МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.14

Дисциплина: «Программирование на Python»

Тема: «Установка пакетов в Python. Виртуальные окружения»

Выполнила: студентка 2

курса группы ИВТ-б-о-21-1

Яковлева Елизавета

Андреевна

Ставрополь 2022

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

Практическая часть:

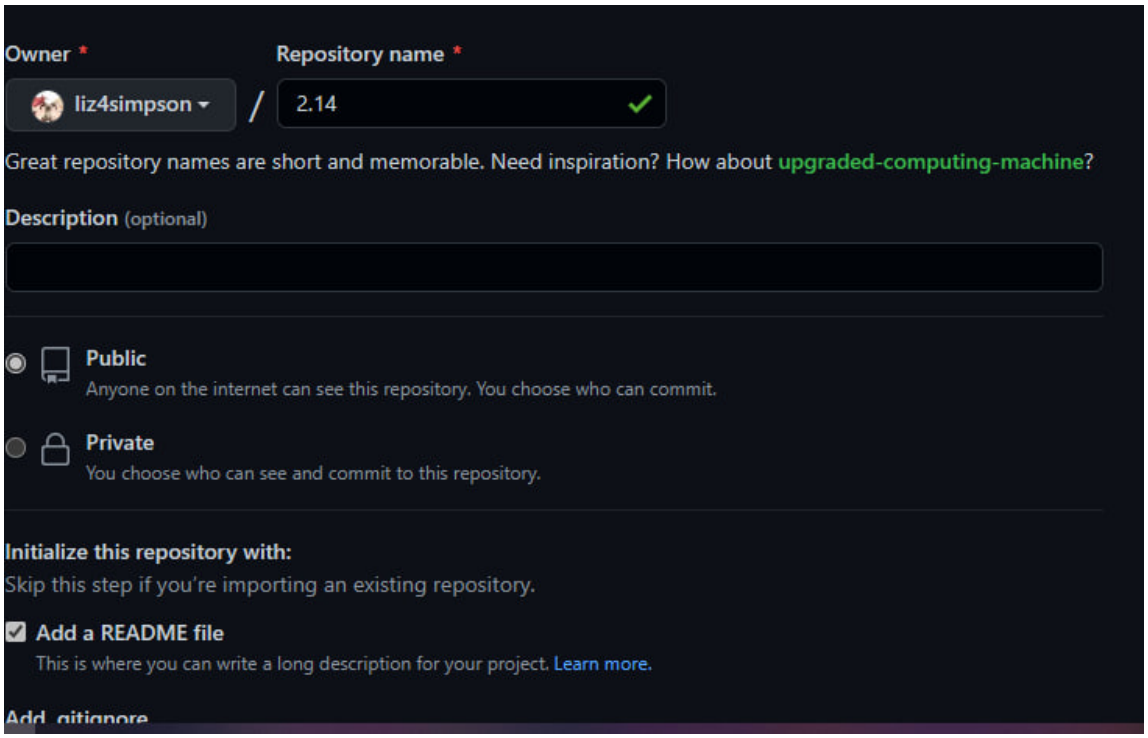
1. Создала репозиторий.
2. 

Рисунок 1. Создание репозитория

3. Клонировала репозиторий.

```
C:\Users\Elizaveta\Desktop\git>git clone https://github.com/liz4simpson/2.14.git
Cloning into '2.14'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), 4.52 KiB | 578.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
```

Рисунок 2. Клонирование репозитория

4. Организовала свой репозиторий в соответствие с моделью ветвления git-flow.

5. Начало работы с виртуальным окружением.

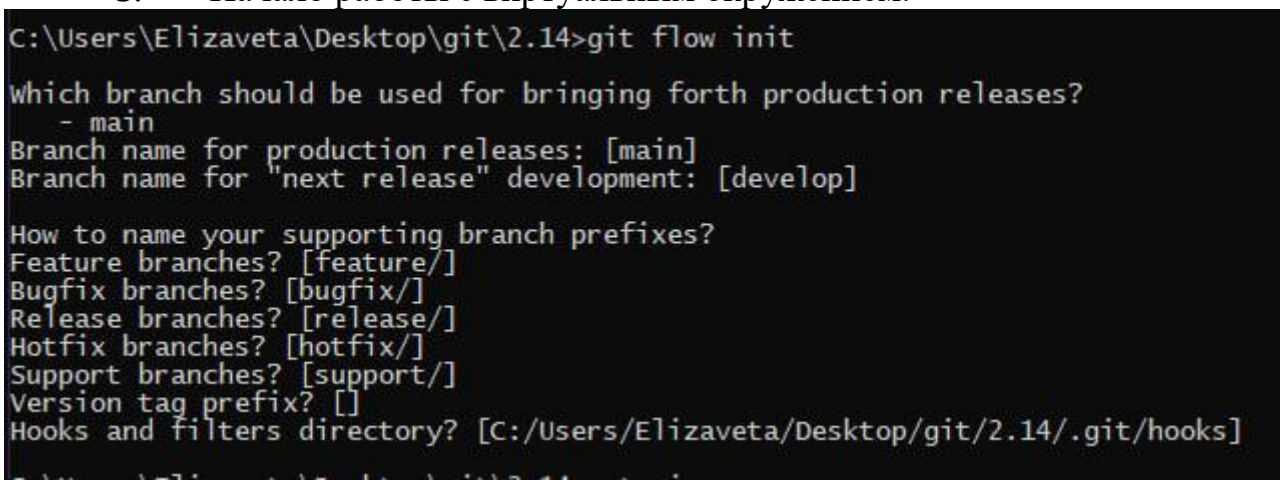


Рисунок 3. Организация репозитория в соответствии с git-flow.

Рисунок 4. Создание и активация виртуального окружения

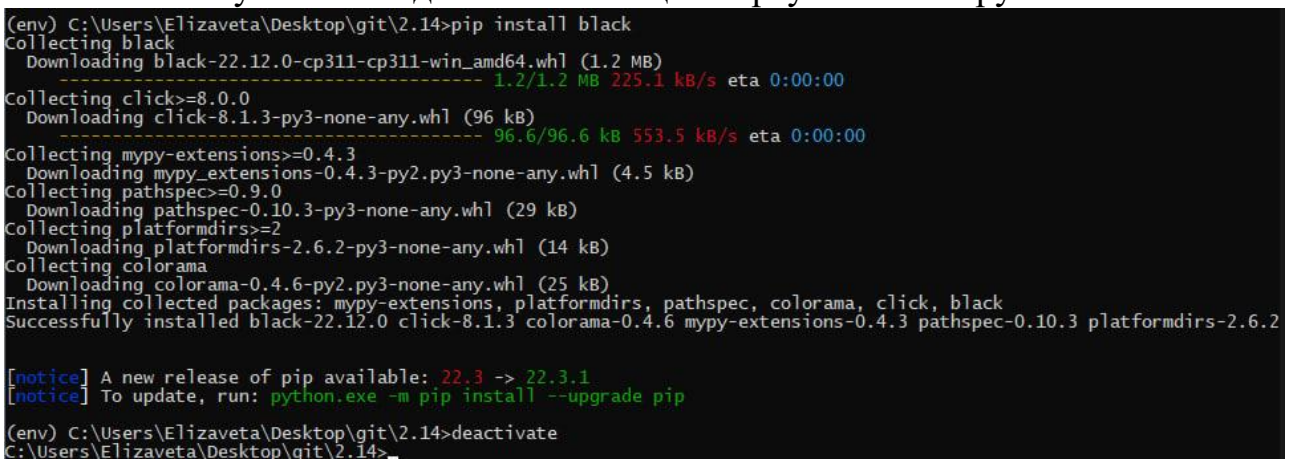


Рисунок 5. Установка пакета

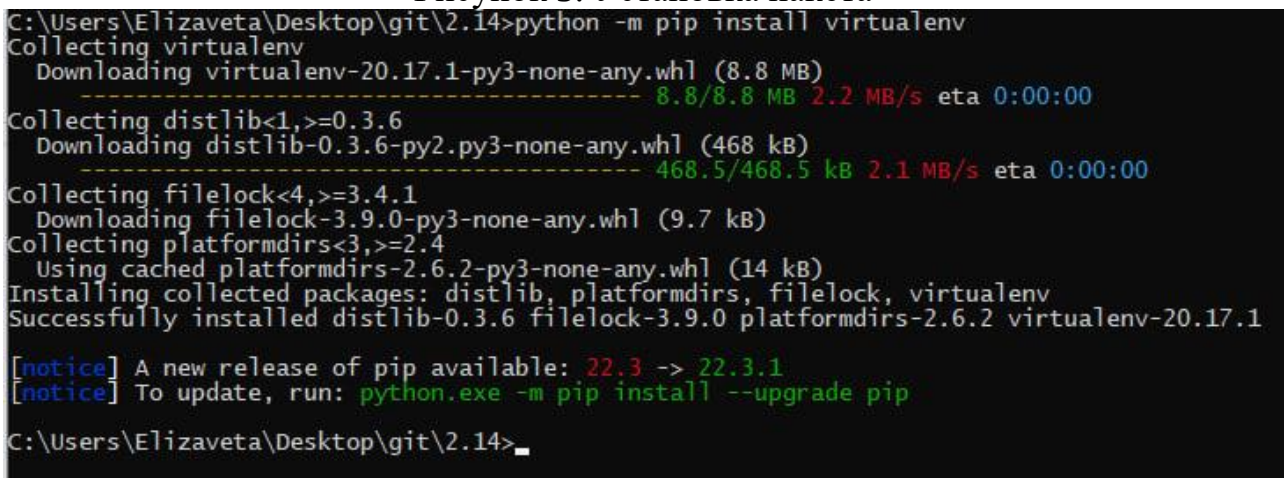


Рисунок 6. Установка virtualenv

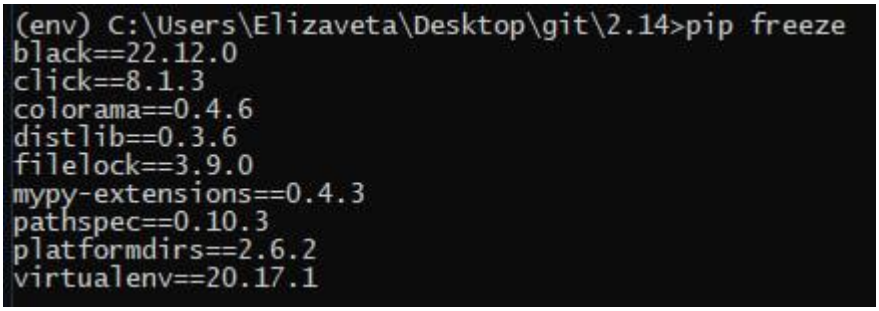


Рисунок 7. Список пакетных зависимостей

Рисунок 8. Сохранение списка в файл

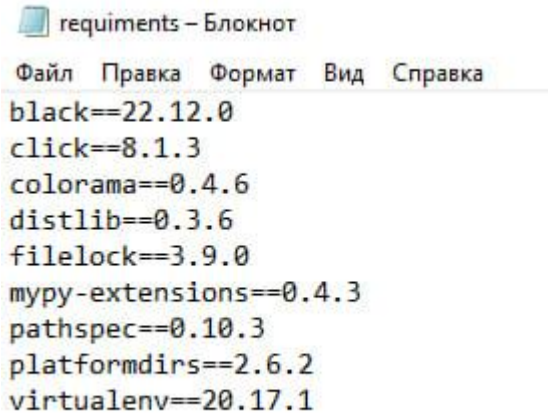


Рисунок 9. Файл requiments.txt

Рисунок 10. Деактивация виртуального окружения

6. Управление пакетами с помощью Conda.

```
(base) PS C:\Users\Elizaveta\Desktop\git\2.14\%python2.14%> conda create -n %python2.14% python=3.9
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 22.11.1

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: C:\Users\Elizaveta\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-65.6.3          |   py39haa95532_0         1.1 MB
    sqlite-3.40.1              |       h2bbff1b_0         889 KB
    tzdata-2022g               |       h04d1e81_0         114 KB
    ------------------------------------------------------------
                                           Total:        30.1 MB

The following NEW packages will be INSTALLED:
```

Рисунок 11. Создание чистой директории и виртуального окружения

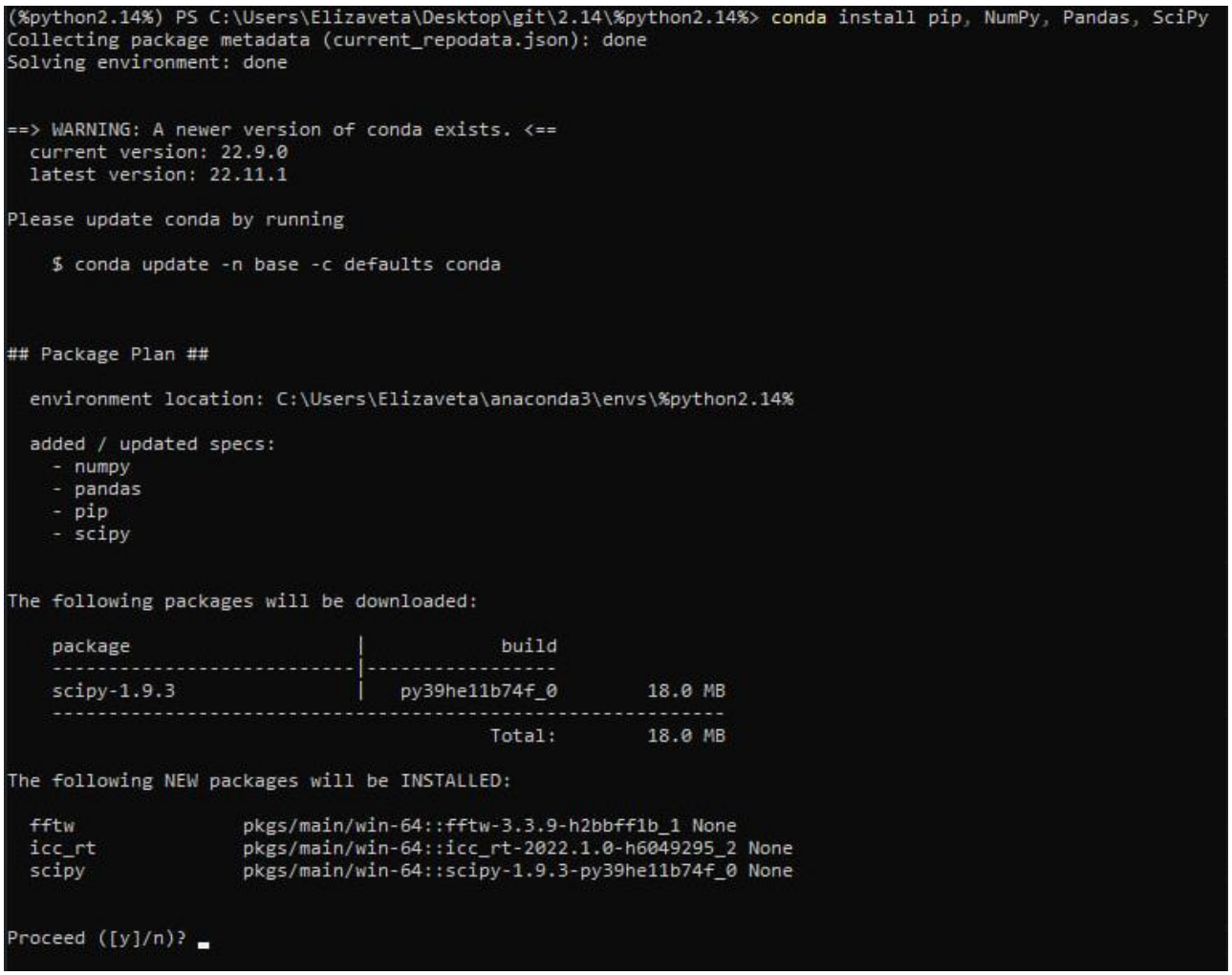


Рисунок 12. Установка пакетов

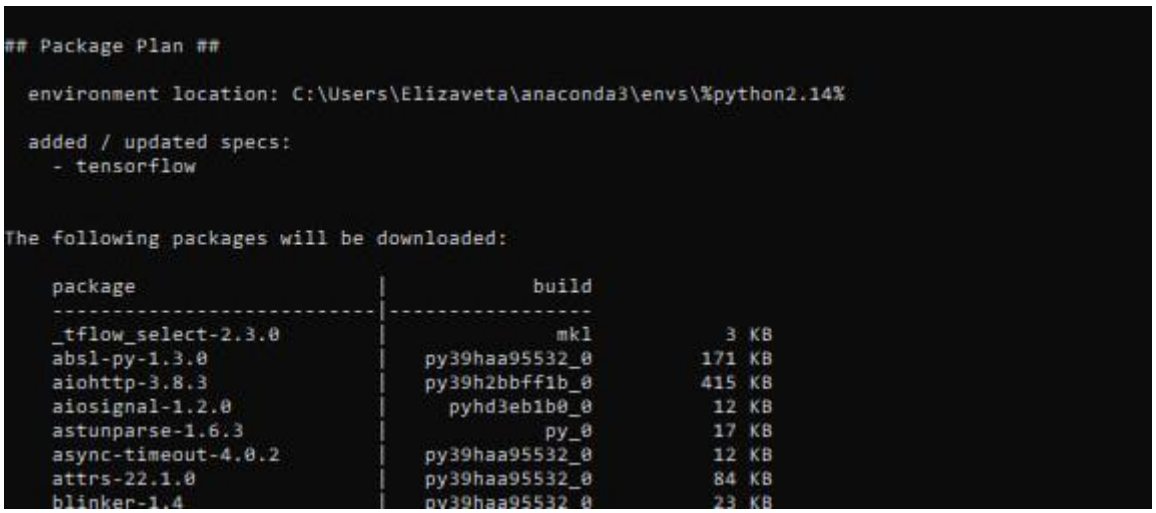


Рисунок 13. Установка пакета TensorFlow

Ошибки при установке данного пакета не возникает.

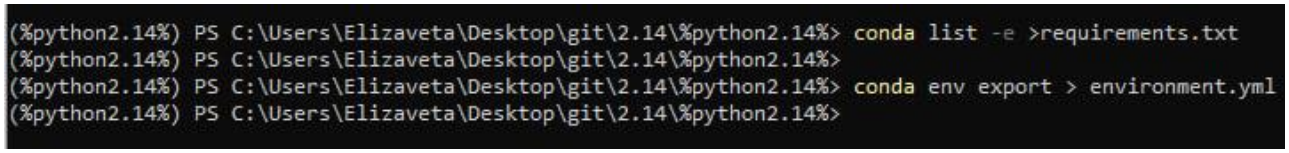


Рисунок 14. Формирование файлов requirements.txt и eniveronment.ylm

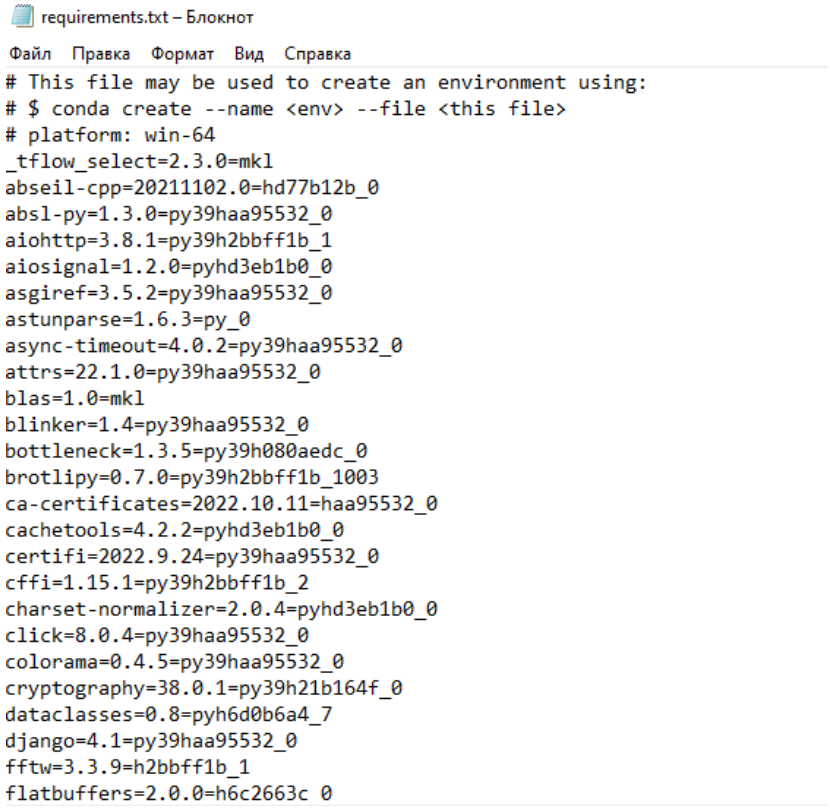


Рисунок 15. Файл requirements.txt

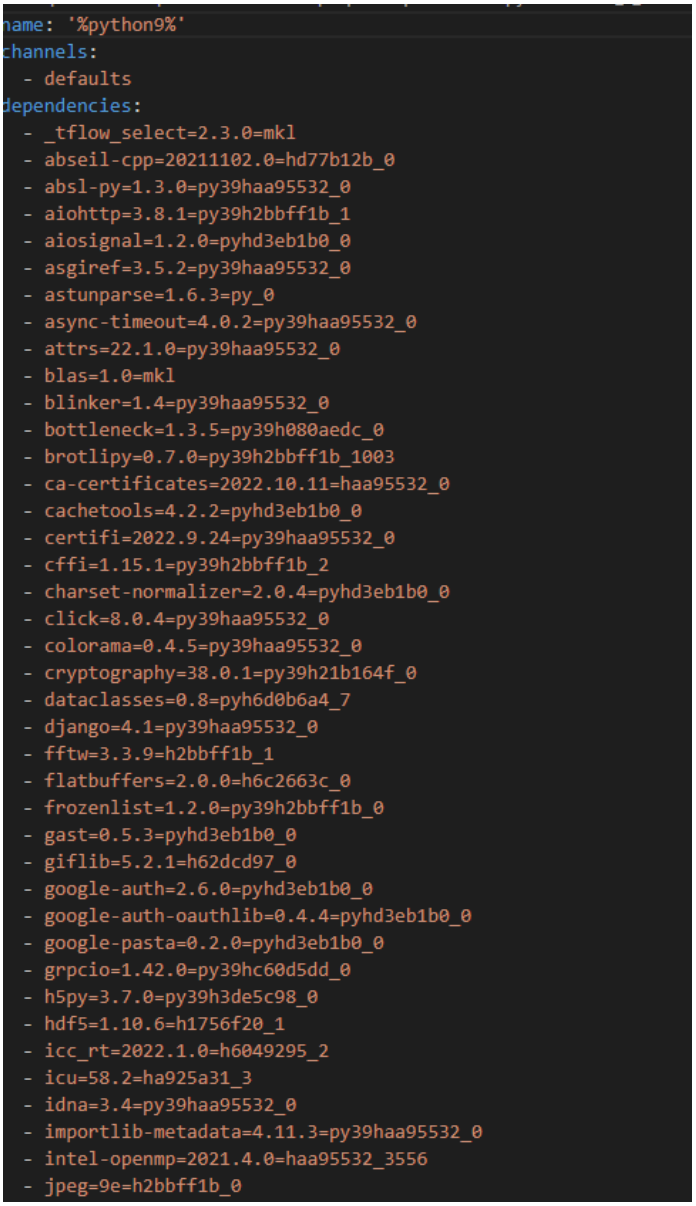


Рисунок 16. Файл eniveronment.ylm

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. Обсуждение этих составляющих выходит за рамки урока, поэтому мы не будем на них

останавливаться.

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==3.2

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 т 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".олько одна версия пакета. Это порождает ряд проблем.

12. Каковы основные этапы работы с виртуальными окружениями?

1. Создаём через утилиту новое виртуальное окружение в отдельной папке для выбранной версии интерпретатора Python.

2. Активируем ранее созданное виртуального окружения для работы.

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

4. Деактивируем после окончания работы виртуальное окружение.

5. 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 под нужно:

> env\\Scripts\\activate

Просто под Windows мы вызываем скрипт активации напрямую.

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

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

Зачем нам нужно уметь работать с утилитой virtualenv? Ведь мы уже научились работать со стандартным модулем Python venv. Просто он очень распространён и поддерживает большее число вариантов и версий интерпретатора Python, например, PyPy и CPython.

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?

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 погрузка переменных окружения из .env файла

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

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 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

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?

1. 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

Для Windows, если испольъзуется 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).

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?

source activate $PROJ_NAME

21. Как деактивировать и удалить виртуальное окружение conda?

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.

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.

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 какихлибо 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.