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 №2.16
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Евдаков Евгений Владимирович
2 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные технологии и системы связи»,
направленность (профиль) «Инфокоммуникационные системы и сети», очная форма обучения

_____
(подпись)

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

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: работа с данными формата JSON в языке Python

Цель: приобретение навыков по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

Ход работы:

Задание 1. Создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер.

```
C:\Users\student-09-510>git clone https://github.com/EvgenyEvdakov/Laba_1.16.git
Cloning into 'Laba_1.16'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 1. Клонирование репозитория

Задание 2. Организовал свой репозиторий в соответствие с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\Users\student-09-510\Laba_1.16>git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/student-09-510/Laba_1.16/.git/hooks]
```

Рисунок 2. Модель ветвления git-flow

Задание 3. Создал виртуальное окружение conda и активировал его, также установил необходимые пакеты isort, black, flake8.

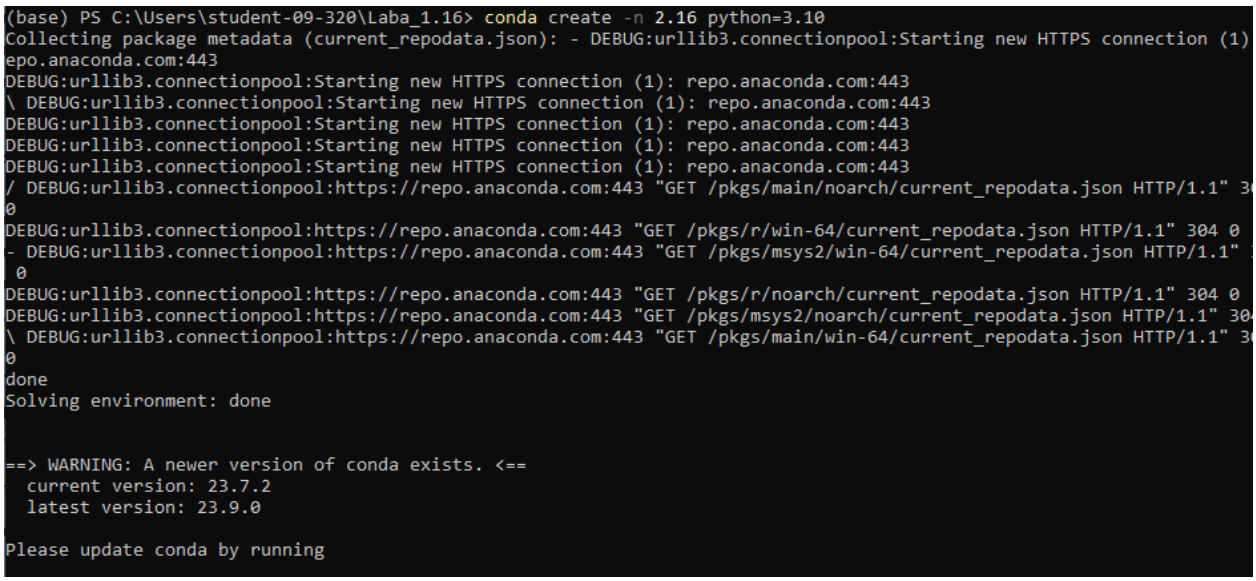

Рисунок 3. Создание виртуального окружения

Задание 4. Создал проект PyCharm в папке репозитория. Приступил к работе с примером. Добавил новый файл primer1.py.

Условие примера: Для примера 1 лабораторной работы 2.8 добавьте возможность сохранения списка в файл формата JSON и чтения данных из файла JSON.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import json
import sys
from datetime import date


def get_worker():
    """
   Запросить данные о работнике.
    """
    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))
    # Создать словарь.
    return {
        "name": name,
        "post": post,
        "year": year,
    }


def display_workers(staff):
    """"
    Отобразить список работников.
    """""
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-" * 8)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^8} |".format(
                "No", "Ф.И.О.", "Должность", "Год"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:>8} |".format(
                    idx,
                    worker.get("name", ""),
                    worker.get("post", ""),
                    worker.get("year", 0),
                )
            )
```

```
            )
        print(line)
    else:
        print("Список работников пуст.")


def select_workers(staff, period):
    """
    Выбрать работников с заданным стажем.
    """
    # Получить текущую дату.
    today = date.today()
    # Сформировать список работников.
    result = []
    for employee in staff:
        if today.year - employee.get("year", today.year) >= period:
            result.append(employee)

    # Возвратить список выбранных работников.
    return result


def save_workers(file_name, staff):
    """
    Сохранить всех работников в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кирилицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)


def load_workers(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


def main():
    """
    Главная функция программы.
    """
    # Список работников.
    workers = []
```

```
    workers = []
    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()
        # Выполнить действие в соответствие с командой.
        if command == "exit":
            break

        elif command == "add":
            # Запросить данные о работнике.
            worker = get_worker()
            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get("name", ""))
        elif command == "list":
            # Отобразить всех работников.
            display_workers(workers)
        elif command.startswith("select "):
            # Разбить команду на части для выделения стажа.
            parts = command.split(maxsplit=1)
            # Получить требуемый стаж.
            period = int(parts[1])
            # Выбрать работников с заданным стажем.
            selected = select_workers(workers, period)
            # Отобразить выбранных работников.
            display_workers(selected)
        elif command.startswith("save "):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]

            # Сохранить данные в файл с заданным именем.
            save_workers(file_name, workers)

        elif command.startswith("load "):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]

            # Сохранить данные в файл с заданным именем.
            workers = load_workers(file_name)

        elif command == "help":
```

```
        elif command == "help":
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить работника;")
            print("list - вывести список работников;")
            print("select <стаж> - запросить работников со стажем;")
            print("help - отобразить справку;")
            print("load - загрузить данные из файла;")
            print("save - сохранить данные в файл;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)


if __name__ == "__main__":
    main()
```

Рисунок 4-7. Пример 1

Задание 4.

Индивидуальное задание

Вариант 10

Создал новый файл под названием idz.py.

Условие задания: Для своего варианта лабораторной работы 2.8 необходимо дополнительно реализовать сохранение и чтение данных из файла формата JSON. Необходимо также проследить за тем, чтобы файлы генерируемый этой программой не попадали в репозиторий лабораторной работы.

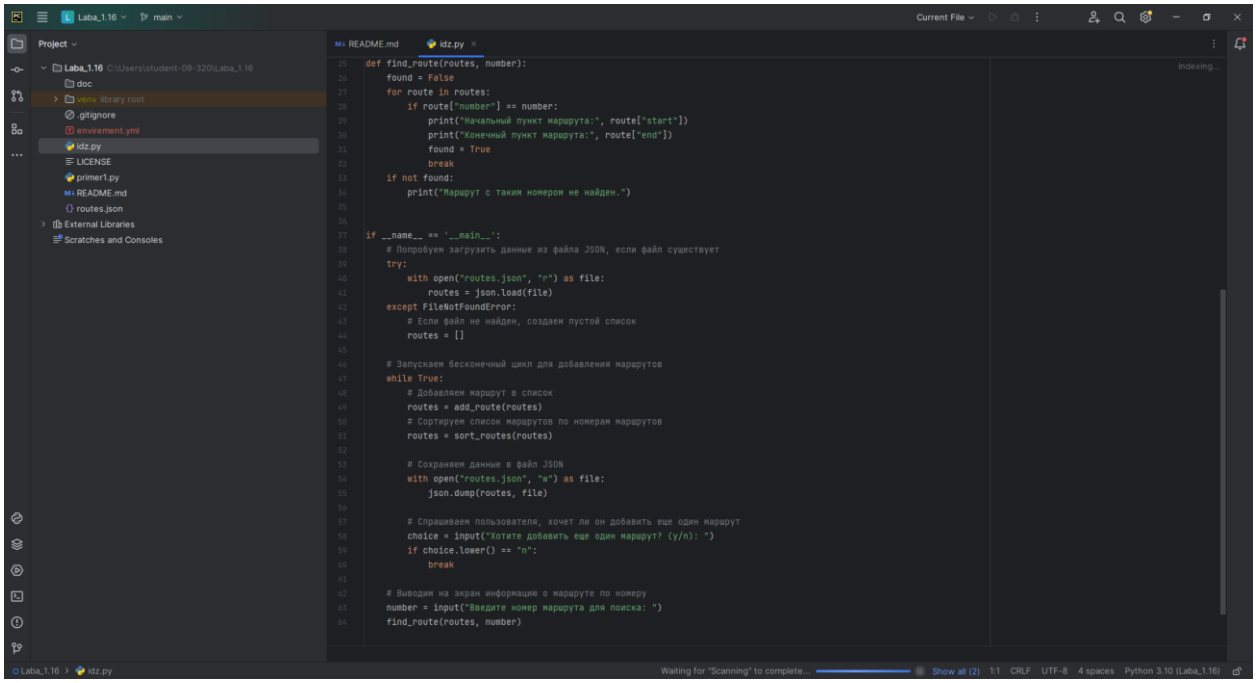

Рисунок 8. Программа индивидуального задания

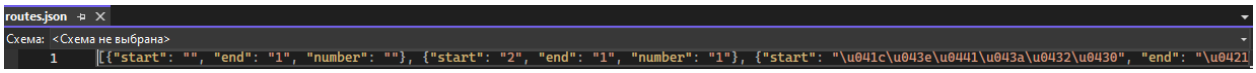

Рисунок 9. Файл json

Задание 7.

После выполнения работы на ветке develop, слил ее с веткой main и отправил изменения на удаленный сервер. Создал файл envirement.yml и деактивировал виртуальное окружение.

```
(2.16) PS C:\Users\student-09-320\Laba_1.16> conda env export > envirement.yml
(2.16) PS C:\Users\student-09-320\Laba_1.16> conda deactivate
```

Ссылка: https://github.com/EvgenyEvdakov/Laba_1.16

Ответы на контрольные вопросы:

1. **Для чего используется JSON?**

JSON — это стандарт обмена данными. Он позволяет легко сериализовать и десериализовать объекты. Стандарт часто применяют, когда разрабатывают API и веб-приложения.

2. **Какие типы значений используются в JSON?**

В качестве значений в JSON могут быть использованы:

- запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.
- массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, то есть не содержать ни одного значения. Значения в пределах одного массива могут иметь разный тип.
- число (целое или вещественное).
- литералы true (логическое значение «истина»), false (логическое значение «ложь») и null.
- строка — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки.

3. **Как организована работа со сложными данными в JSON?**

- JSON позволяет организовать сложные структуры данных, такие как списки и вложенные словари (объекты).
- В JSON можно хранить разные типы данных, включая числа, строки, логические значения, массивы и объекты.
- Для организации сложных данных в JSON используются вложенные объекты и списки, позволяя создавать структуры данных любой сложности.

4. **Самостоятельно ознакомьтесь с форматом данных JSON5? В чем отличие этого формата от формата данных JSON?**

- JSON5 - это расширение формата данных JSON, разработанное для улучшения читаемости и удобства записи JSON-данных.

- Отличие JSON5 от обычного JSON включает в себя дополнительные возможности, такие как использование комментариев, разделителей ключей и значений, а также возможность использования одиночных кавычек вместо двойных.
- JSON5 является более гибким и читаемым форматом для записи данных, но не является стандартом и не поддерживается всеми JSON-парсерами.

5. **Какие средства языка программирования Python могут быть использованы для работы с данными в формате JSON5?**

Для работы с данными в формате JSON5 на Python, вы можете использовать парсеры, поддерживающие JSON5, такие как demjson. Однако, JSON5 не является стандартом, поэтому поддержка может быть ограничена.

6. **Какие средства предоставляет язык Python для сериализации данных в формате JSON?**

Для сериализации данных в формат JSON в Python можно использовать модуль json. Он предоставляет функции json.dump() и json.dumps(), а также класс json.JSONEncoder, который может быть настроен для сериализации данных в формат JSON.

7. **В чем отличие функций json.dump() и json.dumps()?**

json.dump() записывает данные в файл. Вы используете его, когда хотите сохранить данные в файле.

json.dumps() превращает данные в строку. Вы используете его, когда хотите получить данные в виде строки для дальнейшей обработки, но не сохранять их в файле.

8. **Какие средства предоставляет язык Python для десериализации данных из формата JSON?**

Для десериализации данных из формата JSON в Python используется модуль json, предоставляющий функции json.load() и json.loads().

9. **Какие средства необходимо использовать для работы с данными формата JSON, содержащими кирилицу?**

Для работы с данными JSON, содержащими кириллицу, важно убедиться, что данные правильно кодируются и декодируются. Обычно это не вызывает проблем, поскольку JSON поддерживает Unicode, включая кириллические символы. Однако, при чтении и записи JSON-файлов, убедитесь, что правильно установлена кодировка (например, utf-8) для текстовых данных.

10. **Самостоятельно ознакомьтесь со спецификацией JSON Schema? Что такое схема данных?**

JSON Schema - это спецификация, которая описывает формат данных JSON и правила их валидации. С помощью JSON Schema можно определить структуру, типы данных и ограничения для JSON-данных. JSON Schema используется для проверки соответствия данных определенным правилам. Это полезно, например, при валидации данных, получаемых из внешних источников. JSON Schema не является частью стандартной библиотеки Python, но существуют библиотеки и инструменты, поддерживающие JSON Schema, которые могут использоваться в Python.

Вывод: приобрел навыки по работе с данными формата JSON с помощью языка программирования Python версии 3.x.