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 №1**

по дисциплине

«ИНФОРМАТИКА»

Вариант № 17

***Выполнил:***

Студент группы P3112

Файзиев Ф.Р.

***Преподаватель:***

Белозубов А.В.

г.Санкт-Петербург, 2022

Оглавление

[Оглавление 2](#_Toc1677150479)

[Задание](#_Задание) 3

[Обязательное задание 2](#_Задание)

[Дополнительное задание №1](#_Задание) 8

[Дополнительное задание №2](#_Задание) 10

[Дополнительное задание №3](#_Задание) 10

[Заключение](#_Toc1595890911) [11](#_Заключение)

[Источники](#_Toc1079661971) [11](#_Список_литературы.)

# **Задание**

Порядок выполнения работы

1. Обязательное задание (позволяет набрать до 65 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.

2. 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). a) Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов. b) Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать. c) Сравнить полученные результаты и объяснить их сходство/различие.

4.Дополнительное задание №2 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную). a) Переписать исходный код, добавив в него использование регулярных выражений. 2 b) Сравнить полученные результаты и объяснить их сходство/различие.

5.Дополнительное задание №3 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную). a) 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. b) Проанализировать полученные результаты и объяснить их сходство/различие.

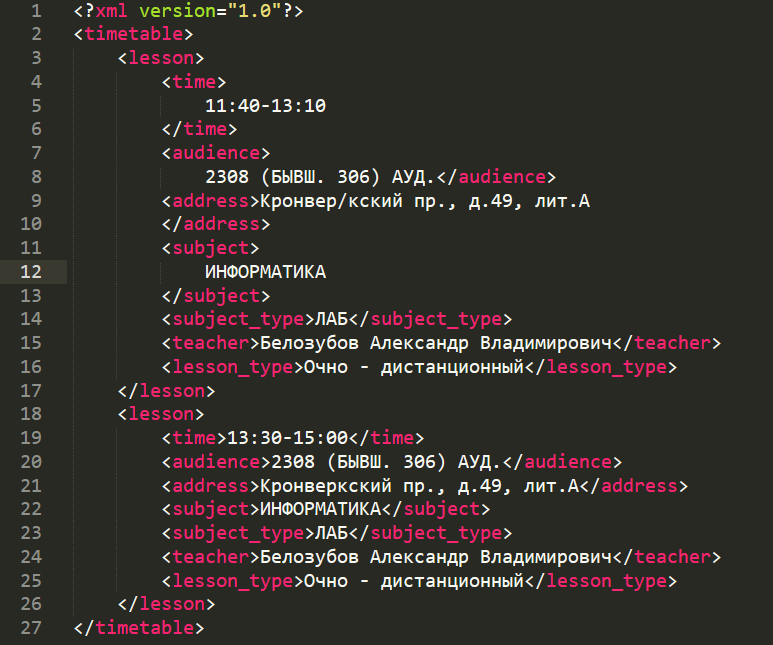
Вариант (17)

|  |  |  |
| --- | --- | --- |
| **Исходный формат** | **Результирующий формат** | **День недели** |
| XML | YAML | Среда |

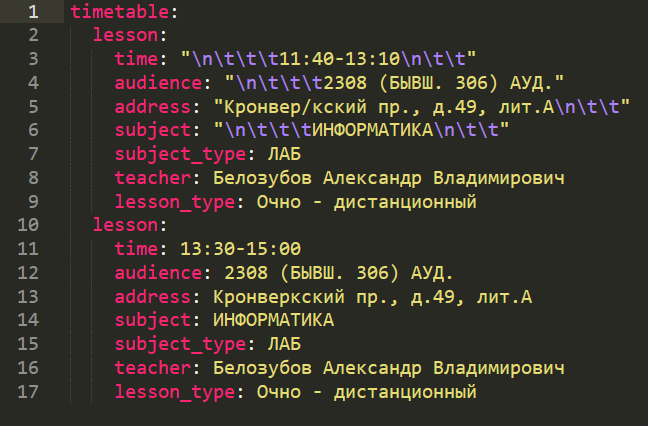
Обязательное задание

from time import \*  
  
tm = time()  
tab = 0  
not\_first = 0  
yml = ""  
content = ""  
is\_open = False  
  
def settabs(tab):  
 tabs = ""  
 for i in range(tab):  
 tabs += "\t"  
 return tabs  
  
def rep\_tab(line):  
 i = line.find("\t")  
 while i != -1:  
 line = line[:i] + "\\t" + line[i + 1:]  
 i = line.find("\t")  
 return line  
  
def brackets(line):  
 idx = line.find("\\t")  
 if idx != -1:  
 return "\"" + line + "\""  
 else:  
 return line  
  
with open("input.xml", "r", encoding="utf-8") as file:  
 for line in file:  
 o\_tag = ""  
 c\_tag = ""  
  
 if not\_first:  
 idx\_of\_ot\_open = line.find('<')  
 idx\_of\_ot\_close = line.find('>')  
  
 idx\_of\_ct\_open = line.rfind('</')  
 idx\_of\_ct\_close = line.rfind('>')  
  
 if idx\_of\_ot\_open != -1 and idx\_of\_ot\_open != idx\_of\_ct\_open:  
 o\_tag = line[idx\_of\_ot\_open + 1 : idx\_of\_ot\_close]  
 if idx\_of\_ct\_open != -1:  
 c\_tag = line[idx\_of\_ct\_open + 2 : idx\_of\_ct\_close]  
  
 if o\_tag and c\_tag:  
 yml += "\n" + settabs(tab) + o\_tag + ': '  
 content += line[idx\_of\_ot\_close + 1 : idx\_of\_ct\_open]  
 yml += brackets(rep\_tab(content))  
 content = ""  
 is\_open = False  
  
 elif o\_tag:  
 yml += "\n" + settabs(tab) + o\_tag + ': '  
 content += line[idx\_of\_ot\_close + 1 : -1]  
 tab += 1  
 is\_open = True  
  
 elif c\_tag:  
 if is\_open:  
 content += "\\n" + line[ : idx\_of\_ct\_open]  
 yml += brackets(rep\_tab(content))  
 content = ""  
 tab -= 1  
 is\_open = False  
  
 else:  
 content += "\\n" + rep\_tab(line[:-1])  
  
  
 not\_first += 1  
  
with open("4.yaml", "w", encoding="utf-8") as o:  
 o.write(yml[1:])  
  
print("Время работы реализации =", time() - tm, "мс")

**input**



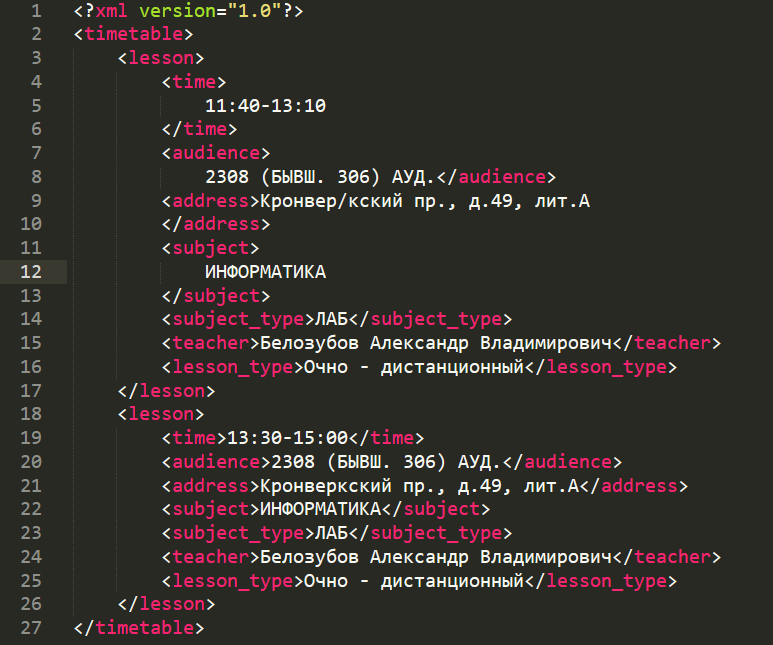
**Output**



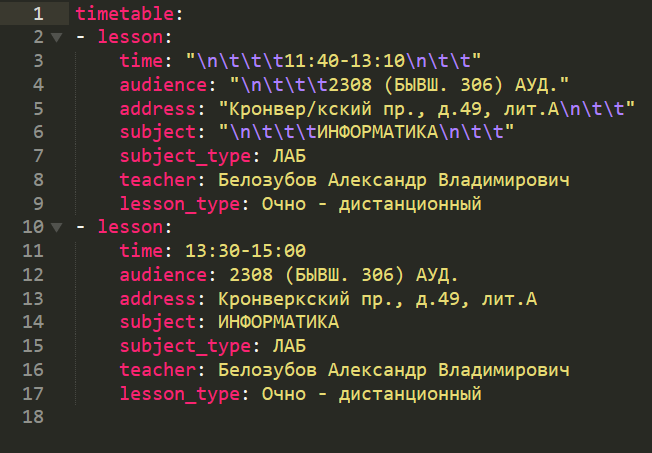
Дополнительное задание №1

import xmlplain  
from time import \*  
  
tm = time()  
  
with open("input.xml", "r", encoding="utf-8") as file:  
 xml = xmlplain.xml\_to\_obj(file.read(), strip\_space=True, fold\_dict=True)  
  
with open("lib.yaml", "w", encoding="utf-8") as yml:  
 xmlplain.obj\_to\_yaml(xml, yml)  
  
print("Время работы с библиотеками =", time() - tm, "мс")

**input**



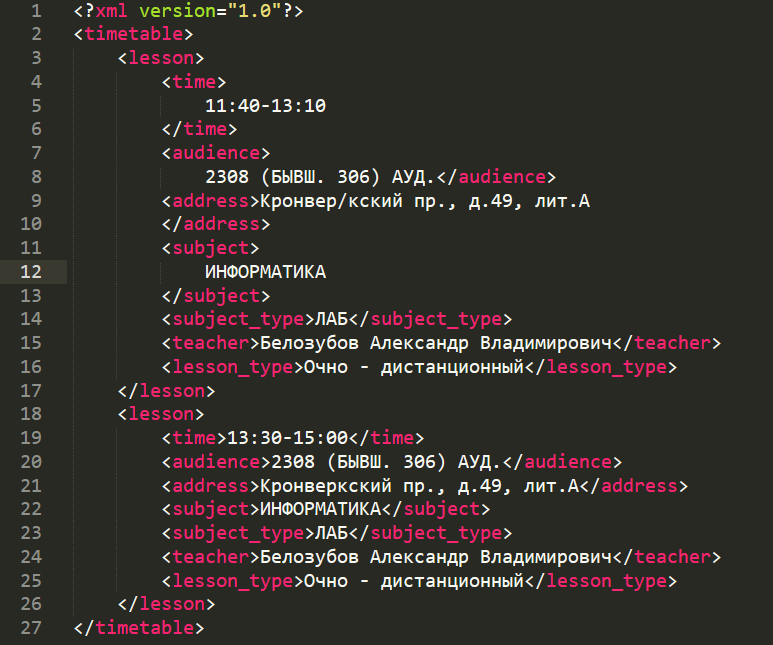
**Output**



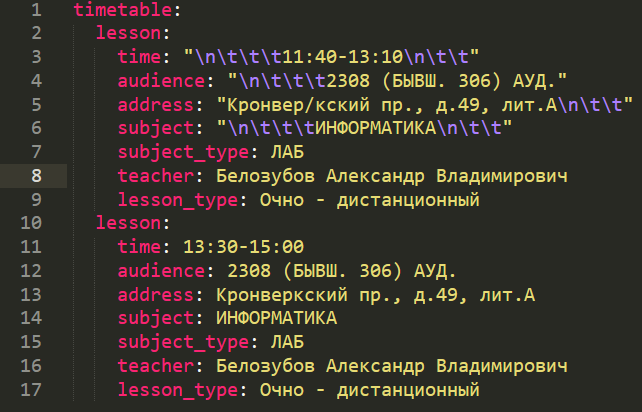
Дополнительное задание №2

from re import \*  
from time import \*  
  
tm = time()  
tab = 0  
not\_first = 0  
reg = ""  
content = ""  
is\_open = False  
  
def settabs(tab):  
 tabs = ""  
 for i in range(tab):  
 tabs += "\t"  
 return tabs  
  
def rep\_tab(line):  
 return sub(r"\t", r"\\t", line)  
  
def brackets(line):  
 count = len(findall(r"\\t", line))  
 if count:  
 return "\"" + line + "\""  
 else:  
 return line  
  
with open("input.xml", "r", encoding="utf-8") as file:  
 for line in file:  
  
 if not\_first:  
 open\_close = findall(r"(<([\w\d]+)>.\*</\2>)", line)  
 open\_ = findall(r"<[\w\d]+>.\*", line)  
 close\_ = findall(r".\*</[\w\d]+>", line)  
  
 if len(open\_close):  
 reg += "\n" + settabs(tab) + findall(r"<([\w\d]+)>.\*", open\_close[0][0])[0] + ": "  
 content += findall(r"<[\w\d]+>(.\*)</[\w\d]+>", open\_close[0][0])[0]  
 reg += brackets(rep\_tab(content))  
 content = ""  
 is\_open = False  
  
 elif len(open\_):  
 reg += "\n" + settabs(tab) + findall(r"<([\w\d]+)>", open\_[0])[0] + ": "  
 content += findall(r"<[\w\d]+>(.\*)", open\_[0])[0]  
 tab += 1  
 is\_open = True  
  
 elif len(close\_):  
 if is\_open:  
 content += "\\n" + findall(r"(.\*)</[\w\d]+>", close\_[0])[0]  
 reg += brackets(rep\_tab(content))  
 content = ""  
 tab -= 1  
 is\_open = False  
  
 else:  
 content += "\\n" + rep\_tab(line[:-1])  
  
 not\_first += 1  
  
  
with open("reg.yaml", "w", encoding="utf-8") as o:  
 o.write(reg[1:])  
  
print("Время работы с регулярными выражениями =", time() - tm, "мс")

**input**



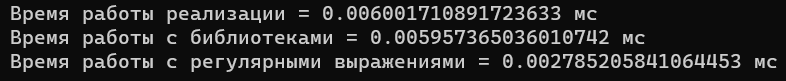
**Output**



Дополнительное задание №3

with open("4.py", "r", encoding="utf-8") as c4:  
 code\_c4 = c4.read()  
  
with open("lib.py", "r", encoding="utf-8") as lib:  
 code\_lib = lib.read()  
  
with open("reg.py", "r", encoding="utf-8") as reg:  
 code\_reg = reg.read()  
  
exec(code\_c4)  
exec(code\_lib)  
exec(code\_reg)

**Output**

****

# **Заключение**

В процессе выполнения лабораторной работы я изучал парсинг в языке программирования Python, как работает XML и YAML и научился работать с этими форматами.

# **Список литературы**.

1. Лямин А.В., Череповская Е.Н. Объектно-ориентированное программирование. Компьютерный практикум. – СПб: Университет ИТМО, 2017. – 143 с..
2. [*https://books.ifmo.ru/file/pdf/2256.pdf*](https://books.ifmo.ru/file/pdf/2256.pdf)