

**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

**«Установка пакетов в Python. Виртуальные окружения»
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » февраля 2023 г.

Подпись студента _____

Работа защищена

« » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2023

Цель работы:

Приобретение навыков по работе с менеджером пакетов `pip` и виртуальными окружениями с помощью языка программирования Python версии 3.x.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: <https://github.com/afk552/lab2.14>

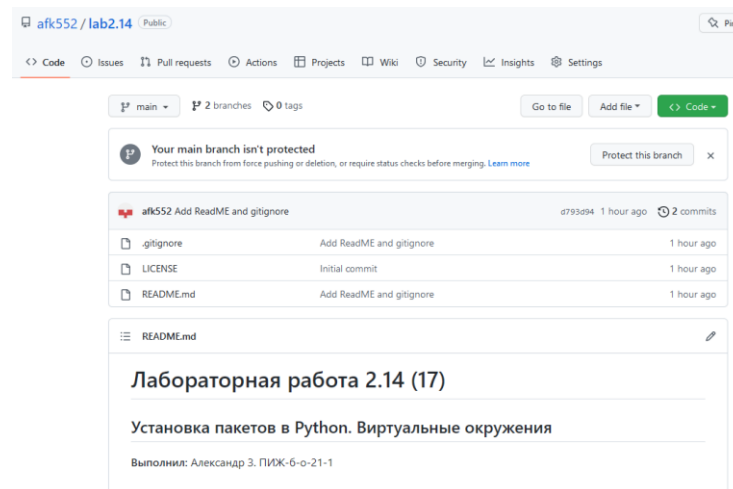


Рисунок 1 – Удаленный репозиторий на GitHub

Дополните файл `.gitignore` необходимыми правилами для работы с IDE PyCharm, рисунок 2.

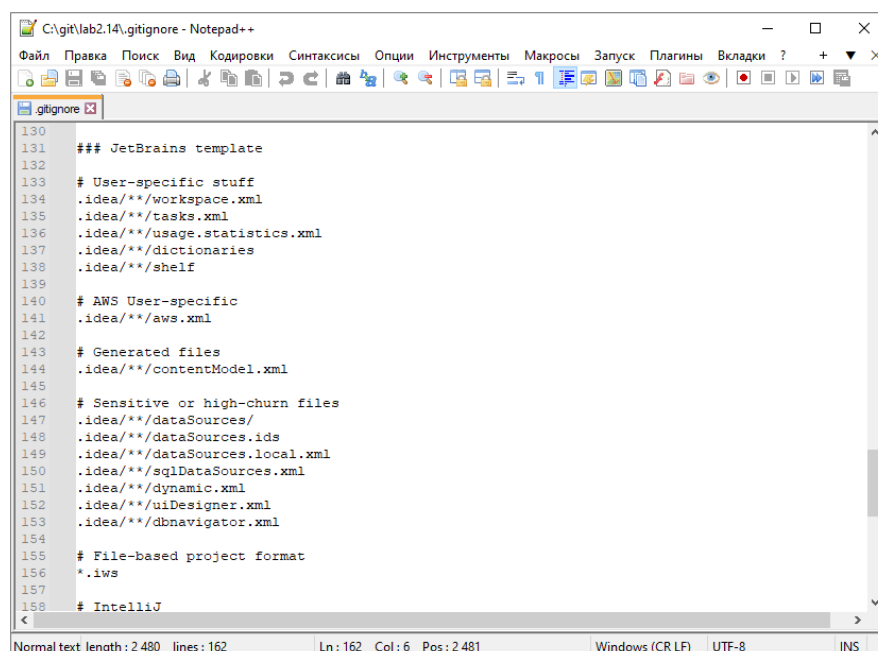
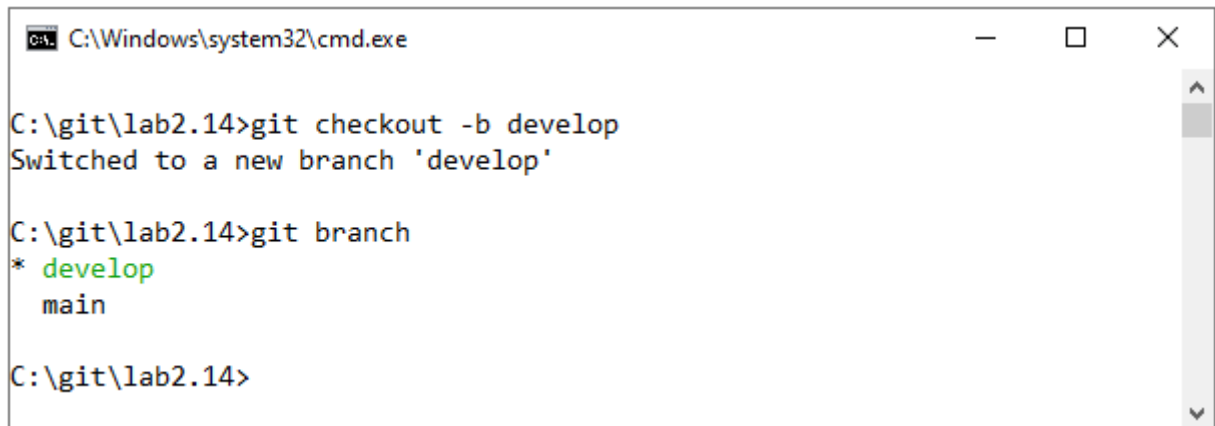


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствии с моделью ветвления git-flow, рисунок 3.



```
C:\Windows\system32\cmd.exe

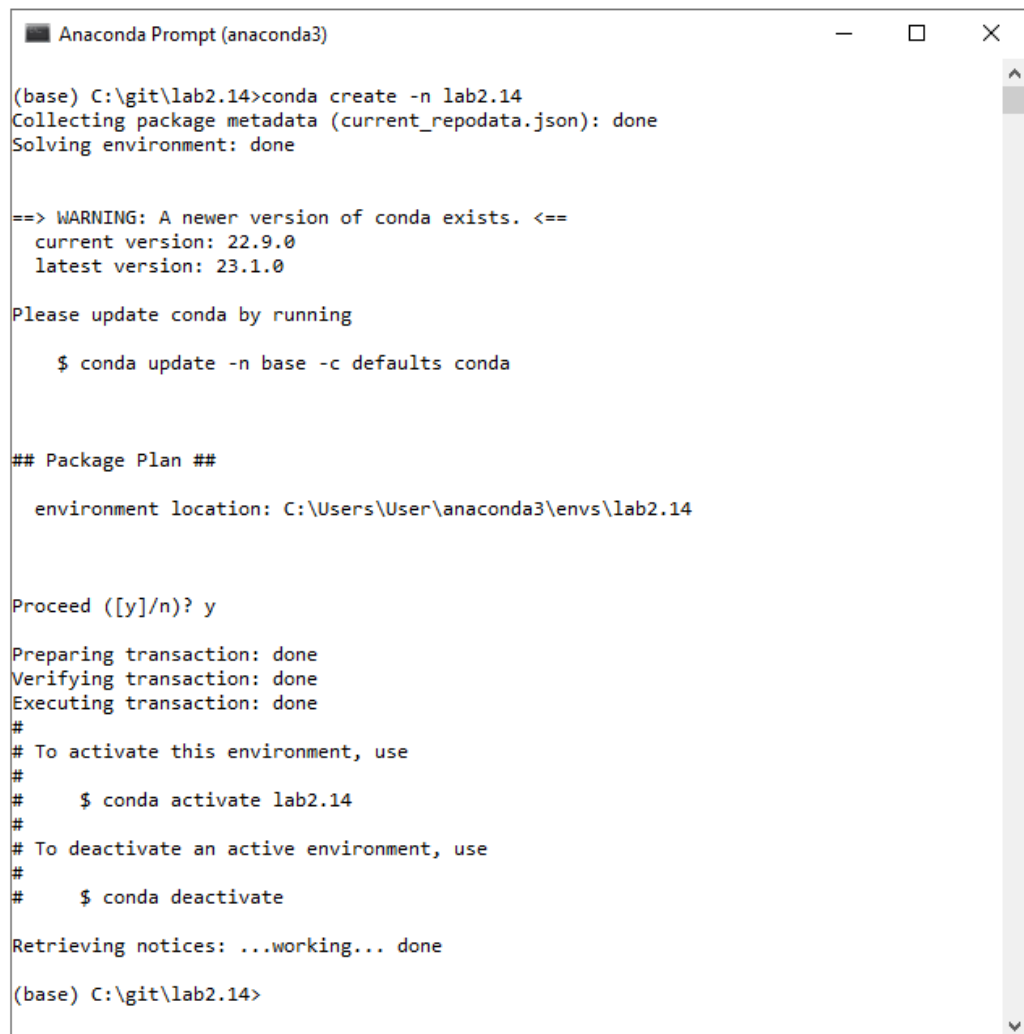
C:\git\lab2.14>git checkout -b develop
Switched to a new branch 'develop'

C:\git\lab2.14>git branch
* develop
  main

C:\git\lab2.14>
```

Рисунок 3 – Окно командной строки

Создайте виртуальное окружение Anaconda с именем репозитория. Создание репозитория – рисунок 4, активация – рисунок 5.



```
Anaconda Prompt (anaconda3)

(base) C:\git\lab2.14>conda create -n lab2.14
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\User\anaconda3\envs\lab2.14

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate lab2.14
#
# To deactivate an active environment, use
#
#     $ conda deactivate
#
Retrieving notices: ...working... done

(base) C:\git\lab2.14>
```

Рисунок 4 – Создание нового виртуального окружения Anaconda

```
Anaconda Prompt (anaconda3)

(base) C:\git\lab2.14>conda activate lab2.14

(lab2.14) C:\git\lab2.14>
```

Рисунок 5 – Активация нового виртуального окружения Anaconda

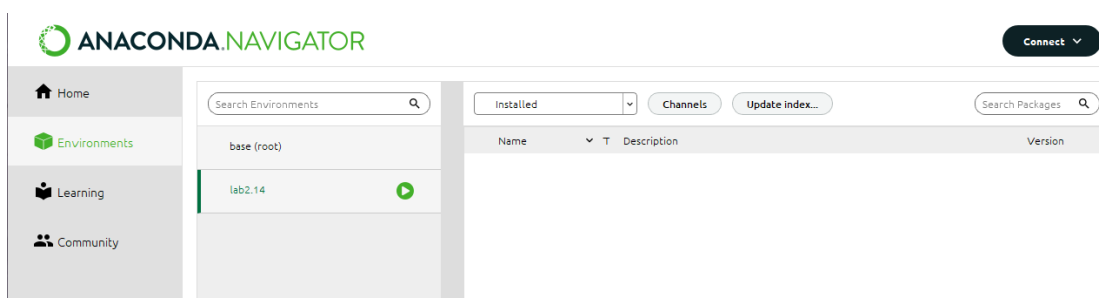


Рисунок 6 – Виртуальное окружение в Anaconda Navigator

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

```
Anaconda Prompt (anaconda3)

(lab2.14) C:\git\lab2.14>python -m pip install --upgrade pip
Requirement already satisfied: pip in c:\python\python311\lib\site-packages (22.3.1)
Collecting pip
  Using cached pip-23.0-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3.1
    Uninstalling pip-22.3.1:
      Successfully uninstalled pip-22.3.1
Successfully installed pip-23.0

(lab2.14) C:\git\lab2.14>
```

Рисунок 7 – Обновление `pip`

```
Anaconda Prompt (anaconda3)

(lab2.14) C:\git\lab2.14>pip install numpy
Collecting numpy
  Downloading numpy-1.24.2-cp311-cp311-win_amd64.whl (14.8 MB)
    ----- 14.8/14.8 MB 7.9 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.24.2

(lab2.14) C:\git\lab2.14>
```

Рисунок 8 – Установка `NumPy`

```
Anaconda Prompt (anaconda3)

(lab2.14) C:\git\lab2.14>pip install pandas
Collecting pandas
  Downloading pandas-1.5.3-cp311-cp311-win_amd64.whl (10.3 MB)
    ----- 10.3/10.3 MB 7.6 MB/s eta 0:00:00
Collecting python-dateutil>=2.8.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    ----- 247.7/247.7 kB 14.8 MB/s eta 0:00:00
Collecting pytz>=2020.1
  Downloading pytz-2022.7.1-py2.py3-none-any.whl (499 kB)
    ----- 499.4/499.4 kB 10.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.21.0 in c:\python\python311\lib\site-packages (from pandas) (1.24.2)
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, six, python-dateutil, pandas
Successfully installed pandas-1.5.3 python-dateutil-2.8.2 pytz-2022.7.1 six-1.16.0

(lab2.14) C:\git\lab2.14>
```

Рисунок 9 – Установка Pandas

```
Anaconda Prompt (anaconda3)

(lab2.14) C:\git\lab2.14>pip install scipy
Collecting scipy
  Downloading scipy-1.10.0-cp311-cp311-win_amd64.whl (42.2 MB)
    ----- 42.2/42.2 MB 6.1 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.27.0,>=1.19.5 in c:\python\python311\lib\site-packages (from scipy) (1.24.2)
Installing collected packages: scipy
Successfully installed scipy-1.10.0

(lab2.14) C:\git\lab2.14>
```

Рисунок 10 – Установка SciPy

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

```
Anaconda Prompt (anaconda3)

(lab2.14) C:\git\lab2.14>conda install TensorFlow
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: HTTPSConnectionPool(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."))

(lab2.14) C:\git\lab2.14>
```

Рисунок 11 – Попытка установки TensorFlow через Anaconda

В системе отсутствует OpenSSL, поэтому Anaconda не может осуществить защищенное соединение к репозиторию. Решение – скопировать библиотеки libcrypto-1_1-x64.dll и libssl-1_1-x64 из anaconda3\Library\bin в папку ...anaconda3\DLLs.

```
Выбрать Anaconda Prompt (anaconda3) - conda install TensorFlow

(lab2.14) C:\git\lab2.14>conda install TensorFlow
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
current version: 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\User\anaconda3\envs\lab2.14

added / updated specs:
- tensorflow

The following packages will be downloaded:

package | build
-----|-----
tf_select-2.3.0 | mkl 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
brotlipy-0.7.0 | py310h2bbff1b_1002 335 KB
ca-certificates-2023.01.10 | haa95532_0 121 KB
cachetools-4.2.2 | pyhd3eb1b0_0 13 KB
certifi-2022.12.7 | py310haa95532_0 149 KB
cffi-1.15.1 | py310h2bbff1b_3 239 KB
click-8.0.4 | py310haa95532_0 157 KB
colorama-0.4.6 | py310haa95532_0 32 KB
cryptography-38.0.4 | py310h21b164f_0 1.0 MB
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
giflib-5.2.1 | h8cc25b3_1 81 KB
google-auth-2.6.0 | pyhd3eb1b0_0 83 KB
google-auth-oauthlib-0.4.4 | pyhd3eb1b0_0 18 KB
google-pasta-0.2.0 | pyhd3eb1b0_0 46 KB
grpcio-1.42.0 | py310hc60d5dd_0 1.7 MB
h5py-3.7.0 | py310hfc34f40_0 822 KB
```

Рисунок 12 – Дополнительные пакеты, что будут загружены

```
Выбрать Anaconda Prompt (anaconda3) - conda install TensorFlow

Total: 147.9 MB

The following NEW packages will be INSTALLED:

tf_select | pkgs/main/win-64::tf_select-2.3.0-mkl None
absl-py | pkgs/main/win-64::absl-py-1.3.0-py310haa95532_0 None
aiohttp | pkgs/main/win-64::aiohttp-3.8.3-py310h2bbff1b_0 None
aiosignal | pkgs/main/noarch::aiosignal-1.2.0-pyhd3eb1b0_0 None
appdirs | pkgs/main/noarch::appdirs-1.4.4-pyhd3eb1b0_0 None
astunparse | pkgs/main/noarch::astunparse-1.6.3-py_0 None
async-timeout | pkgs/main/win-64::async-timeout-4.0.2-py310haa95532_0 None
attrs | pkgs/main/win-64::attrs-22.1.0-py310haa95532_0 None
blas | pkgs/main/win-64::blas-1.0-mkl None
blinker | pkgs/main/win-64::blinker-1.4-py310haa95532_0 None
brotlipy | pkgs/main/win-64::brotlipy-0.7.0-py310h2bbff1b_1002 None
bzip2 | pkgs/main/win-64::bzip2-1.0.8-he774522_0 None
ca-certificates | pkgs/main/win-64::ca-certificates-2023.01.10-haa95532_0 None
cachetools | pkgs/main/noarch::cachetools-4.2.2-pyhd3eb1b0_0 None
certifi | pkgs/main/win-64::certifi-2022.12.7-py310haa95532_0 None
cffi | pkgs/main/win-64::cffi-1.15.1-py310h2bbff1b_3 None
charset-normalizer | pkgs/main/noarch::charset-normalizer-2.0.4-pyhd3eb1b0_0 None
click | pkgs/main/win-64::click-8.0.4-py310haa95532_0 None
colorama | pkgs/main/win-64::colorama-0.4.6-py310haa95532_0 None
cryptography | pkgs/main/win-64::cryptography-38.0.4-py310h21b164f_0 None
fftw | pkgs/main/win-64::fftw-3.3.9-h2bbff1b_1 None
flatbuffers | pkgs/main/win-64::flatbuffers-2.0.0-h6c2663c_0 None
flit-core | pkgs/main/noarch::flit-core-3.6.0-pyhd3eb1b0_0 None
frozenlist | pkgs/main/win-64::frozenlist-1.3.3-py310h2bbff1b_0 None
gast | pkgs/main/noarch::gast-0.4.0-pyhd3eb1b0_0 None
giflib | pkgs/main/win-64::giflib-5.2.1-h8cc25b3_1 None
google-auth | pkgs/main/noarch::google-auth-2.6.0-pyhd3eb1b0_0 None
google-auth-oauthlib | pkgs/main/noarch::google-auth-oauthlib-0.4.4-pyhd3eb1b0_0 None
google-pasta | pkgs/main/noarch::google-pasta-0.2.0-pyhd3eb1b0_0 None
grpcio | pkgs/main/win-64::grpcio-1.42.0-py310hc60d5dd_0 None
h5py | pkgs/main/win-64::h5py-3.7.0-py310hfc34f40_0 None
hdf5 | pkgs/main/win-64::hdf5-1.10.6-h1756f20_1 None
icc_rt | pkgs/main/win-64::icc_rt-2022.1.0-h6049295_2 None
icu | pkgs/main/win-64::icu-58.2-ha925a31_3 None
idna | pkgs/main/win-64::idna-3.4-py310haa95532_0 None
intel-openmp | pkgs/main/win-64::intel-openmp-2021.4.0-haa95532_3556 None
jpeg | pkgs/main/win-64::jpeg-9e-h2bbff1b_0 None
keras | pkgs/main/win-64::keras-2.10.0-py310haa95532_0 None
keras-preprocessing | pkgs/main/noarch::keras-preprocessing-1.1.2-pyhd3eb1b0_0 None
libcurl | pkgs/main/win-64::libcurl-7.87.0-h86230a5_0 None
libffi | pkgs/main/win-64::libffi-3.4.2-hd77b12b_6 None
libpng | pkgs/main/win-64::libpng-1.6.37-h2a8f8b0_0 None
libprotobuf | pkgs/main/win-64::libprotobuf-3.20.3-h23ce88f_0 None
libssh2 | pkgs/main/win-64::libssh2-1.10.0-hcd444a_0 None
markdown | pkgs/main/win-64::markdown-3.4.1-py310haa95532_0 None
markupsafe | pkgs/main/win-64::markupsafe-2.1.1-py310h2bbff1b_0 None
mkl | pkgs/main/win-64::mkl-2021.4.0-haa95532_640 None
mkl-service | pkgs/main/win-64::mkl-service-2.4.0-py310h2bbff1b_0 None
mkl_fft | pkgs/main/win-64::mkl_fft-1.3.1-py310ha0764ea_0 None
mkl_random | pkgs/main/win-64::mkl_random-1.2.2-py310h4ed8f06_0 None
multidict | pkgs/main/win-64::multidict-6.0.2-py310h2bbff1b_0 None
numpy | pkgs/main/win-64::numpy-1.23.5-py310h60c9a35_0 None
```

Рисунок 13 – Дополнительные пакеты, что будут установлены



Рисунок 14 – Загрузка доп. пакетов

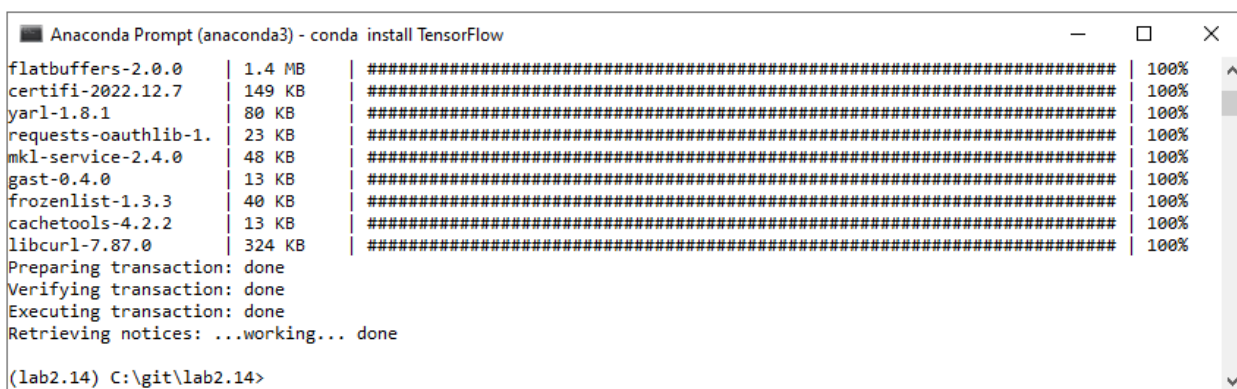


Рисунок 15 – Пакет TensorFlow установлен

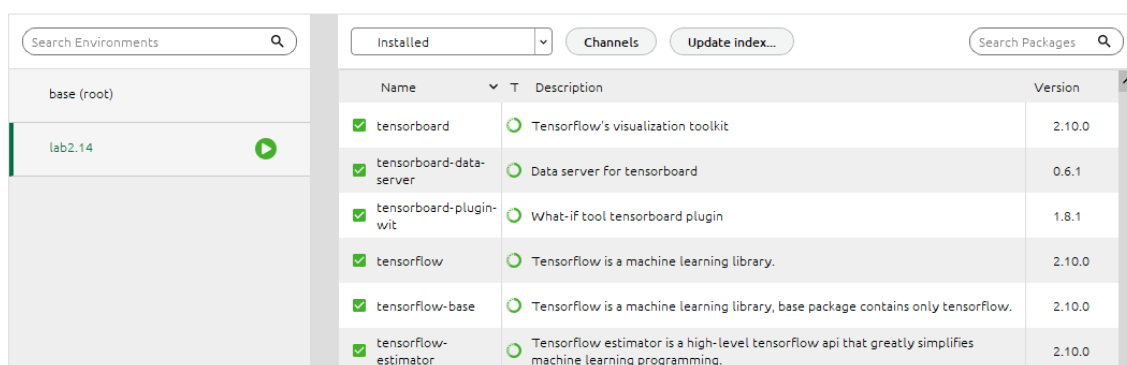
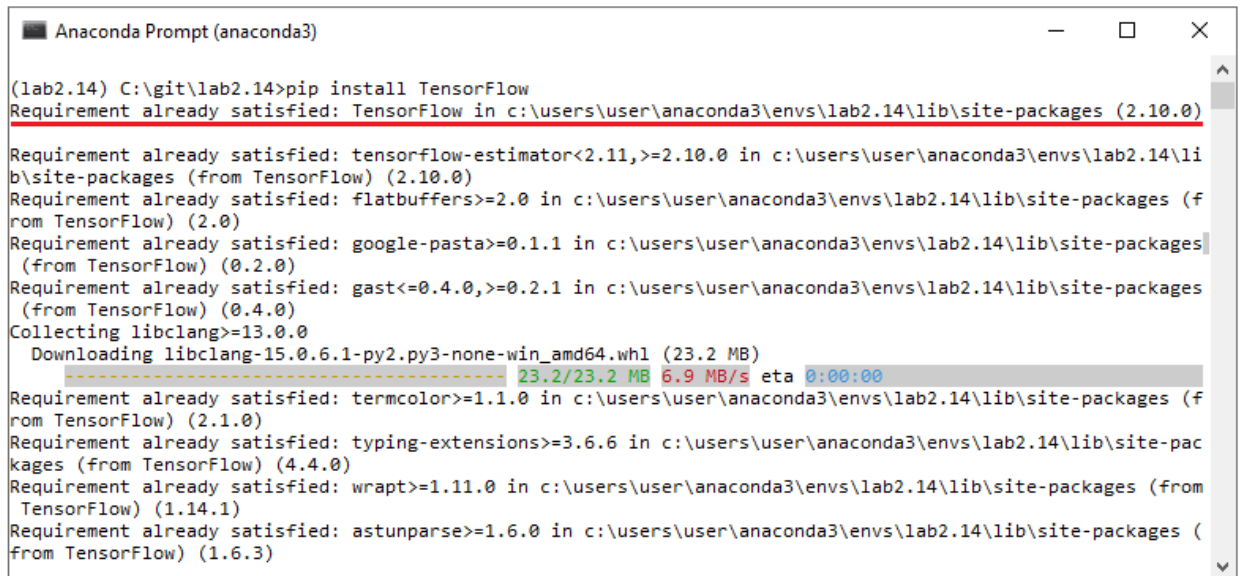


Рисунок 16 – Проверка установки TensorFlow через Anaconda Navigator

Попробуйте установить пакет TensorFlow с помощью менеджера пакетов pip.

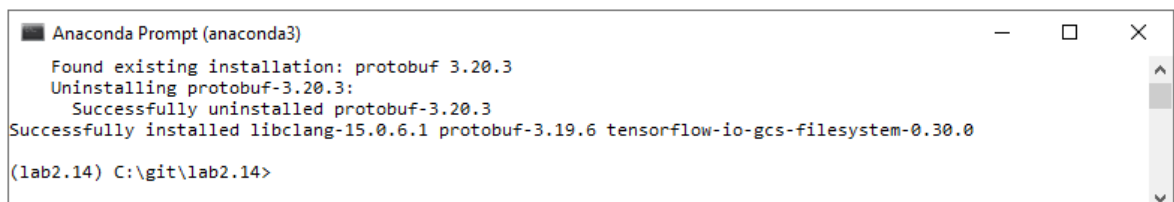


```
Anaconda Prompt (anaconda3)

(lab2.14) C:\git\lab2.14>pip install TensorFlow
Requirement already satisfied: TensorFlow in c:\users\user\anaconda3\envs\lab2.14\lib\site-packages (2.10.0)
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in c:\users\user\anaconda3\envs\lab2.14\lib\site-packages (from TensorFlow) (2.10.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\user\anaconda3\envs\lab2.14\lib\site-packages (from TensorFlow) (2.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\user\anaconda3\envs\lab2.14\lib\site-packages (from TensorFlow) (0.2.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\user\anaconda3\envs\lab2.14\lib\site-packages (from TensorFlow) (0.4.0)
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 6.9 MB/s eta 0:00:00
Requirement already satisfied: termcolor>=1.1.0 in c:\users\user\anaconda3\envs\lab2.14\lib\site-packages (from TensorFlow) (2.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\user\anaconda3\envs\lab2.14\lib\site-packages (from TensorFlow) (4.4.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\user\anaconda3\envs\lab2.14\lib\site-packages (from TensorFlow) (1.14.1)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\user\anaconda3\envs\lab2.14\lib\site-packages (from TensorFlow) (1.6.3)
```

Рисунок 17 – Установка TensorFlow через pip

В данном случае TensorFlow уже установлен, однако pip обновил/откатил версии некоторых доп. пакетов на те, которые находятся в репозитории pip.



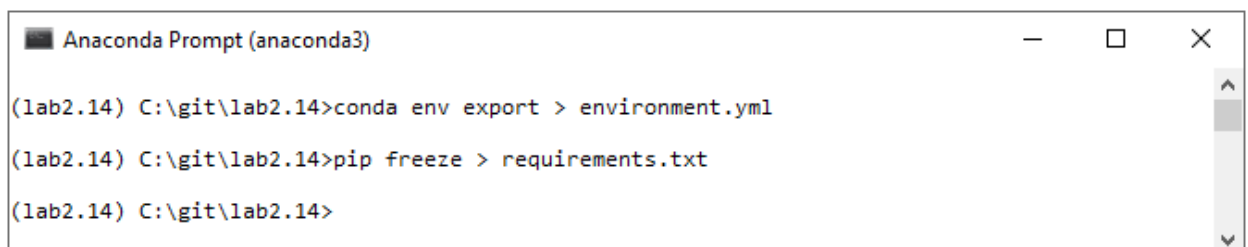
```
Anaconda Prompt (anaconda3)

Found existing installation: protobuf 3.20.3
Uninstalling protobuf-3.20.3:
  Successfully uninstalled protobuf-3.20.3
Successfully installed libclang-15.0.6.1 protobuf-3.19.6 tensorflow-io-gcs-filesystem-0.30.0

(lab2.14) C:\git\lab2.14>
```

Рисунок 18 – Установка TensorFlow через pip

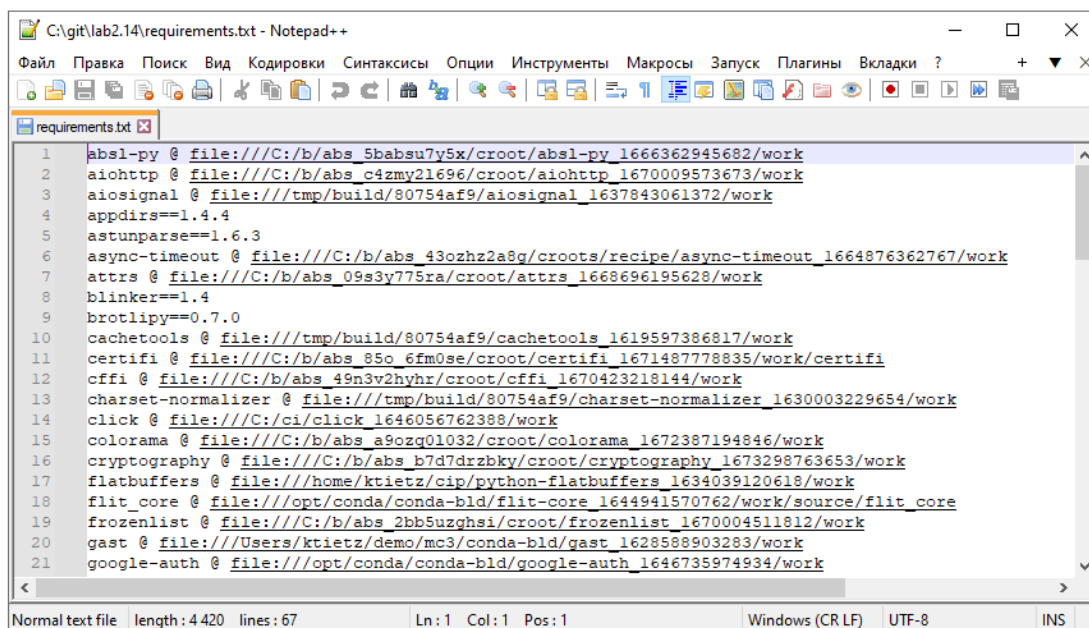
Сформируйте файлы requirements.txt и environment.yml. Проанализируйте содержимое этих файлов



```
Anaconda Prompt (anaconda3)

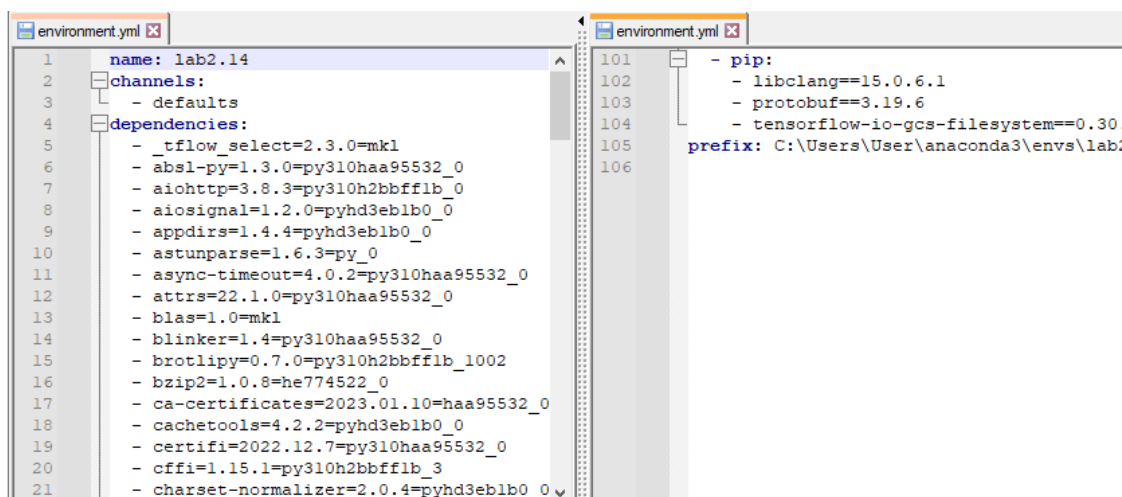
(lab2.14) C:\git\lab2.14>conda env export > environment.yml
(lab2.14) C:\git\lab2.14>pip freeze > requirements.txt
(lab2.14) C:\git\lab2.14>
```

Рисунок 19 – Создание файлов requirements.txt и environment.yml



```
1 abs1-py @ file:///C:/b/abs_5babsu7y5x/croot/abs1-py_1666362945682/work
2 aiohttp @ file:///C:/b/abs_c4zmy2l696/croot/aiohttp_1670009573673/work
3 aiosignal @ file:///tmp/build/80754af9/aiosignal_1637843061372/work
4 appdirs==1.4.4
5 astunparse==1.6.3
6 async-timeout @ file:///C:/b/abs_43ozhz2a8g/croots/recipe/async-timeout_1664876362767/work
7 attrs @ file:///C:/b/abs_09s3y775ra/croot/attrs_1668696195628/work
8 blinker==1.4
9 brotli==0.7.0
10 cachetools @ file:///tmp/build/80754af9/cachetools_1619597386817/work
11 certifi @ file:///C:/b/abs_85o_6fm0se/croot/certifi_1671487778835/work/certifi
12 cffi @ file:///C:/b/abs_49n3v2hyhr/croot/cffi_1670423218144/work
13 charset-normalizer @ file:///tmp/build/80754af9/charset-normalizer_1630003229654/work
14 click @ file:///C:/ci/click_1646056762388/work
15 colorama @ file:///C:/b/abs_a9ozq0l032/croot/colorama_1672387194846/work
16 cryptography @ file:///C:/b/abs_b7d7drzbky/croot/cryptography_1673298763653/work
17 flatbuffers @ file:///home/ktietz/cip/python-flatbuffers_1634039120618/work
18 flit_core @ file:///opt/conda/conda-bld/flit-core_1644941570762/work/source/flit_core
19 frozenlist @ file:///C:/b/abs_2bb5uzghsi/croot/frozenlist_1670004511812/work
20 gast @ file:///Users/ktietz/demo/mc3/conda-bld/gast_1628588903283/work
21 google-auth @ file:///opt/conda/conda-bld/google-auth_1646735974934/work
```

Рисунок 20 – Содержимое файла requirements.txt



```
1 name: lab2.14
2 channels:
3   - defaults
4 dependencies:
5   - _tflo_select=2.3.0=mk1
6   - abs1-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=mk1
14  - blinker=1.4=py310haa95532_0
15  - brotli=0.7.0=py310h2bbff1b_1002
16  - bzip2=1.0.8=he774522_0
17  - ca-certificates=2023.01.10=haa95532_0
18  - cachetools=4.2.2=pyhd3eb1b0_0
19  - certifi=2022.12.7=py310haa95532_0
20  - cffi=1.15.1=py310h2bbff1b_3
21  - charset-normalizer=2.0.4=pyhd3eb1b0_0
101 - pip:
102   - libclang==15.0.6.1
103   - protobuf==3.19.6
104   - tensorflow-io-gcs-filesystem==0.30.
105 prefix: C:\Users\User\anaconda3\envs\lab2
106
```

Рисунок 21 – Содержимое файла environment.yml

Просмотрев полученные файлы, можно заметить, что requirements.txt содержит список всех установленных пакетов с их версиями и ссылками на них. Файл environment.yml же содержит имя виртуального окружения, список зависимостей, отдельный список для пакетов pip и путь до виртуального окружения.

Вывод: В результате выполнения работы были изучено понятие виртуального окружения в Python, а также его установка и настройка, установка пакетов в него средствами pip и Anaconda.

Контрольные вопросы:

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

С помощью команды `pip install <имя пакета>` или `conda install <имя пакета>`.

2. Как осуществить установку менеджера пакетов `pip`?

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

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

Из Python Package Index (PyPI) – открытого репозитория для Python разработчиков.

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

```
pip install ProjectName
```

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

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

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

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

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

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

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

```
pip uninstall ProjectName
```

9. Как обновить установленный пакет с помощью pip?

`pip install --upgrade ProjectName`

10. Как отобразить список установленных пакетов с помощью pip?

`pip list`

11. Каковы причины появления виртуальных окружений в языке Python?

В системе для интерпретатора Python может быть установлена глобально только одна версия пакета. Это порождает ряд проблем.

1. Проблема обратной совместимости

Чем опасно обновление пакетов или версий интерпретатора? В новой версии пакета могут измениться названия функций или методов объектов и число и/или порядок передаваемых в них параметров. В следующей версии интерпретатора могут появиться новые ключевые слова, которые совпадают с именами переменных уже существующих приложений.

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?

Создание виртуального окружения

Активация

Деактивация

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>`

Чтобы активировать виртуальное окружение под Windows: `env\\Scripts\\activate`

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`

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

Для начала пакет нужно установить. Установку можно выполнить командой: `python3 -m pip install virtualenv`

Например, создание в текущей папке виртуального окружения для интерпретатора доступного через команду `python3` с названием папки окружения `env`: `virtualenv -p python3 env`

Активация и деактивация такая же, как у стандартной утилиты Python.

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

Установка – `pip install pipenv`

Создание оболочки виртуальной среды – `pipenv shell`

Установка нового пакета – `pipenv install ProjectName`

Установка пакета с репозитория – `pipenv install -e git+https://github.com/requests/requests.git#egg=requests`

Если необходимо установить зависимости, которые будут нужны только во время процесса разработки можно воспользоваться командой: `pipenv install pytest --dev`

Для переноса в рабочую среду необходимо заблокировать свою локальную среду командой – `pipenv lock`. Теперь, нужно перенести свой код проекта в рабочую среду включая файлы `Pipfile` и `Pipfile.lock`. Далее создать там собственную среду окружения командой `pipenv shell`. И далее установить все зависимости командой: `pipenv install --ignore-pipfile`

`--ignore-pipfile` говорит `Pipenv` игнорировать `Pipfile` для установки и использовать то, что находится в `Pipfile.lock`. Учитывая `Pipfile.lock`, `Pipenv` создаст ту же среду, которая была у нас, когда мы запустили блокировку зависимостей в `pipenv`.

Теперь допустим, что другой разработчик хочет внести некоторые дополнения в наш код. В этой ситуации он склонирует весь код себе на компьютер, включая `Pipfile`, и воспользуются этой командой для установки всех зависимостей у себя локально:

```
pipenv install --dev
```

Эта команда установит все зависимости, необходимые для разработки, которые включают в себя как обычные зависимости, так и те, которые вы указали в аргументе `--dev` во время установки.

Мы также можем отобразить граф зависимостей, воспользовавшись командой `pipenv graph`

Проверить наличие уязвимостей безопасности (и требований PEP 508) в вашей среде: `pipenv check`

Полностью стереть все установленные пакеты из виртуальной среды: `pipenv uninstall --all`

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

Все пакеты, которые вы установили перед выполнением команды и предположительно использовали в каком-либо проекте, будут перечислены в файле с именем «requirements.txt». Формат файла – текстовый (.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).

18. В какие дистрибутивы Python входит пакетный менеджер conda?
Anaconda и Miniconda.

19. Как создать виртуальное окружение conda?

conda create -n <name>

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

Активация: conda activate <name>

Установка пакета в виртуальное окружение: conda install <имя пакета>

21. Как деактивировать и удалить виртуальное окружение conda?

Деактивация: `conda deactivate`

Удаление: `conda remove -n <name>`

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

Экспорт: `conda env export > environment.yml`

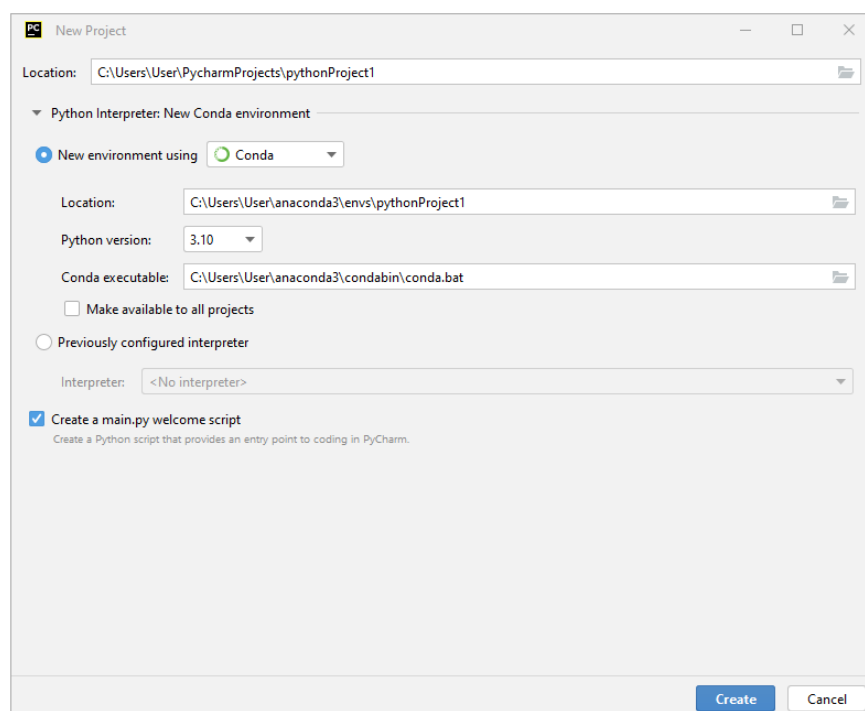
В нем хранятся параметры окружения и зависимости.

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

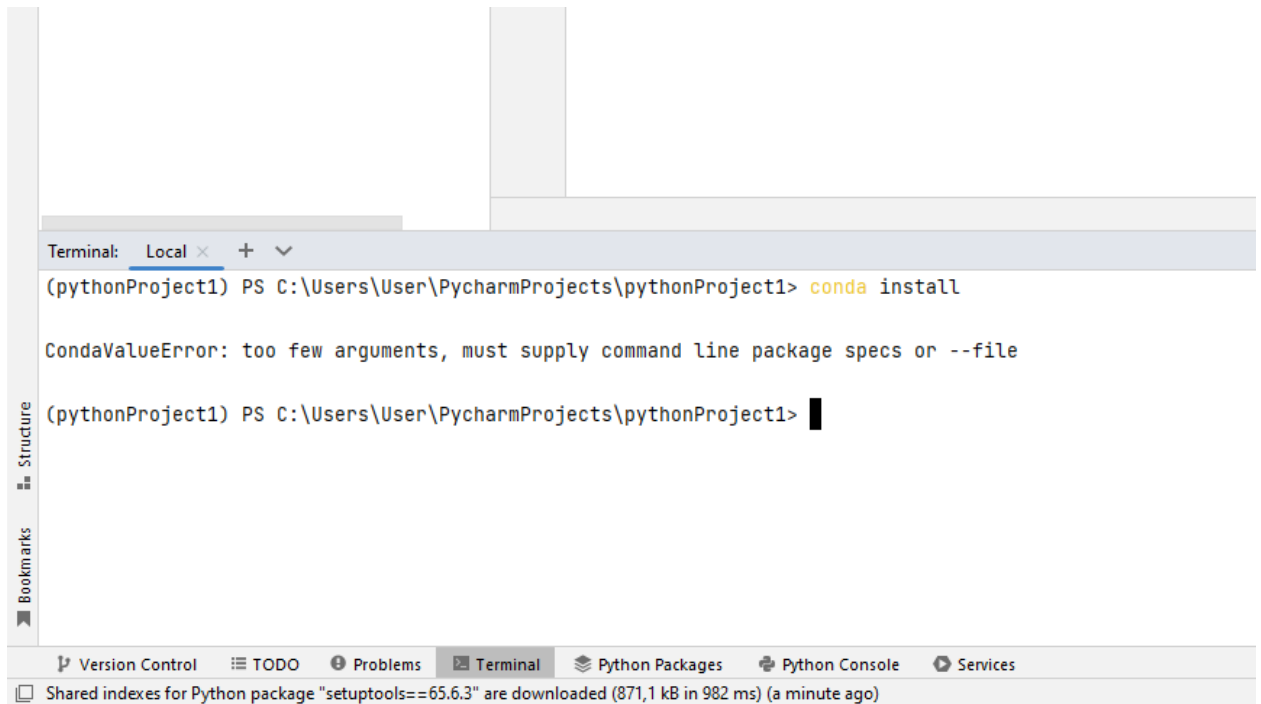
`conda env create -f environment.yml`

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

Создание нового проекта PyCharm и виртуального окружения Conda для него (необходимо в «New environment using» выбрать «conda» и необходимую версию Python).



Созданное виртуальное окружение уже активировано, и мы можем им управлять через Terminal. Например, установить какой-нибудь пакет.



```
Terminal: Local x + v
(pythonProject1) PS C:\Users\User\PycharmProjects\pythonProject1> conda install

CondaValueError: too few arguments, must supply command line package specs or --file

(pythonProject1) PS C:\Users\User\PycharmProjects\pythonProject1> █
```

The screenshot shows the PyCharm IDE interface. On the left is the 'Structure' and 'Bookmarks' sidebar. The main area is a terminal window titled 'Terminal: Local x + v'. It shows a PowerShell prompt for 'pythonProject1' at 'C:\Users\User\PycharmProjects\pythonProject1'. The command 'conda install' has been entered, resulting in a 'CondaValueError: too few arguments, must supply command line package specs or --file'. The prompt is now ready for the next command. At the bottom, the status bar shows tabs for 'Version Control', 'TODO', 'Problems', 'Terminal', 'Python Packages', 'Python Console', and 'Services'. A message at the very bottom states: 'Shared indexes for Python package "setuptools==65.6.3" are downloaded (871,1 kB in 982 ms) (a minute ago)'.

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

Для того, чтобы проект, использующий Python и его пакеты работал корректно. Файлы содержат как названия пакетов, так и их конкретные версии. Также эти файлы нужны для совместной работы над общим проектом, чтобы сразу развернуть окружение со всеми необходимыми зависимостями и приступить к работе.