

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 №4**  
**дисциплины «Основы программной инженерии»**

Выполнил:  
Магомедов Имран Борисович  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., кандидат технических  
наук, доцент кафедры  
инфокоммуникаций

---

(подпись)

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

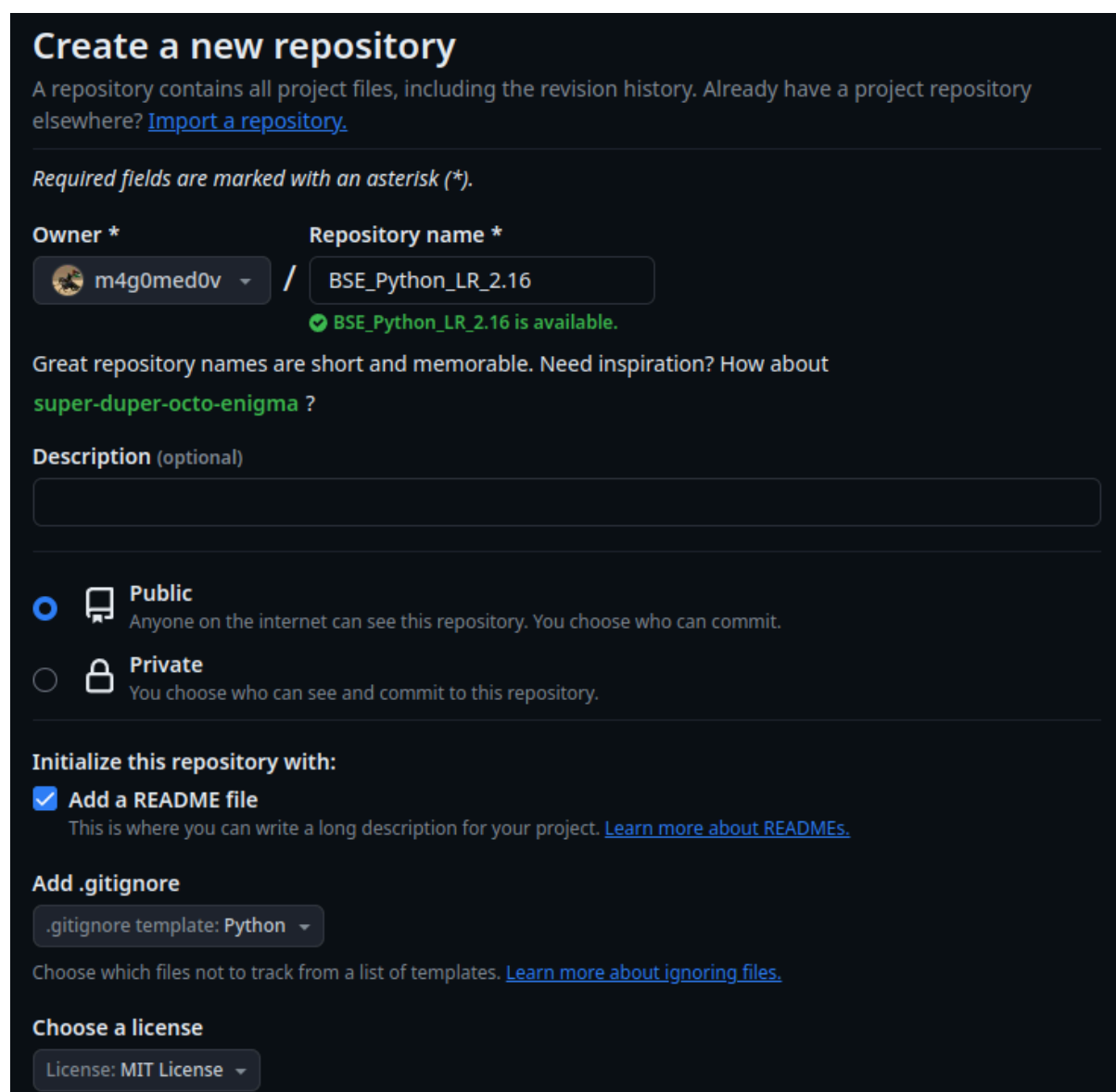
Ставрополь, 2023 г.

**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.

### 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 elsewhere? [Import a repository.](#)


*Required fields are marked with an asterisk (\*).*


**Owner \***  m4g0med0v ▾ / **Repository name \***

✔ BSE\_Python\_LR\_2.16 is available.

Great repository names are short and memorable. Need inspiration? How about **super-duper-octo-enigma** ?

**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:**

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

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

**Choose a license**

3. Выполните клонирование созданного репозитория.

```

~/work/ncfu/bse git clone git@github.com:m4g0med0v/BSE_Python_LR_2.16.git
Клонирование в «BSE_Python_LR_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
Получение объектов: 100% (5/5), готово.

```

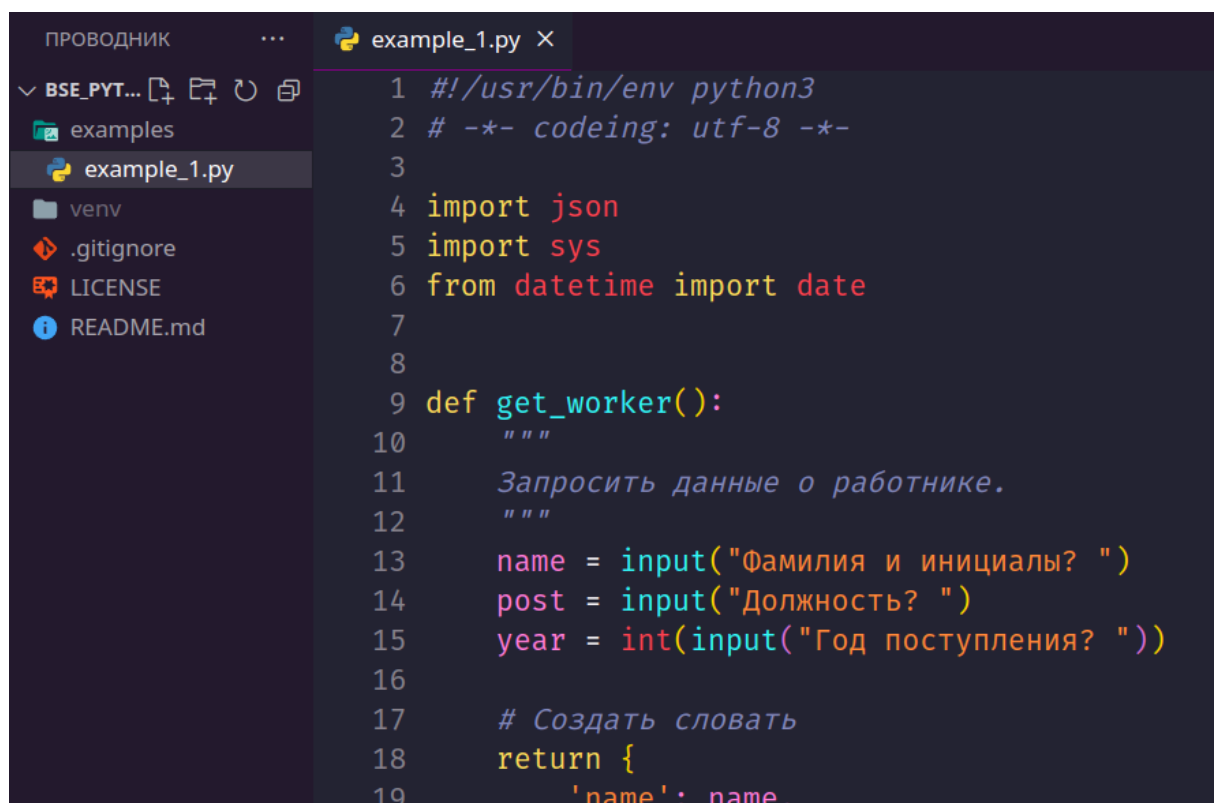
4. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```

~/w/n/b/BSE_Python_LR_2.16 main git checkout -b develop ✓
Переключились на новую ветку «develop»
~/w/n/b/BSE_Python_LR_2.16 develop ✓

```

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



```

PROBODNIK ... example_1.py X
BSE_PYT... [F] [C] [U] [O]
examples
example_1.py
venv
.gitignore
LICENSE
README.md

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import json
5 import sys
6 from datetime import date
7
8
9 def get_worker():
10     """
11     Запросить данные о работнике.
12     """
13     name = input("Фамилия и инициалы? ")
14     post = input("Должность? ")
15     year = int(input("Год поступления? "))
16
17     # Создать словарь
18     return {
19         'name': name,

```

6. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.

```
~/w/n/b/BSE_Python_LR_2.16 > git P develop ?1 > python examples/example_1.py
>>> add
Фамилия и инициалы? Магомедов И.Б.
Должность? Junior
Год поступления? 2022
>>> save ls.txt
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О | Должность | Год |
+-----+-----+-----+-----+
| 1 | Магомедов И.Б. | Junior | 2022 |
+-----+-----+-----+-----+
>>>
```

7. Зафиксируйте сделанные изменения в репозитории.

```
~/w/n/b/BSE_Python_LR_2.16 > git P develop +2 > git commit -m 'added examples
[develop e8d5f84] added examples
2 files changed, 182 insertions(+)
create mode 100644 examples/example_1.py
create mode 100644 ls.txt
```

8. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.

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.

Использовать словарь, содержащий следующие ключи: фамилия, имя; номер телефона; дата рождения (список из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены по алфавиту; вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение.

```

~/w/n/b/BSE_Python_LR_2.16 develop ?1 python individual_task.py
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <месяц> - запросить работников у которых деньрождения совпадает с указанным месяцем;
help - отобразить справку;
load - загрузить данные из файла;
save - сохранить данные в файл;
exit - завершить работу с программой.
>>> add
Фамилия: Магомедов
Имя: Имран
Номер телефона: 89289795120
Дата рождения (пример: 05 07 2004): 05 07 2004
>>> list
+-----+-----+-----+-----+
| № | Фамилия | Имя | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
| 1 | Магомедов | Имран | 89289795120 | 05 07 2004 |
+-----+-----+-----+-----+
>>> save worker.json
>>> exit

```

```

individual_task.py  workers.json X
1  [
2      {
3          "surname": "Магомедов",
4          "name": "Имран",
5          "phone": "89289795120",
6          "year": "05 07 2004"
7      }
8  ]

```

9. Зафиксируйте сделанные изменения в репозитории.

```

~/w/n/b/BSE_Python_LR_2.16 develop +1 ?1 git commit -m 'added individual_task.py'
[develop eff07d9] added individual_task.py
1 file changed, 182 insertions(+)
create mode 100644 individual_task.py

```

10. Приведите в отчете скриншоты работы программ решения задания повышенной сложности.

Очевидно, что программа в примере 1 и в индивидуальном задании никак не проверяет правильность загружаемых данных формата JSON. Вследствие 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 `json-schema`, который не является частью стандартной библиотеки Python. Таким образом, необходимо реализовать валидацию загруженных данных с помощью спецификации JSON Schema.

Json схема для проверки валидации.

A screenshot of a code editor with two tabs: 'individual\_task.py 2' and 'worker-schema.json X'. The 'worker-schema.json' tab is active, displaying a JSON Schema. The schema is a JSON object with a '\$schema' property pointing to 'http://json-schema.org/draft-07/schema#', a 'title' property with the value 'Worker', and a 'properties' object. The 'properties' object contains four sub-objects: 'surname' (type: string), 'name' (type: string), 'phone' (type: string), and 'year' (type: string). The 'required' array lists 'surname', 'name', 'phone', and 'year'. The 'additionalProperties' property is set to false. Line numbers 1 through 21 are visible on the left side of the code.

```
1 {
2   "$schema": "http://json-schema.org/draft-07/schema#",
3   "title": "Worker",
4   "properties": {
5     "surname": {
6       "type": "string"
7     },
8     "name": {
9       "type": "string"
10    },
11    "phone": {
12      "type": "string"
13    },
14    "year": {
15      "type": "string"
16    }
17  },
18  "required": ["surname", "name", "phone", "year"],
19  "additionalProperties": false
20 }
21 }
```

Код который проверяет валидацию

```
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 f:
        document = json.load(f)

    if all(list(map(lambda x: check_validation_json(x), document))):
        return document
    else:
        False

def check_validation_json(file_name):
    with open('worker-schema.json') as f:
        schema = json.load(f)

    try:
        validate(instance=file_name, schema=schema)
        return True
    except ValidationError:
        return False
```

```
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.
    if load_workers(file_name):
        workers = load_workers(file_name)
    else:
        print("Error Validation JSON")|
```

Загрузка правильного файла.

```
~/w/n/b/BSE_Python_LR_2.16 develop !2 ?2 python individual_task.py
>>> list
Список работников пуст.
>>> load list.json
>>> list
```

№	Фамилия	Имя	Номер телефона	Дата рождения
1	Темаев	Идрис	89284563434	06 23 2006
2	Магомедов	Имран	89289795120	05 07 2004

Загрузка неправильного файла.

```
~/w/n/b/BSE_Python_LR_2.16 git P develop !2 ?2 python individual_task.py
>>> load list.json
Error Validation JSON
```

11. Зафиксируйте изменения.

```
~/w/n/b/BSE_Python_LR_2.16 git P develop +3 git commit -m 'adjustment'
[develop 59572f3] adjustment
3 files changed, 47 insertions(+), 11 deletions(-)
delete mode 100644 ls.txt
create mode 100644 worker-schema.json
```

12. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

13. Выполните слияние ветки для разработки с веткой master/main.

```
~/w/n/b/BSE_Python_LR_2.16 git checkout main
ПереклЮчилиcь на ветку «main»
Эта ветка соответствует «origin/main».
~/w/n/b/BSE_Python_LR_2.16 git merge develop
Обновление 7e86ea5..59572f3
Fast-forward
 examples/example_1.py | 175 +++++
 individual_task.py    | 204 +++++
 worker-schema.json    | 21  +
 3 files changed, 400 insertions(+)
 create mode 100644 examples/example_1.py
 create mode 100644 individual_task.py
 create mode 100644 worker-schema.json
```

14. Отправьте сделанные изменения на сервер GitHub.

15. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

### Вопросы для защиты работы

1. Для чего используется JSON?

JSON используется для:

- Обмена данными между веб-приложениями и серверами.
- Хранения данных в NoSQL-базах данных.
- Сериализации объектов в JavaScript.
- Конфигурации приложений.
- И т.д.

2. Какие типы значений используются в JSON?

JSON поддерживает следующие типы значений:



- Строки: Текстовые данные, заключенные в кавычки.
- Числа: Целые числа (например, 123) и числа с плавающей запятой (например, 3.14).
- Булевы значения: true или false.
- Массивы: Упорядоченные коллекции значений.
- Объекты: Неупорядоченные коллекции пар ключ-значение.

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

Сложные данные в JSON могут быть представлены в виде объектов или массивов.

- Объекты: Используются для представления данных, которые имеют естественную структуру ключ-значение. Например, информация о пользователе может быть представлена в виде объекта с ключами name, email и age.
- Массивы: Используются для представления данных, которые не имеют естественной структуры ключ-значение. Например, список покупок может быть представлен в виде массива строк.

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

JSON5 - это расширение формата JSON, которое добавляет поддержку следующих функций:

- Комментарии: Однострочные и многострочные комментарии.
- Неограниченные числа: Числа, которые не ограничены диапазоном значений JSON.
- Объекты с произвольными ключами: Ключи объектов могут быть не только строками, но и числами, булевыми значениями или null.

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

Для работы с JSON5 в Python можно использовать библиотеку `jsonschema`

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

В Python для сериализации есть функции `json.dump()` и `json.dumps()`.

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

`json.dump()`: Сериализует данные в JSON и записывает их в файл.

`json.dumps()`: Сериализует данные в JSON и возвращает их в виде строки.

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

Для десериализации данных из JSON в Python можно использовать следующие функции:

- `json.load()`: Десериализует данные JSON из файла и возвращает их в виде объекта Python.
- `json.loads()`: Десериализует данные JSON из строки и возвращает их в виде объекта Python.

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

При загрузке файла в python с помощью оператора `with`, в функции `open()` нужно прописать `encoding='utf-8'`.

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

```
{
  "$schema":
    "http://json-schema.org/draft-07/schema#",
  "title": "Worker", "properties": {
    "name": {
      "type": "string"
    },
    "post": {
      "type": "string"
    },
    "year":
      {
        "type": "int"
      }
  },
  "required": ["surname", "name", "phone",
    "year"],
  "additionalProperties": false
}
```