1. списки, в отличие от строк, являются изменяемыми. <https://docs.python.org/3/library/functions.html#enumerate>

<https://pythonist.ru/funkczii-i-ih-argumenty-v-python-3/> - книги тут Думаю эта –

python-forum.io

https://www.dreamincode.net/forums/forum/29-python/

StackOverflow.com

<https://docs-python.ru/>

<https://python-scripts.com/json>

Курс от Диджитализируй: <https://course01.to.digital/> - 30 000 и так себе

Объяснение @classmethod и @staticmethod в Python

https://webdevblog.ru/obyasnenie-classmethod-i-staticmethod-v-python/

1. Словарь:

dict\_1 = {(9, 10): 'Joe', 'Fish': 78, 12.4: ['k', 'l']}

print(dict\_1[(9, 10)])

print(dict\_1['Fish'])

print(dict\_1[12.4])

В первом случае ключ - кортеж, а значение - строка; во втором ключ - строка, значение - целое число; в третьем ключ - число с плавающей точкой, значение - список.

dict\_1 = {(9, 10): 'Joe', 'Fish': 78, 12.4: ['k', 'l']}

dict\_2 = {'Kcal': 560, 'Fish': 166}

dict\_1['Color'] = 'Red' # Присвоить ключу значение. Если такого ключа нет, он создается

print(dict\_1.pop((9, 10))) # Удалить указанный ключ из словаря и вернуть соответствующее значение

print(dict\_1)

print(dict\_1.popitem()) # Удалить (случайный) элемент словаря и вернуть удаленный ключ и значение

print(dict\_1)

dict\_1.update(dict\_2) # Включить dict\_2 в dict\_1. Значение совпадающих ключей перезаписывается

print(dict\_1)

1. Кортеж:

tuple\_1 = (4, 10, 'str1', [12, 190])

tuple\_1[3][0] = 1

print(tuple\_1)

1. Список:

list\_1 = [11, 10, 92, 4, 10, 'str1', [76, 'Bobby']]

1. Множества:

set\_1 = {10, 2, 62, 101, 56}

set\_1.add(90) # Добавляет во множество новый элемент

print(set\_1)

set\_1.discard(62) # Удаляет из множества элемент

print(set\_1)

set\_1.pop() # Удаляет из множества первый элемент

print(set\_1)

1. Просмотр кодировки:

import locale

print(locale.getpreferredencoding(False)

1. Django:

Работает так: python manage.py startapp rbr\_srv\_side

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'rest\_framework',

'rbr\_srv\_side',

]

{

"id": ...,

"ip\_address": "...",

"description": "...",

"name": "..."

}

python manage.py runserver

**Миграция**:

python3 -m venv .venv

.venv/Scripts/activate

**Из config (E:\Куликов\Juniper и GNS3\Python Rebrainme\projects\config>):**

python manage.py makemigrations

python manage.py migrate

python manage.py runserver

python manage.py startapp rbr\_srv\_side

Добавление сервера:

<http://127.0.0.1:8000/api/servers/add> - добавляем, а потом смотрим - http://127.0.0.1:8000/api/servers

User:

python manage.py createsuperuser --username=andrey --email=kulikofff@outlook.com

http://127.0.0.1:8000/admin/

Сделайте еще одну страницу, которая будет выводить вам список серверов в сокращенном виде: только ip сервера и его статус (поле 'server\_is\_active'). Для этого вам понадобится создать еще один сериалайзер, но на этот раз всего с двумя полями. Затем вам нужно будет импортировать этот сериалайзер во views.py и описать там представление, подобное представлению ServerViewSet, но с другим сериалайзером.