

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Лабораторная работа 2.14

Установка пакетов в Python. Виртуальные окружения

Выполнил студент группы ИВТ-б-о-20-1

Пушкин Н.С. « _____ » 20__ г.

Подпись студента _____

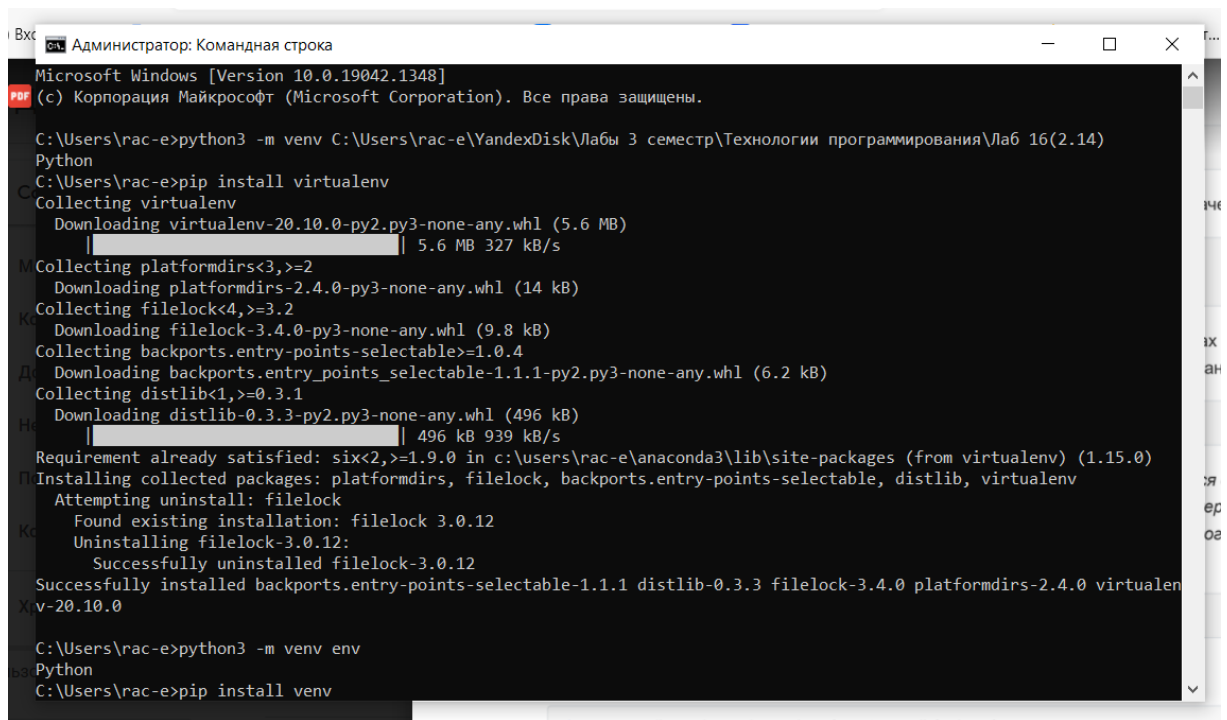
Работа защищена « _____ » 20__ г.

Проверил Воронкин Р.А. _____

(подпись)

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

Так как менеджер пакетов уже установлен приступим к установке `venv` (рис. 1).



```
Администратор: Командная строка
Microsoft Windows [Version 10.0.19042.1348]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\rac-e>python3 -m venv C:\Users\rac-e\YandexDisk\Лаб 3 семестр\Технологии программирования\Лаб 16(2.14)\Python
C:\Users\rac-e>pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.10.0-py2.py3-none-any.whl (5.6 MB)
    | 5.6 MB 327 kB/s
Collecting platformdirs<3,>=2
  Downloading platformdirs-2.4.0-py3-none-any.whl (14 kB)
Collecting filelock<4,>=3.2
  Downloading filelock-3.4.0-py3-none-any.whl (9.8 kB)
Collecting backports.entry-points-selectable>=1.0.4
  Downloading backports.entry_points_selectable-1.1.1-py2.py3-none-any.whl (6.2 kB)
Collecting distlib<1,>=0.3.1
  Downloading distlib-0.3.3-py2.py3-none-any.whl (496 kB)
    | 496 kB 939 kB/s
Requirement already satisfied: six<2,>=1.9.0 in c:\users\rac-e\anaconda3\lib\site-packages (from virtualenv) (1.15.0)
Installing collected packages: platformdirs, filelock, backports.entry-points-selectable, distlib, virtualenv
  Attempting uninstall: filelock
    Found existing installation: filelock 3.0.12
    Uninstalling filelock-3.0.12:
      Successfully uninstalled filelock-3.0.12
Successfully installed backports.entry-points-selectable-1.1.1 distlib-0.3.3 filelock-3.4.0 platformdirs-2.4.0 virtualenv-20.10.0

C:\Users\rac-e>python3 -m venv env
Python
C:\Users\rac-e>pip install venv
```

Рисунок 1 – Установка `venv`

Теперь для создания виртуального окружения запусти команду вида – «`python3 -m venv <путь к папке виртуального окружения>`», а также активируем виртуальное окружение(рис. 2).

```
Администратор: Командная строка
Collecting backports.entry-points-selectable>=1.0.4
  Downloading backports.entry_points_selectable-1.1.1-py2.py3-none-any.whl (6.2 kB)
Collecting distlib<1,>=0.3.1
  Downloading distlib-0.3.3-py2.py3-none-any.whl (496 kB)
Requirement already satisfied: six<2,>=1.9.0 in c:\users\rac-e\anaconda3\lib\site-packages (from virtualenv) (1.15.0)
Installing collected packages: platformdirs, filelock, backports.entry-points-selectable, distlib, virtualenv
  Attempting uninstall: filelock
    Found existing installation: filelock 3.0.12
    Uninstalling filelock-3.0.12:
      Successfully uninstalled filelock-3.0.12
Successfully installed backports.entry-points-selectable-1.1.1 distlib-0.3.3 filelock-3.4.0 platformdirs-2.4.0 virtualenv-
v-20.10.0

C:\Users\rac-e>python3 -m venv env
Python
C:\Users\rac-e>python3 -m venv C:\Users\rac-e\YandexDisk\Лабы 3 семестр\Технологии программирования\Лаб 16(2.14)
Python
C:\Users\rac-e>cd C:\Users\rac-e\YandexDisk\Лабы 3 семестр\Технологии программирования\Лаб 16(2.14)

C:\Users\rac-e\YandexDisk\Лабы 3 семестр\Технологии программирования\Лаб 16(2.14)>virtualenv -p python3 env
created virtual environment CPython3.8.5.final.0-64 in 5870ms
  creator CPython3Windows(dest=C:\Users\rac-e\YandexDisk\Лабы 3 семестр\Технологии программирования\Лаб 16(2.14)\env, cl
ear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\rac-e\
AppData\Local\pypa\virtualenv)
    added seed packages: pip==21.3.1, setuptools==58.3.0, wheel==0.37.0
    activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

C:\Users\rac-e\YandexDisk\Лабы 3 семестр\Технологии программирования\Лаб 16(2.14)>
```

Рисунок 2 – Установка виртуального окружения

Просмотрим список пакетных зависимостей (рис. 3).

```
Администратор: Командная строка
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\rac-e\
AppData\Local\pypa\virtualenv)
    added seed packages: pip==21.3.1, setuptools==58.3.0, wheel==0.37.0
    activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

C:\Users\rac-e\YandexDisk\Лабы 3 семестр\Технологии программирования\Лаб 16(2.14)>pip freeze
alabaster==0.7.12
anaconda-client==1.7.2
anaconda-navigator==1.10.0
anaconda-project==0.8.3
argh==0.26.2
argon2-cffi @ file:///C:/ci/argon2-cffi_1596828585465/work
asn1crypto @ file:///tmp/build/80754af9/asn1crypto_1596577642040/work
astroid @ file:///C:/ci/astroid_1592487315634/work
astropy==4.0.2
async-generator==1.10
atomicwrites==1.4.0
attrs @ file:///tmp/build/80754af9/attrs_1604765588209/work
autopep8 @ file:///tmp/build/80754af9/autopep8_1596578164842/work
Babel @ file:///tmp/build/80754af9/babel_1605108370292/work
backcall==0.2.0
backports.entry-points-selectable==1.1.1
backports.functools-lru-cache==1.6.1
backports.shutil-get-terminal-size==1.0.0
backports.tempfile==1.0
backports.weakref==1.0.post1
bcrypt @ file:///C:/ci/bcrypt_1597936263757/work
beautifulsoup4 @ file:///tmp/build/80754af9/beautifulsoup4_1601924105527/work
bitarray @ file:///C:/ci/bitarray_1605065210072/work
bkcharts==0.2
```

Рисунок 3 – пакетные зависимости

Создадим виртуальное окружение с conda(рис. 4) и активируем его(рис. 5).

```
(base) PS C:\Users\rac-e> mkdir %PROJ_NAME%

Каталог: C:\Users\rac-e

Mode                LastWriteTime         Length Name
----                -
d-----          07.12.2021    12:00          %PROJ_NAME%

(base) PS C:\Users\rac-e> cd %PROJ_NAME%
(base) PS C:\Users\rac-e\%PROJ_NAME%> copy NUL > main.py
```

Рисунок 4 – Создание чистой директории и виртуального окружения

```
(base) PS C:\Users\rac-e\%PROJ_NAME%> cd C:\Users\rac-e\
(base) PS C:\Users\rac-e> conda create -n %PROJ_NAME% python=3.7
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.9.2
  latest version: 4.10.3

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\rac-e\anaconda3\envs\%PROJ_NAME%

  added / updated specs:
    - python=3.7
```

Рисунок 5 – Активация окружения conda

Установим пакеты необходимые для реализации проекта (рис. 6).

```
Администратор: Anaconda Powershell Prompt (anaconda3)
+ CategoryInfo          : ParserError: (:) [Invoke-Expression], ParseException
+ FullyQualifiedErrorId : EndSquareBracketExpectedAtEndOfAttribute,Microsoft.PowerShell.Commands.InvokeExpressionCommand

(base) PS C:\Users\rac-e> conda install django, pandas
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\rac-e\anaconda3

added / updated specs:
- django
- pandas

The following packages will be downloaded:

package | build | size
-----|-----|-----
asgiref-3.4.1 | pyhd3eb1b0_0 | 25 KB
conda-4.10.3 | py38haa95532_0 | 2.9 MB
django-3.2.5 | pyhd3eb1b0_0 | 3.1 MB
libpq-12.2 | hb652d5d_1 | 2.7 MB
psycopg2-2.8.6 | py38hcd4344a_1 | 156 KB
sqlparse-0.4.1 | py_0 | 35 KB
-----|-----|-----
Total: | 8.9 MB
```

Рисунок 6 – Установка Django и pandas

Сформируем файл конфигурации виртуального окружения для быстрого развертывания в будущем (рис. 7).

```
Администратор: Anaconda Powershell Prompt (anaconda3)
sqlparse pkgs/main/noarch::sqlparse-0.4.1-py_0

The following packages will be UPDATED:

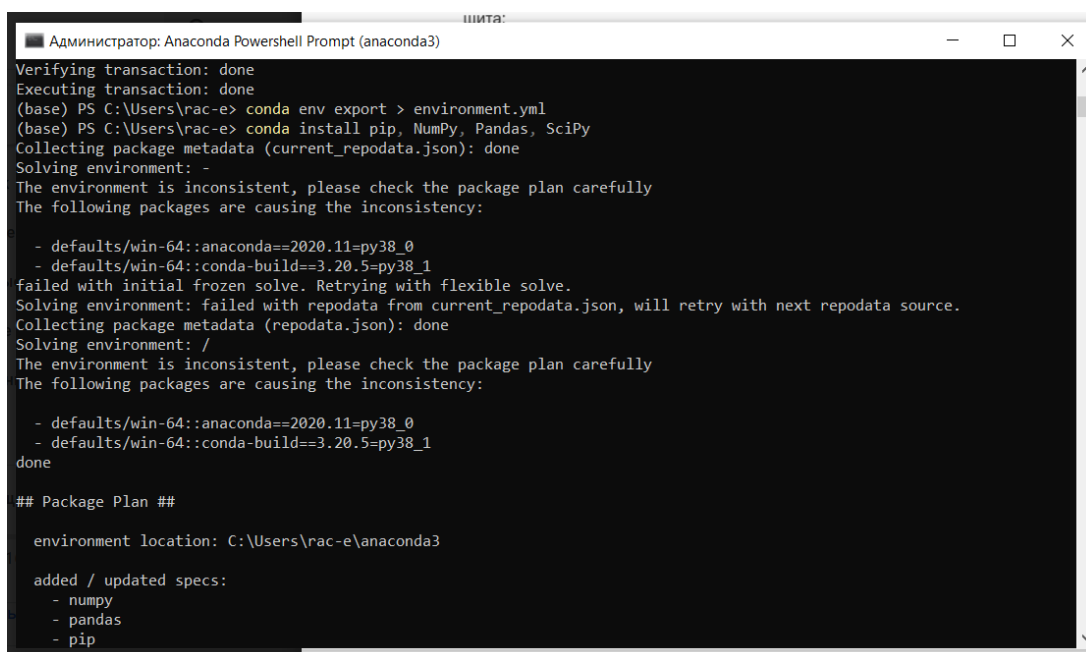
conda 4.9.2-py38haa95532_0 --> 4.10.3-py38haa95532_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
libpq-12.2 | 2.7 MB | ##### | 100%
django-3.2.5 | 3.1 MB | ##### | 100%
asgiref-3.4.1 | 25 KB | ##### | 100%
psycopg2-2.8.6 | 156 KB | ##### | 100%
conda-4.10.3 | 2.9 MB | ##### | 100%
sqlparse-0.4.1 | 35 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) PS C:\Users\rac-e> conda env export > environment.yml
(base) PS C:\Users\rac-e> conda install pip, NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: /
```

Рисунок 7 – Создание файла environment.yml

Установим в виртуальное окружение следующие пакеты: `pip`, `NumPy`, `Pandas`, `SciPy`(рис. 9).



```
Администратор: Anaconda Powershell Prompt (anaconda3)
Verifying transaction: done
Executing transaction: done
(base) PS C:\Users\rac-e> conda env export > environment.yml
(base) PS C:\Users\rac-e> conda install pip, NumPy, Pandas, SciPy
Collecting package metadata (current_repodata.json): done
Solving environment: -
The environment is inconsistent, please check the package plan carefully
The following packages are causing the inconsistency:

- defaults/win-64::anaconda==2020.11=py38_0
- defaults/win-64::conda-build==3.20.5=py38_1
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: /
The environment is inconsistent, please check the package plan carefully
The following packages are causing the inconsistency:

- defaults/win-64::anaconda==2020.11=py38_0
- defaults/win-64::conda-build==3.20.5=py38_1
done

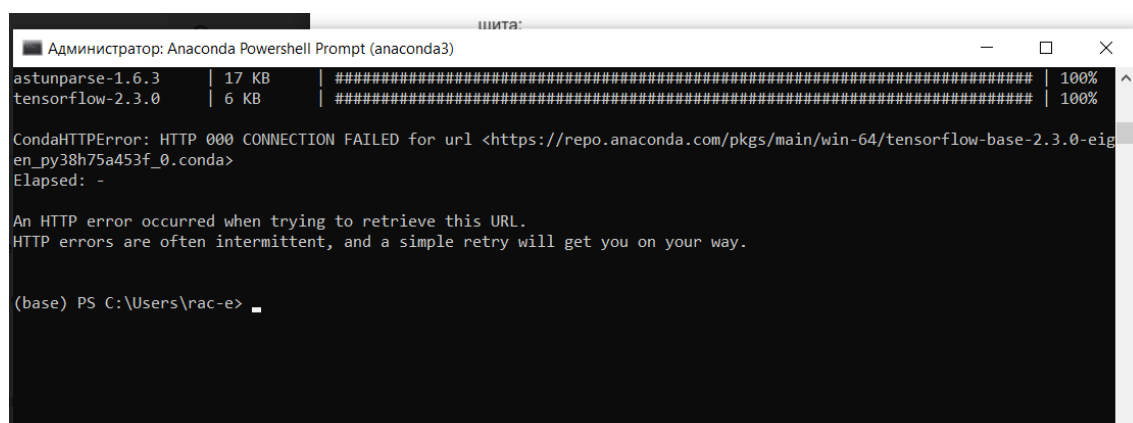
## Package Plan ##

environment location: C:\Users\rac-e\anaconda3

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

Рисунок 8 – Установка пакетов

Установка пакета `TensorFlow` через `conda` завершилась неудачно (рис. 9).



```
Администратор: Anaconda Powershell Prompt (anaconda3)
astunparse-1.6.3 | 17 KB | ##### | 100%
tensorflow-2.3.0 | 6 KB | ##### | 100%

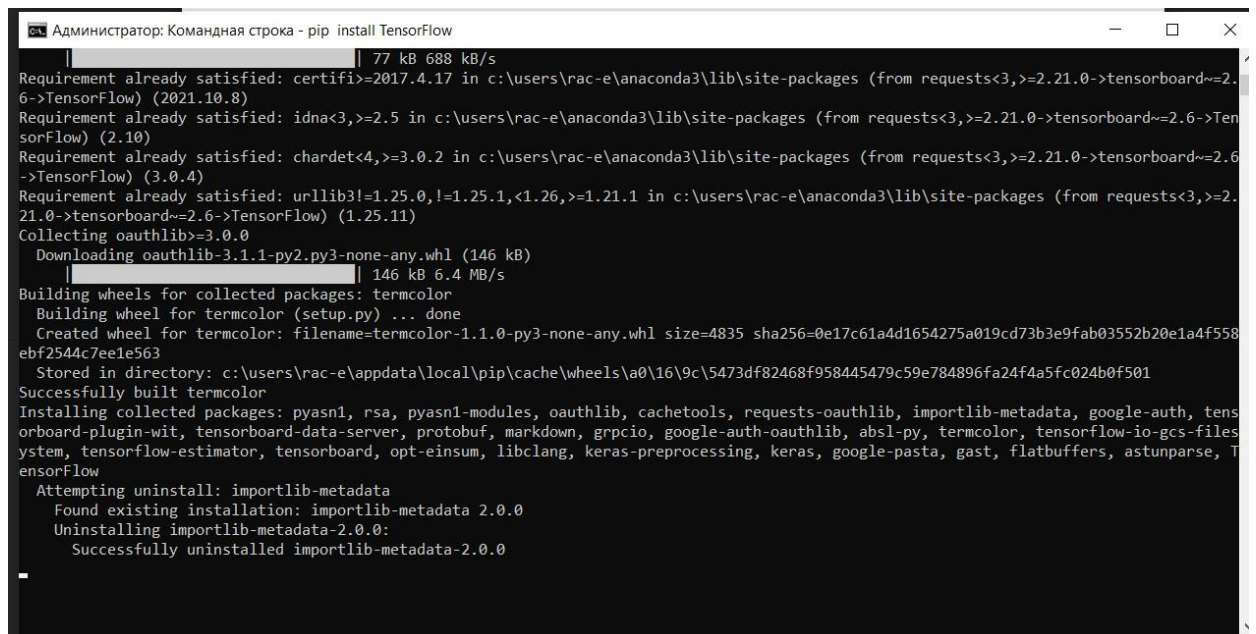
CondaHTTPError: HTTP 000 CONNECTION FAILED for url <https://repo.anaconda.com/pkgs/main/win-64/tensorflow-base-2.3.0-eigen_py38h75a453f_0.conda>
Elapsed: -

An HTTP error occurred when trying to retrieve this URL.
HTTP errors are often intermittent, and a simple retry will get you on your way.

(base) PS C:\Users\rac-e> 
```

Рисунок 9 – Установка TensorFlow через conda

Установим `TensorFlow` через менеджера пакетов `pip` (рис. 10).



```
Администратор: Командная строка - pip install TensorFlow
Requirement already satisfied: certifi<=2017.4.17 in c:\users\rac-e\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->TensorFlow) (2021.10.8)
Requirement already satisfied: idna<3,>=2.5 in c:\users\rac-e\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->TensorFlow) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\rac-e\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->TensorFlow) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!>=1.25.1,<1.26,>=1.21.1 in c:\users\rac-e\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->TensorFlow) (1.25.11)
Collecting oauthlib>=3.0.0
  Downloading oauthlib-3.1.1-py2.py3-none-any.whl (146 kB)
    | 146 kB 6.4 MB/s
Building wheels for collected packages: termcolor
  Building wheel for termcolor (setup.py) ... done
  Created wheel for termcolor: filename=termcolor-1.1.0-py3-none-any.whl size=4835 sha256=0e17c61a4d1654275a019cd73b3e9fab03552b20e1a4f558ebf2544c7ee1e563
  Stored in directory: c:\users\rac-e\appdata\local\pip\cache\wheels\16\9c\5473df82468f958445479c59e784896fa24f4a5fc024b0f501
Successfully built termcolor
Installing collected packages: pyasn1, rsa, pyasn1-modules, oauthlib, cachetools, requests-oauthlib, importlib-metadata, google-auth, tensorboard-plugin-wit, tensorboard-data-server, protobuf, markdown, grpcio, google-auth-oauthlib, absl-py, termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator, tensorboard, opt-einsum, libclang, keras-preprocessing, keras, google-pasta, gast, flatbuffers, astunparse, TensorFlow
Attempting uninstall: importlib-metadata
  Found existing installation: importlib-metadata 2.0.0
  Uninstalling importlib-metadata-2.0.0:
    Successfully uninstalled importlib-metadata-2.0.0
```

Рисунок 10 – Установка прошла успешно

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 виртуального окружения для работы.

Работаем в виртуальном окружении, а именно управляем пакетами используя `pip` и запускаем выполнение кода.

Деактивируем после окончания работы виртуальное окружение.

Удаляем папку с виртуальным окружением, если оно нам больше не нужно.

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

С его помощью можно создать виртуальную среду, в которую можно устанавливать пакеты независимо от основной среды или других виртуальных окружений. Основные действия с виртуальными окружениями с помощью `venv`: создание виртуального окружения, его активация и деактивация.

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

Для начала пакет нужно установить. Установку можно выполнить командой: `python3 -m pip install virtualenv` `Virtualenv` позволяет создать абсолютно изолированное виртуальное окружение для каждой из программ. Окружением является обычная директория, которая содержит копию всего

необходимого для запуска определенной программы, включая копию самого интерпретатора, полной стандартной библиотеки, `pip`, и, что самое главное, копии всех необходимых пакетов.

15. Изучите работу с виртуальными окружениями `pipenv`. Как осуществляется работа с виртуальными окружениями `pipenv`?

Для формирования и развертывания пакетных зависимостей используется утилита `pip`.

Основные возможности `pipenv`:

- Создание и управление виртуальным окружением
- Синхронизация пакетов в `Pipfile` при установке и удалении пакетов
- Автоматическая подгрузка переменных окружения из `.env` файла

После установки `pipenv` начинается работа с окружением. Его можно создать в любой папке. Достаточно установить любой пакет внутри папки. Используем `requests`, он автоматически установит окружение и создаст `Pipfile` и `Pipfile.lock`.

16. Каково назначение файла `requirements.txt`? Как создать этот файл? Какой он имеет формат?

Установить пакеты можно с помощью команды: `pip install -r requirements.txt`. Также можно использовать команду `pip freeze > requirements.txt`, которая создаст `requirements.txt` наполнив его названиями и версиями тех пакетов что используются вами в текущем окружении. Это

удобно если вы разработали проект и в текущем окружении все работает, но вы хотите перенести проект в иное окружение (например, заказчику или на сервер). С помощью закрепления зависимостей мы можем быть уверены, что пакеты, установленные в нашей производственной среде, будут точно соответствовать пакетам в нашей среде разработки, чтобы ваш проект неожиданно не ломался.

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

Conda способна управлять пакетами как для Python, так и для C/ C++, R, Ruby, Lua, Scala и других. Conda устанавливает двоичные файлы, поэтому работу по компиляции пакета самостоятельно выполнять не требуется (по сравнению с pip).

18. В какие дистрибутивы Python входит пакетный менеджер conda?

Все чаще среди Python-разработчиков заходит речь о менеджере пакетов conda, включенный в состав дистрибутивов Anaconda и Miniconda. JetBrains включил этот инструмент в состав PyCharm.

19. Как создать виртуальное окружение conda?

С помощью команды: `conda create -n %PROJ_NAME% python=3.7`

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

Чтобы установить пакеты, необходимо воспользоваться командой: — `conda install A` для активации: `conda activate %PROJ_NAME%`

21. Как деактивировать и удалить виртуальное окружение conda?

Для деактивации использовать команду: `conda deactivate`, а для удаления: `conda remove -n $PROJ_NAME`.

22. Каково назначение файла `environment.yml` ? Как создать этот файл?

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.

Вывод: в ходе занятия были приобретены навыки по работе с менеджером пакетов pip и виртуальными окружениями с помощью языка программирования Python версии 3.x.