

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий

Информатика

Лабораторная работа № 3

Выполнил студент

Шнейдерис Герардас

Группа № Р3120

Преподаватель: Болдырева Елена Александровна

г. Санкт-Петербург

2023

Лабораторная работа № 3	1
Варианты:	3
Задание:	3
Отчет:	4
Задание 1	4
Задание 2	4
Задание 3	5
Задание 3.1	6
Задание 3.2	7
Вывод:	8
Список литературы:	8

Варианты:

- I. 024**
- II. 2**
- III. 14**

Задание:

1. Реализуйте программный продукт на языке Python. Программа должна считать количество смайликов определённого вида. Для своей программы придумайте минимум 5 тестов. Каждый тест является отдельной сущностью, передаваемой регулярному выражению для обработки. Для каждого теста необходимо самостоятельно (без использования регулярных выражений) найти правильный ответ. После чего сравнить ответ, выданный программой, и полученный самостоятельно.
2. Реализуйте программный продукт на языке Python, используя регулярные выражения. Для своей программы придумайте минимум 5 тестов. Протестируйте свою программу на этих тестах.
3. Написать программу на языке Python 3.x, которая бы осуществляла парсинг и конвертацию исходного XML файла в JSON.
 - 3.1. Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать. Сравнить полученные результаты и объяснить их сходство/различие.
 - 3.2. Переписать исходный код, добавив в него использование регулярных выражений. Сравнить полученные результаты и объяснить их сходство/различие.

Отчет:

Задание 1

```
№1.py x №2.py x №3.py x №3_libs.py x №3_re.py x
1  # emoji :-{\
2      # answers for tests:
3      #   1   2   3   4   5
4  #   40  14   8  22  22
5      import re
6
7      text = open(r"Tests_for_first_task/input_1_1.txt", encoding='utf-8').read() # читаем с файла
8      re_emoji = r':-{\{' # шаблон для поиска эмодзи
9      print(len(re.findall(re_emoji, text))) # находим длину объекта содержащего все найденные совпадения
10
```

скриншот программы для 1-го задания

Задание 2

```
№1.py x №2.py x №3.py x №3_libs.py x №3_re.py x
1  import re
2
3  text = [x for x in open(r"Tests_for_second_task/input_2_5.txt", encoding='utf-8')] # записываем файл в массив строк
4  res = []
5  my_group = r"\bP3120\b" # задаем номер своей группы
6  for s in text:
7      if re.findall(r'[A-ZА-ЯЁ]\.', s)[0] == re.findall(r'[A-ZА-ЯЁ]\.', s)[1]: # условие на инициалы вида А.А.
8          if re.search(my_group, s): # проверяем из нашей ли группы человек
9              continue # этих людей НЕ записываем в ответ
10         res.append(s) # всех остальных записываем
11     print(*res, sep='')
12
```

скриншот программы для 2-го задания

Задание 3

```
№1.py x №2.py x №3.py x №3_libs.py x №3_re.py x
1 schedule_xml = open(r"schedule.xml", encoding='utf-8').read().split('\n') # открываем xml файл для чтения
2 schedule_json = open("schedule.json", 'w', encoding='utf-8') # открываем json файл для записи
3 schedule_json.write('\n') # расписание - массив уроков, поэтому начнем и закончим файл квадратной скобкой
4 schedule_xml.pop(0) # первая строка xml файла - корневой тег, удалим за ненадобностью
5 for word in schedule_xml:
6     # находим и удаляем закрывающие теги
7     if '/' in word:
8         word = word[:word.find('/') - 1]
9     # меняем xml теги на json теги
10    word = word.replace('<', '').replace('>', '')
11    # ставим фигурную скобку в начале блока
12    if len(word) > 0 and word[-1] == ':': #
13        word = '\t{'
14    # помещаем текст в тегах в кавычки
15    if ':' in word and len(word[word.find(':') + 1:]) > 0:
16        word = word[:word.find(':') + 1] + ' ' + word[word.find(':') + 1:] + ','
17    # убираем запятую и ставим фигурную скобку в конце блока
18    if len(word) > 0 and word[-1] == ':':
19        schedule_json.seek(schedule_json.tell() - 3)
20        schedule_json.write('\n')
21        word += '},'
22    schedule_json.write(word + '\n') # записываем отформатированную строку в файл
23 schedule_json.seek(schedule_json.tell() - 5) # убираем последнюю запятую
24 schedule_json.write('\n') # конец массива
25
```

скриншот программы для 3-го задания

<pre>№1.py x №2.py x №3.py x №3_libs.py x №3_re.py x sch 1 <root> 2 <lesson> 3 <module>Программирование</module> 4 <teacher>Письмак Алексей Евгеньевич</teacher> 5 <auditory>Актный зал</auditory> 6 <campus>Ломоносова д. 9</campus> 7 <start_time>08:20</start_time> 8 </lesson> 9 <lesson> 10 <module>Иностранный язык</module> 11 <teacher>Филатов Андрей Сергеевич</teacher> 12 <auditory>3211</auditory> 13 <campus>Ломоносова д. 9</campus> 14 <start_time>11:40</start_time> 15 </lesson> 16 <lesson> 17 <module>Иностранный язык</module> 18 <teacher>Филатов Андрей Сергеевич</teacher> 19 <auditory>3211</auditory> 20 <campus>Ломоносова д. 9</campus> 21 <start_time>13:30</start_time> 22 </lesson> 23 </root></pre>	<pre>1 [2 { 3 "module": "Программирование", 4 "teacher": "Письмак Алексей Евгеньевич", 5 "auditory": "Актный зал", 6 "campus": "Ломоносова д. 9", 7 "start_time": "08:20" 8 }, 9 { 10 "module": "Иностранный язык", 11 "teacher": "Филатов Андрей Сергеевич", 12 "auditory": "3211", 13 "campus": "Ломоносова д. 9", 14 "start_time": "11:40" 15 }, 16 { 17 "module": "Иностранный язык", 18 "teacher": "Филатов Андрей Сергеевич", 19 "auditory": "3211", 20 "campus": "Ломоносова д. 9", 21 "start_time": "13:30" 22 } 23]</pre>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

исходный файл XML(слева) и получившийся JSON(справа)

Задание 3.1

```
Nº1.py × Nº2.py × Nº3.py × Nº3_libs.py × Nº3_re.py ×
1 import xmltodict
2 import json
3
4
5 def translation(s): # функция перевода для корректного отображения кириллицы
6     dict_utf = {
7         'u00410': 'А', 'u00430': 'а',
8         'u00411': 'Б', 'u00431': 'б',
9         'u00412': 'В', 'u00432': 'в',
10        'u00413': 'Г', 'u00433': 'г',
11        'u00414': 'Д', 'u00434': 'д',
12        'u00415': 'Е', 'u00435': 'е',
13        'u00401': 'Ё', 'u00451': 'ё',
14        'u00416': 'Ж', 'u00436': 'ж',
15        'u00417': 'З', 'u00437': 'з',
16        'u00418': 'И', 'u00438': 'и',
17        'u00419': 'Й', 'u00439': 'й',
18        'u0041a': 'К', 'u0043a': 'к',
19        'u0041b': 'Л', 'u0043b': 'л',
20        'u0041c': 'М', 'u0043c': 'м',
21        'u0041d': 'Н', 'u0043d': 'н',
22        'u0041e': 'О', 'u0043e': 'о',
23
24        'u0041f': 'П', 'u0043f': 'п',
25        'u00420': 'Р', 'u00440': 'р',
26        'u00421': 'С', 'u00441': 'с',
27        'u00422': 'Т', 'u00442': 'т',
28        'u00423': 'У', 'u00443': 'у',
29        'u00424': 'Ф', 'u00444': 'ф',
30        'u00425': 'Х', 'u00445': 'х',
31        'u00426': 'Ц', 'u00446': 'ц',
32        'u00427': 'Ч', 'u00447': 'ч',
33        'u00428': 'Ш', 'u00448': 'ш',
34        'u00429': 'Щ', 'u00449': 'щ',
35        'u0042a': 'Ъ', 'u0044a': 'ъ',
36        'u0042b': 'Ы', 'u0044b': 'ы',
37        'u0042c': 'Ь', 'u0044c': 'ь',
38        'u0042d': 'Э', 'u0044d': 'э',
39        'u0042e': 'Ю', 'u0044e': 'ю',
40        'u0042f': 'Я', 'u0044f': 'я',
41    }
42    for k, v in dict_utf.items():
43        s = s.replace('\\" + k, v)
44    return s
45
46 dict_from_xml = xmltodict.parse(open(r"schedule.xml", encoding='utf-8').read()) # переводим xml-файл в словарь
47 json_from_dict = json.dumps(dict_from_xml, indent=4) # делаем строку формата json из словаря
48
49 open("schedule.json", 'w', encoding='utf-8').write(translation(json_from_dict))
50
```

скриншот программы для дополнительного задания 3.1

```

1  {
2  "root": {
3    "lesson": [
4      {
5        "module": "Программирование",
6        "teacher": "Письмак Алексей Евгеньевич",
7        "auditory": "Актный зал",
8        "campus": "Ломоносова д. 9",
9        "start_time": "08:20"
10     },
11     {
12       "module": "Иностранный язык",
13       "teacher": "Филатов Андрей Сергеевич",
14       "auditory": "3211",
15       "campus": "Ломоносова д. 9",
16       "start_time": "11:40"
17     },
18     {
19       "module": "Иностранный язык",
20       "teacher": "Филатов Андрей Сергеевич",
21       "auditory": "3211",
22       "campus": "Ломоносова д. 9",
23       "start_time": "13:30"
24     }
25   ]
26 }
27 }

```

JSON файл получившийся с использованием сторонних библиотек xmldict и json. Заметим, что файл получился более полным, сохранились все теги, но структура немного нарушена(в теге lesson данные сразу обо всех занятиях)

Задание 3.2

```
1 import re
2
3 schedule_xml = open(r"schedule.xml", encoding='utf-8').read() # открываем xml файл для чтения
4 schedule_json = open("schedule.json", 'w', encoding='utf-8') # открываем json файл для записи
5
6 re_tag_with_text_xml = r'<(\w+)>([\w \d.:!?]+)</\w+>' # шаблон для однострочных тегов с текстом
7 schedule_xml = re.sub(re_tag_with_text_xml, r'"1": "2"', schedule_xml) # меняем формат однострочных тегов на JSON
8
9 re_tag_with_text_json = r'"(\w+)\": \ "[\w \d.:!?]+\ "' # шаблон для поиска новых однострочных тегов
10 # шаблон для поиска тегов внутри, которых массив тегов
11 re_tag_with_array_of_tags = r'(\s+)<\w+>(\s+(?:' + re_tag_with_text_json + r'\s+)+)</\w+>'
12 schedule_xml = re.sub(re_tag_with_array_of_tags, r'\1{\2}', schedule_xml) # превращаем эти массивы в JSON объекты
13
14 schedule_xml = re.sub(r'"(\s+)"', r'",\1', schedule_xml) # расставляем запятые между однострочными тегами
15 schedule_xml = re.sub(r'(\s+)\}', r'},\1', schedule_xml) # расставляем запятые между объектами
16
17 # шаблон для поиска новых массивов
18 re_tag_with_array_of_json_objects = r'<(\w+)>(\s+({[\w\":,.\s]+},\s+)+){[\w\":,.\s]+}\s+</\w+>'
19 schedule_xml = re.sub(re_tag_with_array_of_json_objects, r'"1": [\2]', schedule_xml)
20
21 schedule_json.write('\n') # JSON файл должен начинаться и заканчиваться фигурными скобками
22 for s in schedule_xml.split('\n'): # добавим табуляцию к каждой строчке
23     schedule_json.write('\t' + s + '\n')
24 schedule_json.write('}')
25
```

скриншот программы для дополнительного задания 3.2

```
{
  "root": [
    {
      "module": "Программирование",
      "teacher": "Письмак Алексей Евгеньевич",
      "auditory": "Актный зал",
      "campus": "Ломоносова д. 9",
      "start_time": "08:20"
    },
    {
      "module": "Иностранный язык",
      "teacher": "Филатов Андрей Сергеевич",
      "auditory": "3211",
      "campus": "Ломоносова д. 9",
      "start_time": "11:40"
    },
    {
      "module": "Иностранный язык",
      "teacher": "Филатов Андрей Сергеевич",
      "auditory": "3211",
      "campus": "Ломоносова д. 9",
      "start_time": "13:30"
    }
  ]
}
```

JSON файл получившийся с использованием регулярных выражений, он полнее чем результат программы из задания 3, но зато логика сохранилась, хотя от повторяющихся тегов пришлось отказаться.

Вывод:

Я изучил синтаксис регулярных выражений, Python библиотеку для работы с ними, а также области, где их стоит и не стоит применять. Также я изучил форму Бэкуса-Наура и ее расширенную версию, синтаксис языков разметки XML и JSON. Попрактиковался в конвертации из одного формата в другой разными способами.

Список литературы:

[Регулярные выражения в Python / Хабр](#)

[Методичка.pdf](#)

[Информатика лекции - Overleaf](#)

[Что такое JSON / Хабр](#)

[Что такое XML / Хабр](#)

[JSON](#)

[regex101](#)

[Форма Бэкуса — Наура — Википедия](#)

[Расширенная форма Бэкуса — Наура — Википедия](#)