НИУ ИТМО

Факультет программной инженерии и компьютерной техники

Отчет по Лабораторной работе №4    
по курсу “Информатика”

“Исследование протоколов, форматов обмена информацией и языков разметки документов”   
Вариант №21

Выполнил:

Студент группы P3118

Шульга Артём Игоревич

Преподаватель:

Малышева Татьяна Алексеевна

Санкт-Петербург

2021

Оглавление

[Задание для Варианта 21 2](#_Toc87312240)

[Файл формата JSON 2](#_Toc87312241)

[Исходный код 3](#_Toc87312242)

[Результат выполнения программы 6](#_Toc87312243)

[Дополнительное задание №1 8](#_Toc87312244)

[Исходный код 8](#_Toc87312245)

[Сравнение форматов 9](#_Toc87312246)

[Дополнительное задание №2 9](#_Toc87312247)

[Результат выполнения программы 10](#_Toc87312248)

[Сравнение результатов работы программ 11](#_Toc87312249)

[Дополнительное задание №3 11](#_Toc87312250)

[Сравнение программ по времени 12](#_Toc87312251)

[Вывод 12](#_Toc87312252)

[Список литературы 12](#_Toc87312253)

# Задание для Варианта 21

Преобразовать файл формата JSON в файл формата YAML.

# Файл формата JSON

{

    "day":"Четверг",

    "weeks": [3, 5, 7, 9, 11, 13, 15, 17],

    "lessons":[

          {

            "time": "10:00-11:30",

            "room":{

                "room":"365 ауд.",

                "address":"Кронверкский пр., д.49, лит.А"

            },

            "lesson":{

                "subject":"Информатика(Лаб)",

                "week":"Нечётная неделя",

                "teacher":"Малышева Татьяна Алексеевна"

            },

            "format": "очно-дистанционный"

        },

        {

            "time": "11:40-13:10",

            "room":{

                "room":"365 ауд.",

                "address":"Кронверкский пр., д.49, лит.А"

            },

            "lesson":{

                "subject":"Информатика(Лаб)",

                "week":"Нечётная неделя",

                "teacher":"Малышева Татьяна Алексеевна"

            },

            "format": "очно-дистанционный"

        },

        {

            "time": "13:30-15:00",

            "room":{

                "room":"285 ауд.",

                "address":"Кронверкский пр., д.49, лит.А"

            },

            "lesson":{

                "subject":"Дискретная математика",

                "week":"Нечётная неделя",

                "teacher":"Поляков Владимир Иванович"

            },

            "format": "очно-дистанционный"

        }

    ]

}

# Исходный код

# Функции для считывания с json

# принимает итератор на строчку, которая начинается на "

# возвращает строку без "", итератор указывает на кавычку

def read\_string(*it*) -> str:

    c = next(it)

    value = ""

    while c != '\"':

        value += c

        c = next(it)

    return value

# принимает итератор, который стоит на [

# возвращает список, итератор стоит на ]

def read\_list(*it*) -> list:

    result\_list = []

    c = next(it)

    while True:

        if c == ' ' or c == ',':

            c = next(it)

            continue

        if c == ']':

            break

        if c == '[':

            result\_list.append(read\_list(it))

        elif c == '{':

            result\_list.append(read\_dict(it))

        elif c == '\"':

            result\_list.append(read\_string(it))

        elif c.isdigit() or c == '.':

            res = read\_number(it,c)

            result\_list.append(res[0])

            c = res[1]

            continue

        c = next(it)

    return result\_list

# принимает итератор, который стоит на {

# возвращает словарь, итератор стоит на }

def read\_dict(*it*) -> dict:

    result\_dict = {}

    c = next(it)

    while True:

        if c == ' ':

            c = next(it)

            continue

        if c == '}':

            break

        if c == '\"':

            key = read\_string(it)

            c = next(it)

            while c == ' ' or c == ':':

                c = next(it)

            if c.isdigit() or c == '.':

                result = read\_number(it,c)

                value = result[0]

                c = result[1]

            elif c == '\"':

                value = read\_string(it)

                c = next(it)

            elif c == '[':

                value = read\_list(it)

            elif c == '{':

                value = read\_dict(it)

            else:

                continue

            result\_dict[key] = value

            continue

        c = next(it)

    return result\_dict

# принимает итератор и его текущую позицию (цифру или точку)

# возвращает список(число,следующий символ), итератор стоит на следующем символе!!!

def read\_number(*it*,*digit*) -> list:

    value = digit

    c = next(it)

    while c.isdigit() or c == '.':

        value+=c

        c = next(it)

    if '.' in value:

        if value.count('.') == 1:

            value = float(value)

        else:

            raise ValueError("Too many points in Number")

    else:

        value = int(value)

    return [value,c]

# функции для преобразования в yaml

# создание count пробелов

def create\_spaces(*count*) -> str:

    return ' '\*count

# функция вывода перевода в yaml

def mprint(*string*):

    file.write(string+'\n')

# вывод словаря

def print\_dict(*jdict*,*pr\_str*,*isList*):

    for i in jdict.keys():

        if not isinstance(jdict.get(i),dict) and not isinstance(jdict.get(i),list):

            mprint(pr\_str[0]+i+": "+str(jdict.get(i)))

        elif isinstance(jdict.get(i),list):

            mprint(pr\_str[0]+i+":")

            print\_list(jdict.get(i),pr\_str)

        else:

            mprint(pr\_str[0]+i+":")

            pr\_str[1] += 2

            pr\_str[0] += create\_spaces(2)

            print\_dict(jdict.get(i),pr\_str,False)

            pr\_str[1]-=2

            pr\_str[0]=create\_spaces(pr\_str[1])

        if isList:

            isList = False

            pr\_str[0] = create\_spaces(pr\_str[1])

# вывод списка

def print\_list(*jlist*,*pr\_str*):

    for i in jlist:

        if not isinstance(i,dict) and not isinstance(i,list):

            mprint(pr\_str[0]+"- "+str(i))

        elif isinstance(i,dict):

            pr\_str[1]+=2

            pr\_str[0]+="- "

            print\_dict(i,pr\_str,True)

            pr\_str[1]-=2

            pr\_str[0] = create\_spaces(pr\_str[1])

        else:

            pr\_str[1]+=2

            pr\_str[0]+="- "

            print\_list(i,pr\_str)

            pr\_str[1]-=2

            pr\_str[0]=create\_spaces(pr\_str[1])

file = open("schedule1.yml","w",*encoding*='UTF-8')

with open("schedule.json","rt",*encoding*="UTF-8") as f:

s = f.read()

p = iter(s)

try:

while True:

c = next(p)

if c == ' ':

continue

      if c == '[':

json = read\_list(p)

          break

      elif c == '{':

          json = read\_dict(p)

          break

except StopIteration:

pass

print\_dict(json,["",0],False)

file.close()

# Результат выполнения программы

Файл YAML

day: Четверг

weeks:

- 3

- 5

- 7

- 9

- 11

- 13

- 15

- 17

lessons:

- time: 10:00-11:30

  room:

    room: 365 ауд.

    address: Кронверкский пр., д.49, лит.А

  lesson:

    subject: Информатика(Лаб)

    week: Нечётная неделя

    teacher: Малышева Татьяна Алексеевна

  format: очно-дистанционный

- time: 11:40-13:10

  room:

    room: 365 ауд.

    address: Кронверкский пр., д.49, лит.А

  lesson:

    subject: Информатика(Лаб)

    week: Нечётная неделя

    teacher: Малышева Татьяна Алексеевна

  format: очно-дистанционный

- time: 13:30-15:00

  room:

    room: 285 ауд.

    address: Кронверкский пр., д.49, лит.А

  lesson:

    subject: Дискретная математика

    week: Нечётная неделя

    teacher: Поляков Владимир Иванович

  format: очно-дистанционный

# Дополнительное задание №1

# Исходный код

import yaml,json

for i in range(10):

    with open("schedule.json","rt",*encoding*="UTF-8") as f:

        data = json.load(f)

    with open("schedule2.yml","w",*encoding*="UTF-8") as f:

        yaml.dump(data,f,*allow\_unicode*=True)

**Результат выполнения программы**

Файл YAML

day: Четверг

lessons:

- format: очно-дистанционный

  lesson:

    subject: Информатика(Лаб)

    teacher: Малышева Татьяна Алексеевна

    week: Нечётная неделя

  room:

    address: Кронверкский пр., д.49, лит.А

    room: 365 ауд.

  time: 10:00-11:30

- format: очно-дистанционный

  lesson:

    subject: Информатика(Лаб)

    teacher: Малышева Татьяна Алексеевна

    week: Нечётная неделя

  room:

    address: Кронверкский пр., д.49, лит.А

    room: 365 ауд.

  time: 11:40-13:10

- format: очно-дистанционный

  lesson:

    subject: Дискретная математика

    teacher: Поляков Владимир Иванович

    week: Нечётная неделя

  room:

    address: Кронверкский пр., д.49, лит.А

    room: 285 ауд.

  time: 13:30-15:00

weeks:

- 3

- 5

- 7

- 9

- 11

- 13

- 15

- 17

# Сравнение форматов

Сравнивая результаты работы программ, можно заметить, что преобразования в формат YAML произошёл одинаково, отличается только порядок ключей: в написанною мною программой ключи идут по порядку, который был в исходном файле, а в программе с библиотеками ключи идут по алфавитному порядку. Структуры преобразованы одинаково, потери информации в обоих программах не произошло. Ключи располагаются не по алфавитному порядку, так как происходит считывание структуры JSON в формат объекта Python по порядку расположения данных в исходном файле JSON, сортировка ключей не производится.

# Дополнительное задание №2

**Исходный код**

import re

def create\_spaces(*count*) -> str:

    return ' '\*count

def process\_object(*s*) -> str:

    new\_s = ""

    counter = -2 # отступ для глобального объекта

    skip = False

    for i in range(len(s)-1):

        if skip:

            skip = False

            continue

        # если не символ объекта, то записываем в новую строчку

        if s[i] != '{' and s[i] != '}' and s[i] != '\n' and s[i] != ']':

            new\_s+=s[i]

        # создаем отступы при каждой новой строчке

        elif s[i] == '\n':

            new\_s+='\n'+create\_spaces(counter)

        # удаляем { добавляем отступ

        elif s[i] == '{':

            counter+=2

        # убираем отступ

        elif s[i] == '}':

            counter-=2

        # убираем запятые после списков

        elif s[i] == ']':

            new\_s+=s[i]

            if s[i+1] == ',':

                skip = True

    return new\_s

file = open("schedule3.yml","w",*encoding*='UTF-8')

with open("schedule.json","rt",*encoding*="UTF-8") as f:

    s = f.read()

s = re.sub(r"\"([\w\d]\*)\"\s\*:\s\*\"([\w\d:\-\s,.\(\)]\*)\",\*",r"\1: \2",s) # пробелы

s = re.sub(r"\s\*([^\n]\*\n)",r"\1",s) # ключи

s = re.sub(r"\"([^\"]\*)\"(:[\{\[])",r"\1\2",s) # ключи объектов и списков

s = re.sub(r"\"([^\"]\*)\":",r"\1:",s) # простые списки

s = process\_object(s) # обработка отступов у объектов

s = re.sub(r"\n\s\*\n",r"\n",s) # удаляем пустые строки

s = re.sub(r"^\n",r"",s) # удаляем пустую строку в начале

s = re.sub(r"\n\s{2},\n\s\*([^\n]\*\n)",r"\n- \1",s) # создание списка

s = re.sub(r"\[\n\s{2}([^\n]\*\n)",r"\n- \1",s) # создание начала списка

s = re.sub(r"[^\w\d]]",r"",s) # удаление конца списка

s = re.sub(r"\n\s+,",r"",s) # удаление лишних запятых

file.write(s)

file.close()

# Результат выполнения программы

Файл YAML

day: Четверг

weeks: [3, 5, 7, 9, 11, 13, 15, 17]

lessons:

- time: 10:00-11:30

  room:

    room: 365 ауд.

    address: Кронверкский пр., д.49, лит.А

  lesson:

    subject: Информатика(Лаб)

    week: Нечётная неделя

    teacher: Малышева Татьяна Алексеевна

  format: очно-дистанционный

- time: 11:40-13:10

  room:

    room: 365 ауд.

    address: Кронверкский пр., д.49, лит.А

  lesson:

    subject: Информатика(Лаб)

    week: Нечётная неделя

    teacher: Малышева Татьяна Алексеевна

  format: очно-дистанционный

- time: 13:30-15:00

  room:

    room: 285 ауд.

    address: Кронверкский пр., д.49, лит.А

  lesson:

    subject: Дискретная математика

    week: Нечётная неделя

    teacher: Поляков Владимир Иванович

  format: очно-дистанционный

# Сравнение результатов работы программ

По структуре преобразованного файла все 3 программы схожи, но в программе с использованием регулярных выражений ключи расположены в порядке, который был в исходном файле, а один из списков представлен в виде другой конструкции, но которая тоже допускается в YAML. Ключи располагаются в таком же порядке, как и в исходном, из-за непосредственного преобразования файла JSON в виде строки. Список представлен в виде иной конструкции из-за простоты реализации и отсутствии особых различий у этих структур в JSON и YAML.

# Дополнительное задание №3

|  |  |  |
| --- | --- | --- |
| Задание | Применённая технология | Десятикратное время выполнения |
| Обязательное задание | - | 0.07740163803100586 |
| Дополнительное задание №1 | Библиотеки для преобразования форматов | 0.10592103004455566 |
| Дополнительное задание №2 | Регулярные выражения | 0.06963443756103516 |

# Сравнение программ по времени

По результатам измерения времени можно заметить, что программа, использующая регулярные выражения, имеет наименьшее время выполнения. Это можно объяснить тем, что сама программа является преобразованием файла как строки, “отрезая” лишнее. При помощи регулярных выражений это происходит очень быстро, поэтому эта программа является самой быстрой.

На втором месте находится программа, которая не использует никаких библиотек. Такую скорость можно объяснить тем, что данная программа не совершает преобразование значений типа bool.

На последнем месте находится программа с использованием библиотек. Из-за загрузки всех функций и классов из библиотек данная программа и проигрывает по времени, но она может совершать преобразование совершенно любых ключей.

# Вывод

Во время выполнения данной лабораторной работы я научился обрабатывать структурированные данные и преобразовывать их в объекты Python, так и в объекты других структур. Также я научился применять совершенно разные подходы к преобразованию структур данных, что пригодится мне в дальнейшем.

# Список литературы

# “Пишем изящный парсер на Питоне”. – Текст электронный // https://habr.com – URL https://habr.com/ru/post/309242/

# “Работа с файлами в формате YAML”. – Текст электронный // https://pyneng.readthedocs.io– URL https://pyneng.readthedocs.io/ru/latest/book/17\_serialization/yaml.html