

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 ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины «Основы программной инженерии»

Выполнил:
Магомедов Имран Борисович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., кандидат технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

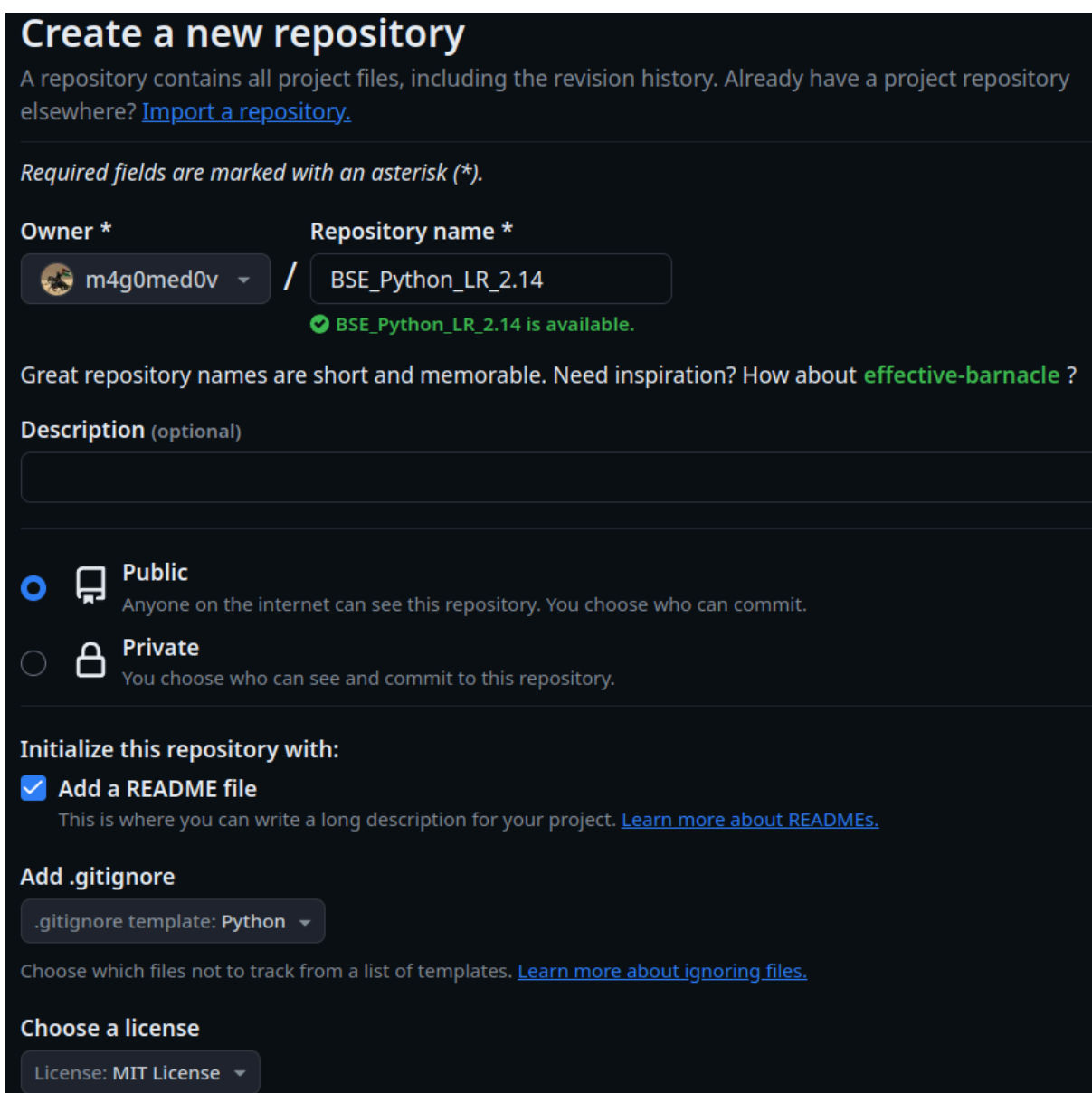
Ставрополь, 2023 г.

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.

МЕТОДИКА ВЫПОЛНЕНИЯ


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



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *  m4g0med0v / **Repository name *** BSE_Python_LR_2.14

✓ BSE_Python_LR_2.14 is available.

Great repository names are short and memorable. Need inspiration? How about **effective-barnacle** ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License

3. Выполните клонирование созданного репозитория.

```

~/work/BSE git clone git@github.com:m4g0med0v/BSE_Python_LR_2.14.git
Клонирование в «BSE_Python_LR_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
Получение объектов: 100% (5/5), готово.

```

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

```

~/w/BSE_Python_LR_2.14 / main git checkout -b develop
Переключились на новую ветку «develop»

```

5. Создайте виртуальное окружение Anaconda с именем репозитория.

```

~/work/BSE_Python_LR_2.14 / git develop sudo conda create -n bse_python_lr_2.14 python=3
[sudo] пароль для m4g0med0v:
WARNING: A conda environment already exists at '/root/.conda/envs/bse_python_lr_2.14'
Remove existing environment (y/[n])? y

Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /root/.conda/envs/bse_python_lr_2.14

added / updated specs:
- python=3

The following packages will be downloaded:

package | build | size
----- | ----- | -----
_libgcc_mutex-0.1 | main | 3 KB
_openmp_mutex-5.1 | 1_gnu | 21 KB
bzip2-1.0.8 | h7b6447c_0 | 78 KB
ca-certificates-2023.12.12 | h06a4308_0 | 126 KB
expat-2.5.0 | h6a678d5_0 | 172 KB
ld_impl_linux-64-2.38 | h1181459_1 | 654 KB
libffi-3.4.4 | h6a678d5_0 | 142 KB
libgcc-ng-11.2.0 | h1234567_1 | 5.3 MB
libgomp-11.2.0 | h1234567_1 | 474 KB
libstdcxx-ng-11.2.0 | h1234567_1 | 4.7 MB
libuuid-1.41.5 | h5eee18b_0 | 27 KB
ncurses-6.4 | h6a678d5_0 | 914 KB
python-3.9.17 | h8c8cd18_0 | 5.9 MB

```

```

~/work/BSE_Python_LR_2.14 / git develop conda activate bse_python_lr_2.14

```

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

```
~/BSE_Python_LR_2.14 / git develop conda install pip numpy pandas scipy ✓ / bse_python_lr_2.14
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /home/m4gomedov/.conda/envs/bse_python_lr_2.14

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

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

```
~/BSE_Python_LR_2.14 / git develop conda install tensorflow ✓ / 1m 2s / bse_python_lr_2.14
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 the existing python installation in your environment:

Specifications:

- tensorflow -> python[version='2.7.*|3.10.*|3.11.*|3.8.*|3.9.*|3.7.*|3.6.*|3.5.*']

Your python: python=3.12.0

If python is on the left-most side of the chain, that's the version you've asked for.
When python appears to the right, that indicates that the thing on the left is somehow
not available for the python version you are constrained to. Note that conda will not
change your python version to a different minor version unless you explicitly specify
that.

The following specifications were found to be incompatible with your system:

- feature:/linux-64::__glibc==2.38=0
- python=3.12.0 -> libgcc-ng[version='>=11.2.0'] -> __glibc[version='>=2.17']

Your installed version is: 2.38
```

```
~/BSE_Python_LR_2.14 / git develop conda install tensorflow 1 x / 7s / bse_python_lr_2.14
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

## Package Plan ##

environment location: /home/m4gomedov/.conda/envs/bse_python_lr_2.14

added / updated specs:
- tensorflow

The following packages will be downloaded:
```

| package | build | size |
|-----------------------|-----------------|---------|
| _tfflow_select-2.3.0 | mk1 | 2 KB |
| abseil-cpp-20211102.0 | hd4dd3e8_0 | 1020 KB |
| absl-py-1.4.0 | py311h06a4308_0 | 238 KB |
| aiohttp-3.9.3 | py311h5eee18b_0 | 825 KB |
| aiohttp-3.9.3 | py311h5eee18b_0 | 12 KB |
| astunparse-1.6.3 | py_0 | 17 KB |
| attrs-23.1.0 | py311h06a4308_0 | 161 KB |
| blinker-1.6.2 | py311h06a4308_0 | 33 KB |
| c-ares-1.19.1 | h5eee18b_0 | 118 KB |
| cachetools-4.2.2 | pyhd3eb1b0_0 | 13 KB |
| certifi-2024.2.2 | py311h06a4308_0 | 160 KB |
| cffi-1.16.0 | py311h5eee18b_0 | 314 KB |

Для установки TensorFlow требовался python 3.11 и ниже.

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

```
~/BSE_Python_LR_2.14 / git P develop ❯ pip install TensorFlow
Collecting TensorFlow
  Downloading tensorflow-2.15.0.post1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.2 kB)
Collecting absl-py>=1.0.0 (from TensorFlow)
  Downloading absl_py-2.1.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse>=1.6.0 (from TensorFlow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting flatbuffers>=23.5.26 (from TensorFlow)
  Downloading flatbuffers-23.5.26-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast>=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from TensorFlow)
  Downloading gast-0.5.4-py3-none-any.whl (19 kB)
Collecting google-pasta>=0.1.1 (from TensorFlow)
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
Collecting h5py>=2.9.0 (from TensorFlow)
  Downloading h5py-3.10.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.5 kB)
Collecting libclang>=13.0.0 (from TensorFlow)
  Downloading libclang-16.0.6-py2.py3-none-manylinux2010_x86_64.whl.metadata (5.2 kB)
Collecting ml-dtypes>=0.2.0 (from TensorFlow)
  Downloading ml_dtypes-0.2.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (20 kB)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in ./bse_python_lr_2.14/lib/python3.11/site-packages (from TensorFlow) (1.26.4)
Collecting opt-einsum>=2.3.2 (from TensorFlow)
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Collecting packaging (from TensorFlow)
  Downloading packaging-23.2-py3-none-any.whl.metadata (3.2 kB)
```

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

```
~/BSE_Python_LR_2.14 / git P develop pip freeze > requirements.txt ✓ bse_python_lr_2.14
~/BSE_Python_LR_2.14 / git P develop ?1 conda env export > environment.yml ✓ bse_python_lr_2.14
```

Файлы `environment.yml` и `requirements.txt` используются для указания зависимостей и конфигурации окружения для проекта на Python.

environment.yml:

- Имя: `bse_python_lr_2.14`
- Каналы: Указывает каналы пакетов, из которых будут установлены зависимости.
 - `defaults`
- Зависимости: Список пакетов, необходимых для окружения, вместе с их версиями.
- Префикс: Указывает каталог установки для окружения.

requirements.txt:

- Специфицирует зависимости пакетов для проекта на Python.

- Каждая строка представляет собой пакет, а также, при необходимости, версию или путь к файлу.
- Пакеты, начинающиеся с `@ file:///`, указывают на локальные пути к распространениям пакетов.

Оба файла содержат исчерпывающий список зависимостей, необходимых для проекта. `environment.yml` обычно используется с `conda` для управления окружениями, в то время как `requirements.txt` является стандартным форматом для указания зависимостей пакетов Python. Версии, указанные в `requirements.txt`, более конкретны по сравнению с теми, что указаны в `environment.yml`. Некоторые пакеты присутствуют в обоих файлах, но с различными версиями.

Эти файлы обеспечивают четкий обзор зависимостей проекта и настройки окружения, позволяя осуществлять воспроизводимые сборки и развертывания.

10. Зафиксируйте сделанные изменения в репозитории.

```

~/w/BSE_Python_LR_2.14 / gitP develop  git push --set-upstream origin develop
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 12 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 3.10 КиБ | 3.10 МБ/с, готово.
Всего 4 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/m4g0med0v/BSE_Python_LR_2.14/pull/new/develop
remote:
To github.com:m4g0med0v/BSE_Python_LR_2.14.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

```

11. Добавьте отчет по лабораторной работе в формате PDF в папку `doc` репозитория. Зафиксируйте изменения.

12. Выполните слияние ветки для разработки с веткой `master/main`.

```

~/w/BSE_Python_LR_2.14 / gitP main  git merge develop
Обновление b8d3480..ed48c59
Fast-forward
 environment.yml | 114 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 requirements.txt | 59 +++++++++++++++++++++++++++++++++++++
 2 files changed, 173 insertions(+)
 create mode 100644 environment.yml
 create mode 100644 requirements.txt

```

13. Отправьте сделанные изменения на сервер GitHub.

14. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

КОНТРОЛЬНЫЕ ВОПРОСЫ

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

Вы можете установить пакет с помощью менеджера пакетов `pip`, указав его имя в командной строке. Например:

```
pip install имя_пакета
```

2. Как осуществить установку менеджера пакетов `pip`?

`pip` обычно устанавливается автоматически при установке Python. Если он не установлен, вы можете установить его, используя инструкции на официальном сайте Python.

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

По умолчанию `pip` устанавливает пакеты из Python Package Index (PyPI), который является центральным репозиторием пакетов Python.

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

Вы можете установить последнюю версию пакета, не указывая конкретную версию, следующим образом:

```
pip install имя_пакета
```

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

Для установки конкретной версии пакета вы можете указать ее после имени пакета, например:

```
pip install имя_пакета==версия
```

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

Вы можете установить пакет напрямую из репозитория Git, указав URL репозитория:

```
pip install git+URL_репозитория
```

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

Для установки пакета из локальной директории вы можете указать путь к нему:

```
pip install /путь/к/пакету
```

8. Как удалить установленный пакет с помощью pip?

Чтобы удалить пакет, используйте команду:

```
pip uninstall имя_пакета
```

9. Как обновить установленный пакет с помощью pip?

Для обновления установленного пакета используйте команду:

```
pip install --upgrade имя_пакета
```

10. Как отобразить список установленных пакетов с помощью pip?

Для просмотра списка установленных пакетов используйте команду:

```
pip list
```

11. Каковы причины появления виртуальных окружений в языке Python?

Виртуальные окружения используются для изоляции зависимостей и проектных настроек между различными проектами, чтобы избежать конфликтов и обеспечить чистоту среды разработки.

12. Каковы основные этапы работы с виртуальными окружениями?

Создание, активация, установка зависимостей, деактивация и удаление.

13. Как осуществляется работа с виртуальными окружениями с помощью `venv`?

Используйте модуль `venv` для создания и управления виртуальными окружениями в Python.

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

Используйте инструмент `virtualenv` для создания и управления виртуальными окружениями.

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

`pipenv` автоматически создает и управляет виртуальными окружениями, а также управляет зависимостями через файл `Pipfile`.

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

Файл `requirements.txt` используется для указания списка зависимостей проекта. Вы можете создать его вручную или автоматически с помощью команды `pip freeze > requirements.txt`. Формат файла - каждая строка содержит имя пакета и версию (опционально).

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

conda управляет как Python пакетами, так и бинарными пакетами, а также имеет возможности для создания и управления виртуальными окружениями.

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

19. Как создать виртуальное окружение conda?

Для создания виртуального окружения conda используйте команду `conda create -n имя_окружения`.

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

Для активации виртуального окружения conda используйте команду `conda activate имя_окружения`, а затем установите пакеты, как обычно, с помощью `pip` или `conda`.

21. Как деактивировать и удалить виртуальное окружение conda?

Для деактивации виртуального окружения conda используйте команду `conda deactivate`, а для удаления - `conda remove -n имя_окружения --all`.

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

Файл `environment.yml` используется для описания окружения и его зависимостей в conda. Его можно создать вручную или сгенерировать с помощью `conda env export > environment.yml`.

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

Вы можете создать виртуальное окружение conda из файла `environment.yml`, используя 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.

В PyCharm вы можете создать новый проект и указать существующее виртуальное окружение conda в настройках проекта. PyCharm автоматически определит окружение и будет использовать его для установки зависимостей.

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

Файлы `requirements.txt` и `environment.yml` должны храниться в репозитории Git для обеспечения воспроизводимости среды разработки и упрощения процесса установки зависимостей для других разработчиков.