

**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 2 курса

группы ИВТ-б-о-21-1

Хашиев Адам Мухарбекович

Ставрополь 2022

### **Выполнение работы:**

1. Создал репозиторий в GitHub «rep 2.6» в который добавил .gitignore, который дополнил правилами для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.

Owner \*

AdamKh ▾


/

Repository name \*


2.14\_9 ✓

Great repository names are available. 2.14\_9 is available. [Need inspiration? How about...](#)

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 ▾

 You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 Создание репозитория

```
C:\Users\adamkh\Desktop\3sem\Python\2.14>git clone https://github.com/AdamKh/2.14_9.git
Cloning into '2.14_9'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
Receiving objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
```

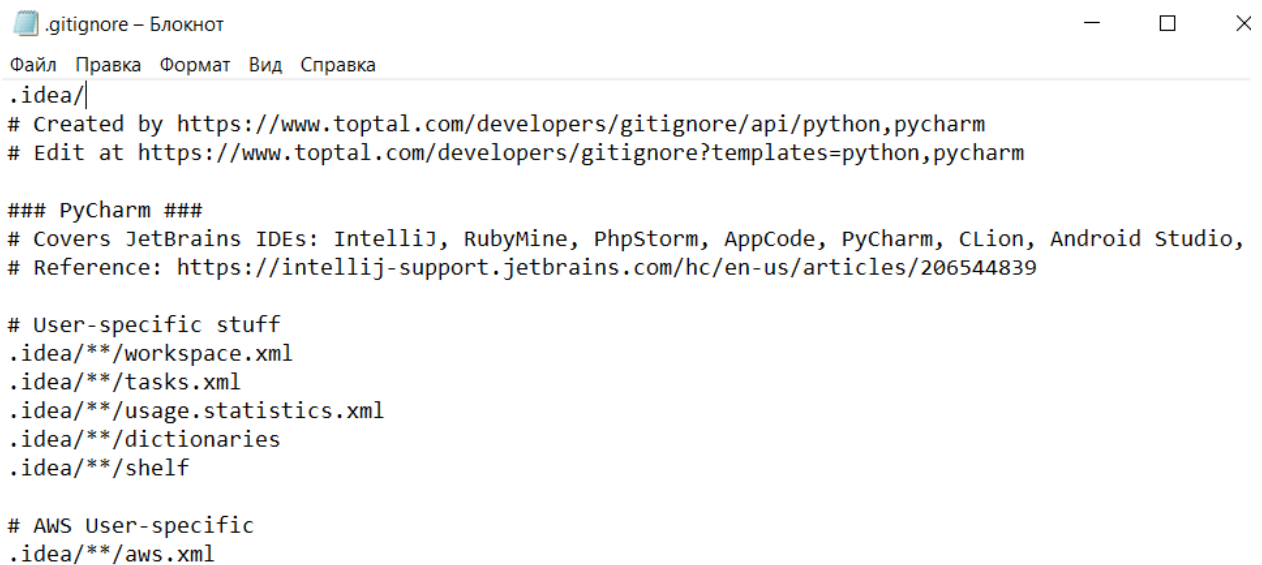
Рисунок 1.2 Клонирование репозитория

```
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>git flow init
which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/adamkh/Desktop/3sem/Python/2.14/2.14_9/.git/hooks]
```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления

git-flow



```
.gitignore – Блокнот
Файл Правка Формат Вид Справка
.idea/
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio,
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml
```

Рисунок 1.4 Изменение .gitignore

## 2. Установка виртуальных окружений и работа с ними.

```
C:\Users\adamkh>pip --version
pip 22.2.2 from C:\Users\adamkh\anaconda3\lib\site-packages\pip (python 3.9)
```

Рисунок 2.1 Проверка версии pip (проверка на наличие pip)

Менеджер пакетов установлен, поэтому можно приступать к установке пакетов.

```
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>python -m venv env
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>
```

Рисунок 2.2 Создание вирт. Окружения

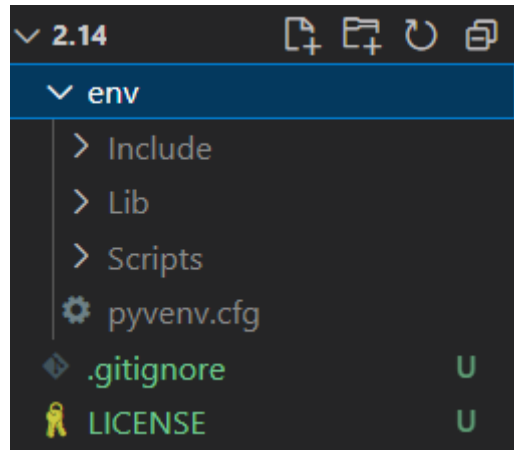


Рисунок 2.3 Созданная папка вирт. окружения

```
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>.\env\Scripts\activate
(env) C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>_
```

Рисунок 2.4 Активация вирт. окружения

```
(env) C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>pip install black
Requirement already satisfied: black in c:\users\adamkh\desktop\3sem\python\2.14\2.14_9\env\lib\site-packages (22.12.0)
Requirement already satisfied: click>=8.0.0 in c:\users\adamkh\desktop\3sem\python\2.14\2.14_9\env\lib\site-packages (from black) (8.1.3)
Requirement already satisfied: typing-extensions>=3.10.0.0 in c:\users\adamkh\desktop\3sem\python\2.14\2.14_9\env\lib\site-packages (from black) (4.4.0)
Requirement already satisfied: mypy-extensions>=0.4.3 in c:\users\adamkh\desktop\3sem\python\2.14\2.14_9\env\lib\site-packages (from black) (0.4.3)
Requirement already satisfied: platformdirs>=2 in c:\users\adamkh\desktop\3sem\python\2.14\2.14_9\env\lib\site-packages (from black) (2.6.0)
Requirement already satisfied: tomli>=1.1.0 in c:\users\adamkh\desktop\3sem\python\2.14\2.14_9\env\lib\site-packages (from black) (2.0.1)
Requirement already satisfied: pathspec>=0.9.0 in c:\users\adamkh\desktop\3sem\python\2.14\2.14_9\env\lib\site-packages (from black) (0.10.3)
Requirement already satisfied: colorama in c:\users\adamkh\desktop\3sem\python\2.14\2.14_9\env\lib\site-packages (from click>=8.0.0->black) (0.4.6)
(env) C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>_
```

Рисунок 2.5 Установка пакета black

```
(env) C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>.\env\Scripts\deactivate
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>_
```

Рисунок 2.6 Деактивация вирт. окружения

**Вирт. окр. virtualenv**

```
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>python -m pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.17.1-py3-none-any.whl (8.8 MB)
    ----- 8.8/8.8 MB 317.8 kB/s eta 0:00:00
Collecting distlib<1,>=0.3.6
  Using cached distlib-0.3.6-py2.py3-none-any.whl (468 kB)
Requirement already satisfied: filelock<4,>=3.4.1 in c:\users\adamkh\anaconda3\lib\site-packages (from virtualenv) (3.6.0)
Requirement already satisfied: platformdirs<3,>=2.4 in c:\users\adamkh\anaconda3\lib\site-packages (from virtualenv) (2.5.2)
Installing collected packages: distlib, virtualenv
Successfully installed distlib-0.3.6 virtualenv-20.17.1
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>
```

Рисунок 2.7 Установка вирт. окр. virtualenv

```
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>virtualenv -p python env
created virtual environment CPython3.9.13.final.0-64 in 6460ms
  creator CPython3Windows(dest=C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9\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\adamkh\AppData\Local\pypa\virtualenv)
    added seed packages: black==22.12.0, click==8.1.3, colorama==0.4.6, mypy_extensions==0.4.3, pathspec==0.10.3, pip==22.3.1, platformdirs==2.6.0, setuptools==65.6.3, tomli==2.0.1, typing_extensions==4.4.0, wheel==0.38.4
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
```

Рисунок 2.8 Создание виртуального окр.

```
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>env\Scripts\activate
(env) C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>deactivate
C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14_9>
```

Рисунок 2.9 Его активация и деактивация

### Перенос вирт. окр.

```
.14\2.14> pip freeze
black==22.10.0
click==8.1.3
colorama==0.4.6
distlib==0.3.6
filelock==3.8.0
mypy_extensions==0.4.3
pathspec==0.10.2
platformdirs==2.5.4
virtualenv==20.17.0
PS C:\Users\adamkh\Desktop\3 семестр\Программирование\2.14\2.14>
```

Рисунок 2.10 Список пакетных зависимостей

```
.14\2.14> pip freeze > requirements.txt
PS C:\Users\adamkh\Desktop\3 семестр\Программирование\2.14\2.14>
```

Рисунок 2.11 Перенаправление вывод команды в файл (сохранение)

requirements.txt – Блокнот

Файл Правка Формат Вид Сп

```
black==22.10.0
click==8.1.3
colorama==0.4.6
distlib==0.3.6
filelock==3.8.0
mypy-extensions==0.4.3
pathspec==0.10.2
platformdirs==2.5.4
virtualenv==20.17.0
```

Рисунок 2.12 Содержимое файла

### Управление пакетами с помощью Conda

```
(base) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> mkdir %python9%

Каталог: C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14

Mode                LastWriteTime         Length Name
----                -
d-----          10.12.2022         8:56             %python9%

(base) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> cd %python9%
(base) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14\%python9%> copy NUL > main.py
```

Рисунок 2.13 Создание чистого вирт. окр. с conda

```
(base) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14\%python9%> cd ../
(base) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> conda create -n %python9% python=3.9
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 22.11.0
  latest version: 22.11.1

Please update conda by running

  $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

  conda install conda=22.11.1

## Package Plan ##

  environment location: C:\Users\adamkh\anaconda3\envs\%python9%

  added / updated specs:
    - python=3.9

The following packages will be downloaded:
```

package	build	size
pip-22.3.1	py39haa95532_0	2.7 MB
python-3.9.15	h6244533_2	19.4 MB
setuptools-65.5.0	py39haa95532_0	1.1 MB
sqlite-3.40.0	h2bbff1b_0	891 KB
tzdata-2022g	h04d1e81_0	114 KB
Total:		24.3 MB

```

The following NEW packages will be INSTALLED:

ca-certificates      pkgs/main/win-64::ca-certificates-2022.10.11-haa95532_0
certifi              pkgs/main/win-64::certifi-2022.9.24-py39haa95532_0
openssl              pkgs/main/win-64::openssl-1.1.1s-h2bbff1b_0
pip                  pkgs/main/win-64::pip-22.3.1-py39haa95532_0
python               pkgs/main/win-64::python-3.9.15-h6244533_2
setuptools           pkgs/main/win-64::setuptools-65.5.0-py39haa95532_0
sqlite               pkgs/main/win-64::sqlite-3.40.0-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-py39haa95532_2

```

```
(base) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> conda activate %python9%
(%python9%) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> _
```

Рисунок 2.14 Его активация

```
(%python9%) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> conda install django, pandas
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Рисунок 2.15 Установка пакетов django и pandas

```
(%python9%) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> conda env export > enviroment.yml
(%python9%) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> _
```

Рисунок 2.16 Создание файла конфигурации для быстрого развертывания  
вирт. окр.



### Задание №1. Установить пакеты pip, NumPy, Pandas, SciPy.

```
(%python9%) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> conda install pip, NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Рисунок 2.17 Установка необходимых пакетов

### Задание №2. Установить менеджером пакетов conda пакет TensorFlow.

```
(%python9%) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> conda install TensorFlow
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Рисунок 2.18 Установка TensorFlow

### Задание №3. Установить пакет TensorFlow с помощью менеджера пакетов pip

```
(%python9%) PS C:\Users\adamkh\Desktop\3sem\Python\2.14\2.14> pip install TensorFlow
Requirement already satisfied: TensorFlow in c:\users\adamkh\anaconda3\envs\%python9%\lib\site-packages (1.16.0)
Requirement already satisfied: six>=1.12.0 in c:\users\adamkh\anaconda3\envs\%python9%\lib\site-packages (from TensorFlow) (1.16.0)
Requirement already satisfied: tensorboard<2.10,>=2.9 in c:\users\adamkh\anaconda3\envs\%python9%\lib\site-packages (from TensorFlow) (2.9.0)
Collecting libclang>=13.0.0
  Using cached libclang-14.0.6-py2.py3-none-win_amd64.whl (14.2 MB)
Requirement already satisfied: numpy>=1.20 in c:\users\adamkh\anaconda3\envs\%python9%\lib\site-packages (from TensorFlow) (1.20.0)
```

Рисунок 2.19 Установка TensorFlow с помощью менеджера пакетов pip

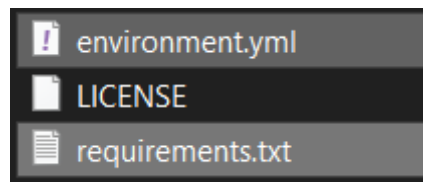


Рисунок 2.20 Сформированные файлы environment.yml и requirements.txt

3. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

```
C:\rep_2.6>git add .
C:\rep_2.6>git commit -m "added programs + modidied .gitignore"
[develop 2582c62] added programs + modidied .gitignore
4 files changed, 379 insertions(+), 3 deletions(-)
create mode 100644 ind.py
create mode 100644 prim.py
create mode 100644 zadaniya.py
C:\rep_2.6>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
C:\rep_2.6>
```

```
C:\rep_2.6>git push
Everything up-to-date
```

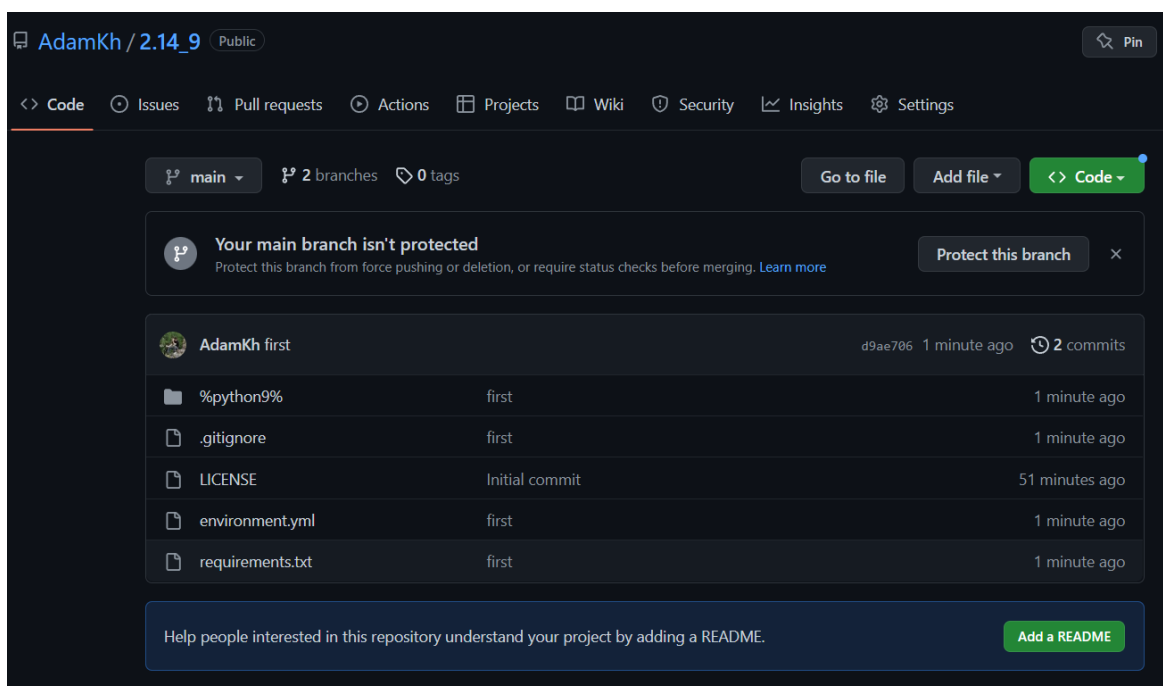
Рисунок 4.1 коммит и пуш изменений и переход на ветку main

```
C:\rep_2.6>git merge develop
Updating 5d4b8d1..2582c62
Fast-forward
 .gitignore | 157 ++++++
--
 ind.py      | 105 ++++++
 prim.py     | 99  ++++++
 zadaniya.py | 21  ++++++
4 files changed, 379 insertions(+), 3 deletions(-)
create mode 100644 ind.py
create mode 100644 prim.py
create mode 100644 zadaniya.py
C:\rep_2.6>
```

Рисунок 4.2 Слияние ветки main с develop

```
C:\rep_2.6>git push
info: please complete authentication in your browser...
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 5.16 KiB | 2.58 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/AdamKh/rep_2.6.git
5d4b8d1..2582c62 main -> main
```

Рисунок 4.3 Пуш изменений на удаленный сервер



AdamKh / 2.14\_9 Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 2 branches 0 tags Go to file Add file <> Code

**Your main branch isn't protected**  
Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#) Protect this branch

AdamKh first d9ae706 1 minute ago 2 commits

File	Commit	Time
%python9%	first	1 minute ago
.gitignore	first	1 minute ago
LICENSE	Initial commit	51 minutes ago
environment.yml	first	1 minute ago
requirements.txt	first	1 minute ago

Help people interested in this repository understand your project by adding a README. Add a README

Рисунок 4.4 Изменения на удаленном сервере

## **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 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?**

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 па-

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.