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 ПРАКТИЧЕСКОЙ РАБОТЕ №2
дисциплины «Анализ данных»
Вариант 23

Выполнил:
Мотовилов Вадим Борисович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Информатика и вычислительная
техника», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 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.

Порядок выполнения работы:

1. Создал репозиторий и скопировал его

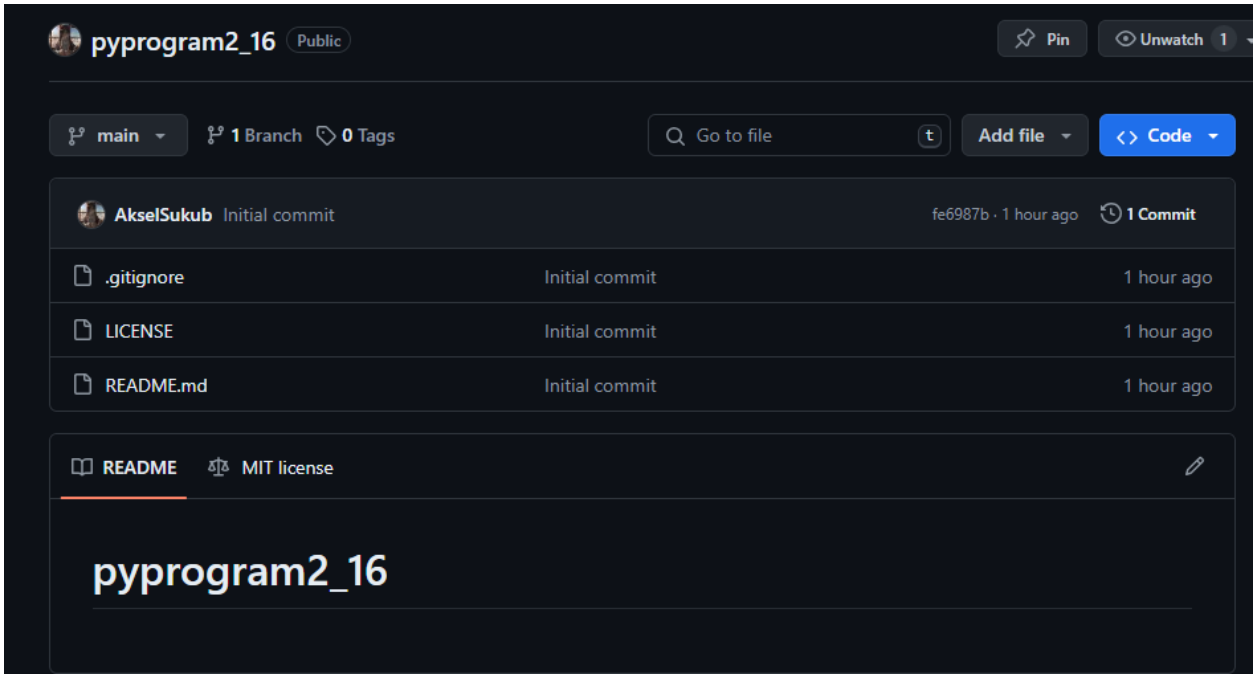

Рисунок 1. Созданный репозиторий

2. Изменил файл .gitignore и README.rm и добавил git flow

```
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
#  JetBrains specific template is maintained in a separate JetBrains.gitignore that can
#  be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
#  and can be added to the global gitignore or merged into this file.  For a more nuclear
#  option (not recommended) you can uncomment the following to ignore the entire idea folder.
.idea/
**/.DS_Store
.vscode
*.dSYM
```

Рисунок 2. Измененный файл .gitignore

3. Выполнил пример и задания

```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,
    }
```

```python
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 = []
    # 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":
            # Вывесфввти 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()
```

```
(base) C:\otkat>python C:\otkat\progr6\prim1.py
>>> add
Фамилия и инициалы? Епифанов А. А
Должность? Студент
Год поступления? 2022
>>> list
+------+-----------------------------+---------------------+----------+
|  No  |           Ф.И.О.            |      Должность      |   Год    |
+------+-----------------------------+---------------------+----------+
|    1 | Епифанов А. А               | Студент             |     2022 |
+------+-----------------------------+---------------------+----------+
>>> load C:\otkat\primer.json
>>> add
Фамилия и инициалы? Епифанов А. А
Должность? Студент
Год поступления? 2022
>>> list
+------+-----------------------------+---------------------+----------+
|  No  |           Ф.И.О.            |      Должность      |   Год    |
+------+-----------------------------+---------------------+----------+
|    1 | Motovilov V. B.             | Student             |     2022 |
|    2 | Епифанов А. А               | Студент             |     2022 |
+------+-----------------------------+---------------------+----------+
>>> save C:\otkat\primer.json
>>>
```

Рисунок 3 . Выполнение примера

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.*

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

import json
import sys
from datetime import date


def get_worker():
    """
    Запросить данные о рейсе.
    """
    point = input("Пункт назначения? ")
    number = int(input("Номер рейса? "))
    type = input("Тип самолета? ")

    # Создать словарь.
    return {
        "point": point,
        "number": number,
        "type": type,
    }


def display_workers(staff):
    """
    Отобразить список рейсов.
    """
    # Проверить, что список рейсов не пуст.
    if staff:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format(
            "-" * 4, "-" * 30, "-" * 10, "-" * 20
        )
        print(line)
        print(
            "| {:^4} | {:^30} | {:^10} | {:^20} |".format(
                "No", "Пункт назначения", "No рейса", "Тип самолета"
            )
        )
        print(line)
        # Вывести данные о всех рейсах.
        for idx, worker in enumerate(staff, 1):
            print(
                "| {:>4} | {:<30} | {:<10} | {:>20} |".format(
                    idx,
```

```
                    worker.get("point", ""),
                    worker.get("number", 0),
                    worker.get("type", ""),
                )
            )
        print(line)
    else:
        print("Список рейсов пуст.")


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


def save_workers(file_name, staff):
    """
    Сохранить все рейсы 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 все рейсы 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 рейсов.
    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 рейсе.
            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 все рейсы.
            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 рейсы с заданным стажем.
            selected = select_workers(workers, period)
            # Отобразить выбранные рейсы.
            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":
            # Вывесфввти 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()
```

```
(base) C:\otkat>python C:\otkat\progr6\ind1.py
>>> add
Пункт назначения? Москва
Номер рейса? 13
Тип самолета? Боинг 737
>>> add
Пункт назначения? Моздок
Номер рейса? 33
Тип самолета? Миг-22
>>> list
+------+----------------------------------+-----------+----------------------+
|  No  |         Пункт назначения         |  No рейса |     Тип самолета     |
+------+----------------------------------+-----------+----------------------+
|    1 | Москва                           | 13        |           Боинг 737 |
|    2 | Моздок                           | 33        |             Миг-22 |
+------+----------------------------------+-----------+----------------------+
>>> save lab2ind1.json
>>> exit
```

Рисунок 4 . Выполнение 1 задания

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.

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

import json
import sys
from datetime import date
from jsonschema import validate, ValidationError


def get_worker():
    """
    Запросить данные о рейсе.
    """
    point = input("Пункт назначения? ")
    number = int(input("Номер рейса? "))
    typex = input("Тип самолета? ")

    # Создать словарь.
    return {
```

```
        "point": point,
        "number": number,
        "typex": typex,
    }


def display_workers(staff):
    """
    Отобразить список рейсов.
    """
    # Проверить, что список рейсов не пуст.
    if staff:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format(
            "-" * 4, "-" * 30, "-" * 10, "-" * 20
        )
        print(line)
        print(
            "| {:^4} | {:^30} | {:^10} | {:^20} |".format(
                "No", "Пункт назначения", "No рейса", "Тип самолета"
            )
        )
        print(line)
        # Вывести данные о всех рейсах.
        for idx, worker in enumerate(staff, 1):
            print(
                "| {:>4} | {:<30} | {:<10} | {:>20} |".format(
                    idx,
                    worker.get("point", ""),
                    worker.get("number", 0),
                    worker.get("typex", ""),
                )
            )
        print(line)
    else:
        print("Список рейсов пуст.")


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

```python
def save_workers(file_name, staff):
    """
    Сохранить все рейсы 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 все рейсы из файла JSON.
    """
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "point": {"type": "string"},
                "number": {"type": "integer"},
                "typex": {"type": "string"},
            },
            "required": [
                "point",
                "number",
                "typex",
            ],
        },
    }
    '''# Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)'''
        # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        data = json.load(fin)

    try:
        # Валидация
        validate(instance=data, schema=schema)
        print("JSON валиден по схеме.")
    except ValidationError as e:
        print(f"Ошибка валидации: {e.message}")
    return data


def main():
    """
```

```
    Главная функция программы.
    """
    # Список рейсов.
    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 рейсе.
            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 все рейсы.
            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 рейсы с заданным стажем.
            selected = select_workers(workers, period)
            # Отобразить выбранные рейсы.
            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":
            # Вывесфввти 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()
```

```
(base) C:\otkat>C:/Users/1/anaconda3/python.exe c:/otkat/progr6/indsl1.py
>>> add
Пункт назначения? СПб
Номер рейса? 55
Тип самолета? Боинг 717
>>> add
Пункт назначения? Ставрополь
Номер рейса? 33
Тип самолета? ВВ-21
>>> list
+------+--------------------------------+------------+----------------------+
|  No  |        Пункт назначения        |  No рейса  |     Тип самолета     |
+------+--------------------------------+------------+----------------------+
|    1 | СПб                            | 55         |           Боинг 717 |
|    2 | Ставрополь                     | 33         |                ВВ-21 |
+------+--------------------------------+------------+----------------------+
>>> save C:\otkat\lab2indsl1.json
>>> load C:\otkat\lab2indsl1.json
JSON валиден по схеме.
>>> list
+------+--------------------------------+------------+----------------------+
|  No  |        Пункт назначения        |  No рейса  |     Тип самолета     |
+------+--------------------------------+------------+----------------------+
|    1 | СПб                            | 55         |           Боинг 717 |
|    2 | Ставрополь                     | 33         |                ВВ-21 |
+------+--------------------------------+------------+----------------------+
>>>
```

Ответы на контрольные вопросы:

1. Для чего используется JSON? - JSON используется для обмена и хранения данных в структурированном формате.
2. Какие типы значений используются в JSON? - JSON использует строки, числа, логические значения, массивы, объекты и null.
3. Как организована работа со сложными данными в JSON? - Сложные данные в JSON строятся с помощью вложенных структур.
4. Самостоятельно ознакомьтесь с форматом данных JSON5? В чем отличие этого формата от формата данных JSON? - JSON5 расширяет JSON, добавляя поддержку комментариев, разных форматов чисел, строк и ключей.

5. Какие средства языка программирования Python могут быть использованы для работы с данными в формате JSON5? - Для работы с JSON5 в Python требуются сторонние библиотеки.