

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное

учреждение высшего образования

**«СЕВЕРО-КАВКАЗСКИЙ
ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт цифрового

развития Кафедра информационных систем и

технологий

Отчет по лабораторной работе №17.

Дисциплина: «Основы программной инженерии»

Выполнил:

Студент группы

ПИЖ-б-о-22-1,

направление

подготовки: 09.03.04

«Программная инженерия»

ФИО: Гуртовой Ярослав

Дмитриевич

Проверил:

Богданов С.С

Ставрополь 2024

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.

1. Изучил теоритический материал работы
2. Создал репозиторий github


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 *

Repository name *

 KingItProgger ▾


/

lr-2.14


 lr-2.14 is available.

Great repository names are short and memorable. Need inspiration? How about [bug-free-octo-funicular](#) ?

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 репоитория

3. Выполните клонирование созданного репозитория.

```
C:\Users\User\Desktop\учеба\4 сем\python> git clone https://github.com/KingItProgger/lr-2.14.git
Cloning into 'lr-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.
```

Рисунок 2. Клонирование репозитория

4. Организовал репозиторий в соответствии с моделью ветвления git-flow

```
C:\Users\User\Desktop\учеба\4 сем\python\lr-2.14> git checkout -b develop
Switched to a new branch 'develop'

C:\Users\User\Desktop\учеба\4 сем\python\lr-2.14>
```

Рисунок 3. Создание ветки develop

5. Создал виртуальное окружение Anaconda с именем репозитория.

```
(base) PS C:\Users\User\Desktop\учеба\4 сем\python\lr-2.14> cd %project%
(base) PS C:\Users\User\Desktop\учеба\4 сем\python\lr-2.14\%project> conda create -n %proj% python=3.11
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.1.2

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

    conda install conda=24.1.2

## Package Plan ##

environment location: D:\anaconda\envs\%proj%

added / updated specs:
- python=3.11
```

Рисунок 4. Создание виртуального окружения

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

```
(%proj%) PS C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14\%project> conda install pip, NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.1.2

Please update conda by running

  $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

  conda install conda=24.1.2

## Package Plan ##

environment location: D:\anaconda\envs\%proj%

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

Рисунок 7. Установка пакетов в виртуальное окружение

7. установил менеджером пакетов conda пакет TensorFlow.

```
(%proj%) PS C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14\%project> 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: unsuccessful initial attempt using 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='3.10.*|3.9.*|3.8.*|3.7.*|3.6.*|3.5.*']

Your python: python=3.11
```

Рисунок 8. Попытка установки Tensorflow

Возникла ошибка, т.к этот пакет поддерживается версиями python 3.5-3.10, у меня установлена версия python 3.11

8. установил пакет TensorFlow с помощью менеджера пакетов pip.

```

----- 300.9/300.9 MB 1.1 MB/s eta 0:00:00
ownloading absl_py-2.1.0-py3-none-any.whl (133 kB)
----- 133.7/133.7 kB 987.7 kB/s eta 0:00:00
ownloading flatbuffers-23.5.26-py2.py3-none-any.whl (26 kB)
ownloading gast-0.5.4-py3-none-any.whl (19 kB)
ownloading grpcio-1.60.1-cp311-cp311-win_amd64.whl (3.7 MB)
----- 3.7/3.7 MB 1.1 MB/s eta 0:00:00
ownloading h5py-3.10.0-cp311-cp311-win_amd64.whl (2.7 MB)
----- 2.7/2.7 MB 1.1 MB/s eta 0:00:00
ownloading keras-2.15.0-py3-none-any.whl (1.7 MB)
----- 1.7/1.7 MB 1.1 MB/s eta 0:00:00
ownloading libclang-16.0.6-py2.py3-none-win_amd64.whl (24.4 MB)

```

Рисунок 9. Установка TensorFlow с помощью pip

Установка прошла успешно.

9. Сформировал файлы requirements.txt и environment.yml .

```

(%proj%) PS C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14\%project> conda %proj% export --file environment.yml
usage: conda-script.py [-h] [--no-plugins] [-V] COMMAND ...
conda-script.py: error: argument COMMAND: invalid choice: '%proj%' (choose from 'clean', 'compare', 'config', 'create',
e', 'render', 'skeleton', 'token', 'server', 'env', 'verify', 'pack', 'repo')
(%proj%) PS C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14\%project>

```

Рисунок 10. Создание файла environment.yml

```

(%proj%) PS C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14\%project> pip freeze > requirements.txt

```

Рисунок 11. Создание файла requirements.txt

10. Зафиксировал сделанные изменения в репозитории.

```

C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14> git add .

C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   %project/requirements.txt
        new file:   main.py

C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14> git commit -m "laba"
[develop 92875f8] laba
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 %project/requirements.txt
 create mode 100644 main.py

C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14>

```

Рисунок 12. Фиксация изменений

11. Выполнил слияние ветки для разработки с веткой master/main.

```
C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14> git merge develop
Updating 285da47..92875f8
Fast-forward
 %project/requirements.txt | Bin 0 -> 3476 bytes
 main.py                  |    0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 %project/requirements.txt
 create mode 100644 main.py
C:\Users\User\Desktop\учеба\4_сем\python\lr-2.14>
```

Рисунок 13. Слияние веток

Вывод: приобретены навыки по работе с менеджером пакетов pip и виртуальными окружениями с помощью языка программирования Python версии 3.x.

Контрольные вопросы:

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

менеджер пакетов pip

pip install <package-name>

2. Как осуществить установку менеджера пакетов pip?

Установка с помощью скрипта get_pip.py. Для этого нужно скачать get_pip.py и запустить его в консоли.

Установка с помощью setuptools. Для этого нужно скачать архив с setuptools из PYPI и распаковать его в отдельный каталог. После этого в терминале перейти в директорию с файлом setup.py и написать: python setup.py install.

3. в AppData\Roaming\Python\Python38\

4. Как установить последнюю версию пакета с помощью pip?

Введите команду ``pip install название_пакета``, где ``название_пакета`` — это название и версия пакета, который вы хотите установить. Если вы хотите установить последнюю версию пакета, то просто напишите название пакета, без указания его версии.

5.Как установить заданную версию пакета с помощью pip?

`pip install Flask==1.1.2` — установить конкретную версию фреймворка Flask;

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

`pip install git+https://github.com/username/repo.git`

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

`pip install /home/user/mypackage/mypackage-1.0.0.tar.gz`

8.Как удалить установленный пакет с помощью pip?

Чтобы удалить установленный пакет с помощью команды `pip install`, вы можете использовать `pip uninstall`.

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?

Изоляция зависимостей. Виртуальное окружение позволяет создавать отдельные изолированные среды для каждого проекта, где можно устанавливать и обновлять зависимости так, чтобы они не влияли на другие проекты.

Чистота и порядок. Создание виртуальных окружений помогает поддерживать чистоту и порядок в рабочем окружении.

Лёгкость переноса. Виртуальные окружения можно легко передавать или копировать на другие компьютеры.

Управление версиями Python. Виртуальные окружения позволяют выбирать, с какой версией Python работать в рамках каждого проекта.

Изоляция от системных пакетов. Создание виртуальных окружений помогает избежать конфликтов между пакетами, установленными на операционной системе, и пакетами, необходимыми для проекта

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

Для создания виртуального окружения с помощью `venv` выполните следующие действия:

Перейдите в директорию своего проекта.

Выполните команду: ``python -m venv``.

В результате будет создан каталог ``venv/``, содержащий копию интерпретатора Python, стандартную библиотеку и другие вспомогательные файлы.

Чтобы начать пользоваться виртуальным окружением, необходимо его активировать:

``venv\Scripts\activate.bat`` — для Windows;

``source venv/bin/activate`` — для Linux и MacOS.

Для деактивации виртуальной среды выполните консольную команду:
`deactivate`

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

VirtualEnv используется для создания виртуальных окружений для Python-программ. Это позволяет устанавливать разные версии библиотек для разных программ.

Основные команды при работе с VirtualEnv:

``mkvirtualenv env-name`` — создание нового окружения.

``workon`` — просмотр списка окружений.

``workon env-name`` — смена окружения.

``deactivate`` — выход из окружения.

``rmvirtualenv env-name`` — удаление окружения.

Находясь в одном из окружений, можно устанавливать пакеты через Pip.

15.Как осуществляется работа с виртуальными окружениями pipenv?

Pipenv — менеджер зависимостей для Python-проектов. С его помощью можно создавать виртуальные среды и управлять зависимостями приложений.

Для своей работы менеджер использует 2 файла:

Pipfile — замена requirements.txt.

Pipfile.lock — связывает версии пакетов, обеспечивая дополнительную безопасность

Когда вы запускаете проект с Pipenv, он автоматически создает виртуальную среду для текущего проекта, даже если вы еще не используете ее.

Pipenv управляет зависимостями, заменяя привычный requirements.txt на новый документ под названием Pipfile. Когда вы устанавливаете библиотеку с помощью Pipenv, файл Pipfile для проекта автоматически обновляется с указанием сведений об этой установке.

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

requirements.txt — это простой текстовый файл, который содержит перечень всех модулей и пакетов, необходимых для корректной работы программы.

Преимущества использования файла зависимостей:

- Возможность отслеживать актуальный список всех модулей и пакетов Python, используемых в проекте.
- Облегчение процесса установки недостающих компонентов.
- Удобство совместной работы. Если на ПК другого пользователя отсутствуют нужные модули, они будут быстро загружены из файла requirements.txt, обеспечив бесперебойный запуск программы.

Чтобы создать файл зависимостей, достаточно перейти в корневой каталог проекта, где хранятся .ру-файлы, и создать текстовый документ requirements.txt. Также этот файл может быть сгенерирован автоматически с помощью команды: `pip freeze > requirements.txt`.

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

Conda — это мощный инструмент для управления пакетами и средами, особенно для научных вычислений и анализа данных. Он позволяет легко

создавать и управлять виртуальными средами с конкретными версиями пакетов, что упрощает воспроизведение результатов и обмен кодом с другими пользователями.

Pip — это легкий и быстрый менеджер пакетов, который широко используется в сообществе Python. Он подходит для установки и управления отдельными пакетами и может использоваться вместе с виртуальными средами, созданными с помощью Conda.

Выбор между Conda и Pip зависит от конкретных потребностей и контекста использования. Если вы работаете над проектом анализа данных со сложными зависимостями и несколькими средами, то Conda может быть лучшим выбором. Если же вы просто устанавливаете и управляете отдельными пакетами или работаете над небольшим проектом, то Pip может быть более подходящим инструментом.

18. В какие дистрибутивы Python входит пакетный менеджер conda?

Пакетный менеджер conda входит в состав дистрибутива anaconda. Более того в дистрибутив anaconda входит ещё и ряд наиболее востребованных библиотек, например, numpy, matplotlib, pandas, scipy, scikit-image, scikit-learn и др. Более редкие библиотеки потребуют установки.

19. Как создать виртуальное окружение conda?

Проверьте, установлена ли conda в вашем пути. Откройте командную строку Anaconda, введите команду `conda -V` и нажмите Enter.

Обновите среду conda. Введите следующую команду в командную строку Anaconda: `conda update conda`.

Создайте виртуальное окружение. Введите команду `conda search ^python$`, чтобы увидеть список доступных версий Python. Теперь замените имя `envname` на имя, которое вы хотите дать своему виртуальному

окружению, и замените x.x на версию Python, которую вы хотите использовать. Например, `conda create -n envname python=x.x anaconda`.

Активируйте виртуальное окружение. Чтобы увидеть список всех доступных окружений, используйте команду `conda info -e`. Для активации виртуального окружения введите следующую команду и замените имя окружения на envname: `conda activate envname`.

Установите необходимые пакеты в виртуальное окружение. Введите следующую команду и замените имя окружения на envname: `conda install -n yourenvname package`.

Деактивируйте виртуальное окружение. Чтобы выйти из конкретного окружения, введите следующую команду: `conda deactivate`.

Удалите виртуальное окружение. Если вам больше не требуется виртуальное окружение, удалите его с помощью следующей команды и замените имя окружения на envname: `conda remove -n envname -a`

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

Откройте Anaconda Prompt или терминал (в зависимости от операционной системы).

Введите команду: ``conda install <имя_пакета>``.

21.Как деактивировать и удалить виртуальное окружение conda?

`conda deactivate`

`conda remove -n ENV_NAME --all`

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

`conda env export > environment.yml`

`environment.yml` — это файл для экспорта и резервного копирования информации об окружении с помощью команды `conda`.

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

Установите Conda.

Создайте файл YAML, содержащий список пакетов и их версий, которые вы хотите включить в окружение Conda.

Создайте окружение Conda с помощью команды: ``conda env create -f environment.yml``. Замените ``environment.yml`` на путь к вашему файлу YAML.

Активируйте окружение Conda с помощью команды: ``conda activate my_env``. Замените ``my_env`` на имя вашей среды.