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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО РАБОТЕ №2.14
дисциплины «Программирование на языке Python»

Выполнил:
Хачатрян Владимир Владимирович
2 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные технологии и системы связи»,
направленность (профиль) «Инфокоммуникационные системы и сети», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Установка пакетов в Python. Виртуальные окружения.

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

Выполнения работы:

1. Изучил теоретический материал работы, создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами.

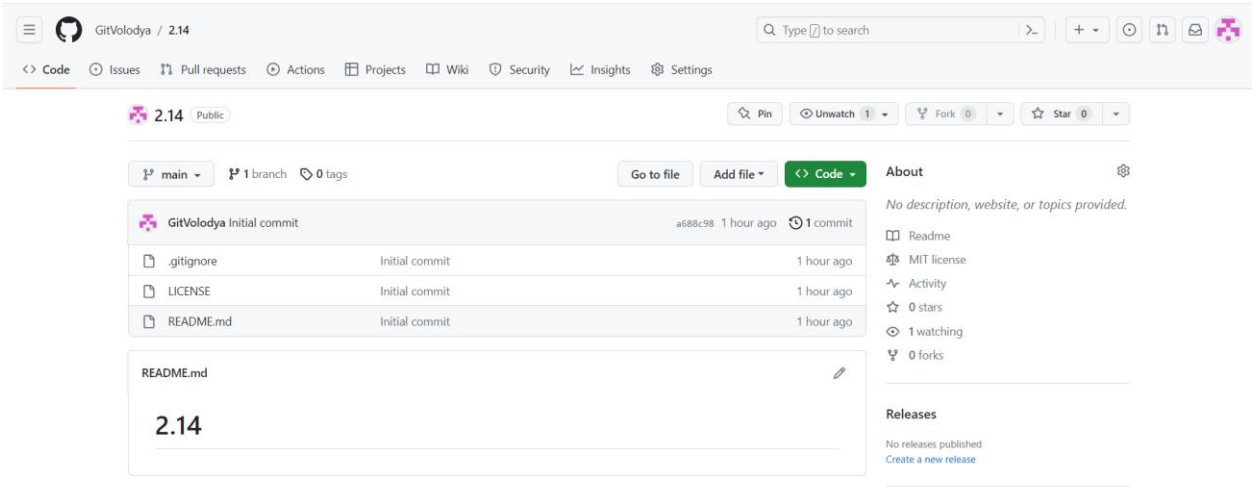

Рисунок 1. Репозиторий

2. Клонировал репозиторий на свой компьютер.

```
C:\Users\vovax>git clone https://github.com/GitVolodya/2.14.git
Cloning into '2.14'...
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.

C:\Users\vovax>cd C:\Users\vovax\2.14
```

Рисунок 2. Клонирование репозитория

Организовал репозиторий в соответствие с моделью ветвления git-flow. Появилась ветка develop.

```
C:\Users\vovax\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? [C:/Users/vovax/2.14/.git/hooks]

C:\Users\vovax\2.14>_
```

Рисунок 3. Модель ветвления git-flow

3. Создал виртуальное окружение Anaconda с именем репозитория.

```
(base) PS C:\Users\vovax\2.14> conda create -n 2.14 python=3.11
Collecting package metadata (current_repodata.json): \ DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
| DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
| DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/noarch/current_repodata.json HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/win-64/current_repodata.json HTTP/1.1" 200 None
/ DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/noarch/current_repodata.json HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/win-64/current_repodata.json HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/noarch/current_repodata.json HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/current_repodata.json HTTP/1.1" 200 None
done
Solving environment: done
```

Рисунок 4. Виртуальное окружение Anaconda

4. Установил необходимые пакеты командой conda install

```
(2.14) PS C:\Users\vovax\2.14> conda install pip, numpy, scipy
Collecting package metadata (current_repodata.json): | DEBUG:urllib3.connectionpo
ol:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com
:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com
:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com
:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com
:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com
:443
| DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-
64/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/noarc
h/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/noarch/cu
rrent_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/win-64/cu
rrent_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/win-6
4/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/noarch
/current_repodata.json HTTP/1.1" 304 0
done
Solving environment: done
```

Рисунок 5. Установка пакетов

5. При установке пакета TensorFlow было обнаружено, что спецификации несовместимы с установленной версией python, а при установке с помощью менеджера пакета pip, пакет успешно установился.

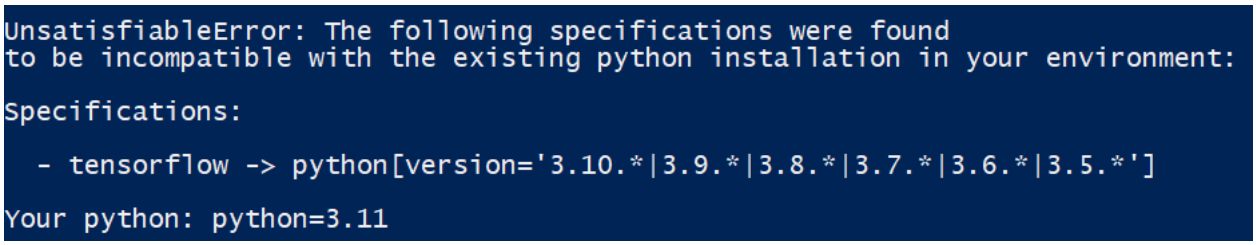

Рисунок 6. Установка TensorFlow

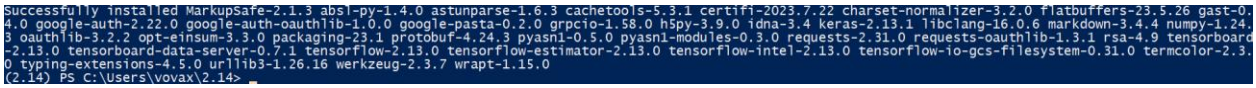

Рисунок 7. Установка TensorFlow с помощью pip

6. Сформировал два файла requirements.txt и environment.yml.

Рисунок 8. Файлы requirements.txt и environment.yml

Requirements.txt и environment.yml указывают, какие библиотеки и их версии необходимы для правильной работы проекта.

Если сравнивать эти файлы между собой можно заметить, что в requirements.txt отображены все пакеты и их версии, когда в environment.yml можно увидеть ещё и название окружающей среды, зависимости и каналы.

```
absl-py==1.4.0
asgiref @ file:///C:/ci_311/asgiref_1676448296195/work
astunparse==1.6.3
Bottleneck @ file:///C:/ci_311/bottleneck_1676500016583/work
cachetools==5.3.1
certifi==2023.7.22
charset-normalizer==3.2.0
Django @ file:///C:/ci_311/django_1676469589007/work
flatbuffers==23.5.26
gast==0.4.0
google-auth==2.22.0
google-auth-oauthlib==1.0.0
google-pasta==0.2.0
grpcio==1.58.0
h5py==3.9.0
idna==3.4
keras==2.13.1
libclang==16.0.6
Markdown==3.4.4
MarkupSafe==2.1.3
mkl-fft==1.3.6
mkl-random @ file:///C:/Users/dev-admin/mkl/mkl_random_1682977971003/work
mkl-service==2.4.0
numexpr @ file:///C:/b/abs_afm0oewmmt/croot/numexpr_1683221839116/work
numpy==1.24.3
oauthlib==3.2.2
opt-einsum==3.3.0
packaging==23.1
pandas @ file:///C:/miniconda3/conda-bld/pandas_1692298018988/work
protobuf==4.24.3
pyasn1==0.5.0
pyasn1-modules==0.3.0
python-dateutil @ file:///tmp/build/80754af9/python-dateutil_1626374649649/work
pytz @ file:///C:/ci_311/pytz_1676427070848/work
requests==2.31.0
requests-oauthlib==1.3.1
rsa==4.9
scipy==1.11.1
six @ file:///tmp/build/80754af9/six_1644875935023/work
sqlparse @ file:///C:/b/abs_88361ub_qu/croot/sqlparse_1690904577514/work
tensorboard==2.13.0
tensorboard-data-server==0.7.1
tensorflow==2.13.0
tensorflow-estimator==2.13.0
tensorflow-intel==2.13.0
tensorflow-io-gcs-filesystem==0.31.0
termcolor==2.3.0
```

Рисунок 9. Файл requirements.txt

```
name: '2.14'
channels:
  - defaults
dependencies:
  - asgiref=3.5.2=py311haa95532_0
  - blas=1.0=mkl
  - bottleneck=1.3.5=py311h5bb9823_0
  - bzip2=1.0.8=he774522_0
  - ca-certificates=2023.08.22=haa95532_0
  - django=4.1=py311haa95532_0
  - icc_rt=2022.1.0=h6049295_2
  - intel-openmp=2023.1.0=h59b6b97_46319
  - libffi=3.4.4=hd77b12b_0
  - mkl=2023.1.0=h6b88ed4_46357
  - mkl-service=2.4.0=py311h2bbff1b_1
  - mkl_fft=1.3.6=py311hf62ec03_1
  - mkl_random=1.2.2=py311hf62ec03_1
  - numexpr=2.8.4=py311h1fcbade_1
  - openssl=3.0.10=h2bbff1b_2
  - pandas=2.0.3=py311hf62ec03_0
  - pip=23.2.1=py311haa95532_0
  - python=3.11.4=he1021f5_0
  - python-dateutil=2.8.2=pyhd3eb1b0_0
  - python-tzdata=2023.3=pyhd3eb1b0_0
  - pytz=2022.7=py311haa95532_0
  - scipy=1.11.1=py311hc1ccb85_0
  - setuptools=68.0.0=py311haa95532_0
  - six=1.16.0=pyhd3eb1b0_1
  - sqlite=3.41.2=h2bbff1b_0
  - sqlparse=0.4.4=py311haa95532_0
  - tbb=2021.8.0=h59b6b97_0
  - tk=8.6.12=h2bbff1b_0
  - tzdata=2023c=h04d1e81_0
  - vc=14.2=h21ff451_1
```

Рисунок 10. Файл environment.yml

7. Слил ветку develop с веткой main и отправил на удаленный сервер.

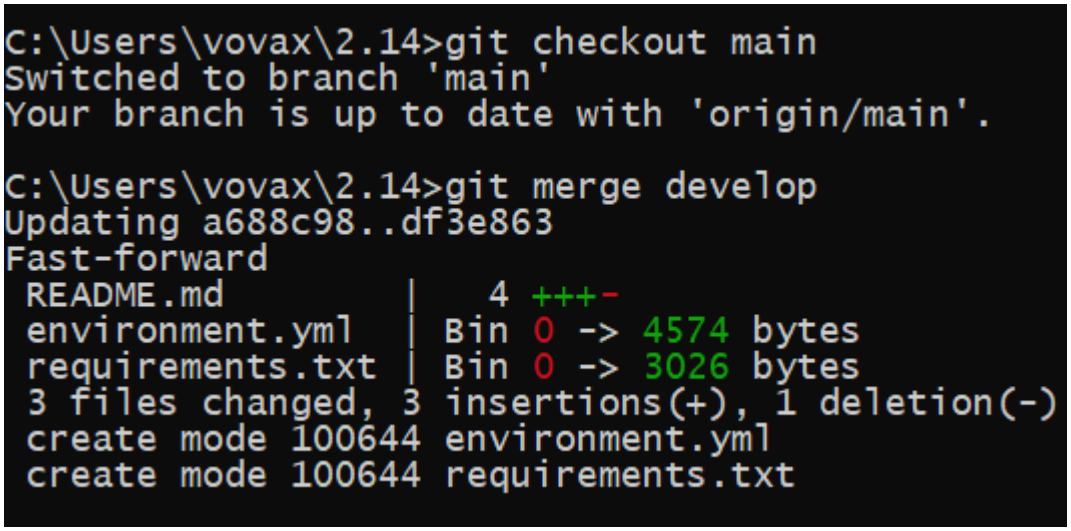

Рисунок 11. Слияние веток

Ссылка на репозиторий: https://github.com/GitVolodya/2.14.git

Контрольные вопросы:

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

Чтобы установить пакет Python, не входящий в стандартную библиотеку, можно использовать менеджер пакетов pip.

2) Как осуществить установку менеджера пакетов pip?

Для установки менеджера пакетов pip нужно сначала установить Python. После установки Python он будет включен в состав стандартной библиотеки. Затем можно проверить наличие pip, введя команду pip --version или python -m pip --version.

3) Откуда менеджер пакетов pip по умолчанию устанавливает пакеты?

Менеджер пакетов pip по умолчанию устанавливает пакеты из Python Package Index (PyPI). PyPI - это репозиторий пакетов Python, где разработчики публикуют свои пакеты для общего использования.

4) Как установить последнюю версию пакета с помощью pip?

Чтобы установить последнюю версию пакета с помощью pip, можно использовать команду pip install <package_name>. При этом pip обнаружит последнюю версию пакета на PyPI и установит ее.

5) Как установить заданную версию пакета с помощью pip?

Чтобы установить заданную версию пакета с помощью pip, можно использовать команду pip install <package_name>==<version>, указав требуемую версию пакета.

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

Чтобы установить пакет из git репозитория с помощью pip, можно использовать команду pip install git+<git_url>, указав URL репозитория. Это может быть как обычный git URL, так и URL GitHub.

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

Чтобы установить пакет из локальной директории с помощью pip, можно использовать команду pip install <path_to_directory>, указав путь к директории, где находится пакет.

8) Как удалить установленный пакет с помощью pip?

Чтобы удалить установленный пакет с помощью pip, можно использовать команду pip uninstall <package_name>. При этом pip удалит указанный пакет и все его зависимости, если они больше не нужны другим установленным пакетам.

9) Как обновить установленный пакет с помощью pip?

Чтобы обновить установленный пакет с помощью pip, можно использовать команду pip install --upgrade <package_name>. При этом pip проверит наличие новых версий пакета на PyPI и установит самую последнюю.

10) Как отобразить список установленных пакетов с помощью pip?

Чтобы отобразить список установленных пакетов с помощью pip, можно использовать команду pip list. Она выведет список всех установленных пакетов вместе с их версиями.

11) Каковы причины появления виртуальных окружений в языке Python?

Виртуальные окружения в языке Python используются для изоляции проектов, чтобы иметь разные зависимости и версии пакетов для разных проектов. Это позволяет избежать конфликтов между зависимостями и облегчает управление пакетами.

12) Каковы основные этапы работы с виртуальными окружениями?

1. Основные этапы работы с виртуальными окружениями включают: Создание виртуального окружения.
2. Активация виртуального окружения.
3. Установка необходимых пакетов в виртуальное окружение.
4. Запуск проекта или скриптов в активированном виртуальном окружении.

13) Как осуществляется работа с виртуальными окружениями с помощью venv?

Работа с виртуальными окружениями с помощью venv включает: Создание виртуального окружения с помощью команды python -m venv <path_to_venv>.

Активация виртуального окружения с помощью команды source <path_to_venv>/bin/activate (для Unix/Linux) или .\<path_to_venv>\Scripts\activate (для Windows).

Установка пакетов в активированное виртуальное окружение с помощью команды pip install <package_name>.

Запуск проекта или скриптов в активированном виртуальном окружении.

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

Работа с виртуальными окружениями с помощью virtualenv аналогична работе с venv, но обычно требует установки отдельным пакетом. Установить virtualenv можно с помощью команды pip install virtualenv. Далее можно

использовать virtualenv для создания, активации и установки пакетов в виртуальных окружениях аналогично venv.

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

Pipenv - это инструмент для управления виртуальными окружениями и зависимостями проектов Python. Он комбинирует функциональность pip и virtualenv в удобном интерфейсе командной строки. Работа с виртуальными окружениями pipenv включает следующие шаги:

Установка pipenv с помощью команды pip install pipenv.

Создание и активация виртуального окружения с помощью команды pipenv shell.

Установка пакетов в виртуальное окружение с помощью команды pipenv install <package_name>.

Запуск проекта или скриптов в активированном виртуальном окружении с помощью команды pipenv run <command>.

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

Файл requirements.txt используется для указания списка зависимостей проекта Python. Он содержит имена пакетов и их версии, которые должны быть установлены для работы проекта. Файл requirements.txt можно создать вручную, перечислив пакеты по одному на строку.

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

Главное преимущество пакетного менеджера conda по сравнению с pip заключается в его способности управлять не только пакетами Python, но и пакетами других языков программирования, а также системными зависимостями. Conda позволяет создавать изолированные окружения со всем необходимым ПО для запуска проектов.

18) В какие дистрибутивы Python входит пакетный менеджер conda?

Пакетный менеджер conda поставляется вместе с дистрибутивами Anaconda и Miniconda.

19) Как создать виртуальное окружение conda?

Для создания виртуального окружения conda нужно использовать команду conda create --name <env_name>. Это создаст новое виртуальное окружение с указанным именем.

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

Чтобы активировать виртуальное окружение conda, нужно использовать команду conda activate <env_name>. После активации окружения можно устанавливать пакеты с помощью команды conda install <package_name>.

21) Как деактивировать и удалить виртуальное окружение conda?

Чтобы деактивировать виртуальное окружение conda, нужно использовать команду conda deactivate. Чтобы удалить виртуальное окружение conda, нужно использовать команду conda env remove --name <env_name>.

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

Файл environment.yml используется для описания зависимостей и настроек среды окружения conda. Он содержит список пакетов и их версий, а также другие параметры среды окружения. Файл environment.yml можно создать вручную.

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

Для создания виртуального окружения conda с помощью файла environment.yml нужно использовать команду conda env create --file <path_to_environment.yml>. Это создаст новое виртуальное окружение и установит все необходимые пакеты.

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

Файлы requirements.txt и environment.yml содержат информацию о зависимостях проекта и версиях пакетов. Хранение этих файлов в репозитории

Git позволяет другим разработчикам легко воссоздать необходимую среду для работы с проектом, установив все зависимости указанных версий. Это повышает переносимость проекта и упрощает совместную работу.