МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №2.14 по дисциплине «Основы программной инженерии»

Выполнил:
Чернова Софья Андреевна,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г

1. Ход работы

```
C:\Users\User> conda create -n Lab_2.14
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.10.3
  latest version: 4.12.0

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: D:\Compilators\Python\Anaconda\envs\Lab_2.14



Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate Lab_2.14
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

Рисунок 1 – создание виртуального окружения

```
C:\Users\User> conda activate Lab_2.14

(Lab_2.14) C:\Users\User>
```

Рисунок 2 – активация окружения

```
(Lab_2.14) C:\Users\User> conda install -n Lab_2.14 pip
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.10.3
  latest version: 4.12.0

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: D:\Compilators\Python\Anaconda\envs\Lab_2.14

  added / updated specs:
    - pip


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2022.3.29  |       haa95532_0         122 KB
    certifi-2021.10.8          |   py39haa95532_2         152 KB
    openssl-1.1.1n             |       h2bbff1b_0         4.8 MB
    python-3.9.12              |       h6244533_0        17.1 MB
    sqlite-3.38.2              |       h2bbff1b_0         807 KB
    tzdata-2022a               |       hda174b7_0         109 KB
    wheel-0.37.1               |     pyhd3eb1b0_0          33 KB
    ------------------------------------------------------------
                                           Total:        23.1 MB
```

Рисунок 3 – установка пакета pip

```
The following NEW packages will be INSTALLED:

  ca-certificates    pkgs/main/win-64::ca-certificates-2022.3.29-haa95532_0
  certifi            pkgs/main/win-64::certifi-2021.10.8-py39haa95532_2
  openssl            pkgs/main/win-64::openssl-1.1.1n-h2bbff1b_0
  pip                pkgs/main/win-64::pip-21.2.4-py39haa95532_0
  python             pkgs/main/win-64::python-3.9.12-h6244533_0
  setuptools         pkgs/main/win-64::setuptools-58.0.4-py39haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.38.2-h2bbff1b_0
  tzdata             pkgs/main/noarch::tzdata-2022a-hda174b7_0
  vc                 pkgs/main/win-64::vc-14.2-h21ff451_1
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
  wheel              pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0
  wincertstore       pkgs/main/win-64::wincertstore-0.2-py39haa95532_2


Proceed ([y]/n)? y


Downloading and Extracting Packages
ca-certificates-2022 | 122 KB    | ############################################################ | 100%
tzdata-2022a         | 109 KB    | ############################################################ | 100%
python-3.9.12        | 17.1 MB   | ############################################################ | 100%
sqlite-3.38.2        | 807 KB    | ############################################################ | 100%
openssl-1.1.1n       | 4.8 MB    | ############################################################ | 100%
certifi-2021.10.8    | 152 KB    | ############################################################ | 100%
wheel-0.37.1         | 33 KB     | ############################################################ | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Рисунок 4 – установка пакета pip (окончание)

```
(Lab_2.14) C:\Users\User> conda install -n Lab_2.14 TensorFlow
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata sour
ce.
Collecting package metadata (repodata.json): done
Solving environment: done
```

Рисунок 5 – ошибка при установке пакета TensorFlow

environment.yml

```
name: Lab_2.14
channels:
  - defaults
dependencies:
  - _tflow_select=2.3.0=mkl
  - abseil-cpp=20210324.2=hd77b12b_0
  - absl-py=0.15.0=pyhd3eb1b0_0
  - aiohttp=3.8.1=py39h2bbff1b_1
  - aiosignal=1.2.0=pyhd3eb1b0_0
  - astor=0.8.1=py39haa95532_0
  - astunparse=1.6.3=py_0
  - async-timeout=4.0.1=pyhd3eb1b0_0
  - attrs=21.4.0=pyhd3eb1b0_0
  - blas=1.0=mkl
  - blinker=1.4=py39haa95532_0
  - bottleneck=1.3.4=py39h080aedc_0
  - brotlipy=0.7.0=py39h2bbff1b_1003
  - ca-certificates=2022.3.29=haa95532_0
  - cachetools=4.2.2=pyhd3eb1b0_0
  - certifi=2021.10.8=py39haa95532_2
  - cffi=1.15.0=py39h2bbff1b_1
  - charset-normalizer=2.0.4=pyhd3eb1b0_0
  - click=8.0.4=py39haa95532_0
  - colorama=0.4.4=pyhd3eb1b0_0
  - cryptography=3.4.8=py39h71e12ea_0
  - dataclasses=0.8=pyh6d0b6a4_7
  - flatbuffers=2.0.0=h6c2663c_0
  - frozenlist=1.2.0=py39h2bbff1b_0
  - gast=0.4.0=pyhd3eb1b0_0
  - giflib=5.2.1=h62dcd97_0
  - google-auth=2.6.0=pyhd3eb1b0_0
  - google-auth-oauthlib=0.4.1=py_2
  - google-pasta=0.2.0=pyhd3eb1b0_0
  - grpcio=1.42.0=py39hc60d5dd_0
  - h5py=3.6.0=py39h3de5c98_0
  - hdf5=1.10.6=h7ebc959_0
```

Рисунок 6 – содержимое файла environment.yml

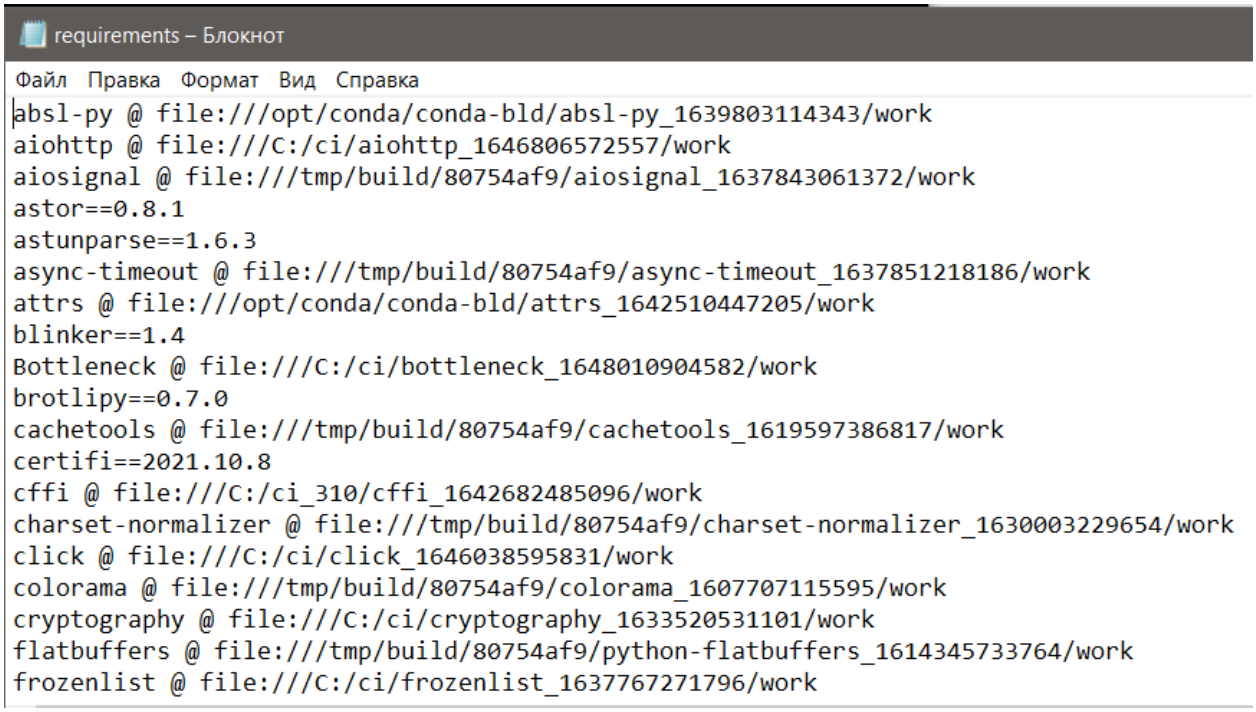

Рисунок 7 – содержимое файла requirements.txt

2. 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. 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.

3) Откуда менеджер пакетов pip по умолчанию устанавливает пакеты?

По умолчанию это Python Package Index (PyPI) – репозиторий, открытый для всех Python разработчиков.

4) Как установить последнюю версию пакета с помощью pip?

$ pip install ProjectName

5) Как установить заданную версию пакета с помощью pip?

$ pip install ProjectName==3.2

$ pip install ProjectName>=3.1

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

$ pip show ProjectName

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

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. Чем опасно обновление пакетов или версий интерпретатора? В новой версии пакета могут измениться названия функций или методов объектов и число и/или порядок передаваемых в них параметров. В следующей версии интерпретатора могут появиться новые ключевые слова, которые совпадают с именами переменных уже существующих приложений. Эта же проблема затрагивает и процесс разработки. Обычно программист работает со множеством проектов, так как приходится поддерживать созданные ранее и не редко даже унаследованные от других: когда-то веб приложения создавались для одной версии фреймворка, а сегодня уже вышла новая, но переписывать все долго и дорого, поэтому проект и дальше поддерживается для старой.

- 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. Если вы уже сталкивались с этой проблемой, то уже задумались, 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?

Вот основные этапы работы с виртуальным окружением:

- ✓ 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?

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

Создадим виртуальное окружение в папке проекта. Для этого перейдём в корень любого проекта на Python >= 3.3 и дадим команду:

$ python3 -m venv env

После её выполнения создастся папка env с виртуальным окружением.

Чтобы активировать виртуальное окружение под Windows команда выглядит иначе: > env\\Scripts\\activate

При размещении виртуального окружения в папке проекта стоит позаботится об его исключении из репозитория системы управления версиями. Для этого, например, при использовании Git нужно добавить папку в файл .gitignore. Это делается для того, чтобы не засорять проект разными вариантами виртуального окружения.

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

$ deactivate

$ source /home/user/envs/project1_env2/bin/activate

Команда deactivate всегда выполняется из контекста текущего виртуального окружения. По этой причине для неё не нужно указывать полный путь. Ранее мы работали с интерпретаторами доступными через стандартные вызовы python. Если в системе установлена другая версия интерпретатора, которая не добавлена в переменную PATH, то для создания виртуального окружения нужно явно задать путь до исполняемого файла интерпретатора. Например, у нас Windows и ранее мы установили Python версии 3.7.6. Сегодня мы решили ознакомится с особенностями Python версии 3.8.2, так как нужно расти в качестве разработчика. Для этого скачали и установили новую версию в папку C:\Python38, но при этом не добавили в PATH (не установили соответствующую галочку в программе установки).

Для создания виртуального окружения для Python 3.8.2 нам придется дать в папке проекта команду с указанием полного пути до интерпретатора, например: > C:\\Python38\\python -m venv env

После выполнения команды мы получим виртуальное окружение для

Python 3.8.2. Активация же будет происходить стандартно для Windows:

> env\\Scripts\\activate

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: > env\\Scripts\\activate

(env) > deactivate

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

Грубо говоря, pipenv можно рассматривать как симбиоз утилит pip и venv (или virtualenv), которые работают вместе, пряча многие неудобные детали от конечного пользователя.

Помимо этого pipenv ещё умеет вот такое:

- автоматически находить интерпретатор Python нужной версии (находит даже интерпретаторы, установленные через pyenv и asdf!);
- запускать вспомогательные скрипты для разработки;
- загружать переменные окружения из файла .env;
- проверять зависимости на наличие известных уязвимостей.

Стоит сразу оговориться, что если вы разрабатываете библиотеку (или что-то, что устанавливается через pip, и должно работать на нескольких версиях интерпретатора), то pipenv — не ваш путь. Этот инструмент создан в первую очередь для разработчиков конечных приложений (консольных утилит, микросервисов, веб-сервисов). Формат хранения зависимостей подразумевает работу только на одной конкретной версии интерпретатора (это имеет смысл для конечных приложений, но для библиотек это, как правило, не приемлемо). Для разработчиков библиотек существует другой прекрасный инструмент — poetry.

Установка на Windows, самый простой способ — это установка в домашнюю директорию пользователя:

$ pip install --user pipenv

Теперь проверим установку:

$ pipenv --version

pipenv, version 2018.11.26

Если вы получили похожий вывод, значит, всё в порядке.

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

Просмотреть список зависимостей мы можем командой:

pip freeze > requirements.txt

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

Установка пакетов из файла зависимостей в новом виртуальном окружении так же выполняется одной командой:

pip install -r requirements.txt

Все пакеты, которые вы установили перед выполнением команды и предположительно использовали в каком-либо проекте, будут перечислены в файле с именем «requirements.txt». Кроме того, будут указаны их точные версии. Расширение: .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, XSL, 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, miniconda и PyCharm.

19) Как создать виртуальное окружение conda?

Начиная проект, создайте чистую директорию и дайте ей понятное короткое имя. Для Windows, если используется дистрибутив Anaconda, то необходимо вначале запустить консоль Anaconda Powershell Prompt. Делается это из системного меню, посредством выбора следующих пунктов: Пуск

Anaconda3 (64-bit) Anaconda Powershell Prompt (Anaconda3). В результате будет отображено окно консоли, показанное на рисунке. Обратите внимание на имя виртуального окружения по умолчанию, которым в данном случае является base. В этом окне необходимо ввести следующую последовательность команд:

mkdir %PROJ_NAME%

cd %PROJ_NAME%

copy NUL > main.py

Здесь PROJ_NAME - это переменная окружения, в которую записано имя проекта. Допускается не использовать переменные окружения, а использовать имя проекта вместо $PROJ_NAME или %PROJ_NAME% .

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

conda create -n %PROJ_NAME% python=3.7

conda activate %PROJ_NAME%

Установите пакеты, необходимые для реализации проекта.

conda install django, pandas

21) Как деактивировать и удалить виртуальное окружение conda?

Деактивация - conda deactivate

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.

conda env export > environment.yml

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

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.

Метод первый:

Используйте анаконду для создания новой среды

Убедитесь, что Anaconda или Miniconda загружена и установлена на вашем компьютере, и вам известен путь к ее исполняемому файлу.

Выполните одно из следующих действий:

- Добавьте новый интерпретатор.
- Нажмите Ctrl+Alt+S, чтобы открыть проект

Настройки/Преференции и перейдите в раздел Проект <имя проекта> | Python Interpreter. Затем щелкните значок и выберите «Добавить». В левой части диалогового окна Добавить интерпретатор Python выберите Среда Conda. Следующие действия зависят от того, существовала ли среда Conda ранее.

Если выбрано Новое окружение:

- Укажите расположение новой среды Conda в текстовом поле или щелкните и найдите расположение в вашей файловой системе. Обратите внимание, что каталог, в котором должно быть расположено новое окружение Conda, должен быть пустым!
- Выберите версию Python из списка.
- Укажите расположение исполняемого файла Conda в текстовом поле, или нажмите и найдите расположение в каталоге установки Conda. В основном вы ищете путь, который вы использовали при установке Conda на вашу машину.
- Установите флажок Сделать доступным для всех проектов, если вы хотите повторно использовать эту среду при создании интерпретаторов Python в PyCharm.

Если выбрано существующее окружение:

- Раскройте список Интерпретатор и выберите любой из существующих интерпретаторов. Или нажмите и укажите путь к исполняемому файлу Conda в вашей файловой системе, например,

C:\Users\jetbrains\Anaconda3\python.exe.

- Установите флажок Сделать доступным для всех проектов, если вы хотите повторно использовать эту среду при создании интерпретаторов Python в PyCharm. Нажмите OK, чтобы завершить задачу.

25) Почему файлы requirements.txt и environment.yml должны храниться в репозитории git?

Эти файлы важны так же, как и сам код. Не рекомендуется загружать всё виртуальное окружение проекта в репозиторий, так как всё окружение можно восстановить на другом компьютере благодаря этим файлам.