

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.14
дисциплины «Программирование на Python»

Выполнил:
Кенесбаев Хилол Куат улы
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

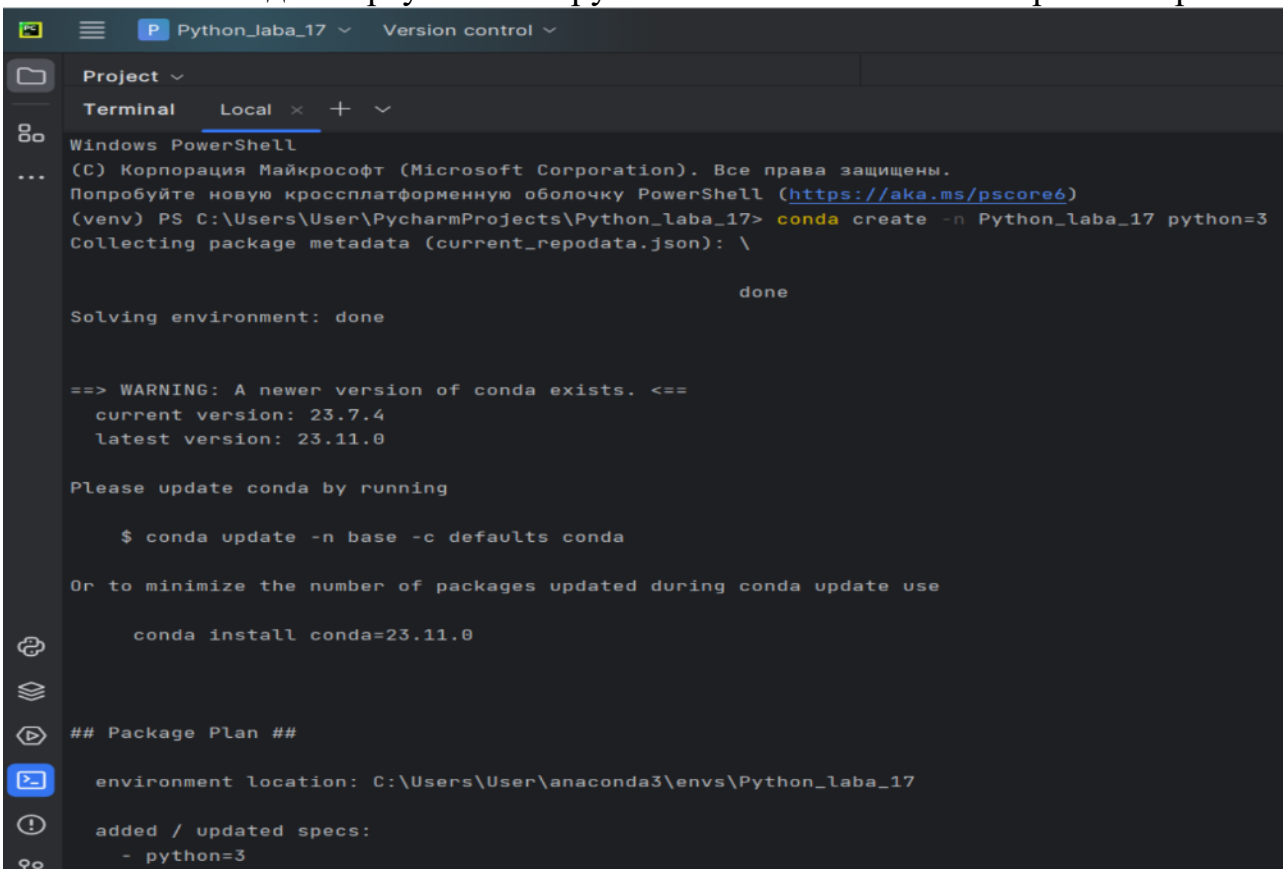
Ставрополь, 2023 г.

Тема: Установка пакетов в Python. Виртуальные окружения

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

Порядок выполнения работы:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.
2. Создал виртуальное акружение Anaconda с именем репозитория:



```
Python_Laba_17 Version control
Terminal Local x + v
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> conda create -n Python_laba_17 python=3
Collecting package metadata (current_repodata.json): \
done

Solving environment: done

==> WARNING: A newer version of conda exists. <==
current version: 23.7.4
latest version: 23.11.0

Please update conda by running

$ conda update -n base -c defaults conda

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

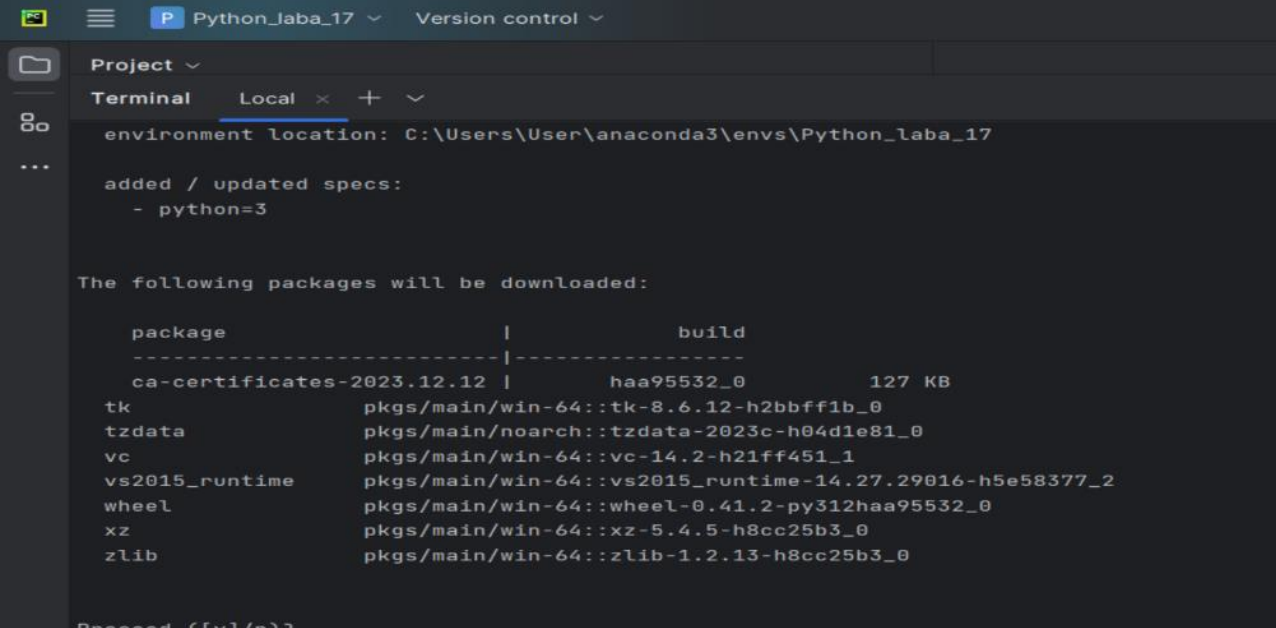
conda install conda=23.11.0

## Package Plan ##

environment location: C:\Users\User\anaconda3\envs\Python_laba_17

added / updated specs:
- python=3
```

Рисунок 1 Создание conda окружения



```
environment location: C:\Users\User\anaconda3\envs\Python_laba_17

added / updated specs:
- python=3

The following packages will be downloaded:

package | build
-----|-----
ca-certificates-2023.12.12 | haa95532_0 127 KB
tk | pkgs/main/win-64::tk-8.6.12-h2bbff1b_0
tzdata | pkgs/main/noarch::tzdata-2023c-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/win-64::wheel-0.41.2-py312haa95532_0
xz | pkgs/main/win-64::xz-5.4.5-h8cc25b3_0
zlib | pkgs/main/win-64::zlib-1.2.13-h8cc25b3_0

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

Рисунок 2 Предустановленные пакеты

3. Установил пакеты с помощью conda install:



```
Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
# $ conda activate Python_laba_17
#
# To deactivate an active environment, use
#
# $ conda deactivate
```

Рисунок 3 Установка пакетов

4. Попробуйте установить менеджером пакетов conda пакет TensorFlow. Возникает ли при этом ошибка? Попробуйте выявить и укажите причину этой ошибки:

```
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> conda install TensorFlow
Collecting package metadata (current_repodata.json): done
Solving environment: unsuccessful initial attempt using frozen solve. Retrying with flexible solve.
Solving environment: unsuccessful attempt using repodata from current_repodata.json, retrying with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: unsuccessful initial attempt using frozen solve. Retrying with flexible solve.
Solving environment: \
Found conflicts! Looking for incompatible packages.
This can take several minutes. Press CTRL-C to abort.
failed

UnsatisfiableError: The following specifications were found
to be incompatible with the existing python installation in your environment:

Specifications:

- tensorflow -> python[version='3.10.*|3.9.*|3.8.*|3.7.*|3.6.*|3.5.*']

Your python: python=3.11

If python is on the left-most side of the chain, that's the version you've asked for.
When python appears to the right, that indicates that the thing on the left is somehow
not available for the python version you are constrained to. Note that conda will not
change your python version to a different minor version unless you explicitly specify
that.
```

Рисунок 4 Попытка установки пакета TensorFlow

Пакет TensorFlow совместим с Python 3.8, для решения возможной проблемы необходимо понизить версию python с помощью команды:

`conda install python=3.7`

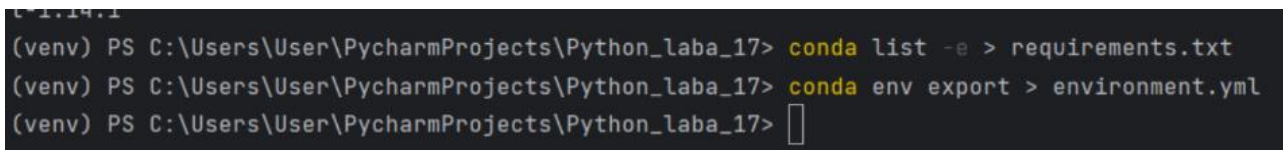
5. Установил пакет TensorFlow с помощью менеджера пакетов pip:

```
Python_laba_17 Version control Current File
Project
Terminal Local x + v
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-2.15.0-cp311-cp311-win_amd64.whl.metadata (3.6 kB)
Collecting tensorflow-intel==2.15.0 (from tensorflow)
  Downloading tensorflow_intel-2.15.0-cp311-cp311-win_amd64.whl.metadata (5.1 kB)
Collecting absl-py>=1.0.0 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading absl_py-2.0.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse>=1.6.0 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting flatbuffers>=23.5.26 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading flatbuffers-23.5.26-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast!=0.5.0,!0.5.1,!0.5.2,>=0.2.1 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading gast-0.5.4-py3-none-any.whl (19 kB)
Collecting google-pasta>=0.1.1 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
  57.5/57.5 kB 761.8 kB/s eta 0:00:00
Collecting h5py>=2.9.0 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading h5py-3.10.0-cp311-cp311-win_amd64.whl.metadata (2.5 kB)
Collecting libclang>=13.0.0 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading libclang-16.0.6-py2.py3-none-win_amd64.whl.metadata (5.3 kB)
Collecting ml-dtypes==0.2.0 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading ml_dtypes-0.2.0-cp311-cp311-win_amd64.whl.metadata (20 kB)
Collecting numpy<2.0.0,>=1.23.5 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading numpy-1.26.2-cp311-cp311-win_amd64.whl.metadata (61 kB)
  61.2/61.2 kB 3.4 MB/s eta 0:00:00
Collecting opt-einsum>=2.3.2 (from tensorflow-intel==2.15.0->tensorflow)
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
  65.5/65.5 kB 3.7 MB/s eta 0:00:00
Collecting packaging (from tensorflow-intel==2.15.0->tensorflow)
  Downloading packaging-23.2-py3-none-any.whl.metadata (3.2 kB)
Collecting protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4.21.4,!4.21.5,<5.0.0dev,>=3.20.3 (from tensorflow-intel==2.15.0->tensorflow)
```

Рисунок 5 Установка TensorFlow при помощи pip

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

Проанализируйте содержимое этих файлов:

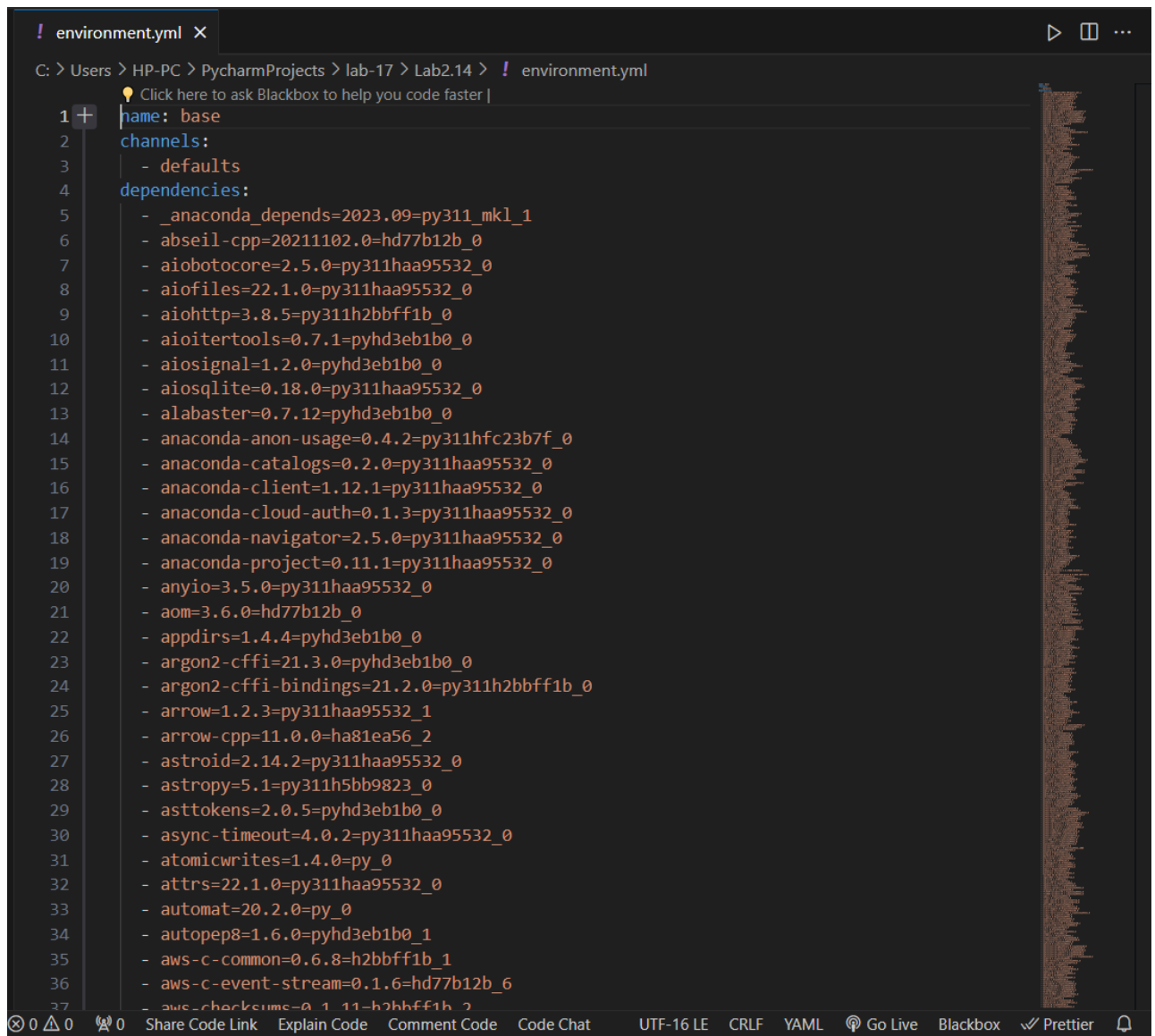


```
C:\1.14.1  
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> conda list -e > requirements.txt  
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> conda env export > environment.yml  
(venv) PS C:\Users\User\PycharmProjects\Python_laba_17> 
```

Рисунок 6 Формирование файлов requirements.txt и environment.yml

```
requirements.txt – Блокнот
Файл Правка Формат Вид Справка
# This file may be used to create an environment using:
# $ conda create --name <env> --file <this file>
# platform: win-64
_anaconda_depends=2023.09=py311_mkl_1
abseil-cpp=20211102.0=hd77b12b_0
aiobotocore=2.5.0=py311haa95532_0
aiofiles=22.1.0=py311haa95532_0
aiohttp=3.8.5=py311h2bbff1b_0
aioitertools=0.7.1=pyhd3eb1b0_0
aiosignal=1.2.0=pyhd3eb1b0_0
aiosqlite=0.18.0=py311haa95532_0
alabaster=0.7.12=pyhd3eb1b0_0
anaconda-anon-usage=0.4.2=py311hfc23b7f_0
anaconda-catalogs=0.2.0=py311haa95532_0
anaconda-client=1.12.1=py311haa95532_0
anaconda-cloud-auth=0.1.3=py311haa95532_0
anaconda-navigator=2.5.0=py311haa95532_0
anaconda-project=0.11.1=py311haa95532_0
anyio=3.5.0=py311haa95532_0
aom=3.6.0=hd77b12b_0
appdirs=1.4.4=pyhd3eb1b0_0
argon2-cffi=21.3.0=pyhd3eb1b0_0
argon2-cffi-bindings=21.2.0=py311h2bbff1b_0
arrow=1.2.3=py311haa95532_1
arrow-cpp=11.0.0=ha81ea56_2
astroid=2.14.2=py311haa95532_0
astropy=5.1=py311h5bb9823_0
asttokens=2.0.5=pyhd3eb1b0_0
async-timeout=4.0.2=py311haa95532_0
atomicwrites=1.4.0=py_0
attrs=22.1.0=py311haa95532_0
automat=20.2.0=py_0
autopep8=1.6.0=pyhd3eb1b0_1
aws-c-common=0.6.8=h2bbff1b_1
aws-c-event-stream=0.1.6=hd77b12b_6
aws-checksums=0.1.11=h2bbff1b_2
aws-sdk-cpp=1.8.185=hd77b12b_1
babel=2.11.0=py311haa95532_0
backcall=0.2.0=pyhd3eb1b0_0
backports=1.1=pyhd3eb1b0_0
backports.functools_lru_cache=1.6.4=pyhd3eb1b0_0
backports.tempfile=1.0=pyhd3eb1b0_1
Стр 1, столб 1 100% Windows (CRLF) UTF-16 LE
```

Рисунок 7 Файл requirements.txt



```
1 + name: base
2 channels:
3   - defaults
4 dependencies:
5   - _anaconda_depends=2023.09=py311_mkl_1
6   - abseil-cpp=20211102.0=hd77b12b_0
7   - aiobotocore=2.5.0=py311haa95532_0
8   - aiofiles=22.1.0=py311haa95532_0
9   - aiohttp=3.8.5=py311h2bbff1b_0
10  - aioitertools=0.7.1=pyhd3eb1b0_0
11  - aiosignal=1.2.0=pyhd3eb1b0_0
12  - aiosqlite=0.18.0=py311haa95532_0
13  - alabaster=0.7.12=pyhd3eb1b0_0
14  - anaconda-anon-usage=0.4.2=py311hfc23b7f_0
15  - anaconda-catalogs=0.2.0=py311haa95532_0
16  - anaconda-client=1.12.1=py311haa95532_0
17  - anaconda-cloud-auth=0.1.3=py311haa95532_0
18  - anaconda-navigator=2.5.0=py311haa95532_0
19  - anaconda-project=0.11.1=py311haa95532_0
20  - anyio=3.5.0=py311haa95532_0
21  - aom=3.6.0=hd77b12b_0
22  - appdirs=1.4.4=pyhd3eb1b0_0
23  - argon2-cffi=21.3.0=pyhd3eb1b0_0
24  - argon2-cffi-bindings=21.2.0=py311h2bbff1b_0
25  - arrow=1.2.3=py311haa95532_1
26  - arrow-cpp=11.0.0=ha81ea56_2
27  - astroid=2.14.2=py311haa95532_0
28  - astropy=5.1=py311h5bb9823_0
29  - asttokens=2.0.5=pyhd3eb1b0_0
30  - async-timeout=4.0.2=py311haa95532_0
31  - atomicwrites=1.4.0=py_0
32  - attrs=22.1.0=py311haa95532_0
33  - automat=20.2.0=py_0
34  - autopep8=1.6.0=pyhd3eb1b0_1
35  - aws-c-common=0.6.8=h2bbff1b_1
36  - aws-c-event-stream=0.1.6=hd77b12b_6
37  - aws-c-http=0.1.11=h2bbff1b_2
```

Рисунок 8 Файл environment.yml

В файле requirements.txt хранятся зависимости созданные pip, а в файле environment.yml хранятся параметры окружения conda.

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. Там также есть возможность выкладывать свои пакеты. Для скачивания и установки используется специальная утилита, которая называется `pip`.

2. Как осуществить установку менеджера пакетов `pip`?

При развертывании современной версии Python (начиная с Python 2.7.9 и Python 3.4), `pip` устанавливается автоматически. Но если, по какой-то причине, `pip` не установлен на вашем ПК, то сделать это можно вручную.

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

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

и выполните его.

```
$ python get-pip.py
```

При этом, вместе с `pip` будут установлены `setuptools` и `wheels`. `Setuptools` – это набор инструментов для построения пакетов Python. `Wheels` – это формат дистрибутива для пакета Python.

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`?

```
$ pip install ProjectName
```

5. Как установить заданную версию пакета с помощью `pip`?

```
$ pip install ProjectName==*
```

6. Как установить пакет из `git` репозитория (в том числе GitHub) с помощью `pip`?


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

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

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

8. Как удалить установленный пакет с помощью pip?

```
$ pip uninstall ProjectName
```

9. Как обновить установленный пакет с помощью pip?

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

10. Как отобразить список установленных пакетов с помощью pip?

```
$ pip list
```

11. Каковы причины появления виртуальных окружений в языке Python?

В системе для интерпретатора Python может быть установлена глобально только одна версия пакета. Это порождает ряд проблем: проблема обратной совместимости и проблема коллективной разработки. Получается, что для каждого проекта нужна своя "песочница", которая изолирует зависимости. Такая "песочница" придумана и называется "виртуальным окружением" или "виртуальной средой".

12. Каковы основные этапы работы с виртуальными окружениями?

- 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?

Для создания виртуального окружения достаточно дать команду в формате: `python3 -m venv <путь к папке виртуального окружения>`

Обычно папку для виртуального окружения называют `env` или `venv`. В описании команды выше явно указан интерпретатор версии 3.x. Под Windows и некоторыми другими операционными системами это будет просто `python`.

Чтобы активировать виртуальное окружение нужно:

```
$ source env/bin/activate
```

В Windows мы вызываем скрипт активации напрямую.

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

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

```
$ deactivate
```

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

Для начала пакет нужно установить. Установку можно выполнить командой:

```
# Для python 3
```

```
python3 -m pip install virtualenv
```

```
# Для единственного python
```

```
python -m pip install virtualenv
```

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

папки окружения env:

```
virtualenv -p python3 env
```

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

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

`pipenv install` – Создание виртуального окружения

`pipenv install <package>` – Установка определённого пакета и добавление его в `Pipfile`.

`pipenv uninstall <package>` – Удаление установленного пакета и его исключение из `Pipfile`.

`pipenv shell` – Активация виртуального окружения.

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

Просмотреть список зависимостей мы можем командой: `pip freeze`

Что бы его сохранить, нужно перенаправить вывод команды в файл:

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

Имя файла хранения зависимостей `requirements.txt` выбрано не зря. Оно является стандартной договорённостью и используется некоторыми утилитами автоматически.

Установка пакетов из файла зависимостей в новом виртуальном окружении так же выполняется одной командой: `pip install -r requirements.txt`

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

Основная проблема заключается в том, что `pip`, `easy_install` и `virtualenv` ориентированы на Python. Эти инструменты игнорируют библиотеки зависимостей, реализованные с использованием других языков. Например, XSLT, HDF5, MKL и другие, которые не имеют `setup.py` в исходном коде и

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

Существуют также некоторые различия, если вы заинтересованы в создании собственных пакетов. Например, pip создан на основе setuptools, тогда как conda использует свой собственный формат, который имеет некоторые преимущества (например, статическая компиляция пакета).

18. В какие дистрибутивы Python входит пакетный менеджер conda? Anaconda и Miniconda.

19. Как создать виртуальное окружение conda?

Начиная проект, создайте чистую директорию и дайте ей понятное короткое имя. Для Linux это будет соответствовать набору команд:

```
mkdir $PROJ_NAME
```

```
cd $PROJ_NAME
```

```
touch README.md main.py
```

Создайте чистое conda-окружение с таким же именем: conda create -n \$PROJ_NAME python=3.7

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

```
conda activate $PROJ_NAME
```

```
conda install $PACKAGE_NAME
```

21. Как деактивировать и удалить виртуальное окружение conda?

```
conda deactivate
```

```
conda remove -n $PACKAGE_NAME
```

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

Файл `environment.yml` позволит воссоздать окружение в любой нужный момент.

```
conda env export > environment.yml
```

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

24. Самостоятельно изучите средства IDE PyCharm для работы с виртуальными окружениями conda. Опишите порядок работы с виртуальными окружениями conda в IDE PyCharm.

Необходимо установить Anaconda или Miniconda.

В Pycharm необходимо настроить интерпретатор Python:

Нужно перейти в `File > Settings` (для Windows/Linux) или `PyCharm > Preferences` (для macOS).

В левой части окна настроек выбрать `Project: ваш_проект > Python Interpreter`.

Нажать на шестерёнку справа от списка интерпретаторов и выбрать `Add`. В открывшемся окне добавления интерпретатора выбрать `Conda Environment`.

Можно либо создать новое окружение, выбрав `New environment`, либо использовать существующее, выбрав `Existing environment`.

Создание нового окружения Conda:

Необходимо указать имя окружения, версию Python и нажать кнопку `ОК`.

PyCharm автоматически создаст новое окружение Conda и установит в него выбранную версию Python.

Использование существующего окружения Conda:

Нужно нажать на кнопку с тремя точками и найти путь к существующему окружению Conda.

Активация окружения Conda:

При использовании терминала в PyCharm окружение Conda должно активироваться автоматически. Если этого не произошло, его можно активировать вручную, введя команду `conda activate имя_окружения` в терминале.

Работа с проектом:

После настройки окружения Conda можно работать с проектом в PyCharm, как обычно.

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 результате выполнения работы были приобретены навыки по работе с менеджером пакетов `pip` и виртуальными окружениями с помощью языка программирования Python версии 3.x.