

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

«Установка пакетов в Python. Виртуальные окружения»

**Отчет по лабораторной работе № 2.14
по дисциплине «Программирование на Python»**

Выполнил студент группы ИВТ-б-о-21-1

Сумин Никита Сергеевич.

« » _____ 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

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

Порядок выполнения работы:

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two main sections: 'Owner' and 'Repository name'. The 'Owner' is set to 'BrandooDi' and the 'Repository name' is 'mywork2.14', which is marked as valid with a green checkmark. A note below these fields suggests that repository names should be short and memorable, and provides a link to 'cuddly-goggles?'. The 'Description' field is optional and currently empty. Below the description field, there are two radio buttons for visibility: 'Public' (selected) and 'Private'. The 'Public' option is described as allowing anyone on the internet to see the repository, while the 'Private' option allows only the owner to see and commit to it. The next section is 'Initialize this repository with:', which includes a checkbox for 'Add a README file' (checked) and a note about writing a long description for the project. Below this is the 'Add .gitignore' section, which allows choosing a template for files not to track, with 'Python' selected. The 'Choose a license' section follows, with a note about what a license does and 'MIT License' selected. At the bottom, it states that the default branch will be set to 'main' and provides a link to change the default name in settings. A final note indicates that a public repository is being created in a personal account. A green 'Create repository' button is at the bottom.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

BrandooDi / mywork2.14 ✓

Great repository names are short and memorable. Need inspiration? How about [cuddly-goggles?](#)

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▼

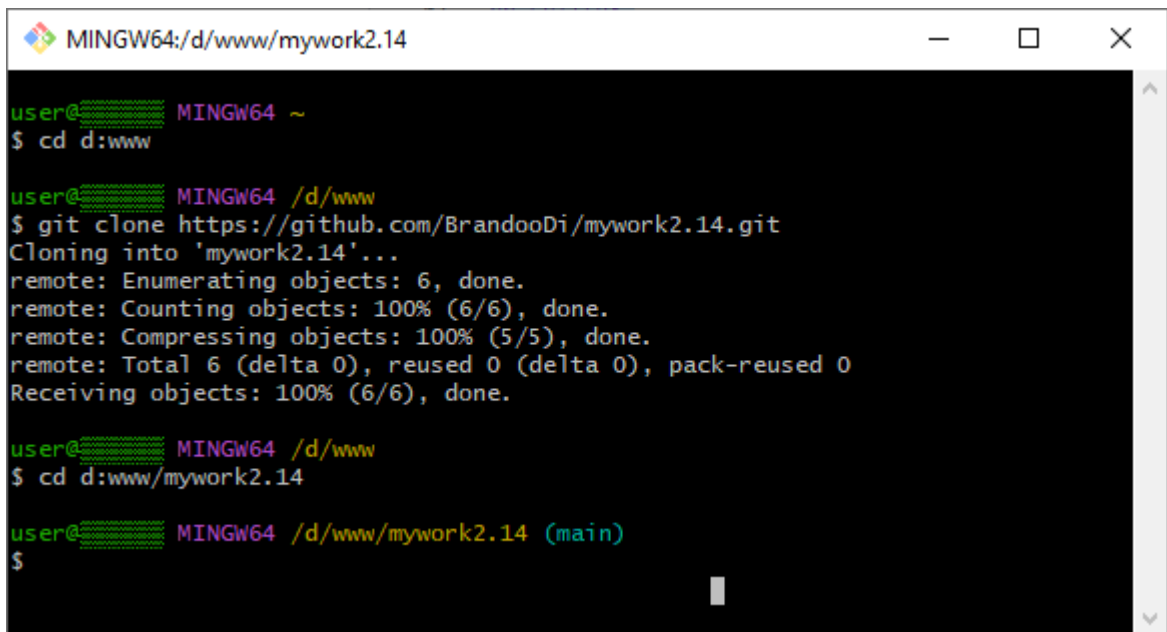
This will set `main` as the default branch. Change the default name in your [settings](#).

❗ You are creating a public repository in your personal account.

Create repository

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.

A screenshot of a terminal window titled 'MINGW64:/d/www/mywork2.14'. The terminal shows a user cloning a repository from GitHub. The commands and output are as follows:

```
user@MINGW64 ~  
$ cd d:www  
  
user@MINGW64 /d/www  
$ git clone https://github.com/BrandooDi/mywork2.14.git  
Cloning into 'mywork2.14'...  
remote: Enumerating objects: 6, done.  
remote: Counting objects: 100% (6/6), done.  
remote: Compressing objects: 100% (5/5), done.  
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0  
Receiving objects: 100% (6/6), done.  
  
user@MINGW64 /d/www  
$ cd d:www/mywork2.14  
  
user@MINGW64 /d/www/mywork2.14 (main)  
$
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

A screenshot of a terminal window showing a user switching to a new branch. The commands and output are as follows:

```
Asus@LAPTOP-09A994CG MINGW64 /c/my projects/3/lw_2.14 (main)  
$ git checkout -b develop  
Switched to a new branch 'develop'  
  
Asus@LAPTOP-09A994CG MINGW64 /c/my projects/3/lw_2.14 (develop)  
$ |
```

Рисунок 3 - Ветвление по модели git-flow

4. Создайте виртуальное окружение Anaconda с именем репозитория.

```
Выбрав C:\Windows\System32\cmd.exe - conda create -n mywork2.14 python=3.10
D:\www>conda create -n mywork2.14 python=3.10
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 22.11.1

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\user\anaconda3\envs\mywork2.14

  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  |
| tk-8.6.12                  | h2bbff1b_0      | 3.1 MB  |
| 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:                     |                 | 30.0 MB |



The following NEW packages will be INSTALLED:



| package         | path                                                    |
|-----------------|---------------------------------------------------------|
| bzip2           | pkgs/main/win-64::bzip2-1.0.8-he774522_0                |
| ca-certificates | pkgs/main/win-64::ca-certificates-2022.10.11-haa95532_0 |
| certifi         | pkgs/main/win-64::certifi-2022.12.7-py310haa95532_0     |
| libffi          | pkgs/main/win-64::libffi-3.4.2-hd77b12b_6               |
| openssl         | pkgs/main/win-64::openssl-1.1.1s-h2bbff1b_0             |
| pip             | pkgs/main/win-64::pip-22.3.1-py310haa95532_0            |
| python          | pkgs/main/win-64::python-3.10.8-h966fe2a_1              |
| setuptools      | pkgs/main/win-64::setuptools-65.5.0-py310haa95532_0     |
| sqlite          | pkgs/main/win-64::sqlite-3.40.0-h2bbff1b_0              |
| tk              | pkgs/main/win-64::tk-8.6.12-h2bbff1b_0                  |
| tzdata          | pkgs/main/noarch::tzdata-2022g-h04d1e81_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-0.37.1-pyhd3eb1b0_0             |
| wincertstore    | pkgs/main/win-64::wincertstore-0.2-py310haa95532_2      |
| xz              | pkgs/main/win-64::xz-5.2.8-h8cc25b3_0                   |
| zlib            | pkgs/main/win-64::zlib-1.2.13-h8cc25b3_0                |


```

Рисунок 4 - Создание виртуального пространства

```
Выбрав C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2364]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

D:\www\mywork2.14>conda activate mywork2.14

(mywork2.14) D:\www\mywork2.14>
```

Рисунок 5 - Активация виртуального пространства

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

```
Выбрать C:\Windows\System32\cmd.exe - conda update -n base -c defaults conda

(mywork2.14) D:\www\mywork2.14>pip install black
Collecting black
  Using cached black-22.12.0-cp310-cp310-win_amd64.whl (1.2 MB)
Collecting pathspec>=0.9.0
  Using cached pathspec-0.10.3-py3-none-any.whl (29 kB)
Collecting mypy-extensions>=0.4.3
  Using cached mypy_extensions-0.4.3-py2.py3-none-any.whl (4.5 kB)
Collecting tomli>=1.1.0
  Using cached tomli-2.0.1-py3-none-any.whl (12 kB)
Collecting click>=8.0.0
  Using cached click-8.1.3-py3-none-any.whl (96 kB)
Collecting platformdirs>=2
  Downloading platformdirs-2.6.2-py3-none-any.whl (14 kB)
Collecting colorama
  Using cached 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-2.6.2 tomli-2.0.1

(mywork2.14) D:\www\mywork2.14>pip install NumPy
Collecting NumPy
  Downloading numpy-1.24.1-cp310-cp310-win_amd64.whl (14.8 MB)
----- 14.8/14.8 MB 11.1 MB/s eta 0:00:00
Installing collected packages: NumPy
Successfully installed NumPy-1.24.1

(mywork2.14) D:\www\mywork2.14>pip install SciPy
Collecting SciPy
  Downloading scipy-1.10.0-cp310-cp310-win_amd64.whl (42.5 MB)
----- 42.5/42.5 MB 9.6 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.27.0,>=1.19.5 in c:\users\user\anaconda3\envs\mywork2.14\lib\site-packages (from SciPy) (1.24.1)
Installing collected packages: SciPy
Successfully installed SciPy-1.10.0

(mywork2.14) D:\www\mywork2.14>
```

Рисунок 6 - Установка пакетов

6. Попробуйте установить менеджером пакетов conda пакет TensorFlow.

```
Выбрать C:\Windows\System32\cmd.exe - conda update -n base -c defaults conda - conda install tensorflow

(mywork2.14) D:\www\mywork2.14>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 repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\Users\user\anaconda3\envs\mywork2.14

added / updated specs:
- tensorflow

The following packages will be downloaded:

package | build | size
-----|-----|-----
_tfselect-2.3.0 | mk1 | 3 KB
absl-py-1.3.0 | py310haa95532_0 | 172 KB
aiohttp-3.8.3 | py310h2bbff1b_0 | 418 KB
astunparse-1.6.3 | py_0 | 17 KB
async-timeout-4.0.2 | py310haa95532_0 | 12 KB
attrs-22.1.0 | py310haa95532_0 | 85 KB
blinker-1.4 | py310haa95532_0 | 22 KB
brotli-1.0.7 | py310h2bbff1b_1002 | 335 KB
cffi-1.15.1 | py310h2bbff1b_3 | 239 KB
click-8.0.4 | py310haa95532_0 | 157 KB
colorama-0.4.6 | py310haa95532_0 | 32 KB
cryptography-38.0.1 | py310h21b164f_0 | 996 KB
fftw-3.3.9 | h2bbff1b_1 | 672 KB
flatbuffers-2.0.0 | h6c2663c_0 | 1.4 MB
flit-core-3.6.0 | pyhd3eb1b0_0 | 42 KB
frozenlist-1.3.3 | py310h2bbff1b_0 | 40 KB
gast-0.4.0 | pyhd3eb1b0_0 | 13 KB
google-auth-2.6.0 | pyhd3eb1b0_0 | 83 KB
google-auth-oauthlib-0.4.4 | pyhd3eb1b0_0 | 18 KB
google-pasta-0.2.0 | pyhd3eb1b0_0 | 46 KB
grpcio-1.42.0 | py310hc60d5dd_0 | 1.7 MB
h5py-3.7.0 | py310hfc34f40_0 | 822 KB
hdf5-1.10.6 | h1756f20_1 | 7.9 MB
icc_rt-2022.1.0 | h6049295_2 | 6.5 MB
idna-3.4 | py310haa95532_0 | 97 KB
keras-2.10.0 | py310haa95532_0 | 1.6 MB
keras-preprocessing-1.1.2 | pyhd3eb1b0_0 | 35 KB
libcurl-7.86.0 | h86230a5_0 | 319 KB
libprotobuf-3.20.1 | h23ce68f_0 | 2.0 MB
markdown-3.4.1 | py310haa95532_0 | 149 KB
markupsafe-2.1.1 | py310h2bbff1b_0 | 26 KB
mkl-service-2.4.0 | py310h2bbff1b_0 | 48 KB
mkl_fft-1.3.1 | py310ha0764ea_0 | 136 KB
mkl_random-1.2.2 | py310h4ed8f06_0 | 221 KB
multidict-6.0.2 | py310h2bbff1b_0 | 46 KB
numpy-1.23.5 | py310h60c9a35_0 | 11 KB
numpy-base-1.23.5 | py310h04254f7_0 | 6.0 MB
oauthlib-3.2.1 | py310haa95532_0 | 195 KB
opt_einsum-3.3.0 | pyhd3eb1b0_1 | 57 KB
packaging-22.0 | py310haa95532_0 | 68 KB
protobuf-3.20.1 | py310hd77b12b_0 | 233 KB
pyjwt-2.4.0 | py310haa95532_0 | 38 KB
pyopenssl-22.0.0 | pyhd3eb1b0_0 | 50 KB
```

Рисунок 7 - Установка Tensorflow

Пакет Tensorflow успешно установился.

7. Сформируйте файлы requirements.txt и environment.yml.

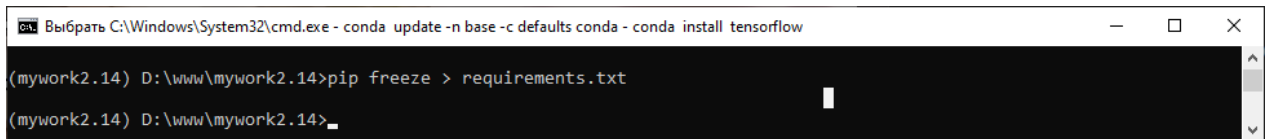
Проанализируйте содержимое этих файлов.

```
Выбрать C:\Windows\System32\cmd.exe - conda update -n base -c defaults conda - conda install tensorflow

(mywork2.14) D:\www\mywork2.14>conda env export > environment.yml

(mywork2.14) D:\www\mywork2.14>
```

Рисунок 8 - Формирование файла environmental.yml



```
Выбрать C:\Windows\System32\cmd.exe - conda update -n base -c defaults conda - conda install tensorflow
(mywork2.14) D:\www\mywork2.14>pip freeze > requirements.txt
(mywork2.14) D:\www\mywork2.14>_
```

Рисунок 9 - Формирование файла requirements.txt

Все пакеты, которые были установлены перед выполнением команды и предположительно использованы в каком-либо проекте, будут перечислены в файлах с именем «requirements.txt» и «environmental.yml».

Контрольные вопросы:

1. Каким способом можно установить пакет Python, не входящий в стандартную библиотеку?

Необходимо узнать, какой пакет содержит функционал, который вам необходим, найти его, скачать, разместить в нужном каталоге и начать использовать.

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?

Будем считать, что Python у вас уже установлен, теперь необходимо установить pip. Для того, чтобы это сделать, скачайте скрипт get-pip.py

```
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

и выполните его. `$ python get-pip.py`

3. Откуда менеджер пакетов pip по умолчанию устанавливает пакеты?

По умолчанию pip устанавливает пакеты из Python Package Index (PyPI) – это репозиторий, открытый для всех Python разработчиков, в нем вы можете найти пакеты для решения практически любых задач.

4. Как установить последнюю версию пакета с помощью pip?

```
$ pip install ProjectName
```

5. Как установить заданную версию пакета с помощью pip?

```
$ pip install ProjectName==3.2
```

6. Как установить пакет из git репозитория (в том числе GitHub)

с помощью pip?

```
$ pip install -e git+https://gitrepo.com/ProjectName.git
```

7. Как установить пакет из локальной директории с помощью pip?

```
$ pip install ./dist/ProjectName.tar.gz
```

8. Как удалить установленный пакет с помощью pip?

```
$ pip uninstall ProjectName
```

9. Как обновить установленный пакет с помощью pip?

```
$ pip install --upgrade ProjectName
```

10. Как отобразить список установленных пакетов с помощью pip?

```
$ pip list
```

11. Каковы причины появления виртуальных окружений в языке Python?

В системе для интерпретатора Python может быть установлена глобально только одна версия пакета. Это порождает ряд проблем.

1. Проблема обратной совместимости

2. Проблема коллективной разработки

Если вы уже сталкивались с этой проблемой, то уже задумались, что для каждого проекта нужна своя "песочница", которая изолирует зависимости. Такая "песочница" придумана и называется "виртуальным окружением" или "виртуальной средой".

12. Каковы основные этапы работы с виртуальными окружениями?

1. Создаём через утилиту новое виртуальное окружение в отдельной папке для выбранной версии интерпретатора Python.
2. Активируем ранее созданное виртуального окружения для работы.
3. Работаем в виртуальном окружении, а именно управляем пакетами используя `pip` и запускаем выполнение кода.
4. Деактивируем после окончания работы виртуальное окружение.
5. Удаляем папку с виртуальным окружением, если оно нам больше не нужно.

13. Как осуществляется работа с виртуальными окружениями с помощью `venv`?

Создание виртуального окружения

Для создания виртуального окружения достаточно дать команду в формате:

```
python3 -m venv <путь к папке виртуального окружения>
```

Чтобы активировать виртуальное окружение под Windows команда выглядит иначе:

```
> env\\Scripts\\activate
```

Чтобы переключиться с одного окружения на другое нам нужно выполнить команду деактивации команду активации другого виртуального окружения, например, так:

```
> C:\\Python38\\python -m venv env
```

14. Как осуществляется работа с виртуальными окружениями с помощью `virtualenv`?

Создание виртуального окружения с утилитой `virtualenv` отличается от стандартного. Например, создание в текущей папке виртуального окружения для интерпретатора доступного через команду `python3` с названием папки окружения `env`:

```
virtualenv -p python3 env
```

Активация и деактивация такая же, как у стандартной утилиты Python.

```
> env\Scripts\activate
```

```
(env) > deactivate
```

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?

Все пакеты, которые вы установили перед выполнением команды и предположительно использовали в каком-либо проекте, будут перечислены в файле с именем «`requirements.txt`». Кроме того, будут указаны их точные версии.

```
pip freeze > requirements.txt
```

```
install -r requirements.txt
```

17. В чем преимущества пакетного менеджера `conda` по сравнению с пакетным менеджером `pip`?

Основная проблема заключается в том, что `pip` , `easy_install` и `virtualenv` ориентированы на Python. Эти инструменты игнорируют библиотеки

зависимостей, реализованные с использованием других языков. Например, XSLT , HDF5 , MKL и другие, которые не имеют setup.py в исходном коде и не устанавливают файлы в директорию site-packages .

Conda же 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?

Anaconda и Miniconda.

19. Как создать виртуальное окружение conda? conda create -n \$PROJ_NAME python=3.7

20. Как активировать и установить пакеты в виртуальное окружение conda?

conda activate %PROJ_NAME%

conda install django, pandas

21. Как деактивировать и удалить виртуальное окружение conda?

conda deactivate

Если вы хотите удалить только что созданное окружение, выполните:

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.

conda env export > environment.yml

23. Как создать виртуальное окружение conda с помощью файла environment.yml ?

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.

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 какихлибо 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 при написании программ с помощью языка программирования Python версии 3.x.