

**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

Лысенко И.А. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверила Воронкин Р.А. _____

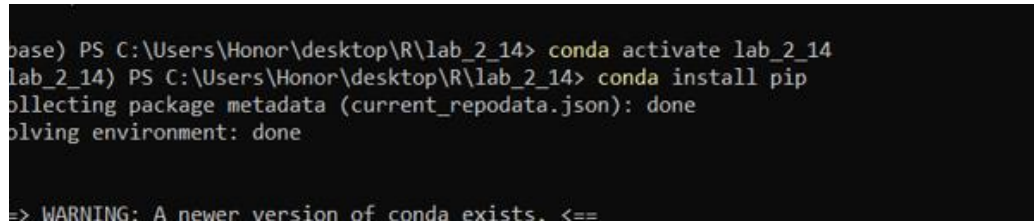
(подпись)

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.

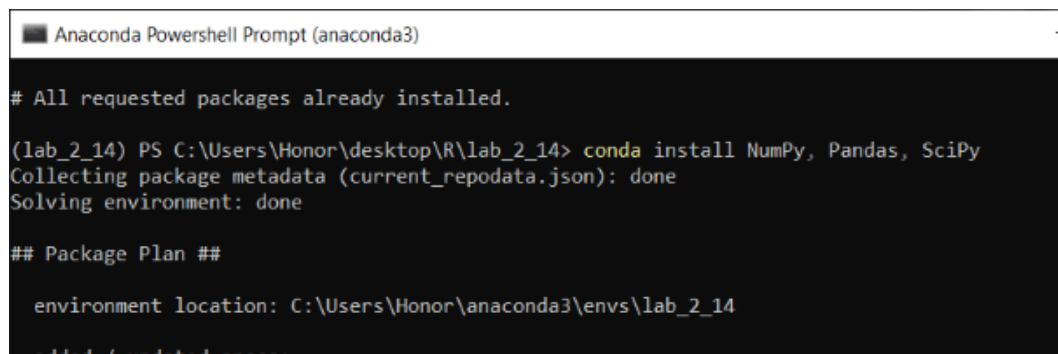


```
base) PS C:\Users\Honor\Desktop\R\lab_2_14> conda activate lab_2_14
lab_2_14) PS C:\Users\Honor\Desktop\R\lab_2_14> conda install pip
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
```

Рисунок 1 – Создание виртуального окружения

2. Установил в виртуальное окружение пакеты `pip`, `NumPy`, `Pandas`, `SciPy`.



```
Anaconda Powershell Prompt (anaconda3)

# All requested packages already installed.

(lab_2_14) PS C:\Users\Honor\Desktop\R\lab_2_14> conda install NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done

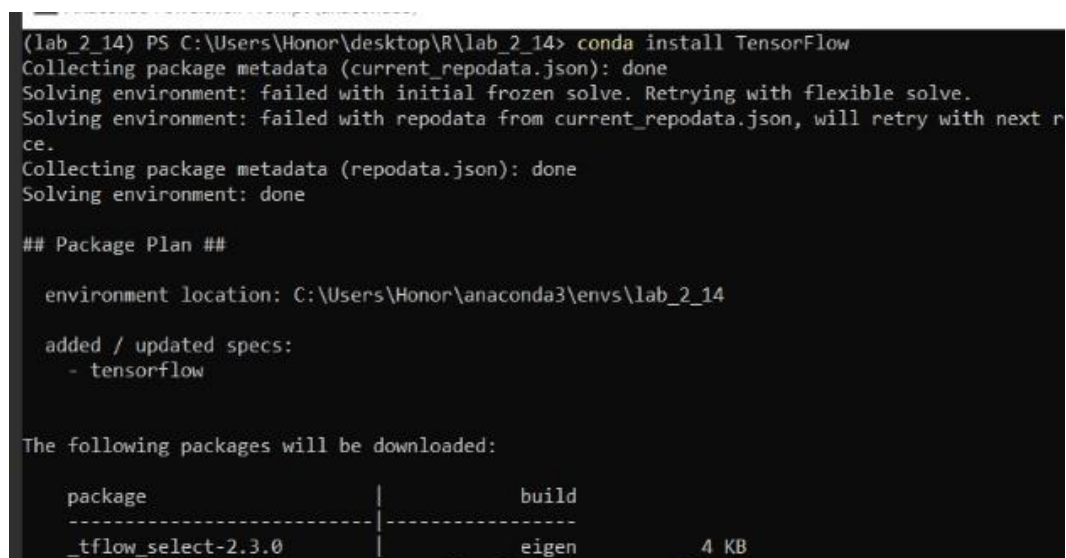
## Package Plan ##

environment location: C:\Users\Honor\anaconda3\envs\lab_2_14

added / updated specs:
```

Рисунок 2 – Установка пакетов

3. Попробовал и успешно установил пакет `TensorFlow` менеджером пакетов `Anaconda`.



```
(lab_2_14) PS C:\Users\Honor\Desktop\R\lab_2_14> 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 r
ce.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\Users\Honor\anaconda3\envs\lab_2_14

added / updated specs:
- tensorflow

The following packages will be downloaded:
```

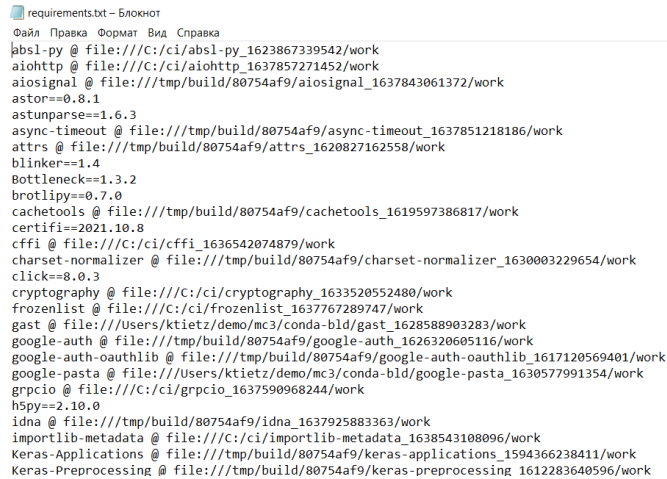
package	build	size
_tflow_select-2.3.0	eigen	4 KB

Рисунок 3 – Установка пакета TensorFlow

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

```
executing transaction: done
(lab_2_14) PS C:\Users\Honor\desktop\R\lab_2_14> conda env export > environment.yml
(lab_2_14) PS C:\Users\Honor\desktop\R\lab_2_14> pip freeze > requirements.txt
(lab_2_14) PS C:\Users\Honor\desktop\R\lab_2_14> █
```

Рисунок 4 – Формирование файлов requirements.txt и environment.yml



```
requirements.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
absl-py @ file:///C:/ci/absl-py_1623867339542/work
aiohttp @ file:///C:/ci/aiohttp_1637857271452/work
aiosignal @ file:///tmp/build/80754af9/aiosignal_1637843061372/work
astor==0.8.1
astunparse==1.6.3
async-timeout @ file:///tmp/build/80754af9/async-timeout_1637851218186/work
attrs @ file:///tmp/build/80754af9/attrs_1620827162558/work
blinker==1.4
bottleneck==1.3.2
brotli==0.7.0
cachetools @ file:///tmp/build/80754af9/cachetools_1619597386817/work
certifi==2021.10.8
cffi @ file:///C:/ci/cffi_1636542074879/work
charset-normalizer @ file:///tmp/build/80754af9/charset-normalizer_1630003229654/work
click==8.0.3
cryptography @ file:///C:/ci/cryptography_1633520552480/work
frozenlist @ file:///C:/ci/frozenlist_1637767289747/work
gast @ file:///Users/ktietz/demo/mc3/conda-bld/gast_1628588903283/work
google-auth @ file:///tmp/build/80754af9/google-auth_1626320605116/work
google-auth-oauthlib @ file:///tmp/build/80754af9/google-auth-oauthlib_1617120569401/work
google-pasta @ file:///Users/ktietz/demo/mc3/conda-bld/google-pasta_1630577991354/work
grpcio @ file:///C:/ci/grpcio_1637590968244/work
h5py==2.10.0
idna @ file:///tmp/build/80754af9/idna_1637925883363/work
importlib-metadata @ file:///C:/ci/importlib-metadata_1638543108096/work
Keras-Applications @ file:///tmp/build/80754af9/keras-applications_1594366238411/work
Keras-Preprocessing @ file:///tmp/build/80754af9/keras-preprocessing_1612283640596/work
```

Рисунок 5 – Текст файла requirements.txt



```
environment.yml - Блокнот
Файл  Правка  Формат  Вид  Справка
name: lab_2_14
channels:
  - defaults
dependencies:
  - _tfflow_select=2.3.0=eigen
  - absl-py=0.13.0=py38haa95532_0
  - aiohttp=3.8.1=py38h2bbff1b_0
  - aiosignal=1.2.0=pyhd3eb1b0_0
  - astor=0.8.1=py38haa95532_0
  - astunparse=1.6.3=py_0
  - async-timeout=4.0.1=pyhd3eb1b0_0
  - attrs=21.2.0=pyhd3eb1b0_0
  - blas=1.0=mkl
  - blinker=1.4=py38haa95532_0
  - bottleneck=1.3.2=py38h2a96729_1
  - brotli=0.7.0=py38h2bbff1b_1003
  - ca-certificates=2021.10.26=haa95532_2
  - cachetools=4.2.2=pyhd3eb1b0_0
  - certifi=2021.10.8=py38haa95532_0
  - cffi=1.15.0=py38h2bbff1b_0
  - charset-normalizer=2.0.4=pyhd3eb1b0_0
  - click=8.0.3=pyhd3eb1b0_0
  - cryptography=3.4.8=py38h71e12ea_0
  - dataclasses=0.8=pyh6d0b6a4_7
  - frozenlist=1.2.0=py38h2bbff1b_0
  - gast=0.4.0=pyhd3eb1b0_0
  - google-auth=1.33.0=pyhd3eb1b0_0
  - google-auth-oauthlib=0.4.4=pyhd3eb1b0_0
  - google-pasta=0.2.0=pyhd3eb1b0_0
  - grpcio=1.42.0=py38hc60d5dd_0
  - h5py=2.10.0=py38h5e291fa_0
```

Рисунок 6 – Текст содержимое environment.yml

5. Проверил программы через flake8.

```
(base) PS C:\Users\Honor> conda activate tools
(tools) PS C:\Users\Honor> cd desktop/r/lab_2_14
(tools) PS C:\Users\Honor\desktop\r\lab_2_14> flake8
(tools) PS C:\Users\Honor\desktop\r\lab_2_14>
```

Рисунок 7 – Проверка flake8

Ответы на 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`?

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