

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций «Установка
пакетов в Python. Виртуальные окружения»**

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

Выполнил студент группы ПИЖ-б-о-21-1

Кучеренко С. Ю. « » 2023г.

Подпись студента_____

Работа защищена « »_____2023г.

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:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.

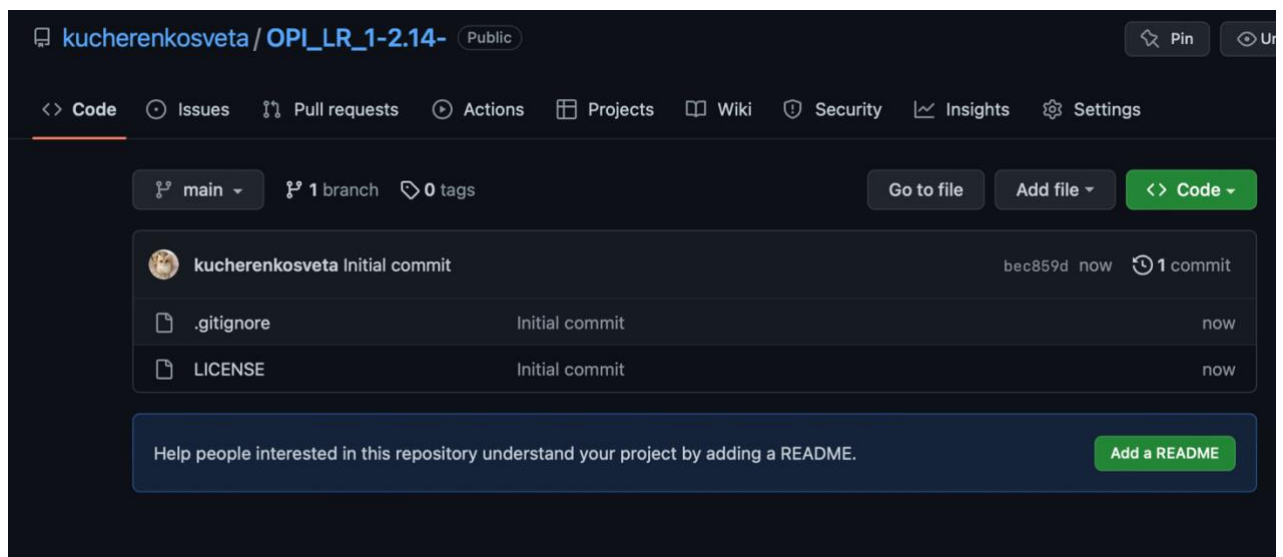


Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
Last login: Mon Feb 13 22:05:38 on ttys000
(base) svetik@MacBook-Air-Svetik ~ % cd ~/Desktop/ОПИ
(base) svetik@MacBook-Air-Svetik ОПИ % git clone https://github.com/kucherenkosveta/OPI_LR_1-2.14-.git
Cloning into 'OPI_LR_1-2.14-'...
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.
(base) svetik@MacBook-Air-Svetik ОПИ %
```

Рисунок 2 – Клонирование репозитория

4. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
(base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- % 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? [/Users/svetik/Desktop/ОПИ/OPI_LR_1-2.14-/.git/hooks]
(base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- %
```

Рисунок 3 – Организация репозитория в соответствии с моделью git-flow

5. Создайте виртуальное окружение Anaconda с именем репозитория.

```
(base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- % conda create -n OPI_LR_1-2.14-
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: /Users/svetik/opt/anaconda3/envs/OPI_LR_1-2.14-

Proceed ([y]/n)?

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate OPI_LR_1-2.14-
#
# To deactivate an active environment, use
#
#     $ conda deactivate

Retrieving notices: ...working... done
(base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- %
```

Рисунок 4 – Создание виртуального окружения

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

```
(base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- % conda install -n OPI_LR_1-2.14- 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: /Users/svetik/opt/anaconda3/envs/OPI_LR_1-2.14-

added / updated specs:
- pip

The following packages will be downloaded:
```

package	build	
ca-certificates-2023.01.10	hca03da5_0	121 KB
certifi-2022.12.7	py310hca03da5_0	151 KB
libffi-3.4.2	hca03da5_6	115 KB
ncurses-6.4	h313beb8_0	884 KB
openssl-1.1.1t	h1a28f6b_0	2.6 MB
pip-22.3.1	py310hca03da5_0	2.8 MB
python-3.10.9	hc0d8a6c_0	12.0 MB
readline-8.2	h1a28f6b_0	353 KB
setuptools-65.6.3	py310hca03da5_0	1.2 MB
sqlite-3.40.1	h7a7dc30_0	1.1 MB
tzdata-2022g	h04d1e81_0	114 KB
xz-5.2.10	h80987f9_1	263 KB
zlib-1.2.13	h5a0b063_0	82 KB
Total:		21.7 MB

```
The following NEW packages will be INSTALLED:

bzip2                pkgs/main/osx-arm64::bzip2-1.0.8-h620ffc9_4 None
ca-certificates      pkgs/main/osx-arm64::ca-certificates-2023.01.10-hca03da5_0 None
certifi              pkgs/main/osx-arm64::certifi-2022.12.7-py310hca03da5_0 None
libffi               pkgs/main/osx-arm64::libffi-3.4.2-hca03da5_6 None
ncurses              pkgs/main/osx-arm64::ncurses-6.4-h313beb8_0 None
```

Рисунок 5 – Установка в виртуальное окружение пакета pip

```

(base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- % conda install -n OPI_LR_1-2.14- 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: /Users/svetik/opt/anaconda3/envs/OPI_LR_1-2.14-

added / updated specs:
- numpy

The following packages will be downloaded:



| package             | build              |        |
|---------------------|--------------------|--------|
| libgfortran-5.0.0   | 11_3_0_hca03da5_28 | 142 KB |
| libgfortran5-11.3.0 | h009349e_28        | 1.0 MB |
| numpy-1.23.5        | py310hb93e574_0    | 11 KB  |
| numpy-base-1.23.5   | py310haf87e8b_0    | 5.7 MB |
| Total:              |                    | 6.9 MB |



The following NEW packages will be INSTALLED:

blas                pkgs/main/osx-arm64::blas-1.0-openblas None
libcxx              pkgs/main/osx-arm64::libcxx-14.0.6-h848a8c0_0 None
libgfortran         pkgs/main/osx-arm64::libgfortran-5.0.0-11_3_0_hca03da5_28 None
libgfortran5        pkgs/main/osx-arm64::libgfortran5-11.3.0-h009349e_28 None
libopenblas         pkgs/main/osx-arm64::libopenblas-0.3.21-h269037a_0 None
llvm-openmp         pkgs/main/osx-arm64::llvm-openmp-14.0.6-hc6e5704_0 None
numpy               pkgs/main/osx-arm64::numpy-1.23.5-py310hb93e574_0 None
numpy-base          pkgs/main/osx-arm64::numpy-base-1.23.5-py310haf87e8b_0 None

Proceed ([y]/n)?

Downloading and Extracting Packages

```

Рисунок 6 – Установка в виртуальное окружение пакета NumPy


```

(base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- % conda install -n OPI_LR_1-2.14- 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: /Users/svetik/opt/anaconda3/envs/OPI_LR_1-2.14-

added / updated specs:
  - pandas

The following packages will be downloaded:



| package          | build           |         |
|------------------|-----------------|---------|
| bottleneck-1.3.5 | py310h96f19d2_0 | 104 KB  |
| numexpr-2.8.4    | py310hecc3335_0 | 121 KB  |
| packaging-22.0   | py310hca03da5_0 | 70 KB   |
| pandas-1.5.2     | py310h46d7db6_0 | 11.3 MB |
| pytz-2022.7      | py310hca03da5_0 | 211 KB  |
| Total:           |                 | 11.8 MB |



The following NEW packages will be INSTALLED:

bottleneck      pkgs/main/osx-arm64::bottleneck-1.3.5-py310h96f19d2_0 None
numexpr         pkgs/main/osx-arm64::numexpr-2.8.4-py310hecc3335_0 None
packaging       pkgs/main/osx-arm64::packaging-22.0-py310hca03da5_0 None
pandas          pkgs/main/osx-arm64::pandas-1.5.2-py310h46d7db6_0 None
python-dateutil pkgs/main/noarch::python-dateutil-2.8.2-pyhd3eb1b0_0 None
pytz            pkgs/main/osx-arm64::pytz-2022.7-py310hca03da5_0 None
six             pkgs/main/noarch::six-1.16.0-pyhd3eb1b0_1 None

Proceed ([y]/n)?

Downloading and Extracting Packages

```

Рисунок 7 – Установка в виртуальное окружение пакета Pandas

```

[(base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- % conda install -n OPI_LR_1-2.14- scipy
y
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: /Users/svetik/opt/anaconda3/envs/OPI_LR_1-2.14-

added / updated specs:
- scipy

The following packages will be downloaded:



| package             | build              |         |
|---------------------|--------------------|---------|
| brotlipy-0.7.0      | py310h1a28f6b_1002 | 312 KB  |
| cffi-1.15.1         | py310h80987f9_3    | 232 KB  |
| cryptography-38.0.4 | py310h834c97f_0    | 1.1 MB  |
| idna-3.4            | py310hca03da5_0    | 98 KB   |
| pooch-1.4.0         | pyhd3eb1b0_0       | 41 KB   |
| pysocks-1.7.1       | py310hca03da5_0    | 28 KB   |
| requests-2.28.1     | py310hca03da5_0    | 94 KB   |
| scipy-1.10.0        | py310h20cbe94_0    | 20.3 MB |
| urllib3-1.26.14     | py310hca03da5_0    | 197 KB  |
| Total:              |                    | 22.4 MB |



The following NEW packages will be INSTALLED:

appdirs pkgs/main/noarch::appdirs-1.4.4-pyhd3eb1b0_0 None
brotlipy pkgs/main/osx-arm64::brotlipy-0.7.0-py310h1a28f6b_1002 None
cffi pkgs/main/osx-arm64::cffi-1.15.1-py310h80987f9_3 None
charset-normalizer pkgs/main/noarch::charset-normalizer-2.0.4-pyhd3eb1b0_0 None
cryptography pkgs/main/osx-arm64::cryptography-38.0.4-py310h834c97f_0 None
fftw pkgs/main/osx-arm64::fftw-3.3.9-h1a28f6b_1 None
idna pkgs/main/osx-arm64::idna-3.4-py310hca03da5_0 None

```

Рисунок 8 – Установка в виртуальное окружение пакета SciPy

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

```
(base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- % conda install -n OPI_LR_1-2.14- 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: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: -
Found conflicts! Looking for incompatible packages.
This can take several minutes. Press CTRL-C to abort.
failed

UnsatisfiableError: The following specifications were found to be incompatible with each other:

Output in format: Requested package -> Available versions
```

Рисунок 9 – Попытка установки пакета TensorFlow

Пакет TensorFlow совместим с Python 3.8, для решения возможной проблемы необходимо понизить версию python с помощью команды:

```
conda install python=3.7
```

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

```
[(tensorflow-metal-test) (base) svetik@MacBook-Air-Svetik tensorflow-metal-test % pip install tensorflow-metal
Collecting tensorflow-metal
  Downloading tensorflow-metal-0.7.1-cp39-cp39-macosx_12_0_arm64.whl (1.4 MB)
    _____ 1.4/1.4 MB 1.7 MB/s eta 0:00:00
Requirement already satisfied: wheel~=0.35 in ./lib/python3.9/site-packages (from tensorflow-metal) (0.38.4)
Requirement already satisfied: six>=1.15.0 in ./lib/python3.9/site-packages (from tensorflow-metal) (1.16.0)
Installing collected packages: tensorflow-metal
Successfully installed tensorflow-metal-0.7.1
(tensorflow-metal-test) (base) svetik@MacBook-Air-Svetik tensorflow-metal-test %
```

Рисунок 8 – Установка в виртуальное окружение пакета SciPy

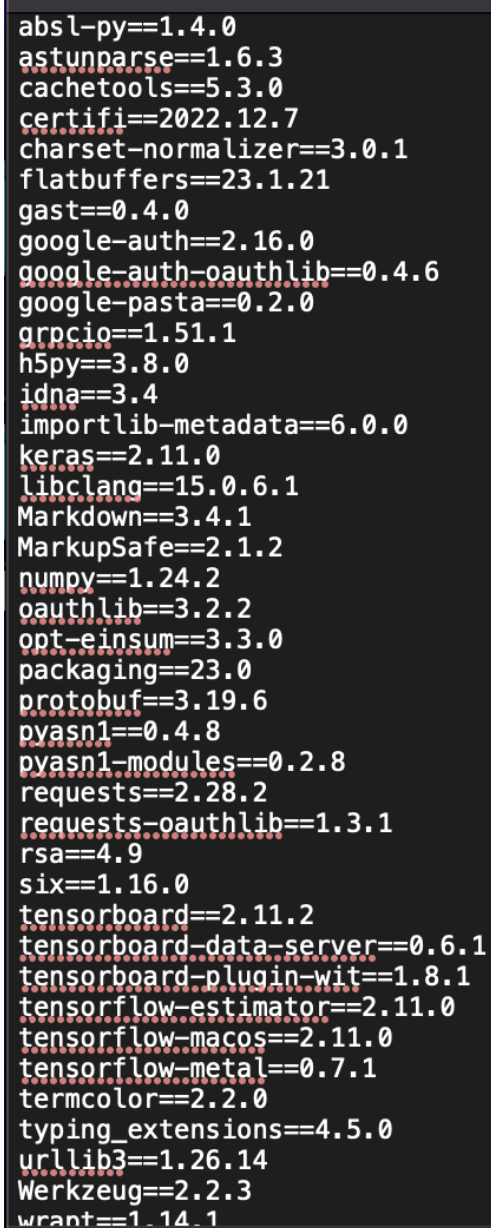
9. Сформируйте файлы requirements.txt и environment.yml. Проанализируйте содержимое этих файлов.

Файл requirements.txt хранит в себе список всех установленных в интерпретатор сторонних пакетов, чтобы потом можно было оперативно и просто установить такой же набор. Команда pip freeze выводит интересующий

нас список, но при желании это можно сделать вручную.

```
(tensorflow-metal-test) (base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- % pip freeze > requirements.txt  
(tensorflow-metal-test) (base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- %
```

Рисунок 9 – Создание файла requirements.txt



```
absl-py==1.4.0  
astunparse==1.6.3  
cachetools==5.3.0  
certifi==2022.12.7  
charset-normalizer==3.0.1  
flatbuffers==23.1.21  
gast==0.4.0  
google-auth==2.16.0  
google-auth-oauthlib==0.4.6  
google-pasta==0.2.0  
grpcio==1.51.1  
h5py==3.8.0  
idna==3.4  
importlib-metadata==6.0.0  
keras==2.11.0  
libclang==15.0.6.1  
Markdown==3.4.1  
MarkupSafe==2.1.2  
numpy==1.24.2  
oauthlib==3.2.2  
opt-einsum==3.3.0  
packaging==23.0  
protobuf==3.19.6  
pyasn1==0.4.8  
pyasn1-modules==0.2.8  
requests==2.28.2  
requests-oauthlib==1.3.1  
rsa==4.9  
six==1.16.0  
tensorboard==2.11.2  
tensorboard-data-server==0.6.1  
tensorboard-plugin-wit==1.8.1  
tensorflow-estimator==2.11.0  
tensorflow-macos==2.11.0  
tensorflow-metal==0.7.1  
termcolor==2.2.0  
typing_extensions==4.5.0  
urllib3==1.26.14  
Werkzeug==2.2.3  
wrapt==1.14.1
```


Рисунок 10 – Содержание requirements.txt

Совместное использование среды проекта между платформами и операционными системами также можно выполнить с помощью опции export для создания файла environment.yml. Разница между списком спецификаций и файлом environment.yml заключается в том, что последний не зависит от

операционной системы и форматируется с использованием YAML, он содержит имя, зависимости и установочные каналы.

```
[(tensorflow-metal-test) (base) svetik@MacBook-Air-Svetik OPI_LR_1-2.14- % conda env export > environment.yml
```

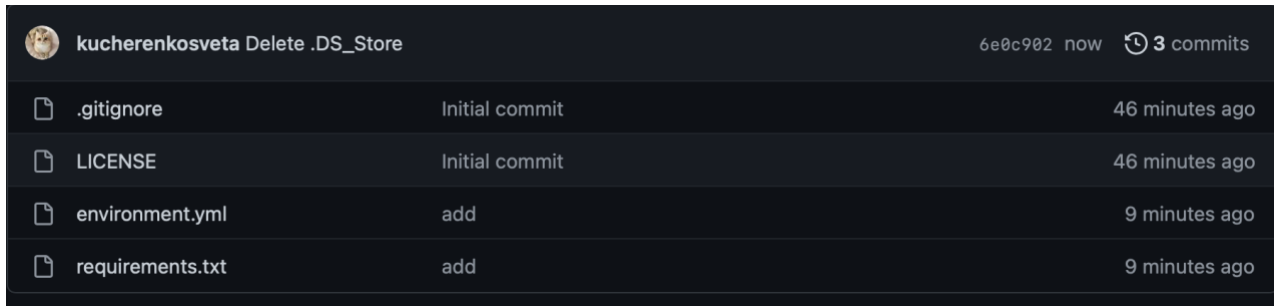
Рисунок 11 – Создание файла environment.yml



```
name: base
channels:
  - defaults
dependencies:
  - ipyw_jlab_nb_ext_conf=0.1.0=py39hca03da5_1
  - alabaster=0.7.12=pyhd3eb1b0_0
  - anaconda=2022.10=py39_0
  - anaconda-client=1.11.0=py39hca03da5_0
  - anaconda-navigator=2.3.1=py39hca03da5_0
  - anaconda-project=0.11.1=py39hca03da5_0
  - anvio=3.5.0=py39hca03da5_0
  - appdirs=1.4.4=pyhd3eb1b0_0
  - appnope=0.1.2=py39hca03da5_1001
  - appscript=1.1.2=py39h1a28f6b_0
  - argon2-cffi=21.3.0=pyhd3eb1b0_0
  - argon2-cffi-bindings=21.2.0=py39h1a28f6b_0
  - astroid=2.11.7=py39hca03da5_0
  - astropy=5.1=py39heec5a64_0
  - asttokens=2.0.5=pyhd3eb1b0_0
  - attrs=21.4.0=pyhd3eb1b0_0
  - automat=20.2.0=py_0
  - babel=2.9.1=pyhd3eb1b0_0
  - backcall=0.2.0=pyhd3eb1b0_0
  - backports=1.1=pyhd3eb1b0_0
  - backports.functiontools_lru_cache=1.6.4=pyhd3eb1b0_0
  - backports.tempfile=1.0=pyhd3eb1b0_1
  - backports.weakref=1.0.post1=py_1
  - bcrypt=3.2.0=py39h1a28f6b_1
  - beautifulsoup4=4.11.1=py39hca03da5_0
  - bitarray=2.5.1=py39h1a28f6b_0
  - bkcharts=0.2=py39hca03da5_1
  - blas=1.0=openblas
  - bleach=4.1.0=pyhd3eb1b0_0
  - blosc=1.21.0=h98b2900_1
  - bokeh=2.4.3=py39hca03da5_0
  - boto3=1.24.28=py39hca03da5_0
  - botocore=1.27.28=py39hca03da5_0
  - bottleneck=1.3.5=py39heec5a64_0
  - brotli=1.0.9=h1a28f6b_7
  - brotli-bin=1.0.9=h1a28f6b_7
  - brotliipy=0.7.0=py39h1a28f6b_1002
  - brunslis=0.1=hc377ac9_1
  - bzip2=1.0.8=h620ffc9_4
  - c-ares=1.18.1=h1a28f6b_0
  - ca-certificates=2022.07.19=hca03da5_0
  - cctools=949.0.1=hc179dcd_25
  - cctools_osx-arm64=949.0.1=h332cad3_25
  - certifi=2022.9.24=py39hca03da5_0
  - cffi=1.15.1=py39h22df2f2_0
```

Рисунок 11 – Содержание environment.yml

10. Зафиксируйте сделанные изменения в репозитории.








 kucherenkosveta	Delete .DS_Store	6e0c902 now	🕒 3 commits
	.gitignore	Initial commit	46 minutes ago
	LICENSE	Initial commit	46 minutes ago
	environment.yml	add	9 minutes ago
	requirements.txt	add	9 minutes ago

Рисунок 12 – Фиксирование изменений в репозитории

Ответы на вопросы:

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

Существует Python Package Index (PyPI) – это репозиторий, открытый для всех разработчиков, в нём можно найти пакеты для решения практических задач.

2. Как осуществить установку менеджера пакетов pip?

Pip – это консольная утилита (без графического интерфейса). После того, как вы её скачаете и установите, она пропишется в PATH и будет доступна для использования.

Чтобы установить утилиту pip, нужно также скачать скрипт get-pip.py.

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?

Используя команду «pip install ProjectName»

5. Как установить заданную версию пакета с помощью pip?

Используя команду «`pip install ProjectName==3.2`» (где 3.2 – нужная версия пакета)

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

Используя команду «`pip install -e git+https://gitrepo.com/ ProjectName.git`»

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

Используя команду «`pip install ./dist/ProjectName.tar.gz`»

8. Как удалить установленный пакет с помощью pip?

Используя команду «`pip uninstall ProjectName`»

9. Как обновить установленный пакет с помощью pip?

Используя команду «`pip install --upgrade ProjectName`»

10. Как отобразить список установленных пакетов с помощью pip?

Используя команду «`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 трудностей. Основная задача виртуальных окружений – это решение проблем с совместимостью и коллективной разработкой. Так можно не бояться нарушить работу ОС, таких как Linux и MacOS, обновив какой-либо пакет, а также свободно устанавливая различные пакеты, не боясь, что другие члены

команды столкнутся с трудностями, связанными с отсутствием тех или иных пакетов при попытке запустить проект.

12. Каковы основные этапы работы с виртуальными окружениями?

1. Создание нового виртуальное окружение в отдельной папке для выбранной версии интерпретатора Python.
2. Активизация окружения.
3. Осуществляется работа.
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: «python 3 -m venv <путь к папке виртуального окружения>»

Чтобы активировать виртуальное окружение под Windows нужно дать команду: «> env\\Scripts\\activate».

После активации приглашение консоли изменится. В его начале в круглых скобках будет отображаться имя папки с виртуальным окружением.

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.

Чтобы переключиться с одного окружения на другое нам нужно выполнить команду деактивации и команду активации другого виртуального окружения.

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

Вначале командой «`python3 -m pip install virtualenv Virtualenv`» необходимо установить пакет, который позволяет создать абсолютно изолированное виртуальное окружение для каждой из программ.

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

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

Грубо говоря, `pipenv` можно рассматривать как симбиоз утилит `pip` и `venv` (или `virtualenv`), которые работают вместе, пряча многие неудобные детали от конечного пользователя.

Основные возможности `pipenv`: создание и управление виртуальным окружением; синхронизация пакетов в `Pipfile` при установке и удалении пакетов; автоматическая подгрузка переменных окружения из `.env` файла.

Установка: `pip install --user pipenv`

`Pipenv` использует свой собственный формат файла для описания зависимостей проекта — `Pipfile`. Этот файл имеет формат `TOML`. В принципе его можно редактировать руками, но `pipenv` достаточно неплохо и сам умеет

обновлять этот файл, когда вы просто работаете с утилитой через командную строку. Структуру этого файла рассмотрим чуть позже.

В паре с Pipfile идёт Pipfile.lock. Он имеет формат JSON и не предназначен для редактирования руками. Этот файл хранит контрольные суммы пакетов, которые вы устанавливаете в проект, что даёт гарантию, что развёрнутые на разных машинах окружения будут идентичны друг другу. pipenv автоматически обновляет контрольные суммы в этом файле, когда вы устанавливаете или обновляете зависимости. При развёртывании окружения pipenv сверит сохранённые контрольные суммы с фактически получившимися, и в случае чего уведомит вас, что развёртывание не удалось.

Это очень важный плюс в копилку pipenv по сравнению с pip.

Оба этих файла можно и нужно сохранять в системе контроля версий (git).

Вообще, идею использовать два файла для описания зависимостей нельзя назвать новой. Здесь явно прослеживается параллель между Gemfile и Gemfile.lock из мира Ruby и package.json и package-lock.json из мира JavaScript. Все эти файлы имеют схожее назначение.

Создание проекта: pipenv --three

После выполнения команды, pipenv создал файл Pipfile и виртуальное окружение где-то в заранее определенной директории (по умолчанию вне директории проекта).

Содержание файла:

```
[[source]] name
= "pypi"
url = "https://pypi.org/simple" verify_ssl
= true
```

```
[dev-packages]
```

```
[packages]
```

```
[requires]
```

```
python_version = "3.8"
```

Установка зависимости: `pipenv install requests`

Удаление зависимостей: `pipenv uninstall`

Обновление зависимостей: `pipenv update`

Удаление виртуального окружения: `pipenv --rm`

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

Какой он имеет формат?

Файл requirements.txt хранит в себе список всех установленных в интерпретатор сторонних пакетов, чтобы потом можно было оперативно и просто установить такой же набор. Его можно написать вручную, а можно автоматически с помощью: `pip freeze > requirements.txt`.

Установить пакеты можно с помощью команды: `pip install -r requirements.txt`

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

Pip – это менеджер пакетов, который облегчает установку, обновление и удаление пакетов python. Он также работает с виртуальными средами python.

Conda – это менеджер пакетов для любого программного обеспечения (установка, обновление и удаление). Он также работает с виртуальными средами system.

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? Anaconda, Miniconda, PyCharm.

19. Как создать виртуальное окружение conda?

Используя команду «conda create -n <имя виртуального окружения>»

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

Для активации использовать команду: «conda activate <имя виртуального окружения>»

Для установки: conda install <имя виртуального окружения> <имя пакета>»

21. Как деактивировать и удалить виртуальное окружение conda?

Для деактивации использовать команду: conda deactivate <имя виртуального окружения>, а для удаления: conda remove -n <имя виртуального окружения>.

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

Файл environment.yml позволит воссоздать окружение. Он не зависит от операционной системы и форматируется с использованием YAML, содержит имя, зависимости и установочные каналы.

Создание файла: conda env export > environment.yml

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

Необходимо использовать команду: «conda env create -f environment.yml»

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

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.

Для первого способа ход работы, следующий: запускаем PyCharm и в окне приветствия выбираем Create New Project. В мастере создания проекта, указываем в поле Location путь расположения создаваемого проекта. Имя конечной директории также является именем проекта. Далее разворачиваем параметры окружения, щелкая по Project Interpreter. И выбираем New environment using Virtualenv. Путь расположения окружения генерируется автоматически. И нажимаем на Create. Теперь установим библиотеки, которые будем использовать в программе. С помощью главного меню переходим в настройки File → Settings. Где переходим в Project: project_name → Project Interpreter. Выходим из настроек. Для запуска программы, необходимо создать профиль с конфигурацией. Для этого в верхнем правом углу нажимаем на кнопку Add Configuration. Откроется окно Run/Debug Configurations, где нажимаем на кнопку с плюсом (Add New Configuration) в правом верхнем углу и выбираем Python. Далее указываем в поле Name имя конфигурации и в поле Script path расположение Python файла с кодом программы. В завершение нажимаем на Apply, затем на ОК. Для второго способа необходимо сделать

следующее: на экране приветствия в нижнем правом углу через Configure → Settings переходим в настройки. Затем переходим в раздел Project Interpreter.

В верхнем правом углу есть кнопка с шестерёнкой, нажимаем на неё и выбираем Add, создавая новое окружение. И указываем расположение для нового окружения. Нажимаем на ОК. Далее в созданном окружении устанавливаем нужные пакеты. И выходим из настроек. В окне приветствия выбираем Create New Project. В мастере создания проекта, указываем имя расположения проекта в поле Location. Разворачиваем параметры окружения, щелкая по Project Interpreter, где выбираем Existing interpreter и указываем нужное нам окружение. Далее создаем конфигурацию запуска программы, также как создавали для раннее. После чего можно выполнить программу.

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

Чтобы другие пользователи могли ознакомиться с набором пакетов, используемом в проекте и воссоздать виртуальное окружение на своем устройстве.