

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Установка пакетов в Python. Виртуальные окружения»

ОТЧЕТ
по лабораторной работе №17
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна

2 курс, группа ПИЖ-б-о-21-1,

011.03.04 «Программная инженерия»,

направленность (профиль) «Разработка

и сопровождение программного

обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

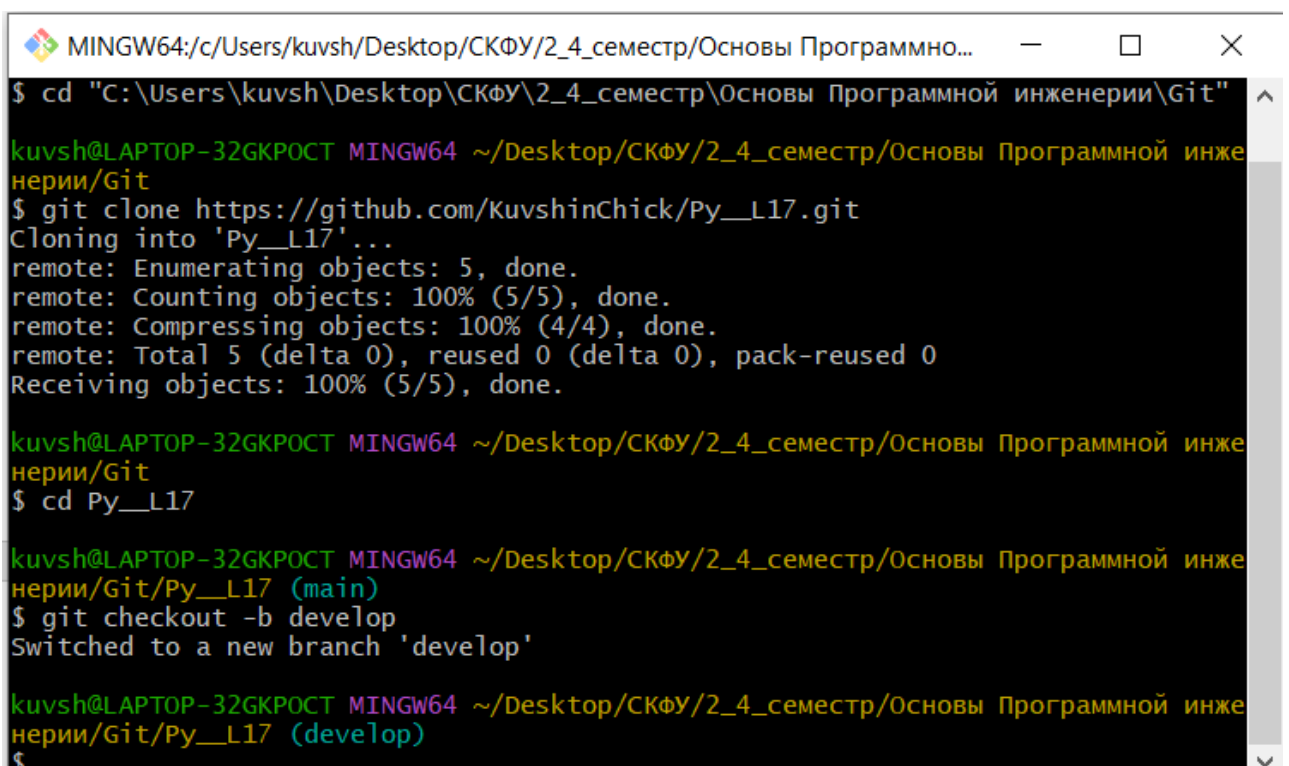
Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

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

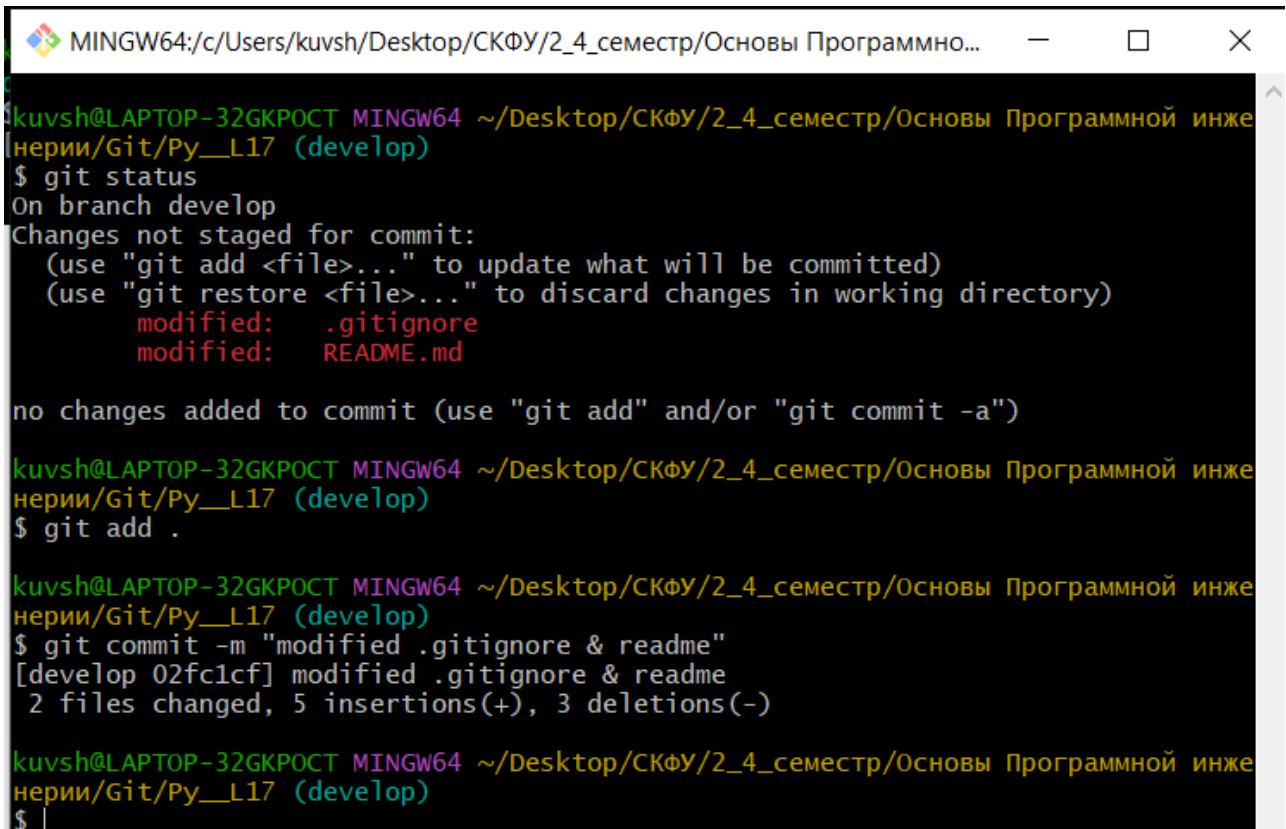
Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.



```
MINGW64/c:/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Основы Программно...
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_4_семестр\Основы Программной инженерии\Git"
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git
$ git clone https://github.com/KuvshinChick/Py__L17.git
Cloning into 'Py__L17'...
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.
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git
$ cd Py__L17
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py__L17 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py__L17 (develop)
$
```

Рисунок 17.1 – Клонирование репозитория и создание ветки develop



```
MINGW64:/c/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Основы Программно...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инже
нерии/Git/Py__L17 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инже
нерии/Git/Py__L17 (develop)
$ git add .

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инже
нерии/Git/Py__L17 (develop)
$ git commit -m "modified .gitignore & readme"
[develop 02fc1cf] modified .gitignore & readme
 2 files changed, 5 insertions(+), 3 deletions(-)

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инже
нерии/Git/Py__L17 (develop)
$
```

Рисунок 17.2 – Обновление .gitignore и readme

5. Создайте виртуальное окружение Anaconda с именем репозитория.

Для Windows, если используется дистрибутив Anaconda, то необходимо вначале запустить консоль Anaconda Powershell Prompt. Делается это из системного меню, посредством выбора следующих пунктов: Пуск Anaconda3 (64-bit) Anaconda Powershell Prompt (Anaconda3).

```
Anaconda Prompt (Anaconda3)

(base) C:\Users\kuvsh>conda create -n Py_L17
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 23.1.0

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\kuvsh\.conda\envs\Py_L17

Proceed ([y]/n)? y
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate Py_L17
#
# To deactivate an active environment, use
#
#     $ conda deactivate
#
Retrieving notices: ...working... done
(base) C:\Users\kuvsh>
```

Рисунок 17.3 – Результат создания conda-окружения

```
(base) C:\Users\kuvsh>conda activate Py_L17
(Py_L17) C:\Users\kuvsh>
```

Рисунок 17.4 – Результат активации conda-окружения

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

```
Anaconda Prompt (Anaconda3) - conda install pip

(base) C:\Users\kuvsh>activate Py__L17

(Py__L17) C:\Users\kuvsh>conda install pip
Collecting package metadata (current_repodata.json): failed

CondaSSLError: OpenSSL appears to be unavailable on this machine. OpenSSL is required to
download and install packages.

Exception: HTTPConnectionPool(host='repo.anaconda.com', port=443): Max retries exceeded with url: /pkgs/main/win-64/current_repodata.
json (Caused by SSLError("Can't connect to HTTPS URL because the SSL module is not available."))

(Py__L17) C:\Users\kuvsh>conda install pip
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
current version: 22.9.0
latest version: 23.1.0

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\Users\kuvsh\.conda\envs\Py__L17

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-1.1.1t | h2bfff1b_0 | 5.5 MB
```

Рисунок 17.5 – Процесс установки пакета pip

```
Anaconda Prompt (Anaconda3) - conda install pip - conda install NumPy

(Py__L17) C:\Users\kuvsh>conda install NumPy
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
current version: 22.9.0
latest version: 23.1.0

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\Users\kuvsh\.conda\envs\Py__L17

added / updated specs:
- numpy

The following packages will be downloaded:

package | build | size
-----|-----|-----
mkl-service-2.4.0 | py310h2bfff1b_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:
```

Рисунок 17.6 – Процесс установки пакета NumPy

```
Anaconda Prompt (Anaconda3) - conda install pip - conda install NumPy - conda install Pandas
Retrieving notices: ...working... done

(Py_L17) C:\Users\kuvsh>conda install Pandas
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 23.1.0

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\Users\kuvsh\.conda\envs\Py_L17

added / updated specs:
- pandas

The following packages will be downloaded:

package | build | size
-----|-----|-----
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:
```

Рисунок 17.7 – Процесс установки пакета Pandas

```
Anaconda Prompt (Anaconda3) - conda install pip - conda install NumPy - conda install Pandas - conda install SciPy
(Py_L17) C:\Users\kuvsh>conda install SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 23.1.0

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\Users\kuvsh\.conda\envs\Py_L17

added / updated specs:
- scipy

The following packages will be downloaded:

package | build | size
-----|-----|-----
brotlipy-0.7.0 | py310h2bbff1b_1002 | 335 KB
cffi-1.15.1 | py310h2bbff1b_3 | 239 KB
cryptography-38.0.4 | py310h21b164f_0 | 1.0 MB
idna-3.4 | py310haa95532_0 | 97 KB
pooch-1.4.0 | pyhd3eb1b0_0 | 41 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
-----|-----|-----
```

Рисунок 17.7 – Процесс установки пакета SciPy

```
(Py__L17) C:\Users\kuvsh>conda list
# packages in environment at C:\Users\kuvsh\.conda\envs\Py__L17:
#
# Name                                Version                                Build      Channel
appdirs                               1.4.4                                pyhd3eb1b0_0
blas                                  1.0                                  mkl
bottleneck                            1.3.5                                py310h9128911_0
brotlipy                              0.7.0                                py310h2bbff1b_1002
bzip2                                 1.0.8                                he774522_0
ca-certificates                       2023.01.10                           haa95532_0
certifi                               2022.12.7                            py310haa95532_0
cffi                                  1.15.1                              py310h2bbff1b_3
charset-normalizer                    2.0.4                                pyhd3eb1b0_0
cryptography                          38.0.4                              py310h21b164f_0
fftw                                  3.3.9                                h2bbff1b_1
icc_rt                                2022.1.0                             h6049295_2
idna                                  3.4                                  py310haa95532_0
intel-openmp                          2021.4.0                             haa95532_3556
libffi                                3.4.2                                hd77b12b_6
mkl                                   2021.4.0                             haa95532_640
mkl-service                           2.4.0                                py310h2bbff1b_0
mkl_fft                              1.3.1                                py310ha0764ea_0
mkl_random                           1.2.2                                py310h4ed8f06_0
numexpr                               2.8.4                                py310hd213c9f_0
numpy                                 1.23.5                              py310h60c9a35_0
numpy-base                           1.23.5                              py310h04254f7_0
openssl                               1.1.1t                               h2bbff1b_0
packaging                             22.0                                  py310haa95532_0
pandas                               1.5.2                                py310h4ed8f06_0
pip                                   22.3.1                              py310haa95532_0
pooch                                 1.4.0                                pyhd3eb1b0_0
pycparser                             2.21                                 pyhd3eb1b0_0
pyopenssl                             22.0.0                              pyhd3eb1b0_0
pysocks                               1.7.1                                py310haa95532_0
python                                3.10.9                              h966fe2a_0
python-dateutil                       2.8.2                                pyhd3eb1b0_0
pytz                                  2022.7                              py310haa95532_0
requests                              2.28.1                              py310haa95532_0
scipy                                 1.10.0                              py310hb9afe5d_0
setuptools                            65.6.3                              py310haa95532_0
six                                   1.16.0                              pyhd3eb1b0_1
sqlite                                 3.40.1                              h2bbff1b_0
tk                                    8.6.12                              h2bbff1b_0
tzdata                                2022g                               h04d1e81_0
urllib3                               1.26.14                             py310haa95532_0
vc                                     14.2                                h21ff451_1
vs2015_runtime                        14.27.29016                         h5e58377_2
wheel                                 0.37.1                              pyhd3eb1b0_0
win_inet_pton                         1.1.0                              py310haa95532_0

wincertstore                          0.2                                py310haa95532_2
xz                                     5.2.10                             h8cc25b3_1
zlib                                  1.2.13                             h8cc25b3_0

(Py__L17) C:\Users\kuvsh>_
```

Рисунок 17.8 – Результат установки пакетов

7. Попробуйте установить менеджером пакетов conda пакет TensorFlow.

Возникает ли при этом ошибка? Попробуйте выявить и укажите причину этой ошибки.

- Ошибка не возникает

```

Anaconda Prompt (Anaconda3) - conda install pip - conda install NumPy - conda install Pandas - conda install SciPy - conda install TensorFlow

(Py_L17) C:\Users\kuvsh>conda install TensorFlow
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 23.1.0

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\kuvsh\.conda\envs\Py_L17

  added / updated specs:
    - tensorflow

The following packages will be downloaded:

  package | build | size
  -----|-----|-----
  _tfflow_select-2.3.0 | mk1 | 3 KB
  absl-py-1.3.0 | py310haa95532_0 | 172 KB
  aiohttp-3.8.3 | py310h2bbff1b_0 | 418 KB
 aiosignal-1.2.0 | pyhd3eb1b0_0 | 12 KB
  astunparse-1.6.3 | py_0 | 17 KB
  async-timeout-4.0.2 | py310haa95532_0 | 12 KB
  attrs-22.1.0 | py310haa95532_0 | 85 KB
  blinker-1.4 | py310haa95532_0 | 22 KB
  cachetools-4.2.2 | pyhd3eb1b0_0 | 13 KB
  click-8.0.4 | py310haa95532_0 | 157 KB
  colorama-0.4.6 | py310haa95532_0 | 32 KB
  flatbuffers-2.0.0 | h6c2663c_0 | 1.4 MB
  flit-core-3.6.0 | pyhd3eb1b0_0 | 42 KB
  frozenlist-1.3.3 | py310h2bbff1b_0 | 40 KB
  gast-0.4.0 | pyhd3eb1b0_0 | 13 KB
```

Рисунок 17.8 – Процесс установки пакета TensorFlow

```

Anaconda Prompt (Anaconda3) - conda install pip - conda install NumPy - conda install Pandas - conda install SciPy - conda install TensorFlow

(Py_L17) C:\Users\kuvsh>conda list
# packages in environment at C:\Users\kuvsh\.conda\envs\Py_L17:
#
# Name Version Build Channel
_tflow_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 py310h9128011_0
brotli 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
google-auth-oauthlib 0.4.4 pyhd3eb1b0_0
google-pasta 0.2.0 pyhd3eb1b0_0
grpcio 1.42.0 py310hc60d5dd_0
h5py 3.7.0 py310hfc34f40_0
hdf5 1.10.6 h1756f20_1
icc_rt 2022.1.0 h6049295_2
icu 58.2 ha925a31_3
idna 3.4 py310haa95532_0
intel-openmp 2021.4.0 haa95532_3556
jpeg 9e h2bbff1b_0
keras 2.10.0 py310haa95532_0
keras-preprocessing 1.1.2 pyhd3eb1b0_0
libcurl 7.87.0 h86230a5_0
```



```
Anaconda Prompt (Anaconda3) - conda install pip - conda install NumPy - conda install Pandas - conda install SciPy - conda install TensorFlow
libprotobuf 3.20.3 h23ce68f_0
libssh2 1.10.0 hcd4344a_0
markdown 3.4.1 py310haa95532_0
markupsafe 2.1.1 py310h2bbff1b_0
mkl 2021.4.0 haa95532_640
mkl-service 2.4.0 py310h2bbff1b_0
mkl_fft 1.3.1 py310ha0764ea_0
mkl_random 1.2.2 py310h4ed8f06_0
multidict 6.0.2 py310h2bbff1b_0
numexpr 2.8.4 py310hd213c9f_0
numpy 1.23.5 py310h60c9a35_0
numpy-base 1.23.5 py310h04254f7_0
oauthlib 3.2.1 py310haa95532_0
openssl 1.1.1t h2bbff1b_0
opt_einsum 3.3.0 pyhd3eb1b0_1
packaging 22.0 py310haa95532_0
pandas 1.5.2 py310h4ed8f06_0
pip 22.3.1 py310haa95532_0
pooch 1.4.0 pyhd3eb1b0_0
protobuf 3.20.3 py310hd77b12b_0
pyasn1 0.4.8 pyhd3eb1b0_0
pyasn1-modules 0.2.8 py_0
pycparser 2.21 pyhd3eb1b0_0
pyjwt 2.4.0 py310haa95532_0
pyopenssl 22.0.0 pyhd3eb1b0_0
pysocks 1.7.1 py310haa95532_0
python 3.10.9 h966fe2a_0
python-dateutil 2.8.2 pyhd3eb1b0_0
python-flatbuffers 2.0 pyhd3eb1b0_0
pytz 2022.7 py310haa95532_0
requests 2.28.1 py310haa95532_0
requests-oauthlib 1.3.0 py_0
rsa 4.7.2 pyhd3eb1b0_1
scipy 1.10.0 py310hb9afe5d_0
setuptools 65.6.3 py310haa95532_0
six 1.16.0 pyhd3eb1b0_1
snappy 1.1.9 h6c2663c_0
sqlite 3.40.1 h2bbff1b_0
tensorboard 2.10.0 py310haa95532_0
tensorboard-data-server 0.6.1 py310haa95532_0
tensorboard-plugin-wit 1.8.1 py310haa95532_0
tensorflow 2.10.0 mkl_py310hd99672f_0
tensorflow-base 2.10.0 mkl_py310h6a7f48e_0
tensorflow-estimator 2.10.0 py310haa95532_0
termcolor 2.1.0 py310haa95532_0
tk 8.6.12 h2bbff1b_0
```

Рисунок 17.9 – Результат установки пакета TensorFlow

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

```
Anaconda Prompt (Anaconda3) - conda install pip - conda install NumPy - conda install Pandas - conda install SciPy - conda install TensorFlow

(Py_L17) C:\Users\kuvsh>pip install TensorFlow
Requirement already satisfied: TensorFlow in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (2.10.0)
Requirement already satisfied: numpy<=1.20 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (1.23.5)
Collecting protobuf<3.20,>=3.9.2
  Downloading protobuf-3.19.6-cp310-win_amd64.whl (895 kB)
----- 895.7/895.7 kB 1.4 MB/s eta 0:00:00
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (1.42.0)
Requirement already satisfied: absl-py<=1.0.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (1.3.0)
Requirement already satisfied: h5py<=2.9.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (3.7.0)
Requirement already satisfied: six<=1.12.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (1.16.0)
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 1.4 MB/s eta 0:00:00
Requirement already satisfied: opt-einsum<=2.3.2 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (3.3.0)
Requirement already satisfied: packaging in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (22.0)
Requirement already satisfied: flatbuffers<=2.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (2.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (0.4.0)
Requirement already satisfied: setuptools in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (65.6.3)
Requirement already satisfied: keras-preprocessing<=1.1.1 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (1.1.2)
Requirement already satisfied: tensorboard<2.11,>=2.10 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (2.10.0)
Requirement already satisfied: google-pasta<=0.1.1 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (0.2.0)
Requirement already satisfied: wrapt<=1.11.0 in c:\users\kuvsh\appdata\roaming\python\python310\site-packages (from TensorFlow) (1.14.1)
Requirement already satisfied: termcolor<=1.1.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (2.1.0)
Requirement already satisfied: keras<2.11,>=2.10.0 in c:\users\kuvsh\.conda\envs\py__117\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 1.4 MB/s eta 0:00:00
Requirement already satisfied: astunparse<=1.6.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (1.6.3)
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (2.10.0)
Requirement already satisfied: typing-extensions<=3.6.6 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from TensorFlow) (4.4.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from astunparse<=1.6.0->TensorFlow) (0.37.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from tensorflow-estimator<2.11,>=2.10->TensorFlow) (2.6.0)
Requirement already satisfied: tensorboard-plugin-wit<=1.6.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from tensorboard<2.11,>=2.10->TensorFlow) (1.8.1)
Requirement already satisfied: werkzeug<=1.0.1 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from tensorboard<2.11,>=2.10->TensorFlow) (2.2.2)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from tensorboard<2.11,>=2.10->TensorFlow) (0.6.1)
Requirement already satisfied: markdown<=2.6.8 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from tensorboard<2.11,>=2.10->TensorFlow) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from tensorboard<2.11,>=2.10->TensorFlow) (2.28.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\users\kuvsh\.conda\envs\py__117\lib\site-packages (from tensorboard<2.11,>=2.10->TensorFlow) (0.4.4)
```

Рисунок 17.10 – Процесс установки пакета TensorFlow с помощью менеджера пакетов pip

9. Сформируйте файлы requirements.txt и environment.yml.

Проанализируйте содержимое этих файлов.

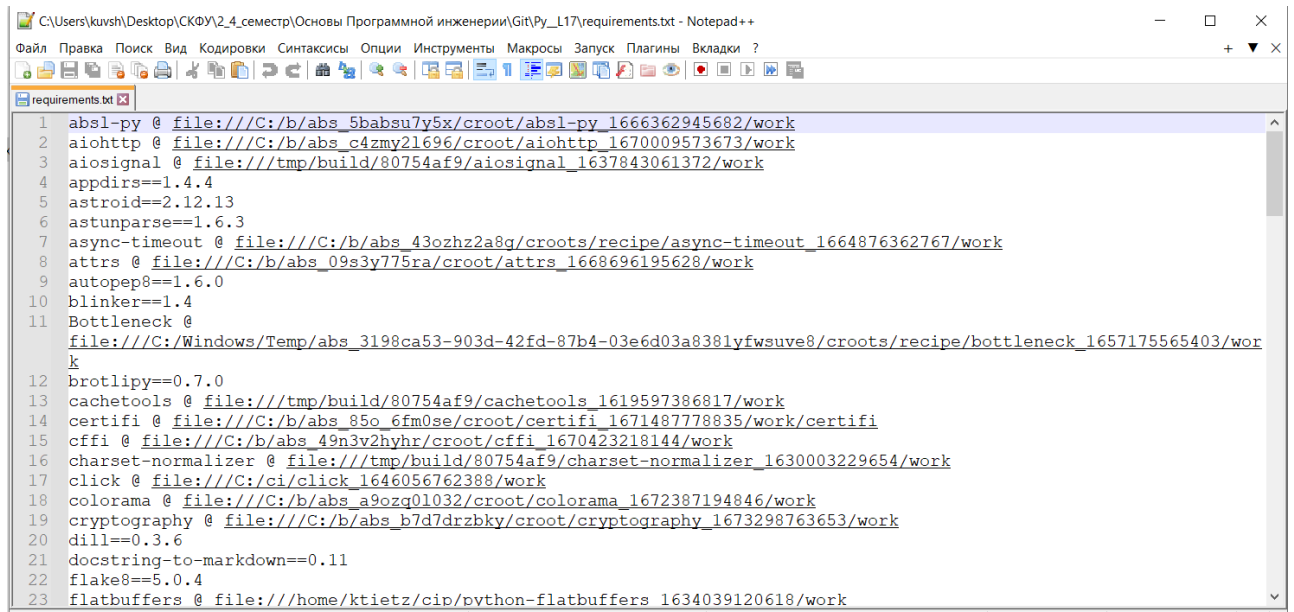
requirements.txt - файл хранения зависимостей, список всех модулей и пакетов Python, которые нужны для полноценной работы вашей программы. Его использование позволяет легко отслеживать весь перечень нужных компонентов, избавляя пользователей от необходимости их ручного поиска и установки.

environment.yml – файл окружения, позволяющий воссоздать окружение в любой нужный момент.

```
Anaconda Prompt (Anaconda3) - conda install pip - conda install NumPy - conda install Pandas - conda install SciPy - conda install TensorFlow

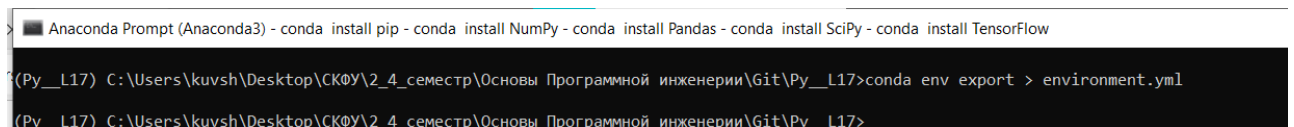
(Py_L17) C:\Users\kuvsh>cd C:\Users\kuvsh\Desktop\СКОУ\2_4_семестр\Основы Программной инженерии\Git\Py_L17
(Py_L17) C:\Users\kuvsh\Desktop\СКОУ\2_4_семестр\Основы Программной инженерии\Git\Py_L17>pip freeze > requirements.txt
(Py_L17) C:\Users\kuvsh\Desktop\СКОУ\2_4_семестр\Основы Программной инженерии\Git\Py_L17>_
```

Рисунок 17.11 – Процесс создания файла requirements.txt



```
1  absl-py @ file:///C:/b/abs_5babsu7y5x/croot/absl-py_1666362945682/work
2  aiohttp @ file:///C:/b/abs_c4zmy21696/croot/aiohttp_1670009573673/work
3  aiosignal @ file:///tmp/build/80754af9/aiosignal_1637843061372/work
4  appdirs==1.4.4
5  astroid==2.12.13
6  astunparse==1.6.3
7  async-timeout @ file:///C:/b/abs_43ozhz2a8g/croots/recipe/async-timeout_1664876362767/work
8  attrs @ file:///C:/b/abs_09s3y775ra/croot/attrs_1668696195628/work
9  autopep8==1.6.0
10 blinker==1.4
11 Bottleneck @
   file:///C:/Windows/Temp/abs_3198ca53-903d-42fd-87b4-03e6d03a8381yfwsuve8/croots/recipe/bottleneck_1657175565403/wor
   k
12 brotli==0.7.0
13 cachetools @ file:///tmp/build/80754af9/cachetools_1619597386817/work
14 certifi @ file:///C:/b/abs_85o_6fm0se/croot/certifi_1671487778835/work/certifi
15 cffi @ file:///C:/b/abs_49n3v2hyhr/croot/cffi_1670423218144/work
16 charset-normalizer @ file:///tmp/build/80754af9/charset-normalizer_1630003229654/work
17 click @ file:///C:/ci/click_1646056762388/work
18 colorama @ file:///C:/b/abs_a9ozq01032/croot/colorama_1672387194846/work
19 cryptography @ file:///C:/b/abs_b7d7drzbky/croot/cryptography_1673298763653/work
20 dill==0.3.6
21 docstring-to-markdown==0.11
22 flake8==5.0.4
23 flatbuffers @ file:///home/ktietz/cip/python-flatbuffers_1634039120618/work
```

Рисунок 17.12 – Файл requirements.txt

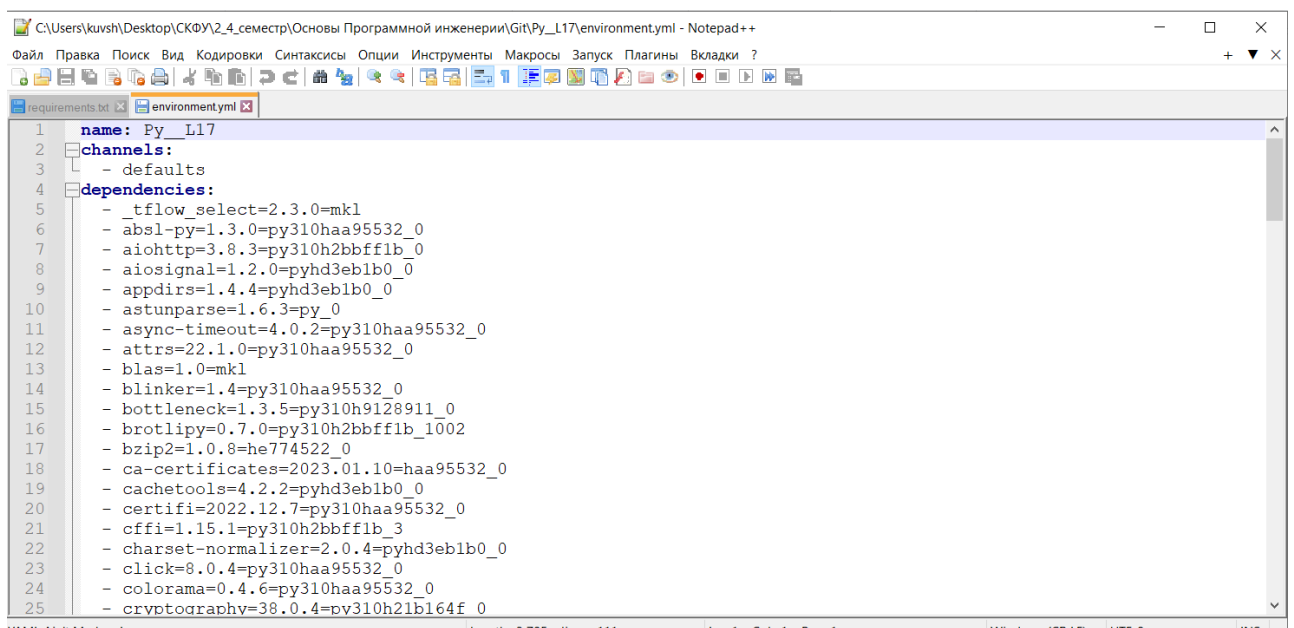


```
Anaconda Prompt (Anaconda3) - conda install pip - conda install NumPy - conda install Pandas - conda install SciPy - conda install TensorFlow

(Py_L17) C:\Users\kuvsh\Desktop\КФУ\2_4_семестр\Основы Программной инженерии\Git\Py_L17>conda env export > environment.yml

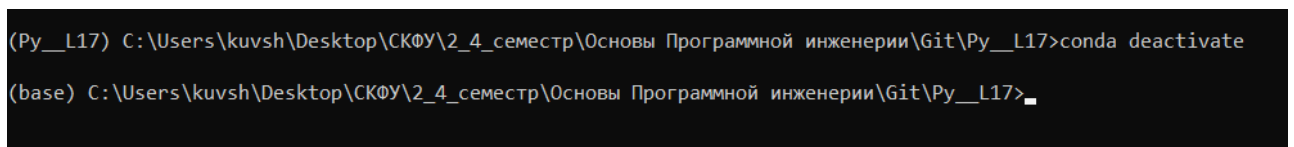
(Py_L17) C:\Users\kuvsh\Desktop\КФУ\2_4_семестр\Основы Программной инженерии\Git\Py_L17>
```

Рисунок 17.13 – Процесс создания файла environment.yml



```
1  name: Py_L17
2  channels:
3    - defaults
4  dependencies:
5    - _tfselect=2.3.0=mkl
6    - absl-py=1.3.0=py310haa95532_0
7    - aiohttp=3.8.3=py310h2bbff1b_0
8    - aiosignal=1.2.0=pyhd3eb1b0_0
9    - appdirs=1.4.4=pyhd3eb1b0_0
10   - astunparse=1.6.3=py_0
11   - async-timeout=4.0.2=py310haa95532_0
12   - attrs=22.1.0=py310haa95532_0
13   - blas=1.0=mkl
14   - blinker=1.4=py310haa95532_0
15   - bottleneck=1.3.5=py310h9128911_0
16   - brotli=0.7.0=py310h2bbff1b_1002
17   - bzip2=1.0.8=he774522_0
18   - ca-certificates=2023.01.10=haa95532_0
19   - cachetools=4.2.2=pyhd3eb1b0_0
20   - certifi=2022.12.7=py310haa95532_0
21   - cffi=1.15.1=py310h2bbff1b_3
22   - charset-normalizer=2.0.4=pyhd3eb1b0_0
23   - click=8.0.4=py310haa95532_0
24   - colorama=0.4.6=py310haa95532_0
25   - cryptography=38.0.4=py310h21b164f_0
```

Рисунок 17.14 – Файл environment.yml



```
(Py_L17) C:\Users\kuvsh\Desktop\КФУ\2_4_семестр\Основы Программной инженерии\Git\Py_L17>conda deactivate

(base) C:\Users\kuvsh\Desktop\КФУ\2_4_семестр\Основы Программной инженерии\Git\Py_L17>
```

Рисунок 17.15 – отключение среды

10. Зафиксируйте сделанные изменения в репозитории.
11. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
12. Выполните слияние ветки для разработки с веткой master/main.
13. Отправьте сделанные изменения на сервер GitHub.
14. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Контрольные вопросы

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

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*.

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.

Универсальный способ

Будем считать, что *Python* у вас уже установлен, теперь необходимо установить *pip*. Для того, чтобы это сделать, скачайте скрипт *get-pip.py*

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

и выполните его.

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

Способ для *Linux*

Если вы используете *Linux*, то для установки *pip* можно воспользоваться имеющимся в вашем дистрибутиве пакетным менеджером. Ниже будут перечислены команды для ряда *Linux* систем, запускающие установку *pip* (будем рассматривать только *Python 3*, т.к. *Python 2* уже морально устарел, а его поддержка и развитие будут прекращены после 2020 года).

Fedora

```
$ sudo dnf install python3 python3-wheel
```

openSUSE

```
$ sudo zypper install python3-pip python3-setuptools python3-wheel
```

Debian/Ubuntu

```
$ sudo apt install python3-venv python3-pip
```

Arch Linux

```
$ sudo pacman -S python-pip
```

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
```

Установка из альтернативного индекса

```
$ pip install --index-url http://pypi.org/ ProjectName
```

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
```

11. Каковы причины появления виртуальных окружений в языке Python?

1. Проблема обратной совместимости

Некоторые операционные системы, например, Linux и MacOS используют содержащиеся в них предустановленные интерпретаторы Python.

Обновив или изменив самостоятельно версию какого-то установленного глобально пакета мы можем непреднамеренно сломать работу утилит и приложений из дистрибутива операционной системы.

2. Проблема коллективной разработки

Если разработчик работает над проектом не один, а с командой, ему нужно передавать и получать список зависимостей, а также обновлять их на своем компьютере таким образом, чтобы не нарушалась работа других его проектов. Значит нам нужен механизм, который вместе с обменом проектами быстро устанавливал бы локально и все необходимые для них пакеты, при этом не мешая работе других проектов.

12. Каковы основные этапы работы с виртуальными окружениями?

Вот 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 виртуальное окружение 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`?

Для создания виртуального окружения достаточно дать команду в формате:

```
python3 -m venv <путь к папке виртуального окружения>
```

Обычно папку для виртуального окружения называют `env` или `venv`.

Создадим виртуальное окружение в папке проекта. Для этого перейдём в корень любого проекта на Python `>= 3.3` и дадим команду:

```
$ python3 -m venv env
```


После её выполнения создастся папка `env` с виртуальным окружением. Вот пример структуры такой папки для Windows (скрыты файлы пакетов и папки кэширования Python):

```
└─env
  │   pyvenv.cfg
  │   └─Include
  │   └─Lib
  │       └─site-packages
  │           │
  │           └─pip
  │               └─pip-19.2.3.dist-info
  │               └─pkg_resources
  │               └─setuptools
  │               └─setuptools-41.2.0.dist-info
  └─Scripts
      activate
      activate.bat
      Activate.ps1
      deactivate.bat
      easy_install-3.7.exe
      easy_install.exe
      pip.exe
      pip3.7.exe
      pip3.exe
      python.exe
      pythonw.exe
```

Чтобы активировать виртуальное окружение под Windows команда выглядит иначе:

```
> env\\Scripts\\activate
Просто под windows мы вызываем скрипт активации напрямую.
```

После активации приглашение консоли изменится. В его начале в круглых скобках будет отображаться имя папки с виртуальным окружением, например, возможный вариант для Linux:

```
(env) user@user:~/venv_test/proj3$
```

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.

```
$ python3 -m venv /home/user/envs/project1_env
```

Виртуальное окружение благополучно создалось. Давайте активируем его:

```
$ source /home/user/envs/project1_env/bin/activate
```

Виртуальное окружение активировано.

Чтобы переключиться с одного окружения на другое нам нужно выполнить команду деактивации и команду активации другого виртуального окружения, например, так:

```
$ deactivate
$ source /home/user/envs/project1_env2/bin/activate
```

Команда `deactivate` всегда выполняется из контекста текущего виртуального окружения. По этой причине для неё не нужно указывать полный путь.

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

Зачем нам нужно уметь работать с утилитой `virtualenv`? Ведь мы уже научились работать со стандартным модулем Python `venv`. Просто он очень распространён и поддерживает большее число вариантов и версий интерпретатора Python, например, PyPy и CPython.

Для начала пакет нужно установить. Установку можно выполнить командой:

```
# Для python 3
python3 -m pip install virtualenv

# Для единственного python
python -m pip install virtualenv
```

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
```

Для операционной системы Windows:

```
> env\\scripts\\activate

(env) > deactivate
```

С созданием, активацией и деактивацией виртуального окружения разобрались. А как обмениваться и поднимать чужое виртуальное окружение?

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

`pipenv` — как `pip`, только удобнее (semakin.dev)

17. Каково назначение файла requirements.txt ? Как создать этот файл?

Какой он имеет формат?

Само виртуальное окружение никуда переносить не нужно. Требуется возможность формирования и развертывания пакетных зависимостей. Для формирования и развертывания пакетных зависимостей используется утилита pip.

Просмотреть список зависимостей мы можем командой:

```
pip freeze
```

Что бы его сохранить, нужно перенаправить вывод команды в файл:

```
pip freeze > requirements.txt
```

Имя файла хранения зависимостей `requirements.txt` выбрано не зря. Оно является стандартной договоренностью и используется некоторыми утилитами автоматически.

Установка пакетов из файла зависимостей в новом виртуальном окружении так же выполняется одной командой:

```
pip install -r 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 (по сравнению с pip).

Существуют также некоторые различия, если вы заинтересованы в создании собственных пакетов. Например, pip создан на основе setuptools, тогда как conda использует свой собственный формат, который имеет некоторые преимущества (например, статическая компиляция пакета).

18. В какие дистрибутивы Python входит пакетный менеджер conda?

Все чаще среди Python-разработчиков заходит речь о менеджере пакетов conda, включенный в состав дистрибутивов Anaconda и Miniconda. JetBrains включил этот инструмент в состав PyCharm.

19. Как создать виртуальное окружение conda?

Для Windows, если используется дистрибутив Anaconda, то необходимо вначале запустить консоль Anaconda Powershell Prompt. Делается это из системного меню, посредством выбора следующих пунктов: *Пуск → Anaconda3 (64-bit) → Anaconda Powershell Prompt (Anaconda3)*. В результате будет отображено окно консоли, показанное на рисунке.

Обратите на имя виртуального окружения по умолчанию, которым в данном случае является *base*. В этом окне необходимо ввести следующую последовательность команд:

```
mkdir %PROJ_NAME%  
cd %PROJ_NAME%  
copy NUL > main.py
```

2. Создайте чистое conda-окружение с таким же именем, как директория проекта, и затем активируйте его. Для Linux это последовательность команд:

```
source deactivate  
conda create -n $PROJ_NAME python=3.7  
source activate $PROJ_NAME
```

Тогда как для Windows эта последовательность будет несколько иной:

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

После выполнения этих команд в приглашении ввода должно отобразиться имя созданного виртуального окружения.

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

3. Установите пакеты, необходимые для реализации проекта.

```
conda install django, pandas
```

4. Периодически экспортируйте параметры окружения. Экспортируйте после установки, перед каждым большим или маленьким коммитом:

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

21. Как деактивировать и удалить виртуальное окружение conda?

Для Windows необходимо использовать следующую команду:

```
conda deactivate
```

Если вы хотите удалить только что созданное окружение, выполните:

```
conda remove -n $PROJ_NAME
```

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

Файл environment.yml позволит воссоздать окружение в любой нужный момент.

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

6. Файл environment.yml позволит воссоздать окружение в любой нужный момент.

Достаточно набрать:

```
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.

[Установка Anaconda + интеграция с Pycharm - Русские Блоги \(russianblogs.com\)](#)

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

Эти файлы дают возможность восстановить виртуальное окружение на другом устройстве/компьютере.