовать команды его установки: в Anaconda - `conda install pip`. В Windows для этого нужно скачать установочный скрипт `get-pip.py`, и открыть его в командной строке при помощи команды `python get-pip.py`.

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

Ответ: менеджер пакетов `pip` находит требуемые пакеты в репозитории Python Package Index (PyPI).

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

Ответ: для установки пакетов при помощи `pip` существует команда `pip install "название пакета"`.

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

Ответ: если необходимо установить не последнюю версию пакета, то к стандартной команде установки `pip install "название пакета"` добавляется `==` "требуемая версия".

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

Ответ: для установки пакета из любого `git` репозитория при помощи `pip` можно воспользоваться командой `pip install git+"ссылка на требуемый репозиторий"`.

7) Как установить пакет из локальной директории с помощью `pip`?

Ответ: `pip` позволяет устанавливать пакеты из локальной директории, если использовать команду `pip install "полный путь к пакету"`.

8) Как удалить установленный пакет с помощью `pip`?

Ответ: команда `pip uninstall "название пакета"` позволяет избавиться от пакета.

9) Как обновить установленный пакет с помощью `pip`?

Ответ: команда `pip install --upgrade "название пакета"` позволяет обновить требуемый пакет до последней версии.

10) Как отобразить список установленных пакетов с помощью pip?

Ответ: все установленные пакеты можно увидеть при помощи команды `pip list`.

11) Каковы причины появления виртуальных окружений в языке Python?

Ответ: виртуальные окружения позволяют изолировать проекты и их зависимости. Появилась такая необходимость из-за ограничения Python – может быть установлено не больше 1 версии пакета.

12) Каковы основные этапы работы с виртуальными окружениями?

Ответ: основные этапы работы с виртуальным окружением: создание (`conda create -n "название окружения"`), активация (`conda activate "название окружения"`), деактивация (`conda deactivate "название окружения"`), удаление (`conda env remove --name "название окружения"`).

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

Ответ: при работе с venv необходимо пользоваться командами: создание окружения (`python -m venv "название окружения"`), активация (`source "название окружения"/bin/activate`), деактивация (`deactivate`).

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

Ответ: при работе с virtualenv необходимо пользоваться командами: создание окружения (`virtualenv "название окружения"`), активация (`source "название окружения"/bin/activate`), деактивация (`deactivate`).

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

Ответ: при работе с virtualenv необходимо пользоваться командами: создание окружения (`pipenv install`), активация (`pipenv shell`), деактивация (`exit`), удаление (`pipenv -remove`).

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

Ответ: requirements.txt является списком зависимостей проекта. Для его создания необходимо использовать команду `pip freeze > requirements.txt`. Зависимости представлены в виде: “пакет”==”версия”.

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

Ответ: conda, в отличие от pip, способен управлять библиотеками и окружением (даже бинарными зависимостями). Pip же способен управлять только пакетами.

18) В какие дистрибутивы Python входит пакетный менеджер conda?

Ответ: conda входит в дистрибутивы Anaconda и Miniconda.

19) Как создать виртуальное окружение conda?

Ответ: команда `conda create -n “название окружения”` позволяет создать новое окружение при помощи conda.

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

Ответ: команда `conda activate “название окружения”` позволяет активировать требуемое окружение, а команда `conda install “название пакета”` позволяет устанавливать требуемые пакеты.

21) Как деактивировать и удалить виртуальное окружение conda?

Ответ: команда `conda deactivate “название окружения”` позволяет деактивировать текущее окружение, а команда `conda env remove --name “название окружения”` удаляет требуемое окружение.

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

Ответ: файл environment.yml хранит в себе параметры окружения, благодаря которым можно соответствующее окружение легко воссоздать. Команда `conda env export > environment.yml` создаёт такой файл относительно активного окружения.

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

Ответ: для создания окружения на основе параметров из файла `environment.yml`, можно воспользоваться командой `conda env create -f environment.yml`.

24) Каков порядок работы с виртуальными окружениями `conda` в IDE PyCharm?

Ответ: если установлены `conda` и PyCharm, то в настройках PyCharm можно в разделе “Интерпретаторы” добавить новый интерпретатор `conda`. После успешной настройки нового интерпретатора появляется возможность работы с виртуальным окружением, установки пакетов.

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

Ответ: так как файлы `requirements.txt` и `environment.yml` содержат информацию о зависимостях проекта, то их наличие в `git` облегчает воссоздание того же окружения на других системах и для других разработчиков.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе с менеджером пакетов `pip` и виртуальными окружениями при помощи языка программирования Python версии 3.x.