

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

Кафедра инфокоммуникаций

Отчет по лабораторной работе №9

Установка пакетов в Python. Виртуальные окружения

По дисциплине «Теории программирования и алгоритмизации»

Выполнил студент группы ИВТ-б-о-20-1

Плотников Д. В. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

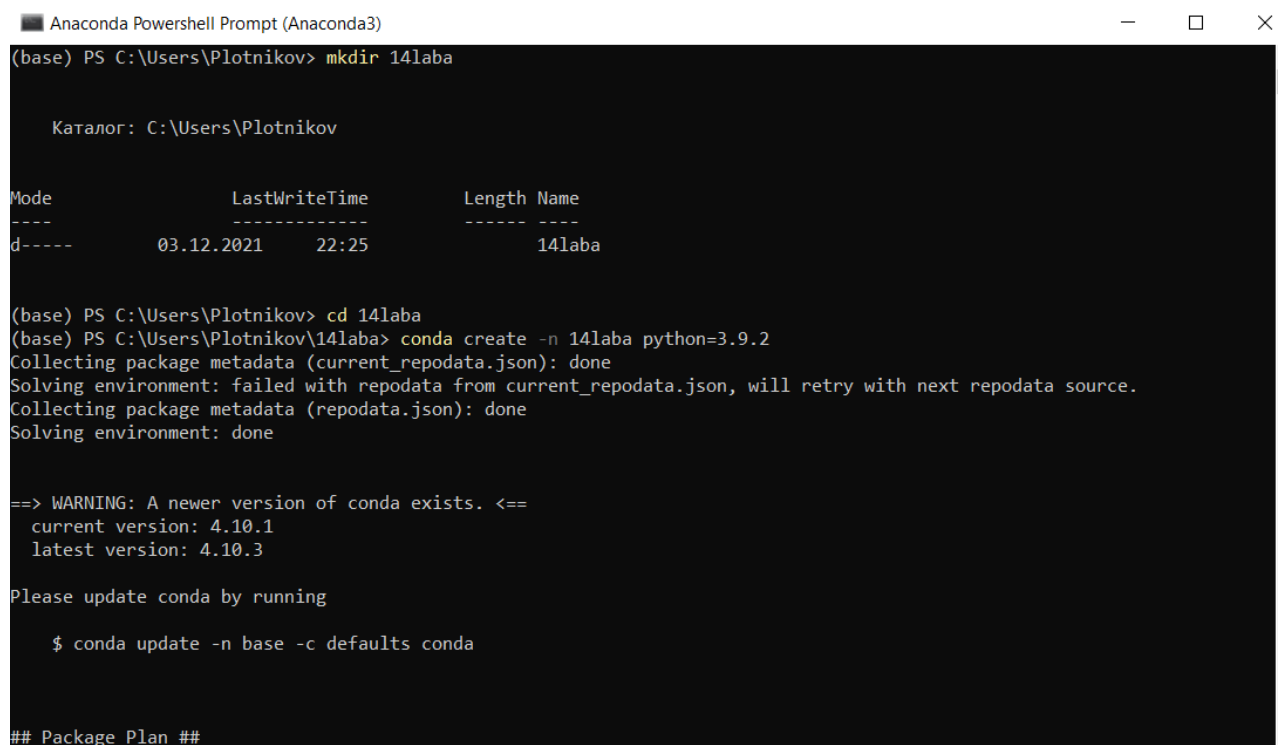
Проверил Воронкин Р. А. _____

(подпись)

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 собственный репозиторий. Ссылка на репозиторий: https://github.com/Dmitry-15/16_laba.git.
2. С помощью команды git clone клонировал удаленный репозиторий на свой ПК. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.
3. Создал виртуальное окружение Anaconda с именем репозитория.



```
Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\Plotnikov> mkdir 14laba

Каталог: C:\Users\Plotnikov

Mode                LastWriteTime         Length Name
----                -
d-----          03.12.2021    22:25             14laba

(base) PS C:\Users\Plotnikov> cd 14laba
(base) PS C:\Users\Plotnikov\14laba> conda create -n 14laba python=3.9.2
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: 4.10.1
  latest version: 4.10.3

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##
```

Рисунок 1. Процесс создания окружения

4. Активировал виртуальное окружение и установил следующие пакеты pip, NumPy, Pandas и SciPy.

```
Anaconda Powershell Prompt (Anaconda3)

(base) PS C:\Users\Plotnikov\141aba> conda activate 141aba
(141aba) PS C:\Users\Plotnikov\141aba> conda install pip
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.10.1
  latest version: 4.10.3

Please update conda by running

    $ conda update -n base -c defaults conda

# All requested packages already installed.

(141aba) PS C:\Users\Plotnikov\141aba> conda install NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.10.1
  latest version: 4.10.3

Please update conda by running

    $ conda update -n base -c defaults conda
```

Рисунок 2. Активация виртуального окружения и распаковка пакетов

```
Proceed ([y]/n)? y

Downloading and Extracting Packages
bottleneck-1.3.2      | 107 KB | ##### | 100%
mkl-2021.4.0         | 114.9 MB | ##### | 100%
six-1.16.0           | 18 KB | ##### | 100%
intel-openmp-2021.4. | 2.2 MB | ##### | 100%
mkl_random-1.2.2     | 225 KB | ##### | 100%
python-dateutil-2.8. | 233 KB | ##### | 100%
numexpr-2.7.3        | 126 KB | ##### | 100%
numpy-1.21.2         | 24 KB | ##### | 100%
mkl_fft-1.3.1        | 139 KB | ##### | 100%
pandas-1.3.4         | 8.6 MB | ##### | 100%
scipy-1.7.1          | 13.8 MB | ##### | 100%
numpy-base-1.21.2    | 4.4 MB | ##### | 100%
pytz-2021.3          | 171 KB | ##### | 100%
mkl-service-2.4.0    | 51 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Рисунок 3. Результат установки пакетов

5. Установил пакет TensorFlow с помощью менеджера пакетов pip.

```
Anaconda Powershell Prompt (Anaconda3)
(14laba) PS C:\Users\Plotnikov\14laba> pip install TensorFlow
Collecting TensorFlow
  Downloading tensorflow-2.7.0-cp39-cp39-win_amd64.whl (430.8 MB)
    | 430.8 MB 99 kB/s
Collecting astunparse==1.6.0
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting flatbuffers<3.0,>=1.12
  Downloading flatbuffers-2.0-py2.py3-none-any.whl (26 kB)
Collecting typing-extensions==3.6.6
  Downloading typing_extensions-4.0.1-py3-none-any.whl (22 kB)
Collecting grpcio<2.0,>=1.24.3
  Downloading grpcio-1.42.0-cp39-cp39-win_amd64.whl (3.3 MB)
    | 3.3 MB 6.4 MB/s
Collecting wrapt==1.11.0
  Downloading wrapt-1.13.3-cp39-cp39-win_amd64.whl (34 kB)
Collecting tensorboard~=2.6
  Downloading tensorboard-2.7.0-py3-none-any.whl (5.8 MB)
    | 5.8 MB 6.4 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 819 kB/s
Collecting libclang>=9.0.1
  Downloading libclang-12.0.0-py2.py3-none-win_amd64.whl (13.1 MB)
    | 13.1 MB 3.3 MB/s
Collecting opt-einsum==2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
    | 65 kB 1.8 MB/s
Collecting protobuf==3.9.2
  Downloading protobuf-3.19.1-cp39-cp39-win_amd64.whl (895 kB)
    | 895 kB 6.4 MB/s
```

Рисунок 4. Результат установки пакета TensorFlow

6. Затем, сформировал файлы requirements.txt и environment.yml, проанализировав содержимое этих файлов.

```
(14laba) PS C:\Users\Plotnikov\14laba> conda env export > environment.yml
(14laba) PS C:\Users\Plotnikov\14laba> pip freeze > requirements.txt
(14laba) PS C:\Users\Plotnikov\14laba>
```

Рисунок 5. Формирование файлов environment и requirements

```
requirements.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
abs1-py==1.0.0
astunparse==1.6.3
Bottleneck @ file:///C:/ci/bottleneck_1607557040328/work
cachetools==4.2.4
certifi==2021.10.8
charset-normalizer==2.0.9
flatbuffers==2.0
gast==0.4.0
google-auth==2.3.3
google-auth-oauthlib==0.4.6
google-pasta==0.2.0
grpcio==1.42.0
h5py==3.6.0
idna==3.3
importlib-metadata==4.8.2
keras==2.7.0
Keras-Preprocessing==1.1.2
libclang==12.0.0
Markdown==3.3.6
mkl-fft==1.3.1
mkl-random @ file:///C:/ci/mkl_random_1626186184308/work
mkl-service==2.4.0
numexpr @ file:///C:/ci/numexpr_1618856728739/work
numpy @ file:///C:/ci/numpy_and_numpy_base_1634088717532/work
oauthlib==3.1.1
opt-einsum==3.3.0
pandas @ file:///C:/ci/pandas_1635506685681/work
protobuf==3.19.1
```

Рисунок 6. Содержимое файла requirements



Dmitry-15 Задания

1 contributor

71 lines (71 sloc) | 3.94 KB

```
1 name: 14laba
2 channels:
3   - defaults
4 dependencies:
5   - blas=1.0=mkl
6   - bottleneck=1.3.2=py39h7cc1a96_1
7   - ca-certificates=2021.10.26=haa95532_2
8   - certifi=2021.10.8=py39haa95532_0
9   - icc_rt=2019.0.0=h0cc432a_1
10  - intel-openmp=2021.4.0=haa95532_3556
11  - mkl=2021.4.0=haa95532_640
12  - mkl-service=2.4.0=py39h2bbff1b_0
13  - mkl_fft=1.3.1=py39h277e83a_0
14  - mkl_random=1.2.2=py39hf11a4ad_0
15  - numexpr=2.7.3=py39hb80d3ca_1
16  - numpy=1.21.2=py39hfca59bb_0
17  - numpy-base=1.21.2=py39h0829f74_0
18  - openssl=1.1.11=h2bbff1b_0
19  - pandas=1.3.4=py39h6214cd6_0
20  - pip=21.2.4=py39haa95532_0
21  - python=3.9.2=h6244533_0
22  - python-dateutil=2.8.2=pyhd3eb1b0_0
23  - pytz=2021.3=pyhd3eb1b0_0
24  - scipy=1.7.1=py39hbe87c03_2
25  - setuptools=58.0.4=py39haa95532_0
```

Рисунок 7. Содержимое файла environment

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 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`

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 **OK**.

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 **OK**. 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 успешно приобрел навыки по работе с менеджером пакетов pip и виртуальными окружениями с помощью языка программирования Python версии 3.x.