

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-  
СКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Основы кроссплатформенного программирования**

**Отчет по лабораторной работе №2.14**

Тема: «Установка пакетов в Python. Виртуальные окружения»

Выполнил студент группы

ИВТ-б-о-21-1

Назаров Н.Ю. «    » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

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

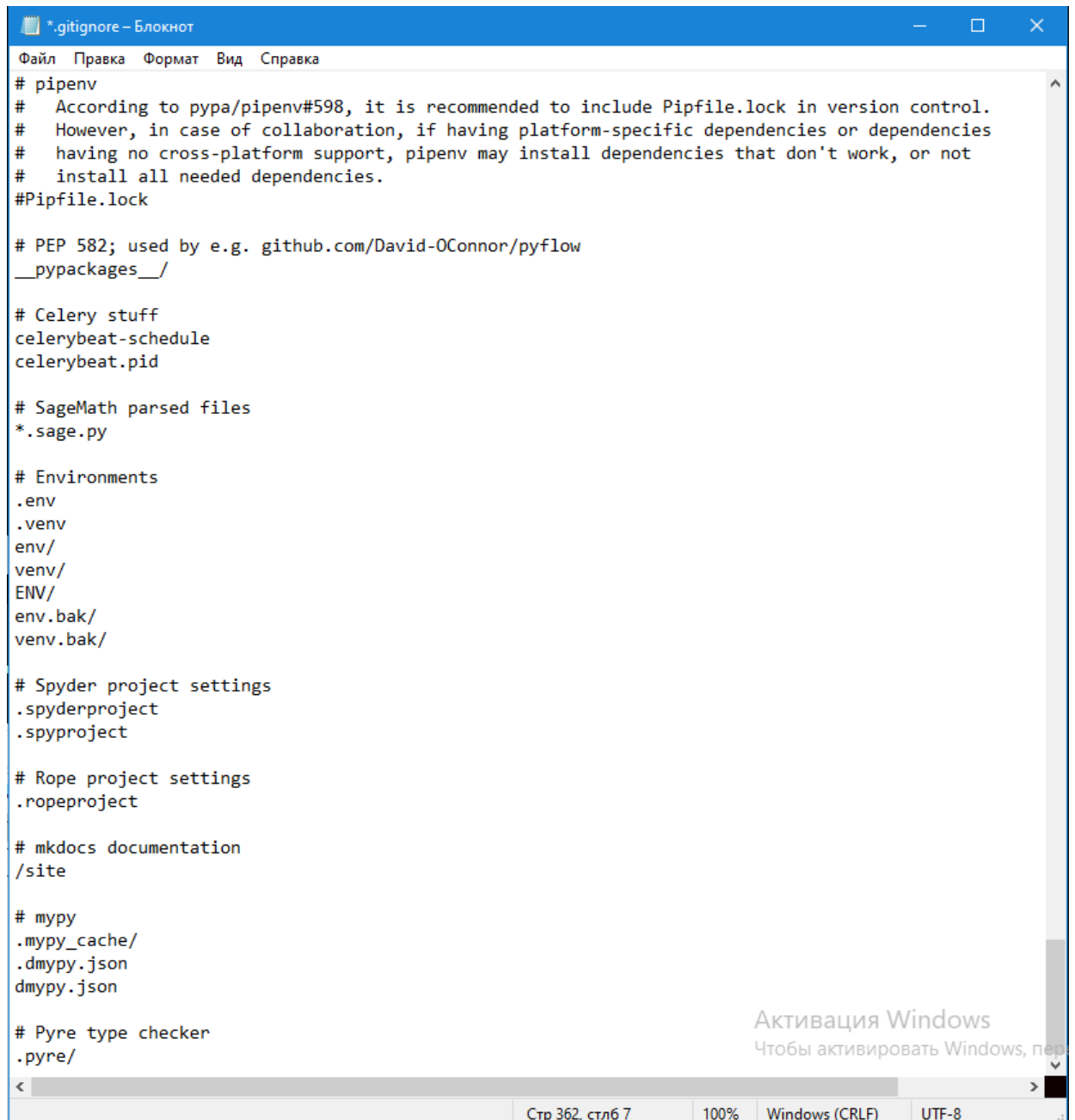
Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А.

\_\_\_\_\_  
(подпись)

## Выполнение работы



```
*.gitignore – Блокнот
Файл  Правка  Формат  Вид  Справка

# pipenv
# According to pya/pipenv#598, it is recommended to include Pipfile.lock in version control.
# However, in case of collaboration, if having platform-specific dependencies or dependencies
# having no cross-platform support, pipenv may install dependencies that don't work, or not
# install all needed dependencies.
#Pipfile.lock

# PEP 582; used by e.g. github.com/David-OConnor/pyflow
__pypackages__/

# Celery stuff
celerybeat-schedule
celerybeat.pid

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

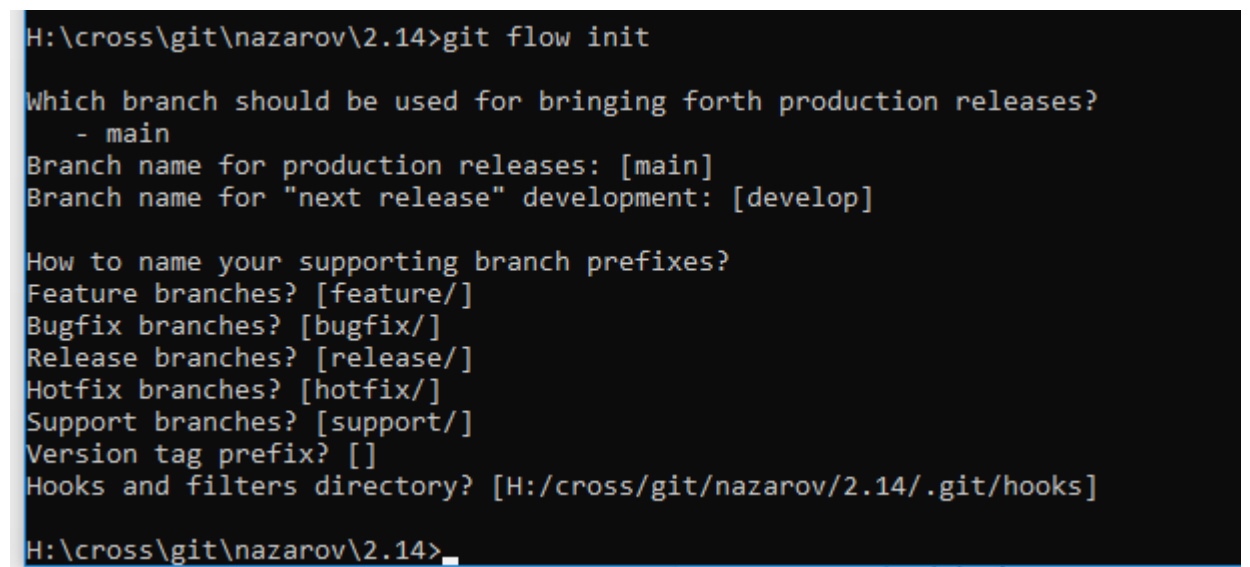
# mkdocs documentation
/site

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

# Pyre type checker
.pyre/

Стр 362, стлб 7    100%    Windows (CRLF)    UTF-8
```

Рисунок 1 – редактирование gitignore



```
H:\cross\git\nazarov\2.14>git flow init

Which branch should be used for bringing forth production releases?
  - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [H:/cross/git/nazarov/2.14/.git/hooks]

H:\cross\git\nazarov\2.14>
```

Рисунок 2 – организация репозитория в соответствии с git-flow

```
H:\cross\git\nazarov\2.14>pip --version
pip 21.3.1 from H:\Python\lib\site-packages\pip (python 3.10)
H:\cross\git\nazarov\2.14>
```

Рисунок 3 – проверка, установлен ли pip

```
H:\cross\git\nazarov\2.14>python -m venv env
H:\cross\git\nazarov\2.14>
```

Рисунок 4 – установка виртуального окружения

```
H:\cross\git\nazarov\2.14>.\env\scripts\activate.ps1
H:\cross\git\nazarov\2.14>.\env\scripts\activate
(env) H:\cross\git\nazarov\2.14>
```

Рисунок 5 – активация виртуального окружения

```
(env) H:\cross\git\nazarov\2.14>pip install black
Collecting black
  Downloading black-22.12.0-cp310-cp310-win_amd64.whl (1.2 MB)
    ----- 1.2/1.2 MB 910.4 kB/s eta 0:00:00
Collecting click>=8.0.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
    ----- 96.6/96.6 KB 1.8 MB/s eta 0:00:00
Collecting mypy_extensions>=0.4.3
  Downloading mypy_extensions-0.4.3-py2.py3-none-any.whl (4.5 kB)
Collecting tomli>=1.1.0
  Downloading tomli-2.0.1-py3-none-any.whl (12 kB)
Collecting pathspec>=0.9.0
  Downloading pathspec-0.10.3-py3-none-any.whl (29 kB)
Collecting platformdirs>=2
  Downloading platformdirs-2.6.0-py3-none-any.whl (14 kB)
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: mypy_extensions, tomli, platformdirs, pathspec, colorama, click, black
Successfully installed black-22.12.0 click-8.1.3 colorama-0.4.6 mypy_extensions-0.4.3 pathspec-0.10.3 platformdirs-2.6.0
tomli-2.0.1
WARNING: You are using pip version 22.0.4; however, version 22.3.1 is available.
You should consider upgrading via the 'H:\cross\git\nazarov\2.14\env\Scripts\python.exe -m pip install --upgrade pip' command.
(env) H:\cross\git\nazarov\2.14>deactivate
```

Рисунок 6 – установка пакета black и деактивация виртуального окружения

```
H:\cross\git\nazarov\2.14>python -m pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.17.1-py3-none-any.whl (8.8 MB)
    ----- 8.8 MB 3.2 MB/s
Collecting platformdirs<3,>=2.4
  Using cached platformdirs-2.6.0-py3-none-any.whl (14 kB)
Collecting distlib<1,>=0.3.6
  Downloading distlib-0.3.6-py2.py3-none-any.whl (468 kB)
    ----- 468 kB 6.4 MB/s
Collecting filelock<4,>=3.4.1
  Downloading filelock-3.8.2-py3-none-any.whl (10 kB)
Installing collected packages: platformdirs, filelock, distlib, virtualenv
Successfully installed distlib-0.3.6 filelock-3.8.2 platformdirs-2.6.0 virtualenv-20.17.1
WARNING: You are using pip version 21.3.1; however, version 22.3.1 is available.
You should consider upgrading via the 'H:\Python\Scripts\python.exe -m pip install --upgrade pip' command.
H:\cross\git\nazarov\2.14>
```

Рисунок 7 – установка virtualenv

```
H:\cross\git\nazarov\2.14>virtualenv -p python env
created virtual environment CPython3.10.5.final.0-64 in 4594ms
creator CPython3Windows(dest=H:\cross\git\nazarov\2.14\env, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\Y\AppData\Local\pypa\virtualenv)
added seed packages: black==22.12.0, click==8.1.3, colorama==0.4.6, mypy_extensions==0.4.3, pathspec==0.10.3, pip==22.3.1, platformdirs==2.6.0, setuptools==65.6.3, tomli==2.0.1, wheel==0.38.4
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
H:\cross\git\nazarov\2.14>_
```

Рисунок 8 – установка виртуального окружения

```
H:\cross\git\nazarov\2.14>.\env\scripts\activate

(env) H:\cross\git\nazarov\2.14>deactivate
H:\cross\git\nazarov\2.14>_
```

Рисунок 9 – активация и деактивация

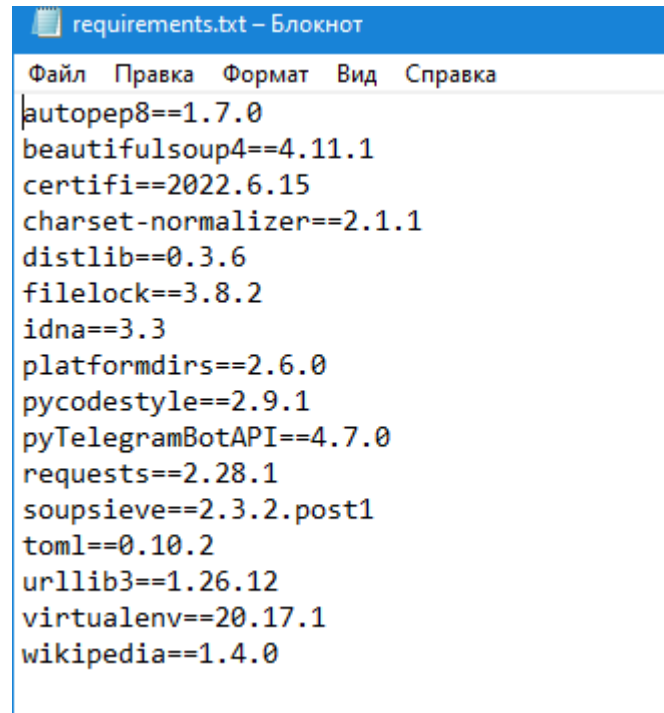
```
H:\cross\git\nazarov\2.14>pip freeze
autopep8==1.7.0
beautifulsoup4==4.11.1
certifi==2022.6.15
charset-normalizer==2.1.1
distlib==0.3.6
filelock==3.8.2
idna==3.3
platformdirs==2.6.0
pycodestyle==2.9.1
pyTelegramBotAPI==4.7.0
requests==2.28.1
soupsieve==2.3.2.post1
toml==0.10.2
urllib3==1.26.12
virtualenv==20.17.1
wikipedia==1.4.0

H:\cross\git\nazarov\2.14>
```

Рисунок 10 – список пакетных зависимостей

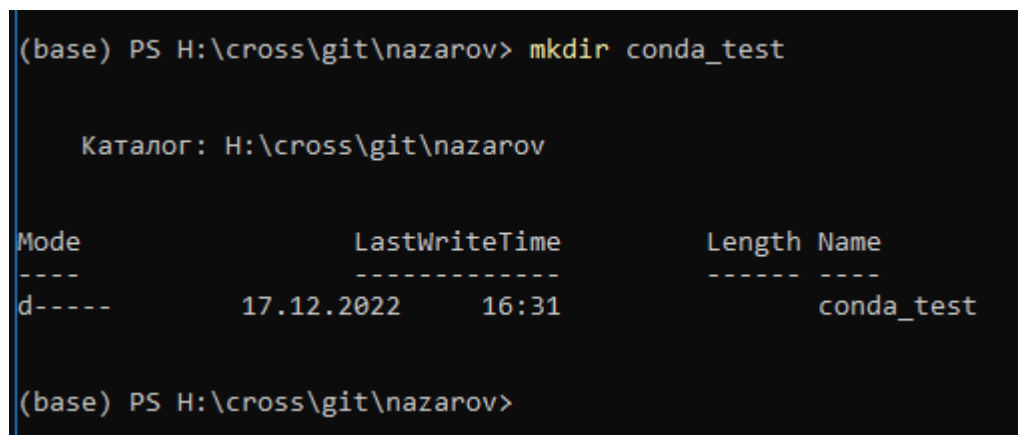
```
H:\cross\git\nazarov\2.14>pip freeze > requirements.txt
H:\cross\git\nazarov\2.14>_
```

Рисунок 11 – сохранение пакетных зависимостей в отдельный файл



```
requirements.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
autopep8==1.7.0
beautifulsoup4==4.11.1
certifi==2022.6.15
charset-normalizer==2.1.1
distlib==0.3.6
filelock==3.8.2
idna==3.3
platformdirs==2.6.0
pycodestyle==2.9.1
pyTelegramBotAPI==4.7.0
requests==2.28.1
soupsieve==2.3.2.post1
toml==0.10.2
urllib3==1.26.12
virtualenv==20.17.1
wikipedia==1.4.0
```

Рисунок 12 – файл requirements.txt



```
(base) PS H:\cross\git\nazarov> mkdir conda_test

Каталог: H:\cross\git\nazarov

Mode                LastWriteTime         Length Name
----                -
d-----          17.12.2022     16:31             conda_test

(base) PS H:\cross\git\nazarov>
```

Рисунок 13 – создание чистой директории и виртуального окружения

```
(base) PS H:\cross\git\nazarov> conda install django, pandas
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.

PackagesNotNotFoundError: The following packages are not available from current channels:

- django

Current channels:

- https://repo.anaconda.com/pkgs/main/win-64
- https://repo.anaconda.com/pkgs/main/noarch
- https://repo.anaconda.com/pkgs/r/win-64
- https://repo.anaconda.com/pkgs/r/noarch
- https://repo.anaconda.com/pkgs/msys2/win-64
- https://repo.anaconda.com/pkgs/msys2/noarch

To search for alternate channels that may provide the conda package you're
looking for, navigate to

    https://anaconda.org

and use the search bar at the top of the page.
```

Рисунок 15 – установка django и pandas

```
(base) PS H:\cross\git\nazarov> conda env export > environment.yml
(base) PS H:\cross\git\nazarov>
```

Рисунок 16 – создание файла конфигурации

```
Anaconda Powershell Prompt (anaconda3)
(base) PS H:\cross\git\nazarov> conda install pip, NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\Y\anaconda3

  added / updated specs:
    - numpy
    - pandas
    - pip
    - scipy

The following packages will be downloaded:

  package | build | size
  -----|-----|-----
  pip-22.3.1 | py39haa95532_0 | 2.7 MB
  -----|-----|-----
                                Total: 2.7 MB

The following packages will be UPDATED:

  pip                22.2.2-py39haa95532_0 --> 22.3.1-py39haa95532_0

Proceed ([y]/n)? n
Invalid choice: n
Proceed ([y]/n)? y

Downloading and Extracting Packages
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) PS H:\cross\git\nazarov>
```

Рисунок 17 – установка необходимых пакетов

```

tensorboard-data~ pkgs/main/win-64::tensorboard-data-server-0.6.1-py39haa95532_0
tensorboard-plugi~ pkgs/main/win-64::tensorboard-plugin-wit-1.8.1-py39haa95532_0
tensorflow         pkgs/main/win-64::tensorflow-2.9.1-mkl_py39hc9ebaa8_1
tensorflow-base    pkgs/main/win-64::tensorflow-base-2.9.1-mkl_py39h6a7f48e_1
tensorflow-estima~ pkgs/main/win-64::tensorflow-estimator-2.9.0-py39haa95532_0
termcolor          pkgs/main/win-64::termcolor-2.1.0-py39haa95532_0
yarl               pkgs/main/win-64::yarl-1.8.1-py39h2bbff1b_0

The following packages will be UPDATED:

ca-certificates      2022.07.19-haa95532_0 --> 2022.10.11-haa95532_0
certifi              2022.9.14-py39haa95532_0 --> 2022.9.24-py39haa95532_0
openssl              1.1.1q-h2bbff1b_0 --> 1.1.1s-h2bbff1b_0

The following packages will be DOWNGRADED:

anaconda              2022.10-py39_0 --> 2022.10-py310_0
scipy                 1.9.1-py39he11b74f_0 --> 1.7.3-py39h7a0a035_2

Proceed ([y]/n)? y

Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) c:\Users\Y>

```

Рисунок 18 – установка TensorFlow

```

collecting pyasn1-modules>=0.2.1
  Downloading pyasn1_modules-0.2.8-py2.py3-none-any.whl (155 kB)
    | 155 kB 2.2 MB/s
collecting requests-oauthlib>=0.7.0
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in h:\python\lib\site-packages (from requests<3,>=2.21.0->tensorboa
rd<2.12,>=2.11->tensorflow-intel==2.11.0->TensorFlow) (1.26.12)
Requirement already satisfied: idna<4,>=2.5 in h:\python\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.12,>
=2.11->tensorflow-intel==2.11.0->TensorFlow) (3.3)
Requirement already satisfied: charset-normalizer<3,>=2 in h:\python\lib\site-packages (from requests<3,>=2.21.0->tensor
board<2.12,>=2.11->tensorflow-intel==2.11.0->TensorFlow) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in h:\python\lib\site-packages (from requests<3,>=2.21.0->tensorboard<
2.12,>=2.11->tensorflow-intel==2.11.0->TensorFlow) (2022.6.15)
collecting MarkupSafe>=2.1.1
  Downloading MarkupSafe-2.1.1-cp310-cp310-win_amd64.whl (17 kB)
collecting pyasn1<0.5.0,>=0.4.6
  Downloading pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
    | 77 kB 1.9 MB/s
collecting oauthlib>=3.0.0
  Downloading oauthlib-3.2.2-py3-none-any.whl (151 kB)
    | 151 kB 1.6 MB/s
Installing collected packages: pyasn1, six, rsa, pyasn1-modules, oauthlib, cachetools, requests-oauthlib, MarkupSafe, go
ogle-auth, werkzeug, tensorboard-plugin-wit, tensorboard-data-server, protobuf, numpy, markdown, grpcio, google-auth-oau
thlib, absl-py, wrapt, typing-extensions, termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator, tensorboard, pa
ckaging, opt-einsum, libclang, keras, h5py, google-pasta, gast, flatbuffers, astunparse, tensorflow-intel, TensorFlow
Successfully installed MarkupSafe-2.1.1 TensorFlow-2.11.0 absl-py-1.3.0 astunparse-1.6.3 cachetools-5.2.0 flatbuffers-22
.12.6 gast-0.4.0 google-auth-2.15.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.51.1 h5py-3.7.0 keras-2.11.0
libclang-14.0.6 markdown-3.4.1 numpy-1.23.5 oauthlib-3.2.2 opt-einsum-3.3.0 packaging-22.0 protobuf-3.19.6 pyasn1-0.4.8
pyasn1-modules-0.2.8 requests-oauthlib-1.3.1 rsa-4.9 six-1.16.0 tensorboard-2.11.0 tensorboard-data-server-0.6.1 tensorb
oard-plugin-wit-1.8.1 tensorflow-estimator-2.11.0 tensorflow-intel-2.11.0 tensorflow-io-gcs-filesystem-0.28.0 termcolor-
2.1.1 typing-extensions-4.4.0 werkzeug-2.2.2 wrapt-1.14.1
WARNING: You are using pip version 21.3.1; however, version 22.3.1 is available.
You should consider upgrading via the 'H:\Python\Scripts\python.exe -m pip install --upgrade pip' command.

H:\cross\git\nazarov\2.14>

```

Рисунок 19 – установка TensorFlow через pip

## Ответы на 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.

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



При разворачивании современной версии Python, pip устанавливается автоматически. Но если, по какой-то причине, pip не установлен на вашем ПК, то сделать это можно вручную. Чтобы установить pip, нужно скачать скрипт get-pip.py и выполнить его.

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

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

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

С помощью команды `$ pip install ProjectName`.

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

С помощью команды `$ pip install ProjectName==3.2`, где вместо 3.2 необходимо указать нужную версию пакета.

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

С помощью команды `$ pip install e git+https://gitrepo.com/ ProjectName.git`

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

С помощью команды `$ pip install ./dist/ProjectName.tar.gz`

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

С помощью команды `$ pip uninstall ProjectName` можно удалить установленный пакет.

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

С помощью команды `$ pip install --upgrade ProjectName` можно обновить необходимый пакет.

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

Командой `$ pip list` можно отобразить список установленных пакетов.

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

Существует несколько причин появления виртуальных окружений в языке Python - проблема обратной совместимости и проблема коллективной разработки. Проблема обратной совместимости - некоторые операционные системы, например, Linux и MacOS используют содержащиеся в них предустановленные интерпретаторы Python. Обновив или изменив самостоятельно версию какого-либо установленного глобально пакета, мы можем непреднамеренно сломать работу утилит и приложений из дистрибутива операционной системы.

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?**

Основные этапы:

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

Активируем ранее созданное виртуальное окружение для работы.

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

Деактивируем после окончания работы виртуальное окружение.

Удаляем папку с виртуальным окружением, если оно нам больше не нужно.

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

С его помощью можно создать виртуальную среду, в которую можно устанавливать пакеты независимо от основной среды или других виртуальных окружений. Основные действия с виртуальными окружениями с помощью venv: создание виртуального окружения, его активация и деактивация.

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

Для начала пакет нужно установить. Установку можно выполнить командой: `python3 -m pip install virtualenv` Virtualenv позволяет создать абсолютно изолированное виртуальное окружение для каждой из программ.

Окружением является обычная директория, которая содержит копию всего необходимого для запуска определенной программы, включая копию самого интерпретатора, полной стандартной библиотеки, `pip`, и, что самое главное, копии всех необходимых пакетов.

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

Для формирования и развертывания пакетных зависимостей используется утилита `pip`.

Основные возможности `pipenv`:

- Создание и управление виртуальным окружением
- Синхронизация пакетов в `Pipfile` при установке и удалении пакетов
- Автоматическая подгрузка переменных окружения из `.env` файла

После установки `pipenv` начинается работа с окружением. Его можно создать в любой папке. Достаточно установить любой пакет внутри папки.

Используем requests, он автоматически установит окружение и создаст Pipfile и Pipfile.lock.

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

Установить пакеты можно с помощью команды: `pip install -r requirements.txt`. Также можно использовать команду `pip freeze > requirements.txt`, которая создаст requirements.txt наполнив его названиями и версиями тех пакетов что используются вами в текущем окружении. Это удобно если вы разработали проект и в текущем окружении все работает, но вы хотите перенести проект в иное окружением (например, заказчику или на сервер). С помощью закрепления зависимостей мы можем быть уверены, что пакеты, установленные в нашей производственной среде, будут точно соответствовать пакетам в нашей среде разработки, чтобы ваш проект неожиданно не ломался.

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

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

## **18. В какие дистрибутивы Python входит пакетный менеджер conda?**

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?**

С помощью команды: `conda create -n %PROJ_NAME% python=3.7`

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

Чтобы установить пакеты, необходимо воспользоваться командой: —

conda install A для активации: conda activate %PROJ\_NAME%

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

Для деактивации использовать команду: conda deactivate, а для удаления: conda remove -n \$PROJ\_NAME.

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

Создание файла: conda env export > environment.yml

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

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

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.**

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

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

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