

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

дисциплина Языки программирования

Отчет по лабораторной работе №2.17

Работа с данными формата JSON в языке Python

Выполнил: студент группы ИТС-б-о-21-1

Пушкин Максим Алексеевич

(подпись)

Проверил: кандидат технических наук, доцент кафедры
инфокоммуникаций,

Роман Александрович Воронкин

(подпись)

Ставрополь, 2022

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

Метод и порядок выполнения

Код задания 1

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import argparse
import json
import os.path
import sys
from datetime import date

def add_worker(staff, name, post, year):
    """
    Добавить данные о работнике.
    """
    staff.append(
        {
            "name": name,
            "post": post,
            "year": year
        }
    )
    return staff

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(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )

    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("workers")

```

```

parser.add_argument(
    "--version",
    action="version",
    version="% (prog)s 0.1.0"
)
subparsers = parser.add_subparsers(dest="command")

# Создать субпарсер для добавления работника.
add = subparsers.add_parser(
    "add",
    parents=[file_parser],
    help="Add a new worker"
)
add.add_argument(
    "-n",
    "--name",
    action="store",
    required=True,
    help="The worker's name"
)
add.add_argument(
    "-p",
    "--post",
    action="store",
    help="The worker's post"
)
add.add_argument(
    "-y",
    "--year",
    action="store",
    type=int,
    required=True,
    help="The year of hiring"
)
# Создать субпарсер для отображения всех работников.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all workers"
)
# Создать субпарсер для выбора работников.
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the workers"
)
select.add_argument(
    "-p",
    "--period",
    action="store",
    type=int,
    required=True,
    help="The required period"
)
# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)

# Загрузить всех работников из файла, если файл существует.
is_dirty = False

if os.path.exists(args.filename):

```

```

        workers = load_workers(args.filename)
    else:
        workers = []

    # Добавить работника.
    if args.command == "add":
        workers = add_worker(
            workers,
            args.name,
            args.post,
            args.year
        )
        is_dirty = True

    # Отобразить всех работников.
    elif args.command == "display":
        display_workers(workers)

    # Выбрать требуемых работников.
    elif args.command == "select":
        selected = select_workers(workers, args.period)
        display_workers(selected)

    # Сохранить данные в файл, если список работников был изменен.
    if is_dirty:
        save_workers(args.filename, workers)

if __name__ == '__main__':
    main()

```

результат программы

```

PS E:\ЯП 2 курс\7\project> python prim.py display data2.json

```

No	Ф.И.О.	Должность	Год
1	Иванов Иван	Директор	2007
2	Петров Петр	Бухгалтер	2010
3	Сидоров Сидор	Главный инженер	2012

Рис 1.

Пример 2

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import argparse
import json
import os.path
import sys

```

```

def add_shop(list_race, name, number, time):
    """
    Добавить данные магазина.
    """
    list_race.append(
        {
            "name": name,
            "number": number,
            "time": time
        }
    )
    return list_race

def display_shop(list_race):
    """
    Отобразить список.
    """
    if list_race:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 6,
            '-' * 20,
            '-' * 30,
            '-' * 20
        )
        print(line)
        print(
            '| {:^6} | {:^20} | {:^30} | {:^20} |'.format(
                "No",
                "пункт назначения",
                "номер",
                "время"
            )
        )
        print(line)
        for idx, listrace in enumerate(list_race, 1):
            print(
                '| {:>6} | {:<20} | {:<30} | {:>20} |'.format(
                    idx,
                    listrace.get('name', ''),
                    listrace.get('number', ''),
                    listrace.get('time', 0)
                )
            )
            print(line)
    else:
        print("Список рейсов пуст.")

def select_product(list_race, race_sear):
    """
    Выбрать.
    """
    search_race = []
    for race_sear_itme in list_race:
        if race_sear == race_sear_itme['name']:
            search_race.append(race_sear_itme)
    return search_race

```

```

def save_race(file_name, list_race):
    """
    Сохранить все в JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(list_race, fout, ensure_ascii=False, indent=4)

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

def main(command_line=None):
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )
    parser = argparse.ArgumentParser("races")
    parser.add_argument(
        "--version",
        action="version",
        version="% (prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")

    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new race"
    )
    add.add_argument(
        "-nm",
        "--name",
        action="store",
        required=True,
        help="The race's name"
    )
    add.add_argument(
        "-nb",
        "--number",
        action="store",
        help="The number"
    )
    add.add_argument(
        "-t",
        "--time",
        action="store",
        type=int,
        required=True,
        help="time"
    )
    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all races"
    )

```

```

)
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the product"
)
select.add_argument(
    "-ss",
    "--name_sear",
    action="store",
    type=str,
    required=True,
    help="The name race"
)
args = parser.parse_args(command_line)
is_dirty = False
if os.path.exists(args.filename):
    race = load_list_race(args.filename)
else:
    race = []
if args.command == "add":
    race = add_shop(
        race,
        args.name,
        args.number,
        args.time
    )
    is_dirty = True
elif args.command == "display":
    display_shop(race)

elif args.command == "select":
    selected = select_product(race, args.race_sear)
    display_shop(selected)
if is_dirty:
    save_race(args.filename, race)

if __name__ == '__main__':
    main()

```

PS E:\ЯП 2 курс\7\project> python ind-1.py display data.json

No	пункт назначения	номер	время
1	Ставрополь	15	13:25

Рис. 2

Контрольные вопросы:

1. Для чего используется JSON?

JSON представляет собой хорошую альтернативу XML и требует куда

меньше форматирования контента. Это информативное руководство

поможет вам быстрее разобраться с данными, которые вы можете

использовать с JSON и основной структурой с синтаксисом этого же формата.

2. Какие типы значений используются в JSON?

Запись, массив, число, литералы, строка

3. Как организована работа со сложными данными в JSON?

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

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

JSON5 расширяет формат обмена данными JSON, чтобы сделать его

немного более удобным в качестве языка конфигурации:

- Комментарии в стиле JavaScript (как однострочные, так и многострочные) являются законными.
- Ключи объектов могут быть без кавычек, если они

являются законными идентификаторами ECMAScript

- Объекты и массивы могут заканчиваться запятыми.
- Строки могут заключаться в одинарные кавычки, и допускаются многострочные строковые литералы.

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

Модуль `json` предоставляет удобный метод `dump()` для записи данных в

файл. Существует также метод `dumps()` для записи данных в обычную строку.

Типы данных Python кодируются в формат JSON в соответствии с интуитивно

понятными правилами преобразования

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

`dump` отличается от `dumps` тем, что `dump` записывает объект Python в

файл JSON, а `dumps` сериализует объект Python и хранит его в виде

строки.

8. Какие средства предоставляет язык Python для десериализации

данных из формата JSON?

В модуле `json` определены методы `load()` и `loads()`, предназначенные для

преобразования кодированных в формате JSON данных в объекты Python.

Подобно операции сериализации, также существует таблица преобразования типов, определяющая правила для обратного декодирования

данных.

9. Какие средства необходимо использовать для работы с данными

формата JSON, содержащими кириллицу?

Параметр `ensure_ascii`

Вывод: в ходе лабораторной работы приобретены навыки по работе с данными формата JSON с помощью языка программирования Python версии 3.x.