

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 №17
дисциплины «Основы программной инженерии»

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

(подпись)

Проверил:
Кандидат технических наук, доцент
кафедры инфокоммуникаций
Воронкин Р. А

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

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.

Ход работы:

1. Изучен теоретический материал работы.
2. Создан общедоступный репозиторий на GitHub, в котором была использована лицензия MIT и язык программирования Python.

Owner * / Repository name *

odik8 / BSE17

✔ BSE17 is available.

Great repository names are short and memorable. Need inspiration? How about [ideal-couscous](#) ?

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

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория

3. Выполнено клонирование созданного репозитория.

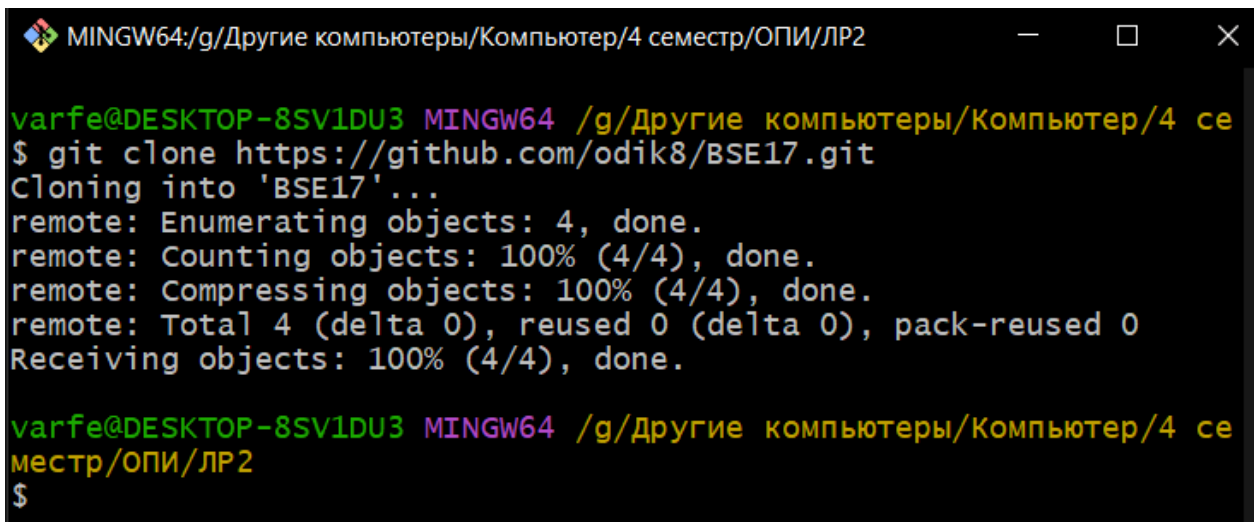
A screenshot of a Windows terminal window with a black background and white text. The title bar shows the path 'MINGW64:/g/Другие компьютеры/Компьютер/4 семестр/ОПИ/ЛР2'. The user 'varfe@DESKTOP-8SV1DU3' is in the 'MINGW64' environment. The command executed is 'git clone https://github.com/odik8/BSE17.git'. The output shows the cloning progress: 'Cloning into 'BSE17'...', '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', and 'Receiving objects: 100% (4/4), done.' The prompt returns to '\$'.

Рисунок 2 – Клонирование репозитория

4. Дополнен файл .gitignore необходимыми правилами для работы с IDE PyCharm.

5. Репозиторий организован в соответствии с моделью ветвления git-flow.

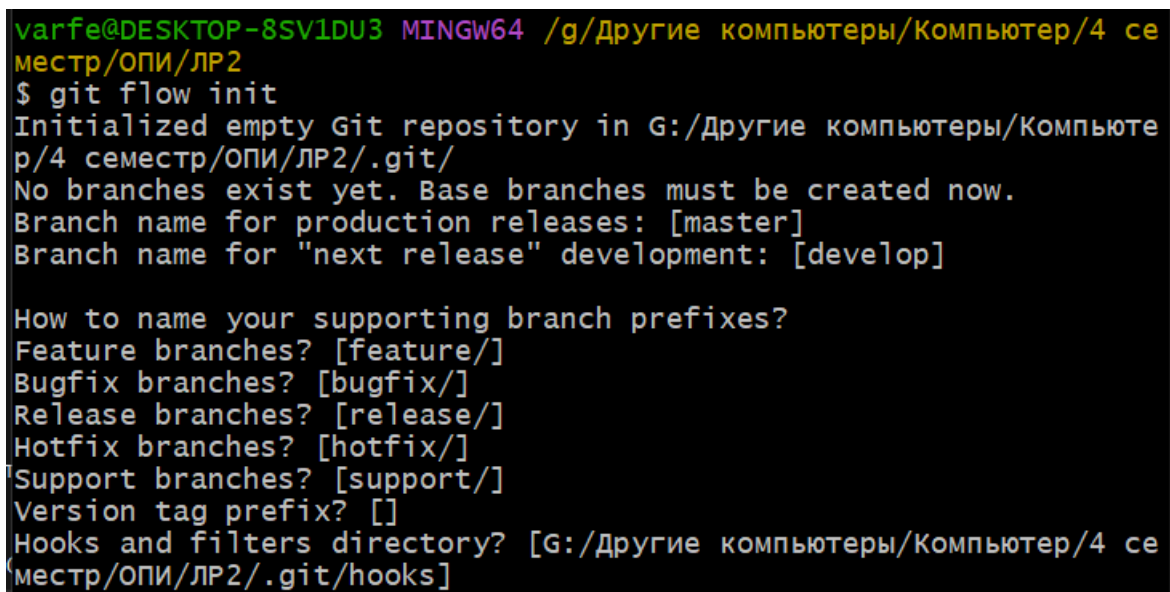
A screenshot of a Windows terminal window with a black background and white text. The user 'varfe@DESKTOP-8SV1DU3' is in the 'MINGW64' environment. The command executed is 'git flow init'. The output shows the initialization of an empty Git repository: 'Initialized empty Git repository in G:/Другие компьютеры/Компьютер/4 семестр/ОПИ/ЛР2/.git/'. It prompts for branch names: 'No branches exist yet. Base branches must be created now.', 'Branch name for production releases: [master]', and 'Branch name for "next release" development: [develop]'. It then asks for supporting branch prefixes: 'How to name your supporting branch prefixes?', 'Feature branches? [feature/]', 'Bugfix branches? [bugfix/]', 'Release branches? [release/]', 'Hotfix branches? [hotfix/]', 'Support branches? [support/]', and 'Version tag prefix? []'. Finally, it asks for the hooks and filters directory: 'Hooks and filters directory? [G:/Другие компьютеры/Компьютер/4 семестр/ОПИ/ЛР2/.git/hooks]'. The prompt returns to '\$'.

Рисунок 3 – Инициализация git-flow

6. Создано виртуальное окружение Anaconda с именем репозитория.

A screenshot of a Windows terminal window with a black background and white text. The user is in the 'base' environment. The command executed is 'conda create -n BSE17 python=3.8'. The output shows a warning: 'WARNING: A conda environment already exists at 'C:/Users/varfe/anaconda3/envs/BSE17''. It asks to remove the existing environment: 'Remove existing environment (y/[n])?'. The user responds with 'y'.

Рисунок 4 – Создание ВО

7. Установлены в виртуальное окружение пакеты: pip, NumPy, Pandas, SciPy

```
(BSE17) PS C:\Users\varfe\Рабочий стол\4 семестр\ОПИ\ЛР2\BSE17> conda install pip, NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Рисунок 5 – Установка пакетов

8. Установлен TensorFlow. Для установки TensorFlow нужен python версии ниже 3.11.

```
(BSE17) PS C:\Users\varfe\Рабочий стол\4 семестр\ОПИ\ЛР2\BSE17> conda install tensorflow
Collecting package metadata (current_repodata.json): done
Solving environment: unsuccessful initial attempt using frozen solve. Retrying with flexible solve.
Solving environment: unsuccessful attempt using repodata from current_repodata.json, retrying with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done
```

Рисунок 6 – Установка tensorflow

9. Установлен TensorFlow с помощью менеджера пакетов pip.

```
(BSE17) PS C:\Users\varfe\Рабочий стол\4 семестр\ОПИ\ЛР2\BSE17> pip install tensorflow
Requirement already satisfied: tensorflow in c:\users\varfe\anaconda3\envs\bse17\lib\site-packages (2.3.0)
Requirement already satisfied: absl-py>=0.7.0 in c:\users\varfe\anaconda3\envs\bse17\lib\site-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse==1.6.3 in c:\users\varfe\anaconda3\envs\bse17\lib\site-packages (from tensorflow) (1.6.3)
Collecting gast==0.3.3 (from tensorflow)
  Downloading gast-0.3.3-py2.py3-none-any.whl (9.7 kB)
Requirement already satisfied: google-pasta==0.1.8 in c:\users\varfe\anaconda3\envs\bse17\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in c:\users\varfe\anaconda3\envs\bse17\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: keras-preprocessing<1.2,>=1.1.1 in c:\users\varfe\anaconda3\envs\bse17\lib\site-packages (from tensorflow) (1.1.2)
Collecting numpy<1.19.0,>=1.16.0 (from tensorflow)
  Downloading numpy-1.18.5-cp38-cp38-win_amd64.whl (12.8 MB)
    12.8/12.8 MB 7.7 MB/s eta 0:00:00
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\varfe\anaconda3\envs\bse17\lib\site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: protobuf>=3.9.2 in c:\users\varfe\anaconda3\envs\bse17\lib\site-packages (from tensorflow) (3.20.3)
Requirement already satisfied: tensorboard<3,>=2.3.0 in c:\users\varfe\anaconda3\envs\bse17\lib\site-packages (from tensorflow) (2.10.0)
Collecting tensorflow-estimator<2.4.0,>=2.3.0 (from tensorflow)
  Downloading tensorflow_estimator-2.3.0-py2.py3-none-any.whl (459 kB)
    459.0/459.0 kB 7.2 MB/s eta 0:00:00
```

Рисунок 7 – Установка tensorflow через pip

10. Сформированы файлы requirements.txt и environment.yml. Проанализируйте содержимое этих файлов.

```
(BSE17) PS C:\Users\varfe\Рабочий стол\4 семестр\ОПИ\ЛР2\BSE17> pip freeze > requirements.txt
(BSE17) PS C:\Users\varfe\Рабочий стол\4 семестр\ОПИ\ЛР2\BSE17> conda env export > environment.yml
```

Рисунок 8 – Формирование файлов requirements.txt и environment.yml

Файл requirements.txt содержит в себе в список зависимостей

Файл environment.yml содержит в себе параметры окружения:

```
name: BSE17
> channels: ...
> dependencies: ...
prefix: C:\Users\varfe\anaconda3\envs\BSE17
```

Рисунок 8 - Файл environment.yml

Контрольные вопросы:

1. Каким способом можно установить пакет Python, не входящий в стандартную библиотеку? – Для скачивания и установки используется специальная утилита, которая называется pip.

2. Как осуществить установку менеджера пакетов pip? – Скачать скрипт и выполнить его:

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

3. Откуда менеджер пакетов pip по умолчанию устанавливает пакеты? – 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. Там также есть возможность выкладывать свои пакеты.

4. Как установить последнюю версию пакета с помощью pip? – Последнюю версию пакета можно установить, не указывая версии:

```
pip install имя_пакета
```

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

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

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

```
pip install git+URL_репозитория
```

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

```
pip install /путь/к/пакету
```

8. Как удалить установленный пакет с помощью pip? –

```
pip uninstall имя_пакета
```

9. Как обновить установленный пакет с помощью pip? –

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

10. Как отобразить список установленных пакетов с помощью pip?

```
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?

Для создания виртуального окружения достаточно дать команду в формате:

```
python3 -m venv <путь к папке виртуального окружения>
```

Чтобы активировать окружение под Linux и macOS нам нужно дать команду:

```
source env/bin/activate
```

Чтобы активировать виртуальное окружение под Windows команда выглядит иначе:

```
> env\\Scripts\\activate
```

Просто под Windows мы вызываем скрипт активации напрямую.

Чтобы деактивировать:

```
deactivate
```

```
source /home/user/envs/project1_env2/bin/activate
```

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

Установку можно выполнить командой:

```
# Для python 3
```

```
python3 -m pip install virtualenv
```

```
# Для единственного python
```

```
python -m pip install virtualenv
```

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

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.

Например, для Linux и macOS:

```
$ source env/bin/activate
```

```
(env) $ deactivate
```

Для операционной системы Windows:

```
> env\\Scripts\\activate
```

```
(env) > deactivate
```

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

Файл requirements.txt содержит в себе список зависимостей

Создание:

```
pip freeze > requirements.txt
```

Формат: .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?

conda входит в дистрибутивы Anaconda и Miniconda

19. Как создать виртуальное окружение conda?

```
conda create -n %PROJ_NAME% python=<version>
```

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

Активация:

```
conda activate %PROJ_NAME%
```

Установка пакетов:

```
conda install <имя_пакета>, <имя_пакета>*...
```

21. Как деактивировать и удалить виртуальное окружение conda?

Деактивация:

```
conda deactivate
```

Удаление:

```
conda remove -n $PROJ_NAME
```

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

Файл environment.yml используется для описания окружения и его зависимостей в conda.

Создание:

```
conda env export > environment.yml.
```

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.

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.