

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 №2.14
дисциплины «Программирование на Python»
Вариант ____

Выполнил:
Иващенко Олег Андреевич
2 курс, группа ИВТ-б-о-22-1,
09.03.02 «Информационные и
вычислительные машины»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем»

(подпись)

Руководитель практики:
Воронкин Роман Александрович,
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

```
(base) PS C:\Users\PackardBell\Desktop\Python_2.14> conda create -n Python_2.14 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 1 – Установка окружения conda

```
(base) PS C:\Users\PackardBell\Desktop\Python_2.14> conda activate Python_2.14
(Python_2.14) PS C:\Users\PackardBell\Desktop\Python_2.14>
```

Рисунок 2 – Активация окружения

```
package                        | build                                |
-----|-----|
intel-openmp-2023.1.0          | h59b6b97_46320                      | 2.7 MB
mkl-2023.1.0                   | h6b88ed4_46358                      | 155.9 MB
numpy-1.26.2                   | py311hdab7c0b_0                    | 11 KB
numpy-base-1.26.2             | py311hd01c5d8_0                    | 7.2 MB
-----|-----|
Total:                          165.8 MB

The following NEW packages will be INSTALLED:

blas                pkgs/main/win-64::blas-1.0-mkl
intel-openmp        pkgs/main/win-64::intel-openmp-2023.1.0-h59b6b97_46320
mkl                 pkgs/main/win-64::mkl-2023.1.0-h6b88ed4_46358
mkl-service         pkgs/main/win-64::mkl-service-2.4.0-py311h2bbff1b_1
mkl_fft             pkgs/main/win-64::mkl_fft-1.3.8-py311h2bbff1b_0
mkl_random          pkgs/main/win-64::mkl_random-1.2.4-py311h59b6b97_0
numpy               pkgs/main/win-64::numpy-1.26.2-py311hdab7c0b_0
numpy-base         pkgs/main/win-64::numpy-base-1.26.2-py311hd01c5d8_0
tbb                 pkgs/main/win-64::tbb-2021.8.0-h59b6b97_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
mkl-2023.1.0          | 155.9 MB | 2 | 0%
numpy-base-1.26.2    | 7.2 MB   | #####5 | 10%
numpy-1.26.2         | 11 KB    | ##### | 100%
intel-openmp-2023.1. | 2.7 MB   | #####3 | 21%
```

Рисунок 3 – Установка пакета NumPy

```
(Python_2.14) PS C:\Users\PackardBell\Desktop\Python_2.14> conda list
# packages in environment at C:\Users\PackardBell\anaconda3\envs\Python_2.14:
#
# 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                        py311h1fcbae_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

```
(Python_2.14) PS C:\Users\PackardBell\Desktop\Python_2.14> 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 – Неудачные попытки установки пакета TensorFlow с помощью conda

```

Downloading google_auth_oauthlib-1.2.0-py2.py3-none-any.whl (24 kB)
Downloading Markdown-3.5.1-py3-none-any.whl (102 kB)
----- 102.2/102.2 kB 1.2 MB/s eta 0:00:00
Downloading requests-2.31.0-py3-none-any.whl (62 kB)
----- 62.6/62.6 kB 670.9 kB/s eta 0:00:00
Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Downloading werkzeug-3.0.1-py3-none-any.whl (226 kB)
----- 226.7/226.7 kB 1.1 MB/s eta 0:00:00
Downloading cachetools-5.3.2-py3-none-any.whl (9.3 kB)
Downloading certifi-2023.11.17-py3-none-any.whl (162 kB)
----- 162.5/162.5 kB 750.2 kB/s eta 0:00:00
Downloading charset_normalizer-3.3.2-cp311-cp311-win_amd64.whl (99 kB)
----- 99.9/99.9 kB 717.0 kB/s eta 0:00:00
Downloading idna-3.6-py3-none-any.whl (61 kB)
----- 61.6/61.6 kB 658.0 kB/s eta 0:00:00
Downloading MarkupSafe-2.1.3-cp311-cp311-win_amd64.whl (17 kB)
Downloading urllib3-2.1.0-py3-none-any.whl (104 kB)
----- 104.6/104.6 kB 669.1 kB/s eta 0:00:00
Downloading pyasn1-0.5.1-py2.py3-none-any.whl (84 kB)
----- 84.9/84.9 kB 803.1 kB/s eta 0:00:00
Installing collected packages: libclang, flatbuffers, wrapt, urllib3, typing-extensions, termcolor,
tensorflow-io-gcs-filesystem, tensorflow-estimator, tensorboard-data-server, pyasn1, protobuf, pac
kaging, opt-einsum, oauthlib, ml-dtypes, MarkupSafe, markdown, keras, idna, h5py, grpcio, google-pa
sta, gast, charset-normalizer, certifi, cachetools, astunparse, absl-py, werkzeug, rsa, requests, p
yasn1-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 lib
clang-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 tensorboa
rd-2.15.1 tensorboard-data-server-0.7.2 tensorflow-estimator-2.15.0 tensorflow-intel-2.15.0 tensorf
low-io-gcs-filesystem-0.31.0 termcolor-2.4.0 typing-extensions-4.9.0 urllib3-2.1.0 werkzeug-3.0.1 w
rapt-1.14.1
(Python_2.14) PS C:\Users\PackardBell\Desktop\Python_2.14>

```

Рисунок 6 – Удалачная установка TensorFlow с помощью pip

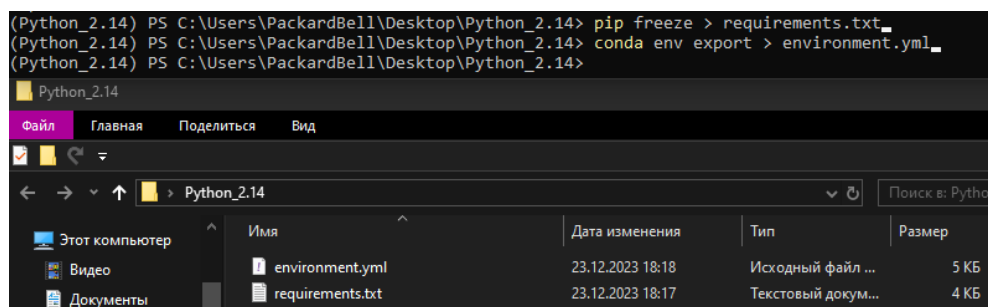


Рисунок 7 – Формирование файлов requirements.txt и environment.yml

```
name: Python_2.14
channels:
- defaults
dependencies:
- 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=py311h1fcbae_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=h21ffa451_1
- vs2015_runtime=14.27.29016=h5e58377_2
- wheel=0.41.2=py311haa95532_0
```

Рисунок 8 – Содержимое файла environment.yml

```
requirements.txt - Блокнот
Файл Правка Формат Вид Справка
absl-py==2.0.0
astunparse==1.6.3
Bottleneck @ file:///C:/ci_311/bottleneck_167650016583/work
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
mkl-fft @ file:///C:/b/abs_19i1y8ykas/croot/mkl_fft_1695058226480/work
mkl-random @ file:///C:/b/abs_edwkj1_069/croot/mkl_random_1695059866750/work
mkl-service==2.4.0
ml-dtypes==0.2.0
numexpr @ file:///C:/b/abs_5fucrt5dc/croot/numexpr_1696515448831/work
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
oauthlib==3.2.2
opt-einsum==3.3.0
packaging==23.2
pandas @ file:///C:/b/abs_fej9bi0gew/croot/pandas_1702318041921/work/dist/pandas-2.1.4-cp311-cp311-win_amd64.whl#sha256=d3609b7cc3e
protobuf==4.23.4
pyasn1==0.5.1
pyasn1-modules==0.3.0
python-dateutil @ file:///tmp/build/80754af9/python-dateutil_1626374649649/work
pytz @ file:///C:/b/abs_19q31jkez4/croot/pytz_1695131651401/work
requests==2.31.0
```

Рисунок 9 – Содержимое файла requirements.txt

Контрольные вопросы

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

Можно использовать инструмент установки пакетов `pip`. Команда для установки пакета выглядит так:

```
pip install <package_name>
```

2. Как осуществить установку менеджера пакетов `pip`?

В большинстве случаев `pip` поставляется вместе с Python. Однако, если его нет, его можно установить, выполнив команду:

```
python -m ensurepip --default-pip
```

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

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

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

```
pip install <package_name> --upgrade
```

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

```
pip install <package_name>==<version>
```

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

```
pip install git+<git-url>
```

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

```
pip install <path>
```

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

```
pip uninstall <package_name>
```

9. Как обновить установленный пакет с помощью pip?

```
pip install <package_name> --upgrade
```

10. Как отобразить список установленных пакетов с помощью pip?

```
pip list
```

11. Каковы причины появления виртуальных окружений в языке Python?

Виртуальные окружения позволяют изолировать зависимости и версии пакетов для конкретного проекта, предотвращая конфликты между различными проектами.

12. Каковы основные этапы работы с виртуальными окружениями?

Этапы:

- Создание виртуального окружения;
- Активация виртуального окружения;
- Установка необходимых пакетов;
- Работа с проектом внутри виртуального окружения.

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

```
python -m venv myenv # Создание виртуального окружения
source myenv/bin/activate # Активация виртуального окружения (для Unix)
```

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

```
pip install virtualenv # Установка virtualenv
virtualenv myenv # Создание виртуального окружения
source myenv/bin/activate # Активация виртуального окружения (для Unix)
```

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

```
pip install pipenv # Установка pipenv
pipenv install    # Создание и установка зависимостей
pipenv shell      # Активация виртуального окружения
```

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

Файл `requirements.txt` содержит список зависимостей проекта. Его можно создать вручную, перечислив зависимости и их версии. Пример формата файла:

```
requests==2.25.1
Flask==2.1.0
```

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

`Conda` является пакетным менеджером и системой управления окружениями, позволяющим управлять зависимостями не только для Python, но и для других языков программирования. `Conda` также обеспечивает установку бинарных пакетов, что может быть полезным для библиотек, имеющих зависимости от сторонних библиотек.

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

`Conda` входит в дистрибутив `Anaconda`, `Miniconda` и некоторые другие дистрибутивы Python.

19. Как создать виртуальное окружение `conda`?

```
conda create --name myenv
```

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

```
conda activate myenv
```


`conda install <package_name>`

21. Как деактивировать и удалить виртуальное окружение conda?

`conda deactivate`

`conda env remove --name myenv`

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

Файл `environment.yml` в Conda используется для определения окружения, включая зависимости. его можно создать вручную, а также сгенерировать с помощью команды:

`conda env export > environment.yml`

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

`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.

В PyCharm можно создать и использовать виртуальные окружения conda следующим образом:

- Открыть проект в PyCharm;
- Перейти в «File» -- «Settings» -- «Project: <project_name>» -- «Python Interpreter»;
- Нажать на иконку шестерёнки и выбрать «Add...»;
- Выбрать «Conda Environment»;
- Указать путь к интерпретатору conda и выбрать нужное окружение;
- Нажать «Ok».

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

Файлы requirements.txt и environment.yml используются для описания зависимостей проекта, и их наличие в репозитории Git позволяет другим разработчикам воссоздать окружение проекта с необходимыми зависимостями. Это обеспечивает воспроизводимость окружения и согласованность зависимостей между разными средствами разработки.

Выводы: В процессе выполнения лабораторной работы были приобретены навыки по работе с менеджером пакетов pip и виртуальными окружениями с помощью языка программирования Python версии 3.x, было создано виртуальное окружение с помощью Anaconda, с помощью менеджера пакетов conda установлен ряд пакетов. Однако, с пакетом TensorFlow возникли проблемы при установке с помощью conda – решение заморожено. Но при помощи менеджера пакетов pip установка прошла успешно. Также были сформированы файлы environment.yml и requirements.txt. В первом файле находятся параметры окружения, нужные для восстановления окружения в любой момент. Во втором находятся пакеты и зависимости, которые были установлены на момент формирования файла.