НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий

Информатика

Лабораторная работа № 3

Выполнил студент

Шнейдерис Герардас

Группа № P3120

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:

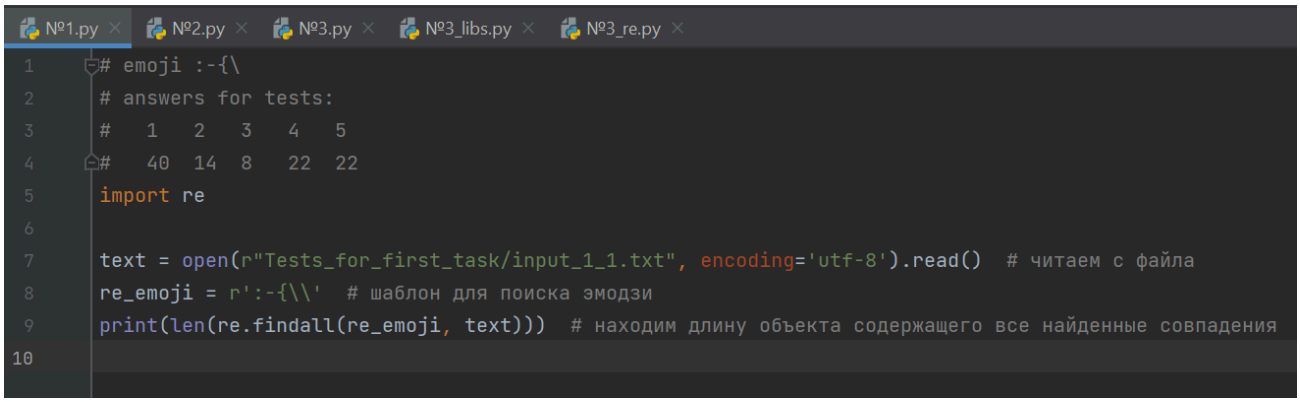
I. 024
II. 2
III. 14

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.

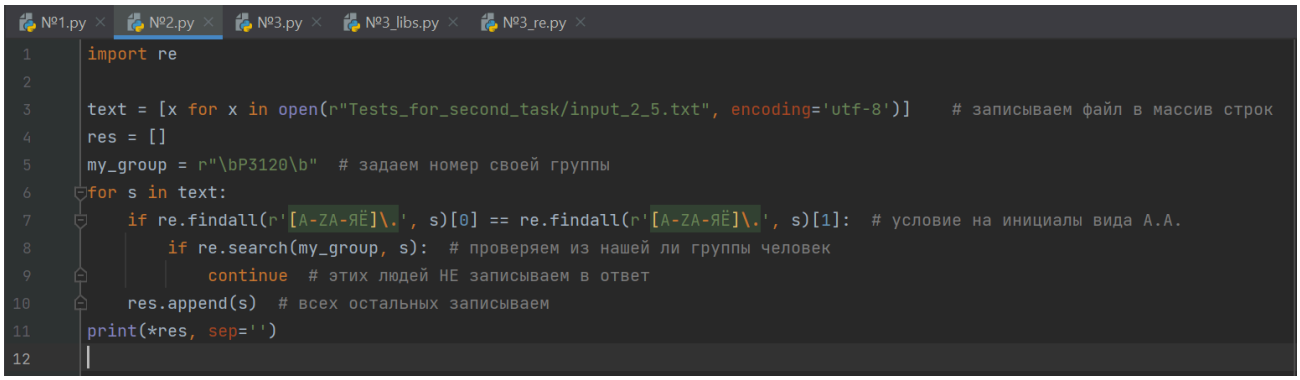
Отчет:

Задание 1



скриншот программы для 1-го задания

Задание 2



скриншот программы для 2-го задания

Задание 3

```python
schedule_xml = open(r"schedule.xml", encoding='utf-8').read().split('\n')  # открываем xml файл для чтения
schedule_json = open("schedule.json", 'w', encoding='utf-8')  # открываем json файл для записи
schedule_json.write('[\n')  # расписание - массив уроков, поэтому начнем и закончим файл квадратной скобкой
schedule_xml.pop(0)  # первая строка xml файла - корневой тег, удалим за ненадобностью
for word in schedule_xml:
    # находим и удаляем закрывающие теги
    if '/' in word:
        word = word[:word.find('/') - 1]
    # меняем xml теги на json теги
    word = word.replace('<', '"').replace('>', '":')
    # ставим фигурную скобку в начале блока
    if len(word) > 0 and word[-1] == ':':  #
        word = '\t{'
    # помещаем текст в тегах в кавычки
    if ':' in word and len(word[word.find(':') + 1:]) > 0:
        word = word[:word.find(':') + 1] + ' "' + word[word.find(':') + 1:] + '",'
    # убираем запятую и ставим фигурную скобку в конце блока
    if len(word) > 0 and word[-1] == ' ':
        schedule_json.seek(schedule_json.tell() - 3)
        schedule_json.write('\n')
        word += '},'
    schedule_json.write(word + '\n')  # записываем отформатированную строку в файл
schedule_json.seek(schedule_json.tell() - 5)  # убираем последнюю запятую
schedule_json.write('\n]')  # конец массива
```

скриншот программы для 3-го задания

```xml
<root>
    <lesson>
        <module>Программирование</module>
        <teacher>Письмак Алексей Евгеньевич</teacher>
        <auditory>Актовый зал</auditory>
        <campus>Ломоносова д. 9</campus>
        <start_time>08:20</start_time>
    </lesson>
    <lesson>
        <module>Иностранный язык</module>
        <teacher>Филатов Андрей Сергеевич</teacher>
        <auditory>3211</auditory>
        <campus>Ломоносова д. 9</campus>
        <start_time>11:40</start_time>
    </lesson>
    <lesson>
        <module>Иностранный язык</module>
        <teacher>Филатов Андрей Сергеевич</teacher>
        <auditory>3211</auditory>
        <campus>Ломоносова д. 9</campus>
        <start_time>13:30</start_time>
    </lesson>
</root>
```

```json
[
    {
        "module": "Программирование",
        "teacher": "Письмак Алексей Евгеньевич",
        "auditory": "Актовый 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"
    }
]
```

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

Задание 3.1

№1.py | №2.py | №3.py | №3_libs.py | №3_re.py

```python
import xmltodict
import json


def translation(s):  # функция перевода для корректного отображения кириллицы
    dict_utf = {
        'u0410': 'А', 'u0430': 'а',
        'u0411': 'Б', 'u0431': 'б',
        'u0412': 'В', 'u0432': 'в',
        'u0413': 'Г', 'u0433': 'г',
        'u0414': 'Д', 'u0434': 'д',
        'u0415': 'Е', 'u0435': 'е',
        'u0401': 'Ё', 'u0451': 'ё',
        'u0416': 'Ж', 'u0436': 'ж',
        'u0417': 'З', 'u0437': 'з',
        'u0418': 'И', 'u0438': 'и',
        'u0419': 'Й', 'u0439': 'й',
        'u041a': 'К', 'u043a': 'к',
        'u041b': 'Л', 'u043b': 'л',
        'u041c': 'М', 'u043c': 'м',
        'u041d': 'Н', 'u043d': 'н',
        'u041e': 'О', 'u043e': 'о',
        'u041f': 'П', 'u043f': 'п',
        'u0420': 'Р', 'u0440': 'р',
        'u0421': 'С', 'u0441': 'с',
        'u0422': 'Т', 'u0442': 'т',
        'u0423': 'У', 'u0443': 'у',
        'u0424': 'Ф', 'u0444': 'ф',
        'u0425': 'Х', 'u0445': 'х',
        'u0426': 'Ц', 'u0446': 'ц',
        'u0427': 'Ч', 'u0447': 'ч',
        'u0428': 'Ш', 'u0448': 'ш',
        'u0429': 'Щ', 'u0449': 'щ',
        'u042a': 'Ъ', 'u044a': 'ъ',
        'u042b': 'Ы', 'u044b': 'ы',
        'u042c': 'Ь', 'u044c': 'ь',
        'u042d': 'Э', 'u044d': 'э',
        'u042e': 'Ю', 'u044e': 'ю',
        'u042f': 'Я', 'u044f': 'я',
    }
    for k, v in dict_utf.items():
        s = s.replace('\\' + k, v)
    return s


dict_from_xml = xmltodict.parse(open(r"schedule.xml", encoding='utf-8').read())  # переводим xml-файл в словарь
json_from_dict = json.dumps(dict_from_xml, indent=4)  # делаем строку формата json из словаря

open("schedule.json", 'w', encoding='utf-8').write(translation(json_from_dict))

```

скриншот программы для дополнительного задания 3.1

```
{
    "root": {
        "lesson": [
            {
                "module": "Программирование",
                "teacher": "Письмак Алексей Евгеньевич",
                "auditory": "Актовый 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 сторонних библиотек xmltodict 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.py | №2.py | №3.py | №3_libs.py | №3_re.py

```python
import re

schedule_xml = open(r"schedule.xml", encoding='utf-8').read()  # открываем xml файл для чтения
schedule_json = open("schedule.json", 'w', encoding='utf-8')  # открываем json файл для записи

re_tag_with_text_xml = r'<(\w+)>([\w \d.:,!?]+)<\/\w+>'  # шаблон для однострочных тегов с текстом
schedule_xml = re.sub(re_tag_with_text_xml, r'"\1": "\2"', schedule_xml)  # меняем формат однострочных тегов на JSON

re_tag_with_text_json = r'\"\w+\": \"[\w \d.:,!?]+\"'  # шаблон для поиска новых однострочных тегов
# шаблон для поиска тегов внутри, которых массив тегов
re_tag_with_array_of_tags = r'(\s+)<\w+>(\s+(?:' + re_tag_with_text_json + r'\s+)+)<\/\w+>'
schedule_xml = re.sub(re_tag_with_array_of_tags, r'\1{\2}', schedule_xml)  # превращаем эти массивы в JSON объекты

schedule_xml = re.sub(r'\"(\s+\")', r'",\1', schedule_xml)  # расставляем запятые между однострочными тегами
schedule_xml = re.sub(r'}(\s+{)', r'},\1', schedule_xml)  # расставляем запятые между объектами

# шаблон для поиска новых массивов
re_tag_with_array_of_json_objects = r'<(\w+)>(\s+({[\w\":,.\s]+},\s+)+{[\w\":,.\s]+}\s+)<\/\w+>'
schedule_xml = re.sub(re_tag_with_array_of_json_objects, r'"\1": [\2]', schedule_xml)

schedule_json.write('{\n')  # JSON файл должен начинаться и заканчиваться фигурными скобками
for s in schedule_xml.split('\n'):  # добавим табуляцию к каждой строчке
    schedule_json.write('\t' + s + '\n')
schedule_json.write('}')

```

скриншот программы для дополнительного задания 3.2


```json
{
    "root": [
        {
            "module": "Программирование",
            "teacher": "Письмак Алексей Евгеньевич",
            "auditory": "Актовый 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