

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**  
**РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное**  
**образовательное учреждение высшего образования**  
**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Отчет по лабораторной работе №2.14**

**Установка пакетов в Python. Виртуальные окружения**  
**по дисциплине «Технологии программирования и алгоритмизации»**

Выполнил студент группы ИВТ-б-о-20-1

Хашиев Х.М. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_

(подпись)

Ставрополь 2021

**Цель работы:** приобретение навыков по работе с окружениями при написании программ с помощью языка программирования Python версии.

## Ход работы: Примеры

<https://github.com/Mirror-Shard/L2.14>

1. Создал виртуальное окружение с именем env:

Рисунок 1 – Создание виртуального окружения

2. Установил пакеты NumPy, Pandas, SciPy:

```
(base) C:\Users\1\Desktop\Алгоритмизация\Lab9\L2.14\env>pip install Pandas
Requirement already satisfied: Pandas in c:\programdata\anaconda3\lib\site-packages (1.1.3)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\programdata\anaconda3\lib\site-packages (from Pandas) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in c:\programdata\anaconda3\lib\site-packages (from Pandas) (2020.1)
Requirement already satisfied: numpy>=1.15.4 in c:\programdata\anaconda3\lib\site-packages (from Pandas) (1.19.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->Pandas) (1.15.0)

(base) C:\Users\1\Desktop\Алгоритмизация\Lab9\L2.14\env>pip install SciPy
Requirement already satisfied: SciPy in c:\programdata\anaconda3\lib\site-packages (1.5.2)
Requirement already satisfied: numpy>=1.14.5 in c:\programdata\anaconda3\lib\site-packages (from SciPy) (1.19.2)
```

Рисунок 2 – Установка пакетов

3. Попробовал установить пакет TensorFlow с помощью conda, вылезла ошибка, которая сообщила что присутствует несовместимые пакеты:

```
(base) C:\Users\1\Desktop\Алгоритмизация\Lab9\L2.14\env>conda install 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

CondaError: KeyboardInterrupt

Завершить выполнение пакетного файла [Y(да)/N(нет)]? y
```


Рисунок 3 – Установка TensorFlow с помощью conda

4. Установил этот пакет с помощью pip:

```
(base) C:\Users\1\Desktop\Алгоритмизация\Lab9\L2.14\env\env>pip install TensorFlow
Collecting TensorFlow
  Downloading tensorflow-2.7.0-cp38-cp38-win_amd64.whl (430.8 MB)
    |████████████████████████████████████████| 430.8 MB 24 kB/s
Collecting absl-py>=0.4.0
  Downloading absl_py-1.0.0-py3-none-any.whl (126 kB)
    |████████████████████████████████████████| 126 kB 6.8 MB/s
Requirement already satisfied: typing-extensions>=3.6.6 in c:\programdata\anaconda3\lib\site-packages (from TensorFlow) (3.7.4.3)
Collecting termcolor>=1.1.0
  Downloading termcolor-1.1.0.tar.gz (3.9 kB)
Requirement already satisfied: wrapt>=1.11.0 in c:\programdata\anaconda3\lib\site-packages (from TensorFlow) (1.11.0)
Collecting astunparse>=1.6.0
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting tensorboard~2.6
  Downloading tensorboard-2.8.0-py3-none-any.whl (5.8 MB)
    |████████████████████████████████████████| 5.8 MB 364 kB/s
Collecting keras-preprocessing>=1.1.1
  Downloading Keras_Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
    |████████████████████████████████████████| 42 kB 441 kB/s
Collecting tensorflow-estimator<2.8,~2.7.0rc0
  Downloading tensorflow_estimator-2.7.0-py2.py3-none-any.whl (463 kB)
    |████████████████████████████████████████| 463 kB 3.3 MB/s
Collecting opt-einsum>=2.3.2
```

Рисунок 4 – Установка TensorFlow с помощью pip

5. Сформировал файл requirements.txt с помощью команды `pip freeze > requirements.txt`

 requirements.txt – Блокнот

Файл Правка Формат Вид Справка

```
-importlib-metadata @ file:///tmp/build/80754af9/importlib-metadata_1602276842396/work
absl-py==1.0.0
alabaster==0.7.12
anaconda-client==1.7.2
anaconda-navigator==1.10.0
anaconda-project==0.8.3
argh==0.26.2
argon2-cffi @ file:///C:/ci/argon2-cffi_1596828585465/work
asn1crypto @ file:///tmp/build/80754af9/asn1crypto_1596577642040/work
astroid @ file:///C:/ci/astroid_1592487315634/work
astropy==4.0.2
astunparse==1.6.3
async-generator==1.10
atomicwrites==1.4.0
attrs @ file:///tmp/build/80754af9/attrs_1604765588209/work
autopep8 @ file:///tmp/build/80754af9/autopep8_1596578164842/work
Babel @ file:///tmp/build/80754af9/babel_1605108370292/work
backcall==0.2.0
backports.functools-lru-cache==1.6.1
backports.shutil-get-terminal-size==1.0.0
backports.tempfile==1.0
backports.weakref==1.0.post1
bcrypt @ file:///C:/ci/bcrypt_1597936263757/work
```

Рисунок 5 – Создание списка пакетный зависимостей

6. Экспортировал параметры окружения в файл `environment.yml` с помощью `conda`:

```
name: env
channels:
  - defaults
dependencies:
  - _ipyw_jlab_nb_ext_conf=0.1.0=py38_0
  - alabaster=0.7.12=py_0
  - anaconda=2020.11=py38_0
  - anaconda-client=1.7.2=py38_0
  - anaconda-navigator=1.10.0=py38_0
  - anaconda-project=0.8.4=py_0
  - argh=0.26.2=py38_0
  - argon2-cffi=20.1.0=py38he774522_1
  - asn1crypto=1.4.0=py_0
  - astroid=2.4.2=py38_0
  - astropy=4.0.2=py38he774522_0
  - async_generator=1.10=py_0
  - atomicwrites=1.4.0=py_0
  - attrs=20.3.0=pyhd3eb1b0_0
  - autopep8=1.5.4=py_0
  - babel=2.8.1=pyhd3eb1b0_0
  - backcall=0.2.0=py_0
  - backports=1.0=py_2
  - backports.functools_lru_cache=1.6.1=py_0
  - backports.shutil_get_terminal_size=1.0.0=py38_2
```

Рисунок 6 – Экспорт параметров окружения

### Контрольные вопросы:

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

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.

2. Как осуществить установку менеджера пакетов pip?

При развертывании современной версии Python, pip устанавливается автоматически. Но если, по какой-то причине, pip не установлен на вашем

ПК, то сделать это можно вручную. Чтобы установить pip, нужно скачать скрипт get-pip.py и выполнить его.

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?

С помощью команды `$ pip install ProjectName`.

5. Как установить заданную версию пакета с помощью pip?

С помощью команды `$ pip install ProjectName==3.2`, где вместо 3.2 необходимо указать нужную версию пакета.

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

С помощью команды `$ pip install e git+https://gitrepo.com/ ProjectName.git`

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

С помощью команды `$ pip install ./dist/ProjectName.tar.gz`

8. Как удалить установленный пакет с помощью pip?

С помощью команды `$ pip uninstall ProjectName` можно удалить установленный пакет.

9. Как обновить установленный пакет с помощью pip?

С помощью команды `$ pip install --upgrade ProjectName` можно обновить необходимый пакет.

10. Как отобразить список установленных пакетов с помощью pip?

Командой `$ pip list` можно отобразить список установленных пакетов.

11. Каковы причины появления виртуальных окружений в языке Python?

Существует несколько причин появления виртуальных окружений в языке Python - проблема обратной совместимости и проблема коллективной разработки.

Проблема обратной совместимости - некоторые операционные системы, например, Linux и MacOS используют содержащиеся в них предустановленные интерпретаторы Python. Обновив или изменив самостоятельно версию какого-то установленного глобально пакета, мы можем непреднамеренно сломать работу утилит и приложений из дистрибутива операционной системы.

Проблема коллективной разработки - Если разработчик работает над проектом не один, а с командой, ему нужно передавать и получать список зависимостей, а также обновлять их на своем компьютере таким образом,

чтобы не нарушалась работа других его проектов. Значит нам нужен механизм, который вместе с обменом проектами быстро устанавливал бы локально и все необходимые для них пакеты, при этом не мешая работе других проектов.

## 12. Каковы основные этапы работы с виртуальными окружениями?

Основные этапы:

- Создаём через утилиту новое виртуальное окружение в отдельной папке для выбранной версии интерпретатора Python.
- Активируем ранее созданное виртуального окружения для работы.
- Работаем в виртуальном окружении, а именно управляем пакетами используя `pip` и запускаем выполнение кода.
- Деактивируем после окончания работы виртуальное окружение.
- Удаляем папку с виртуальным окружением, если оно нам больше не нужно.

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

С его помощью можно создать виртуальную среду, в которую можно устанавливать пакеты независимо от основной среды или других виртуальных окружений. Основные действия с виртуальными окружениями с помощью `venv`: создание виртуального окружения, его активация и деактивация.

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

Для начала пакет нужно установить. Установку можно выполнить командой: `python3 -m pip install virtualenv`

`Virtualenv` позволяет создать абсолютно изолированное виртуальное окружение для каждой из программ. Окружением является обычная директория, которая содержит копию всего необходимого для запуска определенной программы, включая копию самого интерпретатора, полной

стандартной библиотеки, `pip`, и, что самое главное, копии всех необходимых пакетов.

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

Для формирования и развертывания пакетных зависимостей используется утилита `pip`.

Основные возможности `pipenv`:

- Создание и управление виртуальным окружением
- Синхронизация пакетов в `Pipfile` при установке и удалении пакетов
- Автоматическая подгрузка переменных окружения из `.env` файла

После установки `pipenv` начинается работа с окружением. Его можно создать в любой папке. Достаточно установить любой пакет внутри папки. Используем `requests`, он автоматически установит окружение и создаст `Pipfile` и `Pipfile.lock`.

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

Установить пакеты можно с помощью команды: `pip install -r requirements.txt`. Также можно использовать команду `pip freeze > requirements.txt`, которая создаст `requirements.txt` наполнив его названиями и версиями тех пакетов что используются вами в текущем окружении. Это удобно если вы разработали проект и в текущем окружении все работает, но вы хотите перенести проект в иное окружение (например, заказчику или на сервер).

С помощью закрепления зависимостей мы можем быть уверены, что пакеты, установленные в нашей производственной среде, будут точно соответствовать пакетам в нашей среде разработки, чтобы ваш проект неожиданно не ломался.

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

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?

Все чаще среди Python-разработчиков заходит речь о менеджере пакетов conda, включенный в состав дистрибутивов Anaconda и Miniconda. JetBrains включил этот инструмент в состав PyCharm.

19. Как создать виртуальное окружение conda?

С помощью команды: `conda create -n %PROJ_NAME% python=3.7`

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

Чтобы установить пакеты, необходимо воспользоваться командой:

– `conda install`

А для активации:

`conda activate %PROJ_NAME%`

21. Как деактивировать и удалить виртуальное окружение conda?

Для деактивации использовать команду: `conda deactivate`, а для удаления: `conda remove -n $PROJ_NAME`.

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

Файл `environment.yml` позволит воссоздать окружение в любой нужный момент.

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

Достаточно набрать:

`conda env create -f environment.yml`

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

Чтобы пользователи, которые скачивают какие-либо программы, скрипты, модули могли без проблем посмотреть, какие пакеты им нужно



установить дополнительно для корректной работы. За описание о наличии каких-либо пакетов в среде как раз и отвечают файлы requirements.txt и environment.yml.

**Вывод:** в ходе работы приобрёл навыки по работе с модулями и пакетами при написании программ с помощью языка программирования Python версии 3.