

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 №3  
дисциплины «Программирование на языке Python»

Вариант\_15\_

Выполнила:  
Маньшина Дарья Алексеевна  
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., канд. тех. наук,  
доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

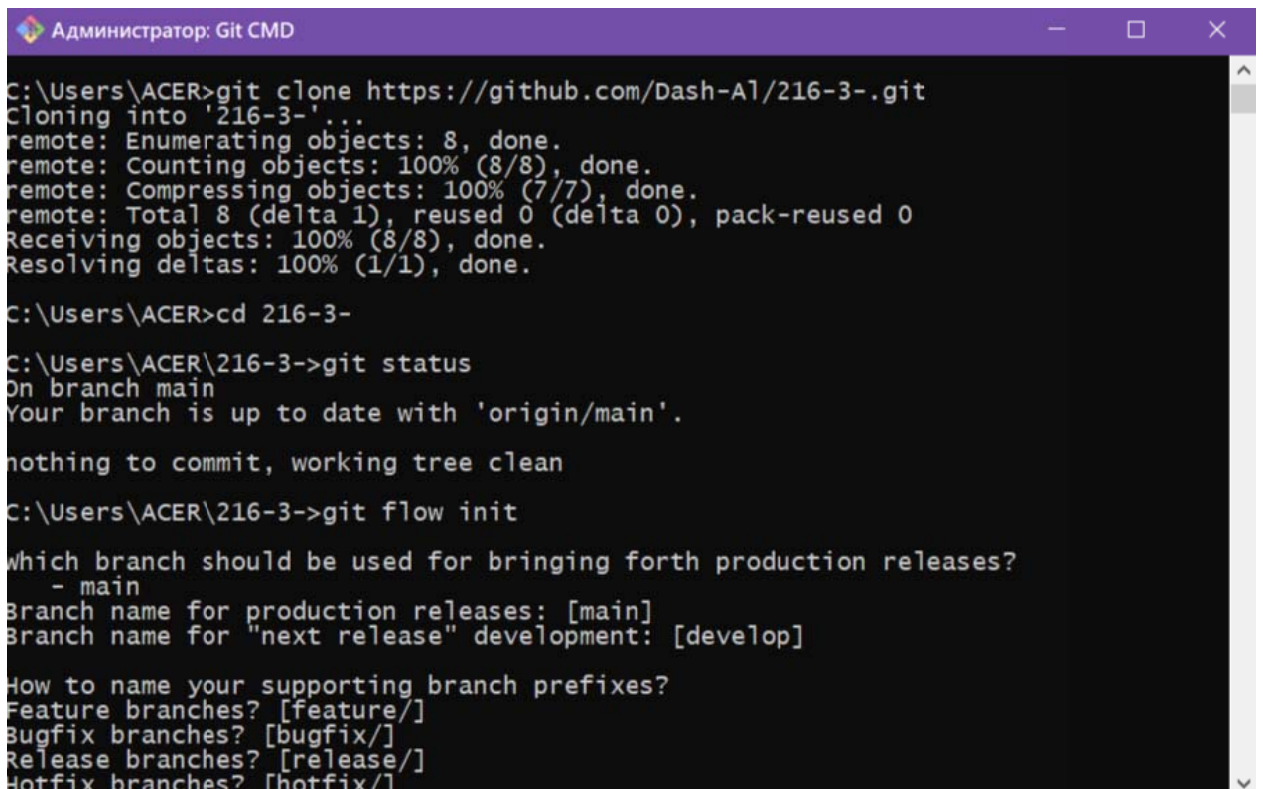
Ставрополь, 2023 г.

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:

1. Создадим новый репозиторий. Клонировать его и сделаем способ ветвления git-flow. Также установим пакеты: black, isort, flake8



```
Администратор: Git CMD
C:\Users\ACER>git clone https://github.com/Dash-A1/216-3-.git
Cloning into '216-3-'...
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), done.
Resolving deltas: 100% (1/1), done.

C:\Users\ACER>cd 216-3-

C:\Users\ACER\216-3->git status
On branch main
Your branch is up to date with 'origin/main'.

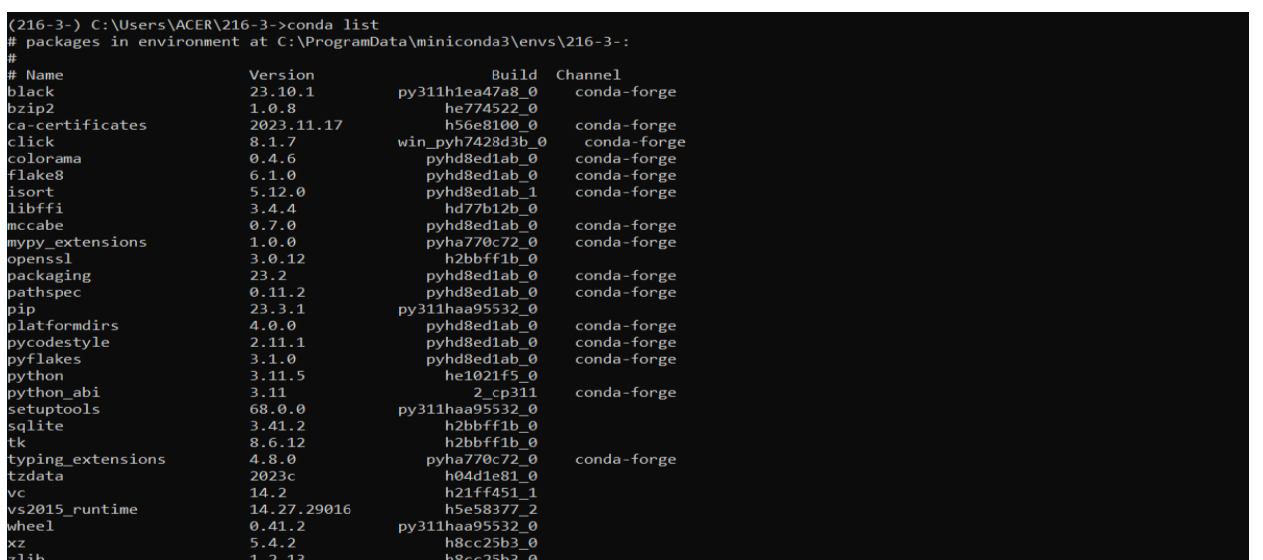
nothing to commit, working tree clean

C:\Users\ACER\216-3->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/]
```

Рисунок 1. Подготовка к лабораторной работе



```
(216-3-) C:\Users\ACER\216-3->conda list
# packages in environment at C:\ProgramData\miniconda3\envs\216-3-:
#
# Name                   Version           Build    Channel
black                    23.10.1          py311h1ea47a8_0  conda-forge
bzip2                   1.0.8            he774522_0      conda-forge
ca-certificates         2023.11.17       h56e8100_0      conda-forge
click                   8.1.7            win_pyh7428d3b_0  conda-forge
colorama                0.4.6            pyhd8ed1ab_0    conda-forge
flake8                  6.1.0            pyhd8ed1ab_0    conda-forge
isort                   5.12.0           pyhd8ed1ab_1    conda-forge
libffi                  3.4.4            hd77b12b_0      conda-forge
mccabe                  0.7.0            pyhd8ed1ab_0    conda-forge
mypy_extensions        1.0.0            pyha770c72_0    conda-forge
openssl                 3.0.12           h2bbff1b_0      conda-forge
packaging               23.2             pyhd8ed1ab_0    conda-forge
pathspec                0.11.2           pyhd8ed1ab_0    conda-forge
pip                     23.3.1           py311haa95532_0  conda-forge
platformdirs            4.0.0            pyhd8ed1ab_0    conda-forge
pycodestyle             2.11.1           pyhd8ed1ab_0    conda-forge
pyflakes                3.1.0            pyhd8ed1ab_0    conda-forge
python                  3.11.5           he1021f5_0      conda-forge
python_abi              3.11             2_cp311          conda-forge
setuptools              68.0.0           py311haa95532_0  conda-forge
sqlite                  3.41.2           h2bbff1b_0      conda-forge
tk                      8.6.12           h2bbff1b_0      conda-forge
typing_extensions       4.8.0            pyha770c72_0    conda-forge
tzdata                  2023c            h04d1e81_0      conda-forge
vc                      14.2             h21ff451_1      conda-forge
vs2015_runtime          14.27.29016      h5e58377_2      conda-forge
wheel                   0.41.2           py311haa95532_0  conda-forge
xz                      5.4.2            h8cc25b3_0      conda-forge
zlib                    1.2.13           h8cc25b3_0      conda-forge
```

Рисунок 2. Установка пакетов

## 2. Проработка примеров лабораторной работы.

### Пример 1.

#### Программа:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import json
import sys
from datetime import date
if __name__ == "__main__":
    def get_worker():
        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):
        # Проверить, что список работников не пуст.
        if staff:
            # Заголовок таблицы.
            line = '+-{}-{}-{}-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 8
            )
            print(line)
            print(
                '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                    "№",
                    "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):
        # 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):
    # 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 кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)
def load_workers(file_name):
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)
def main():
    # Список работников.
    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':
            # 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()

```

## Решение:

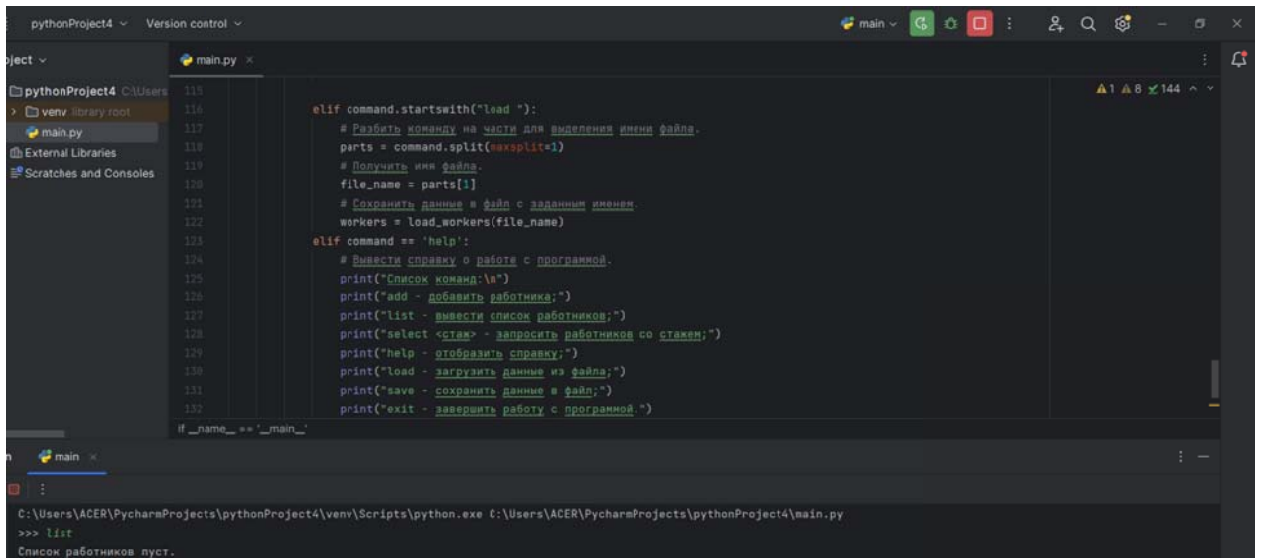


Рисунок 3. Результат работы примера №1

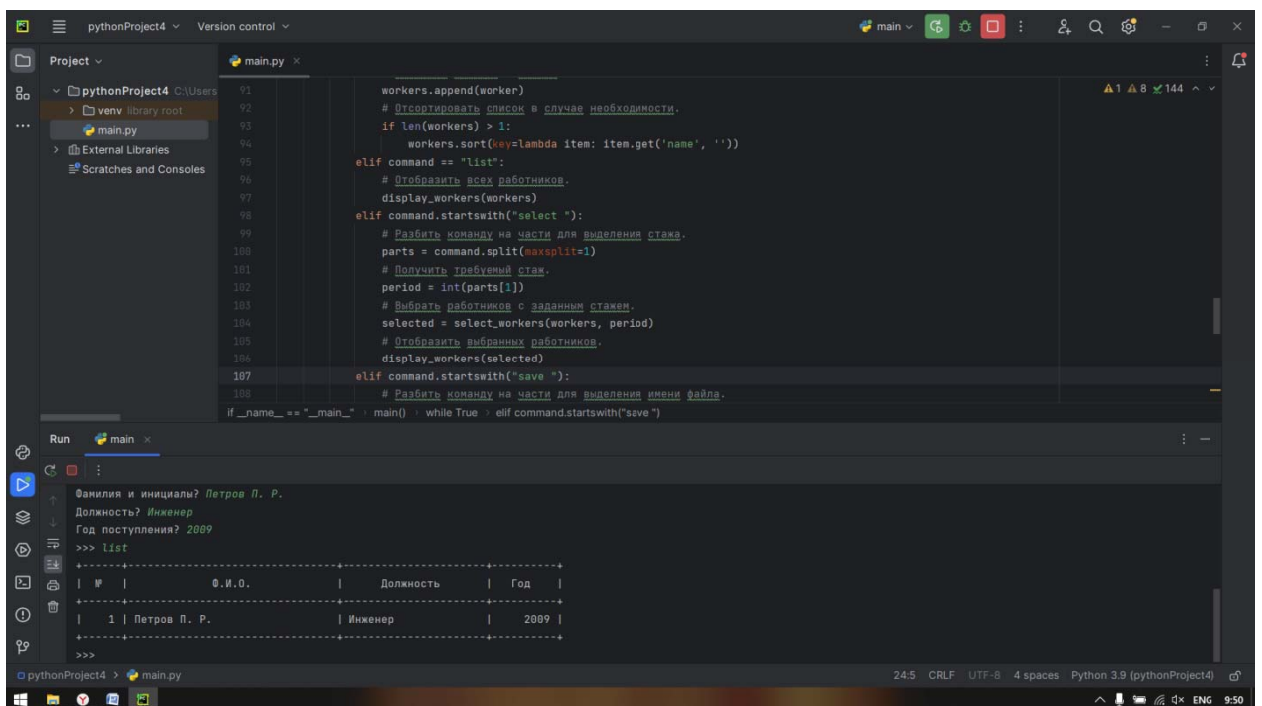


Рисунок 4. Результат работы примера №1

### 3. Решение индивидуального задания.

**Условие:** Использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата рождения (список из трёх чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по датам рождения; вывод на экран информацию о людях, родившихся под знаком, название которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение. Оформив каждую команду в виде отдельной функции. Дополнительно реализовать сохранение и чтение данных из файла формата JSON.

#### Программа:

```
import json
def add_person(people):
    last_name = input("Введите фамилию: ")
    first_name = input("Введите имя: ")
    zodiac = input("Введите знак Зодиака: ")
    birth_date = [int(x) for x in input("Введите дату рождения (через пробел): ").split()]
    person = {
        "фамилия": last_name,
        "имя": first_name,
        "знак Зодиака": zodiac,
        "дата рождения": birth_date
    }
    people.append(person)
    people.sort(key=lambda x: x["дата рождения"])
def search_by_zodiac(people):
    zodiac = input("Введите знак Зодиака для поиска: ")
    found = False
    for person in people:
        if person["знак Зодиака"] == zodiac:
            print("Фамилия:", person["фамилия"])
            print("Имя:", person["имя"])
            print("Знак Зодиака:", person["знак Зодиака"])
            print("Дата рождения:", "/".join(str(x) for x in person["дата рождения"]))
            print()
            found = True
    if not found:
        print("Люди с указанным знаком Зодиака не найдены.")
def save_to_file(filename, data):
    with open(filename, "w") as file:
        json.dump(data, file)
def load_from_file(filename):
    with open(filename, "r") as file:
        data = json.load(file)
    return data
def main():
    people = []
    filename = "data.json"
```

```

while True:
    print("1. Добавить человека")
    print("2. Поиск по знаку Зодиака")
    print("3. Сохранить данные в файл")
    print("4. Загрузить данные из файла")
    print("5. Выйти")
    choice = input("Выберите действие: ")
    if choice == "1":
        add_person(people)
    elif choice == "2":
        search_by_zodiac(people)
    elif choice == "3":
        save_to_file(filename, people)
        print("Данные сохранены в файл:", filename)
    elif choice == "4":
        people = load_from_file(filename)
        print("Данные загружены из файла:", filename)
    elif choice == "5":
        break
    else:
        print("Некорректный выбор.")
if __name__ == "__main__":
    main()

```

## Решение:

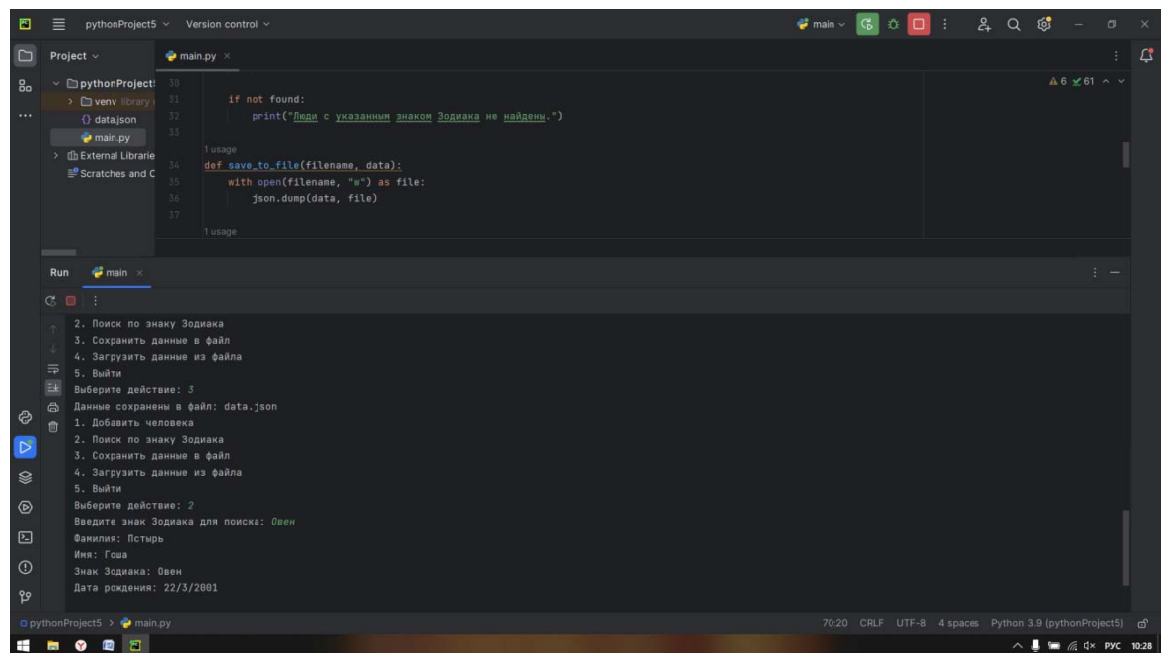


Рисунок 4. Результат работы примера №1

Вывод: в ходе лабораторной работы приобретение навыков по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

## Контрольные вопросы

### 1. Для чего используется JSON?

JSON (JavaScript Object Notation) используется для хранения и передачи данных между различными системами и языками программирования. Он представляет собой компактный формат данных, который может быть легко прочитан человеком и обработан компьютером. JSON используется для обмена данными между веб-сервисами, а также для сериализации и десериализации объектов в JavaScript.

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

В формате JSON используются следующие типы значений: числа (целочисленные и с плавающей запятой), строки, логические значения (true/false), null (пустое значение), массивы (списки значений), объекты (именованные пары “ключ”:значение).

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

Работа со сложными данными (например, массивами или объектами) в формате JSON осуществляется через итерацию по ключам и значениям. Например, для перебора всех элементов массива можно воспользоваться циклом for, а для получения значений по ключам в объекте - обращаться к ним через квадратные скобки ([]).

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

JSON5 - это расширенный формат JSON, который добавляет поддержку комментариев, форматирования строк и некоторые другие функции. В отличие от JSON, JSON5 может представлять данные более удобным для человека образом, что делает его более подходящим для разработки и отладки.



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

Для работы с данными в формате JSON в Python можно использовать встроенные функции `json.loads()` и `json.dumps()`, а также библиотеки, такие как `json5` и `jsonschema`. Эти библиотеки предоставляют дополнительные возможности для работы с форматом JSON5 и JSON Schema соответственно.

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

Сериализация данных в формат JSON в Python осуществляется с помощью функции `json.dumps()`. Она преобразует произвольные данные в строку в формате JSON.

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

Функция `json.dump()` записывает данные в указанный файл или поток, а функция `json.dumps()` возвращает отформатированную строку JSON. Обе функции принимают одинаковые аргументы, но первая также записывает данные на диск, а вторая возвращает результат форматирования.

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

Для десериализации (преобразования из строки JSON в объект Python) данных из формата JSON можно использовать функцию `json.loads()`. Она принимает строку JSON в качестве аргумента и возвращает объект Python.

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

Для работы с кириллицей в формате JSON необходимо использовать кодировку UTF-8. Это означает, что при сериализации и десериализации

данных необходимо указывать соответствующую кодировку. Например, функция `json.dumps()` имеет параметр `ensure_ascii`, который позволяет указать кодировку, используемую при сериализации.

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema?

Что такое схема данных?

JSON Schema - это язык описания структуры данных в формате JSON. Он позволяет описывать структуру данных, включая обязательные и необязательные поля, ограничения на типы данных и т.д. Схема данных - это описание структуры и ограничений на данные, которые могут быть представлены в формате JSON.