Университет ИТМО

Лабораторная работа №4

“Исследование протоколов,

форматов обмена информацией

и

языков разметки документов ”

Вариант 34

Выполнил

Студент группы Р3112

Шаматульский Роман Константинович

Преподаватель

Рыбаков Степан Дмитриевич

Санкт-Петербург, 2023

Оглавление

[Задания 3](#_Toc149862809)

[Основные этапы вычисления 4](#_Toc149862810)

[Выводы 8](#_Toc149862811)

[Источники 9](#_Toc149862812)

## Задания

1. Исходя из структуры расписания конкретного дня, сформировать файл с расписанием в формате, указанном в задании в качестве исходного. При этом необходимо, чтобы в выбранном дне было не менее двух занятий (можно использовать своё персональное). В случае, если в данный день недели нет таких занятий, то увеличить номер варианта ещё на восемь.
2. Обязательное задание (позволяет набрать до 45 процентов от максимального числа баллов БаРС за данную лабораторную): написать программу на языке Python 3.x, которая бы осуществляла парсинг и конвертацию исходного файла в новый путём простой замены метасимволов исходного формата на метасимволы результирующего формата. (Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML файлов.)
3. Дополнительное задание №1 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
   * Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.
   * Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать.
   * Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
4. Дополнительное задание №2 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
   * Переписать исходный код, добавив в него использование регулярных выражений.
   * Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
5. Дополнительное задание №4 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную).
   * Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле.
   * Проанализировать полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

## Основные этапы вычисления

**Обязательное задание:**

lines = []

result = ''

i = 0

def sub\_list(tag, spaces, body):

i = 0

sub = []

result = ''

while i < len(body):

if body[i].endswith(':\n'):

if body[i].lstrip().startswith('-'):

body[i] = body[i].replace('-', ' ', 1)

cont = 0

sub = []

for j in range(i + 1, len(body)):

if body[j].startswith(' ' \* (spaces + 2)):

cont += 1

sub.append(body[j])

else:

break

result += sub\_list(body[i][:body[i].find(':')].lstrip(), spaces + 2, sub)

i += cont

elif body[i].lstrip().startswith('-') and ':' not in body[i]:

body[i] = body[i].replace('-', ' ', 1)

result += body[i][:body[i].find(body[i].lstrip()[0]) - 2] + '<' + body[i].strip() + ' type=listelement>\n'

elif body[i].strip() != '':

if body[i].lstrip().startswith('-'):

body[i] = body[i].replace('-', ' ', 1)

body[i] = body[i][2:]

result += body[i][:body[i].find(body[i].lstrip()[0])] + '<' + body[i].lstrip()[:body[i].find(':') - spaces] + '>' + body[i][body[i].find(':') + 2 : -1] + '</' + body[i].lstrip()[:body[i].find(':') - spaces] + '>\n'

i += 1

return ' ' \* (spaces - 2) + '<' + tag + '>\n' + result + ' ' \* (spaces - 2) + '</' + tag + '>\n'

with open('lab4\_1.yaml', 'r') as yaml\_file:

for line in yaml\_file: #добавляя, убираем коммы

if '#' in line:

line = line[:line.find('#')]

if line.strip() != '':

lines.append(line.rstrip())

for h in range(len(lines)): # убираем кавычки, если они есть

if '\"' in lines[h]:

lines[h] = lines[h].replace('\"', '')

while i < len(lines):

if lines[i].endswith(':'):

sub = []

cont = 0

for j in range(i + 1, len(lines)):

if lines[j].startswith(' '):

sub.append(lines[j] + '\n')

cont += 1

else:

break

result += sub\_list(lines[i][:lines[i].find(':')], 2, sub)

i += cont

elif lines[i].strip() != '':

result += '<' + lines[i][:lines[i].find(':')] + '>' + lines[i][lines[i].find(':') + 2:] + '</' + lines[i][:lines[i].find(':')] + '>\n'

i += 1

with open('lab4\_1.xml', 'w') as xml\_file:

xml\_file.write(result)

*Решение на python*

saturday:

first-pair:

name: web-programming

week: odd

time: 13:30-15:00

room: 2334

adress: Kronversky pr. h.49 let.A

professor: Makarevich Roman Dmitrievich

format: full-time/distance

second-pair:

name: physics

week: odd

time: 15:20-16:50

room: 1410

adress: Kronversky pr. h.49 let.A

professor: Unnamed

format: full-time/distance

*Исходный yaml-файл*

<saturday>

<first-pair>

<name>web-programming</name>

<week>odd</week>

<time>13:30-15:00</time>

<room>2334</room>

<adress>Kronversky pr. h.49 let.A</adress>

<professor>Makarevich Roman Dmitrievich</professor>

<format>full-time/distance</format>

</first-pair>

<second-pair>

<name>physics</name>

<week>odd</week>

<time>15:20-16:50</time>

<room>1410</room>

<adress>Kronversky pr. h.49 let.A</adress>

<professor>Unnamed</professor>

<format>full-time/distance</format>

</second-pair>

</saturday>

*Результат – xml-файл*

**Задание 2 (доп. №1)**

from pprint import pprint

import xmltodict

import yaml

with open(‘lab4\_2.yaml’, ‘r’) as yaml\_file:

input\_file = yaml.safe\_load(yaml\_file)

result = xmltodict.unparse(input\_file, pretty=True)

with open(‘lab4\_2.xml’, ‘w’) as xml\_file:

xml\_file.write(result)

*Решение на python*

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

<saturday>

<first-pair>

<name>web-programming</name>

<week>odd</week>

<time>13:30-15:00</time>

<room>2334</room>

<adress>Kronversky pr. h.49 let.A</adress>

<professor>Makarevich Roman Dmitrievich</professor>

<format>full-time/distance</format>

</first-pair>

<second-pair>

<name>physics</name>

<week>odd</week>

<time>15:20-16:50</time>

<room>1410</room>

<adress>Kronversky pr. h.49 let.A</adress>

<professor>Unnamed</professor>

<format>full-time/distance</format>

</second-pair>

</saturday>

*Результат*

Результат схож с исходной программой, так как работа обоих алгоритмов схожа: первый считает отступы строк, разделяя их по уровням, а второй преобразует строки во вложенные словари.

**Задание 3 (Доп. №2)**

import re

lines = []

result = ''

i = 0

def sub\_list(tag, spaces, body):

i = 0

sub = []

result = ''

while i < len(body):

if re.fullmatch(r'[- A-Za-z0-9]\*:\n', body[i]):

if re.fullmatch(r'[ ]\*-[ ][-a-zA-Z0-9]\*:\n', body[i]):

body[i] = body[i].replace('-', ' ', 1)

cont = 0

sub = []

for j in range(i + 1, len(body)):

if body[j].startswith(' ' \* (spaces + 2)):

cont += 1

sub.append(body[j])

else:

break

result += sub\_list(body[i][:body[i].find(':')].lstrip(), spaces + 2, sub)

i += cont

elif re.fullmatch(r'[ ]\*-[ ][A-Z0-9a-z]\*\n', body[i]):

body[i] = body[i].replace('-', ' ', 1)

result += body[i][:body[i].find(body[i].lstrip()[0]) - 2] + '<' + body[i].strip() + ' type=listelement>\n'

elif not re.fullmatch(r'[ \n\t]\*', body[i]):

if re.fullmatch(r'[ ]\*-[ ][a-zA-Z0-9]\*: [A-Za-z0-9. ]\*\n', body[i]):

body[i] = body[i].replace('-', ' ', 1)

body[i] = body[i][2:]

result += body[i][:body[i].find(body[i].lstrip()[0])] + '<' + body[i].lstrip()[:body[i].find(':') - spaces] + '>' + body[i][body[i].find(':') + 2 : -1] + '</' + body[i].lstrip()[:body[i].find(':') - spaces] + '>\n'

i += 1

return ' ' \* (spaces - 2) + '<' + tag + '>\n' + result + ' ' \* (spaces - 2) + '</' + tag + '>\n'

with open('lab4\_3.yaml', 'r') as yaml\_file:

for line in yaml\_file: #добавляя, убираем коммы

if '#' in line:

line = line[:line.find('#')]

if line.strip() != '':

lines.append(line.rstrip())

for h in range(len(lines)): # убираем кавычки, если они есть

if '\"' in lines[h]:

lines[h] = lines[h].replace('\"', '')

while i < len(lines):

if re.fullmatch(r'[A-Za-z0-9]\*:', lines[i]):

sub = []

cont = 0

for j in range(i + 1, len(lines)):

if lines[j].startswith(' '):

sub.append(lines[j] + '\n')

cont += 1

else:

break

result += sub\_list(lines[i][:lines[i].find(':')], 2, sub)

i += cont

elif not re.fullmatch(r'[ \n\t]\*', lines[i]):

result += '<' + lines[i][:lines[i].find(':')] + '>' + lines[i][lines[i].find(':') + 2:] + '</' + lines[i][:lines[i].find(':')] + '>\n'

i += 1

with open('lab4\_3.xml', 'w') as xml\_file:

xml\_file.write(result)

*Решение на python*

<saturday>

<first-pair>

<name>web-programming</name>

<week>odd</week>

<time>13:30-15:00</time>

<room>2334</room>

<adress>Kronversky pr. h.49 let.A</adress>

<professor>Makarevich Roman Dmitrievich</professor>

<format>full-time/distance</format>

</first-pair>

<second-pair>

<name>physics</name>

<week>odd</week>

<time>15:20-16:50</time>

<room>1410</room>

<adress>Kronversky pr. h.49 let.A</adress>

<professor>Unnamed</professor>

<format>full-time/distance</format>

</second-pair>

*Результат*

Результат одинаков с исходной программой, так как общий алгоритм выполнения не поменялся, поменялся лишь способ фильтрации данных.

**Задание 4 (Доп. №4)**

|  |  |
| --- | --- |
| **Задание** | **Время выполнения (x100)** |
| Обязательное | 0.03792346499994892 |
| Доп. №1 | 0.1997266739999759 |
| Доп. №2 | 0.042411081999944145 |

Программа, написанная с использованием сторонних библиотек работает дольше всего, так как библиотеки способны парсить любые файлы в заданном формате, соответственно занимают больше места в памяти и дольше выполняются, в отличие от программы из обязательного задания.

Программа, написанная с использованием регулярных выражений работает немного дольше программы из обязательного задания, так как с помощью регулярных выражений мы ищем соответствие всей строки, в то время, как в программе из обязательного задания мы проверяем лишь начало строки на определённое количество пробелов.

## Выводы

В ходе выполнения данной лабораторной работы я познакомился с такими форматами файлов, как YAML, XML, научился читать их. Кроме того я научился взаимодействовать с такими файлами, используя python.

## Источники

* 1. YAML за пять минут. Синтаксис и остальные возможности. -URL: <https://tproger.ru/translations/yaml-za-5-minut-sintaksis-i-osnovnye-vozmozhnosti> (дата обращения: 02.11.23)
  2. PyYAML. -URL: <https://pypi.org/project/PyYAML/> <https://tproger.ru/translations/yaml-za-5-minut-sintaksis-i-osnovnye-vozmozhnosti> (дата обращения: 02.11.23)
  3. Синтаксис языка XML. -URL: <https://script-coding.com/XML.html> (дата обращения: 02.11.23)