

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 «Основы кроссплатформенного программирования»**

Выполнил:  
Евдаков Евгений Владимирович  
2 курс, группа ИТС-б-о-22-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи»,  
направленность (профиль)  
«Инфокоммуникационные системы и  
сети», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р. А., 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.

### **Ход работы:**

**Задание 1.** Создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер.

```
C:\Users\Gaming-PC>git clone https://github.com/EvgenyEvdakov/Laba_2.14.git
Cloning into 'Laba_2.14'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
```

Рисунок 1. Клонирование репозитория

**Задание 2.** Организовал свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\Users\Gaming-PC\Laba_2.14>git flow init

which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Gaming-PC/Laba_2.14/.git/hooks]
```

Рисунок 2. Модель ветвления git-flow

**Задание 3.** Создал виртуальное окружение Anaconda с именем репозитория.

```
(base) PS C:\Users\Gaming-PC\Laba_2.14> conda create -n Laba_2.14 python=3.7
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.1.0
  latest version: 23.7.3

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=23.7.3

## Package Plan ##

environment location: C:\Users\Gaming-PC\.conda\envs\Laba_2.14

added / updated specs:
- python=3.7

The following packages will be downloaded:
```

package	build	size
ca-certificates-2023.08.22	haa95532_0	123 KB
certifi-2022.12.7	py37haa95532_0	149 KB
openssl-1.1.1v	h2bbff1b_0	5.5 MB
pip-22.3.1	py37haa95532_0	2.7 MB
python-3.7.16	h6244533_0	17.2 MB
setuptools-65.6.3	py37haa95532_0	1.1 MB
sqlite-3.41.2	h2bbff1b_0	894 KB
wheel-0.38.4	py37haa95532_0	82 KB
wincertstore-0.2	py37haa95532_2	15 KB
<b>Total:</b>		<b>27.8 MB</b>

```
The following NEW packages will be INSTALLED:

ca-certificates pkgs/main/win-64::ca-certificates-2023.08.22-haa95532_0
certifi         pkgs/main/win-64::certifi-2022.12.7-py37haa95532_0
openssl        pkgs/main/win-64::openssl-1.1.1v-h2bbff1b_0
pip            pkgs/main/win-64::pip-22.3.1-py37haa95532_0
python         pkgs/main/win-64::python-3.7.16-h6244533_0
setuptools     pkgs/main/win-64::setuptools-65.6.3-py37haa95532_0
sqlite        pkgs/main/win-64::sqlite-3.41.2-h2bbff1b_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-0.38.4-py37haa95532_0
wincertstore   pkgs/main/win-64::wincertstore-0.2-py37haa95532_2
```

Рисунок 3. Создание виртуального окружения

```
(base) PS C:\Users\Gaming-PC\Laba_2.14> conda activate Laba_2.14
(Laba_2.14) PS C:\Users\Gaming-PC\Laba_2.14> conda list
# packages in environment at C:\Users\Gaming-PC\.conda\envs\Laba_2.14:
#
# Name                                Version                                Build                                Channel
ca-certificates                       2023.08.22                             haa95532_0
certifi                               2022.12.7                              py37haa95532_0
openssl                               1.1.1v                                 h2bbff1b_0
pip                                    22.3.1                                 py37haa95532_0
python                                3.7.16                                 h6244533_0
setuptools                             65.6.3                                py37haa95532_0
sqlite                                 3.41.2                                 h2bbff1b_0
vc                                     14.2                                  h21ff451_1
vs2015_runtime                        14.27.29016                            h5e58377_2
wheel                                  0.38.4                                 py37haa95532_0
wincertstore                          0.2                                    py37haa95532_2
```

Рисунок 4. Активация окружения

**Задание 4.** Установил в виртуальное окружение следующие пакеты:  
pip, NumPy, Pandas, SciPy.

```
(Laba_2.14) PS C:\Users\Gaming-PC\Laba_2.14> conda install pip, numpy, pandas, scipy
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.1.0
  latest version: 23.7.3

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=23.7.3

## Package Plan ##

environment location: C:\Users\Gaming-PC\.conda\envs\Laba_2.14

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

The following packages will be downloaded:
```

package	build	size
bottleneck-1.3.5	py37h080aadc_0	105 KB
fftw-3.3.9	h2bbff1b_1	672 KB
mk1-service-2.4.0	py37h2bbff1b_0	49 KB
mk1_fft-1.3.1	py37h277e83a_0	135 KB
mk1_random-1.2.2	py37hf11a4ad_0	216 KB
numexpr-2.8.4	py37h5b0cc5e_0	127 KB
numpy-1.21.5	py37h7a0a035_3	25 KB
numpy-base-1.21.5	py37hca35cd5_3	4.4 MB
packaging-22.0	py37haa95532_0	67 KB
pandas-1.3.5	py37h6214cd6_0	8.4 MB
pytz-2022.7	py37haa95532_0	210 KB
scipy-1.7.3	py37h7a0a035_2	13.8 MB
Total:		28.1 MB

Рисунок 5. Установка пакетов

**Задание 5.** Попытался установить менеджером пакетов conda пакет TensorFlow. В результате выдало предупреждение о разных версиях.

```
(Laba_2.14) PS C:\Users\Gaming-PC\Laba_2.14> conda install TensorFlow
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.1.0
  latest version: 23.7.3

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=23.7.3

## Package Plan ##

environment location: C:\Users\Gaming-PC\.conda\envs\Laba_2.14

added / updated specs:
- tensorflow
```


Рисунок 6. Установка TensorFlow через conda

После попытки установки пакета TensorFlow через conda, воспользовался менеджером пакетов pip. После чего выполнил установку данного пакета.

```
tensorboard 2.13.0
tensorboard-data-server 0.7.1
```

Рисунок 7. Установка TensorFlow через pip

**Задание 6.** Сформировал файл `environment.yml` он определяет окружение conda, включая название, каналы (источники пакетов) и список зависимостей. Он позволяет указать не только Python-пакеты, но и другие зависимости (например, библиотеки, необходимые для компиляции), а также версию Python.

A screenshot of a code editor window showing the content of a file named `environment.yml`. The file is located at `C:\Users\Gaming-PC\Lab_2_14\environment.yml`. The content of the file is as follows:

```
1 name: Lab_2_14
2 channels:
3   - defaults
4 dependencies:
5   - _tfselect=2.2.0=eigen
6   - absl-py=1.3.0=py37haa95532_0
7   - aiohttp=3.8.3=py37h2bbff1b_0
8   - aiosignal=1.2.0=pyhd3eb1b0_0
9   - astunparse=1.6.3=py_0
10  - async-timeout=4.0.2=py37haa95532_0
11  - async-test=0.13.0=py_0
12  - attrs=22.1.0=py37haa95532_0
13  - blas=1.0=mkl
14  - blinker=1.4=py37haa95532_0
15  - bottleneck=1.3.5=py37h080aedc_0
16  - brotli=0.7.0=py37h2bbff1b_1003
17  - ca-certificates=2023.08.22=haa95532_0
18  - cachetools=4.2.2=pyhd3eb1b0_0
19  - certifi=2022.12.7=py37haa95532_0
20  - cffi=1.15.1=py37h2bbff1b_3
21  - charset-normalizer=2.0.4=pyhd3eb1b0_0
22  - click=8.0.4=py37haa95532_0
23  - colorama=0.4.6=py37haa95532_0
24  - cryptography=39.0.1=py37h2b164f_0
25  - fftw=3.3.9=h2bbff1b_1
26  - flatbuffers=2.0.0=h6c2663c_0
27  - frozenlist=1.3.3=py37h2bbff1b_0
28  - gast=0.4.0=pyhd3eb1b0_0
29  - gflags=5.2.1=h8cc25b3_3
30  - google-auth=2.6.0=pyhd3eb1b0_0
31  - google-auth-oauthlib=0.4.4=pyhd3eb1b0_0
32  - google-pasta=0.2.0=pyhd3eb1b0_0
33  - grpcio=1.42.0=py37hc60d5dd_0
34  - h5py=3.7.0=py37h3de5c98_0
35  - hdf5=1.10.6=h1756f20_1
36  - icc_rt=2022.1.0=h6049295_2
37  - icu=58.2=ha925a31_3
38  - idna=3.4=py37haa95532_0
39  - importlib-metadata=4.11.3=py37haa95532_0
40  - intel-openmp=2021.4.0=haa95532_3556
41  - jpeg=9e=h2bbff1b_1
42  - keras=2.10.0=py37haa95532_0
43  - keras-preprocessing=1.1.2=pyhd3eb1b0_0
44  - libcurl=8.2.1=h86230a5_0
45  - libpng=1.6.39=h8cc25b3_0
46  - libprotobuf=3.20.3=h23ce68f_0
47  - libssh2=1.10.0=hcd4344a_2
```

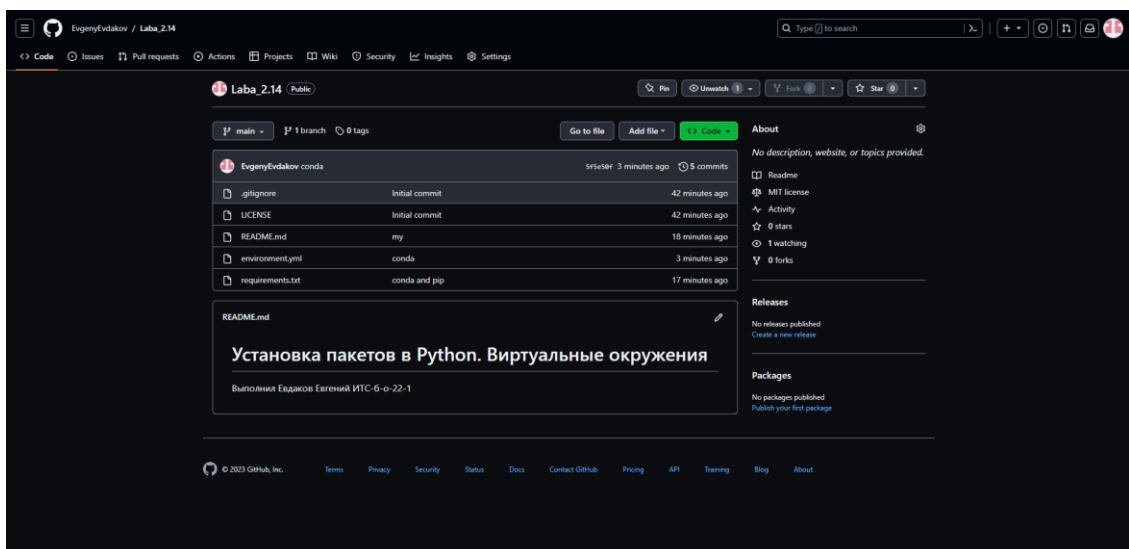
Рисунок 8. Содержимое файла

Далее сформировал файл `requirements.txt` он содержит список зависимостей и их версий в текстовом формате. Каждая строка представляет собой один пакет с указанием версии или диапазона версий.

```
requirements – Блокнот
Файл  Правка  Формат  Вид  Справка
abs1-py==1.4.0
astunparse==1.6.3
cachetools==5.3.1
certifi==2023.7.22
charset-normalizer==3.2.0
flatbuffers==23.5.26
gast==0.4.0
google-auth==2.22.0
google-auth-oauthlib==1.0.0
google-pasta==0.2.0
grpcio==1.58.0
h5py==3.9.0
idna==3.4
keras==2.13.1
libclang==16.0.6
Markdown==3.4.4
MarkupSafe==2.1.3
numpy==1.24.3
oauthlib==3.2.2
opt-einsum==3.3.0
packaging==23.1
protobuf==4.24.3
pyasn1==0.5.0
pyasn1-modules==0.3.0
requests==2.31.0
requests-oauthlib==1.3.1
rsa==4.9
six==1.16.0
tensorboard==2.13.0
tensorboard-data-server==0.7.1
tensorflow==2.13.0
tensorflow-estimator==2.13.0
tensorflow-intel==2.13.0
tensorflow-io-gcs-filesystem==0.31.0
termcolor==2.3.0
typing_extensions==4.5.0
urllib3==1.26.16
Werkzeug==2.3.7
wrapt==1.15.0
```

Рисунок 9. Содержимое файла requirements.txt

## Задание 7. Закомитил изменения и отправил на GitHub.



Ссылка: [https://github.com/EvgenyEvdakov/Laba\\_2.14](https://github.com/EvgenyEvdakov/Laba_2.14)

## Ответы на контрольные вопросы:

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

Для установки пакета Python, не входящего в стандартную библиотеку, можно воспользоваться менеджером пакетов `pip`. Для этого нужно выполнить команду `pip install package_name`

## **2. Как осуществить установку менеджера пакетов `pip`?**

`Pip` обычно устанавливается автоматически вместе с Python. Если он не установлен можно использовать 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)

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

Для установки последней версии пакета с помощью `pip`, нужно выполнить команду `pip install --upgrade package_name`

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

Для установки заданной версии пакета с помощью `pip`, нужно выполнить команду `pip install package_name==desired_version`

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

Для установки пакета из `git` репозитория, можно использовать команду `pip install git+URL`

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

Для установки пакета из локальной директории, нужно выполнить команду `pip install /path/to/package_directory`

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

Чтобы удалить установленный пакет с помощью `pip`, нужно выполнить команду `pip uninstall package_name`

## **9. Как обновить установленный пакет с помощью `pip`?**

Для обновления установленного пакета с помощью `pip`, нужно использовать команду `pip install --upgrade package_name`

#### **10. Как отобразить список установленных пакетов с помощью pip?**

Чтобы отобразить список установленных пакетов с помощью pip, нужно выполнить команду `pip list` или `pip freeze`.

#### **11. Каковы причины появления виртуальных окружений в языке Python?**

Виртуальные окружения в Python используются для изоляции проектов и их зависимостей. Основные причины их использования включают изоляцию проектов друг от друга, управление версиями зависимостей и предотвращение конфликтов между пакетами.

#### **12. Каковы основные этапы работы с виртуальными окружениями?**

1. Создание виртуального окружения.
2. Активация виртуального окружения.
3. Установка необходимых пакетов в виртуальное окружение.
4. Работа с проектом в активированном виртуальном окружении.
5. Деактивация виртуального окружения (по завершении работы).

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

- 1) Создание виртуального окружения: `python -m venv myenv`
- 2) Активация виртуального окружения
- 3) Установка пакетов: `pip install package_name`
- 4) Деактивация: `deactivate`

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

Работа с виртуальными окружениями с помощью `virtualenv` аналогична `venv`. Создаем окружение с помощью `virtualenv myenv`, активируем его, устанавливаем пакеты и деактивируем.

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



1) Создание виртуального окружения: `pipenv --python 3.8` (здесь указывается версия Python)

2) Установка пакетов: `pipenv install package_name`

3) Запуск оболочки в виртуальном окружении: `pipenv shell`

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

Файл `requirements.txt` используется для указания списка зависимостей проекта. Создать его можно с помощью команды `pip freeze > requirements.txt`, и он будет содержать список установленных пакетов и их версий.

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

Преимущества `conda` по сравнению с `pip` включают в себя возможность установки бинарных зависимостей, управление средами и зависимостями, а также поддержку многих языков программирования, не только Python.

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

Пакетный менеджер `conda` входит в дистрибутивы `Anaconda` и `Miniconda`. Он также может быть установлен отдельно на другие дистрибутивы Python.

**19. Как создать виртуальное окружение `conda`?**

Для создания виртуального окружения `conda` нужно использовать команду `conda create --name myenv`

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

Для активации и установки пакетов в виртуальное окружение `conda`, нужно выполнить команду `conda activate myenv`, а затем команду `conda install package_name` для установки пакетов.

**21. Как деактивировать и удалить виртуальное окружение `conda`?**

Чтобы деактивировать и удалить виртуальное окружение conda, нужно выполнить команду `conda deactivate` для деактивации, а затем `conda env remove --name myenv` для удаления.

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

Файл `environment.yml` используется для определения окружения conda, включая зависимости. Его можно создать вручную, указав пакеты и их версии, или автоматически с помощью команды `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`, нужно использовать команду `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 среде PyCharm можно создавать и управлять виртуальными окружениями conda через интерфейс IDE. Это делается через "File" -> "Settings" -> "Project: <your project>" -> "Python Interpreter

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

Файлы `requirements.txt` и `environment.yml` содержат информацию о зависимостях проекта и их версиях. Хранение их в репозитории Git позволяет другим разработчикам воссоздать окружение проекта и установить необходимые зависимости для работы с кодом.

**Вывод:** приобрел навыки по работе с менеджером пакетов `pip` и виртуальными окружениями с помощью языка программирования Python версии 3.x.