

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

Отчет по лабораторной работе № 2.17

«Работа с данными формата JSON в языке Python»

по дисциплине «Основы программной инженерии»

Выполнила:
Образцова Мария Дмитриевна,
2 курс, группа ПИЖ-б-о-21-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2023 г.


Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

Create a new repository

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

Owner *

 obrMaria ▾

Repository name *


/ OPI_2.16 ✓

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

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:

Skip this step if you're importing an existing repository.

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore


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

.gitignore template: Python ▾

Choose a license

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

License: MIT License ▾

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – создание репозитория

```

M@DESKTOP-UVM9NOL MINGW64 ~/desktop (master)
$ git clone https://github.com/obrMaria/OPI_2.16.git
Cloning into 'OPI_2.16'...
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.

M@DESKTOP-UVM9NOL MINGW64 ~/desktop (master)
$ cd OPI_2.16

```

Рисунок 2 – клонирование репозитория

```

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_2.16 (main)
$ git add .

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_2.16 (main)
$ git commit -m "gitignore"
[main 7ee9513] gitignore
3 files changed, 140 insertions(+), 2 deletions(-)
create mode 100644 OPI_2.16.docx
create mode 100644 ~$I_2.16.docx

```

Рисунок 3 – сохранение изменений в файле gitignore

Проработайте примеры лабораторной работы. Создайте для них отдельные модули языка Python. Зафиксируйте изменения в репозитории.

```

/Users\A\Desktop\OPI_2.16\PY [
py      2      {
py      3          "name": "asdasd",
py      4          "post": "asd",
py      5          "year": 2021
py      6      },
py      7      {
py      8          "name": "asdasda",
py      9          "post": "qwe",
py     10          "year": 1996
py     11      },
py     12      {
py     13          "name": "kjhgjkh",
py     14          "post": "lkjg",
py     15          "year": 2001
py     16      }
py     17  ]

primer <
>>> load
>>> Неизвестная команда load
>>> save
>>> Неизвестная команда save
>>> save for primer
>>> add
>>> Неизвестная команда add
>>> list
Фамилия и инициалы? kjhgjkh
Должность? lkjg
Год поступления? 2001
>>> list
+-----+-----+-----+-----+
No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
1 | asdasd                    | asd                  | 2021          |
2 | asdasda                   | qwe                  | 1996          |
3 | kjhgjkh                   | lkjg                 | 2001          |
+-----+-----+-----+-----+
>>> save for primer
>>>

```

Индивидуальные 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.

```
1.py x my.py x my_1 x primer.py x
for student in staff:
    grade = student.get('grade', '')
    if sum(grade) / (len(grade)) >= 4.0:
        result.append(student)
        count += 1

return result

def save_students(file_name, staff):
    with open(file_name, "w", encoding="utf-8") as f:
        json.dump(staff, f, ensure_ascii=False, indent=4)

def load_students(file_name):
    with open(file_name, "r", encoding="utf-8") as f:
        return json.load(f)

def main():
    """
    Главная функция программы.
    """
    students = []

    # 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':
            get_student(students)
            save_students('my_1.json', students)

        elif command == 'load':
            students = load_students('my_1.json')

        elif command == 'show':
            show_students(students)

        else:
            print("Неизвестная команда")

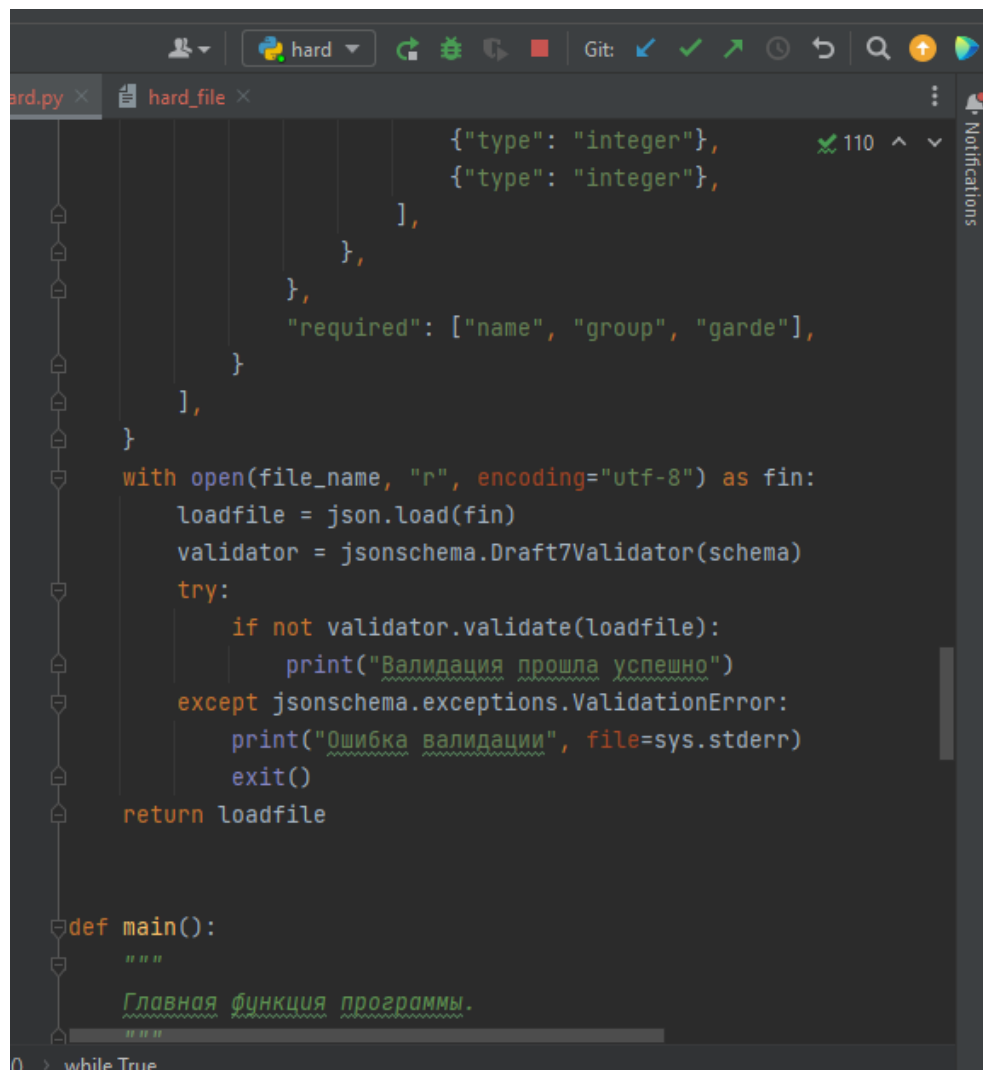
if __name__ == '__main__':
    main()
```

```
1.py x my.py x my_1 x
[
  {
    "name": "asdaasd",
    "group": "11",
    "grade": [
      4,
      4,
      5,
      5,
      5
    ]
  },
  {
    "name": "asdf",
    "group": "12",
    "grade": [
      4,
      5,
      3,
      2,
      5
    ]
  },
  {
    "name": "asfff",
    "group": "4",
    "grade": [
      3,
      3,
      3,
      2,
      3
    ]
  }
]
```

Задание повышенной сложности

Очевидно, что программа в примере 1 и в индивидуальном задании никак не проверяет правильность загружаемых данных формата JSON. В

следствие чего, необходимо после загрузки из файла JSON выполнять валидацию загруженных данных. Валидацию данных необходимо производить с использованием спецификации JSON Schema, описанной на сайте <https://json-schema.org/>. Одним из возможных вариантов работы с JSON Schema является использование пакета `jsonschema`, который не является частью стандартной библиотеки Python. Таким образом, необходимо реализовать валидацию загруженных данных с помощью спецификации JSON Schema.



```
ard.py x hard_file x
{"type": "integer"},
{"type": "integer"},
],
},
},
"required": ["name", "group", "garde"],
}
],
}
with open(file_name, "r", encoding="utf-8") as fin:
    loadfile = json.load(fin)
    validator = jsonschema.Draft7Validator(schema)
    try:
        if not validator.validate(loadfile):
            print("Валидация прошла успешно")
    except jsonschema.exceptions.ValidationError:
        print("Ошибка валидации", file=sys.stderr)
        exit()
    return loadfile

def main():
    """
    Главная функция программы.
    """
    while True
```

The screenshot shows an IDE with a project named '123' located at 'C:\Users\student-09-525\Desktop\123'. The file explorer on the left shows a 'venv' directory and files '1', '2.17_primer.py', 'main.py', and 'mars.py'. The main editor displays the code for 'mars.py', which uses the 'marshmallow' library for validation. The code includes imports for 'sys', 'json', and 'marshmallow' (Schema, fields, ValidationError, EXCLUDE). A 'StudentsSchema' class is defined with fields for 'name' (Str), 'group' (Str), and 'grade' (Dict). A 'get_student' function is partially visible. The Run console at the bottom shows the command 'C:\Users\student-09-525\Desktop\123\venv\Scripts\python.exe C:\Users\student-09-525\Desktop\123\mars.py' and the interactive session output: '>>> add', 'Фамилия и инициалы? asd', 'Номер группы? asd', 'введите свои оценки: 1 2 3 4 5', '>>> save 1', '>>> load 1', and 'Validation was successful'.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5 import json
6
7 from marshmallow import Schema, fields, ValidationError, EXCLUDE
8
9 class StudentsSchema(Schema):
10     name = fields.Str()
11     group = fields.Str()
12     grade = fields.Dict()
13
14 def get_student(staff):
```

Run: mars x

C:\Users\student-09-525\Desktop\123\venv\Scripts\python.exe C:\Users\student-09-525\Desktop\123\mars.py

>>> add
Фамилия и инициалы? asd
Номер группы? asd
введите свои оценки: 1 2 3 4 5
>>> save 1
>>> load 1
Validation was successful

С использованием библиотеки marshmallow

The screenshot shows an IDE with a project named '123' located at 'C:\Users\student-09-525\Desktop\123'. The file explorer on the left shows a 'venv' directory and files '1', '2.17_primer.py', 'mars.py', and 'pyd.py'. The main editor displays the code for 'pyd.py', which uses the 'pydantic' library for validation. The code includes imports for 'sys', 'json', and 'pydantic' (BaseModel, ValidationError, validator). A 'StudentsSchema' class is defined as a 'BaseModel' with fields 'name' (str), 'group' (str), and 'grade' (list). A 'get_student' function is defined. The Run console at the bottom shows the command 'C:\Users\student-09-525\Desktop\123\venv\Scripts\python.exe C:\Users\student-09-525\Desktop\123\pyd.py' and the interactive session output: '>>> add', 'Фамилия и инициалы? asd', 'Номер группы? asd', 'введите свои оценки: 1 2 3 4 5', '>>> save 1', '>>> load 1', and 'Validation was successful'.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5 import json
6
7 from pydantic import BaseModel, ValidationError, validator
8
9 class StudentsSchema(BaseModel):
10     name: str
11     group: str
12     grade: list
13
14
15 def get_student(staff):
16     """
17     get_student()
18 """
```

Run: pyd x

C:\Users\student-09-525\Desktop\123\venv\Scripts\python.exe C:\Users\student-09-525\Desktop\123\pyd.py

>>> add
Фамилия и инициалы? asd
Номер группы? asd
введите свои оценки: 1 2 3 4 5
>>> save 1
>>> load 1
Validation was successful

С использованием библиотеки pydantic

Вопросы для защиты работы

1. Для чего используется JSON?

JSON – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом.

2. Какие типы значений используются в JSON?

Как было показано ранее JSON-текст представляет собой (в закодированном виде) одну из двух структур:

- Набор пар ключ: значение. В различных языках это реализовано как запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка (регистрозависимость не регулируется стандартом, это остаётся на усмотрение программного обеспечения. Как правило, регистр учитывается программами — имена с буквами в разных регистрах считаются разными, значением — любая форма. Повторяющиеся имена ключей допустимы, но не рекомендуются стандартом; обработка таких ситуаций происходит на усмотрение программного обеспечения, возможные варианты — учитывать только первый такой ключ, учитывать только последний такой ключ, генерировать ошибку.

- Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

3. Как организована работа со сложными данными в JSON?

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

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

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

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

`json.dump()` # конвертировать python объект в json и записать в файл

`json.dumps()` # тоже самое, но в строку

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

`json.load()` # прочитать json из файла и конвертировать в python объект

`json.loads()` # тоже самое, но из строки с json (s на конце от string/строка)

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

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema? Что такое схема данных? Приведите схему данных для примера 1.