

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

Кафедра инфокоммуникаций  
Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.14

Дисциплина: «Программирование на Python»

Тема: «Установка  
пакетов в Python. Виртуальные  
окружения»

Выполнил: студент 1 курса  
группы ИВТ-б-о-21-1  
Толубаев Рамиль Ахметович

Ставрополь 2022

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

**Ход работы:**

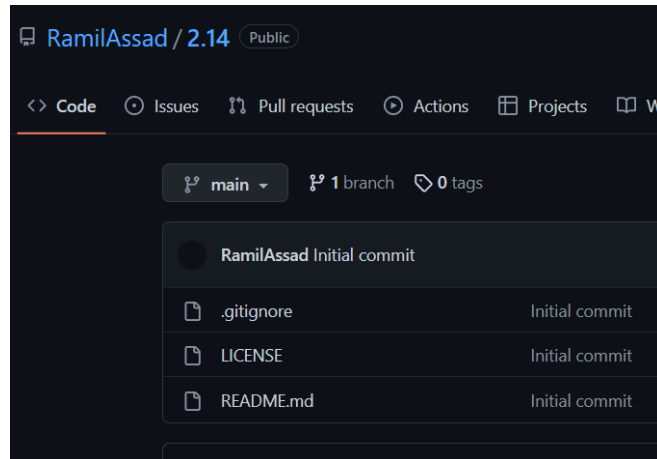


Рисунок 1 - Создание репозитория

Создайте виртуальное окружение Anaconda с именем репозитория.

```
Anaconda Prompt (anaconda3) - conda create -n 2.14 python=3.10

(base) C:\Users\Asus>conda create -n 2.14 python=3.10
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
current version: 22.9.0
latest version: 22.11.1

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\Users\Asus\anaconda3\envs\2.14

added / updated specs:
- python=3.10

The following packages will be downloaded:

package | build | size
-----|-----|-----
ca-certificates-2022.10.11 | haa95532_0 | 125 KB
certifi-2022.12.7 | py310haa95532_0 | 149 KB
libffi-3.4.2 | hd77b12b_6 | 109 KB
openssl-1.1.1s | h2bbff1b_0 | 5.5 MB
pip-22.3.1 | py310haa95532_0 | 2.8 MB
python-3.10.8 | h966fe2a_1 | 15.8 MB
setuptools-65.5.0 | py310haa95532_0 | 1.2 MB
sqlite-3.40.0 | h2bbff1b_0 | 891 KB
tzdata-2022g | h04d1e81_0 | 114 KB
wincertstore-0.2 | py310haa95532_2 | 15 KB
xz-5.2.8 | h8cc25b3_0 | 205 KB
zlib-1.2.13 | h8cc25b3_0 | 113 KB
-----|-----|-----
Total: | | 26.9 MB

The following NEW packages will be INSTALLED:
```

Рисунок 2. Создание виртуального пространства

```
(base) C:\Users\Asus>conda activate 2.14

(2.14) C:\Users\Asus>_
```

Рисунок 3 - Активация виртуального пространства

Установите в виртуальное окружение следующие пакеты: pip, NumPy, Pandas, SciPy.

```
(2.14) C:\Users\Asus>pip --version
pip 22.3.1 from C:\Users\Asus\anaconda3\envs\2.14\lib\site-packages\pip (python 3.10)

(2.14) C:\Users\Asus>
```

Рисунок 4 – Проверка версии pip

```
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> python3 -m venv 2.14
Python
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> python3 -m venv env
Python
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> python -m venv env
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> |
```

Рисунок 5 – Установка виртуального окружения venv

```
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> .\env\scripts\activate
(env) PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> |
```

Рисунок 6 – Активация

```
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> .\env\scripts\activate
(env) PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> pip install black
Collecting black
  Downloading black-22.12.0-cp310-cp310-win_amd64.whl (1.2 MB)
    1.2/1.2 MB 3.6 MB/s eta 0:00:00
Collecting platformdirs>=2
  Downloading platformdirs-2.6.0-py3-none-any.whl (14 kB)
Collecting click>=8.0.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
    96.6/96.6 kB 7 eta 0:00:00
Collecting pathspec>=0.9.0
  Downloading pathspec-0.10.3-py3-none-any.whl (29 kB)
Collecting tomli>=1.1.0
  Downloading tomli-2.0.1-py3-none-any.whl (12 kB)
Collecting mypy_extensions>=0.4.3
  Downloading mypy_extensions-0.4.3-py2.py3-none-any.whl (4.5 kB)
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: mypy_extensions, tomli, platformdirs, pathspec, colorama, click, black
Successfully installed black-22.12.0 click-8.1.3 colorama-0.4.6 mypy_extensions-0.4.3 pathspec-0.10.3 platformdirs-2.6.0 tomli-2.0.1
WARNING: You are using pip version 22.0.4; however, version 22.3.1 is available.
You should consider upgrading via the 'C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14\env\Scripts\python.exe -m pip install --upgrade pip' command.
(env) PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> |
```

```
Windows PowerShell
+ CategoryInfo          : ObjectNotFound: (m:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

(env) PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> python.exe -m pip install --upgrade pip

Usage:
C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14\env\scripts\python.exe -m pip install [options] <requirement specifier> [package-index-options] ...
C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14\env\scripts\python.exe -m pip install [options] -r <requirements file> [package-index-options] ...
C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14\env\scripts\python.exe -m pip install [options] [-e] <vcs project url> ...
C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14\env\scripts\python.exe -m pip install [options] [-e] <local project path> ...
C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14\env\scripts\python.exe -m pip install [options] <archive url/path> ...

(env) PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> pip install flake8
Collecting flake8
  Downloading flake8-6.0.0-py2.py3-none-any.whl (57 kB)
    |#####| 57.8/57.8 KB 755.2 kB/s eta 0:00:00
Collecting mccabe<0.8.0,>=0.7.0
  Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Collecting pyflakes<3.1.0,>=3.0.0
  Downloading pyflakes-3.0.1-py2.py3-none-any.whl (62 kB)
    |#####| 62.8/62.8 KB 3.3 MB/s eta 0:00:00
Collecting pycodestyle<2.11.0,>=2.10.0
  Downloading pycodestyle-2.10.0-py2.py3-none-any.whl (41 kB)
    |#####| 41.3/41.3 KB 1.9 MB/s eta 0:00:00
Installing collected packages: pyflakes, pycodestyle, mccabe, flake8
Successfully installed flake8-6.0.0 mccabe-0.7.0 pycodestyle-2.10.0 pyflakes-3.0.1
WARNING: You are using pip version 22.0.4; however, version 22.3.1 is available.
You should consider upgrading via the 'C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14\env\Scripts\python.exe -m pip install --upgrade pip' command.
(env) PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> |
```

Рисунок 7,8,9 – Установка пакетов black

```
(env) PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> deactivate
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> |
```

Рисунок 10 – Деактивация

```
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> python -m pip install virtualenv
Collecting virtualenv
  WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ConnectTimeoutError(<spip._vendor.urllib3.connection.HTTPSConnection object at 0x00000258ABD1EB60>, 'Connection to files.pythonhosted.org timed out. (connect timeout=15)')': /packages/18/a2/7931d40ecb02b5236a34ac53770f2f6931e3082b7a7d4fe915d892d749d6/virtualenv-20.17.1-py3-none-any.whl
  Downloading virtualenv-20.17.1-py3-none-any.whl (8.8 MB)
    |#####| 8.8/8.8 MB 6.6 MB/s eta 0:00:00
Collecting distlib<1,>=0.3.6
  Downloading distlib-0.3.6-py2.py3-none-any.whl (468 kB)
    |#####| 468.5/468.5 KB 7.3 MB/s eta 0:00:00
Collecting platformdirs<3,>=2.4
  Using cached platformdirs-2.6.0-py3-none-any.whl (14 kB)
Collecting filelock<4,>=3.4.1
  Downloading filelock-3.8.2-py3-none-any.whl (10 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.6 filelock-3.8.2 platformdirs-2.6.0 virtualenv-20.17.1
WARNING: You are using pip version 22.0.4; however, version 22.3.1 is available.
You should consider upgrading via the 'C:\Users\Asus\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' command.
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> |
```

Рисунок 11 – Установка виртуального окружения virtualenv

```
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> virtualenv -p python env
created virtual environment CPython3.10.4.final.0-64 in 3267ms
  creator CPython3Windows(dest=C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14\env, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\Asus\AppData\Local\pypa\virtualenv)
    added seed packages: black==22.12.0, click==8.1.3, colorama==0.4.6, flake8==6.0.0, mccabe==0.7.0, mypy_extensions==0.4.3, pathspec==0.10.3, pip==22.3.1, platformdirs==2.6.0, pycodestyle==2.10.0, pyflakes==3.0.1, setuptools==65.6.3, tomli==2.0.1, wheel==0.38.4
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
PS C:\Users\Asus\OneDrive\Рабочий стол\Питон\2.14> |
```

Рисунок 12 – Установка 2 виртуального окружения virtualenv

```
(base) PS C:\Users\Asus> mkdir %%2.14%%%

Каталог: C:\Users\Asus

Mode                LastWriteTime         Length Name
----                -
d-----          27.12.2022   9:00             %%2.14%%%

(base) PS C:\Users\Asus> _
```

Рисунок 13 – Чистое виртуальное окружение

Контр. вопросы и ответы на них:

**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 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 глобального 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 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?

Создание файла: conda env export > environment.yml

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

