Федеральное государственное автономное образовательное учреждение

высшего образования «Национальный исследовательский университет

ИТМО»

*Факультет программной инженерии и компьютерной техники*

**Лабораторная работа №4**

**По дисциплине “Информатика”**

Вариант: 17

XML –> YAML

Выполнил:

Саранча Павел

Группа: Р3109

Преподаватель:

Рудникова Тамара Владимировна

Санкт-Петербург, 2023г

Оглавление

[Задание: 2](#_Toc150615012)

[Исходный XML файл с именем ‘schedule.xml’: 3](#_Toc150615013)

[Обязательное задание: 3](#_Toc150615014)

[Доп задание №1: 5](#_Toc150615015)

[Доп задание №3: 7](#_Toc150615016)

[Доп задание №4: 9](#_Toc150615017)

[Доп задание №5, Перевод из XML в html: 9](#_Toc150615018)

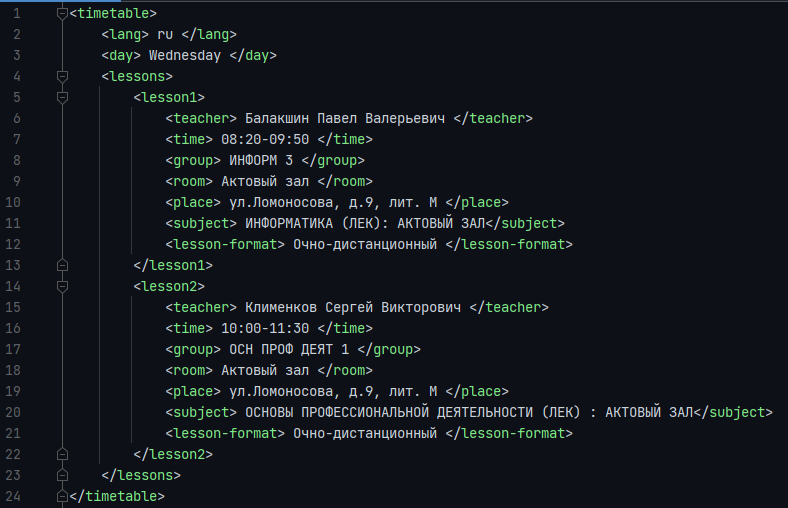
[Заключение: 11](#_Toc150615019)

[Список литературы: 12](#_Toc150615020)

# Задание:

1. (Обязательное задание) 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.
2. (Доп задание №1) Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов. Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать.
3. (Доп задание №3) Переписать исходный код таким образом, чтобы для решения задачи использовались формальные грамматики. То есть ваш код должен уметь осуществлять парсинг и конвертацию любых данных, представленных в исходном формате, в данные, представленные в результирующем формате: как с готовыми библиотеками из дополнительного задания №1.
4. (Доп задание №4) Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле.
5. (Доп задание №4) 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.

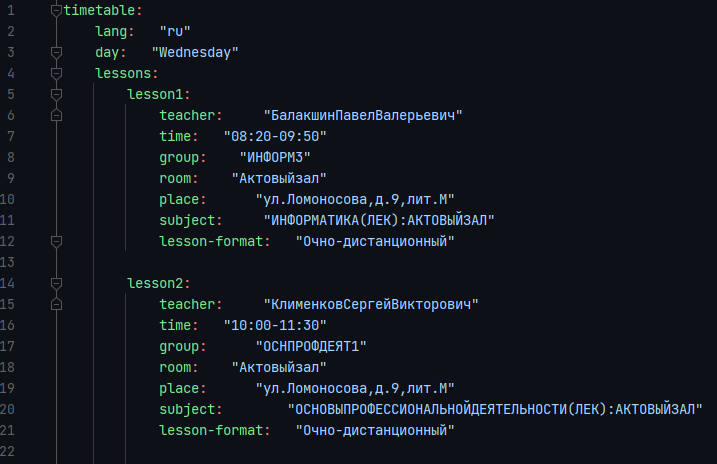
# Исходный XML файл с именем ‘schedule.xml’:



# 

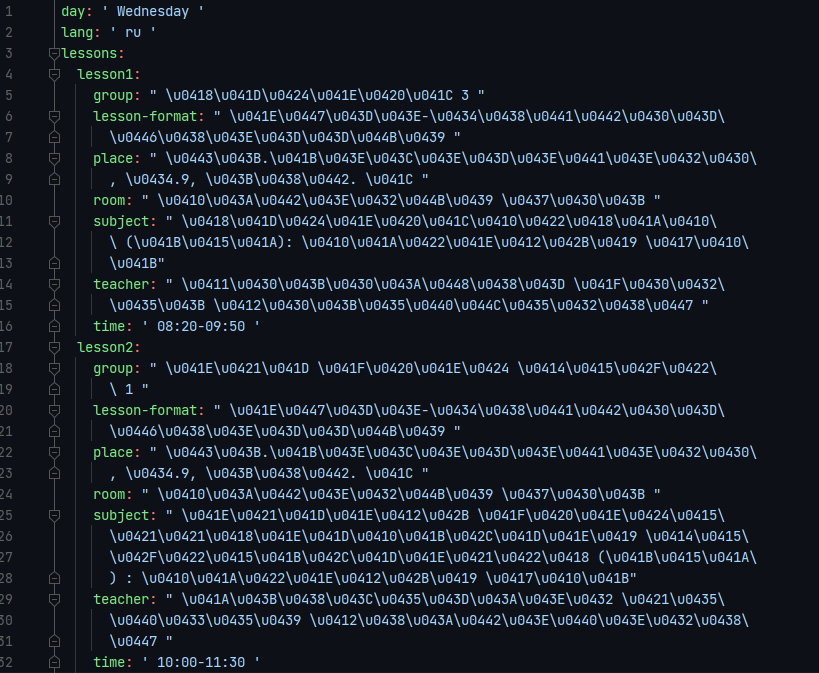
# Обязательное задание:

1. # -\*- coding: utf-8 -\*-
2. **import** time
4. start\_time **=** time.time()
6. fin **=** open('schedule.xml', encoding**=**'utf-8')
7. fout **=** open('schedule.yaml', 'w+')
9. # read = 1, считывание названия тегов внутри < >
10. # read = 2, считывание текста заключенного между открывающим и закрывающим тегом <> здесь </>
11. # read = -1, в конец файла ничего не добавляется
12. read **=** 0
13. word **=** ''
14. **for** line **in** fin:
15. xml **=** line  # каждая строка в xml файле
16. yaml **=** ''  # строка, которая будет добавляться в yaml файл
18. **for** i **in** range(len(xml)):
20. **if** (read !**=** 2 **and** xml[i] **==** '\t') **or** xml[i] **==** ' ' **or** xml[i] **==** '\n':
21. yaml **=** yaml **+** xml[i]
22. **elif** xml[i] **==** '<':
23. **if** xml[i **+** 1] !**=** '/':
24. read **=** 1
25. **else**:
26. read **=** **-**1
27. **elif** xml[i] **==** '>':
28. **if** read **==** 1:
29. yaml **=** yaml **+** ': '
30. read **=** 2
31. **elif** read **==** **-**1:
32. **if** word !**=** '':
33. yaml **=** yaml **+** '"' + word + '"'
34. word **=** ''
35. read **=** 0
36. **elif** read **==** 1:
37. yaml **=** yaml **+** xml[i]
38. **elif** read **==** 2:
39. word **=** word **+** xml[i]
40. fout.write(yaml)
42. fout.close()
44. end\_time **=** time.time()
46. # Стократное время выполнения программы
47. execution\_time **=** (end\_time **-** start\_time) **\*** 100
48. print(f"General Стократное время выполнения программы: {execution\_time} секунд")

Вывод (Обязательного задание): 

# Доп задание №1:

1. start\_time **=** time.time()
2. # Чтение содержимого XML-файла
3. tree **=** ET.parse('schedule.xml')
4. root **=** tree.getroot()
6. # Функция для рекурсивного преобразования элементов XML в словарь
7. **def** xml\_to\_dict(element):
8. **if** len(element) **==** 0:
9. **return** element.text
10. result **=** {}
11. **for** child **in** element:
12. child\_data **=** xml\_to\_dict(child)
13. **if** child.tag **in** result:
14. **if** type(result[child.tag]) **is** list:
15. result[child.tag].append(child\_data)
16. **else**:
17. result[child.tag] **=** [result[child.tag], child\_data]
18. **else**:
19. result[child.tag] **=** child\_data
20. **return** result
22. # Преобразование XML в словарь
23. xml\_dict **=** xml\_to\_dict(root)
25. # Преобразование словаря в YAML
26. yaml\_content **=** yaml.dump(xml\_dict, default\_flow\_style**=**False)
28. # Запись результата в YAML-файл
29. with open('schedule.yaml', 'w') as yaml\_file:
30. yaml\_file.write(yaml\_content)
32. print("Конвертация завершена успешно.")
34. end\_time **=** time.time()
36. # Стократное время выполнения программы
37. execution\_time **=** (end\_time **-** start\_time) **\*** 100
38. print(f"dop1 Стократное время выполнения программы: {execution\_time} секунд")

Вывод (Доп задания № 1):

# Доп задание №3:

1. # -\*- coding: utf-8 -\*-
2. **import** time

5. PUNCTUATION **=** r"""!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~"""

8. **def** get\_open\_tag\_name(s: str) **-**> str:
9. **for** i **in** range(len(s)):
10. **if** s[i] **==** '>':
11. **return** s[:i**+**1]
13. **raise** ValueError('Key not found')

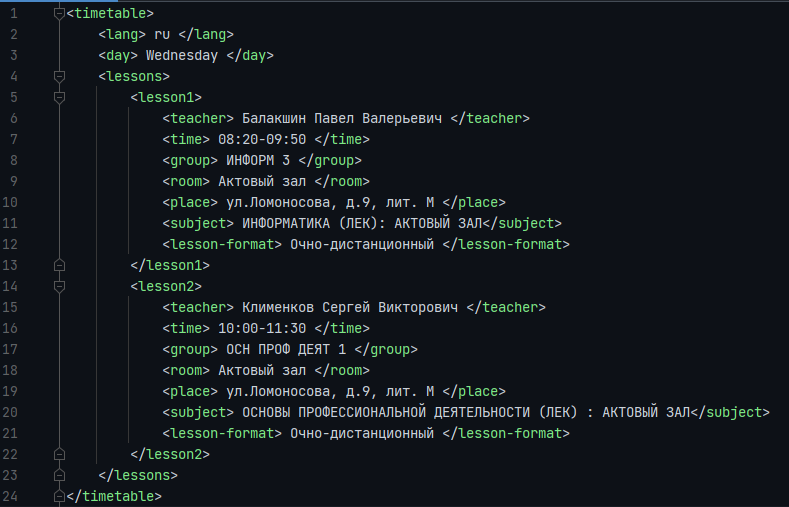
16. **def** get\_close\_tag\_name(s: str) **-**> str:
17. **return** s.replace('<', '</')

20. **def** get\_text\_without\_tag(text: str, open\_tag: str, close\_tag: str):
21. result **=** text.replace(open\_tag, '').replace(close\_tag, '').strip()
23. **if** result:
24. **if** **not** result.isalnum() **or** **not** result.isascii():
25. result **=** f'\'{result}\''
27. **return** result

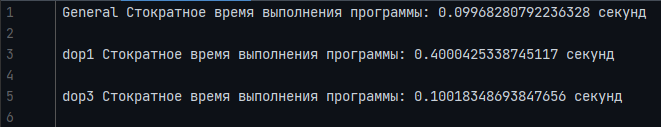
30. **def** get\_yaml\_content(lines) **-**> str:
31. open\_tags **=** []
33. yml\_content **=** []
34. **for** row **in** lines:
36. tab\_lenght **=** '\t' **\*** len(open\_tags)
37. tmp\_row **=** row.strip()
39. **if** tmp\_row[0] **==** '<' **and** tmp\_row[1] **not** **in** PUNCTUATION:
40. open\_tag **=** get\_open\_tag\_name(tmp\_row)
41. close\_tag **=** get\_close\_tag\_name(open\_tag)
42. text **=** get\_text\_without\_tag(tmp\_row, open\_tag, close\_tag)
44. **if** tmp\_row[**-**len(close\_tag):] !**=** close\_tag:
45. open\_tags.append(close\_tag)
47. yml\_content.append(f'{tab\_lenght}{open\_tag[1:-1]}: {text}')
49. **elif** tmp\_row[0] **==** '<' **and** tmp\_row[1] **==** '/':
50. **if** open\_tags[**-**1] !**=** tmp\_row:
51. ValueError('Bad XML file')
52. open\_tags.pop()
54. **if** len(open\_tags) !**=** 0:
55. ValueError('Bad XML file')
57. **return** '\n'.join(yml\_content)

60. **def** main():
61. start\_time **=** time.time()
62. with open('schedule.xml', mode**=**'r', encoding**=**'utf-8') as f:
63. xml\_lines **=** f.readlines()
65. yaml\_content **=** get\_yaml\_content(xml\_lines)
67. with open('schedule.yaml', mode**=**'w') as f:
68. f.write(yaml\_content)
69. end\_time **=** time.time()
70. # Стократное время выполнения программы
71. execution\_time **=** (end\_time **-** start\_time) **\*** 100
72. print(f"dop 3 Стократное время выполнения программы: {execution\_time} секунд")

75. **if** \_\_name\_\_ **==** '\_\_main\_\_':
76. main()

Вывод (Доп задание №3):

# Доп задание №4:

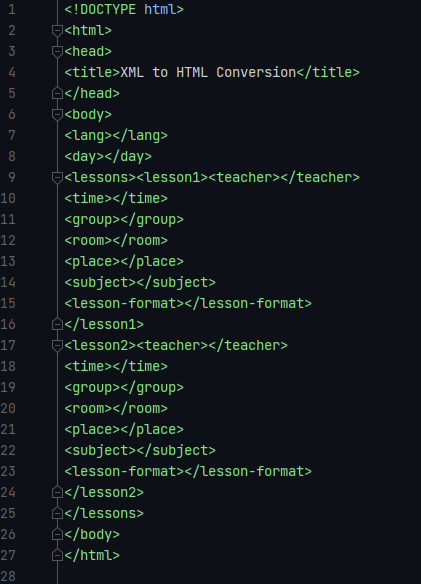


# Доп задание №5, Перевод из XML в html:

1. # -\*- coding: utf-8 -\*-
2. **import** xml.etree.ElementTree as ET
3. **from** html **import** escape

6. **def** convert\_xml\_to\_html(xml\_file\_path, html\_file\_path):
7. # Парсинг XML файла
8. tree **=** ET.parse(xml\_file\_path)
9. root **=** tree.getroot()
11. # Открытие HTML файла для записи
12. with open(html\_file\_path, 'w', encoding**=**'utf-8') as html\_file:
13. # Запись начальных тегов HTML
14. html\_file.write('<!DOCTYPE html>\n<html>\n<head>\n<title>XML to HTML Conversion</title>\n</head>\n<body>\n')
16. # Рекурсивная функция для обхода элементов XML и их преобразования в HTML
17. **def** convert\_element\_to\_html(element):
18. nonlocal html\_file
20. # Запись открывающего тега
21. html\_file.write(f'<{escape(element.tag)}>')
23. # Запись атрибутов
24. **for** key, value **in** element.attrib.items():
25. html\_file.write(f' {escape(key)}="{escape(value)}"')
27. # Рекурсивный вызов для дочерних элементов
28. **for** child **in** element:
29. convert\_element\_to\_html(child)
31. # Запись закрывающего тега
32. html\_file.write(f'</{escape(element.tag)}>\n')
34. # Начало обхода элементов XML
35. **for** child **in** root:
36. convert\_element\_to\_html(child)
38. # Запись закрывающих тегов HTML
39. html\_file.write('</body>\n</html>')
41. # Пример использования
42. convert\_xml\_to\_html('schedule.xml', 'schedule.html')

Вывод (Доп задание №5):



# 

# Заключение:

В процессе выполнения этой работы я узнал об особенностях структуре таких форматов, как XML, YAML и html, совершал преобразования между этими форматами, познакомился с парсингом на языке python.

# Список литературы:

1. Балакшин П.В., Соснин В.В., Калинин И.В., Малышева Т.А., Раков С.В., Рущенко Н.Г., Дергачев А.М. Информатика: лабораторные работы и тесты: Учебно-методическое пособие / Рецензент: Поляков В.И. - Санкт-Петербург: Университет ИТМО, 2019. - 56 с. - экз. - Режим доступа:

<https://books.ifmo.ru/book/2248/informatika:_laboratornye_raboty_i_testy:_uchebno-metodicheskoe_posobie_/_recenzent:_polyakov_v.i..htm>

1. Грошев А.С. Г89 Информатика: Учебник для вузов / А.С. Грошев. – Архангельск, Арханг. гос. техн. ун-т, 2010. -470с. -Режим доступа <https://narfu.ru/university/library/books/0690.pdf>
2. Орлов С. А., Цилькер Б. Я. Организация ЭВМ и систем: Учебник для  
   вузов. 2-е изд. – СПб.: Питер, 2011. – 688 с.: ил.
3. Алексеев Е.Г., Богатырев С.Д. Информатика. Мультимедийный  
   электронный учебник. – Режим доступа: http://inf.e-  
   alekseev.ru/text/toc.html