

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 №2.16**  
**дисциплины**  
**«Программирование на языке Python»**

Выполнил:  
Быковская Стефания Станиславовна  
2 курс, группа ИТС-б-о-22-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи», очная  
форма обучения

---

(подпись)

Проверил:

Воронкин Р. А., доцент кафедры  
инфокоммуникаций

---

(подпись)

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

Ставрополь, 2023 г.

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

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

### **Ход работы:**

Задание 1. Создала общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавила файл .gitignore с необходимыми правилами. Клонировала свой репозиторий на свой компьютер.

```
Cloning into 'Laba_1.16'...
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.
```

Рисунок 1. Клонирование репозитория

Задание 2. Организовала свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\Users\student-09-510\Laba_1.16>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/student-09-510/Laba_1.16/.git/hooks]
```

Рисунок 2. Модель ветвления git-flow

Задание 3. Создали виртуальное окружение conda и активировала его, также установила необходимые пакеты isort, black, flake8.



```

49         )
50         print(line)
51     else:
52         print("Список работников пуст.")
53
54
55 def select_workers(staff, period):
56     """
57     Выбрать работников с заданным стажем.
58     """
59     # Получить текущую дату.
60     today = date.today()
61     # Сформировать список работников.
62     result = []
63     for employee in staff:
64         if today.year - employee.get("year", today.year) >= period:
65             result.append(employee)
66
67     # Возвратить список выбранных работников.
68     return result
69
70
71 def save_workers(file_name, staff):
72     """
73     Сохранить всех работников в файл JSON.
74     """
75     # Открыть файл с заданным именем для записи.
76     with open(file_name, "w", encoding="utf-8") as fout:
77         # Выполнить сериализацию данных в формат JSON.
78         # Для поддержки кириллицы установим ensure_ascii=False
79         json.dump(staff, fout, ensure_ascii=False, indent=4)
80
81
82 def load_workers(file_name):
83     """
84     Загрузить всех работников из файла JSON.
85     """
86     # Открыть файл с заданным именем для чтения.
87     with open(file_name, "r", encoding="utf-8") as fin:
88         return json.load(fin)
89
90
91 def main():
92     """
93     Главная функция программы.
94     """
95     # Список работников.
96     workers = []

```

```

96     workers = []
97     # Организовать бесконечный цикл запроса команд.
98     while True:
99         # Запросить команду из терминала.
100         command = input(">>> ").lower()
101         # Выполнить действие в соответствие с командой.
102         if command == "exit":
103             break
104
105         elif command == "add":
106             # Запросить данные о работнике.
107             worker = get_worker()
108             # Добавить словарь в список.
109             workers.append(worker)
110             # Отсортировать список в случае необходимости.
111             if len(workers) > 1:
112                 workers.sort(key=lambda item: item.get("name", ""))
113
114         elif command == "list":
115             # Отобразить всех работников.
116             display_workers(workers)
117
118         elif command.startswith("select "):
119             # Разбить команду на части для выделения стажа.
120             parts = command.split(maxsplit=1)
121             # Получить требуемый стаж.
122             period = int(parts[1])
123             # Выбрать работников с заданным стажем.
124             selected = select_workers(workers, period)
125             # Отобразить выбранных работников.
126             display_workers(selected)
127
128         elif command.startswith("save "):
129             # Разбить команду на части для выделения имени файла.
130             parts = command.split(maxsplit=1)
131             # Получить имя файла.
132             file_name = parts[1]
133
134             # Сохранить данные в файл с заданным именем.
135             save_workers(file_name, workers)
136
137         elif command.startswith("load "):
138             # Разбить команду на части для выделения имени файла.
139             parts = command.split(maxsplit=1)
140             # Получить имя файла.
141             file_name = parts[1]
142
143             # Сохранить данные в файл с заданным именем.
144             workers = load_workers(file_name)
145
146         elif command == "help":
147             # Вывести справку о работе с программой.
148             print("Список команд:\n")
149             print("add - добавить работника;")
150             print("list - вывести список работников;")
151             print("select <стаж> - запросить работников <стажем>")
152             print("help - отобразить справку;")
153             print("load - загрузить данные из файла;")
154             print("save - сохранить данные в файл;")
155             print("exit - завершить работу <программой>.")
156         else:
157             print(f"Неизвестная команда {command}", file=sys.stderr)
158
159 if __name__ == "__main__":
160     main()

```

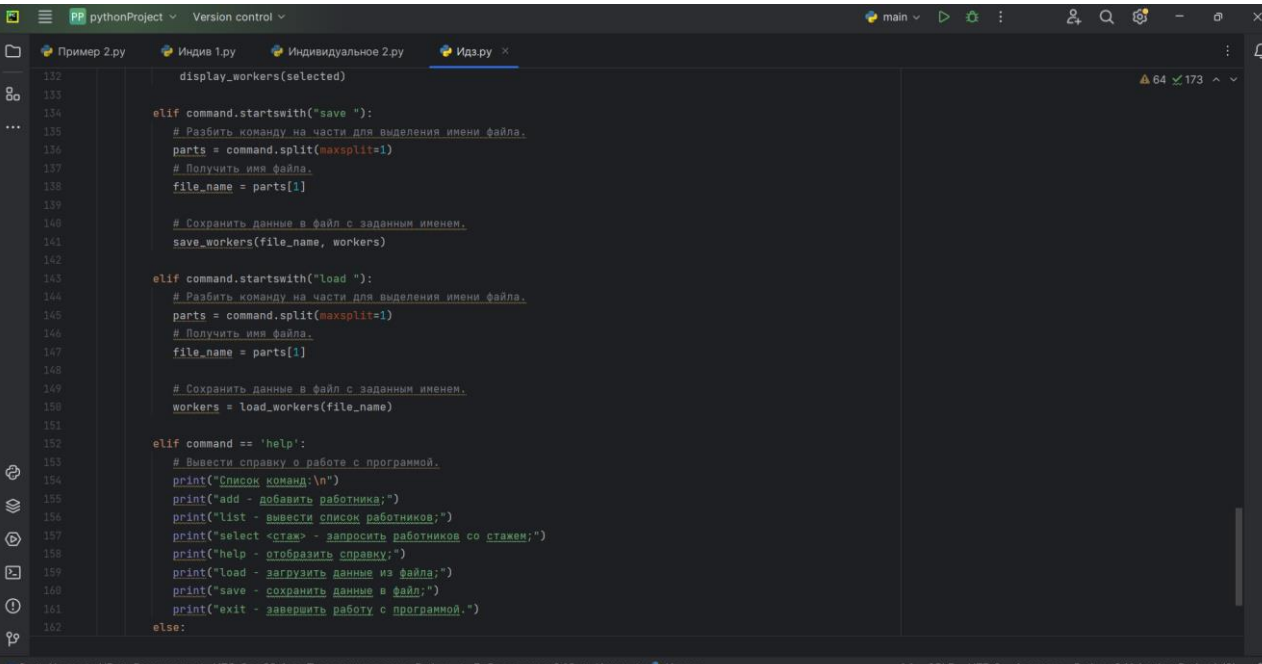
Рисунок 4-7. Пример 1

## Индивидуальное задание

### Вариант 2

Создали новый файл под названием idz.py.

Условие 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



The screenshot shows a Python IDE with a file named 'Идз.py' open. The code is a script for a worker management system. It uses a loop to read commands from the user and performs actions based on the command. Comments in Russian explain each step.

```

132     display_workers(selected)
133
134     elif command.startswith("save "):
135         # Разбить команду на части для выделения имени файла.
136         parts = command.split(maxsplit=1)
137         # Получить имя файла.
138         file_name = parts[1]
139
140         # Сохранить данные в файл с заданным именем.
141         save_workers(file_name, workers)
142
143     elif command.startswith("load "):
144         # Разбить команду на части для выделения имени файла.
145         parts = command.split(maxsplit=1)
146         # Получить имя файла.
147         file_name = parts[1]
148
149         # Сохранить данные в файл с заданным именем.
150         workers = load_workers(file_name)
151
152     elif command == 'help':
153         # Вывести справку о работе с программой.
154         print("\nСписок команд:\n")
155         print("add - добавить работника;")
156         print("list - вывести список работников;")
157         print("select <стак> - запросить работников со стакаем;")
158         print("help - отобразить справку;")
159         print("load - загрузить данные из файла;")
160         print("save - сохранить данные в файл;")
161         print("exit - завершить работу с программой.")
162     else:

```

The IDE interface includes a top bar with file explorer, search, and window management icons. The left sidebar shows a project tree with files like 'Пример 2.py', 'Индия 1.py', 'Индивидуальное 2.py', and 'Идз.py'. The bottom status bar shows the file path, encoding (CRLF), and other details.

```
routes.json  X
```

Схема: <Схема не выбрана>

```
1 [{"start": "", "end": "1", "number": ""}, {"start": "2", "end": "1", "number": "1"}, {"start": "\u0041\u0043\u0044\u0043\u0043\u0042\u00430", "end": "\u0042"}]
```

### Задание 5.

```
(2.16) PS C:\Users\student-09-320\Laba_1.16> conda env export > envirement.yml
(2.16) PS C:\Users\student-09-320\Laba_1.16> conda deactivate
```

Ссылка: <https://github.com/stefa-b/Phyton-lab-3.git>

## Ответы на контрольные вопросы:

### 1. Для чего используется JSON?

JSON — это стандарт обмена данными. Он позволяет легко сериализовать и десериализовать объекты. Стандарт часто применяют, когда разрабатывают API и веб-приложения.

### 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, то есть 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.

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

- JSON позволяет организовать сложные структуры данных, такие как списки и вложенные словари (объекты).

- В JSON можно хранить разные типы данных, включая числа, строки, логические значения, массивы и объекты.

- Для организации сложных данных в JSON используются вложенные объекты и списки, позволяя создавать структуры данных любой сложности.

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

- JSON5 - это расширение формата данных JSON, разработанное для улучшения читаемости и удобства записи JSON-данных.

- Отличие JSON5 от обычного JSON включает в себя дополнительные возможности, такие как использование комментариев, разделителей ключей и значений, а также возможность использования одиночных кавычек вместо двойных.

- JSON5 является более гибким и читаемым форматом для записи данных, но не является стандартом и не поддерживается всеми JSON-парсерами.

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

Для работы с данными в формате JSON5 на Python, вы можете использовать парсеры, поддерживающие JSON5, такие как `demjson`. Однако, JSON5 не является стандартом, поэтому поддержка может быть ограничена.

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

Для сериализации данных в формат JSON в Python можно использовать модуль `json`. Он предоставляет функции `json.dump()` и `json.dumps()`, а также класс `json.JSONEncoder`, который может быть настроен для сериализации данных в формат JSON.

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

`json.dump()` записывает данные в файл. Вы используете его, когда хотите сохранить данные в файле.

`json.dumps()` превращает данные в строку. Вы используете его, когда хотите получить данные в виде строки для дальнейшей обработки, но не сохранять их в файле.

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

Для десериализации данных из формата JSON в Python используется модуль `json`, предоставляющий функции `json.load()` и `json.loads()`.

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



Для работы с данными JSON, содержащими кириллицу, важно убедиться, что данные правильно кодируются и декодируются. Обычно это не вызывает проблем, поскольку JSON поддерживает Unicode, включая кириллические символы. Однако, при чтении и записи JSON-файлов, убедитесь, что правильно установлена кодировка (например, utf-8) для текстовых данных.

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema?  
Что такое схема данных?

JSON Schema - это спецификация, которая описывает формат данных JSON и правила их валидации. С помощью JSON Schema можно определить структуру, типы данных и ограничения для JSON-данных. JSON Schema используется для проверки соответствия данных определенным правилам. Это полезно, например, при валидации данных, получаемых из внешних источников. JSON Schema не является частью стандартной библиотеки Python, но существуют библиотеки и инструменты, поддерживающие JSON Schema, которые могут использоваться в Python.

**Вывод:** приобрела навыки по работе с данными формата JSON с помощью языка программирования Python версии 3.x.