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

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 16 (2.14)

Установка пакетов в Python. Виртуальное окружение

по дисциплине «Технологии программирования и алгоритмизации»

Выполнил студент группы ИВТ-б-о-20-1
Злыгостев И.С. « »_____20__г.
Подпись студента_____
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 виртуальное окружение env и затем активировал его.

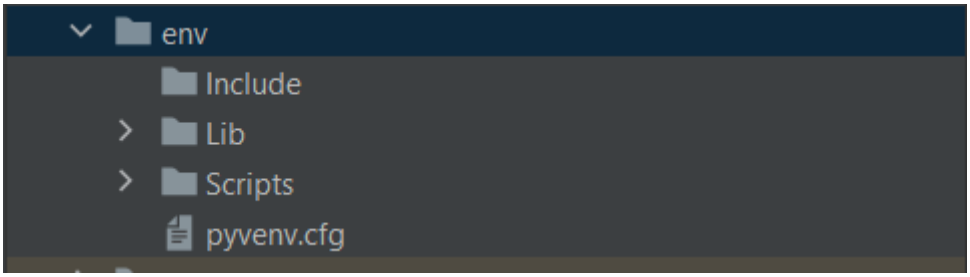

Рисунок 16.1 – Виртуальное окружение

```
(venv) C:\Users\zligo\git>env\\Scripts\\activate

(env) C:\Users\zligo\git>
```

Рисунок 16.2 – Активация окружения

2. После этого приступил к установке пакетов в виртуальном окружении.

```
C:\Users\zligo\git>pip freeze
backports.entry-points-selectable==1.1.0
distlib==0.3.2
filelock==3.0.12
platformdirs==2.3.0
six==1.16.0
virtualenv==20.7.2

C:\Users\zligo\git>pip freeze > requirements.txt

C:\Users\zligo\git>pip install -r requirements.txt
Requirement already satisfied: backports.entry-points-selectable==1.1.0 in c:
s\python39\site-packages (from -r requirements.txt (line 1)) (1.1.0)
Requirement already satisfied: distlib==0.3.2 in c:\users\zligo\appdata\local
from -r requirements.txt (line 2)) (0.3.2)
Requirement already satisfied: filelock==3.0.12 in c:\users\zligo\appdata\loc
 (from -r requirements.txt (line 3)) (3.0.12)
Requirement already satisfied: platformdirs==2.3.0 in c:\users\zligo\appdata\
ges (from -r requirements.txt (line 4)) (2.3.0)
Requirement already satisfied: six==1.16.0 in c:\users\zligo\appdata\local\pa
m -r requirements.txt (line 5)) (1.16.0)
Requirement already satisfied: virtualenv==20.7.2 in c:\users\zligo\appdata\l
es (from -r requirements.txt (line 6)) (20.7.2)
```

Рисунок 16.3 – Установка при помощи pip

3. Затем приступил к выполнению общего задания

4. Создал окружение с названием лабораторной работы

```
(%PROJ_NAME%) PS C:\Users\zligo\git> conda create -n demo-2.14
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



## Package Plan ##

  environment location: C:\Users\zligo\.conda\envs\demo-2.14



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

Рисунок 16.4 – Создание окружения

5. Активировал её и приступил к установке пакетов

```
(demo-2.14) PS C:\Users\zligo\git\demo-2.14> 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: 4.10.1
  latest version: 4.10.3

Please update conda by running

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

Рисунок 16.5 – Установка пакетов

```
(demo-2.14) PS C:\Users\zligo\git\demo-2.14> pip install TensorFlow
Collecting TensorFlow
  Downloading tensorflow-2.7.0-cp38-cp38-win_amd64.whl (430.8 MB)
     |                                | 7.7 MB 1.3 MB/s eta 0:05:23
```

Рисунок 16.6 – Установка TensorFlow при помощи pip

6. Создал файлы requirements.txt и environment.yml и ознакомился с их содержимым.

```
name: demo-2.14
channels:
  - defaults
dependencies:
  - _tflow_select=2.3.0=mkl
  - abseil-cpp=20210324.2=hd77b12b_0
  - absl-py=0.13.0=py39haa95532_0
  - aiohttp=3.7.4.post0=py39h2bbff1b_2
  - astor=0.8.1=py39haa95532_0
  - astunparse=1.6.3=py_0
  - async-timeout=3.0.1=py39haa95532_0
  - attrs=21.2.0=pyhd3eb1b0_0
  - blas=1.0=mkl
  - blinker=1.4=py39haa95532_0
  - bottleneck=1.3.2=py39h7cc1a96_1
  - brotlipy=0.7.0=py39h2bbff1b_1003
  - ca-certificates=2021.10.26=haa95532_2
  - cachetools=4.2.2=pyhd3eb1b0_0
  - certifi=2021.10.8=py39haa95532_0
  - cffi=1.15.0=py39h2bbff1b_0
  - chardet=4.0.0=py39haa95532_1003
  - charset-normalizer=2.0.4=pyhd3eb1b0_0
  - click=8.0.3=pyhd3eb1b0_0
  - coverage=5.5=py39h2bbff1b_2
  - cryptography=3.4.8=py39h71e12ea_0
  - cython=0.29.24=py39h604cdb4_0
  - dataclasses=0.8=pyh6d0b6a4_7
  - flatbuffers=2.0.0=h6c2663c_0
  - gast=0.4.0=pyhd3eb1b0_0
  - giflib=5.2.1=h62dcd97_0
  - google-auth=1.33.0=pyhd3eb1b0_0
  - google-auth-oauthlib=0.4.1=py_2
  - google-pasta=0.2.0=pyhd3eb1b0_0
  - grpcio=1.36.1=py39hc60d5dd_1
  - h5py=3.6.0=py39h3de5c98_0
  - hdf5=1.10.6=h7ebc959_0
  - icc_rt=2019.0.0=h0cc432a_1
  - icu=68.1=h6c2663c_0
  - idna=3.2=pyhd3eb1b0_0
  - importlib-metadata=4.8.1=py39haa95532_0
  - intel-openmp=2021.4.0=haa95532_3556
  - jpeg=9d=h2bbff1b_0
```

Рисунок 16.7 – Содержимое файла requirements.txt

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/ ProjectNa me.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 совмести - 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 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.

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 были приобретены навыки по установке пакетов в языке программирования Python, создании виртуальных окружений и работе с ними.