

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.16**

**дисциплина «Программирование на языке Python»**

Выполнил:

Тихоненко Борис Витальевич

2 курс, группа ИТС-б-з-22-1,

11.03.02

---

(подпись)

Проверил:

доцент, кандидат технических наук

Воронкин Роман Александрович

---

(подпись)

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

Ставрополь, 2024 г.

## Цель работы:

Приобретение навыков по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

## Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: <https://github.com/afk552/lab2.16>

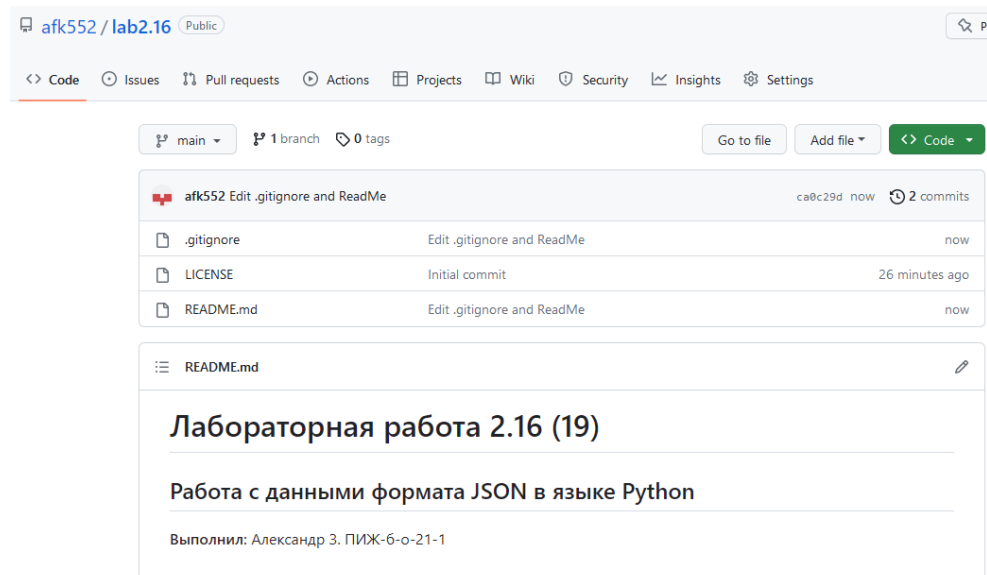


Рисунок 1 – Удаленный репозиторий на GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

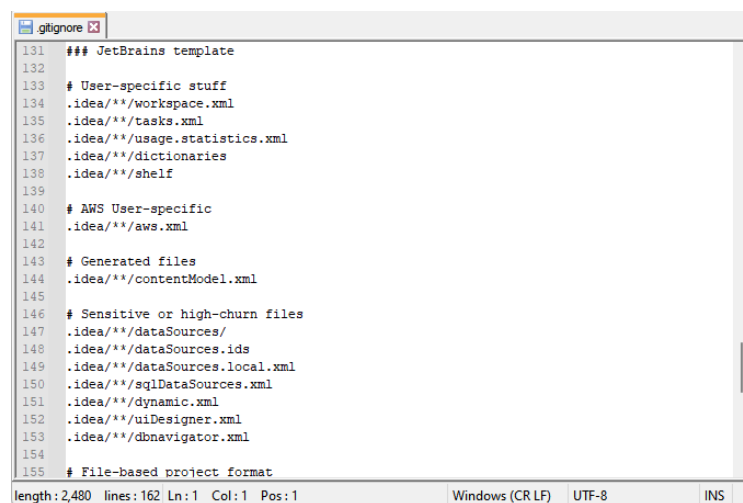
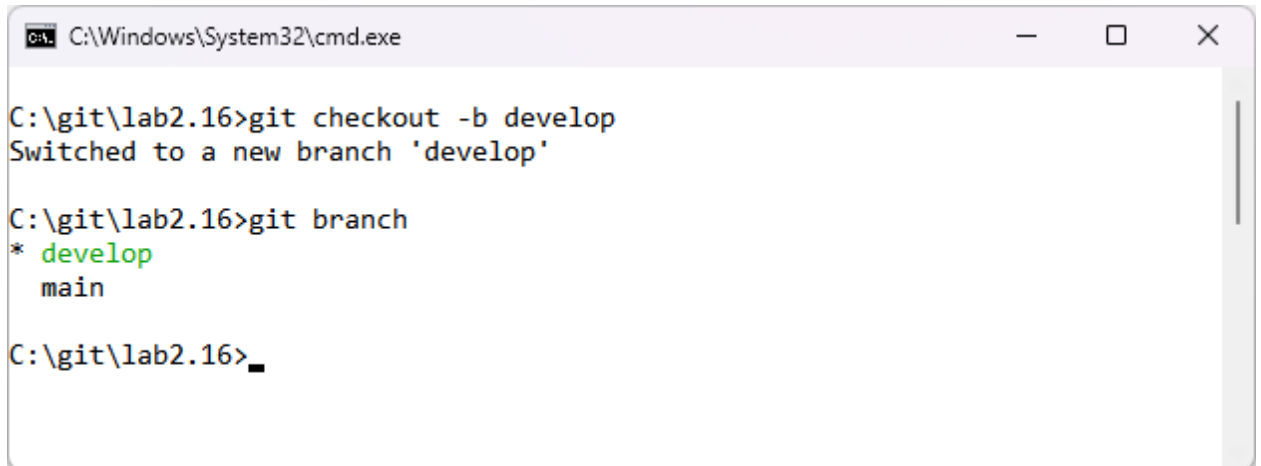


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствии с моделью ветвления git-flow, рисунок 3.



```
C:\Windows\System32\cmd.exe

C:\git\lab2.16>git checkout -b develop
Switched to a new branch 'develop'

C:\git\lab2.16>git branch
* develop
  main

C:\git\lab2.16>_
```

Рисунок 3 – Окно командной строки

Создайте проект PyCharm в папке репозитория, рисунок 4.

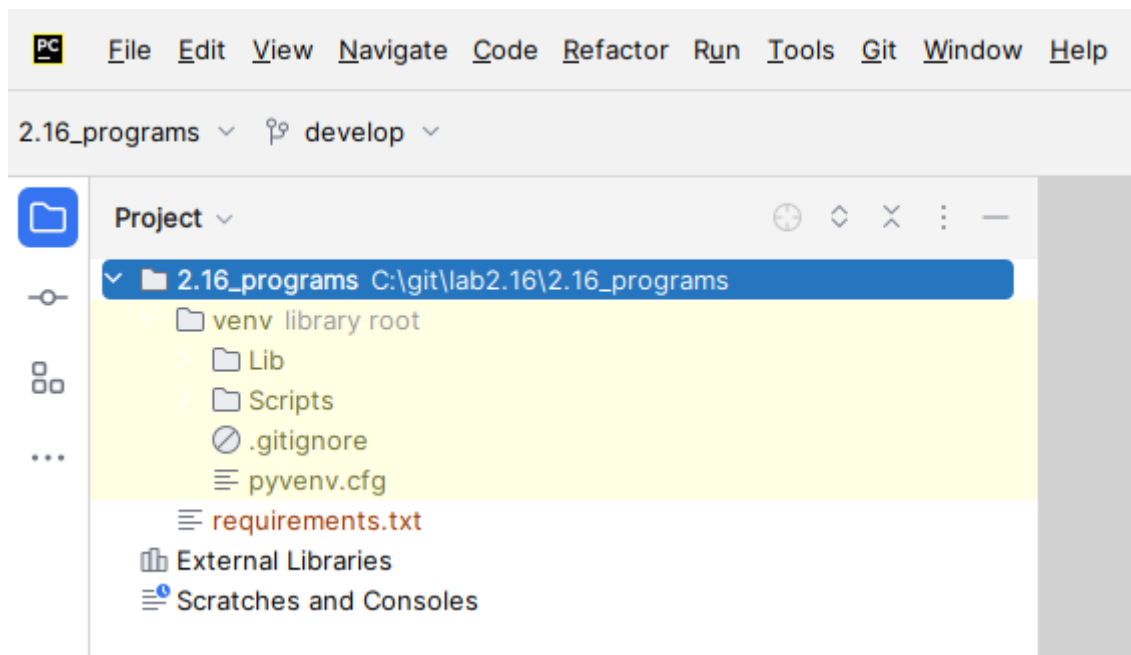
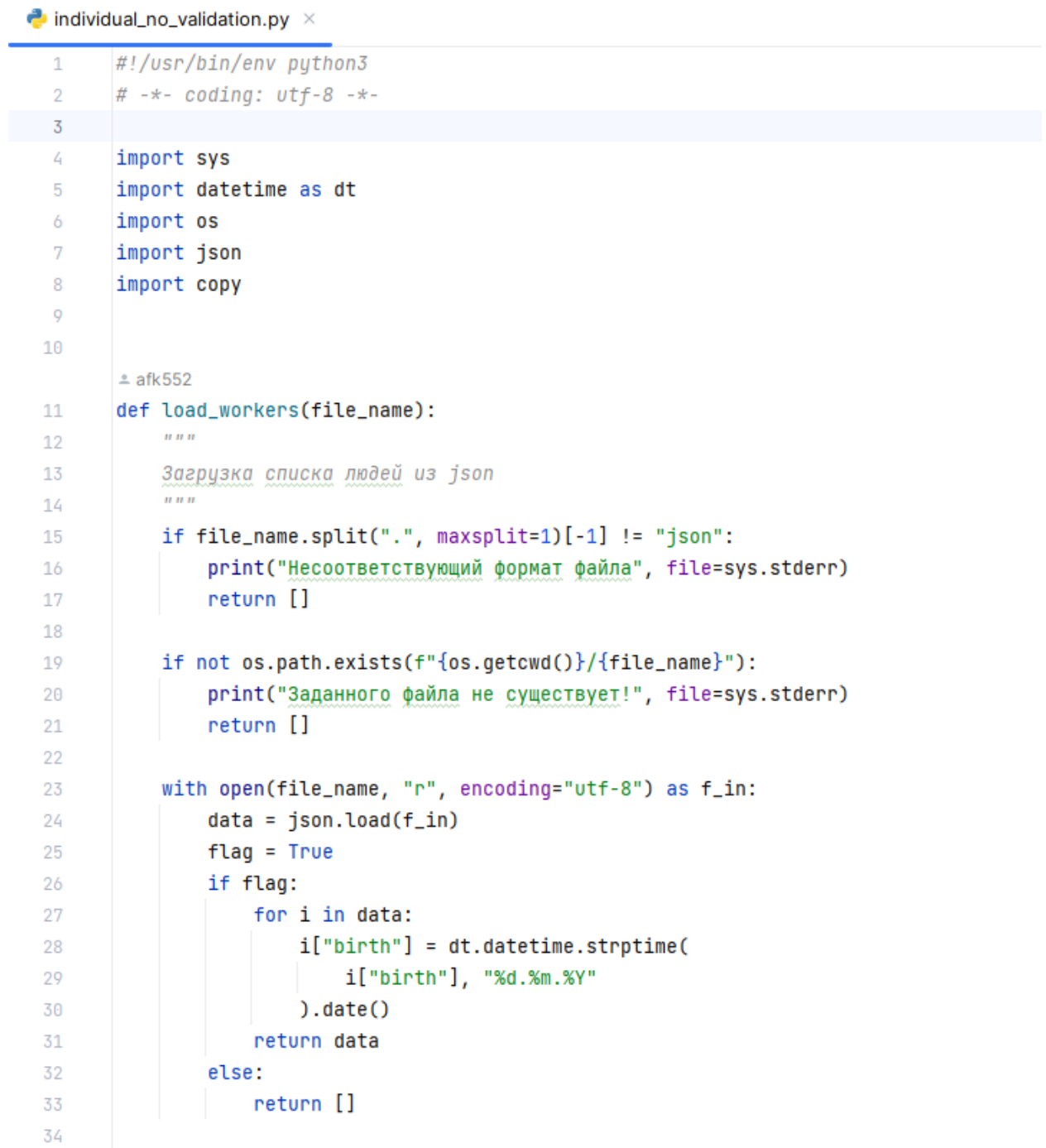


Рисунок 4 – Окно проекта в PyCharm

## Индивидуальное задание.

**Задание 1.** 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.



```
individual_no_validation.py ×
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  import datetime as dt
6  import os
7  import json
8  import copy
9
10
11  def load_workers(file_name):
12      """
13      Загрузка списка людей из json
14      """
15      if file_name.split(".", maxsplit=1)[-1] != "json":
16          print("Несоответствующий формат файла", file=sys.stderr)
17          return []
18
19      if not os.path.exists(f"{os.getcwd()}/{file_name}"):
20          print("Заданного файла не существует!", file=sys.stderr)
21          return []
22
23      with open(file_name, "r", encoding="utf-8") as f_in:
24          data = json.load(f_in)
25          flag = True
26          if flag:
27              for i in data:
28                  i["birth"] = dt.datetime.strptime(
29                      i["birth"], "%d.%m.%Y"
30                  ).date()
31              return data
32          else:
33              return []
34
```

Рисунок 5 – Код программы индивидуального задания 1 (1)

```
individual_no_validation.py x
afk552
36 def save_workers(file_name, people_list):
37     """
38     Сохранение списка людей в json
39     """
40     flag = False
41     # Проверка заданного имени файла
42     if file_name.split(".", maxsplit=1)[-1] != "json":
43         print("Заданный формат файла не .json", file=sys.stderr)
44         return False
45
46     # Проверка файла gitignore, если есть - далее
47     content = os.listdir(os.getcwd())
48     if ".gitignore" in content:
49         flag = True
50     if not flag:
51         file = open(".gitignore", "w")
52         file = f"{os.getcwd()}/.gitignore"
53         with open(file, "r+", encoding="utf-8") as gi:
54             if file_name not in gi:
55                 gi.write(f"{file_name}\n")
56
57     # Делаем копию списка, чтобы его не затронуть
58     lst = copy.deepcopy(people_list)
59     # Сериализация даты в строку для записи в файл
60     for i in lst:
61         i['birth'] = i['birth'].strftime("%d.%m.%Y")
62
63     # Дамп в json списка
64     with open(file_name, "w", encoding="utf-8") as f_out:
65         json.dump(lst, f_out, ensure_ascii=False, indent=4)
66     lst.clear()
67
```

Рисунок 6 – Код программы индивидуального задания 1 (2)

```
184 elif command.startswith("save "):
185     parts = command.split(" ", maxsplit=1)
186     fn = parts[1].split(".")
187     file_name = f"{fn[0]}.{fn[1]}"
188     save_workers(file_name, people)
189
190 elif command.startswith("load "):
191     parts = command.split(" ", maxsplit=1)
192     file_name = parts[1]
193     temp = copy.deepcopy(people)
194     people = load_workers(file_name)
195     if not people:
196         people = copy.deepcopy(temp)
197     temp.clear()
198
```

Рисунок 7 – Код программы индивидуального задания 1 (3)

```
Run
C:\git\lab2.16\2.16_programs\venv\Scripts\python.exe C:\git\lab2.16\2.16_programs\ind
Программа запущена, введите help для просмотра команд!
>>> add
Введите фамилию и имя через пробел: Имя Фамилия
Введите номер телефона: +79855310868
Введите дату рождения (01.01.2077): 02.02.2001
>>> list
+-----+-----+-----+-----+
| №п/п |      Фамилия Имя      | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
|    1 | Имя Фамилия          | +79855310868   | 02.02.2001   |
+-----+-----+-----+-----+
>>> save test.json
>>> list
+-----+-----+-----+-----+
| №п/п |      Фамилия Имя      | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
|    1 | Имя Фамилия          | +79855310868   | 02.02.2001   |
+-----+-----+-----+-----+
>>> load test.json
>>> list
+-----+-----+-----+-----+
| №п/п |      Фамилия Имя      | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
|    1 | Имя Фамилия          | +79855310868   | 02.02.2001   |
+-----+-----+-----+-----+
>>> load not_exist.json
>>> Заданного файла не существует!
load something
>>> Несоответствующий формат файла
```

Рисунок 8 – Результат выполнения программы индивидуального задания 1  
(1)

```
.gitignore
1 test.json
2

test.json
1 [
2   {
3     "name": "Имя Фамилия",
4     "pnumber": "+79855310868",
5     "birth": "02.02.2001"
6   }
7 ]
```

Рисунок 9 – Результат выполнения программы индивидуального задания 1  
(2)

**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://jsonschema.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, marshmallow, pydantic.

Валидация будет проверяться на основе этих JSON-файлов

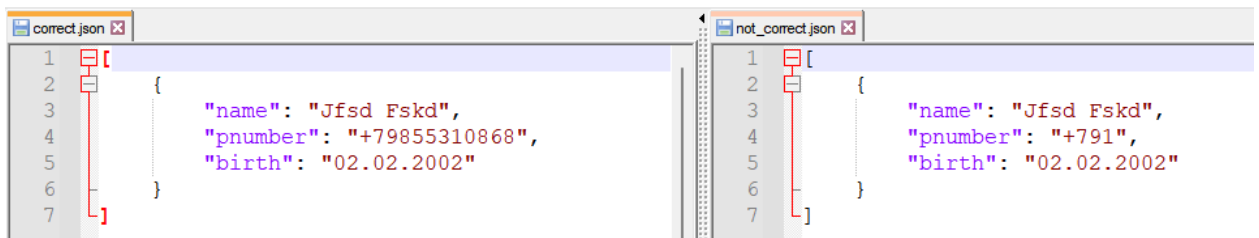


Рисунок 10 – JSON-файлы для валидации

### 1) Валидация через Jonschema

```
9      import jsonschema
10     from jsonschema import validate
11
12
13     def validate_json(json_data):
14         with open("scheme.json", "r", encoding="utf-8") as json_schema:
15             schema = json.load(json_schema)
16         try:
17             validate(instance=json_data, schema=schema)
18         except jsonschema.exceptions.ValidationError as err:
19             return False
20         return True
```

Рисунок 11 – Код программы индивидуального задания (jsonschema) (1)

```

23 def load_workers(file_name):
24     """
25     Загрузка списка людей из json
26     """
27     if file_name.split(".", maxsplit=1)[-1] != "json":
28         print("Несоответствующий формат файла", file=sys.stderr)
29         return []
30
31     if not os.path.exists(f"{os.getcwd()}/{file_name}"):
32         print("Заданного файла не существует!", file=sys.stderr)
33         return []
34
35     with open(file_name, "r", encoding="utf-8") as f_in:
36         data = json.load(f_in)
37         is_valid = validate_json(data)
38
39         if is_valid:
40             for i in data:
41                 i["birth"] = dt.datetime.strptime(
42                     i["birth"], "%d.%m.%Y"
43                 ).date()
44             return data
45         else:
46             print("Задан некорректный файл!", file=sys.stderr)
47             return []
48

```

Рисунок 12 – Код программы индивидуального задания (jsonschema) (2)

```

C:\git\lab2.16\2.16_programs\venv\Scripts\python.exe C:\git\lab2.16\2.16_programs\in
Программа запущена, введите help для просмотра команд!
>>> load correct.json
>>> list
+-----+-----+-----+-----+
| №п/п |      Фамилия Имя      | Номер телефона |  Дата рождения  |
+-----+-----+-----+-----+
|    1  | Jfsd Fskd             | +79855310868   | 02.02.2002      |
+-----+-----+-----+-----+
>>> load not_correct.json
>>> Задан некорректный файл!
list
+-----+-----+-----+-----+
| №п/п |      Фамилия Имя      | Номер телефона |  Дата рождения  |
+-----+-----+-----+-----+
|    1  | Jfsd Fskd             | +79855310868   | 02.02.2002      |
+-----+-----+-----+-----+
>>>

```

Рисунок 13 – Результат выполнения программы индивидуального задания (jsonschema)



## 2) Валидация через marshmallow

```
individual_marshmallow_validation.py ×
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  import datetime as dt
6  import os
7  import json
8  import copy
9  from marshmallow import Schema, fields, ValidationError, validate
10
11
12  def validate_json(json_data):
13      name_reg = r"^[a-zA-ZА-Яа-я]+ [a-zA-ZА-Яа-я]+$"
14      pnumber_reg = r"^(\\+7|7|8)?[\\s\\-]?\\(?[489][0-9]{2}\\)?[\\s\\-]" \\
15                  r"?[0-9]{3}[\\s\\-]?[0-9]{2}[\\s\\-]?[0-9]{2}$"
16      birth_reg = (
17          r"^(0[1-9]|[12][0-9]|3[01])[.](0[1-9]|1[012])[.](19|20)[0-9]{2}$"
18      )
19
20      class PeopleSchema(Schema):
21          name = fields.Str(validate=validate.Regexp(name_reg))
22          pnumber = fields.Str(validate=validate.Regexp(pnumber_reg))
23          birth = fields.Str(validate=validate.Regexp(birth_reg))
24
25      try:
26          people = PeopleSchema().load(json_data, many=True)
27          return people
28      except ValidationError:
29          return False
```

Рисунок 14 – Код программы индивидуального задания (marshmallow) (1)

```
32  def load_workers(file_name):
33      """
34      Загрузка списка людей из json
35      """
36      if file_name.split(".", maxsplit=1)[-1] != "json":
37          print("Несоответствующий формат файла", file=sys.stderr)
38          return []
39
40      if not os.path.exists(f"{os.getcwd()}/{file_name}"):
41          print("Заданного файла не существует!", file=sys.stderr)
42          return []
43
44      with open(file_name, "r", encoding="utf-8") as f_in:
45          data = json.load(f_in)
46          is_valid = validate_json(data)
47
48          if is_valid:
49              for i in data:
50                  i["birth"] = dt.datetime.strptime(
51                      i["birth"], "%d.%m.%Y"
52                  ).date()
53              return data
54          else:
55              print("Задан некорректный файл!", file=sys.stderr)
56              return []
```

Рисунок 15 – Код программы индивидуального задания (marshmallow) (2)

```

Run
C:\git\lab2.16\2.16_programs\venv\Scripts\python.exe C:\git\lab2.16\2.16_programs\in
Программа запущена, введите help для просмотра команд!
>>> load correct.json
>>> list
+-----+-----+-----+-----+
| №п/п |      Фамилия Имя      | Номер телефона |   Дата рождения   |
+-----+-----+-----+-----+
|    1  | Jfsd Fskd             | +79855310868   |    02.02.2002    |
+-----+-----+-----+-----+

>>> load not_correct.json
>>> Задан некорректный файл!
list
+-----+-----+-----+-----+
| №п/п |      Фамилия Имя      | Номер телефона |   Дата рождения   |
+-----+-----+-----+-----+
|    1  | Jfsd Fskd             | +79855310868   |    02.02.2002    |
+-----+-----+-----+-----+

>>>

```

Рисунок 16 – Результат выполнения программы индивидуального задания (marshmallow)

### 3) Валидация через pydantic

```

individual_pydantic_validation.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  import datetime as dt
6  import os
7  import json
8  import copy
9  from pydantic import BaseModel, constr, ValidationError
10
11
12  def validate_json(json_data):
13      name_reg = r"^[a-zA-ZА-Яа-я]+ [a-zA-ZА-Яа-я]+$"
14      pnumber_reg = (
15          r"^(\\+7|7|8)?[\\s\\-]?\\((?[489][0-9]{2})\\)?[\\s\\-]"
16          r"?[0-9]{3}[\\s\\-]?[0-9]{2}[\\s\\-]?[0-9]{2}$"
17      )
18      birth_reg = (
19          r"^(0[1-9]|[12][0-9]|3[01])[.](0[1-9]|1[012])[.](19|20)[0-9]{2}$"
20      )
21
22      class PersonModel(BaseModel):
23          name: constr(regex=name_reg)
24          pnumber: constr(regex=pnumber_reg)
25          birth: constr(regex=birth_reg)
26
27      try:
28          for row in json_data:
29              PersonModel.parse_obj(row)
30          return json_data
31      except ValidationError:
32          return False

```

Рисунок 17 – Код программы индивидуального задания (pydantic) (1)

```

35 def load_workers(file_name):
36     """
37     Загрузка списка людей из json
38     """
39     if file_name.split(".", maxsplit=1)[-1] != "json":
40         print("Несоответствующий формат файла", file=sys.stderr)
41         return []
42
43     if not os.path.exists(f"{os.getcwd()}/{file_name}"):
44         print("Заданного файла не существует!", file=sys.stderr)
45         return []
46
47     with open(file_name, "r", encoding="utf-8") as f_in:
48         data = json.load(f_in)
49         is_valid = validate_json(data)
50
51         if is_valid:
52             for i in data:
53                 i["birth"] = dt.datetime.strptime(
54                     i["birth"], "%d.%m.%Y"
55                 ).date()
56             return data
57         else:
58             print("Задан некорректный файл!", file=sys.stderr)
59             return []

```

Рисунок 18 – Код программы индивидуального задания (pydantic) (2)

```

C:\git\lab2.16\2.16_programs\venv\Scripts\python.exe C:\git\lab2.16\2.16_programs\in
Программа запущена, введите help для просмотра команд!
>>> load correct.json
>>> list
+-----+-----+-----+-----+
| №п/п |      Фамилия Имя      | Номер телефона |   Дата рождения   |
+-----+-----+-----+-----+
|    1 | Jfsd Fskd             | +79855310868   |    02.02.2002     |
+-----+-----+-----+-----+
>>> load not_correct.json
>>> Задан некорректный файл!
list
+-----+-----+-----+-----+
| №п/п |      Фамилия Имя      | Номер телефона |   Дата рождения   |
+-----+-----+-----+-----+
|    1 | Jfsd Fskd             | +79855310868   |    02.02.2002     |
+-----+-----+-----+-----+
>>>

```

Рисунок 19 – Результат выполнения программы индивидуального задания (pydantic)

**Вывод:** В результате выполнения работы была изучена работа с json-файлами в языке Python, а именно: чтение, запись, а также валидация данных в этих файлах.

## **Контрольные вопросы:**

### **1. Для чего используется JSON?**

JSON (англ. *JavaScript Object Notation*, обычно произносится как /'dʒeɪsən/ JAY-sən) – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом. Используется для сериализации сложных структур данных.

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

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, т.е. 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. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты ' , " , \ , \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.

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

Данные JSON записываются в виде пар "имя/значение".

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

JSON5 — предложенное расширение формата json в соответствии с синтаксисом ECMAScript 5, вызванное тем, что json используется не только для общения между программами, но и создаётся/редактируется вручную. Файл JSON5 всегда является корректным кодом ECMAScript 5. JSON5 обратно совместим с JSON. Для некоторых языков программирования уже существуют парсеры json5.

Нововведения:

- Поддерживаются как однострочные `//`, так и многострочные `/* */` комментарии.
- Записи и списки могут иметь запятую после последнего элемента (удобно при копировании элементов).
- Ключи записей могут быть без кавычек, если они являются валидными идентификаторами ECMAScript 5.
- Строки могут заключаться как в одинарные, так и в двойные кавычки.
- Числа могут быть в шестнадцатеричном виде, начинаться или заканчиваться десятичной точкой, включать Infinity, -Infinity, NaN и -NaN, начинаться со знака +.

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

Библиотеки: PyJSON5, json5.

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

Модуль json, который позволяет использовать методы dump и dumps.

## 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?

`ensure_ascii=False`

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

JSON-schema — это стандарт описания структур данных в формате JSON, разрабатываемый на основе XML-Schema.

