

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 №19**  
**дисциплины «Основы программной инженерии»**

Выполнил:  
Матвеев Александр Иванович  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка и  
сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Проверил Воронкин Роман Александрович

---

(подпись)

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

Ставрополь, 2024 г.

**Тема:** Работа с данными формата JSON в языке Python.

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

Ход работы.

### 1. Создание нового репозитория с лицензией MIT.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*

**Owner \*** SashkaHacker / **Repository name \*** laba10  
✔ laba10 is available.

Great repository names are short and memorable. Need inspiration? How about [fluffy-guide](#) ?

**Description** (optional)

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**  
.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**  
License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ You are creating a public repository in your personal account.

**Create repository**

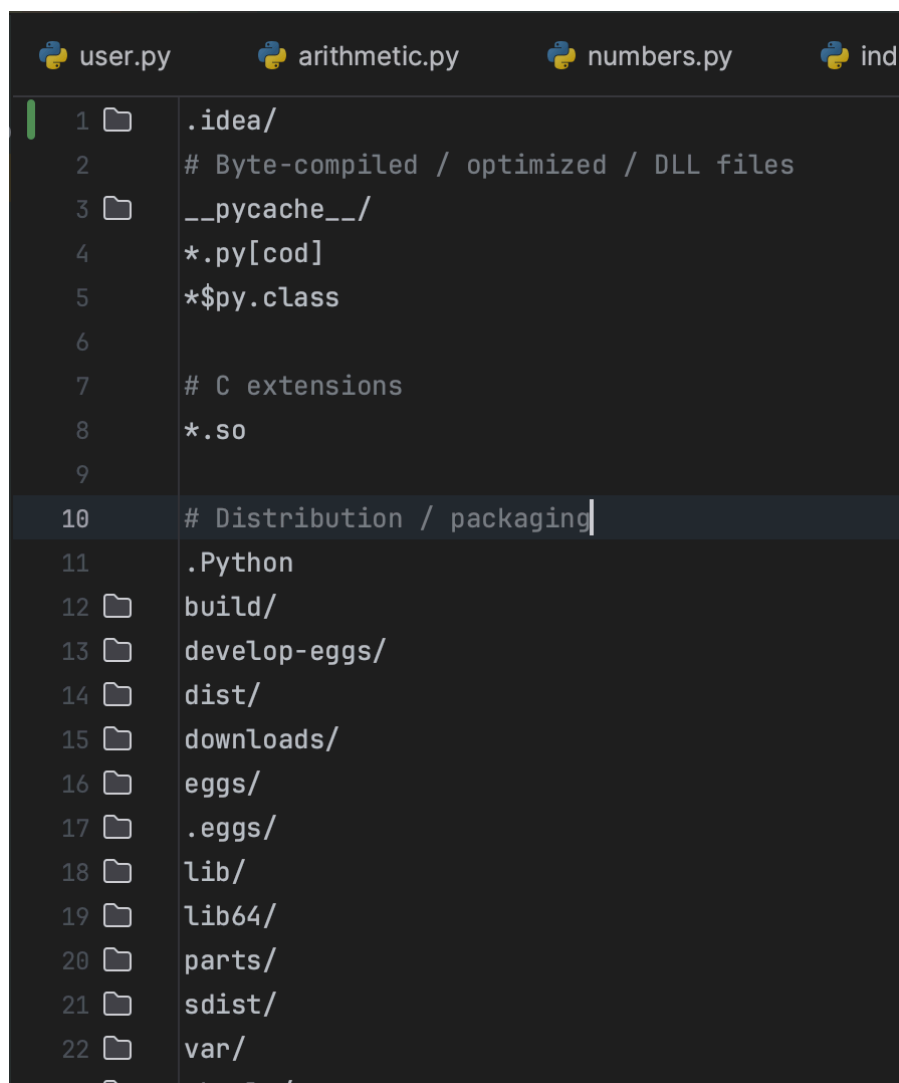
Рисунок 1 – Создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```
[→ GitHub git clone https://github.com/SashkaHacker/laba16.git
Cloning into 'laba16'...
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 2 – Клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.



The screenshot shows a code editor with a dark theme. At the top, there are four tabs: 'user.py', 'arithmetic.py', 'numbers.py', and 'ind'. The active tab is 'user.py'. The editor displays the content of the '.gitignore' file, which is being edited. The file contains the following text:

```
1  .idea/
2  # Byte-compiled / optimized / DLL files
3  __pycache__/
4  *.py[cod]
5  *$py.class
6
7  # C extensions
8  *.so
9
10 # Distribution / packaging
11 .Python
12 build/
13 develop-eggs/
14 dist/
15 downloads/
16 eggs/
17 .eggs/
18 lib/
19 lib64/
20 parts/
21 sdist/
22 var/
```

Рисунок 3 – Файл .gitignore

4. Проработка примеров из лабораторной работы.



```
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 кириллицы 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:
        data = json.load(fin)

    try:
        valid_data = ListWorkers(lst=data)
        return valid_data
    except json.JSONDecodeError:
        print("Invalid JSON")
    except ValidationError as e:
        print("Validation failed:", e.errors())
```

Рисунок 5 – Реализация чтения и записи в файл json

```
from datetime import datetime

from pydantic import BaseModel, field_validator
from typing import List

class Worker(BaseModel):
    surname: str
    name: str
    phone: int
    date: List[str]

    @field_validator("date")
    def validate_date_parts(cls, v):
        if len(v) != 3:
            raise ValueError("Expected a list of 3 elements representing a date.")

        try:
            day, month, year = map(int, v)
            date = datetime(day=day, month=month, year=year)
        except ValueError as e:
            raise ValueError(f"Error parsing date: {e}")

        if not (datetime.min < date < datetime.now()):
            raise ValueError("Date is out of acceptable range.")
        return v

class ListWorkers(BaseModel):
    lst: List[Worker]
```

Рисунок 6 – Реализация валидации

## Контрольные вопросы:

1. JSON (JavaScript Object Notation) используется для хранения и передачи данных. Он часто применяется в веб-разработке для обмена данными между клиентом и сервером, а также как формат для сериализации и передачи структурированных данных в различных приложениях.

2. В JSON используются следующие типы значений: строки (в двойных кавычках), числа, объекты (коллекция пар ключ: значение, содержащаяся в фигурных скобках), массивы (упорядоченный список значений в квадратных скобках), логические значения `true` и `false` и специальное значение `null`.

3. Работа со сложными данными в JSON организуется с помощью объектов и массивов. Объекты представляют коллекции пар ключ: значение, где ключи являются строками, а значениями могут быть данные любого типа, включая другие объекты и массивы. Массивы используются для хранения упорядоченных списков значений, включая вложенные массивы и объекты.

4. JSON5 – это расширение стандарта JSON, которое предоставляет дополнительные возможности, например: комментарии, передачу ключей объектов без кавычек, использование одинарных кавычек для строк и дополнительные форматы для чисел. Он предназначен для упрощения написания конфигурационных файлов с дополнительным синтаксическим сахаром по сравнению с JSON.

5. Для работы с данными в формате JSON5 в Python можно использовать сторонние библиотеки, такие как `json5`, которые можно установить с помощью менеджера пакетов `pip`.

6. Для сериализации данных в формат JSON в Python предоставляется стандартный модуль `json`. С его помощью можно преобразовать данные из словарей, списков и других встроенных типов в строку JSON.

7. Функция `json.dump()` используется для сериализации объекта Python и сохранения результата в файл, в то время как `json.dumps()` сериализует объект Python и возвращает строку в формате JSON.

8. Для десериализации данных из формата JSON в Python используются функции ``json.load()`` и ``json.loads()``. ``json.load()`` читает JSON из файла и преобразует его в объект Python, тогда как ``json.loads()`` выполняет то же действие, но принимает JSON в виде строки.

9. Для работы с данными JSON, содержащими кириллицу, может потребоваться указать правильную кодировку при сериализации и десериализации данных (``encoding='utf-8'``). Модуль ``json`` автоматически кодирует и декодирует строки в ``utf-8``, что позволяет корректно работать с кириллицей.

10. JSON Schema – это спецификация, которая позволяет описывать структуру JSON-данных. С помощью схемы можно определить, какие поля должны присутствовать в JSON-объекте, их типы, ограничения и дополнительные правила.