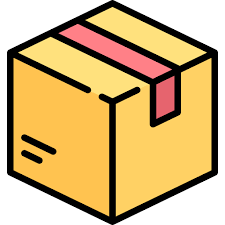
****

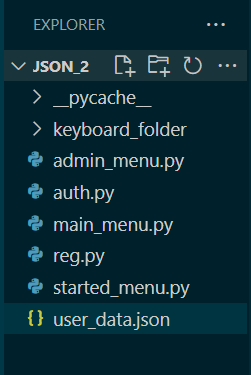
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Работа с json**

**Лабораторная работа №1**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Реализуйте работу с json данными посредством написания консольного приложения, которое должно включать в себя:**

****

**\_\_pycache\_\_ - просто кэш, который Питон создает автоматически когда пользователь проводит первую отладку файла, чтобы следующие отладки проходили быстрее.**

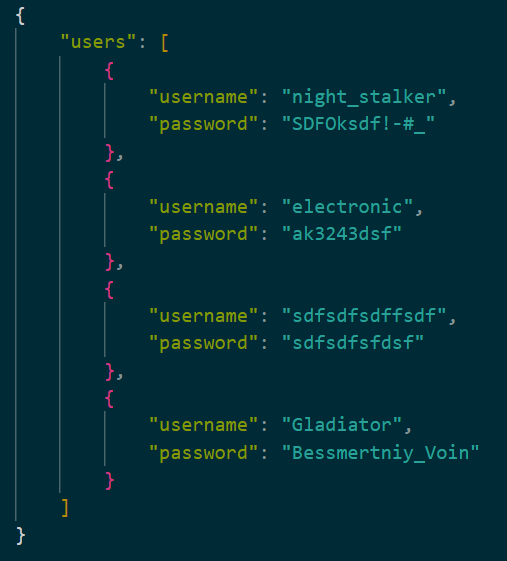
**Keyboard\_folder – папочка с библиотекой keyboard которая нам нужна для того чтобы подключить данную библиотеку, в связи с проблемами доступа приходится обращаться к ним напрямую.**

**Каждый файл должен быть подключен друг к другу по его нуждам**

****

**Т.е если нам нужна библиотека keyboard мы пишем что из папки keyboard мы забираем нашу библиотеку. Если же мы вызываем функцию которая нам необходима из другого файла (далее модуль), мы используем оператор import.**

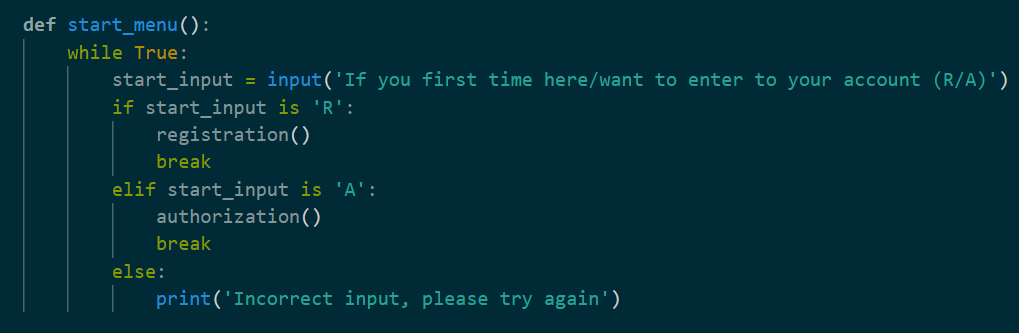


**User\_data.json (файл json) имеющий следующий вид:** ****

**В основе всего мы будем оперировать с этими конкретными данными, а именно: добавлять и удалять пользователей, регистрировать новых и заходить под аккаунтами уже существующих.**

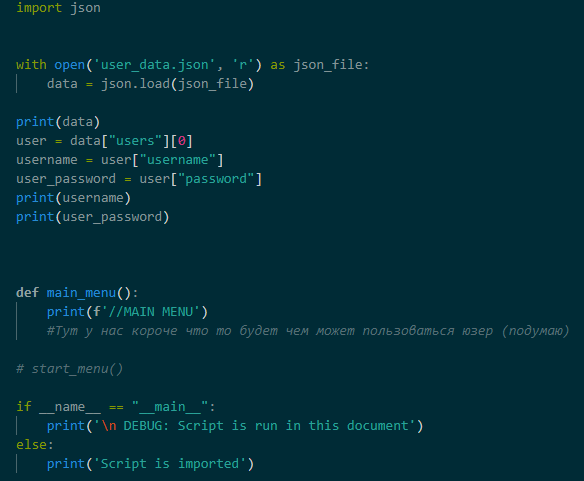
**Основным файлом для запуска будет started\_menu.py в котором должна быть реализовано меню с созданием аккаунта, в котором вызываются функции registration() и authorization() из двух разных файлов: auth.py и reg.py**

**Started.menu.py имеет следующий вид:**

****

**Создаем бесконечный цикл While в котором пишем один input (ввод данных) в котором спрашиваем пользователя хочет он зарегистрироваться или войти в аккаунт. В зависимости от условий нас выбрасывает в разные функции, либо в registration() либо в authorization().**

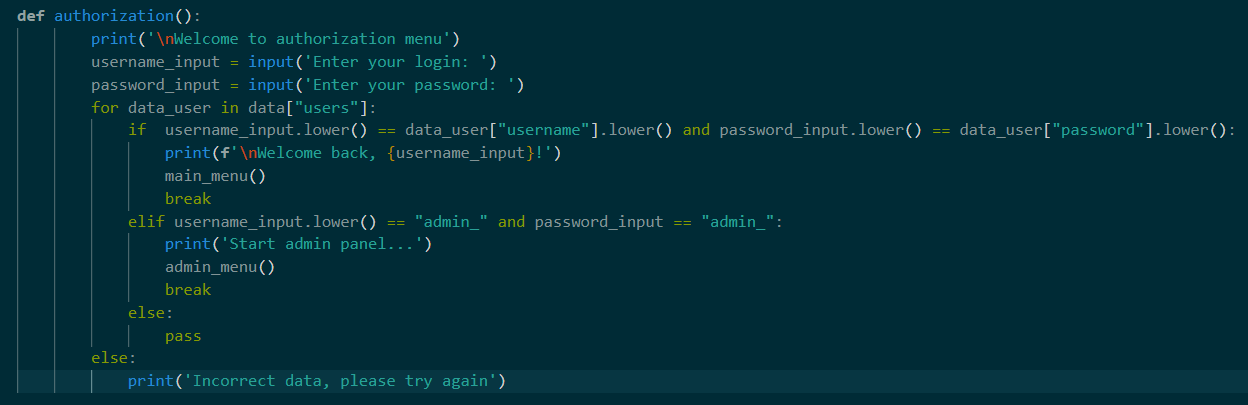
**Вторичным но не менее важным для нас будет файл main\_menu.py, именно с него мы забираем основные данные файла (data), и переопределяем их.**

****

**Определяем user как data забирающая ключ «users », это нам понадобится в будущем для наших других модулей, и так же даем определение «username» и «user\_password», что тоже нам понадобится в будущем. В функции main\_menu() будет реализован функционал для авторизованных пользователей.**

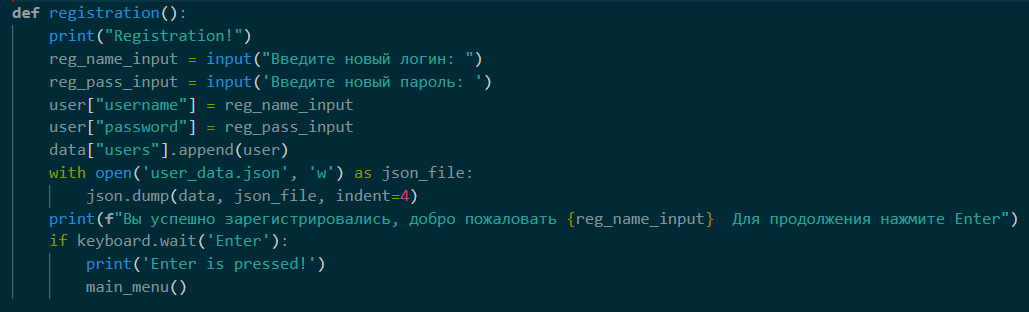
**If \_\_name\_\_ же позволяет нам понимать, запускается ли этот скрипт (внутри \_\_name\_\_ который мы введем) из этого документа, либо скрипт импортирован**

**В файле auth.py должно быть реализована авторизация пользователя:**

****

**Создаем два input’a для ввода логина и пароля, потом перебираем нашу базу данных обращаясь к ключу из json файла «users», после выполняем проверку наших введенных данных на совпадения с нашей псевдо базой данных, в случае если совпадение есть вызывается функция main\_menu() из другого файла. Проверяем условие если вошел администратор под кодовым словом, если проверка прошла вызывается функция admin\_menu() из другого файла. Выполняем последнюю проверку на правильность введенных данных.**

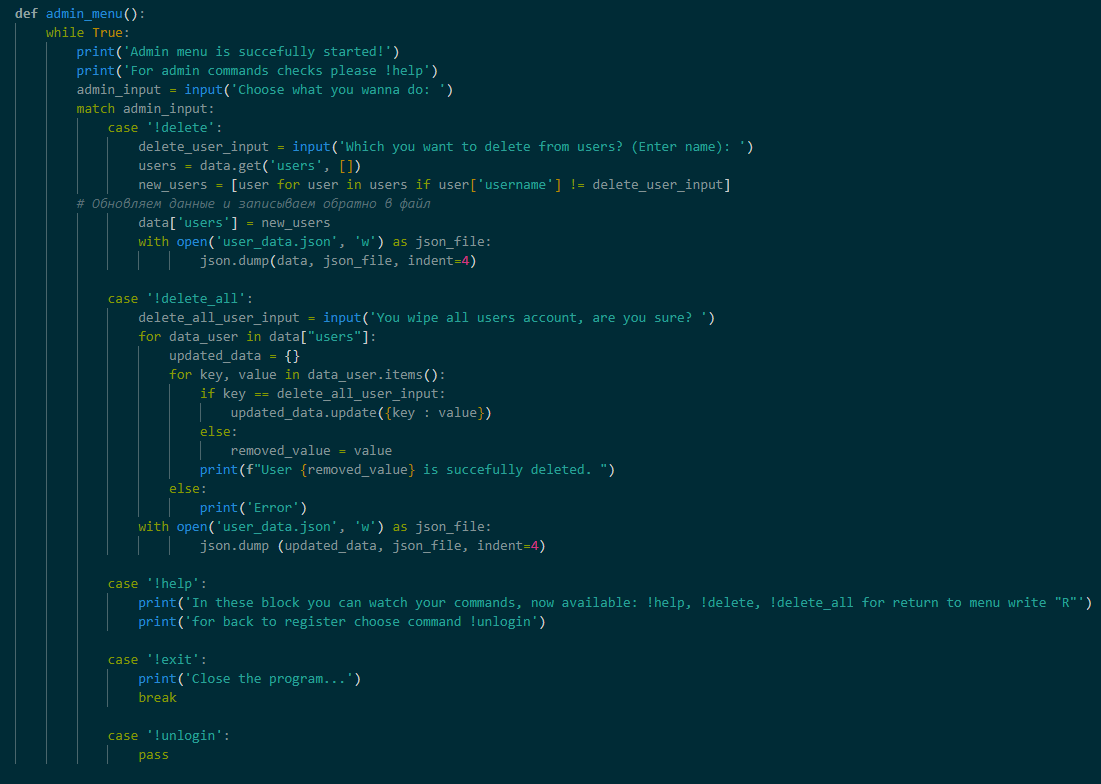
**В файле reg.py должна быть реализована регистрация пользователя, все данные должны заноситься в json файл, файл должен иметь следующий вид:**

****

**Так же реализуем два ввода данных, после переопределяем наши данные для двух ключей «username» и «password», обращаемся к data (название нашего json файла откуда берутся данные) и через функцию (метод) .append() вставляем нашего нового user’a, после обращаемся к оператору with open и перезаписываем нашу псевдо базу данных. Чтобы войти в главное меню должно быть реализовано нажатие клавиши Enter через метод wait (программа ожидает нашего нажатия именно на Enter), после вызывается функция main\_menu()**

**Admin\_menu.py 🡪**

**В файле admin\_menu.py должно быть реализована возможность управлением нашей псевдо базы данных путем разных команд (!delete, !help) и т.д.**

****

**Для начала мы должны реализовать ввод данных спрашивая администратора, что он хочет сделать. После этого должен выдаваться набор из пар match/case при вводе данных, в котором выполняются определенные команды написанные администратором. В частности нам необходимы:**

1. **команда !delete чтобы удалить конкретного пользователя из нашей псевдо базы данных по его логину**
2. **команда !delete\_all позволяющая уничтожить все данные пользователей (очистить)**
3. **команда !help позволяющая администратору посмотреть на список возможных команд**
4. **команда !exit завершающая процесс выполнения программы**

**После того как мы все с вами напишем основной костяк, нам нужно добавить следующие вещи:**

1. **Самостоятельно реализуйте новую команду в match admin\_input (!help, !delete и т.д), которая должна как-либо помогать администратору работать с данными**
2. **Самостоятельно допишите функцию main\_menu, где для авторизованных пользователей должна начинаться игра в бросок кубика (случайная сторона кубика 1d6)**

**(Дополнительно)**

**Реализуйте профиль авторизованного пользователя где он может указать о себе больше информации, например его почту, дату рождения, номер телефона и т.д.**