

**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 №2.8 Дисциплина:

«Программирование на Python» Тема:

«Рекурсия в языке Python»

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

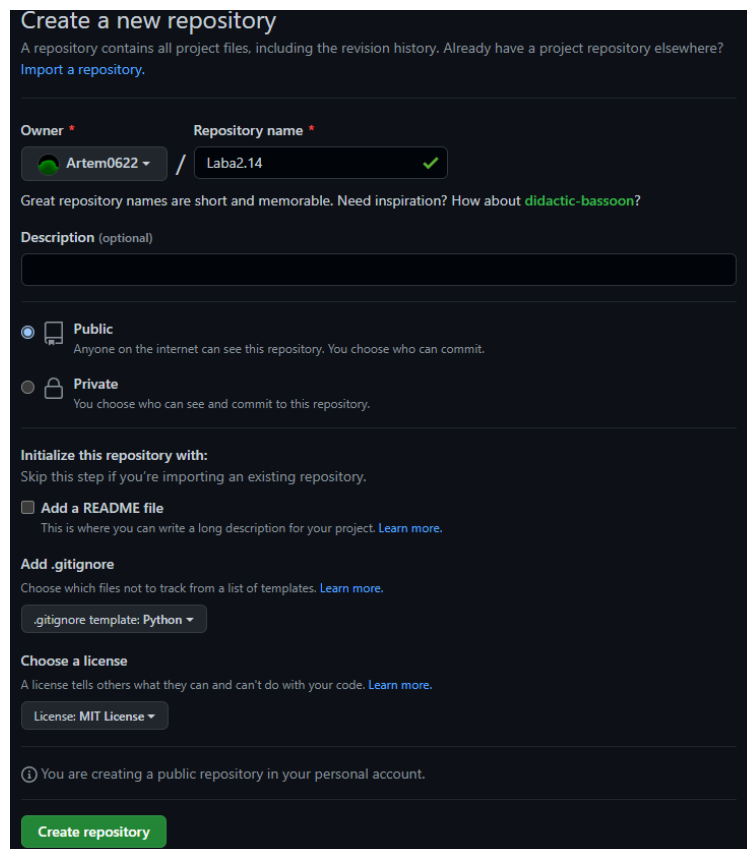
Богдашов Артём
Владимирович

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «Laba2.14» в который добавил .gitignore, который дополнил правила 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.



The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there are two input fields: 'Owner' with a dropdown menu showing 'Artem0622' and 'Repository name' with a text input showing 'Laba2.14'. A green checkmark is next to the repository name. Below these fields, there is a note: 'Great repository names are short and memorable. Need inspiration? How about [didactic-bassoon?](#)'. Then, there is a 'Description (optional)' text area. Below that, there are two radio buttons for visibility: 'Public' (selected) and 'Private'. The 'Public' option has a subtext: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option has a subtext: 'You choose who can see and commit to this repository.' Below the visibility options, there is a section 'Initialize this repository with:' with a subtext: 'Skip this step if you're importing an existing repository.' There is a checkbox 'Add a README file' with a subtext: 'This is where you can write a long description for your project. [Learn more.](#)'. Below that, there is a section 'Add .gitignore' with a subtext: 'Choose which files not to track from a list of templates. [Learn more.](#)'. There is a dropdown menu for '.gitignore template: Python'. Below that, there is a section 'Choose a license' with a subtext: 'A license tells others what they can and can't do with your code. [Learn more.](#)'. There is a dropdown menu for 'License: MIT License'. At the bottom, there is a note: 'You are creating a public repository in your personal account.' and a green 'Create repository' button.

Рисунок 1.1 Создание репозитория

```
C:\Games\Программы\СУ\1.3\Программирование на Python\2.14>git clone https://github.com/Artem0622/Laba2.14.git
Cloning into 'Laba2.14'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

C:\Games\Программы\СУ\1.3\Программирование на Python\2.14>
```

Рисунок 1.2 Клонирование репозитория

```
C:\Games\Программы\СУ\1.3\Программирование на Python\2.14\Laba2.14>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:/Games/Программы/СУ/1.3/Программирование на Python/2.14/Laba2.14/.git/hooks]
```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления
git-flow

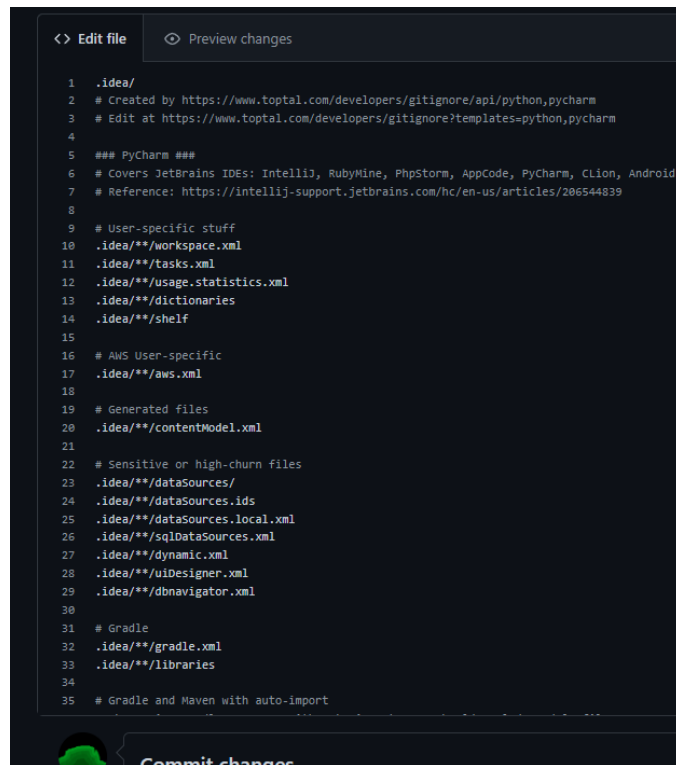


Рисунок 1.4 Изменение .gitignore

1. Начало работы с виртуальными окружениями и их установка.

```

C:\Users\Kwil>pip --version
pip 22.3.1 from C:\Users\Kwil\AppData\Local\Programs\Python\Python310\lib\site-packages\pip (python 3.10)
C:\Users\Kwil>

```

Рисунок 2.1 – Проверка установки pip

```

C:\Games\Программы\СУ\1.3\Программирование на Python\2.14\git\Python9>python -m venv env
C:\Games\Программы\СУ\1.3\Программирование на Python\2.14\git\Python9>

```

Рисунок 2.2 – Создание виртуального окружения

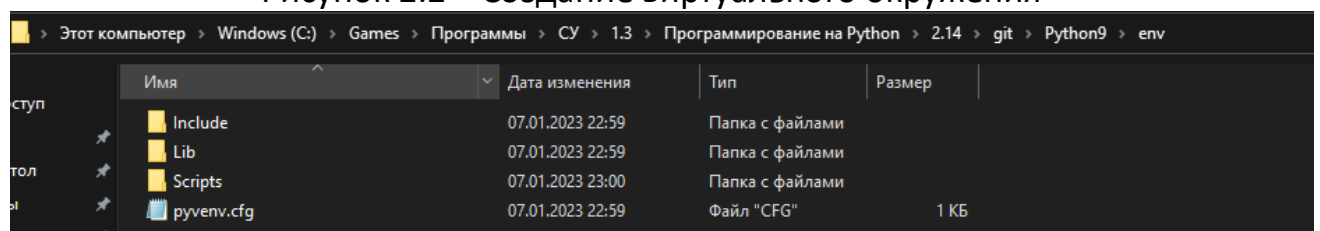


Рисунок 2.3 – Созданная папка виртуального окружения

```
C:\Games\Программы\СУ\1.3\Программирование на Python\2.14\git\Python9>.\env\scripts\activate
(env) C:\Games\Программы\СУ\1.3\Программирование на Python\2.14\git\Python9>.\env\scripts\activate.ps1
(env) C:\Games\Программы\СУ\1.3\Программирование на Python\2.14\git\Python9>
```

Рисунок 2.4 – Активация виртуального окружения

```
(env) C:\Users\Kwil\Desktop>pip install black
Requirement already satisfied: black in c:\users\kwil\desktop\env\lib\site-packages (22.12.0)
Requirement already satisfied: platformdirs>=2 in c:\users\kwil\desktop\env\lib\site-packages (from black) (2.6.2)
Requirement already satisfied: pathspec>=0.9.0 in c:\users\kwil\desktop\env\lib\site-packages (from black) (0.10.3)
Requirement already satisfied: click>=8.0.0 in c:\users\kwil\desktop\env\lib\site-packages (from black) (8.1.3)
Requirement already satisfied: tomli>=1.1.0 in c:\users\kwil\desktop\env\lib\site-packages (from black) (2.0.1)
Requirement already satisfied: mypy-extensions>=0.4.3 in c:\users\kwil\desktop\env\lib\site-packages (from black) (0.4.3)
Requirement already satisfied: colorama in c:\users\kwil\desktop\env\lib\site-packages (from click>=8.0.0->black) (0.4.6)
WARNING: You are using pip version 22.0.4; however, version 22.3.1 is available.
You should consider upgrading via the 'C:\Users\Kwil\Desktop\env\Scripts\python.exe -m pip install --upgrade pip' command.

(env) C:\Users\Kwil\Desktop>pip freeze
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

(env) C:\Users\Kwil\Desktop>deactivate
C:\Users\Kwil\Desktop>
```

Рисунок 2.5 – Деактивация виртуального окружения

2. Установим виртуальное окружение с virtualenv.

1. Для начала пакет нужно установить. Выполним установку с помощью команды: `python -m pip install virtualenv`:

```
C:\Users\Kwil\Desktop>python -m pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.17.1-py3-none-any.whl (8.8 MB)
----- 8.8/8.8 MB 4.2 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 4.2 MB/s eta 0:00:00
Collecting filelock<4,>=3.4.1
  Downloading filelock-3.9.0-py3-none-any.whl (9.7 kB)
Collecting platformdirs<3,>=2.4
  Using cached platformdirs-2.6.2-py3-none-any.whl (14 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.6 filelock-3.9.0 platformdirs-2.6.2 virtualenv-20.17.1

C:\Users\Kwil\Desktop>
```

Рисунок 3.1 – Процесс установки

2. Создадим с помощью virtualenv в текущей папке виртуально окружение с помощью команды virtualenv -p python env:

```
C:\Users\Kwil\Desktop\lab2.14>python -m virtualenv env2
created virtual environment CPython3.10.4.final.0-64 in 689ms
creator CPythonWindows(dest=C:\Users\Kwil\Desktop\lab2.14\env2, 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\Kwil\AppData\Local\pypa\virtualenv)
added seed packages: pip==22.3.1, setuptools==65.6.3, wheel==0.38.4
activators BashActivator,BatchActivator,FishActivator,PowerShellActivator,PythonActivator
```

Рисунок 3.2– Создание виртуального окружения

```
C:\Users\Kwil\Desktop\lab2.14>.\env2\scripts\activate

(env2) C:\Users\Kwil\Desktop\lab2.14>pip freeze

(env2) C:\Users\Kwil\Desktop\lab2.14>deactivate
C:\Users\Kwil\Desktop\lab2.14>
```

Рисунок 3.3– активация и деактивация

4. Перенос виртуального окружения.

Просмотрим список пакетных зависимостей с помощью команды pip freeze:

```
black==22.12.0
click==8.1.3
colorama==0.4.6
ipython==8.12.0
ipython-genutils==0.2.0
jupyter==1.0.0
jupyter-console==6.4.0
jupyter-core==4.12.0
matplotlib==3.7.1
matplotlib-inline==0.1.6
mypy==1.0.0
mypy-extensions==0.4.3
pathspec==0.10.3
platformdirs==2.6.2
tomli==2.0.1
```

Рисунок 3.4– Список пакетных зависимостей

Сохраним его. Нужно перенаправить вывод команды в файл:

```
c:\Users\Admin\Desktop\git\Python9>pip freeze > requirements.txt  
c:\Users\Admin\Desktop\git\Python9>
```

Рисунок 3.5 – Сохранение

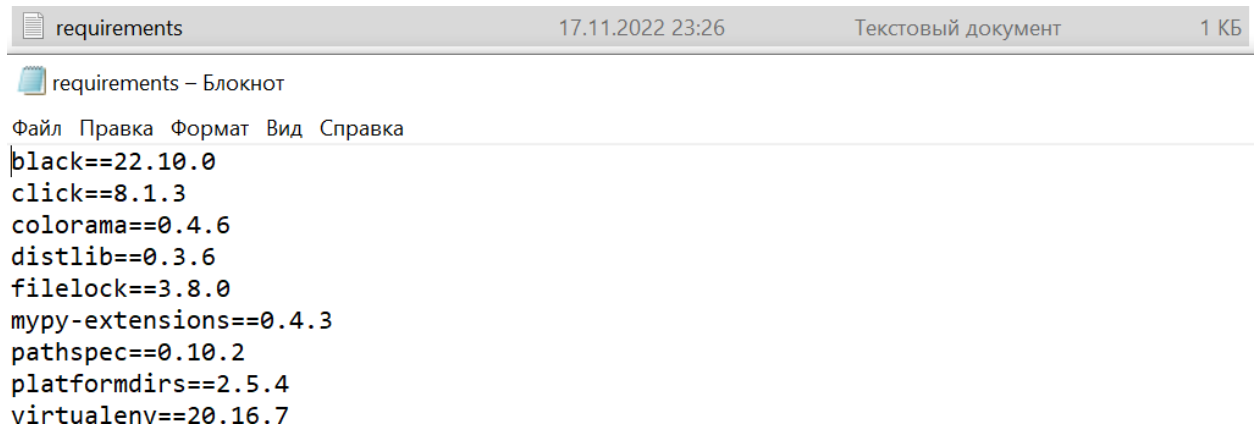


Рисунок 3.5 – Сам файл

Теперь установка пакетов из файла зависимостей в новом виртуальном окружении может выполняться одной командой: `pip install -r requirements.txt`

5. Управление пакетами с помощью Conda.

1. Создадим чистое виртуальное окружение с conda и активируем его.

```
(base) C:\Users\Kwil>mkdir %python9%  
Подпапка или файл %python9% уже существует.  
  
(base) C:\Users\Kwil>cd %python9%  
  
(base) C:\Users\Kwil\%python9%>copy NUL>main.py
```

Рисунок 4.1 – Создание чистой директории и виртуального окружения

```

(base) C:\Users\Kwil\Desktop\anconda>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: 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\Kwil\.conda\envs\python=3.9

Proceed ([y]/n)? y
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate python=3.9
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\Kwil\Desktop\anconda>conda active %python9%
CommandNotFoundError: No command 'conda active'.

```

Рисунок 4.2 – Активация окружения conda

2. Необходимо установить пакеты для реализации проекта, а именно Django и pandas.

```
(base) C:\Users\Kwil\Desktop\anconda>conda activate %python9%

(base) C:\Users\Kwil\Desktop\anconda>conda install django, pandas
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.

PackagesNotNotFoundError: The following packages are not available from current channels:

  - django,

Current channels:

  - https://repo.anaconda.com/pkgs/main/win-64
  - https://repo.anaconda.com/pkgs/main/noarch
  - https://repo.anaconda.com/pkgs/r/win-64
  - https://repo.anaconda.com/pkgs/r/noarch
  - https://repo.anaconda.com/pkgs/msys2/win-64
  - https://repo.anaconda.com/pkgs/msys2/noarch

To search for alternate channels that may provide the conda package you're
looking for, navigate to

    https://anaconda.org

and use the search bar at the top of the page.
```

```
(base) C:\Users\Kwil\Desktop\anconda>conda install pandas
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3

  added / updated specs:
    - pandas

The following packages will be downloaded:



| package                | build          |        |
|------------------------|----------------|--------|
| conda-22.11.1          | py39haa95532_4 | 892 KB |
| ruamel.yaml-0.17.21    | py39h2bbff1b_0 | 174 KB |
| ruamel.yaml.clib-0.2.6 | py39h2bbff1b_1 | 101 KB |
| Total:                 |                | 1.1 MB |



The following NEW packages will be INSTALLED:

  ruamel.yaml      pkgs/main/win-64::ruamel.yaml-0.17.21-py39h2bbff1b_0
  ruamel.yaml.clib pkgs/main/win-64::ruamel.yaml.clib-0.2.6-py39h2bbff1b_1

The following packages will be UPDATED:

  conda                4.12.0-py39haa95532_0 --> 22.11.1-py39haa95532_4

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

Рисунок 4.3 – Установка пакетов

3. Необходимо сформировать файл конфигурации виртуального окружения, для быстрого развёртывания в будущем.

```
(base) C:\Users\Kwil\Desktop\anconda>conda env export >environment.yml  
(base) C:\Users\Kwil\Desktop\anconda>_
```

Рисунок 4.4 – Создание файла конфигурации environment.yml

4. (Задание №1) Установим согласно заданию №6 в виртуальное окружение следующие пакеты: pip, NumPy, Pandas, SciPy.

```
(base) C:\Users\Kwil\Desktop\anconda> conda install pip, NumPy, Pandas, SciPy  
Collecting package metadata (current_repodata.json): done  
Solving environment: failed with initial frozen solve. Retrying with flexible solve.  
Collecting package metadata (repodata.json): done  
Solving environment: failed with initial frozen solve. Retrying with flexible solve.  
  
PackagesNotFoundError: The following packages are not available from current channels:  
  
- pandas,  
- numpy,  
- pip,  
  
Current channels:  
  
- https://repo.anaconda.com/pkgs/main/win-64  
- https://repo.anaconda.com/pkgs/main/noarch  
- https://repo.anaconda.com/pkgs/r/win-64  
- https://repo.anaconda.com/pkgs/r/noarch  
- https://repo.anaconda.com/pkgs/msys2/win-64  
- https://repo.anaconda.com/pkgs/msys2/noarch  
  
To search for alternate channels that may provide the conda package you're  
looking for, navigate to  
  
    https://anaconda.org  
  
and use the search bar at the top of the page.
```

Рисунок 4.5 – Установка необходимых пакетов

5. (Задание №2) Попробуем установить менеджером пакетов conda пакет TensorFlow.

```

anaconda==2022.05 -> gensim==4.1.2[build=py38hd77b12b_0|py37hd77b12b_0|py39hd77b12b_0]

Package pytest-runner conflicts for:
itemloaders -> parsel[version='>=1.5.0'] -> pytest-runner
scrapy -> parsel[version='>=1.5.0'] -> pytest-runner
parsel -> pytest-runner

Package paramiko conflicts for:
anaconda==2022.05 -> paramiko==2.8.1=pyhd3eb1b0_0
spyder -> paramiko[version='>=2.4.0']
anaconda==2022.05 -> spyder==5.1.5=py37haa95532_1 -> paramiko[version='>=2.4.0']

Package itemadapter conflicts for:
scrapy -> itemadapter[version='>=0.1.0']
anaconda==2022.05 -> itemloaders==1.0.4=pyhd3eb1b0_1 -> itemadapter[version='>=0.1.0']
anaconda==2022.05 -> itemadapter==0.3.0=pyhd3eb1b0_0
itemloaders -> itemadapter[version='>=0.1.0']

Package pathtools conflicts for:
spyder -> watchdog[version='>=0.10.3'] -> pathtools[version='>=0.1.1']
watchdog -> pathtools[version='>=0.1.1']

Package protego conflicts for:
anaconda==2022.05 -> protego==0.1.16=py_0
anaconda==2022.05 -> scrapy==2.6.1=py37haa95532_0 -> protego[version='>=0.1.15']
scrapy -> protego[version='>=0.1.15']

Package ecdsa conflicts for:
paramiko -> ecdsa[version='>=0.11,<2.0']
spyder -> paramiko -> ecdsa[version='>=0.11,<2.0']

Package textdistance conflicts for:
anaconda==2022.05 -> spyder==5.1.5=py37haa95532_1 -> textdistance[version='>=4.2.0']
spyder -> textdistance[version='>=4.2.0']
anaconda==2022.05 -> textdistance==4.2.1=pyhd3eb1b0_0

Package qdarkstyle conflicts for:
anaconda==2022.05 -> qdarkstyle==3.0.2=pyhd3eb1b0_0
spyder -> qdarkstyle[version='3.0.2.*|>=3.0.2,<3.1.0|>=2.8,<3.0|>=2.8|>=2.7']
anaconda==2022.05 -> spyder==5.1.5=py37haa95532_1 -> qdarkstyle=3.0.2

Package glob2 conflicts for:
conda-build -> glob2[version='>=0.6']
anaconda==2022.05 -> glob2==0.7=pyhd3eb1b0_0

(base) C:\Users\Kwii\Desktop\anconda>

```

Рсуюнок 4.6 – Успешная установка TensorFlow

Задание №3. Попробуйте установить пакет TensorFlow с помощью менеджера пакетов `pip`.

```
Collecting markdown>2.6.8
  Downloading Markdown-3.4.1-py3-none-any.whl (93 kB)
Collecting google-auth-httplib2>0.4.1
  Downloading google_auth_httplib2-0.4.6-py2.py3-none-any.whl (18 kB)
Collecting werkzeug>1.0.1
  Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
Collecting tensorboard-plugin-wit>1.6.0
  Downloading tensorboard_plugin_wit-1.8.1-py3-none-any.whl (781 kB)
Collecting tensorboard-data-server<0.7.0,>=0.6.0
  Downloading tensorboard_data_server-0.6.1-py3-none-any.whl (2.4 kB)
Collecting requests<3,>=2.21.0
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
Collecting pyasn1-modules>0.2.1
  Downloading pyasn1_modules-0.2.8-py2.py3-none-any.whl (155 kB)
Collecting cachetools<5.2.0,>=2.0.0
  Downloading cachetools-5.2.0-py3-none-any.whl (9.3 kB)
Collecting rsa<4.9,>=4.0
  Downloading rsa-4.9-py3-none-any.whl (34 kB)
Collecting requests-oauthlib>0.7.0
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Collecting charset-normalizer<3,>=2
  Downloading charset-normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting lxml4>2.6
  Downloading lxml-3.4-py3-none-any.whl (61 kB)
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.11-py2.py3-none-any.whl (140 kB)
Collecting certifi>2017.4.17
  Downloading certifi-2022.12.7-py3-none-any.whl (155 kB)
Collecting MarkupSafe>2.1.1
  Downloading MarkupSafe-2.1.1-cp310-cp310-win_arm64.whl (17 kB)
Collecting pyasn1<0.5.0,>=0.4.6
  Downloading pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
Collecting oauthlib>3.0.0
  Downloading oauthlib-3.2.2-py3-none-any.whl (151 kB)
Installing collected packages: tensorboard-plugin-wit, pyasn1, libclang, flatbuffers, wrapt, wheel, urllib3, typing-extensions, termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator, tensorboard-data-server, six, rsa, pyasn1-modules, protobuf, packaging, oauthlib, numpy, MarkupSafe, markdown, keras, idna, grpcio, gast, charset-normalizer, certifi, cachetools, absl-py, werkzeug, requests, opt-einsum, h5py, google-pasta, google-auth, astunparse, requests-oauthlib, google-auth-httplib2, tensorboard, tensorflow-intel, TensorFlow
```

Рисунок 4.7 – Успешная установка TensorFlow с помощью `pip`.

Ответы на 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 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 окружением (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 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.