

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 алгоритмизация»

Выполнил

студент группы ИВТ-б-о-20-1

Дыбов Д.В. « » _____ 20__ г.

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2021

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

Ход работы

1. Создал новый репозиторий для лабораторной работы №2.14;
2. Клонировал созданный репозиторий на компьютер;
3. Создал новый PyCharm проект в папке репозитория;
4. Создал виртуальное окружение;

```
(base) PS C:\Users\megad> cd PycharmProjects
(base) PS C:\Users\megad\PycharmProjects> mkdir laba2.14

Каталог: C:\Users\megad\PycharmProjects

Mode                LastWriteTime         Length Name
----                -
d-----          08.01.2022    21:14             laba2.14

(base) PS C:\Users\megad\PycharmProjects> cd laba2.14
(base) PS C:\Users\megad\PycharmProjects\laba2.14> conda create -n laba2.14 python=3.8.5
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.9.2
  latest version: 4.11.0

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\Users\megad\anaconda3\envs\laba2.14

added / updated specs:
- python=3.8.5

The following packages will be downloaded:
```

package	build	size
ca-certificates-2021.10.26	haa95532_2	115 KB
certifi-2021.10.8	py38haa95532_0	152 KB
openssl-1.1.1l	h2bbff1b_0	4.8 MB
pip-21.2.2	py38haa95532_0	1.9 MB
setuptools-58.0.4	py38haa95532_0	779 KB
sqlite-3.37.0	h2bbff1b_0	785 KB
vc-14.2	h21ff451_1	8 KB
vs2015_runtime-14.27.29016	h5e58377_2	1007 KB

Рисунок 1 – Создание виртуального окружения

5. Установил в виртуальное окружение пакет `pip`;

```

(base) PS C:\Users\megad\PycharmProjects\laba2.14> conda install pip
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\megad\anaconda3

added / updated specs:
- pip

The following packages will be downloaded:



| package      | build          |         |
|--------------|----------------|---------|
| conda-4.11.0 | py38haa95532_0 | 14.4 MB |
| Total:       |                | 14.4 MB |



The following packages will be UPDATED:

conda 4.9.2-py38haa95532_0 --> 4.11.0-py38haa95532_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
conda-4.11.0 | 14.4 MB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) PS C:\Users\megad\PycharmProjects\laba2.14>

```

Рисунок 2 – Установка пакета pip

6. Написал команду для установки пакетов NumPy, Pandas и SciPy;
7. Консоль выдала сообщение, что эти пакеты уже установлены;

```

(base) PS C:\Users\megad\PycharmProjects\laba2.14> conda install NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.

(base) PS C:\Users\megad\PycharmProjects\laba2.14>

```

Рисунок 3 – Результат выполнения установки пакетов

8. Попытался установить пакет TensorFlow;


```
(base) PS C:\Users\megad\PycharmProjects\laba2.14> pip install tensorflow
Collecting TensorFlow
  Using cached tensorflow-2.7.0-cp38-cp38-win_amd64.whl (430.8 MB)
Collecting keras-preprocessing>=1.1.1
  Downloading Keras_Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
    |████████████████████| 42 kB 1.7 MB/s
Collecting termcolor>=1.1.0
  Downloading termcolor-1.1.0.tar.gz (3.9 kB)
Collecting absl-py>=0.4.0
  Downloading absl_py-1.0.0-py3-none-any.whl (126 kB)
    |████████████████████| 126 kB 1.1 MB/s
Requirement already satisfied: wheel<1.0,>=0.32.0 in c:\users\megad\anaconda3\lib\site-packages (from TensorFlow) (0.35.1)
Collecting opt-einsum>=2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
    |████████████████████| 65 kB 1.3 MB/s
Collecting grpcio<2.0,>=1.24.3
  Downloading grpcio-1.43.0-cp38-cp38-win_amd64.whl (3.4 MB)
    |████████████████████| 3.4 MB 3.3 MB/s
Collecting libclang>=9.0.1
  Downloading libclang-12.0.0-2-py2.py3-none-win_amd64.whl (13.0 MB)
    |████████████████████| 13.0 MB 595 kB/s
Requirement already satisfied: h5py>=2.9.0 in c:\users\megad\anaconda3\lib\site-packages (from TensorFlow) (2.10.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\megad\anaconda3\lib\site-packages (from TensorFlow) (1.11.2)
Collecting keras<2.8,>=2.7.0rc0
  Downloading keras-2.7.0-py2.py3-none-any.whl (1.3 MB)
    |████████████████████| 1.3 MB 3.3 MB/s
Collecting protobuf>=3.9.2
  Downloading protobuf-3.19.1-cp38-cp38-win_amd64.whl (895 kB)
    |████████████████████| 895 kB 3.2 MB/s
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\megad\anaconda3\lib\site-packages (from TensorFlow) (3.7.4.3)
Collecting tensorflow-io-gcs-filesystem>=0.21.0
  Downloading tensorflow_io_gcs_filesystem-0.23.1-cp38-cp38-win_amd64.whl (1.5 MB)
    |████████████████████| 1.5 MB 3.2 MB/s
Collecting google-pasta>=0.1.1
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
    |████████████████████| 57 kB 1.9 MB/s
Collecting gast<0.5.0,>=0.2.1
  Downloading gast-0.4.0-py3-none-any.whl (9.8 kB)
Requirement already satisfied: numpy>=1.14.5 in c:\users\megad\anaconda3\lib\site-packages (from TensorFlow) (1.19.2)
Collecting tensorboard~2.6
  Downloading tensorboard-2.7.0-py3-none-any.whl (5.8 MB)
    |████████████████████| 5.8 MB 6.8 MB/s
Collecting tensorflow-estimator<2.8,~2.7.0rc0
  Downloading tensorflow_estimator-2.7.0-py2.py3-none-any.whl (463 kB)
    |████████████████████| 463 kB 6.4 MB/s
Requirement already satisfied: six>=1.12.0 in c:\users\megad\anaconda3\lib\site-packages (from TensorFlow) (1.15.0)
Collecting flatbuffers<3.0,>=1.12
  Downloading flatbuffers-2.0-py2.py3-none-any.whl (26 kB)
Collecting astunparse>=1.6.0
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
```

Рисунок 5 – Установка TensorFlow через pip

10. Сформировал файлы environment.yml и requirements.txt;

```
(base) PS C:\Users\megad\PycharmProjects\laba2.14> conda env export > environment.yml
(base) PS C:\Users\megad\PycharmProjects\laba2.14> pip freeze > requirements.txt
(base) PS C:\Users\megad\PycharmProjects\laba2.14>
```

Рисунок 6 – Формирование файлов environment.yml и requirements.txt

11. Отправил файлы в репозиторий.

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 виртуальное окружение 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 окружением (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.

Чтобы создать этот файл достаточно использовать команду: `conda env export > environment.yml`

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

Достаточно набрать: `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.

Работа с виртуальными окружениями в PyCharm зависит от способа взаимодействия с виртуальным окружением:

Создаём проект со своим собственным виртуальным окружением, куда затем будут устанавливаться необходимые библиотеки.

Предварительно создаём виртуальное окружение, куда установим нужные библиотеки. И затем при создании проекта в PyCharm можно будет его выбирать, т.е. использовать для нескольких проектов.

Для первого способа ход работы следующий: запускаем PyCharm и в окне приветствия выбираем `Create New Project`. В мастере создания проекта, указываем в поле `Location` путь расположения создаваемого проекта. Имя конечной директории также является именем проекта. Далее разворачиваем параметры окружения, щелкая по `Project Interpreter`. И выбираем `New environment using Virtualenv`. Путь расположения окружения генерируется автоматически. И нажимаем на `Create`. Теперь установим библиотеки, которые будем использовать в программе. С помощью главного меню переходим в настройки `File → Settings`. Где переходим в `Project: project_name → Project Interpreter`. Выходим из настроек. Для запуска программы, необходимо создать профиль с конфигурацией. Для этого в верхнем правом углу нажимаем на кнопку `Add Configuration`. Откроется окно `Run/Debug Configurations`, где нажимаем на кнопку с плюсом (`Add New Configuration`) в

правом верхнем углу и выбираем Python. Далее указываем в поле Name имя конфигурации и в поле Script path расположение Python файла с кодом программы. В завершение нажимаем на Apply, затем на ОК. Для второго способа необходимо сделать следующее: на экране приветствия в нижнем правом углу через Configure → Settings переходим в настройки. Затем переходим в раздел Project Interpreter. В верхнем правом углу есть кнопка с шестерёнкой, нажимаем на неё и выбираем Add, создавая новое окружение. И указываем расположение для нового окружения. Нажимаем на ОК. Далее в созданном окружении устанавливаем нужные пакеты. И выходим из настроек. В окне приветствия выбираем Create New Project. В мастере создания проекта, указываем имя расположения проекта в поле Location. Разворачиваем параметры окружения, щелкая по Project Interpreter, где выбираем Existing interpreter и указываем нужное нам окружение. Далее создаем конфигурацию запуска программы, также как создавали для раннее. После чего можно выполнить программу.

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 выполнения лабораторной работы были приобретены навыки по работе с менеджером пакетов pip и виртуальными окружениями с помощью языка программирования Python версии 3.x.