

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе № 11

Дисциплина: «Программирование на Python»

Тема: «Работа с данными формата JSON в языке Python»

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

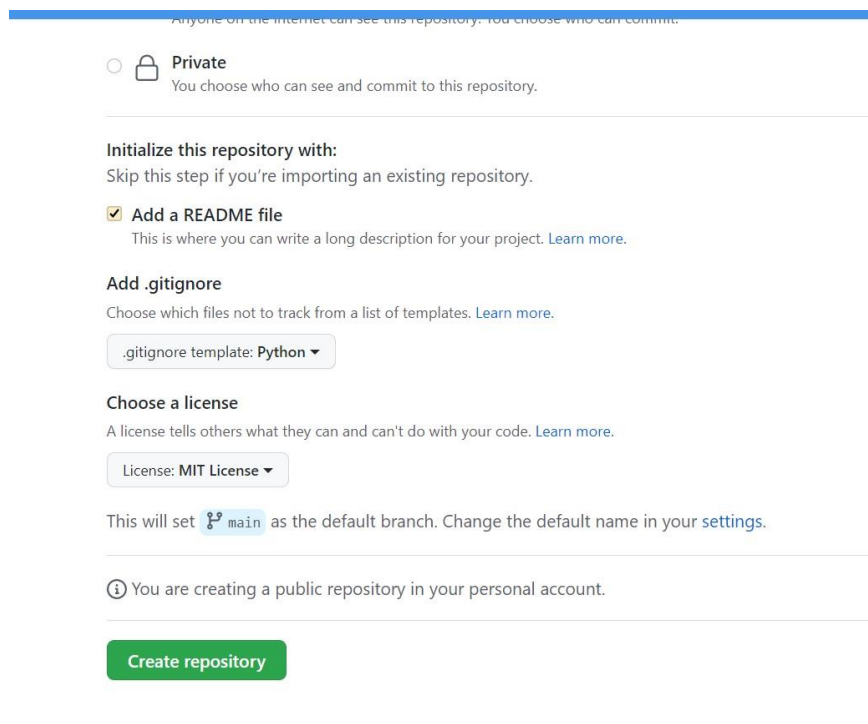
Толубаев Рамиль Ахметович

Ставрополь 2022

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

Порядок выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.



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

2. Выполните клонирование созданного репозитория.

```
C:\Users\User>cd C:\Users\User\Desktop\2 кypc Python\lab 19
C:\Users\User\Desktop\2 кypc Python\lab 19>git clone https://github.com/aikanyshkauanbekova/2.16.git
Cloning into '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.
C:\Users\User\Desktop\2 кypc Python\lab 19>
```

Рисунок 2. Клонирование репозитория

4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```

C:\Users\User\Desktop\2 курс Python\lab 10\lab-10>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/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/User/Desktop/2 курс Python/lab 10/lab-10/.git/hooks]
C:\Users\User\Desktop\2 курс Python\lab 10\lab-10>

```

Рисунок 4. Организован модель ветвления git flow

5.Проработайте примеры лабораторной работы. Создайте для них отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

```

#!/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,
    }

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 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:
        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':
        # 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("exit - завершить работу с программой.")
    else:
        print(f"Неизвестная команда {command}", file=sys.stderr)

```

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

Рисунок 5. Примеры лаб работы

6. 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.

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
  
import json  
import sys  
  
def get_route():  
    """  
    Запросить данные о маршруте  
    """  
    start = input("Название начального пункта маршрута? ")  
    finish = input("Название конечного пункта маршрута? ")  
    number = int(input("Номер маршрута? "))  
  
    return {  
        'start': start,  
        'finish': finish,  
        'number': number,  
    }  
  
def display_route(routes):  
    """  
    Отобразить список маршрутов  
    """  
    if routes:  
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(  
            '-' * 4,  
            '-' * 30,  
            '-' * 20,  
            '-' * 8  
        )  
        print(line)  
        print(  
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(  
                "№",  
                "Начальный пункт",  
                "Конечный пункт",  
                "Номер маршрута"  
            )  
        )  
    )
```

```

        print(line)

        for idx, worker in enumerate(routes, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('start', ''),
                    worker.get('finish', ''),
                    worker.get('number', 0)
                )
            )
        print(line)
    else:
        print("Список маршрутов пуст.")

def select_route(routes, period):
    """
    Выбрать маршрут
    """
    result = []
    for employee in routes:
        if employee.get('number') == period:
            result.append(employee)

    return result

def save_routes(file_name, staff):
    """
    Сохранить всех работников в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_routes(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

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

    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break

        elif command == 'add':
            route = get_route()
            routes.append(route)
            if len(routes) > 1:
                routes.sort(key=lambda item: item.get('number', ''))

        elif command == 'list':
            display_route(routes)

```

```

elif command.startswith('select'):
    parts = command.split(' ', maxsplit=1)
    period = int(parts[1])

    selected = select_route(routes, period)
    display_route(selected)

elif command.startswith("save "):
    parts = command.split(maxsplit=1)
    file_name = parts[1]
    save_routes(file_name, routes)

elif command.startswith("load "):
    parts = command.split(maxsplit=1)
    file_name = parts[1]
    routes = load_routes(file_name)

elif command == 'help':
    print("Список команд:\n")
    print("add - добавить маршрут;")
    print("list - вывести список маршрутов;")
    print("select <номер маршрута> - запросить данные о маршруте;")
    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()

```

Рисунки 6. Выполненное индивидуальное задание

Задание повышенной сложности.

```

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

import json
import jsonschema
import sys
import os

def get_route()::
    """
    Запросить данные о маршруте
    """
    start = input("Название начального пункта маршрута? ")
    finish = input("Название конечного пункта маршрута? ")
    number = int(input("Номер маршрута? "))

    return {
        'start': start,
        'finish': finish,
        'number': number,
    }

def display_route(routes):

```



```

"""
Отобразить список маршрутов
"""
if routes:
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 8
    )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
            "№",
            "Начальный пункт",
            "Конечный пункт",
            "Номер маршрута"
        )
    )
    print(line)

    for idx, worker in enumerate(routes, 1):
        print(
            '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                idx,
                worker.get('start', ''),
                worker.get('finish', ''),
                worker.get('number', 0)
            )
        )
        print(line)
else:
    print("Список маршрутов пуст.")

def select_route(routes, period):
    """
    Выбрать маршрут
    """
    result = []
    for employee in routes:
        if employee.get('number') == period:
            result.append(employee)

    return result

def save_routes(file_name, staff):
    """
    Сохранить всех работников в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_routes(file_name, load):
    os.chdir("C:\\Users\\Honor\\Desktop\\R\\lab_2_16\\json")
    """
    Загрузить всех работников из файла JSON.
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        file = json.load(fin)
    print("Файл загружен")
    validate(file, load)

```

```

return file

def validate(file, schema):
    validator = jsonschema.Draft7Validator(schema)
    try:
        if not validator.validate(file):
            print("Нет ошибок валидации")
    except jsonschema.exceptions.ValidationError:
        print("Ошибка валидации", file=sys.stderr)
        exit(1)

def main():
    """
    Главная функция программы
    """
    os.chdir("C:\\Users\\Honor\\Desktop\\R\\lab_2_16\\json")
    with open('check.json', 'r') as check:
        first_load = json.load(check)

    routes = []

    while True:
        command = input(">>> ").lower()
        if command == 'exit':
            break

        elif command == 'add':
            route = get_route()
            routes.append(route)
            if len(routes) > 1:
                routes.sort(key=lambda item: item.get('number', ''))

        elif command == 'list':
            display_route(routes)

        elif command.startswith('select'):
            parts = command.split(' ', maxsplit=1)
            period = int(parts[1])

            selected = select_route(routes, period)
            display_route(selected)

        elif command.startswith("save "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            save_routes(file_name, routes)

        elif command.startswith("load "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            routes = load_routes(file_name, first_load)

        elif command == 'help':
            print("Список команд:\n")
            print("add - добавить маршрут;")
            print("list - вывести список маршрутов;")
            print("select <номер маршрута> - запросить данные о маршруте;")
            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()
```

Рисунок 8. Выполненное индивидуальное задание

8. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

```
C:\Users\User\Desktop\2 курс Python\lab 19\2.16>git add .
C:\Users\User\Desktop\2 курс Python\lab 19\2.16>git commit -m "key"
"fit" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.
C:\Users\User\Desktop\2 курс Python\lab 19\2.16>git commit -m "key"
[main 1385347] key
7 files changed, 489 insertions(+), 1 deletion(-)
create mode 100644 f
create mode 100644 file_name
create mode 100644 indiv 1.py
create mode 100644 level up.py
create mode 100644 one
create mode 100644 primer.py
C:\Users\User\Desktop\2 курс Python\lab 19\2.16>git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 3.94 KiB | 1.31 MiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/aikanyshkauanbekova/2.16.git
f513f06..1385347 main -> main
C:\Users\User\Desktop\2 курс Python\lab 19\2.16>
```

Рисунок 8. Сохранения

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.

За счёт своей лаконичности по сравнению с XML формат JSON может быть более подходящим для сериализации сложных структур. Применяется в веб-приложениях как для обмена данными между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения).

2. Какие типы значений используются в JSON?

В качестве значений в JSON могут быть использованы:

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?

JSON может содержать другие вложенные объекты в JSON, в дополнение к вложенным массивам. Такие объекты и массивы будут передаваться, как значения назначенные ключам и будут представлять собой связку ключ-значение.

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

Формат обмена данными JSON5 — это расширенная JSON-версия, которая призвана смягчить некоторые ограничения JSON, расширив его синтаксис и включив в него некоторые функции из ECMAScript 5.1.

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

Чтобы использовать JSON5, нужно установить пакет json5.

json5 становится частью зависимостей в package.json.

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

```
json.dump() json.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?

```
#Для поддержки кириллицы установим ensure_ascii=False
```

```
json.dump(staff, fout, ensure_ascii=False, indent=4)
```

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema?

Что такое схема данных? Приведите схему данных для примера

1. Схема является JSON-объектом, предназначенным для описания какихлибо данных в формате JSON.

Вывод: были приобретены навыки по работе с данными формата JSON при написании программ с помощью языка программирования Python версии 3.x.