

Отчет по лабораторной работе №2

Иванчиков Борис

1 декабря 2023 г.

Разработать XML-формат для представления расписания учебных занятий вашей группы. В расписании должна храниться информация о занятиях на каждый день недели. Для каждого учебного занятия указаны название предмета, аудитория, преподаватель, время начала и окончания, тип занятия (лекция или практика).

Задание 1

В разработанном формате представить расписание текущей недели. (1 балл)

Я разработал следующий документ:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <timetable>
4
5      <session day="mon">
6          <subject>Микроэкономика-2</subject>
7          <classroom>341</classroom>
8          <instructor>А. Кальчевский</instructor>
9          <start>12:30:00</start>
10         <end>13:50:00</end>
11         <type>seminar</type>
12     </session>
13
14     <session day="tue">
15         <subject>Макроэкономика-1</subject>
16         <classroom>R503</classroom>
17         <instructor>В. Черноокый</instructor>
18         <start>13:00:00</start>
19         <end>14:20:00</end>
20         <type>lecture</type>
21     </session>
22
23     ...
24
25     <session day="sat">
26         <subject>Машинное обучение</subject>
27         <classroom>R307</classroom>
28         <instructor>С. Корпачев</instructor>
29         <start>14:40:00</start>
30         <end>16:00:00</end>
31         <type>seminar</type>
32     </session>
33
34 </timetable>
```

Задание 2

Разработать *xPath*-запросы (2 балла):

- Получить все занятия на данной неделе
- Получить все аудитории, в которых проходят занятия.
- Получить все практические занятия на неделе.
- Получить все лекции, проводимые в указанной аудитории.
- Получить список всех преподавателей, проводящих практики в указанной аудитории.
- Получить последнее занятие для каждого дня недели.
- Получить общее количество занятий за всю неделю.

Я использовал XPath 1.0 и не разобрался, как установить версию 2.0, что усложнило мне задачу. Тем не менее, получилось реализовать все запросы

- `//session`

Результат:

```
1      [Line 10] session:
2          Микроэкономика-2
3          341
4          А. Кальчевский
5          12:30:00
6          13:50:00
7          seminar
8      ...
9      [Line 145] session:
10         Машинное обучение
11         R307
12         С. Корпачев
13         14:40:00
14         16:00:00
15         seminar
```

- `/timetable/session[not(classroom=preceding-sibling::session/classroom)]/classroom`

Результат:

```
1      [Line 12] classroom: 341
2      [Line 21] classroom: R503
3      [Line 30] classroom: R205
4      [Line 48] classroom: S224
5      [Line 57] classroom: G119
6      [Line 66] classroom: R504
7      [Line 75] classroom: R506
8      [Line 102] classroom: online
9      [Line 147] classroom: R307
```

Пояснение: чтобы аудитории не повторялись, выбираем только те элементы `session`, у которых аудитория не повторяется в каком-то из предшествующих элементов. Затем из них выбираем элементы `classroom`.

c. //session[type="seminar"]

Результат:

```
1      [Line 10] session:
2          Микроэкономика-2
3          341
4          А. Кальчевский
5          12:30:00
6          13:50:00
7          seminar
8
9      [Line 46] session:
10         Математический анализ-2
11         S224
12         Д. Мухин
13         13:00:00
14         14:20:00
15         seminar
16     ...
17     [Line 145] session:
18         Машинное обучение
19         R307
20         С. Корпачев
21         14:40:00
22         16:00:00
23         seminar
```

d. //session[type="lecture" and classroom="R504"]

Результат:

```
1      [Line 64] session:
2          Макроэкономика-1
3          R504
4          В. Черноокый
5          11:10:00
6          12:30:00
7          lecture
8
9      [Line 91] session:
10         Микроэкономика-2
11         R504
12         О. Баранов
13         11:10:00
14         12:30:00
15         lecture
```

e. //session[type="seminar" and classroom="S224" and not(instructor=preceding-sibling:
:session[type="seminar" and classroom="S224"]/instructor)]/instructor

Результат:

```
1 [Line 49] instructor: Д. Мухин
2 [Line 140] instructor: А. Кальчевский
```

Пояснение: цель была выбрать всех преподавателей, проводящих семинары в аудитории S224, чтобы они не повторялись. Будем выбирать соответствующие занятия и отсекаать те, у которых преподаватель уже встречался в предыдущих элементах **session**, причем именно с типом **seminar** и аудиторией S224 (иначе мы рискуем отсечь лишних преподавателей). Далее из них извлечем имена преподавателей.

- f. Если предположить, что пары идут в хронологическом порядке, то запрос, например, для пятницы будет выглядеть так: `//session[@day="fri"] [last()]`.

Результат:

```
1 [Line 110] session:
2 Микроэкономика-2
3 S224
4 А. Кальчевский
5 16:20:00
6 17:40:00
7 seminar
```

Если же такого допущения не делать, то к сожалению я не придумал функцию, работающую в XPath 1.0. Время можно преобразовать в число с помощью `number(translate(start, ":", ""))` (например 12:30:00 превратится в 123000), это позволит нам сравнивать время начала пар. В теории можно выбрать только такие элементы **session**, что у них нет соседей с таким же атрибутом **day** и большим значением **start**. Но у меня так и не получилось написать рабочую функцию. В XPath 2.0 функция выглядит так (протестировал на онлайн-ресурсе <http://xpather.com/>):

```
//session[(@day="fri" and number(translate(start, ":", ""))
= max(//session[@day="fri"]/start/number(translate(., ":", ""))))]
```

Результат:

```
1 Микроэкономика-2 S224 А. Кальчевский 16:20:00 17:40:00 seminar
```

В данном случае у нас нет совпадающих последних пар, но если бы были, функция бы выбрала все.

- g. `count(//session)`

Результат:

```
1 13
```

Задание 3

Описать DTD схему для разработанного формата. Произвести валидацию xml-документа (1 балл).

У меня получилась следующая схема

```
1 <!ELEMENT timetable (session*)>
2 <!ELEMENT session (subject,classroom,instructor,start,end,type)>
3 <!ATTLIST session day (mon|tue|wed|thu|fri|sat|sun) #REQUIRED>
4 <!ELEMENT subject (#PCDATA)>
5 <!ELEMENT classroom (#PCDATA)>
6 <!ELEMENT instructor (#PCDATA)>
7 <!ELEMENT start (#PCDATA)>
8 <!ELEMENT end (#PCDATA)>
9 <!ELEMENT type (#PCDATA)>
```

Моя среда разработки автоматически валидировала документ при подключении DTD схемы (для сравнения показал документ, который валидацию не проходит)

При использовании DTD схемы нельзя вводить строгие ограничения на входные данные (удалось ограничить разве что значение атрибута day). Для решения этой проблемы лучше использовать XSD схему.

Задание 4

Описать XML Schema для разработанного формата. Произвести валидацию xml-документа. (1 балл)

Получилась следующая схема

```
1 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2
3 <xs:complexType name="Session">
4   <xs:all>
5     <xs:element name="subject" type="xs:string"/>
6     <xs:element name="classroom" type="xs:string"/>
7     <xs:element name="instructor" type="xs:string"/>
8     <xs:element name="start" type="xs:time"/>
9     <xs:element name="end" type="xs:time"/>
10    <xs:element name="type">
11      <xs:simpleType>
12        <xs:restriction base="xs:string">
13          <xs:enumeration value="lecture"/>
14          <xs:enumeration value="seminar"/>
15        </xs:restriction>
16      </xs:simpleType>
17    </xs:element>
18  </xs:all>
19  <xs:attribute name="day" use="required">
20    <xs:simpleType>
21      <xs:restriction base="xs:string">
22        <xs:pattern value="mon|tue|wed|thu|fri|sat|sun"/>
23      </xs:restriction>
24    </xs:simpleType>
25  </xs:attribute>
26 </xs:complexType>
27
28 <xs:element name="timetable">
29   <xs:complexType>
30     <xs:choice maxOccurs="unbounded">
31       <xs:element name="session" type="Session"/>
32     </xs:choice>
33   </xs:complexType>
34 </xs:element>
35
36 </xs:schema>
```

Как и в DTD схеме

1. Атрибут `day` может принимать значения `mon`, `tue`, `wed`, `thu`, `fri`, `sat`, `sun`
2. Единственным корневым элементом может быть элемент `timetable`

Но с помощью XSD удалось ввести дополнительные ограничения:

1. Значения элементов `start` и `end` должны иметь формат HH:MM:SS

2. Элемент `type` может принимать значение `lecture` или `seminar`
3. Элементы внутри `session`, как и сами элементы `session` внутри `timetable`, могут идти в любом порядке (в DTD схеме нельзя было менять порядок элементов внутри `session`).

Также автоматически произвелась валидация документа, справа показано, что работают некоторые из перечисленных ограничений

```
timetable.xml x scheme.dtd scheme.xsd
timetable.xml > timetable > session
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <timetable
4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5 xsi:noNamespaceSchemaLocation=
6 "C:\Users\Ibori\Desktop\Homeworks\sem3\internet_data_analysys\Lab2\scheme.xsd">
7
8   <session day="mon">
9     <subject>Микроэкономика-2</subject>
10    <classroom>341</classroom>
11    <instructor>А. Кальчевский</instructor>
12    <start>12:30:00</start>
13    <end>13:50:00</end>
14    <type>seminar</type>
15  </session>
16
17  <session day="tue">
18    <subject>Макроэкономика-1</subject>
19    <classroom>R503</classroom>
20    <instructor>В. Черноокый</instructor>
21    <start>13:00:00</start>
22    <end>14:20:00</end>
23    <type>lecture</type>
24  </session>
25
26  <session day="tue">
27    <subject>Машинное обучение</subject>
28    <classroom>R205</classroom>
29    <instructor>М. Гуцин</instructor>
30    <start>16:20:00</start>
31    <end>17:40:00</end>
32    <type>lecture</type>
33  </session>
34</timetable>
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

No problems have been detected in the workspace.

```
timetable.xml 6 scheme.dtd scheme.xsd
timetable.xml > timetable > session
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <timetable
4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5 xsi:noNamespaceSchemaLocation=
6 "C:\Users\Ibori\Desktop\Homeworks\sem3\internet_data_analysys\Lab2\scheme.xsd">
7
8   <session day="pon">
9     <subject>Микроэкономика-2</subject>
10    <classroom>341</classroom>
11    <instructor>А. Кальчевский</instructor>
12    <start>twelve</start>
13    <end>13:50:00</end>
14    <type>memoire</type>
15  </session>
16
17  <session day="tue">
18    <subject>Макроэкономика-1</subject>
19    <classroom>R503</classroom>
20    <instructor>В. Черноокый</instructor>
21    <start>13:00:00</start>
22    <end>14:20:00</end>
23    <type>lecture</type>
24  </session>
25
26  <session day="tue">
27    <subject>Машинное обучение</subject>
28    <classroom>R205</classroom>
29    <instructor>М. Гуцин</instructor>
30    <start>16:20:00</start>
31    <end>17:40:00</end>
32    <type>lecture</type>
33  </session>
34</timetable>
```

PROBLEMS 6 OUTPUT TERMINAL PORTS DEBUG CONSOLE

timetable.xml 6

- cvc-pattern-valid: Value 'pon' is not facet-valid with respect to pattern 'mon[tue]wed[thu]fri[sat]sun' for type '#AnonType_0'
- cvc-attribute.3: The value 'pon' of attribute 'day' on element 'session' is not valid with respect to its type, '#AnonType_0'
- Content of type 'time' is expected. xml(cvc-datatype-valid.1.2.1) [Ln 12, Col 16] ^

The following content is not a valid type:
'twelve'

Задание 5

Описать XSLT-преобразование xml-документа в текстовый вид (*.txt). (1 балл)

У меня получилось следующее XSLT-преобразование

```
1 <xsl:stylesheet version="1.0"
2 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4   <xsl:output version="1.0" encoding="UTF-8" indent="yes" method="text"/>
5
6   <xsl:template match="timetable">
7     <xsl:variable name="sortOrder" select="' mon tue wed thu fri sat sun '"/>
8     <xsl:for-each select="//session[not(@day=preceding::session/@day)]/@day">
9       <xsl:sort data-type="number" select="string-length(
10         substring-before($sortOrder, concat(' ', self::node(), ' ')))"/>
11       <xsl:variable name="DAY" select="self::node()"/>
12       <xsl:if test="$DAY='mon'">&#xa;Понедельник:&#xa;</xsl:if>
13       <xsl:if test="$DAY='tue'">&#xa;Вторник:&#xa;</xsl:if>
14       <xsl:if test="$DAY='wed'">&#xa;Среда:&#xa;</xsl:if>
15       <xsl:if test="$DAY='thu'">&#xa;Четверг:&#xa;</xsl:if>
16       <xsl:if test="$DAY='fri'">&#xa;Пятница:&#xa;</xsl:if>
17       <xsl:if test="$DAY='sat'">&#xa;Суббота:&#xa;</xsl:if>
18       <xsl:if test="$DAY='sun'">&#xa;Воскресенье:&#xa;</xsl:if>
19       <xsl:for-each select="//session[@day=$DAY]">
20         <xsl:sort select="start"/>
21         <xsl:text>&#xa;&#09;</xsl:text>
22         <xsl:value-of select="subject"/>
23         <xsl:text>&#xa;&#09;&#09;</xsl:text>
24         <xsl:value-of select="concat('Аудитория: ', classroom)"/>
25         <xsl:text>&#xa;&#09;&#09;</xsl:text>
26         <xsl:value-of select="concat('Преподаватель: ', instructor)"/>
27         <xsl:text>&#xa;&#09;&#09;</xsl:text>
28         <xsl:value-of select="concat(start, '-', end)"/>
29         <xsl:text>&#xa;&#09;&#09;</xsl:text>
30         <xsl:if test="type='lecture'">Лекция</xsl:if>
31         <xsl:if test="type='seminar'">Семинар</xsl:if>
32         <xsl:text>&#xa;</xsl:text>
33       </xsl:for-each>
34     </xsl:for-each>
35   </xsl:template>
36
37 </xsl:stylesheet>
```

Логика:

1. Сначала перебираются все различные значения атрибутов day в отсортированном порядке (трюк для сортировки позаимствовал [здесь](#) – с помощью переменной sortOrder и функции substring-before выбирается строка, длина которой тем больше, чем дальше значение атрибута в переменной sortOrder, сортировка производится по длине этой строки).

2. Далее значение атрибута записывается в переменную DAY, выводится соответствующий день недели, а затем выбираются элементы `session` с совпадающим атрибутом, сортируются по времени начала, выводится информация о них (таким образом, в итоговом файле пропущен четверг, так как в этот день нет пар).

Результат:

```
1 Понедельник:
2
3     Микроэкономика-2
4         Аудитория: 341
5         Преподаватель: А. Кальчевский
6         12:30:00-13:50:00
7         Семинар
8
9 Вторник:
10
11     Макроэкономика-1
12         Аудитория: R503
13         Преподаватель: В. Черноокий
14         13:00:00-14:20:00
15         Лекция
16
17     Машинное обучение
18         Аудитория: R205
19         Преподаватель: М. Гущин
20         16:20:00-17:40:00
21         Лекция
22     ...
23     Математический анализ-2
24         Аудитория: S224
25         Преподаватель: Д. Мухин
26         14:40:00-16:00:00
27         Семинар
28
29     Микроэкономика-2
30         Аудитория: S224
31         Преподаватель: А. Кальчевский
32         16:20:00-17:40:00
33         Семинар
34
35 Суббота:
36
37     Машинное обучение
38         Аудитория: R307
39         Преподаватель: С. Корпачев
40         14:40:00-16:00:00
41         Семинар
```

Задание 6

Описать XSLT-преобразование xml-документа в html-страницу (расписание должно быть представлено в виде таблицы) (1 балл)

У нас нет групп и у всех разные курсы, поэтому в XML-документе могут встречаться пересекающиеся пары. В таком случае нужно будет как-то высчитывать максимальное количество одновременных пар и на основе этого считать ширину столбца, что довольно сложно. Поэтому я просто оставил пары, которые я посещаю.

После долгих и мучительных попыток получилось следующее преобразование:

```
1 <xsl:stylesheet version="1.0"
2 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4   <xsl:output version="1.0" encoding="UTF-8" indent="yes" method="html"/>
5   <xsl:template match="timetable">
6     <html>
7       <h1>Расписание</h1>
8       <body>
9         <table bgcolor="#CACCE">
10           <thead>
11             <tr>
12               <th width="200px" bgcolor="#86888A">Понедельник</th>
13               <th width="200px" bgcolor="#86888A">Вторник</th>
14               <th width="200px" bgcolor="#86888A">Среда</th>
15               <th width="200px" bgcolor="#86888A">Четверг</th>
16               <th width="200px" bgcolor="#86888A">Пятница</th>
17               <th width="200px" bgcolor="#86888A">Суббота</th>
18               <th width="200px" bgcolor="#86888A">Воскресенье</th>
19             </tr>
20           </thead>
21           <tbody>
22             <xsl:for-each select="(//node())[60 >= position()]">
23               <tr>
24                 <xsl:variable name="currTime" select="31800+600*position()"/>
25                 <xsl:variable name="currTimeAsNum"
26                   select="
27                     floor($currTime div 3600)*10000+
28                     floor($currTime mod 3600 div 60)*100+
29                     $currTime mod 60"/>
30                 <xsl:call-template name="fillCells">
31                   <xsl:with-param name="DAY" select="'mon'"/>
32                   <xsl:with-param name="currTimeAsNum" select="$currTimeAsNum"/>
33                 </xsl:call-template>
34                 <xsl:call-template name="fillCells">
35                   <xsl:with-param name="DAY" select="'tue'"/>
36                   <xsl:with-param name="currTimeAsNum" select="$currTimeAsNum"/>
37                 </xsl:call-template>
38                 <xsl:call-template name="fillCells">
39                   <xsl:with-param name="DAY" select="'wed'"/>
40                   <xsl:with-param name="currTimeAsNum" select="$currTimeAsNum"/>
41                 </xsl:call-template>
42                 <xsl:call-template name="fillCells">
43                   <xsl:with-param name="DAY" select="'thu'"/>
44                   <xsl:with-param name="currTimeAsNum" select="$currTimeAsNum"/>
45                 </xsl:call-template>
46                 <xsl:call-template name="fillCells">
47                   <xsl:with-param name="DAY" select="'fri'"/>
```

```

48         <xsl:with-param name="currTimeAsNum" select="$currTimeAsNum"/>
49     </xsl:call-template>
50     <xsl:call-template name="fillCells">
51         <xsl:with-param name="DAY" select="'sat'"/>
52         <xsl:with-param name="currTimeAsNum" select="$currTimeAsNum"/>
53     </xsl:call-template>
54     <xsl:call-template name="fillCells">
55         <xsl:with-param name="DAY" select="'sun'"/>
56         <xsl:with-param name="currTimeAsNum" select="$currTimeAsNum"/>
57     </xsl:call-template>
58 </tr>
59 </xsl:for-each>
60 </tbody>
61 </table>
62 </body>
63 </html>
64 </xsl:template>
65
66 <xsl:template name="fillCells">
67     <xsl:param name="DAY"/>
68     <xsl:param name="currTimeAsNum"/>
69     <xsl:for-each select="//session[@day=$DAY]">
70         <xsl:variable name="startAsNum" select="number(translate(start, ':', ''))"/>
71         <xsl:variable name="endAsNum" select="number(translate(end, ':', ''))"/>
72         <xsl:if test="$startAsNum=$currTimeAsNum">
73             <td bgcolor="#89CFEF" rowspan="8">
74                 <xsl:value-of select="subject"/><br/>
75                 <xsl:value-of select="classroom"/><br/>
76                 <xsl:value-of select="start"/>-<xsl:value-of select="end"/>
77             </td>
78         </xsl:if>
79     </xsl:for-each>
80     <xsl:if test="count(//session[@day=$DAY and number(translate(start, ':', '')) &lt;=
81 $currTimeAsNum and number(translate(end, ':', '')) &gt;= $currTimeAsNum])=0">
82         <td bgcolor="#CACCCE"></td>
83     </xsl:if>
84 </xsl:template>
85
86 </xsl:stylesheet>

```

Логика:

1. Создается таблица и её заголовок. Ячейка таблицы – 10-минутный интервал. Если в какие-то промежутки времени есть пара, соответствующие ячейки должны объединиться и в получившейся ячейке должна быть отображена информация о занятии.
2. Далее перебираются все промежутки времени с 9:00 до 18:50. Для каждого промежутка и каждого дня вызывается 7 шаблонов `fillCells` для каждого дня с параметрами `DAY` и `currTimeAsNum` (время в виде числа ННММДД).
3. Шаблон `fillCells` ищет занятия, которые начинаются в переданное время и день. Если такое находится, объединяются и заполняются 8 ячеек в столбце (т.к. пара длится 80 минут). Если же такого не нашлось, нужно вставить пустую ячейку, однако в XLST нельзя реализовать логический флаг. Поэтому запускается проверка на наличие пары, которая идет в данный промежуток времени. Если такой нет, вставляем пустую ячейку, если есть – значит ячейка уже заполнена информацией об этой паре.
4. В общем, программа идет сначала по строкам, затем по столбцам, и создает либо ячейки с информацией о парах, либо пустые (еще и закрашивает их по разному).

Результат:

Расписание

Понедельник	Вторник	Среда	Четверг	Пятница	Суббота	Воскресенье
		Математический анализ-2 R503 09:30:00-10:50:00		Математический анализ-2 R503 09:30:00-10:50:00		
		Макроэкономика-1 R504 11:10:00-12:30:00		Макроэкономика-2 R504 11:10:00-12:30:00		
Микроэкономика-2 341 12:30:00-13:50:00	Макроэкономика-1 R503 13:00:00-14:20:00	Математический анализ-2 G119 13:00:00-14:20:00		Технологии анализа данных в сети Интернет online 13:00:00-14:20:00		
		Машинное обучение R506 14:40:00-16:00:00		Технологии анализа данных в сети Интернет online 14:40:00-16:00:00	Машинное обучение R307 14:40:00-16:00:00	
	Машинное обучение R205 16:20:00-17:40:00			Микроэкономика-2 S224 16:20:00-17:40:00		

Конечно, процесс довольно кривой и сломается, если появятся пары, идущие не 80 минут или начинающиеся не в заданные 10-минутные промежутки времени. Но при желании это можно исправить, уменьшив интервал (например до 1 минуты) и каждый раз вычисляя необходимое количество ячеек для объединения. Однако если появятся пересекающиеся пары, придется как-то дополнительно расширять таблицу, что уже довольно сложно.

Задание 7

Разработать программу на любом языке программирования (Python, C, Java) для выполнения XSLT-преобразования и разработанных XPath-запросов. (1 балл)

Программу можно найти в файле `runner.ipynb`

```
1 import lxml.etree as ET
2
3 #XSLT
4 xmlfile = ET.parse(r"timetable.xml")
5 xslt_to_text = ET.parse(r"totext.xslt")
6 xslt_to_html = ET.parse(r"tohtml.xslt")
7
8 transform1 = ET.XSLT(xslt_to_text)
9 output = transform1(xmlfile)
10 with open("output_text.txt", "w", encoding="UTF-8") as f:
11     f.write(str(output))
12
13 transform2 = ET.XSLT(xslt_to_html)
14 output = transform2(xmlfile)
15 with open("output_html.html", "w", encoding="UTF-8") as f:
16     f.write(str(output))
17
18 #XPath
19 path1 = xmlfile.xpath("//session")
20 for el in path1:
21     print(ET.tostring(el).decode())
22
23 path2 = xmlfile.xpath("/timetable/session[not(classroom=\
24     preceding-sibling::session/classroom)]/classroom")
25 for el in path2:
26     print(el.text)
27
28 path3 = xmlfile.xpath("//session[type='seminar']")
29 for el in path3:
30     print(ET.tostring(el).decode())
31
32 path4 = xmlfile.xpath("//session[type='lecture' and classroom='R504']")
33 for el in path4:
34     print(ET.tostring(el).decode())
35
36 path5 = xmlfile.xpath('//session[type="seminar" and classroom="S224" and \
37     not(instructor=preceding-sibling::session[type="seminar" and \
38     classroom="S224"]/instructor)]/instructor')
39 for el in path5:
40     print(el.text)
41
42 path7 = xmlfile.xpath('count(//session)')
43 print(path7)
```

В результате в файлах `output_text.txt` и `output_html.html` появляются преобразованные документы, а на экран выводятся результаты XPath запросов (кроме 6-ого).

Задание 8

Решить аналогичные задачи для формата JSON (2 балла).

С JSON получился немного другой формат.

```
1 {
2   "timetable": {
3     "sessions": [
4       {
5         "day": "mon",
6         "subject": "Микроэкономика-2",
7         "classroom": 341,
8         "instructor": "А. Кальчевский",
9         "start": "12:30:00",
10        "end": "13:50:00",
11        "type": "seminar"
12      },
13      {
14        "day": "tue",
15        "subject": "Макроэкономика-1",
16        "classroom": "R503",
17        "instructor": "В. Черноокий",
18        "start": "13:00:00",
19        "end": "14:20:00",
20        "type": "lecture"
21      },
22      ...
23      {
24        "day": "sat",
25        "subject": "Машинное обучение",
26        "classroom": "R307",
27        "instructor": "С. Корпачев",
28        "start": "14:40:00",
29        "end": "16:00:00",
30        "type": "seminar"
31      }
32    ]
33  }
34 }
```

Сразу же составим JSON Schema и проведем валидацию.

```
1 {
2   "$schema": "http://json-schema.org/draft-04/schema#",
3   "type": "object",
4   "properties": {
5     "timetable": {
6       "type": "object",
7       "properties": {
8         "sessions": {
9           "type": "array",
10          "items": [
```

```

11     {
12         "type": "object",
13         "properties": {
14             "day": {
15                 "type": "string",
16                 "enum" : ["mon", "tue", "wed", "thu", "fri", "sat", "sun"]
17             },
18             "subject": {
19                 "type": "string"
20             },
21             "classroom": {
22                 "type": "integer"
23             },
24             "instructor": {
25                 "type": "string"
26             },
27             "start": {
28                 "type": "string",
29                 "pattern": "^(?:[01]\\d|2[0123]):(?:[012345]\\d):(?:[012345]\\d)$"
30             },
31             "end": {
32                 "type": "string",
33                 "pattern": "^(?:[01]\\d|2[0123]):(?:[012345]\\d):(?:[012345]\\d)$"
34             },
35             "type": {
36                 "type": "string",
37                 "enum" : ["lecture", "seminar"]
38             }
39         },
40         "required": [
41             "day",
42             "subject",
43             "classroom",
44             "instructor",
45             "start",
46             "end",
47             "type"
48         ]
49     }
50 ]
51 }
52 },
53 "required": [
54     "sessions"
55 ]
56 }
57 },
58 "required": [
59     "timetable"
60 ]
61 }

```

Здесь множество значений элементов `day` и `type` ограничены с помощью `"enum"`, а формат `start` и `end` – с помощью регулярного выражения.

Валидацию произвел на ресурсе <https://www.jsonschemavalidator.net/>

The screenshot shows the JSON Schema Validator interface. On the left, the 'Select schema' dropdown is set to 'Custom', and a JSON schema is displayed. The schema defines a 'timetable' object with a 'sessions' array. Each session has properties: 'day' (enum: 'fri', 'sat'), 'subject' (string), 'classroom' (string), 'instructor' (string), 'start' (string), 'end' (string), and 'type' (enum: 'lecture', 'seminar'). On the right, the 'Input JSON' field contains a JSON array of session objects. The status bar at the bottom indicates 'No errors found. JSON validates against the schema'.

```
34 {
35   "type": {
36     "type": "string",
37     "enum": ["lecture", "seminar"]
38   },
39   "required": [
40     "day",
41     "subject",
42     "classroom",
43     "instructor",
44     "start",
45     "end",
46     "type"
47   ]
48 }
49 }
50 }
51 }
52 }
53 "required": [
54   "sessions"
55 ]
56 }
57 }
58 "required": [
59   "timetable"
60 ]
61 }
```

```
95 {
96   "subject": "Технологии анализа данных в сети Интернет",
97   "classroom": "online",
98   "instructor": "В. Ланин",
99   "start": "14:40:00",
100  "end": "16:00:00",
101  "type": "seminar"
102 }
103 }
104 {
105   "day": "fri",
106   "subject": "Микроэкономика-2",
107   "classroom": "S224",
108   "instructor": "А. Кальчевский",
109   "start": "16:20:00",
110   "end": "17:40:00",
111   "type": "seminar"
112 }
113 }
114 {
115   "day": "sat",
116   "subject": "Машинное обучение",
117   "classroom": "R307",
118   "instructor": "С. Корпачев",
119   "start": "14:40:00",
120   "end": "16:00:00",
121   "type": "seminar"
122 }
123 }
```

✓ No errors found. JSON validates against the schema

JSONPath-запросы:

- `$.timetable.sessions`
- `$.timetable.sessions[*].classroom` – выводит все аудитории, а не только уникальные, в JSONPath уникальные значения можно отобразить лишь с помощью дальнейшей обработки
- `$.timetable.sessions[?(@.type=="seminar")]`
- `$.timetable.sessions[?(@.type=="lecture" & @.classroom=="R504")]`
- `$.timetable.sessions[?(@.type=="seminar" & @.classroom=="S224")].instructor` – опять же, выбираются даже повторяющиеся преподаватели
- Этот запрос невозможно осуществить в JSONPath, ведь необходимо отфильтровать пары по дням, а затем выбрать последнюю пару. Не предполагается, что будет возможно совмещать фильтры и доступ по индексу, о чем пишут на официальной странице репозитория JSONPath (<https://github.com/json-path/JsonPath/issues/272>).

Возможно, сработает `$.timetable.sessions[?(@.day == "fri" && @.start == $.timetable.sessions[*].start.max())]`, однако я не нашел онлайн-тестировщик или библиотеку, поддерживающий функцию `max`.

В любом случае, последний элемент выборки можно будет легко выбрать в любой программе, где используется JSONPath (как показано в файле `runner.ipynb`).

- Тоже невозможно сделать в JSONPath. Альтернативный вариант также в файле `runner.ipynb`, как и тесты предыдущих JSONPath-запросов.

Чтобы выполнить XSLT-преобразования, можно преобразовать JSON-файл в формат XML и использовать слегка модифицированные преобразования, которые уже имеются.

Код для преобразования на Python:

```
1 import dicttoxml
2
3 with open("timetable_fromjson.xml", "w", encoding="UTF-8") as f:
4     f.write(dicttoxml.dicttoxml(jsonfile, attr_type=False).decode("UTF-8"))
5
6 xmlfile_fromjson = ET.parse("timetable_fromjson.xml")
7
8 xslt_to_text = ET.parse(r"totext_jsonmod.xslt")
9 xslt_to_html = ET.parse(r"totext_jsonmod.xslt")
10
11 transform1 = ET.XSLT(xslt_to_text)
12 output = transform1(xmlfile_fromjson)
13 with open("output_text2.txt", "w", encoding="UTF-8") as f:
14     f.write(str(output))
15
16 transform2 = ET.XSLT(xslt_to_html)
17 output = transform2(xmlfile_fromjson)
18 with open("output_html2.html", "w", encoding="UTF-8") as f:
19     f.write(str(output))
```

С помощью библиотеки dicttoxml можно преобразовать Python-словарь в XML. Автоматически корневой элемент называется root, а данные о сессиях хранятся в массиве sessions с тегом item. Поэтому в файлах totext_jsonmod.xslt и totext_jsonmod.xslt я лишь заменил timetable на root, session на item и @day на day. В файлах output_text2.txt и output_html2.html можно найти результат преобразований (результат совпадает).