

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций  
«Установка пакетов в Python. Виртуальные окружения»**

**Отчет по лабораторной работе № 2.14  
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2023г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2023

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

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

**Owner \*** **Repository name \***

Yana-Kh / Lab-2.14-OPJ

Great repository names are short and memorable. Need inspiration? How about [super-barnacle?](#)

**Description (optional)**

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

**.gitignore template:** Python

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

**License:** MIT License

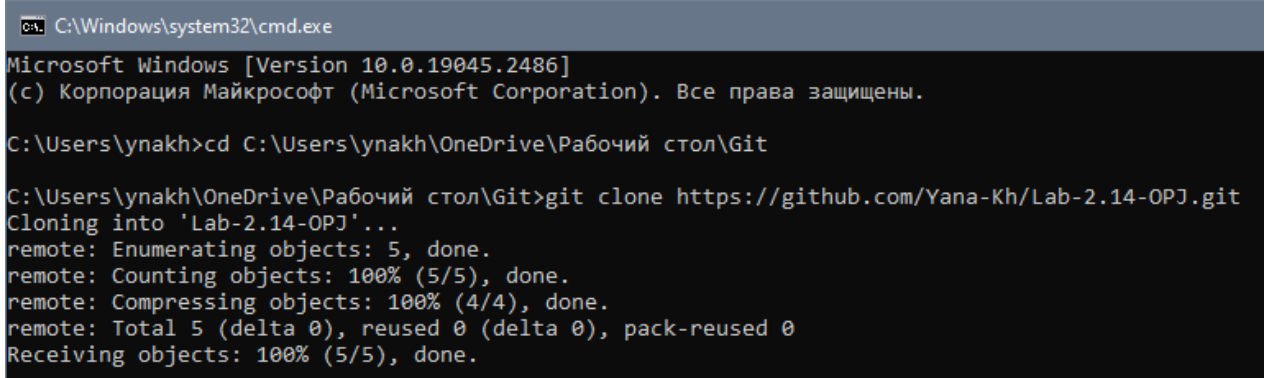
This will set **main** as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

**Creating repository...**

Рисунок 1 – Создание репозитория

### 3. Выполните клонирование созданного репозитория.

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The command prompt displays the following text: 'Microsoft Windows [Version 10.0.19045.2486] (c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.' The user has navigated to 'C:\Users\ynakh>cd C:\Users\ynakh\OneDrive\Рабочий стол\Git'. Then, they executed 'git clone https://github.com/Yana-Kh/Lab-2.14-OPJ.git'. The output shows the cloning process: 'Cloning into 'Lab-2.14-OPJ'...', 'remote: Enumerating objects: 5, done.', 'remote: Counting objects: 100% (5/5), done.', 'remote: Compressing objects: 100% (4/4), done.', 'remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0', and 'Receiving objects: 100% (5/5), done.'

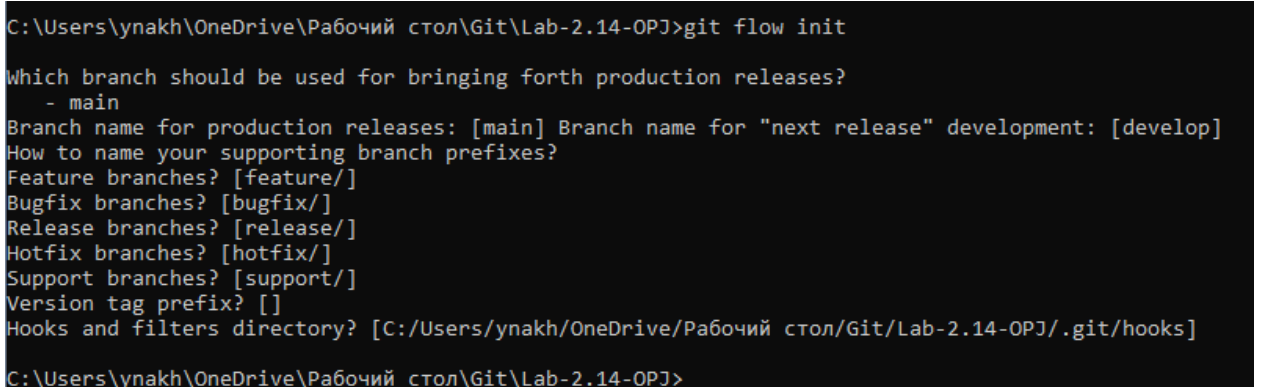
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.2486]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\ynakh>cd C:\Users\ynakh\OneDrive\Рабочий стол\Git

C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/Lab-2.14-OPJ.git
Cloning into 'Lab-2.14-OPJ'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 2 – Клонирование репозитория

### 4. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.14-OPJ'. The command prompt displays the following text: 'C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.14-OPJ>git flow init'. It then asks 'Which branch should be used for bringing forth production releases?' with a default of '- main'. The user has entered 'main'. It then asks 'Branch name for production releases: [main] Branch name for "next release" development: [develop]'. The user has entered 'develop'. It then asks 'How to name your supporting branch prefixes?'. The user has entered 'feature/'. It then asks 'Bugfix branches? [bugfix/]'. The user has entered 'bugfix/'. It then asks 'Release branches? [release/]'. The user has entered 'release/'. It then asks 'Hotfix branches? [hotfix/]'. The user has entered 'hotfix/'. It then asks 'Support branches? [support/]'. The user has entered 'support/'. It then asks 'Version tag prefix? []'. The user has entered '[]'. It then asks 'Hooks and filters directory? [C:/Users/ynakh/OneDrive/Рабочий стол/Git/Lab-2.14-OPJ/.git/hooks]'. The user has entered '[]'. The command prompt ends with 'C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.14-OPJ>'.

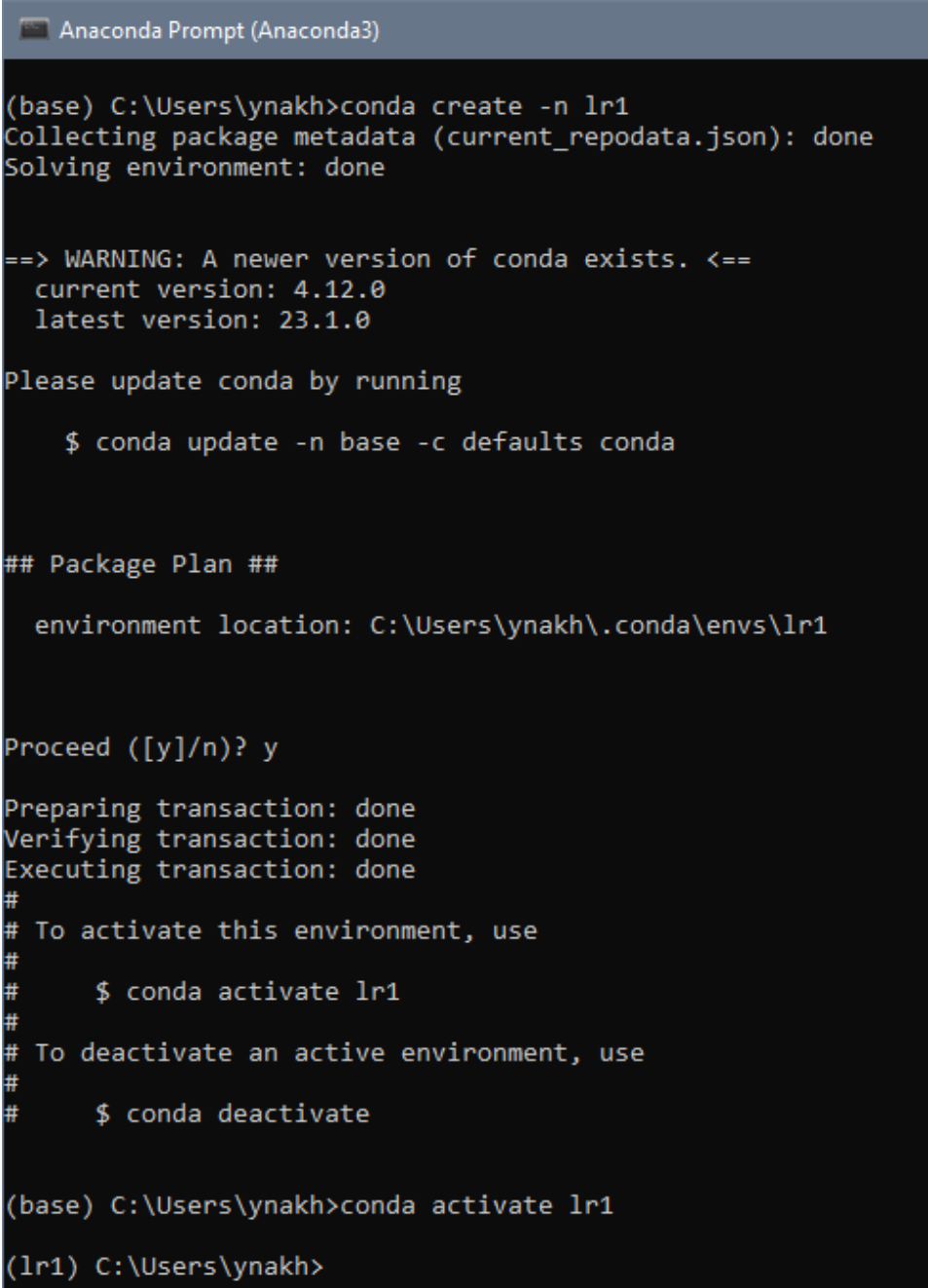
```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.14-OPJ>git flow init

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/ynakh/OneDrive/Рабочий стол/Git/Lab-2.14-OPJ/.git/hooks]

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.14-OPJ>
```

Рисунок 3 – Организация репозитория в соответствии с моделью git-flow

5. Создайте виртуальное окружение Anaconda с именем репозитория.



```
Anaconda Prompt (Anaconda3)

(base) C:\Users\ynakh>conda create -n lr1
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 23.1.0

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\ynakh\.conda\envs\lr1

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate lr1
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\ynakh>conda activate lr1

(lr1) C:\Users\ynakh>
```

Рисунок 4 – Создание виртуального окружения и его активация

6. Установите в виртуальное окружение следующие пакеты: `pip`, `NumPy`, `Pandas`, `SciPy`.

```
Anaconda Prompt (Anaconda3) - conda install -n lr1 pip
(lr1) C:\Users\ynakh>conda install -n lr1 pip
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 23.1.0

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\Users\ynakh\.conda\envs\lr1

added / updated specs:
- pip

The following packages will be downloaded:

package | build | size
-----|-----|-----
ca-certificates-2023.01.10 | haa95532_0 | 121 KB
certifi-2022.12.7 | py310haa95532_0 | 149 KB
libffi-3.4.2 | hd77b12b_6 | 109 KB
openssl-1.1.1s | h2bbff1b_0 | 5.5 MB
pip-22.3.1 | py310haa95532_0 | 2.8 MB
python-3.10.9 | h966fe2a_0 | 15.8 MB
setuptools-65.6.3 | py310haa95532_0 | 1.2 MB
sqlite-3.40.1 | h2bbff1b_0 | 889 KB
tk-8.6.12 | h2bbff1b_0 | 3.1 MB
tzdata-2022g | h04d1e81_0 | 114 KB
wincertstore-0.2 | py310haa95532_2 | 15 KB
xz-5.2.10 | h8cc25b3_1 | 520 KB
zlib-1.2.13 | h8cc25b3_0 | 113 KB
-----|-----|-----
Total: | 30.3 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-2023.01.10-haa95532_0
certifi | pkgs/main/win-64::certifi-2022.12.7-py310haa95532_0
libffi | pkgs/main/win-64::libffi-3.4.2-hd77b12b_6
openssl | pkgs/main/win-64::openssl-1.1.1s-h2bbff1b_0
pip | pkgs/main/win-64::pip-22.3.1-py310haa95532_0
python | pkgs/main/win-64::python-3.10.9-h966fe2a_0
setuptools | pkgs/main/win-64::setuptools-65.6.3-py310haa95532_0
sqlite | pkgs/main/win-64::sqlite-3.40.1-h2bbff1b_0
tk | pkgs/main/win-64::tk-8.6.12-h2bbff1b_0
tzdata | pkgs/main/noarch::tzdata-2022g-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/noarch::wheel-0.37.1-pyhd3eb1b0_0
wincertstore | pkgs/main/win-64::wincertstore-0.2-py310haa95532_2
xz | pkgs/main/win-64::xz-5.2.10-h8cc25b3_1
zlib | pkgs/main/win-64::zlib-1.2.13-h8cc25b3_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
setuptools-65.6.3 | 1.2 MB | ##### | 100%
tk-8.6.12 | 3.1 MB | ##### | 100%
xz-5.2.10 | 520 KB | ##### | 100%
certifi-2022.12.7 | 149 KB | ##### | 100%
sqlite-3.40.1 | 889 KB | ##### | 100%
libffi-3.4.2 | 109 KB | ##### | 100%
openssl-1.1.1s | 5.5 MB | ##### | 100%
python-3.10.9 | 15.8 MB | ##### | 100%
pip-22.3.1 | 2.8 MB | ##### | 100%
ca-certificates-2023 | 121 KB | ##### | 100%
tzdata-2022g | 114 KB | ##### | 100%
zlib-1.2.13 | 113 KB | ##### | 100%
wincertstore-0.2 | 15 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Рисунок 5 – Установка в виртуальное окружение пакета `pip`

```

(lr1) C:\Users\ynakh>conda install -n lr1 numpy
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 23.1.0

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\ynakh\.conda\envs\lr1

  added / updated specs:
    - numpy

The following packages will be downloaded:



| package           | build           |        |
|-------------------|-----------------|--------|
| mkl-service-2.4.0 | py310h2bbff1b_0 | 48 KB  |
| mkl_fft-1.3.1     | py310ha0764ea_0 | 136 KB |
| mkl_random-1.2.2  | py310h4ed8f06_0 | 221 KB |
| numpy-1.23.5      | py310h60c9a35_0 | 11 KB  |
| numpy-base-1.23.5 | py310h04254f7_0 | 6.0 MB |
| Total:            |                 | 6.4 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-2021.4.0-haa95532_3556
mkl                 pkgs/main/win-64::mkl-2021.4.0-haa95532_640
mkl-service         pkgs/main/win-64::mkl-service-2.4.0-py310h2bbff1b_0
mkl_fft             pkgs/main/win-64::mkl_fft-1.3.1-py310ha0764ea_0
mkl_random          pkgs/main/win-64::mkl_random-1.2.2-py310h4ed8f06_0
numpy               pkgs/main/win-64::numpy-1.23.5-py310h60c9a35_0
numpy-base         pkgs/main/win-64::numpy-base-1.23.5-py310h04254f7_0
six                 pkgs/main/noarch::six-1.16.0-pyhd3eb1b0_1

Proceed ([y]/n)? y

Downloading and Extracting Packages
mkl-service-2.4.0 | 48 KB | ##### | 100%
numpy-base-1.23.5 | 6.0 MB | ##### | 100%
numpy-1.23.5 | 11 KB | ##### | 100%
mkl_fft-1.3.1 | 136 KB | ##### | 100%
mkl_random-1.2.2 | 221 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(lr1) C:\Users\ynakh>

```

Рисунок 6 – Установка в виртуальное окружение пакета NumPy

```

(lr1) C:\Users\ynakh>conda install -n lr1 pandas
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 23.1.0

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\ynakh\.conda\envs\lr1

  added / updated specs:
    - pandas

The following packages will be downloaded:



| package          | build           |         |
|------------------|-----------------|---------|
| bottleneck-1.3.5 | py310h9128911_0 | 106 KB  |
| numexpr-2.8.4    | py310hd213c9f_0 | 128 KB  |
| packaging-22.0   | py310haa95532_0 | 68 KB   |
| pandas-1.5.2     | py310h4ed8f06_0 | 10.5 MB |
| pytz-2022.7      | py310haa95532_0 | 210 KB  |
| Total:           |                 | 11.0 MB |



The following NEW packages will be INSTALLED:

bottleneck      pkgs/main/win-64::bottleneck-1.3.5-py310h9128911_0
numexpr         pkgs/main/win-64::numexpr-2.8.4-py310hd213c9f_0
packaging       pkgs/main/win-64::packaging-22.0-py310haa95532_0
pandas          pkgs/main/win-64::pandas-1.5.2-py310h4ed8f06_0
python-dateutil pkgs/main/noarch::python-dateutil-2.8.2-pyhd3eb1b0_0
pytz            pkgs/main/win-64::pytz-2022.7-py310haa95532_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
pytz-2022.7      | 210 KB | ##### | 100%
numexpr-2.8.4   | 128 KB | ##### | 100%
pandas-1.5.2    | 10.5 MB | ##### | 100%
bottleneck-1.3.5 | 106 KB | ##### | 100%
packaging-22.0  | 68 KB  | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(lr1) C:\Users\ynakh>

```

Рисунок 7 – Установка в виртуальное окружение пакета Pandas

```
Anaconda Prompt (Anaconda3) - conda install -n lr1 pip - conda install -n lr1 numpy - conda install -n lr1 pandas - conda install -n lr1 scipy
(lr1) C:\Users\ynakh>conda install -n lr1 scipy
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 23.1.0

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\ynakh\.conda\envs\lr1

  added / updated specs:
    - scipy

The following packages will be downloaded:

  package                                     | build                                     |
  -----|-----|
  brotli-1.0.7                                | py310h2bbff1b_1002                     | 335 KB
  cffi-1.15.1                                 | py310h2bbff1b_3                         | 239 KB
  cryptography-38.0.4                         | py310h21b164f_0                         | 1.0 MB
  fftw-3.3.9                                  | h2bbff1b_1                             | 672 KB
  icc_rt-2022.1.0                             | h6049295_2                             | 6.5 MB
  idna-3.4                                     | py310haa95532_0                         | 97 KB
  pooch-1.4.0                                 | pyhd3eb1b0_0                           | 41 KB
  pyopenssl-22.0.0                           | pyhd3eb1b0_0                           | 50 KB
  pysocks-1.7.1                              | py310haa95532_0                         | 28 KB
  requests-2.28.1                            | py310haa95532_0                         | 101 KB
  scipy-1.10.0                                | py310hb9afe5d_0                         | 18.8 MB
  urllib3-1.26.14                            | py310haa95532_0                         | 195 KB
  win_inet_pton-1.1.0                        | py310haa95532_0                         | 9 KB
  -----|-----|
  Total:                                     |                                         | 28.0 MB

The following NEW packages will be INSTALLED:

  appdirs                pkgs/main/noarch::appdirs-1.4.4-pyhd3eb1b0_0
  brotli                 pkgs/main/win-64::brotli-1.0.7-py310h2bbff1b_1002
  cffi                   pkgs/main/win-64::cffi-1.15.1-py310h2bbff1b_3
  charset-normalizer     pkgs/main/noarch::charset-normalizer-2.0.4-pyhd3eb1b0_0
  cryptography           pkgs/main/win-64::cryptography-38.0.4-py310h21b164f_0
  fftw                   pkgs/main/win-64::fftw-3.3.9-h2bbff1b_1
  icc_rt                 pkgs/main/win-64::icc_rt-2022.1.0-h6049295_2
  idna                   pkgs/main/win-64::idna-3.4-py310haa95532_0
  pooch                  pkgs/main/noarch::pooch-1.4.0-pyhd3eb1b0_0
  pycparser              pkgs/main/noarch::pycparser-2.21-pyhd3eb1b0_0
  pyopenssl              pkgs/main/noarch::pyopenssl-22.0.0-pyhd3eb1b0_0
  pysocks                pkgs/main/win-64::pysocks-1.7.1-py310haa95532_0
  requests               pkgs/main/win-64::requests-2.28.1-py310haa95532_0
  scipy                  pkgs/main/win-64::scipy-1.10.0-py310hb9afe5d_0
  urllib3                pkgs/main/win-64::urllib3-1.26.14-py310haa95532_0
  win_inet_pton          pkgs/main/win-64::win_inet_pton-1.1.0-py310haa95532_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
pyopenssl-22.0.0 | 50 KB | ##### | 100%
requests-2.28.1 | 101 KB | ##### | 100%
pooch-1.4.0 | 41 KB | ##### | 100%
brotli-1.0.7 | 335 KB | ##### | 100%
win_inet_pton-1.1.0 | 9 KB | ##### | 100%
idna-3.4 | 97 KB | ##### | 100%
cffi-1.15.1 | 239 KB | ##### | 100%
icc_rt-2022.1.0 | 6.5 MB | ##### | 100%
scipy-1.10.0 | 18.8 MB | ##### | 100%
cryptography-38.0.4 | 1.0 MB | ##### | 100%
pysocks-1.7.1 | 28 KB | ##### | 100%
fftw-3.3.9 | 672 KB | ##### | 100%
urllib3-1.26.14 | 195 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(lr1) C:\Users\ynakh>
```

Рисунок 8 – Установка в виртуальное окружение пакета SciPy



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

```
(lr1) C:\Users\ynakh>conda install -n lr1 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
```

Рисунок 9 – Попытка установки пакета TensorFlow

Пакет TensorFlow совместим с Python 3.8, для решения возможной проблемы необходимо понизить версию python с помощью команды:

```
conda install python=3.7
```

8. Попробуйте установить пакет TensorFlow с помощью менеджера пакетов pip.

```
(lr1) C:\Users\ynakh>pip install tensorflow
Requirement already satisfied: tensorflow in c:\users\ynakh\.conda\envs\lr1\lib\site-packages (2.10.0)
Requirement already satisfied: keras<2.11,>=2.10.0 in c:\users\ynakh\.conda\envs\lr1\lib\site-packages (from tensorflow) (2.10.0)
Collecting tensorflow-io-gcs-filesystem>=0.23.1
  Downloading tensorflow_io_gcs_filesystem-0.30.0-cp310-cp310-win_amd64.whl (1.5 MB)
----- 1.5/1.5 MB 3.5 MB/s eta 0:00:00
Requirement already satisfied: h5py>=2.9.0 in c:\users\ynakh\.conda\envs\lr1\lib\site-packages (from tensorflow) (3.7.0)
Requirement already satisfied: tensorboard<2.11,>=2.10 in c:\users\ynakh\.conda\envs\lr1\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\ynakh\.conda\envs\lr1\lib\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\ynakh\.conda\envs\lr1\lib\site-packages (from tensorflow) (0.2.0)
Collecting protobuf<3.20,>=3.9.2
  Downloading protobuf-3.19.6-cp310-cp310-win_amd64.whl (895 kB)
----- 895.7/895.7 kB 4.7 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\ynakh\.conda\envs\lr1\lib\site-packages (from tensorflow) (4.4.0)
Requirement already satisfied: numpy>=1.20 in c:\users\ynakh\.conda\envs\lr1\lib\site-packages (from tensorflow) (1.23.5)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\ynakh\.conda\envs\lr1\lib\site-packages (from tensorflow) (1.14.1)
Collecting libclang>=13.0.0
  Downloading libclang-15.0.6.1-py2.py3-none-win_amd64.whl (23.2 MB)
----- 23.2/23.2 MB 6.2 MB/s eta 0:00:00
```

Рисунок 8 – Установка в виртуальное окружение пакета SciPy

9. Сформируйте файлы requirements.txt и environment.yml. Проанализируйте содержимое этих файлов.

Файл requirements.txt хранит в себе список всех установленных в интерпретатор сторонних пакетов, чтобы потом можно было оперативно и просто установить такой же набор. Команда pip freeze выводит интересующий нас список, но при желании это можно сделать вручную.

```
(lr1) C:\Users\ynakh>pip freeze > requirements.txt
```

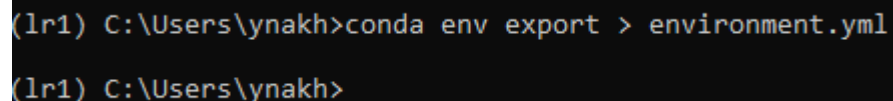
Рисунок 9 – Создание файла requirements.txt



```
requirements.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
abs1-py @ file:///C:/b/abs_5babsu7y5x/croot/abs1-py_1666362945682/work
aiohttp @ file:///C:/b/abs_c4zmy2l696/croot/aiohttp_1670009573673/work
aiosignal @ file:///tmp/build/80754af9/aiosignal_1637843061372/work
appdirs==1.4.4
astunparse==1.6.3
async-timeout @ file:///C:/b/abs_43ozhz2a8g/croots/recipe/async-timeout_1664876362767/work
attrs @ file:///C:/b/abs_09s3y775ra/croot/attrs_1668696195628/work
blinker==1.4
bottle==0.12.23
bottle-websocket==0.2.9
Bottleneck @ file:///C:/Windows/Temp/abs_3198ca53-903d-42fd-87b4-03e6d03a8381yfwsuve8/croots/recipe/bottleneck_1657175565403/work
brotlipy==0.7.0
cachetools @ file:///tmp/build/80754af9/cachetools_1619597386817/work
certifi @ file:///C:/b/abs_85o_6fm0se/croot/certifi_1671487778835/work/certifi
cffi==1.15.1
charset-normalizer @ file:///tmp/build/80754af9/charset-normalizer_1630003229654/work
click==8.1.3
colorama==0.4.6
cryptography @ file:///C:/b/abs_b7d7drzbky/croot/cryptography_1673298763653/work
Eel==0.15.1
Flask==2.2.2
flatbuffers @ file:///home/ktietz/cip/python-flatbuffers_1634039120618/work
flit_core @ file:///opt/conda/conda-bld/flit-core_1644941570762/work/source/flit_core
frozenlist @ file:///C:/b/abs_2bb5uzghsi/croot/frozenlist_1670004511812/work
future==0.18.2
gast @ file:///Users/ktietz/demo/mc3/conda-bld/gast_1628588903283/work
gevent==22.10.2
gevent-websocket==0.10.1
google-auth @ file:///opt/conda/conda-bld/google-auth_1646735974934/work
google-auth-oauthlib @ file:///tmp/build/80754af9/google-auth-oauthlib_1617120569401/work
google-pasta @ file:///Users/ktietz/demo/mc3/conda-bld/google-pasta_1630577991354/work
greenlet==2.0.1
```

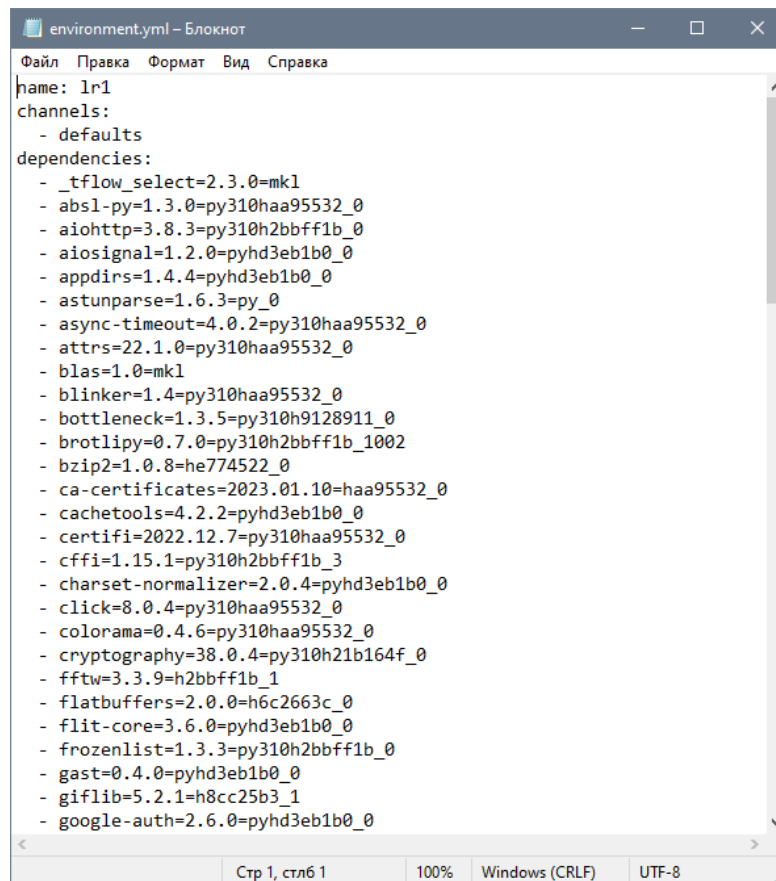
Рисунок 10 – Содержание requirements.txt

Совместное использование среды проекта между платформами и операционными системами также можно выполнить с помощью опции - export для создания файла environment.yml. Разница между списком спецификаций и файлом environment.yml заключается в том, что последний не зависит от операционной системы и форматируется с использованием YAML, он содержит имя, зависимости и установочные каналы.



```
(lr1) C:\Users\ynakh>conda env export > environment.yml
(lr1) C:\Users\ynakh>
```

Рисунок 11 – Создание файла environment.yml



```
environment.yml – Блокнот
Файл  Правка  Формат  Вид  Справка
name: lr1
channels:
  - defaults
dependencies:
  - _tfflow_select=2.3.0=mk1
  - absl-py=1.3.0=py310haa95532_0
  - aiohttp=3.8.3=py310h2bbff1b_0
  - aiosignal=1.2.0=pyhd3eb1b0_0
  - appdirs=1.4.4=pyhd3eb1b0_0
  - astunparse=1.6.3=py_0
  - async-timeout=4.0.2=py310haa95532_0
  - attrs=22.1.0=py310haa95532_0
  - blas=1.0=mk1
  - blinker=1.4=py310haa95532_0
  - bottleneck=1.3.5=py310h9128911_0
  - brotliipy=0.7.0=py310h2bbff1b_1002
  - bzip2=1.0.8=he774522_0
  - ca-certificates=2023.01.10=haa95532_0
  - cachetools=4.2.2=pyhd3eb1b0_0
  - certifi=2022.12.7=py310haa95532_0
  - cffi=1.15.1=py310h2bbff1b_3
  - charset-normalizer=2.0.4=pyhd3eb1b0_0
  - click=8.0.4=py310haa95532_0
  - colorama=0.4.6=py310haa95532_0
  - cryptography=38.0.4=py310h21b164f_0
  - fftw=3.3.9=h2bbff1b_1
  - flatbuffers=2.0.0=h6c2663c_0
  - flit-core=3.6.0=pyhd3eb1b0_0
  - frozenlist=1.3.3=py310h2bbff1b_0
  - gast=0.4.0=pyhd3eb1b0_0
  - giflib=5.2.1=h8cc25b3_1
  - google-auth=2.6.0=pyhd3eb1b0_0
Стр 1, столб 1    100%    Windows (CRLF)    UTF-8
```

Рисунок 11 – Содержание environment.yml

## 10. Зафиксируйте сделанные изменения в репозитории.






	Yana-Kh upd
	.gitignore Initial commit
	LICENSE Initial commit
	README.md Initial commit
	environment.yml upd
	requirements.txt upd

Рисунок 12 – Фиксирование изменений в репозитории

## Вопросы для защиты работы

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

Существует Python Package Index (PyPI) – это репозиторий, открытый для всех разработчиков, в нём можно найти пакеты для решения практических задач.

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

`Pip` – это консольная утилита (без графического интерфейса). После того, как вы её скачаете и установите, она пропишется в PATH и будет доступна для использования.

Чтобы установить утилиту `pip`, нужно также скачать скрипт `get-pip.py`.

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

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

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

Используя команду «`pip install ProjectName`»

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

Используя команду «`pip install ProjectName==3.2`» (где 3.2 – нужная версия пакета)

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

Используя команду «`pip install -e git+https://gitrepo.com/ ProjectName.git`»

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

Используя команду «`pip install ./dist/ProjectName.tar.gz`»

8. Как удалить установленный пакет с помощью pip?

Используя команду «`pip uninstall ProjectName`»

9. Как обновить установленный пакет с помощью pip?

Используя команду «`pip install --upgrade ProjectName`»

10. Как отобразить список установленных пакетов с помощью pip?

Используя команду «`pip list`»

11. Каковы причины появления виртуальных окружений в языке Python?

В системе для интерпретатора Python может быть установлена глобально только одна версия пакета. Это порождает ряд трудностей. Основная задача виртуальных окружений – это решение проблем с совместимостью и коллективной разработкой. Так можно не бояться нарушить работу ОС, таких как Linux и MacOS, обновив какой-либо пакет, а также свободно устанавливать различные пакеты, не боясь, что другие члены команды столкнутся с трудностями, связанными с отсутствием тех или иных пакетов при попытке запустить проект.

12. Каковы основные этапы работы с виртуальными окружениями?

1. Создание нового виртуальное окружение в отдельной папке для выбранной версии интерпретатора Python.
2. Активизация окружения.
3. Осуществляется работа.
4. Деактивируем после окончания работы виртуальное окружение.
5. Удаляем папку с виртуальным окружением, если оно нам больше не нужно.

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

Для создания виртуального окружения достаточно дать команду в формате: «python 3 -m venv <путь к папке виртуального окружения>»

Чтобы активировать виртуальное окружение под Windows нужно дать команду: «> env\\Scripts\\activate».

После активации приглашение консоли изменится. В его начале в круглых скобках будет отображаться имя папки с виртуальным окружением.

При размещении виртуального окружения в папке проекта стоит позаботиться об его исключении из репозитория системы управления версиями. Для этого, например, при использовании Git нужно добавить папку в файл .gitignore. Это делается для того, чтобы не засорять проект разными вариантами виртуального окружения.

Чтобы переключиться с одного окружения на другое нам нужно выполнить команду деактивации и команду активации другого виртуального окружения.

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

Вначале командой «python3 -m pip install virtualenv Virtualenv» необходимо установить пакет, который позволяет создать абсолютно изолированное виртуальное окружение для каждой из программ.

Создание виртуального окружения с утилитой virtualenv отличается от стандартного. Например, создание в текущей папке виртуального окружения для интерпретатора доступного через команду python3 с названием папки окружения env: «virtualenv -p python3 env». Активация и деактивация такая же, как у стандартной утилиты Python.

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

Грубо говоря, `pipenv` можно рассматривать как симбиоз утилит `pip` и `venv` (или `virtualenv`), которые работают вместе, пряча многие неудобные детали от конечного пользователя.

Основные возможности `pipenv`: создание и управление виртуальным окружением; синхронизация пакетов в `Pipfile` при установке и удалении пакетов; автоматическая подгрузка переменных окружения из `.env` файла.

Установка: `pip install --user pipenv`

`Pipenv` использует свой собственный формат файла для описания зависимостей проекта — `Pipfile`. Этот файл имеет формат TOML. В принципе его можно редактировать руками, но `pipenv` достаточно неплохо и сам умеет обновлять этот файл, когда вы просто работаете с утилитой через командную строку. Структуру этого файла рассмотрим чуть позже.

В паре с `Pipfile` идёт `Pipfile.lock`. Он имеет формат JSON и не предназначен для редактирования руками. Этот файл хранит контрольные суммы пакетов, которые вы устанавливаете в проект, что даёт гарантию, что развёрнутые на разных машинах окружения будут идентичны друг другу. `pipenv` автоматически обновляет контрольные суммы в этом файле, когда вы устанавливаете или обновляете зависимости. При развёртывании окружения `pipenv` сверит сохранённые контрольные суммы с фактически получившимися, и в случае чего уведомит вас, что развёртывание не удалось. Это очень важный плюс в копилку `pipenv` по сравнению с `pip`.

Оба этих файла можно и нужно сохранять в системе контроля версий (`git`).

Вообще, идею использовать два файла для описания зависимостей нельзя назвать новой. Здесь явно прослеживается параллель между `Gemfile` и `Gemfile.lock` из мира Ruby и `package.json` и `package-lock.json` из мира JavaScript. Все эти файлы имеют схожее назначение.

Создание проекта: `pipenv --three`

После выполнения команды, `pipenv` создал файл `Pipfile` и виртуальное окружение где-то в заранее определенной директории (по умолчанию вне директории проекта).

Содержание файла:

```
[[source]]
name = "pypi"
url = "https://pypi.org/simple"
verify_ssl = true
```

```
[dev-packages]
```

```
[packages]
```

```
[requires]
python_version = "3.8"
```

Установка зависимости: `pipenv install requests`

Удаление зависимостей: `pipenv uninstall`

Обновление зависимостей: `pipenv update`

Удаление виртуального окружения: `pipenv --rm`

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

Файл `requirements.txt` хранит в себе список всех установленных в интерпретатор сторонних пакетов, чтобы потом можно было оперативно и просто установить такой же набор. Его можно написать вручную, а можно автоматически с помощью: `pip freeze > requirements.txt`.

Установить пакеты можно с помощью команды: `pip install -r requirements.txt`

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

`Pip` – это менеджер пакетов, который облегчает установку, обновление и удаление пакетов `python`. Он также работает с виртуальными средами `python`.



Conda – это менеджер пакетов для любого программного обеспечения (установка, обновление и удаление). Он также работает с виртуальными средами system.

Conda способна управлять пакетами как для Python, так и для C/ C++, R, Ruby, Lua, Scala и других. Conda устанавливает двоичные файлы, поэтому работу по компиляции пакета самостоятельно выполнять не требуется (по сравнению с pip).

18. В какие дистрибутивы Python входит пакетный менеджер conda?

Anaconda, Miniconda, PyCharm.

19. Как создать виртуальное окружение conda?

Используя команду «conda create -n <имя виртуального окружения>»

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

Для активации использовать команду: «conda activate <имя виртуального окружения>»

Для установки: conda install <имя виртуального окружения> <имя пакета>»

21. Как деактивировать и удалить виртуальное окружение conda?

Для деактивации использовать команду: conda deactivate <имя виртуального окружения>, а для удаления: conda remove -n <имя виртуального окружения>.

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

Файл `environment.yml` позволит воссоздать окружение. Он не зависит от операционной системы и форматируется с использованием YAML, содержит имя, зависимости и установочные каналы.

Создание файла: `conda env export > environment.yml`

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

Необходимо 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.

Работа с виртуальными окружениями в PyCharm зависит от способа взаимодействия с виртуальным окружением:

Создаём проект со своим собственным виртуальным окружением, куда затем будут устанавливаться необходимые библиотеки.

Предварительно создаём виртуальное окружение, куда установим нужные библиотеки. И затем при создании проекта в PyCharm можно будет его выбирать, т.е. использовать для нескольких проектов.

Для первого способа ход работы, следующий: запускаем PyCharm и в окне приветствия выбираем `Create New Project`. В мастере создания проекта, указываем в поле `Location` путь расположения создаваемого проекта. Имя конечной директории также является именем проекта. Далее разворачиваем параметры окружения, щелкая по `Project Interpreter`. И выбираем `New environment using Virtualenv`. Путь расположения окружения генерируется автоматически. И нажимаем на `Create`. Теперь установим библиотеки, которые будем использовать в программе. С помощью главного меню переходим в настройки `File → Settings`. Где переходим в `Project: project_name → Project Interpreter`. Выходим из настроек. Для запуска программы, необходимо создать профиль с конфигурацией. Для этого в верхнем правом углу нажимаем на

кнопку Add Configuration. Откроется окно Run/Debug Configurations, где нажимаем на кнопку с плюсом (Add New Configuration) в правом верхнем углу и выбираем Python. Далее указываем в поле Name имя конфигурации и в поле Script path расположение Python файла с кодом программы. В завершение нажимаем на Apply, затем на ОК. Для второго способа необходимо сделать следующее: на экране приветствия в нижнем правом углу через Configure → Settings переходим в настройки. Затем переходим в раздел Project Interpreter.

В верхнем правом углу есть кнопка с шестерёнкой, нажимаем на неё и выбираем Add, создавая новое окружение. И указываем расположение для нового окружения. Нажимаем на ОК. Далее в созданном окружении устанавливаем нужные пакеты. И выходим из настроек. В окне приветствия выбираем Create New Project. В мастере создания проекта, указываем имя расположения проекта в поле Location. Разворачиваем параметры окружения, щелкая по Project Interpreter, где выбираем Existing interpreter и указываем нужное нам окружение. Далее создаем конфигурацию запуска программы, также как создавали для раннее. После чего можно выполнить программу.

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

Чтобы другие пользователи могли ознакомиться с набором пакетов, используемом в проекте и воссоздать виртуальное окружение на своем устройстве.