

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 программной инженерии»**

Выполнил:  
Соколов Михаил Романович  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Руководитель практики:  
Богданов С.С., ассистент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2024 г.

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 3.x.

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и .gitignore файл для языка 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.](#)

*Required fields are marked with an asterisk (\*).*

**Repository template**

No template ▾

Start your repository with a template repository's contents.

**Owner \*** PoTtaTto ▾ / **Repository name \*** fundamentals\_of\_software

✔ fundamentals\_of\_software\_engineering\_lab2\_14 is available.

Great repository names are short and memorable. Need inspiration? How about curly-enigma ?

**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:**

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

**Add .gitignore**

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

License: MIT License ▾

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

📘 You are creating a public repository in your personal account.

**Create repository**

Рисунок 1 – Создание репозитория с заданными настройками

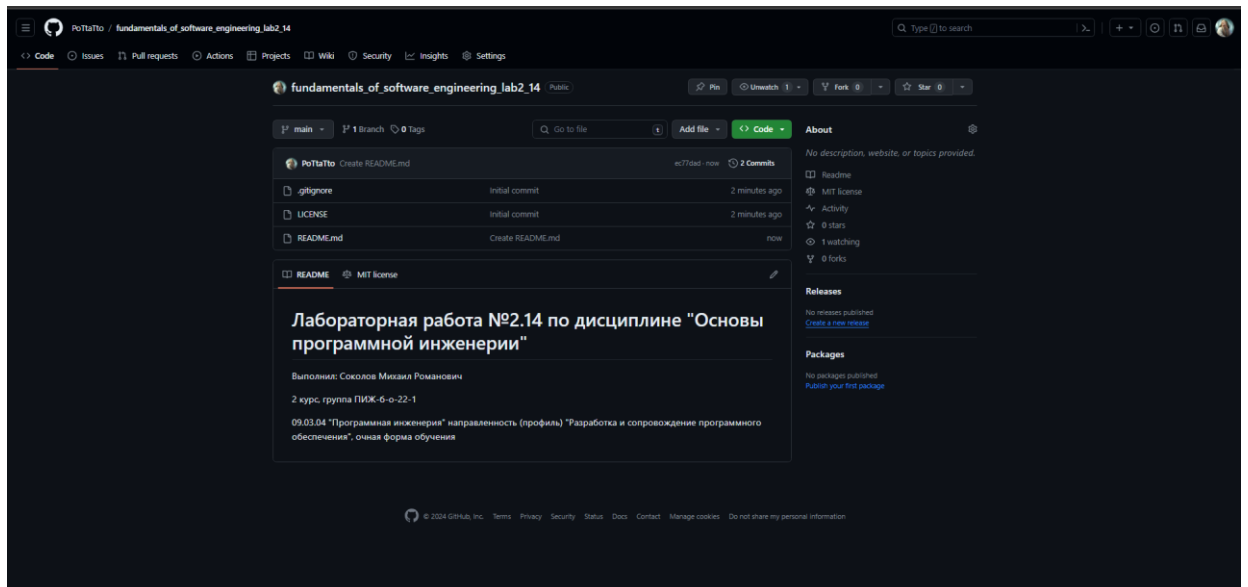


Рисунок 2 – Созданный репозиторий

```
Windows PowerShell
PS C:\Study\NCFU\Основы программной инженерии\2.14> git clone https://github.com/PoTtaTto/fundamentals_of_software_engineering_lab2_14
Cloning into 'fundamentals_of_software_engineering_lab2_14'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), 4.07 KiB | 4.07 MiB/s, done.
Resolving deltas: 100% (1/1), done.
PS C:\Study\NCFU\Основы программной инженерии\2.14> |
```

Рисунок 3 – Клонирование репозитория

```
Terminal Local x + v
PS C:\Study\NCFU\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Study\NCFU\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> |
```

Рисунок 4 – Создание ветки develop, где будут происходить изменения проекта до его полного релиза

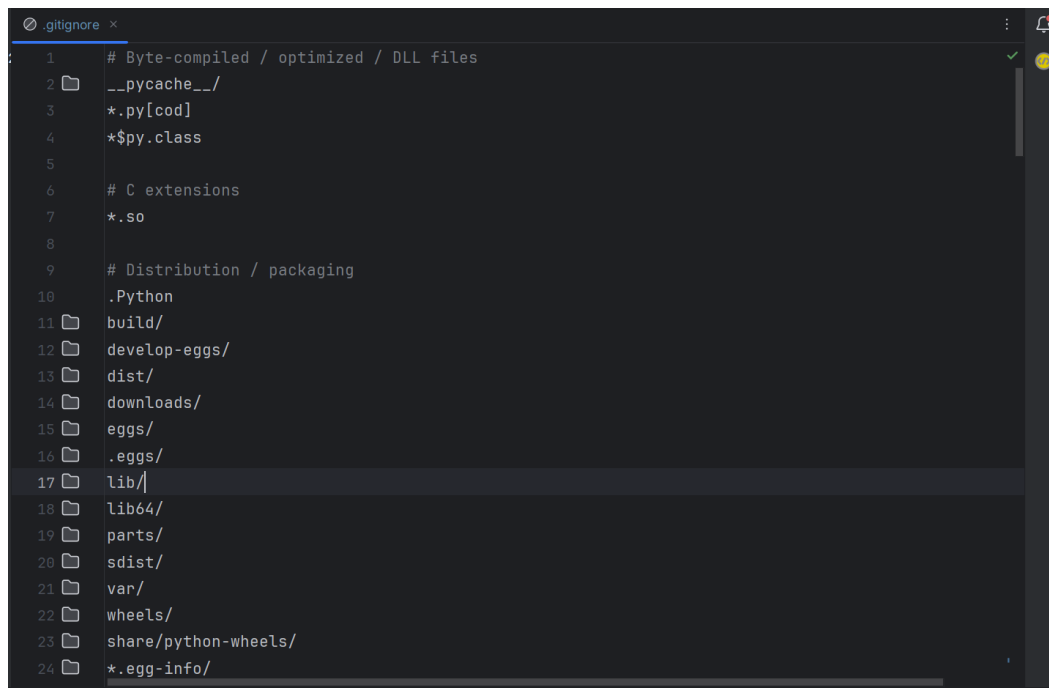


Рисунок 5 – Часть .gitignore, созданного GitHub

2. Создадим и активируем виртуальное окружение Anaconda с именем репозитория:

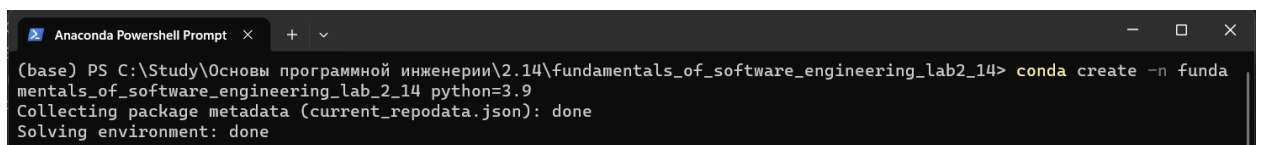


Рисунок 6 – Создание виртуального окружения

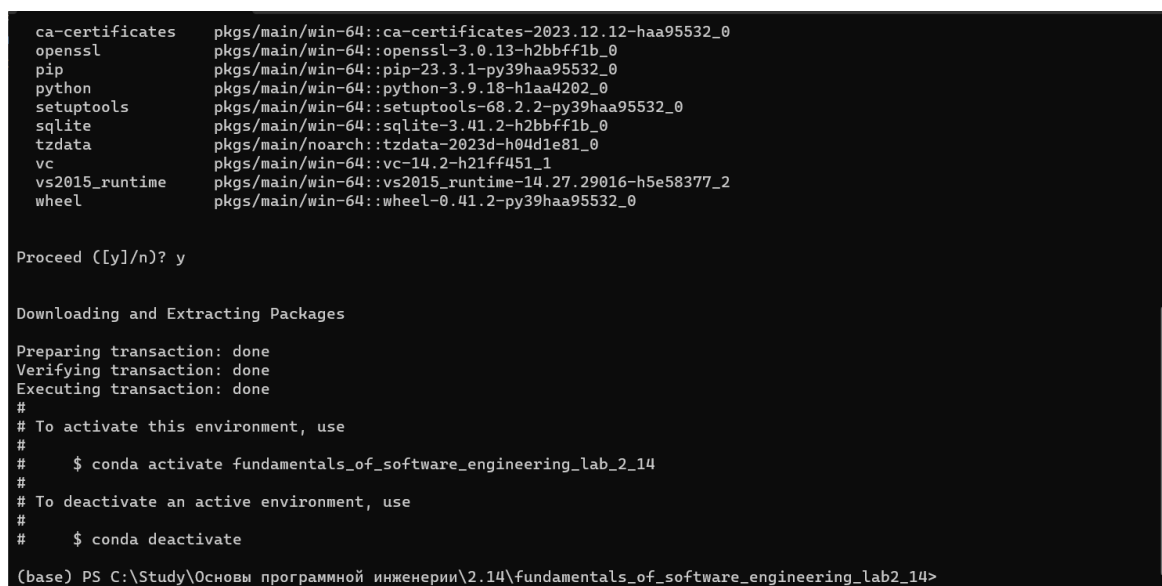


Рисунок 7 – Успешная установка

```
(base) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> conda activate fundam
entals_of_software_engineering_lab_2_14
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_e
ngineering_lab2_14> |
```

Рисунок 8 – Активация виртуального окружения

### 3. Установим пакеты pip, NumPy, Pandas и SciPy:

```
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14>
conda install pip, numpy, pandas, scipy
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: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.1.1

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=24.1.1

## Package Plan ##

environment location: C:\Users\MrPot\.conda\envs\fundamentals_of_software_engineering_lab_2_14

added / updated specs:
```

Рисунок 9 – Начало установки нужных пакетов

```
icc_rt          pkgs/main/win-64::icc_rt-2022.1.0-h6049295_2
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-py312h2bbff1b_1
mkl_fft         pkgs/main/win-64::mkl_fft-1.3.8-py312h2bbff1b_0
mkl_random      pkgs/main/win-64::mkl_random-1.2.4-py312h59b6b97_0
numexpr         pkgs/main/win-64::numexpr-2.8.7-py312h96b7d27_0
numpy           pkgs/main/win-64::numpy-1.26.3-py312hfd52020_0
numpy-base      pkgs/main/win-64::numpy-base-1.26.3-py312h4dde369_0
pandas          pkgs/main/win-64::pandas-2.1.4-py312hc7c4135_0
python-dateutil pkgs/main/noarch::python-dateutil-2.8.2-pyhd3eb1b0_0
python-tzdata   pkgs/main/noarch::python-tzdata-2023.3-pyhd3eb1b0_0
pytz            pkgs/main/win-64::pytz-2023.3.post1-py312haa95532_0
scipy           pkgs/main/win-64::scipy-1.11.3-py312h3d2928d_0
six             pkgs/main/noarch::six-1.16.0-pyhd3eb1b0_1
tbb             pkgs/main/win-64::tbb-2021.8.0-h59b6b97_0

Proceed ([y]/n)? y

Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14>|
```

Рисунок 10 – Завершение установки пакетов

### 4. Попробуем установить пакет TensorFlow:

```
Anaconda Powershell Prompt x + v
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_e
ngineering_lab2_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: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.1.1

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=24.1.1

## Package Plan ##

environment location: C:\Users\MrPot\.conda\envs\fundamentals_of_software_engineering_lab_2_14

added / updated specs:
- tensorflow
```

Рисунок 11 – Попытка установки с помощью утилиты conda

```
tensorflow          pkgs/main/win-64::tensorflow-2.10.0-mkl_py39ha510bab_0
tensorflow-base     pkgs/main/win-64::tensorflow-base-2.10.0-mkl_py39h6a7f48e_0
tensorflow-estima-  pkgs/main/win-64::tensorflow-estimator-2.10.0-py39haa95532_0
termcolor           pkgs/main/win-64::termcolor-2.1.0-py39haa95532_0
typing_extensions   pkgs/main/win-64::typing_extensions-4.9.0-py39haa95532_1
urllib3             pkgs/main/win-64::urllib3-2.1.0-py39haa95532_0
werkzeug            pkgs/main/win-64::werkzeug-2.3.8-py39haa95532_0
wrap               pkgs/main/win-64::wrap-1.14.1-py39h2bbff1b_0
yarl                pkgs/main/win-64::yarl-1.9.3-py39h2bbff1b_0
zipp                pkgs/main/win-64::zipp-3.17.0-py39haa95532_0
zlib                pkgs/main/win-64::zlib-1.2.13-h8cc25b3_0

The following packages will be DOWNGRADED:
  openssl            3.0.13-h2bbff1b_0 --> 1.1.1w-h2bbff1b_0
  python             3.9.18-h1aa4202_0 --> 3.9.18-h6244533_0

Proceed ([y]/n)? y

Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_e
ngineering_lab2_14> |
```

Рисунок 12 – Успешная установка пакета

```
Executing transaction: done
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_e
ngineering_lab2_14> pip install tensorflow
Requirement already satisfied: tensorflow in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\li
b\site-packages (2.10.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_1
4\lib\site-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_
2_14\lib\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2
_14\lib\site-packages (from tensorflow) (2.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_la
b_2_14\lib\site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_la
b_2_14\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\l
ib\site-packages (from tensorflow) (3.9.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\users\mrpot\.conda\envs\fundamentals_of_software_enginee
ring_lab_2_14\lib\site-packages (from tensorflow) (1.1.2)
Collecting libclang>=13.0.0 (from tensorflow)
  Downloading libclang-16.0.6-py2.py3-none-win_amd64.whl.metadata (5.3 kB)
Requirement already satisfied: numpy>=1.20 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\l
ib\site-packages (from tensorflow) (1.26.3)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_
2_14\lib\site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib
\site-packages (from tensorflow) (23.1)
Collecting protobuf<3.20,>=3.9.2 (from tensorflow)
  Downloading protobuf-3.19.6-cp39-cp39-win_amd64.whl (895 kB)
      895.9/895.9 kB 2.8 MB/s eta 0:00:00
```

Рисунок 13 – Переустановка пакета с помощью pip

```
Anaconda Powershell Prompt
Requirement already satisfied: importlib-metadata>=4.4 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib\site-packages (from markdown>=2.6.8->tensorboard<2.11,>=2.10->tensorflow) (7.0.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (2024.2.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.11,>=2.10->tensorflow) (2.1.3)
Requirement already satisfied: zipp>=0.5 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib\site-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.11,>=2.10->tensorflow) (3.17.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\mrpot\.conda\envs\fundamentals_of_software_engineering_lab_2_14\lib\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.11,>=2.10->tensorflow) (3.2.2)
Downloading libclang-16.0.6-py2.py3-none-win_amd64.whl (24.4 MB)
24.4/24.4 MB 11.1 MB/s eta 0:00:00
Installing collected packages: libclang, tensorflow-io-gcs-filesystem, protobuf
Attempting uninstall: protobuf
Found existing installation: protobuf 3.20.3
Uninstalling protobuf-3.20.3:
Successfully uninstalled protobuf-3.20.3
Successfully installed libclang-16.0.6 protobuf-3.19.6 tensorflow-io-gcs-filesystem-0.31.0
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14>
```

Рисунок 14 – Завершение переустановки с помощью pip

## 5. Сформируем файлы requirements.txt и environment.yml:

```
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> pip freeze > requirements.txt
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> conda env export > environment.yml
(fundamentals_of_software_engineering_lab_2_14) PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> |
```

Рисунок 15 – Создание файлов

```
pyparser @ file:///tmp/build/80754af9/pyparser_1636541352034/work
PyJWT @ file:///C:/ci/pyjwt_1657511236979/work
pyOpenSSL @ file:///C:/b/abs_08f38zyck4/croot/pyopenssl_1690225407403/work
python-dateutil @ file:///tmp/build/80754af9/python-dateutil_1626374649649/work
pytz @ file:///C:/b/abs_19q3ljkez4/croot/pytz_1695131651401/work
requests @ file:///C:/b/abs_474vaa3x9e/croot/requests_1707355619957/work
requests-oauthlib==1.3.0
rsa @ file:///tmp/build/80754af9/rsa_1614366226499/work
scipy==1.11.4
six @ file:///tmp/build/80754af9/six_1644875935023/work
tensorboard @ file:///C:/Users/builder/adi Pietro/mc3/tf210/conda-bld/tensorboard_1669760968711/work/tensorboard-2.10.0-py3-none-any.whl
tensorboard-data-server @ file:///C:/b/abs_2fhvpo862s/croot/tensorboard-data-server_1670853600144/work/tensorboard_data_server-0.6.1-py3-none-any.whl
tensorboard-plugin-wit @ file:///C:/tf/b/tensorboard-plugin-wit_1660162132996/work/tensorboard_plugin_wit-1.8.1-py3-none-any.whl
tensorflow==2.10.0
tensorflow-estimator @ file:///C:/Users/builder/adi Pietro/mc3/tf210/conda-bld/tensorflow-estimator_1669761460695/work/tensorflow_estimator-2.10.0-py2.py3-none-any.whl
tensorflow-io-gcs-filesystem==0.31.0
termcolor @ file:///C:/b/abs_16qe7jhw7n/croot/termcolor_1668084642458/work
typing_extensions @ file:///C:/b/abs_72cdotwc_6/croot/typing_extensions_1705599364138/work
tzdata @ file:///croot/python-tzdata_1690578112552/work
urllib3 @ file:///C:/b/abs_a9n398cvep/croot/urllib3_1700840517182/work
Werkzeug @ file:///C:/b/abs_3901hwqcw0/croot/werkzeug_1706210134718/work
wrapt @ file:///C:/Windows/Temp/abs_7c3dd407-1390-477a-b542-fd15df6a24085_diwiza/croots/recipe/wrapt_1657814452175/work
yarl @ file:///C:/b/abs_8bxwdyhjvp/croot/yarl_1701105248152/work
zipp @ file:///C:/b/abs_b0beoc270a/croot/zipp_1704206963359/work
```

Рисунок 16 – Часть файла requirements.txt

```

name: fundamentals_of_software_engineering_lab_2_14
channels:
  - defaults
dependencies:
  - _tfselect=2.3.0=mkl
  - absl-py=1.4.0=py39haa95532_0
  - aiohttp=3.9.3=py39h2bbff1b_0
  - aiosignal=1.2.0=pyhd3eb1b0_0
  - astunparse=1.6.3=py_0
  - async-timeout=4.0.3=py39haa95532_0
  - attrs=23.1.0=py39haa95532_0
  - blas=1.0=mkl
  - blinker=1.6.2=py39haa95532_0
  - bottleneck=1.3.7=py39h9128911_0
  - ca-certificates=2023.12.12=haa95532_0
  - cachetools=4.2.2=pyhd3eb1b0_0
  - certifi=2024.2.2=py39haa95532_0
  - cffi=1.16.0=py39h2bbff1b_0
  - charset-normalizer=2.0.4=pyhd3eb1b0_0
  - click=8.1.7=py39haa95532_0
  - colorama=0.4.6=py39haa95532_0
  - cryptography=41.0.3=py39h3438e0d_0
  - flatbuffers=2.0.0=h6c2663c_0
  - frozenlist=1.4.0=py39h2bbff1b_0
  - gast=0.4.0=pyhd3eb1b0_0
  - giflib=5.2.1=h8cc25b3_3
  - google-auth=2.6.0=pyhd3eb1b0_0
  - google-auth-oauthlib=0.4.4=pyhd3eb1b0_0
  - google-pasta=0.2.0=pyhd3eb1b0_0
-- More --

```

Рисунок 17 – Часть файла environments.yml

Содержимое файлов requirements.txt и environment.yml представляют собой список зависимостей для проекта, но в различных форматах, соответствующих разным менеджерам пакетов.

requirements.txt содержит список Python-пакетов с указанием их версий, кроме того некоторые пакеты указаны с URL-адресами, указывающими на локальные файлы, что означает установку из локальных источников.

environment.yml представляет собой файл YAML, определяющий среду conda. Он включает зависимости с конкретными версиями для пакетов, таких как TensorFlow, PyJWT и других.

Оба файла определяют необходимые зависимости для проекта, но в разных форматах, подходящих для различных менеджеров пакетов.

6. Сольем ветки develop и main/master и отправим изменения на удаленный репозиторий:



```
PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> git log --oneline
6a35251 (HEAD -> develop) end working with conda
ec77dad (origin/main, origin/HEAD, main) Create README.md
63b036c Initial commit
PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14>
```

Рисунок 18 – История коммитов

```
PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> git merge develop
Updating ec77dad..6a35251
Fast-forward
 .gitignore      | 2 +-
 environment.yml | Bin 0 -> 7024 bytes
 requirements.txt | Bin 0 -> 10162 bytes
 3 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 environment.yml
 create mode 100644 requirements.txt
PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14>
```

Рисунок 19 – Слияние веток

```
PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14> git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 4.50 KiB | 4.50 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/PoItaTo/fundamentals_of_software_engineering_lab2_14
   ec77dad..6a35251  main -> main
PS C:\Study\Основы программной инженерии\2.14\fundamentals_of_software_engineering_lab2_14>
```

Рисунок 20 – Отправка изменений на удаленный репозиторий

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 использовать менеджер пакетов `pip`, указав имя пакета после команды установки, например: `pip install package_name`.

2. Как осуществить установку менеджера пакетов `pip`?

Установку менеджера пакетов `pip` можно осуществить вместе с установкой Python. В более поздних версиях Python `pip` устанавливается автоматически. В старых версиях можно установить `pip`, загрузив и запустив скрипт `get-pip.py`: `python get-pip.py`.

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

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

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

Для установки последней версии пакета с помощью `pip` следует использовать флаг `--upgrade` вместе с именем пакета: `pip install --upgrade package_name`.

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

Для установки заданной версии пакета с помощью `pip` следует указать конкретную версию пакета после его имени, например: `pip install package_name==1.2.3`.

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

Для установки пакета из `git` репозитория с помощью `pip` можно использовать команду `pip install git+URL`, где URL - это адрес репозитория на GitHub или другом `git`-хостинге.

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

Для установки пакета из локальной директории с помощью `pip` следует указать путь к директории после команды установки, например: `pip install /path/to/package`.

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

Для удаления установленного пакета с помощью `pip` можно использовать команду `pip uninstall package_name`.

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

Для обновления установленного пакета с помощью `pip` можно использовать команду `pip install --upgrade package_name`.

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

Для отображения списка установленных пакетов с помощью `pip` следует использовать команду `pip list`.

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

Виртуальные окружения в языке Python используются для изоляции проекта от других проектов и системных установок, что позволяет иметь разные версии пакетов для разных проектов и избежать конфликтов зависимостей.

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

Основные этапы работы с виртуальными окружениями включают создание, активацию, установку и использование пакетов в виртуальном окружении, а также деактивацию и удаление виртуального окружения.

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

Работа с виртуальными окружениями с помощью `venv` осуществляется с помощью стандартного модуля Python `venv`, который позволяет создавать и управлять виртуальными окружениями.

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

Работа с виртуальными окружениями с помощью `virtualenv` осуществляется с помощью утилиты `virtualenv`, которая создает изолированные окружения и позволяет управлять ими.

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

`pipenv` – это инструмент для управления зависимостями и виртуальными окружениями в Python. Работа с виртуальными окружениями `pipenv` включает создание, активацию, установку и использование пакетов в виртуальном окружении, а также деактивацию и удаление виртуального окружения.

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

Файл `requirements.txt` используется для указания списка зависимостей проекта. Создать этот файл можно вручную, просто перечислив имена пакетов с версиями или используя команду `pip freeze > requirements.txt`. Формат файла - каждый пакет с указанием имени и версии на отдельной строке.

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

Основные преимущества пакетного менеджера `conda` по сравнению с `pip` включают управление зависимостями более сложных пакетов, включая не только Python, но и бинарные зависимости, а также возможность создания изолированных сред с помощью виртуальных окружений.

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

Пакетный менеджер `conda` входит в дистрибутив `Anaconda` и `Miniconda`.

19. Как создать виртуальное окружение `conda`?

Для создания виртуального окружения `conda` следует использовать команду `conda create --n myenv`.

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

Для активации и установки пакетов в виртуальное окружение conda следует использовать команду `conda activate myenv` для активации окружения, а затем устанавливать пакеты с помощью `conda install package_name`.

#### 21. Как деактивировать и удалить виртуальное окружение conda?

Для деактивации и удаления виртуального окружения conda следует использовать команды `conda deactivate` для деактивации окружения и `conda env remove --name myenv` для удаления окружения.

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

Файл `environment.yml` используется для определения среды conda. Создать этот файл можно вручную, перечислив зависимости, или сгенерировать автоматически с помощью команды `conda env export > environment.yml`. Формат файла - YAML секция `dependencies` с указанием пакетов и их версий.

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

Для создания виртуального окружения conda с помощью файла `environment.yml` следует 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 IDE PyCharm работа с виртуальными окружениями conda осуществляется с помощью встроенных инструментов. Для этого следует перейти в настройки проекта, выбрать интерпретатор Python и указать путь к интерпретатору из виртуального окружения conda.

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

Файлы `requirements.txt` и `environment.yml` должны храниться в репозитории git, чтобы другие разработчики могли легко создать ту же среду и установить необходимые зависимости для работы с проектом. Это также обеспечивает воспроизводимость среды и управление зависимостями.