Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

**Лабораторная работа №4**

“Исследование протоколов, форматов обмена информацией и языков разметки документов”

Вариант №13

Выполнила:

Алвари Юсеф

Группа:

P3130

Преподаватель:

Авксентьева Е.Ю., доцент факультета

г. Санкт-Петербург

2023

**СОДЕРЖАНИЕ**

[Задание 3](#_Toc154397313)

[Основные этапы вычисления 5](#_Toc154397314)

[Дополнительное задание №1 8](#_Toc154397315)

[Дополнительное задание №2 10](#_Toc154397316)

[Дополнительное задание №3 15](#_Toc154397317)

[Дополнительное задание №4 21](#_Toc154397318)

[Заключение 24](#_Toc154397319)

[Список использованной литературы 25](#_Toc154397320)

# Задание

Составить файл с расписанием занятий своей учебной группы в указанный день в исходном формате, написать программу на языке Python 3.x, которая бы осуществляла парсинг и конвертацию исходного файла в новый.

День недели: среда

Исходный формат: JSON

Результирующий формат: XML

Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML-файлов.

Дополнительное задание №1:

1. Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.

2. Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать.

3. Сравнить полученные результаты и объяснить их сходство/различие.

Дополнительное задание №2:

1. Переписать исходный код, добавив в него использование регулярных выражений.

2. Сравнить полученные результаты и объяснить их сходство/различие.

Дополнительное задание №3:

1. Используя свою исходную программу из обязательного задания, программу из дополнительного задания №1 и программу из дополнительного задания №2, сравнить стократное время выполнения парсинга + конвертации в цикле.

2. Проанализировать полученные результаты и объяснить их сходство/различие.

Дополнительное задание №4:

1. Переписать исходную программу, чтобы она осуществляла парсинг и конвертацию исходного файла в любой другой формат (кроме JSON, YAML, XML, HTML): PROTOBUF, TSV, CSV, WML и т.п.

2. Проанализировать полученные результаты, объяснить особенности использования формата.

# Основные этапы вычисления

Сначала был создан JSON-файл, содержащий расписание предметов на среду:

prog\_main/sch.json

{"Wednesday":[{"period":{"time":"08:20-09:50","week":"3, 5, 9, 11, 13, 15, 17"},"place":{"room":"1216/0 ауд.","building":"Ломоносова пр., д.9, лит.М"},"class-info":{"lesson":"информатика","teacher":"Балакшин Павел Валерьевич"},"lesson-format":"Очно - дистанционный"}]}

Конвертация данных из формата JSON в формат XML будет выполнена с помощью следующих шагов:

1. Чтение JSON-файла и сохранение его в виде «сырого» текста

2. Парсинг этого текста и его интерпретация с помощью объектов языка Python.

3. Сериализация полученной совокупности объектов в структуру, соответствующую XML-файлу.

4. Создание нового XML-файла и запись в него.

Чтение и парсинг JSON-файла выполняет модуль prog\_main/prog\_main.py:

def DictToXML(d):

global xml

global counter

global maxcounter

for k, v in d.items():

xml += "\t" \* (counter + 1) + "<" + k + ">"

if isinstance(v, list):

for i in v:

if xml[-1] != "\n":

xml += "\n"

counter += 1

maxcounter = counter

DictToXML(i)

counter -= 1

elif isinstance(v, dict):

if xml[-1] != "\n":

xml += "\n"

counter += 1

maxcounter = counter

DictToXML(v)

counter -= 1

else:

maxcounter = counter

xml += str(v)

if counter == maxcounter:

xml += "</" + k + ">\n"

else:

xml += "\t" \* (counter + 1) + "</" + k + ">\n"

maxcounter = 0

counter = 0

xml = '<?xml version="1.0" encoding="UTF-8" ?>\n<main>\n'

with open("sch.json", "r", encoding="utf-8") as f:

text = f.read()

true = True

false = False

null = None

l = eval(text)

DictToXML(l)

xml += "</main>"

print(xml)

with open("output.xml", "w", encoding="utf-8") as f:

f.write(xml)

После запуска программа создаст файл output.xml в своей директории. Его содержимое:

prog\_main/output.xml

<?xml version="1.0" encoding="UTF-8" ?>

<main>

<Wednesday>

<period>

<time>08:20-09:50</time>

<week>3, 5, 9, 11, 13, 15, 17</week>

</period>

<place>

<room>1216/0 ауд.</room>

<building>Ломоносова пр., д.9, лит.М</building>

</place>

<class-info>

<lesson>информатика</lesson>

<teacher>Балакшин Павел Валерьевич</teacher>

</class-info>

<lesson-format>Очно - дистанционный</lesson-format>

</Wednesday>

</main>

## Дополнительное задание №1

Для сериализации данных в XML-файл можно использовать библиотеку json2xml. Её требуется дополнительно установить следующим образом:

pip install json2xml

Выполним конвертацию файла prog1/sch.json:

prog1/prog1.py

from json2xml import json2xml

from json2xml.utils import readfromjson

data = readfromjson("sch.json")

xml = json2xml.Json2xml(data).to\_xml()

print(xml)

with open("output.xml", "w", encoding="utf-8") as f:

f.write(xml)

Результат конвертации доступен в файле prog2/output.xml:

<?xml version="1.0" ?>

<all>

<Wednesday type="list">

<item type="dict">

<period type="dict">

<time type="str">08:20-09:50</time>

<week type="str">3, 5, 9, 11, 13, 15, 17</week>

</period>

<place type="dict">

<room type="str">1216/0 ауд.</room>

<building type="str">Ломоносова пр., д.9, лит.М</building>

</place>

<class-info type="dict">

<lesson type="str">информатика</lesson>

<teacher type="str">Балакшин Павел Валерьевич</teacher>

</class-info>

<lesson-format type="str">Очно - дистанционный</lesson-format>

</item>

</Wednesday>

f</all>

Отличие этого результата от результата программы, написанной вручную, состоит в следующем:

1. Тег main заменен на all.

2. У объектов указывается их тип.

3. Добавлен тег <item> для разных блоков.

Проанализировав различия, видно, что выводы обеих программ отличаются в некоторых стилистических моментах, которые не влияют на корректность их результатов, а также в новом теге, который может облегчить дальнейшее разделение блоков.

## Дополнительное задание №2

Напишем новый файл prog2.py, в котором для определения типов данных будут использоваться регулярные выражения.

prog2/prog2.py

import re

def JsonToDict(d):

for k, v in d.items():

array = []

nd = {}

regex = r"(?<=\")([\w|-]+)\"(?::)(?:(\".+?\")|\{(\".+?\")\})(?=[, ]?)"

for i in re.finditer(regex, v):

if i.group(1) in nd.keys():

array.append(JsonToDictPart2(nd))

nd = {}

if i.group(2) == None:

nd[i.group(1)] = i.group(3)

else:

nd[i.group(1)] = i.group(2)

array.append(JsonToDictPart2(nd))

d[k] = array

return d

def JsonToDictPart2(d):

for k, v in d.items():

a = re.search("\":", v)

if (re.search("\":", v)):

nd = {}

regex = r"(?<=\")([\w|-]+)\"(?::)(?:(\".+?\")|\{(\".+?\")\})(?=[, ]?)"

for i in re.finditer(regex, v):

if i.group(2) == None:

ch = str(i.group(3)).replace('"', "")

nd[i.group(1)] = ch

else:

ch = str(i.group(2)).replace('"', "")

nd[i.group(1)] = ch

d[k] = nd

else:

v = v.replace('"', "")

d[k] = v

return d

def DictToXML(d):

global xml

global counter

global maxcounter

for k, v in d.items():

xml += "\t" \* (counter + 1) + "<" + k + ">"

if isinstance(v, list):

for i in v:

if xml[-1] != "\n":

xml += "\n"

counter += 1

maxcounter = counter

DictToXML(i)

counter -= 1

elif isinstance(v, dict):

if xml[-1] != "\n":

xml += "\n"

counter += 1

maxcounter = counter

DictToXML(v)

counter -= 1

else:

maxcounter = counter

xml += str(v)

if (counter == maxcounter):

xml += "</" + k + ">\n"

else:

xml += "\t" \* (counter + 1) + "</" + k + ">\n"

maxcounter = 0

counter = 0

xml = '<?xml version="1.0" encoding="UTF-8" ?>\n<main>\n'

with open("sch.json", "r", encoding="utf-8") as f:

text = f.read()

text = text[1:-1]

d = {}

regex = r'(?<=")[А-Яа-яёЁ\w]+(?=":\[)'

km = re.findall(regex, text)

regex = r'(?<=:\[{)[^\]]+(?=})'

vm = re.findall(regex, text)

for i in range(len(km)):

d[km[i]] = vm[i]

a = JsonToDict(d)

DictToXML(a)

xml += "</main>"

print(xml)

with open("output.xml", "w", encoding="utf-8") as f:

f.write(xml)

Результат сохранён в файл output.xml.

<?xml version="1.0" encoding="UTF-8" ?>

<main>

<Wednesday>

<period>

<time>08:20-09:50</time>

<week>3, 5, 9, 11, 13, 15, 17</week>

</period>

<place>

<room>1216/0 ауд.</room>

<building>Ломоносова пр., д.9, лит.М</building>

</place>

<class-info>

<lesson>информатика</lesson>

<teacher>Балакшин Павел Валерьевич</teacher>

</class-info>

<lesson-format>Очно - дистанционный</lesson-format>

</Wednesday>

</main>

Никаких различий с файлом из обязательного задания нет.

## Дополнительное задание №3

Для замера времени многократного запуска функций конвертации будем использовать функцию time из встроенной библиотеки time.

from time import time

from json2xml import json2xml

from json2xml.utils import readfromjson

import re

st\_time1 = time()

#---Обязательное---#

def DictToXML(d):

global xml

global counter

global maxcounter

for k, v in d.items():

xml += " " \* counter + "<" + k + ">"

if isinstance(v, list):

for i in v:

if xml[-1] != "\n":

xml += "\n"

counter += 1

maxcounter = counter

DictToXML(i)

elif isinstance(v, dict):

if xml[-1] != "\n":

xml += "\n"

counter += 1

maxcounter = counter

DictToXML(v)

else:

maxcounter = counter

xml += str(v)

if counter == maxcounter:

xml += "</" + k + ">\n"

else:

xml += " " \* counter + "</" + k + ">\n"

counter -= 1

if counter == 0:

maxcounter = 0

maxcounter = 0

counter = 0

xml = '<?xml version="1.0" encoding="UTF-8" ?>\n'

with open("sch.json", "r") as f:

text = f.read()

true = True

false = False

null = None

l = eval(text)

DictToXML(l)

#print(xml)

t1 = (time() - st\_time1)\*(10)

print("--- Обязательное %s секунд \* 10 ---" % (t1))

st\_time2 = time()

#-----------№1------------#

data = readfromjson("sch.json")

xml = json2xml.Json2xml(data).to\_xml()

# print(xml)

t2 = (time() - st\_time2)\*(10)

print("--- №1 %s секунд \* 10 ---" % (t2))

st\_time3 = time()

#-----------№2------------#

def JsonToDict(d):

for k, v in d.items():

array = []

nd = {}

regex = r"(?<=\")([\w|-]+)\"(?::)(?:(\".+?\")|\{(\".+?\")\})(?=[, ]?)"

for i in re.finditer(regex, v):

if i.group(1) in nd.keys():

array.append(JsonToDictPart2(nd))

nd = {}

if i.group(2) is None:

nd[i.group(1)] = i.group(3)

else:

nd[i.group(1)] = i.group(2)

array.append(JsonToDictPart2(nd))

d[k]=array

return d

def JsonToDictPart2(d):

for k, v in d.items():

if re.search("\":", v):

nd = {}

regex = r"(?<=\")([\w|-]+)\"(?::)(?:(\".+?\")|\{(\".+?\")\})(?=[, ]?)"

for i in re.finditer(regex, v):

if i.group(2) is None:

ch = str(i.group(3)).replace('"', "")

nd[i.group(1)] = ch

else:

ch = str(i.group(2)).replace('"', "")

nd[i.group(1)] = ch

d[k] = nd

else:

v = v.replace('"', "")

d[k] =v

return d

def DictToXML(d):

global xml

global counter

global maxcounter

for k, v in d.items():

xml += " " \* counter + "<" + k + ">"

if isinstance(v, list):

for i in v:

if xml[-1] != "\n":

xml += "\n"

counter += 1

maxcounter = counter

DictToXML(i)

elif isinstance(v, dict):

if xml[-1] != "\n":

xml += "\n"

counter += 1

maxcounter = counter

DictToXML(v)

else:

maxcounter = counter

xml += str(v)

if counter == maxcounter:

xml += "</" + k + ">\n"

else:

xml += " " \* counter + "</" + k + ">\n"

counter -= 1

if counter == 0:

maxcounter = 0

maxcounter = 0

counter = 0

xml = '<?xml version="1.0" encoding="UTF-8" ?>\n'

with open("sch.json", "r") as f:

text = f.read()

text = text[1:-1]

d = {}

regex = r'(?<=")[А-Яа-яёЁ\w]+(?=":\[)'

km = re.findall(regex, text)

regex = r'(?<=:\[{)[^\]]+(?=})'

vm = re.findall(regex, text)

for i in range(len(km)):

d[km[i]]=vm[i]

a = JsonToDict(d)

DictToXML(a)

#print(xml)

t3 = (time() - st\_time3) \* 10

print("--- №2 %s секунд \* 10 ---" % t3)

Вывод программы представлен на рисунке 1.

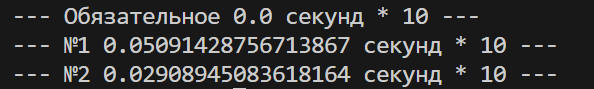


Рисунок 1 – Вывод программы

Выходит, что самая быстрая реализация – конвертация вручную без регулярных выражений. Можно предположить, что рекурсивный алгоритм, который в ней реализован, оптимален для небольшого файла, а также не требует дополнительных обращений к библиотекам, таким как re.

## Дополнительное задание №4

Будем конвертировать исходный JSON-файл в формат Markdown. Он позволяет создавать структурированные документы с использованием минимального количества символов и форматирования текста с помощью некоторых символов. Markdown применяется для создания простых веб-страниц, блогов, документации и прочего.

prog4/prog4.py

def DictToMarkdown(d):

global md

global rekdeep

for k, v in d.items():

md += ("#" \* rekdeep) + " " + k + "\n"

if isinstance(v, list):

for i in v:

md += "\n------\n"

rekdeep += 1

DictToMarkdown(i)

rekdeep -= 1

elif isinstance(v, dict):

rekdeep += 1

DictToMarkdown(v)

rekdeep -= 1

else:

md += v + "\n"

rekdeep = 1

md = ''

with open("sch.json", "r", encoding="utf-8") as f:

text = f.read()

true = True

false = False

null = None

l = eval(text)

DictToMarkdown(l)

print(md)

with open("markdown.md", "w", encoding="utf-8") as f:

f.write(md)

После выполнения блока с сериализацией считанных данных из JSON-файла в директории появится файл markdown.md. Его содержимое:

# Wednesday

------

## period

### time

08:20-09:50

### week

3, 5, 9, 11, 13, 15, 17

## place

### room

1216/0 ауд.

### building

Ломоносова пр., д.9, лит.М

## class-info

### lesson

информатика

### teacher

Балакшин Павел Валерьевич

## lesson-format

Очно - дистанционный

Как видно, его содержимое довольно легко прочитать и понять, что оно из себя представляет.

# Заключение

В результате выполнения данной лабораторной работы были изучены следующие вещи:

* Форматы JSON, XML и Markdown;
* Понятие парсинга данных и его практическая реализация для считывания JSON-файла;
* Понятие сериализации данных и её практическая реализация для записи данных в XML-файл;
* Библиотеки для работы с JSON- и XML-файлами;
* Применение регулярных выражений при парсинге файла.

# Список использованной литературы

1. Балакшин Е.А., Соснин П.В., Машина В.В. Информатика. –   
СПб: Университет ИТМО, 2020.

2. Лямин А. В., Череповская Е. Н. Объектно-ориентированное программирование. Компьютерный практикум. – СПб: Университет ИТМО, 2017.