Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Дисциплина: «Информатика»

**Лабораторная работа по информатике №4**

**«Исследование протоколов, форматов обмена информацией и языков разметки документов»**

­

Выполнил: Третьяков Илья Антонович

Группа: P3108

Вариант: 11

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург

2021

# Оглавление

[Оглавление 2](#_Toc88939435)

[Задание 2](#_Toc88939436)

[Обязательное задание 3](#_Toc88939437)

[Дополнительное задание №1 9](#_Toc88939438)

[Дополнительное задание №2 10](#_Toc88939439)

[Дополнительное задание №3 12](#_Toc88939440)

[Дополнительное задание №4 13](#_Toc88939441)

[Вывод 15](#_Toc88939442)

[Список литературы 15](#_Toc88939443)

# Задание

1. Определить номер варианта как остаток деления на 36 порядкового номера в списке группы в ISU. В случае, если в данный день недели нет занятий, то увеличить номер варианта на восемь.
2. Изучить форму Бэкуса-Наура.
3. Изучить особенности протоколов и форматов обмена информацией между системами: JSON, YAML, XML.
4. Понять устройство страницы с расписанием для своей группы: https://itmo.ru/ru/schedule/0/P3110/schedule.htm
5. Исходя из структуры расписания конкретного дня, сформировать файл с расписанием в формате, указанном в задании в качестве исходного.
6. 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.
7. Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML-файлов.
8. Дополнительное задание 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.

1. Дополнительное задание задание №2 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).

a)  Переписать исходный код, добавив в него использование регулярных выражений.

b)  Сравнить полученные результаты и объяснить их сходство/различие.

10.Дополнительное задание 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 десятикратное время выполнения парсинга + конвертации в цикле.

b)  Проанализировать полученные результаты и объяснить их сходство/различие.

11.Дополнительное задание задание №4 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную.

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

b)  Проанализировать полученные результаты, объяснить осоебнности использованного формата.

# Обязательное задание

Файл с расписанием группы в формате XML (вторник):

*<?***xml version="1.1" encoding="UTF-8"***?>*<**schedule**>  
 <**day**>Вт</**day**>  
 <**class id="1"**>  
 <**time**>  
 <**hours**>10:00-11:30</**hours**>  
 <**days**>четная неделя</**days**>  
 <**room**>306 ауд.</**room**>  
 <**address**>Кронверкский пр., д.49, лит.А</**address**>  
 </**time**>  
 <**location**>  
 <**room**>306 ауд.</**room**>  
 <**address**>Кронверкский пр., д.49, лит.А</**address**>  
 </**location**>  
 <**lesson**>  
 <**name**>Информатика</**name**>  
 <**type**>Лабораторное занятие</**type**>  
 <**week**>четная неделя</**week**>  
 <**professor**>Малышева Татьяна Алексеевна</**professor**>  
 </**lesson**>  
 <**lesson-format**>Очно - дистанционный</**lesson-format**>  
 </**class**>  
 <**class id="2"**>  
 <**time**>  
 <**hours**>11:40-13:10</**hours**>  
 <**days**>четная неделя</**days**>  
 <**room**>306 ауд.</**room**>  
 <**address**>Кронверкский пр., д.49, лит.А</**address**>  
 </**time**>  
 <**location**>  
 <**room**>306 ауд.</**room**>  
 <**address**>Кронверкский пр., д.49, лит.А</**address**>  
 </**location**>  
 <**lesson**>  
 <**name**>Информатика</**name**>  
 <**type**>Лабораторное занятие</**type**>  
 <**week**>четная неделя</**week**>  
 <**professor**>Малышева Татьяна Алексеевна</**professor**>  
 </**lesson**>  
 <**lesson-format**>Очно - дистанционный</**lesson-format**>  
 </**class**>  
 <**class id="3"**>  
 <**time**>  
 <**hours**>13:30-15:00</**hours**>  
 <**days**>четная неделя</**days**>  
 <**room**>285 ауд.</**room**>  
 <**address**>Кронверкский пр., д.49, лит.А</**address**>  
 </**time**>  
 <**location**>  
 <**room**>285 ауд.</**room**>  
 <**address**>Кронверкский пр., д.49, лит.А</**address**>  
 </**location**>  
 <**lesson**>  
 <**name**>Дискретная математика</**name**>  
 <**type**>Лекция</**type**>  
 <**week**>четная неделя</**week**>  
 <**professor**>Поляков Владимир Иванович</**professor**>  
 </**lesson**>  
 <**lesson-format**>Очно - дистанционный</**lesson-format**>  
 </**class**>  
 <**class id="4"**>  
 <**time**>  
 <**hours**>18:40-20:10</**hours**>  
 <**days**>2 ноября</**days**>  
 <**room**>Дистанционный</**room**>  
 <**address**></**address**>  
 </**time**>  
 <**location**>  
 <**room**></**room**>  
 <**address**></**address**>  
 </**location**>  
 <**lesson**>  
 <**name**>Математика</**name**>  
 <**type**>Лекция</**type**>  
 <**week**></**week**>  
 <**professor**>Холодова Светлана Евгеньевна</**professor**>  
 </**lesson**>  
 <**lesson-format**>Дистанционный</**lesson-format**>  
 </**class**>  
</**schedule**>

Программа по переводу файлов в формате XML в формат YAML:

*#!/usr/bin/python3***import** sys  
  
  
**class** Node:  
 **def** \_\_init\_\_(self, name):  
 self.\_\_name = name  
 self.\_\_body = **""** self.\_\_children = []  
 self.\_\_parent = **None  
  
 def** add\_child(self, child):  
 **if** type(child) == Node:  
 self.\_\_children.append(child)  
 child.\_\_parent = self  
  
 **def** set\_body(self, body):  
 body = body.replace(**"&gt;"**, **">"**).replace(**"&lt;"**, **">"**).replace(**"&#38;"**, **"&"**)  
 self.\_\_body = body  
  
 **def** get\_name(self):  
 **return** self.\_\_name  
  
 **def** get\_body(self):  
 **return** self.\_\_body  
  
 **def** get\_children(self):  
 **return** self.\_\_children  
  
 **def** get\_parent(self):  
 **return** self.\_\_parent  
  
  
**def** open\_xml(file):  
 **with** open(file) **as** file:  
 file = file.read()  
 *# change formatting from 3 spaces to 2 spaces  
 # validate header if present and remove it for parsing* header = file[file.find(**'<'**, 0):file.find(**'>'**, 0) + 1]  
 **if** header.startswith(**"<?xml "**) **and** header.endswith(**"?>"**):  
 header = header.replace(**"'"**, **""**)  
 variants = [**'<?xml version="1.0" encoding="UTF-8"?>'**, **'<?xml version="1.1" encoding="UTF-8"?>'**,  
 **'<?xml version="1.0"?>'**, **'<?xml version="1.1"?>'**]  
 **if** header **in** variants:  
 file = file.replace(header, **''**)  
 **else**:  
 sys.exit(**"ParseError: error parsing header"**)  
 *# check that the root element is present and enclosed* first\_tag = file[file.find(**'<'**, 0):file.find(**'>'**, 0) + 1].replace(**"<"**, **"</"**)  
 file = file.strip()  
 **if** file.endswith(first\_tag):  
 **return** file  
 **else**:  
 sys.exit(**"ParseError: "** + first\_tag + **" missing"**)  
  
  
**def** strip\_attrs(name):  
 attrs = name.split()  
 name = attrs.pop(0)  
 **for** i **in** range(len(attrs)):  
 attr\_name, attr\_value = attrs[i].split(**"="**)  
 *# chooses quotes* **if** attr\_value[0] == attr\_value[-1] == **'"'**:  
 attr\_value = attr\_value.strip(**'"'**)  
 **elif** attr\_value[0] == attr\_value[-1] == **"'"**:  
 attr\_value = attr\_value.strip(**"'"**)  
 **else**:  
 exit(**"ParseError: error parsing attribute value"**)  
 *# int to str (xml attributes can't be ints)* **if** attr\_value.isnumeric():  
 attr\_value = **"'"** + attr\_value + **"'"** attrs[i] = attr\_name, attr\_value  
 **return** name, attrs  
  
  
**def** write\_line(file, n, name, body, indent):  
 **if "\n" in** body.strip():  
 body = body.strip()  
 body = body + **"\n"  
 if** indent == **'dict'**:  
 file.write(n \* **' '** + name + **": "** + body)  
 **elif** indent == **'list'**:  
 file.write((n - 1) \* **' '** + **'- '** + name + **": "** + body)  
  
  
**def** write\_to\_file(node, file, n=-1, indent=**'dict'**):  
 *# changes the indentation to list-like instead of the default to allow for duplicate keys* **if** node.get\_name() != **"root" and** node.get\_parent().get\_children():  
 names = [x.get\_name() **for** x **in** node.get\_parent().get\_children()]  
 **if** len(set(names)) == len(names):  
 indent = **'dict'  
 else**:  
 indent = **'list'  
  
 if** node.get\_name() == **"root"**:  
 **for** child **in** node.get\_children():  
 write\_to\_file(child, file, n + 1, indent)  
 **else**:  
 *# case 1: node only has children -> print its children* **if** node.get\_children() **and not** node.get\_body():  
 write\_line(file, n, node.get\_name(), **''**, indent)  
 **for** child **in** node.get\_children():  
 write\_to\_file(child, file, n + 1, indent)  
 *# case 2: node has a body -> print the body and children if present* **elif** node.get\_body():  
 body = node.get\_body()  
 *# tries to format newlines (it's hard)* **if "\n " in** body.strip():  
 body = **">"** + body  
 **elif "\n" in** body.strip():  
 body = **"'"** + body.replace(**"\n"**, **r"\n"**)  
 **if ": " in** body **or** ((body.startswith(**" "**) **or** body.endswith(**" "**)) **and "\n" not in** body):  
 **if "'" not in** body:  
 body = **"'"** + body + **"'"  
 elif '"' not in** body:  
 body = **'"'** + body + **'"'  
 else**:  
 sys.exit(**"ParseError: quote problems in '"** + node.get\_name() + **"'"**)  
 *# prints children and body in a list* **if** node.get\_children():  
 write\_line(file, n, node.get\_name(), **''**, indent)  
 **for** child **in** node.get\_children():  
 write\_line(file, (n + 1), child.get\_name(), child.get\_body(), **'list'**)  
 file.write(n \* **' '** + **'- '** + node.get\_body() + **"\n"**)  
 *# prints only body* **else**:  
 write\_line(file, n, node.get\_name(), body, indent)  
 *# case 3: node has neither -> print a null character* **elif not** node.get\_children():  
 write\_line(file, n, node.get\_name(), **" ~"**, indent)  
  
  
**def** main(args):  
 file\_in = file\_out = **''** usage = **"usage: ./parser.py [-f XML file] [-o YAML output file name]"  
 try**:  
 **if** args[1] == **"-f"**:  
 file\_in = args[2]  
 **if** args[3] == **"-o"**:  
 file\_out = args[4]  
 **except** IndexError:  
 sys.exit(usage)  
  
 i = 0  
 f = open\_xml(file\_in).replace(**" "**, **" "**)  
 name = body = **""** in\_name = in\_body = **False** root = Node(**"root"**)  
 this\_node = root  
 parents = []  
 **while** i < len(f):  
 *# open name tag* **if** f[i] == **'<' and** f[i + 1] != **'/'**:  
 name = **""** in\_name = **True** *# getting name insides* **elif** in\_name **and** f[i] != **"<" and** f[i] != **">"**:  
 name += f[i]  
 *# closing name tag* **elif** in\_name **and** f[i] == **">"**:  
 in\_name = **False** name, attrs = strip\_attrs(name)  
 node = Node(name)  
 *# adds xml attributes as separate keys* **if** attrs:  
 **for** attr\_name, attr\_value **in** attrs:  
 attr\_name = **"'@"** + attr\_name + **"'"** attr\_node = Node(attr\_name)  
 attr\_node.set\_body(attr\_value)  
 node.add\_child(attr\_node)  
 this\_node.add\_child(node)  
 parents.append(this\_node)  
 this\_node = node  
 in\_body = **True** body = **""** *# getting body* **elif** in\_body **and** f[i] != **"<" and** f[i] != **">"**:  
 body += f[i]  
 *# closing body* **elif** f[i] == **"<" and** f[i + 1] == **"/"**:  
 close\_tag = f[(i + 2):f.find(**">"**, i)]  
 parent\_children = [x.get\_name() **for** x **in** parents[-1].get\_children()]  
  
 in\_body = **False** this\_node.set\_body(body)  
 this\_node = parents.pop()  
 body = **""  
  
 if** close\_tag **not in** parent\_children:  
 sys.exit(**"ParseError: tag error somewhere in '"** + parents[-1].get\_name() + **"'"**)  
  
 i += 1  
  
 **with** open(file\_out, **"w"**) **as** out:  
 write\_to\_file(root, out)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main(sys.argv)

Результат работы программы:

**schedule**:  
- **day**: Вт  
- **class**:  
 **'@id'**: **'1'  
 time**:  
 **hours**: 10:00-11:30  
 **days**: четная неделя  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **location**:  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **lesson**:  
 **name**: Информатика  
 **type**: Лабораторное занятие  
 **week**: четная неделя  
 **professor**: Малышева Татьяна Алексеевна  
 **lesson-format**: Очно - дистанционный  
- **class**:  
 **'@id'**: **'2'  
 time**:  
 **hours**: 11:40-13:10  
 **days**: четная неделя  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **location**:  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **lesson**:  
 **name**: Информатика  
 **type**: Лабораторное занятие  
 **week**: четная неделя  
 **professor**: Малышева Татьяна Алексеевна  
 **lesson-format**: Очно - дистанционный  
- **class**:  
 **'@id'**: **'3'  
 time**:  
 **hours**: 13:30-15:00  
 **days**: четная неделя  
 **room**: 285 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **location**:  
 **room**: 285 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **lesson**:  
 **name**: Дискретная математика  
 **type**: Лекция  
 **week**: четная неделя  
 **professor**: Поляков Владимир Иванович  
 **lesson-format**: Очно - дистанционный  
- **class**:  
 **'@id'**: **'4'  
 time**:  
 **hours**: 18:40-20:10  
 **days**: 2 ноября  
 **room**: Дистанционный  
 **address**: ~  
 **location**:  
 **room**: ~  
 **address**: ~  
 **lesson**:  
 **name**: Математика  
 **type**: Лекция  
 **week**: ~  
 **professor**: Холодова Светлана Евгеньевна  
 **lesson-format**: Дистанционный

# Дополнительное задание №1

Так как язык Python не имеет встроенных инструментов для работы с XML файлами, установим стороннюю библиотеку xmlplain с помощью команды

**$ pip install xmlplain**

Исходный код новой программы:

**import** xmlplain  
  
xml\_name = **"schedule.xml"**yaml\_name = **"schedule1.yaml"  
  
with** open(xml\_name) **as** inf:  
 root = xmlplain.xml\_to\_obj(inf, strip\_space=**True**, fold\_dict=**True**)  
  
**with** open(yaml\_name, **"w"**) **as** outf:  
 xmlplain.obj\_to\_yaml(root, outf)

Результат работы программы:

**schedule**:  
- **day**: Вт  
- **class**:  
 **'@id'**: **'1'  
 time**:  
 **hours**: 10:00-11:30  
 **days**: четная неделя  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **location**:  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **lesson**:  
 **name**: Информатика  
 **type**: Лабораторное занятие  
 **week**: четная неделя  
 **professor**: Малышева Татьяна Алексеевна  
 **lesson-format**: Очно - дистанционный  
- **class**:  
 **'@id'**: **'2'  
 time**:  
 **hours**: 11:40-13:10  
 **days**: четная неделя  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **location**:  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **lesson**:  
 **name**: Информатика  
 **type**: Лабораторное занятие  
 **week**: четная неделя  
 **professor**: Малышева Татьяна Алексеевна  
 **lesson-format**: Очно - дистанционный  
- **class**:  
 **'@id'**: **'3'  
 time**:  
 **hours**: 13:30-15:00  
 **days**: четная неделя  
 **room**: 285 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **location**:  
 **room**: 285 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **lesson**:  
 **name**: Дискретная математика  
 **type**: Лекция  
 **week**: четная неделя  
 **professor**: Поляков Владимир Иванович  
 **lesson-format**: Очно - дистанционный  
- **class**:  
 **'@id'**: **'4'  
 time**:  
 **hours**: 18:40-20:10  
 **days**: 2 ноября  
 **room**: Дистанционный  
 **address**: **''  
 location**:  
 **room**: **''  
 address**: **''  
 lesson**:  
 **name**: Математика  
 **type**: Лекция  
 **week**: **''  
 professor**: Холодова Светлана Евгеньевна  
 **lesson-format**: Дистанционный

Единственным отличием между выводами программ является то, что библиотека на место пустых значений ставит пустую строку (**''**) вместо специального символа ~.

# Дополнительное задание №2

Исходный код программы:

**import** re  
**from** parser **import** Node, open\_xml, write\_to\_file  
  
  
file\_in = **"schedule.xml"**file\_out = **"schedule2.yaml"**i = 0  
name = body = **""**in\_name = in\_body = **False**f = open\_xml(file\_in)  
f = f.replace(**" "**, **""**).replace(**"\n"**, **""**)  
root = Node(**"root"**)  
this\_node = root  
parents = []  
name\_reg = **r'<([A-Za-z\\_]+[A-Za-z0-9\-\\_\.]\*)(?:\ ([A-Za-z\\_]+[A-Za-z0-9\-\\_\.]\*(?=\=))\=(?:\"([^\"]+)\"|\'([^\']+)\'))\*>'**body\_reg = **r"(?<=>)([^<>]\*?)</([A-Za-z\\_]+[A-Za-z0-9\-\\_\.]\*)>"  
while** i < len(f):  
 **if** f[i] == **'<' and** f[i+1] != **'/'**:  
 in\_name = **True** reg\_res = re.search(name\_reg, f[i:])  
 name = reg\_res.group(1)  
 node = Node(name)  
 **if** reg\_res.group(2):  
 attr\_name = **"'@"** + reg\_res.group(2) + **"'"** attr\_value = reg\_res.group(3) **if** reg\_res.group(3) **else** reg\_res.group(4)  
 **if** attr\_value.isnumeric():  
 attr\_value = **"'"** + attr\_value + **"'"** attr\_node = Node(attr\_name)  
 attr\_node.set\_body(attr\_value)  
 node.add\_child(attr\_node)  
 i += len(name)  
 **elif** in\_name **and** f[i] == **">"**:  
 in\_name = **False** this\_node.add\_child(node)  
 parents.append(this\_node)  
 this\_node = node  
 name\_temp = name  
 name = **""** in\_body = **True** body = **""  
 elif** in\_body **and** f[i] != **"<" and** f[i] != **">"**:  
 body\_rxd = re.search(body\_reg, f[i-1:])  
 body = body\_rxd.group(1)  
 **if** body\_rxd:  
 j = f.find(body\_rxd.group(1), i)  
 i = j + len(body)  
 i = f.find(**"</"**, i) - 1  
 **elif** f[i] == **"<" and** f[i+1] == **"/"**:  
 in\_body = **False** this\_node.set\_body(body)  
 this\_node = parents.pop()  
  
 body = **""** i += 1  
  
  
**with** open(file\_out, **"w"**) **as** out:  
 write\_to\_file(root, out)

Результат работы программы:

**schedule**:  
- **day**: Вт  
- **class**:  
 **'@id'**: **'1'  
 time**:  
 **hours**: 10:00-11:30  
 **days**: четная неделя  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **location**:  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **lesson**:  
 **name**: Информатика  
 **type**: Лабораторное занятие  
 **week**: четная неделя  
 **professor**: Малышева Татьяна Алексеевна  
 **lesson-format**: Очно - дистанционный  
- **class**:  
 **'@id'**: **'2'  
 time**:  
 **hours**: 11:40-13:10  
 **days**: четная неделя  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **location**:  
 **room**: 306 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **lesson**:  
 **name**: Информатика  
 **type**: Лабораторное занятие  
 **week**: четная неделя  
 **professor**: Малышева Татьяна Алексеевна  
 **lesson-format**: Очно - дистанционный  
- **class**:  
 **'@id'**: **'3'  
 time**:  
 **hours**: 13:30-15:00  
 **days**: четная неделя  
 **room**: 285 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **location**:  
 **room**: 285 ауд.  
 **address**: Кронверкский пр., д.49, лит.А  
 **lesson**:  
 **name**: Дискретная математика  
 **type**: Лекция  
 **week**: четная неделя  
 **professor**: Поляков Владимир Иванович  
 **lesson-format**: Очно - дистанционный  
- **class**:  
 **'@id'**: **'4'  
 time**:  
 **hours**: 18:40-20:10  
 **days**: 2 ноября  
 **room**: Дистанционный  
 **address**: ~  
 **location**:  
 **room**: ~  
 **address**: ~  
 **lesson**:  
 **name**: Математика  
 **type**: Лекция  
 **week**: ~  
 **professor**: Холодова Светлана Евгеньевна  
 **lesson-format**: Дистанционный

Между выводами программ нет разницы, так как регулярные выражения использовались для замены участка цикла при сканировании имен тэгов и их значений. Но стоит отметить, что с помощью регулярных выражений затруднительно доставать атрибута и их значения из тэгов при наличии больше одного атрибута, что в нашем случае и не понадобилось.

# Дополнительное задание №3

**$ time ./parser.py -f schedule.xml -o schedule.yaml**

**real 0m0.087s**

**user 0m0.011s**

**sys 0m0.013s**

**$ time python3 task1.py**

**real 0m0.115s**

**user 0m0.043s**

**sys 0m0.024s**

**$ time python3 task2.py**

**real 0m0.048s**

**user 0m0.012s**

**sys 0m0.007s**

Разница между временем исполнения программ из обязательного задания и доп. задания 2 минимальна, так как структурно программы не особо различаются, кроме двух небольших регулярных выражений. Существующая разница может заключаться в том, что в первой программе существенно больше кода, объявляемых функций, классов и т. д., а также происходит больше проверок файла на ошибки. В то же время программа из доп. задания 1 исполняется дольше в несколько раз (user time). Скорее всего, это связано с тем, что готовая библиотека реализована иначе и, вероятно, использует более строгие методы конвертации и проверяет множество возможных синтаксических ошибок в исходном файле.

# Дополнительное задание №4

Так как форматы XML и CSV значительно отличаются в своей сути (CSV это формат представления табличных данных, а XML это язык разметки), то затруднительным представляется написать 1-в-1 конвертер между этими форматами. Поэтому написанная программа подходит лишь для данного файла и ему подобным (с отличающимися значениями, но не структурой).

Исходный код программы:

**from** parser **import** Node, open\_xml, strip\_attrs  
  
file\_in = **"schedule.xml"**file\_out = **"schedule.csv"**i = 0  
f = open\_xml(file\_in)  
name = body = **""**in\_name = in\_body = **False**root = Node(**"root"**)  
this\_node = root  
parents = []  
**while** i < len(f):  
 **if** f[i] == **'<' and** f[i + 1] != **'/'**:  
 name = **""** in\_name = **True  
 elif** in\_name **and** f[i] != **"<" and** f[i] != **">"**:  
 name += f[i]  
 **elif** in\_name **and** f[i] == **">"**:  
 in\_name = **False** name, attrs = strip\_attrs(name)  
 node = Node(name)  
 this\_node.add\_child(node)  
 parents.append(this\_node)  
 this\_node = node  
 in\_body = **True** body = **""  
 elif** in\_body **and** f[i] != **"<" and** f[i] != **">"**:  
 body += f[i]  
 **elif** f[i] == **"<" and** f[i + 1] == **"/"**:  
 in\_body = **False** this\_node.set\_body(body)  
 this\_node = parents[len(parents) - 1]  
 parents.pop()  
 body = **""** i += 1  
  
**with** open(file\_out, **"w"**) **as** file:  
 rows = root.get\_children()[0].get\_children()[1:]  
 temp\_line = **""  
 for** i **in** rows[0].get\_children():  
 **if** i.get\_children():  
 **for** j **in** i.get\_children():  
 temp\_line += j.get\_name() + **","** temp\_line = temp\_line.rstrip(**","**) + **"\n"** file.write(temp\_line)  
 **for** i **in** rows:  
 temp\_line = **""  
 for** j **in** i.get\_children():  
 **if** j.get\_children():  
 **for** k **in** j.get\_children():  
 temp\_line += k.get\_body().replace(**","**, **""**) + **","** temp\_line = temp\_line.rstrip(**","**) + **"\n"** file.write(temp\_line)

Результат работы программы:

hours,days,room,address,room,address,name,type,week,professor  
10:00-11:30,четная неделя,306 ауд.,Кронверкский пр. д.49 лит.А,306 ауд.,Кронверкский пр. д.49 лит.А,Информатика,Лабораторное занятие,четная неделя,Малышева Татьяна Алексеевна  
11:40-13:10,четная неделя,306 ауд.,Кронверкский пр. д.49 лит.А,306 ауд.,Кронверкский пр. д.49 лит.А,Информатика,Лабораторное занятие,четная неделя,Малышева Татьяна Алексеевна  
13:30-15:00,четная неделя,285 ауд.,Кронверкский пр. д.49 лит.А,285 ауд.,Кронверкский пр. д.49 лит.А,Дискретная математика,Лекция,четная неделя,Поляков Владимир Иванович  
18:40-20:10,2 ноября,Дистанционный,,,,Математика,Лекция,,Холодова Светлана Евгеньевна

Представление в табличном виде:

Graphical user interface, text, application

Description automatically generated

Рисунок 1 Представление CSV-файла в табличном виде

# Вывод

Проделав лабораторную работу, я познакомился и научился работать с наиболее популярными форматами обмена и сериализации данных, такими как XML, YAML и JSON, а также с менее известными, такими как CSV, TSV и PROTOBUF. Помимо этого, я научился использовать готовые библиотеки для быстрого перевода между вышеупомянутыми форматами.

# Список литературы

1. XML [Электронный ресурс]: Википедия. Свободная энциклопедия. – Режим доступа: <https://ru.wikipedia.org/wiki/XML>.
2. YAML [Электронный ресурс]: Википедия. Свободная энциклопедия. – Режим доступа: <https://ru.wikipedia.org/wiki/YAML>.
3. Лямин А.В., Череповская Е.Н. Объектно-ориентированное программирование. Компьютерный практикум. – СПб: Университет ИТМО, 2017. – 143 с. – Режим доступа: <https://books.ifmo.ru/file/pdf/2256.pdf>.