

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

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

Выполнил студент группы ПИЖ-б-о-22-1
Душин Александр Владимирович.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2024

Тема: Установка пакетов в Python. Виртуальное окружение.

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

Ход выполнения работы:


1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT 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.](#)

Required fields are marked with an asterisk (*).

Owner *

 MrPlatinum ▾

Repository name *

/ ProgrammEngineering17

✓ ProgrammEngineering17 is available.

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

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 общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering17.git
Cloning into 'ProgrammEngineering17'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование созданного репозитория на локальный компьютер

```
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
.idea/
```

Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering17 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering17 (develop)
$
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

2. Выполнить индивидуальное задание лабораторной работы 2.11, оформив все функции программы в виде отдельного модуля. Разработанный модуль должен быть подключен в основную программу с помощью одного из вариантов команды `import`. Номер варианта уточнить у преподавателя:

```
(base) PS C:\Users\Alexander> cd C:\Users\Alexander\Documents\ProgrammEngineering17
(base) PS C:\Users\Alexander\Documents\ProgrammEngineering17> conda create --n ProgrammEngineering17 python=3.12.0
Retrieving notices: ...working... done
WARNING: A conda environment already exists at 'C:\Users\Alexander\anaconda3\envs\ProgrammEngineering17'
Remove existing environment (y/[n])? y

Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\Users\Alexander\anaconda3\envs\ProgrammEngineering17

added / updated specs:
- python=3.12.0

The following packages will be downloaded:



| package           | build           | size    |
|-------------------|-----------------|---------|
| expat-2.5.0       | hd77b12b_0      | 225 KB  |
| pip-23.3.1        | py312haa95532_0 | 2.9 MB  |
| python-3.12.0     | h1d929f7_0      | 16.2 MB |
| setuptools-68.2.2 | py312haa95532_0 | 1.2 MB  |
| wheel-0.41.2      | py312haa95532_0 | 150 KB  |
| Total:            |                 | 20.7 MB |



The following NEW packages will be INSTALLED:



| package         | url                                                     |
|-----------------|---------------------------------------------------------|
| bzip2           | pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_5                |
| ca-certificates | pkgs/main/win-64::ca-certificates-2023.12.12-haa95532_0 |
| expat           | pkgs/main/win-64::expat-2.5.0-hd77b12b_0                |
| libffi          | pkgs/main/win-64::libffi-3.4.4-hd77b12b_0               |
| openssl         | pkgs/main/win-64::openssl-3.0.13-h2bbff1b_0             |
| pip             | pkgs/main/win-64::pip-23.3.1-py312haa95532_0            |
| python          | pkgs/main/win-64::python-3.12.0-h1d929f7_0              |
| setuptools      | pkgs/main/win-64::setuptools-68.2.2-py312haa95532_0     |
| sqlite          | pkgs/main/win-64::sqlite-3.41.2-h2bbff1b_0              |
| tk              | pkgs/main/win-64::tk-8.6.12-h2bbff1b_0                  |
| tzdata          | pkgs/main/noarch::tzdata-2024a-h04d1e81_0               |
| vc              | pkgs/main/win-64::vc-14.2-h21f451_1                     |
| vs2015_runtime  | pkgs/main/win-64::vs2015_runtime-14.27.29016-h3e58377_2 |
| wheel           | pkgs/main/win-64::wheel-0.41.2-py312haa95532_0          |
| xz              | pkgs/main/win-64::xz-5.4.6-h8cc25b3_0                   |
| zlib            | pkgs/main/win-64::zlib-1.2.13-h8cc25b3_0                |



Proceed (y/n)? y

Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
```

Рисунок 5 – Создание виртуального окружения Anaconda с именем репозитория (1)

```
#
# To activate this environment, use
#
# $ conda activate ProgrammEngineering17
#
# To deactivate an active environment, use
#
# $ conda deactivate

(base) PS C:\Users\Alexander\Documents\ProgrammEngineering17>
```

Рисунок 6 – Создание виртуального окружения Anaconda с именем репозитория (2)

3. Установить в виртуальное окружение следующие пакеты: `pip`, `NumPy`, `Pandas`, `SciPy`:

```
(base) PS C:\Users\Alexander\Documents\ProgrammEngineering17> conda activate ProgrammEngineering17
(ProgrammEngineering17) PS C:\Users\Alexander\Documents\ProgrammEngineering17> conda install pip numpy pandas scipy
Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: c:\Users\Alexander\anaconda3\envs\ProgrammEngineering17
added / updated specs:
- numpy
- pandas
- pip
- scipy

The following packages will be downloaded:
```

package	build	size
bottleneck-1.3.7	py312he58020_0	131 KB
mk1-service-2.4.0	py312h2b6ff1b_1	55 KB
mk1-fft-1.3.8	py312h2b6ff1b_0	160 KB
mk1-random-1.2.4	py312h596b97_0	196 KB
numexpr-2.8.7	py312h6b6d27_0	144 KB
numpy-1.26.4	py312hfd52020_0	11 KB
numpy-base-1.26.4	py312hdde369_0	6.6 MB
pandas-2.0.1	py312h0159946_0	14.2 MB
pytz-2023.3.post1	py312haa95532_0	199 KB
scipy-1.11.3	py312h3d2928d_0	19.3 MB
Total:		41.0 MB

```
The following NEW packages will be INSTALLED:
blas pkgs/main/win-64::blas-1.0-mkl
bottleneck pkgs/main/win-64::bottleneck-1.3.7-py312he58020_0
icc-rt pkgs/main/win-64::icc-rt-2022.1.0-h6049295_2
intel-openmp pkgs/main/win-64::intel-openmp-2023.1.0-h596b97_46320
mk1 pkgs/main/win-64::mk1-2023.1.0-h6b88ed4_46338
mk1-service pkgs/main/win-64::mk1-service-2.4.0-py312h2b6ff1b_1
mk1-fft pkgs/main/win-64::mk1-fft-1.3.8-py312h2b6ff1b_0
mk1-random pkgs/main/win-64::mk1-random-1.2.4-py312h596b97_0
numexpr pkgs/main/win-64::numexpr-2.8.7-py312h6b6d27_0
numpy pkgs/main/win-64::numpy-1.26.4-py312hfd52020_0
numpy-base pkgs/main/win-64::numpy-base-1.26.4-py312hdde369_0
pandas pkgs/main/win-64::pandas-2.0.1-py312h0159946_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_0
tbb pkgs/main/win-64::tbb-2021.8.0-h596b97_0

Proceed ([y]/n)? y
Downloading and Extracting Packages:
```

Рисунок 7 – Установка пакетов в виртуальное окружение (1)

4. Установить TensorFlow через conda:

```
(ProgrammEngineering17) PS C:\Users\Alexander\Documents\ProgrammEngineering17> conda install tensorflow
Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: \ warning libmamba Added empty dependency for problem type SOLVER_RULE_UPDATE
failed

LibMambaUnsatisfiableError: Encountered problems while solving:
- nothing provides bleach 1.5.0 needed by tensorboard-1.7.0-py35he025d50_1

Could not solve for environment specs
The following packages are incompatible
└─ pin-1 is installable and it requires
└─ python 3.12.* , which can be installed;
└─ tensorflow is not installable because there are no viable options
└─ tensorflow [1.10.0|1.9.0] would require
└─ python 3.5.* , which conflicts with any installable versions previously reported;
└─ tensorflow [1.10.0|1.11.0]...[2.1.0] would require
└─ python 3.6.* , which conflicts with any installable versions previously reported;
└─ tensorflow [1.13.1|1.14.0]...[2.9.1] would require
└─ python 3.7.* , which conflicts with any installable versions previously reported;
└─ tensorflow [1.7.0|1.7.1|1.8.0] would require
└─ tensorboard [>=1.7.0,<1.8.0 ]>=1.8.0,<1.9.0 ] , which requires
└─ bleach 1.5.0 , which does not exist (perhaps a missing channel);
└─ tensorflow [2.10.0|2.8.2|2.9.1] would require
└─ python 3.10.* , which conflicts with any installable versions previously reported;
└─ tensorflow [2.10.0|2.3.0]...[2.9.1] would require
└─ python 3.8.* , which conflicts with any installable versions previously reported;
└─ tensorflow [2.10.0|2.5.0|2.6.0|2.8.2|2.9.1] would require
└─ python 3.9.* , which conflicts with any installable versions previously reported.

(ProgrammEngineering17) PS C:\Users\Alexander\Documents\ProgrammEngineering17>
```

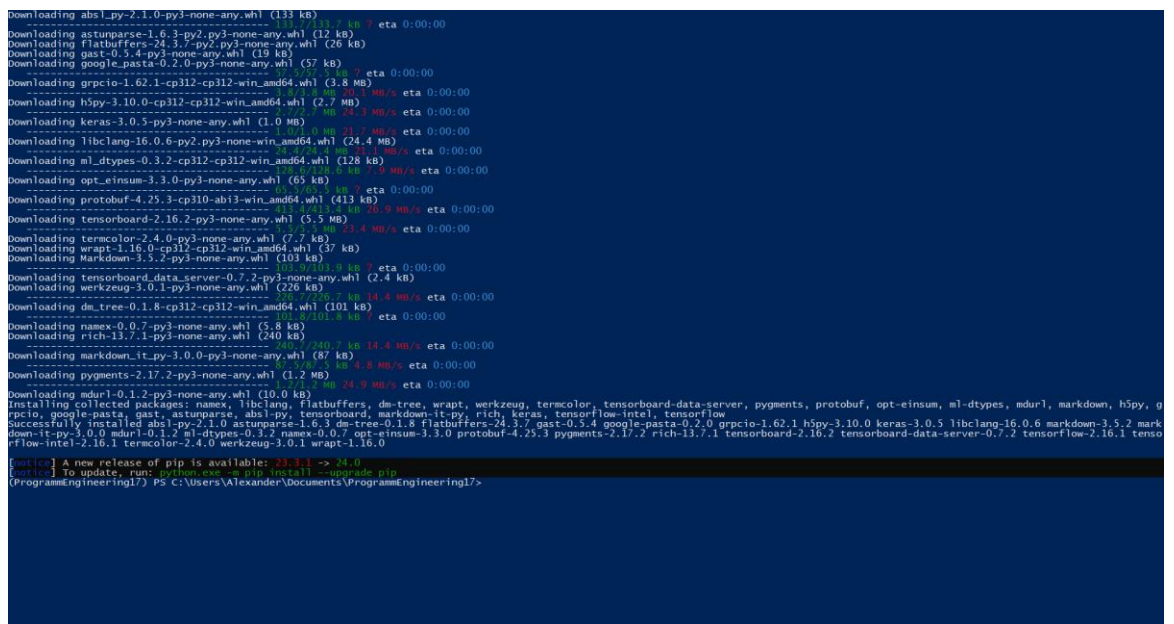
Рисунок 8 – Попытка установки пакета TensorFlow менеджером пакетов conda

Причина ошибки заключается в том, что версия Python, установленная в вашем виртуальном окружении, не совместима с требуемой версией TensorFlow.

Для решения этой проблемы есть несколько вариантов:

- 1) Обновить версию Python в вашем виртуальном окружении до одной из совместимых версий, перечисленных в сообщении об ошибке.
- 2) Создать новое виртуальное окружение с совместимой версией Python и установить TensorFlow в нем.
- 3) Установить TensorFlow с использованием инструмента управления пакетами Python `pip` вместо `conda`.

5. Установить TensorFlow через `pip`:

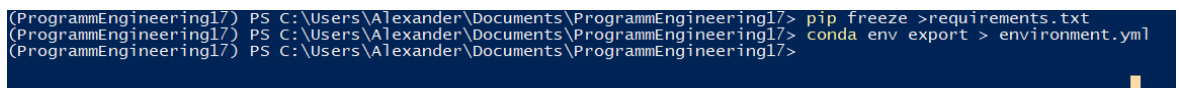


```
Downloading absl-py-2.1.0-py3-none-any.whl (133 kB)
Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Downloading flatbuffers-24.3.7-py2.py3-none-any.whl (20 kB)
Downloading gast-0.5.4-py3-none-any.whl (19 kB)
Downloading google-pasta-0.2.0-py3-none-any.whl (57 kB)
Downloading grpcio-1.62.1-cp312-cp312-win_amd64.whl (3.8 MB)
Downloading h5py-3.10.0-cp312-cp312-win_amd64.whl (2.7 MB)
Downloading keras-3.0.5-py3-none-any.whl (1.0 MB)
Downloading libclang-16.0.6-py2.py3-none-win_amd64.whl (24.4 MB)
Downloading ml-dtypes-0.3.2-cp312-cp312-win_amd64.whl (128 kB)
Downloading opt-einsum-3.3.0-py3-none-any.whl (65 kB)
Downloading protobuf-4.25.3-cp310-abi3-win_amd64.whl (413 kB)
Downloading tensorboard-2.16.2-py3-none-any.whl (5.5 MB)
Downloading termcolor-2.4.0-py3-none-any.whl (7.2 kB)
Downloading wrapt-1.16.0-cp312-cp312-win_amd64.whl (37 kB)
Downloading Werkzeug-3.0.1-py3-none-any.whl (226 kB)
Downloading dm-tree-0.1.8-cp312-cp312-win_amd64.whl (101 kB)
Downloading namex-0.0.7-py3-none-any.whl (5.8 kB)
Downloading rich-13.7.1-py3-none-any.whl (240 kB)
Downloading markdown-it-py-3.0.0-py3-none-any.whl (87 kB)
Downloading pygments-2.17.2-py3-none-any.whl (1.2 MB)
Downloading mdurl-0.1.2-py3-none-any.whl (10.0 kB)
Installing collected packages: namex, libclang, flatbuffers, dm-tree, wrapt, Werkzeug, termcolor, tensorboard-data-server, pygments, protobuf, opt-einsum, ml-dtypes, mdurl, markdown, h5py, g
Successfully installed absl-py-2.1.0 astunparse-1.6.3 dm-tree-0.1.8 flatbuffers-24.3.7 gast-0.5.4 google-pasta-0.2.0 grpcio-1.62.1 h5py-3.10.0 keras-3.0.5 libclang-16.0.6 markdown-3.5.2 mark
Flow-intel-2.16.1 termcolor-2.4.0 Werkzeug-3.0.1 wrapt-1.16.0
[notice] A new release of pip is available: 23.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
(ProgrammEngineering17) PS C:\Users\Alexander\Documents\ProgrammEngineering17>
```

Рисунок 9 – Установка пакета TensorFlow менеджером пакетов `pip` (1)

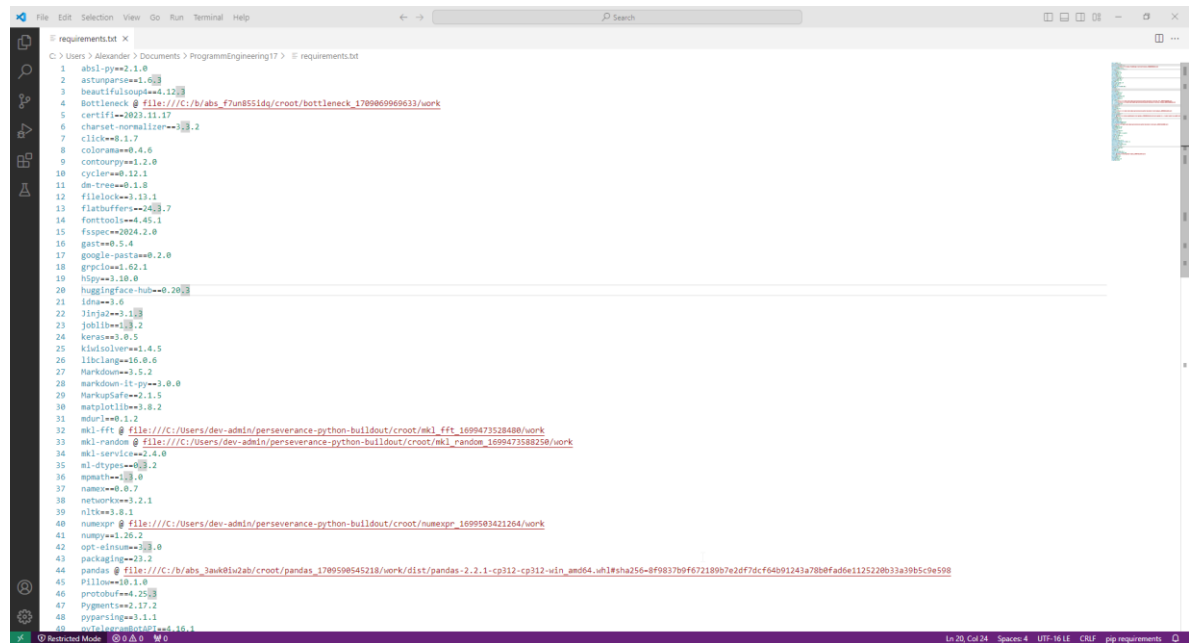
6. Сформировать файлы `requirements.txt` и `environment.yml`.

Проанализировать содержимое этих файлов.



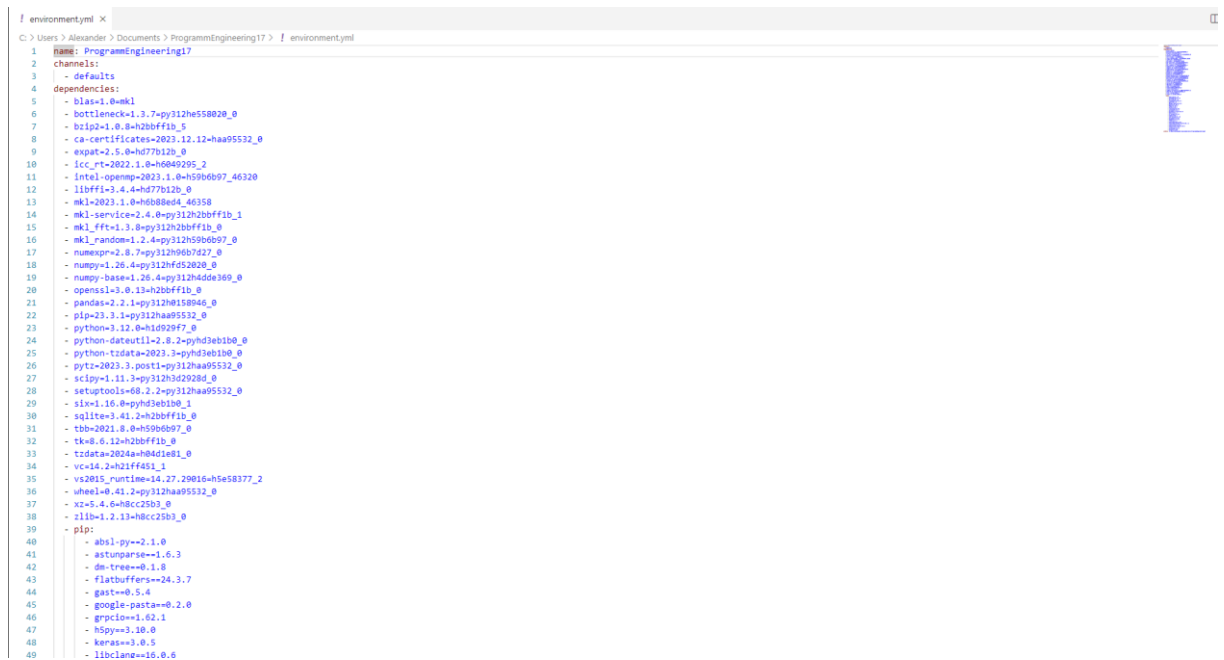
```
(ProgrammEngineering17) PS C:\Users\Alexander\Documents\ProgrammEngineering17> pip freeze > requirements.txt
(ProgrammEngineering17) PS C:\Users\Alexander\Documents\ProgrammEngineering17> conda env export > environment.yml
(ProgrammEngineering17) PS C:\Users\Alexander\Documents\ProgrammEngineering17>
```

Рисунок 10 – Создание файлов `requirements.txt` и `environment.yml`



```
1 absl-py==2.1.0
2 astunparse==1.6.3
3 beautifulsoup4==4.12.3
4 Bottleneck @ file:///C:/b/abs_47un855ld/croot/bottleneck_1789869969631/work
5 certifi==2023.11.17
6 charset-normalizer==3.2
7 click==8.1.7
8 colorama==0.4.6
9 contourpy==1.2.0
10 cycler==0.12.1
11 dm-tree==0.1.8
12 filelock==3.13.1
13 flatbuffers==24.3.7
14 fonttools==4.45.1
15 fspec==2024.2.0
16 gast==0.5.4
17 google-pasta==0.2.0
18 grpcio==1.62.1
19 h5py==3.10.0
20 huggingface-hub==0.20.3
21 idna==3.6
22 Jinja2==3.1.3
23 joblib==1.3.2
24 keras==3.0.5
25 Keras-Preprocessing==1.1.2
26 libclang==16.0.6
27 Markdown==3.5.2
28 markdown-it-py==3.0.0
29 MarkupSafe==2.1.5
30 matplotlib==3.8.2
31 ndbc==0.1.2
32 nkl-fft @ file:///C:/Users/dev-admin/perseverance-python-buildout/croot/nkl-fft_1699473528488/work
33 nkl-random @ file:///C:/Users/dev-admin/perseverance-python-buildout/croot/nkl-random_1699473528488/work
34 nkl-service==2.4.0
35 nl-cvtypes==0.3.2
36 numpy==1.26.2
37 names==0.0.7
38 networkx==3.2.1
39 nltk==3.8.1
40 numexpr @ file:///C:/Users/dev-admin/perseverance-python-buildout/croot/numexpr_1699503421264/work
41 numpy==1.26.2
42 opt-einsum==3.3.0
43 packaging==23.2
44 pandas @ file:///C:/b/abs_3auk8iuv2ab/croot/pandas_1789508545218/work/dist/pandas-2.2.1-cp312-cp312-win_amd64.whl#sha256=8f983769f672189b7e2df7dc64b01243a78b8fa06e1125228033a39b5c9e598
45 pillow==10.1.0
46 protobuf==4.25.3
47 Pygments==2.17.2
48 pyparsing==3.1.1
49 PyTorch==2.1.0
```

Рисунок 11 – Содержимое файла requirements.txt



```
1 name: ProgramEngineering17
2 channels:
3   - defaults
4 dependencies:
5   - blas=1.0-mkl
6   - bottleneck=1.2.7-py312h558020_0
7   - bzip2=1.0.8-h2b0ff1b_5
8   - ca-certificates=2023.12.12-ha95532_0
9   - expat=2.5.0-hd77b12b_0
10  - icc_rt=2022.1.0-h0649295_2
11  - intel-openmp=2023.1.0-h9b6097_46320
12  - libbfg=3.4.4-hd77b12b_0
13  - mkl=2023.1.0-h0b80e04_46350
14  - mkl-service=2.4.0-py312h2b0ff1b_1
15  - mkl_fft=1.3.0-py312h2b0ff1b_0
16  - mkl_random=1.2.4-py312h558020_0
17  - numexpr=2.8.7-py312h0607027_0
18  - numpy=1.26.4-py312h558020_0
19  - numpy-base=1.26.4-py312h558020_0
20  - openssl=3.0.13-h2b0ff1b_0
21  - pandas=2.2.1-py312h100046_0
22  - pip=23.3.1-py312ha95532_0
23  - python=3.12.0-h10929f7_0
24  - python-dateutil=2.8.2-pyhd3eb1b0_0
25  - python-tzdata=2023.3-pyhd3eb1b0_0
26  - pytz=2023.3-post1-py312ha95532_0
27  - scipy=1.11.3-py312h2d2028d_0
28  - setuptools=68.2.2-py312ha95532_0
29  - six=1.16.0-pyhd3eb1b0_1
30  - sqlalchemy=1.4.42-h2b0ff1b_0
31  - tbb=2021.8.0-h59b6097_0
32  - tk=8.6.12-h2b0ff1b_0
33  - tzdata=2024a-h04d1e81_0
34  - vc=14.2-h21f4d51_1
35  - vs2015_runtime=14.22.29016-h5e58377_2
36  - wheel=0.41.2-py312ha95532_0
37  - xz=5.4.6-h8cc25b3_0
38  - zlib=1.2.13-h8cc25b3_0
39
40  - pip:
41    - absl-py==2.1.0
42    - astunparse==1.6.3
43    - dm-tree==0.1.8
44    - flatbuffers==24.3.7
45    - gast==0.5.4
46    - google-pasta==0.2.0
47    - grpcio==1.62.1
48    - h5py==3.10.0
49    - keras==3.0.5
50    - libclang==16.0.6
```

Рисунок 12 – Содержимое файла environment.yml

7. Зафиксируем проделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering17 (develop)
$ git log
commit 6d109386b2f727516aa583b2704b3cb5b2461257 (HEAD -> develop)
Author: MrPlatynum <71084177+MrPlatynum@users.noreply.github.com>
Date: Mon Mar 11 23:24:48 2024 +0300

    add requirements.txt, environment.yml

commit 30051c0dd6db86263a494c2edc09f57bf9ad28a0 (origin/main, origin/HEAD, main)
Author: MrPlatynum <71084177+MrPlatynum@users.noreply.github.com>
Date: Mon Feb 19 22:44:03 2024 +0300

    Initial commit

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering17 (develop)
$
```

Рисунок 13 – Коммиты ветки develop во время выполнения лабораторной Работы

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering17 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering17 (main)
$ git merge develop
Updating 30051c0..6d10938
Fast-forward
 .gitignore | 2 +-
 ...1\203\321\210\320\270\320\275\320\220\320\222.docx" | Bin 0 -> 239753 bytes
 environment.yml | Bin 0 -> 3902 bytes
 requirements.txt | Bin 0 -> 4100 bytes
 4 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 "doc/\320\233\320\24017_\320\224\321\203\321\210\320\270\320\275\320\220\320\222.docx"
 create mode 100644 environment.yml
 create mode 100644 requirements.txt

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering17 (main)
$
```

Рисунок 14 – Слияние веток main и develop

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering17 (main)
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 214.81 KiB | 23.87 MiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/MrPlatynum/ProgrammEngineering17.git
 30051c0..6d10938 main -> main
```

Рисунок 15 – Отправка изменений на удаленный репозиторий

Ответы на контрольные вопросы:

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

Пакет Python, не входящий в стандартную библиотеку, можно установить с помощью менеджера пакетов Python, такого как `pip` или `conda`

2. Как осуществить установку менеджера пакетов `pip`?

При развертывании современной версии Python (начиная с Python 2.7.9 и Python 3.4), `pip` устанавливается автоматически.

Если же необходимо установить `pip` - скачайте скрипт `get-pip.py`:

```
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

и выполните его.

```
$ python get-pip.py
```

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

Существует так называемый Python Package Index (PyPI) – это репозиторий, открытый для всех Python разработчиков, в нем вы можете найти пакеты для решения практически любых задач. Там также есть возможность выкладывать свои пакеты. Для скачивания и установки используется специальная утилита, которая называется `pip`.

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

Установка последней версии пакета:

```
$ pip install ProjectName
```

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

Установка определенной версии:

```
$ pip install ProjectName==3.2
```

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

Установка Python пакета из `git` репозитория:

```
$ pip install -e git+https://gitrepo.com/ProjectName.git
```

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

Установка пакета из локальной директории:

```
$ pip install ./dist/ProjectName.tar.gz
```

8. Как удалить установленный пакет с помощью pip?

Для того, чтобы удалить пакет воспользуйтесь командой:

```
$ pip uninstall ProjectName
```

9. Как обновить установленный пакет с помощью pip?

Для обновления пакета используйте ключ `--upgrade`.

```
$ pip install --upgrade ProjectName
```

10. Как отобразить список установленных пакетов с помощью pip?

Для вывода списка всех установленных пакетов применяется команда `pip list`:

```
$ pip list
```

Если вы хотите получить более подробную информацию о конкретном пакете, то используйте аргумент `show`:

```
$ pip show ProjectName
```

11. Каковы причины появления виртуальных окружений в языке Python?

Виртуальные окружения в языке Python созданы для обеспечения изоляции проектов, управления зависимостями, упрощения тестирования и разработки, а также для изоляции системных установок, предотвращая конфликты между версиями пакетов и Python.

12. Каковы основные этапы работы с виртуальными окружениями?

основные этапы работы с виртуальным окружением:

1) Создаём через утилиту новое виртуальное окружение в отдельной папке для выбранной версии интерпретатора Python.

2) Активируем ранее созданное виртуального окружения для работы.

3) Работаем в виртуальном окружении, а именно управляем пакетами используя `pip` и запускаем выполнение кода.

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?

С помощью модуля venv в Python можно создавать виртуальные окружения и активировать их с помощью команды `python -m venv <имя_окружения>`. После создания окружения его можно активировать с помощью команды `source <имя_окружения>/bin/activate` на Unix-подобных системах или `.\<имя_окружения>\Scripts\activate` на Windows. Активировав окружение, можно устанавливать и использовать пакеты, не влияя на глобальные установки Python.

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

С помощью virtualenv в Python можно создавать виртуальные окружения и активировать их с помощью команды `virtualenv <имя_окружения>`. После создания окружения его можно активировать с помощью команды `source <имя_окружения>/bin/activate` на Unix-подобных системах или `.\<имя_окружения>\Scripts\activate` на Windows. Активировав окружение, можно устанавливать и использовать пакеты, не влияя на глобальные установки Python.

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

С помощью pipenv в Python можно создавать и управлять виртуальными окружениями и их зависимостями. Для создания виртуального окружения с pipenv используется команда `pipenv install`, которая создает виртуальное окружение и устанавливает зависимости из файла Pipfile. После этого можно активировать окружение с помощью команды `pipenv shell`. Для установки пакетов в виртуальное окружение используется команда `pipenv install <имя_пакета>`.

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

Файл requirements.txt используется для хранения списка всех

зависимостей проекта Python. Он содержит названия пакетов и их версии, необходимые для правильной работы проекта.

Чтобы создать файл requirements.txt, можно использовать команду
`pip freeze > requirements.txt`.

Эта команда сохранит текущий список установленных пакетов и их версий в файле requirements.txt.

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

Основная проблема заключается в том, что pip, easy_install и virtualenv ориентированы на Python. Эти инструменты игнорируют библиотеки зависимостей, реализованные с использованием других языков. Например, XSLT, HDF5, MKL и другие, которые не имеют setup.py в исходном коде и не устанавливают файлы в директорию site-packages.

Conda же 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.

18. В какие дистрибутивы Python входит пакетный менеджер conda?

Пакетный менеджер conda входит в дистрибутив Anaconda и Miniconda. Anaconda представляет собой полный пакет, включающий в себя множество научных пакетов и библиотек для анализа данных, машинного обучения и научных вычислений, включая conda. Miniconda, с другой стороны, представляет собой минимальную установку Python и conda, без предустановленных пакетов.

19. Как создать виртуальное окружение conda?

Выполнением следующих команд:

```
conda create -n %PROJ_NAME% python=3.7
```

```
conda activate %PROJ_NAME%
```

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

Чтобы активировать виртуальное окружение conda, необходимо

выполнить команду:

```
conda activate <имя_окружения>
```

Для установки пакетов в активированное виртуальное окружение conda используйте команду:

```
conda install <имя_пакета>
```

21. Как деактивировать и удалить виртуальное окружение conda?

Чтобы деактивировать виртуальное окружение conda, выполните следующую команду:

```
conda deactivate
```

Чтобы удалить виртуальное окружение conda, выполните команду:

```
conda env remove -n <имя_окружения>
```

Где <имя_окружения> - это имя виртуального окружения, которое вы хотите удалить.

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

Файл environment.yml используется для определения окружения с помощью conda, включая список всех пакетов и их версии, необходимых для воспроизведения окружения на другой системе. Чтобы создать файл environment.yml, выполните команду

```
conda env export > environment.yml.
```

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

Чтобы создать виртуальное окружение conda с помощью файла environment.yml, выполните следующую команду:

```
conda env create -f environment.yml
```

Эта команда создаст новое виртуальное окружение, используя информацию из файла environment.yml, и установит все необходимые пакеты с их версиями.

24. Самостоятельно изучите средства IDE PyCharm для работы с виртуальными окружениями conda. Опишите порядок работы с виртуальными окружениями conda в IDE PyCharm.

В PyCharm вы можете создавать и управлять виртуальными окружениями conda через интерфейс пользователя. Для этого перейдите в раздел "Settings" (Настройки) -> "Project: [ваш проект]" -> "Python Interpreter" (Интерпретатор Python). Здесь вы можете добавить новое виртуальное окружение, выбрав опцию "Add" (Добавить) и указав путь к исполняемому файлу python в вашем виртуальном окружении conda. После этого PyCharm будет использовать выбранное виртуальное окружение для работы с вашим проектом.

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

Файлы requirements.txt и environment.yml хранят информацию о зависимостях вашего проекта, что делает воспроизведение окружения и установку пакетов удобными для других разработчиков. Помещение этих файлов в репозиторий git позволяет легко синхронизировать окружение между разными компьютерами и убеждаться, что все члены команды используют одинаковое окружение.