

**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»**

Выполнил студент группы ИВТ-б-о-20-1

Ищенко Т.С. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверила Воронкин Р.А. _____

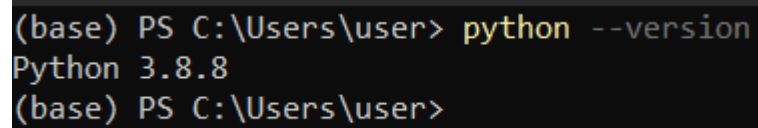
(подпись)

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. Проверил версию python



```
(base) PS C:\Users\user> python --version
Python 3.8.8
(base) PS C:\Users\user>
```

Рисунок 1 – Результат проверки версии

2. Произвёл создание нового виртуального окружения

```
(base) PS C:\Users\user> conda create -n lab-2.14 python=3.8
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\user\.conda\envs\lab-2.14

added / updated specs:
  - python=3.8

The following NEW packages will be INSTALLED:

ca-certificates      pkgs/main/win-64::ca-certificates-2021.10.26-haa95532_2
certifi              pkgs/main/win-64::certifi-2021.10.8-py38haa95532_0
openssl              pkgs/main/win-64::openssl-1.1.1l-h2bbff1b_0
pip                  pkgs/main/win-64::pip-21.0.1-py38haa95532_0
python               pkgs/main/win-64::python-3.8.12-h6244533_0
setuptools           pkgs/main/win-64::setuptools-58.0.4-py38haa95532_0
sqlite               pkgs/main/win-64::sqlite-3.36.0-h2bbff1b_0
vc                   pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime       pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel                pkgs/main/noarch::wheel-0.37.0-pyhd3eb1b0_1
wincertstore         pkgs/main/win-64::wincertstore-0.2-py38haa95532_2

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

Рисунок 2 – Процесс создания нового виртуального окружения

3. Активировал ново созданное виртуальное окружение

```
(base) PS C:\Users\user> conda activate lab-2.14
(lab-2.14) PS C:\Users\user>
```

Рисунок 3 – Процесс активации ново созданного виртуального окружения

4. Установил менеджер пакетов pip

```
(lab-2.14) PS C:\Users\user> 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.

(lab-2.14) PS C:\Users\user>
```

Рисунок 4 – Процесс установки менеджера пакетов pip

5. Установил пакет Numpy

```

lab-2.14) PS C:\Users\user> conda install NumPy
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\user\.conda\envs\lab-2.14

added / updated specs:
- numpy

The following packages will be downloaded:


```

package	build	
intel-openmp-2021.4.0	haa95532_3556	2.2 MB
mkl-2021.4.0	haa95532_640	114.9 MB
mkl-service-2.4.0	py38h2bbff1b_0	51 KB
mkl_fft-1.3.1	py38h277e83a_0	139 KB
mkl_random-1.2.2	py38hf11a4ad_0	225 KB
numpy-1.21.2	py38hfca59bb_0	24 KB
numpy-base-1.21.2	py38h0829f74_0	4.4 MB
six-1.16.0	pyhd3eb1b0_0	18 KB
Total:		121.9 MB

```

The following NEW packages will be INSTALLED:

blas                pkgs/main/win-64::blas-1.0-mkl
intel-openmp        pkgs/main/win-64::intel-openmp-2021.4.0-haa95532_3556
mkl                 pkgs/main/win-64::mkl-2021.4.0-haa95532_640
mkl-service         pkgs/main/win-64::mkl-service-2.4.0-py38h2bbff1b_0
mkl_fft             pkgs/main/win-64::mkl_fft-1.3.1-py38h277e83a_0
mkl_random          pkgs/main/win-64::mkl_random-1.2.2-py38hf11a4ad_0

```

Рисунок 5 – Процесс установки пакета NumPy

6. Установил пакет Pandas

```
(lab-2.14) PS C:\Users\user> conda install Pandas
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\user\.conda\envs\lab-2.14

added / updated specs:
- pandas

The following packages will be downloaded:

package | build | size
-----|-----|-----
pandas-1.3.4 | py38h6214cd6_0 | 8.6 MB
python-dateutil-2.8.2 | pyhd3eb1b0_0 | 233 KB
pytz-2021.3 | pyhd3eb1b0_0 | 171 KB
-----|-----|-----
Total: | | 9.0 MB

The following NEW packages will be INSTALLED:

bottleneck | pkgs/main/win-64::bottleneck-1.3.2-py38h2a96729_1
numexpr | pkgs/main/win-64::numexpr-2.7.3-py38hb8b31ca_1
pandas | pkgs/main/win-64::pandas-1.3.4-py38h6214cd6_0
python-dateutil | pkgs/main/noarch::python-dateutil-2.8.2-pyhd3eb1b0_0
pytz | pkgs/main/noarch::pytz-2021.3-pyhd3eb1b0_0
```

Рисунок 6 – Процесс установки пакета Pandas

7. Установил пакет SciPy

```
(lab-2.14) PS C:\Users\user> conda install 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

## Package Plan ##

environment location: C:\Users\user\.conda\envs\lab-2.14

added / updated specs:
- scipy

The following packages will be downloaded:

package | build | size
-----|-----|-----
scipy-1.7.1 | py38hbe87c03_2 | 13.8 MB
-----|-----|-----
Total: | | 13.8 MB

The following NEW packages will be INSTALLED:

icc_rt | pkgs/main/win-64::icc_rt-2019.0.0-h0cc432a_1
scipy | pkgs/main/win-64::scipy-1.7.1-py38hbe87c03_2
```

Рисунок 7 – Процесс установки пакета SciPy

8. Попытался установить пакет Tensorflow и столкнулся с ошибками.

```
(lab-2.14) PS C:\Users\user> 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: done
```

Рисунок 8 – Попытка установить Tensorflow

9. Установил пакет Tensorflow с помощью пакетного менеджера pip

```
(lab-2.14) PS C:\Users\user> pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-2.7.0-cp38-cp38-win_amd64.whl (430.8 MB)
    Requirement already satisfied: wheel<1.0, >=0.32.0 in c:\users\user\.conda\envs\lab-2.14\lib\site-packages (from tensorflow) (0.37.0)
Collecting opt-einsum<2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
    Requirement already satisfied: numpy>=1.14.5 in c:\users\user\.conda\envs\lab-2.14\lib\site-packages (from tensorflow) (1.21.2)
Collecting typing-extensions<3.6.6
  Downloading typing_extensions-4.0.0-py3-none-any.whl (22 kB)
Requirement already satisfied: numpy>=1.14.5 in c:\users\user\.conda\envs\lab-2.14\lib\site-packages (from tensorflow) (1.21.2)
Collecting absl-py<0.4.0
  Downloading absl_py-1.0.0-py3-none-any.whl (126 kB)
Collecting grpcio<2.0, >=1.24.1
  Downloading grpcio-1.41.1-cp38-cp38-win_amd64.whl (3.2 MB)
Collecting google-pasta<0.1.1
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
Collecting libclang<9.0.1
  Downloading libclang-12.0.0-py2.py3-none-win_amd64.whl (13.1 MB)
Requirement already satisfied: six>=1.12.0 in c:\users\user\.conda\envs\lab-2.14\lib\site-packages (from tensorflow) (1.16.0)
Collecting keras<2.8, >=2.7.0rc0
  Downloading keras-2.7.0-py2.py3-none-any.whl (1.3 MB)
Collecting flatbuffers<3.0, >=1.12
  Downloading flatbuffers-2.0-py2.py3-none-any.whl (26 kB)
Collecting keras-preprocessing>=1.1.1
  Downloading Keras-Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
Collecting gast<0.5.0, >=0.2.1
  Downloading gast-0.4.0-py3-none-any.whl (9.8 kB)
Collecting termcolor>=1.1.0
  Downloading termcolor-1.1.0.tar.gz (3.9 kB)
Collecting protobuf<3.19.1-cp38-cp38-win_amd64.whl (895 kB)
  Downloading h5py-3.6.0-cp38-cp38-win_amd64.whl (2.8 MB)
Collecting tensorboard<2.7.0-py3-none-any.whl (5.8 MB)
```

Рисунок 9 – Процесс установки Tensorflow

10. Экспортировал файлы environment.yml и requirements.txt

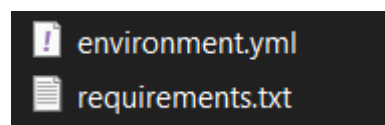


Рисунок 10 – Результат экспорта файлов

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 разворачивании 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=3.8`

20. Как активировать и установить пакеты в виртуальное окружение `conda`? – Чтобы установить пакеты, необходимо воспользоваться командой: –
`conda install A` 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 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 приобрел навыки по работе с менеджером пакетов pip и виртуальными окружениями с помощью языка программирования Python версии 3.x