

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-  
СКОЙ ФЕДЕРАЦИИ**

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

**Кафедра инфокоммуникаций**

**Основы кроссплатформенного программирования**

**Отчет по лабораторной работе №2.14**

Тема: «Установка пакетов в Python. Виртуальные окружения»

Выполнил студент группы

ИВТ-б-о-21-1

Криворот В.Г. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А.

\_\_\_\_\_  
(подпись)

Ставрополь 2022

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

## Ход работы:

**1. Создал репозиторий в GitHub,** дополнил правила в `.gitignore` для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления `git-flow`.

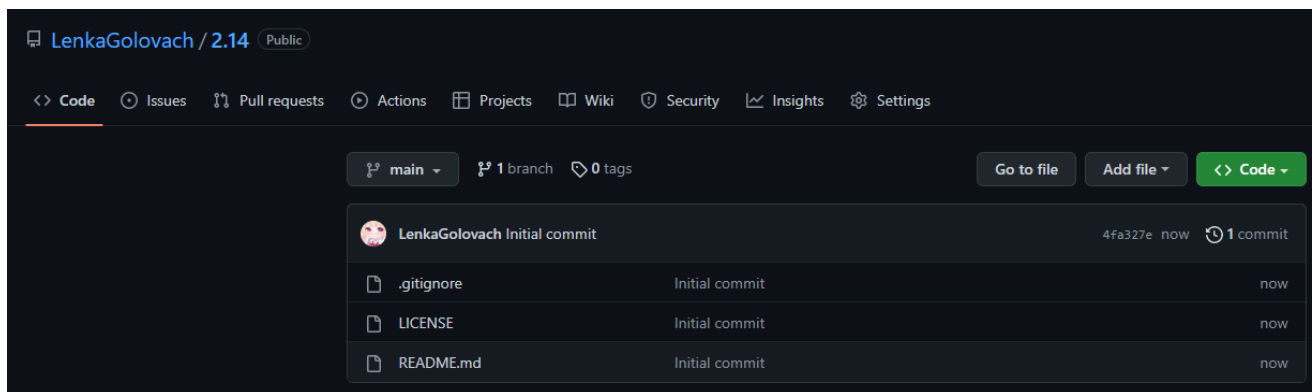


Рисунок 1.1 – Созданный репозиторий

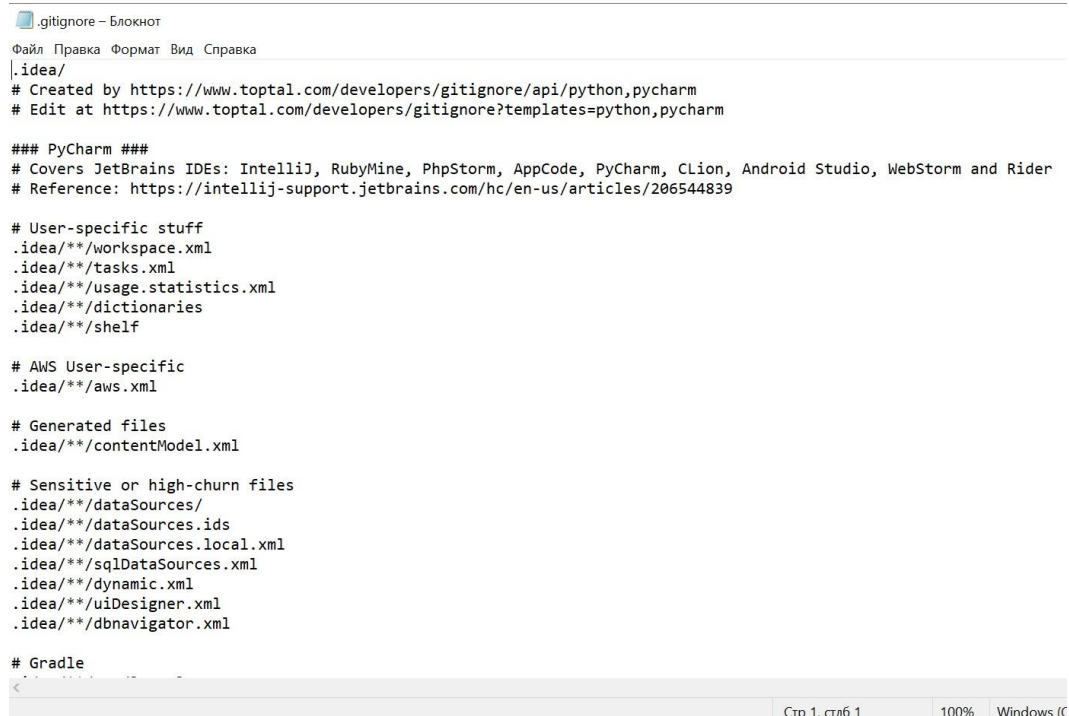


Рисунок 1.2 – Дополнил правила в `.gitignore`

```
c:\Users\Admin\Desktop\git\Python9>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/Admin/Desktop/git/Python9/.git/hooks]

c:\Users\Admin\Desktop\git\Python9>
```

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

## 2. Начало работы с виртуальными окружениями и их установка.

Проверим установлен ли менеджер пакетов pip.

```
c:\Users\Admin\Desktop\git\Python9>pip --version
pip 22.3 from C:\Users\Admin\AppData\Local\Programs\Python\Python311\Lib\site-packages\pip (python 3.11)

c:\Users\Admin\Desktop\git\Python9>
```

Рисунок 2 – Проверка установки pip

Так как менеджер пакетов уже установлен (рисунок 2), то приступим к установке venv.

Установка venv введём команду вида: python3 -m venv «название папки куда будет установлено виртуальное окружение»:

```
c:\Users\Admin\Desktop\git\Python9>python -m venv env

c:\Users\Admin\Desktop\git\Python9>
```

Рисунок 3 – Создание виртуального окружения

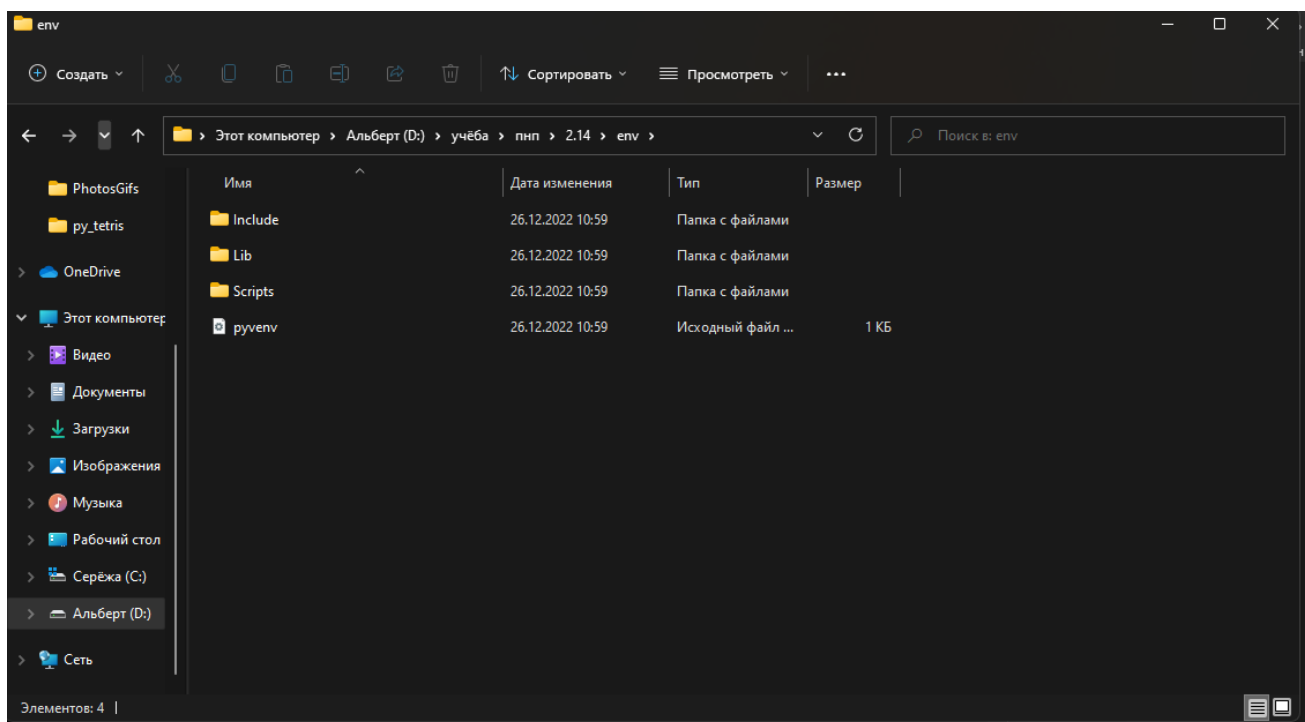


Рисунок 4 – Созданная папка виртуального окружения

Теперь активируем наше виртуальное окружение с помощью команды вида: `.\env\scripts\activate`

```
D:\учёба\пнп\2.14>.\env\scripts\activate.ps1
D:\учёба\пнп\2.14>.\env\scripts\activate
(env) D:\учёба\пнп\2.14>
```

Рисунок 5 – Активация виртуального окружения

В качестве примера работоспособности виртуального окружения, установил в виртуальное окружение пакет `black` и деактивировал его, с помощью команды `deactivate`.

```
(env) c:\Users\Admin\Desktop\git\Python9>pip install black
Collecting black
  Downloading black-22.10.0-cp311-cp311-win_amd64.whl (1.2 MB)
    ----- 1.2/1.2 MB 4.2 MB/s eta 0:00:00
Collecting click>=8.0.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
    ----- 96.6/96.6 kB 5.4 MB/s eta 0:00:00
Collecting mypy_extensions>=0.4.3
  Downloading mypy_extensions-0.4.3-py2.py3-none-any.whl (4.5 kB)
Collecting pathspec>=0.9.0
  Downloading pathspec-0.10.2-py3-none-any.whl (28 kB)
Collecting platformdirs>=2
  Downloading platformdirs-2.5.4-py3-none-any.whl (14 kB)
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: mypy_extensions, platformdirs, pathspec, colorama, click, black
Successfully installed black-22.10.0 click-8.1.3 colorama-0.4.6 mypy_extensions-0.4.3 pathspec-0.10.2 platformdirs-2.5.4

[notice] A new release of pip available: 22.3 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(env) c:\Users\Admin\Desktop\git\Python9>deactivate
c:\Users\Admin\Desktop\git\Python9>
```

Рисунок 6 – Деактивация виртуального окружения

### 3. Установим виртуальное окружение с `virtualenv`.

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

```
D:\учёба\пнп\2.14>python -m pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.17.1-py3-none-any.whl (8.8 MB)
    ----- 8.8 MB 6.4 MB/s
Collecting distlib<1,>=0.3.6
  Downloading distlib-0.3.6-py2.py3-none-any.whl (468 kB)
    ----- 468 kB 6.4 MB/s
Collecting platformdirs<3,>=2.4
  Using cached platformdirs-2.6.0-py3-none-any.whl (14 kB)
Requirement already satisfied: filelock<4,>=3.4.1 in c:\users\karam\anaconda3\lib\site-packages (from virtualenv) (3.6.0)
Installing collected packages: platformdirs, distlib, virtualenv
Successfully installed distlib-0.3.6 platformdirs-2.6.0 virtualenv-20.17.1
```

Рисунок 7 – Процесс установки

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

```
c:\Users\Admin\Desktop\git\Python9>virtualenv -p python env
created virtual environment CPython3.11.0.final.0-64 in 8852ms
creator CPython3Windows(dest=C:\Users\Admin\Desktop\git\Python9\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\Admin\AppData\Local\pypa\virtualenv)
added seed packages: black==22.10.0, click==8.1.3, colorama==0.4.6, mypy_extensions==0.4.3, pathspec==0.10.2, pip==22.3.1, platformdirs==2.5.4, setuptools==65.5.1, wheel==0.38.4
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
c:\Users\Admin\Desktop\git\Python9>
```

Рисунок 8 – Создание виртуального окружения

Попробуем его активировать и деактивировать.

```
c:\Users\Admin\Desktop\git\Python9>.\env\scripts\activate

(env) c:\Users\Admin\Desktop\git\Python9>deactivate
c:\Users\Admin\Desktop\git\Python9>
```

Рисунок 10 – активация и деактивация

#### 4. Перенос виртуального окружения.

Просмотрим список пакетных зависимостей с помощью команды pip freeze:

```
c:\Users\Admin\Desktop\git\Python9>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==20.16.7

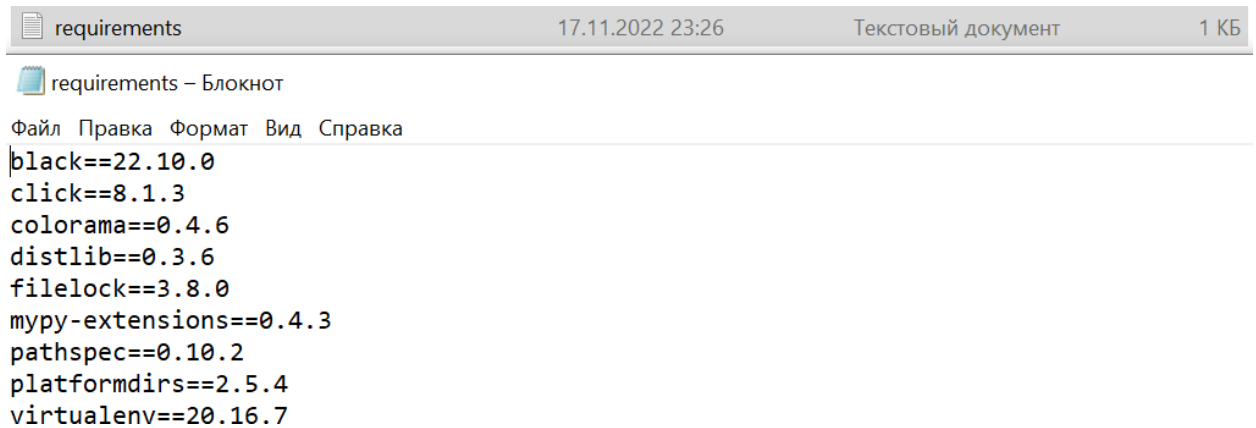
c:\Users\Admin\Desktop\git\Python9>
```

Рисунок 11 – 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 12 – Сохранение



requirements 17.11.2022 23:26 Текстовый документ 1 КБ

requirements – Блокнот

Файл Правка Формат Вид Справка

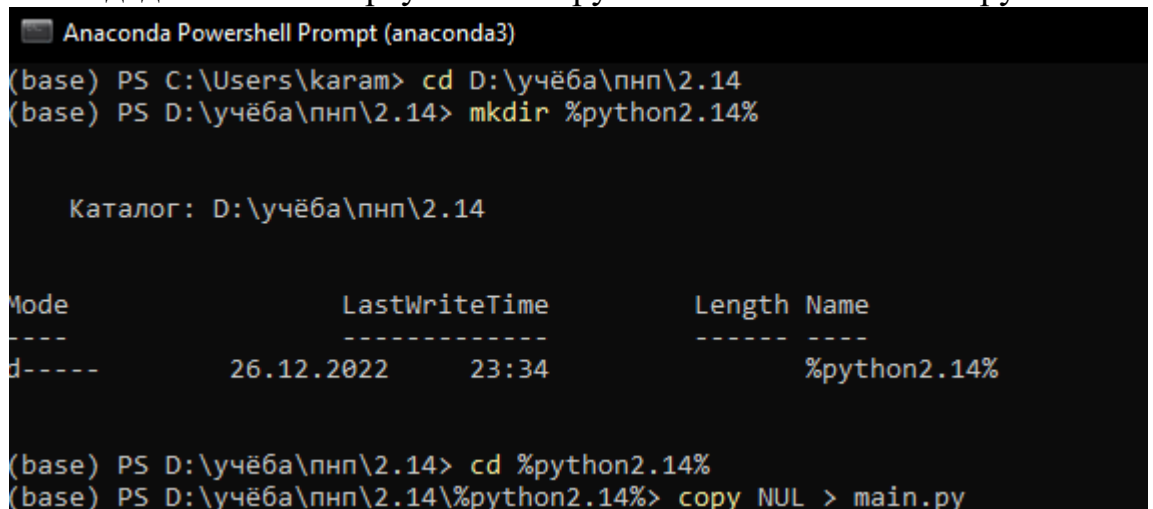
black==22.10.0  
click==8.1.3  
colorama==0.4.6  
distlib==0.3.6  
filelock==3.8.0  
мур-extensions==0.4.3  
pathspec==0.10.2  
platformdirs==2.5.4  
virtualenv==20.16.7

Рисунок 13 – 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.



```
Anaconda Powershell Prompt (anaconda3)  
(base) PS C:\Users\karam> cd D:\учёба\пнп\2.14  
(base) PS D:\учёба\пнп\2.14> mkdir %python2.14%  
  
Каталог: D:\учёба\пнп\2.14  
  
Mode                LastWriteTime         Length Name  
----                -  
d-----          26.12.2022    23:34             %python2.14%  
  
(base) PS D:\учёба\пнп\2.14> cd %python2.14%  
(base) PS D:\учёба\пнп\2.14\%python2.14%> copy NUL > main.py
```

Рисунок 14 – Создание чистой директории и виртуального окружения

```
(base) PS D:\учёба\пнп\2.14> conda create -n %python2.14% python=3.9
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\karam\anaconda3\envs\%python2.14%

  added / updated specs:
    - python=3.9

The following packages will be downloaded:
```

package	build	
ca-certificates-2022.10.11	haa95532_0	125 KB
certifi-2022.12.7	py39haa95532_0	149 KB
openssl-1.1.1s	h2bbff1b_0	5.5 MB
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:		30.1 MB

Рисунок 15 – Активация окружения conda

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

```
(base) PS D:\учёба\пнп\2.14> conda activate %python2.14%
(%python2.14%) PS D:\учёба\пнп\2.14> conda install django, pandas
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\karam\anaconda3\envs\%python2.14%

  added / updated specs:
    - django
    - pandas

The following packages will be downloaded:
```

package	build	
asgiref-3.5.2	py39haa95532_0	39 KB
bottleneck-1.3.5	py39h080aedc_0	106 KB
django-4.1	py39haa95532_0	4.2 MB
numexpr-2.8.4	py39h5b0cc5e_0	127 KB
numpy-1.23.4	py39h3b20f71_0	11 KB
numpy-base-1.23.4	py39h4da318b_0	6.0 MB
packaging-22.0	py39haa95532_0	67 KB
pandas-1.5.2	py39hf11a4ad_0	10.5 MB
python-tzdata-2021.1	pyhd3eb1b0_0	137 KB

Рисунок 16 – Установка пакетов



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

```
(%python2.14%) PS D:\учёба\пнп\2.14> conda env export > enviroment.yml
(%python2.14%) PS D:\учёба\пнп\2.14> _
```

Рисунок 17 – Создание файла конфигурации environment.yml

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

```
(%python2.14%) PS D:\учёба\пнп\2.14> conda env export > enviroment.yml
(%python2.14%) PS D:\учёба\пнп\2.14> conda install pip, NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\Users\karam\anaconda3\envs\%python2.14%

added / updated specs:
- numpy
- pandas
- pip
- scipy

The following packages will be downloaded:
```

package	build	
fftw-3.3.9	h2bbff1b_1	672 KB
icc_rt-2022.1.0	h6049295_2	6.5 MB
scipy-1.9.3	py39he11b74f_0	18.0 MB
Total:		25.2 MB

Рисунок 18 – Установка необходимых пакетов

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

Downloading and Extracting Packages			
libcurl-7.86.0	319 KB	#####	100%
async-timeout-4.0.2	12 KB	#####	100%
tensorboard-plugin-w	671 KB	#####	100%
tensorflow-estimator	491 KB	#####	100%
pyopenssl-22.0.0	50 KB	#####	100%
tensorboard-2.10.0	5.6 MB	#####	100%
cffi-1.15.1	238 KB	#####	100%
_tfselect-2.3.0	3 KB	#####	100%
tensorboard-data-ser	17 KB	#####	100%
python-flatbuffers-2	34 KB	#####	100%
tensorflow-base-2.10	75.4 MB	####	5%
typing_extensions-4.	47 KB	#####	100%
google-auth-oauthlib	18 KB	#####	100%
flatbuffers-2.0.0	1.4 MB	#####	100%
keras-2.10.0	1.5 MB	#####	100%
werkzeug-2.2.2	340 KB	#####	100%
gast-0.4.0	13 KB	#####	100%
markupsafe-2.1.1	26 KB	#####	100%
libprotobuf-3.20.1	2.0 MB	#####	100%
wrapt-1.14.1	49 KB	#####	100%
hdf5-1.10.6	7.9 MB	#####4	68%
multidict-6.0.2	46 KB	#####	100%
protobuf-3.20.1	231 KB	#####	100%
idna-3.4	93 KB	#####	100%
markdown-3.4.1	148 KB	#####	100%
requests-2.28.1	99 KB	#####	100%
opt_einsum-3.3.0	57 KB	#####	100%
colorama-0.4.5	28 KB	#####	100%
... (more hidden) ...			

Рисунок 19 – Успешная установка TensorFlow

TensorFlow успешно установился менеджером пакетов conda без ошибок.

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

```
(env) D:\учёба\пнп\2.14>pip install TensorFlow
Collecting TensorFlow
  Downloading tensorflow-2.11.0-cp39-cp39-win_amd64.whl (1.9 kB)
Collecting tensorflow-intel==2.11.0
  Downloading tensorflow_intel-2.11.0-cp39-cp39-win_amd64.whl (266.3 MB)
    | 266.3 MB 6.8 MB/s
Collecting absl-py>=1.0.0
  Downloading absl_py-1.3.0-py3-none-any.whl (124 kB)
    | 124 kB 6.8 MB/s
Collecting flatbuffers>=2.0
  Downloading flatbuffers-22.12.6-py2.py3-none-any.whl (26 kB)
Collecting astunparse>=1.6.0
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: h5py>=2.9.0 in c:\users\karam\anaconda3\lib\site-packages (from tensorflow-intel==2.11.0->TensorFlow) (3.6.0)
Requirement already satisfied: six>=1.12.0 in c:\users\karam\anaconda3\lib\site-packages (from tensorflow-intel==2.11.0->TensorFlow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\karam\anaconda3\lib\site-packages (from tensorflow-intel==2.11.0->TensorFlow) (1.42.0)
Collecting opt-einsum>=2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
    | 65 kB ...
Collecting tensorflow-io-gcs-filesystem>=0.23.1
  Downloading tensorflow_io_gcs_filesystem-0.29.0-cp39-cp39-win_amd64.whl (1.5 MB)
    | 1.5 MB 6.8 MB/s
Collecting tensorboard<2.12,>=2.11
  Downloading tensorboard-2.11.0-py3-none-any.whl (6.0 MB)
```

Рисунок 20 – Успешная установка TensorFlow с помощью pip

Сформировал файлы environment.yml и requirements.txt.




 environment.yml	18.11.2022 15:34	Файл "YML"	8 КБ
 LICENSE	17.11.2022 22:32	Файл	2 КБ
 requirements	18.11.2022 16:01	Текстовый документ	1 КБ

Рисунок 21 – Сформированные файлы environment.yml и requirements.txt

**Вывод:** в результате выполнения лабораторной работы были приобретены теоретические сведения и практические навыки для работы с менеджером пакетов pip и виртуальными окружениями с помощью языка программирования Python версии 3.x..

### 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-либо 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.