НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет ПИиКТ

Дисциплина: Информатика

Лабораторная работа № 3

Выполнил студент

Григорьев Никита Александрович

Группа № P3124

Преподаватель: Болдырева Елена Александровна

г. Санкт-Петербург

2023

Содержание

Вариант: 34

Задание 1

Задание 1. (20% из 100)

1) Реализуйте программный продукт на языке Python, используя регулярные выражения по варианту, представленному в таблице.

2) 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.

3) Программа должна считать количество смайликов определённого вида (вид смайлика описан в таблице вариантов) в предложенном тексте. Все смайлики имеют такую структуру:

[*глаза*][*нос*][*рот*].
Вариантом является различные наборы глаз, носов и ртов.

Номер в ИСУ % 5	Глаза	Номер в ИСУ % 4	Нос	Номер в ИСУ % 7	Рот
0	:	0	-	0	(
1	;	1	<	1	)
2	X	2	-{	2	O
3	8	3	<{	3	|
4	=			4	\
				5	/
				6	P

Пример смайлика: 8<{P

Задание 2

Задание 2. (40% из 100)

1) Реализуйте программный продукт на языке Python, используя регулярные выражения по варианту, представленному в таблице.

2) Для своей программы придумайте минимум 5 тестов.

3) Протестируйте свою программу на этих тестах.

<table>
<tr><th>Номер в ИСУ % 6</th><th>Задание</th></tr>
<tr><td>0</td><td>Написать регулярное выражение, которое проверяет корректность email и в качестве ответа выдаёт почтовый сервер (почтовый сервер – часть email идущая после «@»).
Для простоты будем считать, что почтовый адрес может содержать в себе буквы, цифры, «.» и «_», а почтовый сервер только буквы и «.». При этом почтовый сервер, обязательно должен содержать верхний уровень домена («.ru», «.com», etc.)

Пример:

<table>
<tr><td>Ввод</td><td>Вывод</td></tr>
<tr><td>students.spam@yandex.ru</td><td>yandex.ru</td></tr>
<tr><td>example@example</td><td>Fail!</td></tr>
<tr><td>example@example.com</td><td>example.com</td></tr>
</table>
</td></tr>
<tr><td>1</td><td>Студент Вася очень любит курс «Компьютерная безопасность». Однажды Васе задали домашнее задание зашифровать данные, переданные в сообщение. Недолго думая, Вася решил заменить все целые числа на функцию от этого числа. Функцию он придумал не сложную $3x2+5$,где x−исходное число. Помогите Васе с его домашним заданием.
Пример:

<table>
<tr><td>Ввод</td><td>Вывод</td></tr>
<tr><td>20 + 22 = 42</td><td>1205 + 1457 = 5297</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>Вывесили списки стипендиатов текущего семестра, которые представляют из себя список людей ФИО и номер группы этого человека. Вы решили подшутить над некоторыми из своих одногруппников и удалить их из списка.
С помощью регулярного выражения найдите всех студентов своей группы, у которых инициалы начинаются на одну и туже букву и исключите их из списка.
Пример (группа P000):

<table>
<tr><td>Ввод</td><td>Вывод</td></tr>
<tr><td>Петров П.П. P000
Анищенко А.А. P33113</td><td>Анищенко А.А. P33113
Примеров Е.В. P000</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td>Примеров Е.В. Р000
Иванов И.И. Р000</td><td></td></tr></table></td></tr>
<tr><td>3</td><td>Дан текст. Необходимо найти в нём любой фрагмент, где сначала идёт слово «ВТ», затем не более 4 слов, и после этого идёт слово «ИТМО». Для простоты будем считать словом любую последовательность букв, цифр и знаков «_» (то есть символов \w).
Пример:
<table><tr><td>Ввод</td><td>Вывод</td></tr><tr><td>А ты знал, что ПИиКТ – лучший факультет в ИТМО?</td><td>ПИиКТ лучший факультет в ИТМО</td></tr></table></td></tr>
<tr><td>4</td><td>Дан текст. Требуется найти в тексте все фамилии, отсортировав их по алфавиту.
Фамилией для простоты будем считать слово с заглавной буквой, после которого идут инициалы.

Пример:
<table><tr><td>Ввод</td><td>Вывод</td></tr><tr><td>Студент Вася вспомнил, что на своей лекции Болдырева Е.А. упоминала про старшекурсников, которые будут ей помогать: Анищенко А.А. и Машина Е.А.</td><td>Анищенко
Болдырева
Машина</td></tr></table></td></tr>
<tr><td>5</td><td>Анатолий выложил пост с расписанием доп. занятий по информатике, но везде перепутал время. Поэтому нужно заменить все вхождения времени на строку (TBD).
Время – это строка вида HH:MM:SS или HH:MM, в которой HH – число от 00 до 23, а MM и SS – число от 00 до 59.

Пример:
<table><tr><td>Ввод</td><td>Вывод</td></tr><tr><td>Уважаемые студенты! В эту субботу в 15:00 планируется доп. занятие на 2 часа. То есть в 17:00:01 оно уже точно кончится.</td><td>Уважаемые студенты! В эту субботу в (TBD) планируется доп. занятие на 2 часа. То есть в (TBD) оно уже точно кончится.</td></tr></table></td></tr>
</table>

Задание 3. (40% из 100)

1. Определить номер варианта как остаток деления номера в ИСУ на 36. 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**: 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 №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.

9. **Дополнительное 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.

Отчет

Реализация задания 1:

Task 1

```
import re

def count_smileys(text: str) -> int:
    # Регулярное выражение для поиска смайликов ":-{/"
    smiley_pattern = r':-\{/'

    # Ищем все смайлики в тексте
    smileys = re.findall(smiley_pattern, text)

    # Возвращаем количество найденных смайликов
    return len(smileys)

# Тестовый пример
sample_text = "Hello :-{/! How are you? :-{/ I'm fine, thanks! :-{/"
print('example: ' + str(count_smileys(sample_text)))

# Тестовые данные
tests = [
    ("Hello! How are you?", 0),
    ("I'm doing great :-{/!", 1),
    (":-{/ :-{/ :-{/ :-{/", 4),
    (":-{ :-} :-/ :-{/", 1),
    ("", 0)
]

# Проверка функции на тестах
results = []
for test_text, expected in tests:
    result = count_smileys(test_text)
    results.append((test_text, result, expected, result == expected))

results
```

```
example: 3
```

```
[('Hello! How are you?', 0, 0, True),
 ("I'm doing great :-{/!", 1, 1, True),
 (':-{/ :-{/ :-{/ :-{/', 4, 4, True),
 (':-{ :-} :-/ :-{/', 1, 1, True),
 ('', 0, 0, True)]
```

Реализация задания 2:

Task 2

```
def extract_sorted_surnames(text: str) -> list:
    # Регулярное выражение для поиска фамилий с инициалами
    surname_pattern = r'([А-Я][а-я]+)\s+[А-Я]\.[А-Я]\.'

    # Ищем все фамилии в тексте
    surnames = re.findall(surname_pattern, text)

    # Сортируем фамилии по алфавиту
    sorted_surnames = sorted(surnames)

    return sorted_surnames

# Тестовый пример
sample_text = ("Студент Вася вспомнил, что на "
               "своей лекции Болдырева Е.А. "
               "упоминала про старшекурсников, "
               "которые будут ей помогать: "
               "Анищенко А.А. и Машина Е.А.")
extract_sorted_surnames(sample_text)

print("example: " + str(extract_sorted_surnames(sample_text)))

# Тестовые данные
tests = [
    ("Студент Вася вспомнил, что на своей лекции Болдырева Е.А. упоминала про старшекурсников, которые будут ей помо
    ("Преподаватели университета: Зайцев И.И., Медведев С.П., Орлова А.С. и Попов Д.В.", ['Зайцев', 'Медведев', 'Орл
    ("На занятии присутствовали: Лисичкин В.В. и Котов А.А.", ['Котов', 'Лисичкин']),
    ("В соревновании участвовали: Кролик Р.Р., Олененок О.О. и Собакин С.С.", ['Кролик', 'Олененок', 'Собакин']),
    ("Директор института — Петров П.П.", ['Петров'])
]

# Проверка функции на тестах
results = []
for test_text, expected in tests:
    result = extract_sorted_surnames(test_text)
    results.append((result, expected, result == expected))

results
```

```
example: ['Анищенко', 'Болдырева', 'Машина']
```

```
[(['Анищенко', 'Болдырева', 'Машина'],
  ['Анищенко', 'Болдырева', 'Машина'],
  True),
 (['Зайцев', 'Медведев', 'Орлова', 'Попов'],
  ['Зайцев', 'Медведев', 'Орлова', 'Попов'],
  True),
 (['Котов', 'Лисичкин'], ['Котов', 'Лисичкин'], True),
 (['Кролик', 'Олененок', 'Собакин'], ['Кролик', 'Олененок', 'Собакин'], True),
 (['Петров'], ['Петров'], True)]
```

Реализация задания 3:

```python
def yaml_to_list(yaml_content):
    """
    Преобразует содержимое YAML в список.

    :param yaml_content: Содержимое YAML в виде строки.
    :return: Список, представляющий содержимое YAML.
    """

    # Разбиваем содержимое на строки
    lines = yaml_content.split('\n')

    # Инициализируем результирующий список
    result = []

    # Перебираем каждую строку
    for line in lines:
        # Удаляем лишние пробелы по краям строки
        stripped_line = line.strip()

        # Если строка пустая или начинается с комментария, пропускаем её
        if not stripped_line or stripped_line.startswith('#'):
            continue

        # Если строка начинается с дефиса, считаем её элементом списка
        if stripped_line.startswith('- '):
            # Добавляем элемент списка в результирующий список
            result.append(stripped_line[2:])
        else:
            # Иначе считаем строку ключ-значение и добавляем её в результирующий список
            key, value = stripped_line.split(':', 1)
            result.append(f"{key.strip()}: {value.strip()}")

    return result
```

```python
def list_to_json(data_list):
    """
    Преобразует список в JSON-формат, где каждый день содержит список уроков.

    :param data_list: Список для преобразования.
    :return: Строка в формате JSON.
    """

    # Начинаем JSON-объект с открывающей фигурной скобки
    json_content = "{\n"

    # Счетчик для отслеживания текущего элемента в списке
    i = 0
    while i < len(data_list):
        # Извлекаем ключ и значение из строки списка
        key, value = data_list[i].split(':', 1)

        # Если значение пустое, это день недели
        if not value.strip():
            json_content += f'  "{key.strip()}": ' + "[\n"
            i += 1
            while i < len(data_list) and (':' not in data_list[i] or not data_list[i].split(':', 1)[1].strip()):
                subject, _ = data_list[i].split(':', 1)
                json_content += f'    ' + "{\n"
                json_content += f'      "name": "{subject.strip()}",\n'
                i += 1
                while i < len(data_list) and ':' in data_list[i] and data_list[i].split(':', 1)[1].strip():
                    sub_key, sub_value = data_list[i].split(':', 1)
                    json_content += f'      "{sub_key.strip()}": {sub_value.strip()},\n'
                    i += 1
                json_content = json_content.rstrip(',\n')  # Удаляем последнюю запятую
                json_content += "\n    },\n"
            json_content = json_content.rstrip(',\n')  # Удаляем последнюю запятую
            json_content += "\n  ],\n"

    # Удаляем последнюю запятую и добавляем закрывающую фигурную скобку
    json_content = json_content.rstrip(',\n')
    json_content += "\n}"

    return json_content
```

```python
start_time_hands = time()

with open("schedule.yaml", "r", encoding="utf-8") as file:
    yaml_schedule = file.read()
print(yaml_schedule)
parsed_data = yaml_to_list(yaml_schedule)
print(parsed_data)
json_data = list_to_json(parsed_data)
print(json_data)

with open("schedule.json", 'w') as f:
    f.write(json_data)

end_time_hands = time()
```

```
Вторник:
  - Основы дискретной математики (базовый уровень)(практика):
    time: "11:40-13:10"
    weeks: "2-16"
    location: "Кронверкский пр., д.49, лит.А"
    classroom: "2435/7"
    teacher: "Лисицына Любовь Сергеевна"
    type: "Очно-дистанционный"
  - Математический анализ(лекция):
    time: "13:30-15:00"
    weeks: "2-16"
    location: "Кронверкский пр., д.49, лит.А"
    classroom: "1405"
    teacher: "Правдин Константин Владимирович"
    type: "Очно-дистанционный"
  - Математический анализ(практика):
    time: "15:20-16:50"
    weeks: "2-16"
    location: "Кронверкский пр., д.49, лит.А"
    classroom: "2416"
    teacher: "Правдин Константин Владимирович"
    type: "Очно"

['Вторник: ', 'Основы дискретной математики (базовый уровень)(практика):', 'time: "11:40-13:10"', 'weeks: "2-16"',
'location: "Кронверкский пр., д.49, лит.А"', 'classroom: "2435/7"', 'teacher: "Лисицына Любовь Сергеевна"', 'type:
"Очно-дистанционный"', 'Математический анализ(лекция):', 'time: "13:30-15:00"', 'weeks: "2-16"', 'location: "Кронве
ркский пр., д.49, лит.А"', 'classroom: "1405"', 'teacher: "Правдин Константин Владимирович"', 'type: "Очно-дистанци
онный"', 'Математический анализ(практика):', 'time: "15:20-16:50"', 'weeks: "2-16"', 'location: "Кронверкский пр.,
д.49, лит.А"', 'classroom: "2416"', 'teacher: "Правдин Константин Владимирович"', 'type: "Очно"']
{
  "Вторник": [
    {
      "name": "Основы дискретной математики (базовый уровень)(практика)",
      "time": "11:40-13:10",
      "weeks": "2-16",
      "location": "Кронверкский пр., д.49, лит.А",
      "classroom": "2435/7",
      "teacher": "Лисицына Любовь Сергеевна",
      "type": "Очно-дистанционный"
    },
    {
      "name": "Математический анализ(лекция)",
      "time": "13:30-15:00",
      "weeks": "2-16",
      "location": "Кронверкский пр., д.49, лит.А",
      "classroom": "1405",
      "teacher": "Правдин Константин Владимирович",
      "type": "Очно-дистанционный"
    },
    {
      "name": "Математический анализ(практика)",
      "time": "15:20-16:50",
      "weeks": "2-16",
      "location": "Кронверкский пр., д.49, лит.А",
      "classroom": "2416",
      "teacher": "Правдин Константин Владимирович",
      "type": "Очно"
    }
  ]
}
```

Реализация доп. задания 1:

Libs

In [23]:
```
start_time_lib = time()

with open("schedule.yaml", "r", encoding="utf-8") as file:
    pyyaml_parsed_data = yaml.safe_load(file)
pyyaml_json = json.dumps(pyyaml_parsed_data)

end_time_lib = time()
```

Compare

In [24]:
```
hands_time = end_time_hands - start_time_hands
lib_time = end_time_lib - start_time_lib

print(f"My parser time: {hands_time}")
print(f"Lib parser time: {lib_time}")
```

```
My parser time: 0.003997087478637695
Lib parser time: 0.01137399673461914
```

Конвертированные файлы отличаются лишь порядком характеристик из-за сортировки. Сравнение времени: основное решение работает быстрее решения с библиотеками (оно медленнее, чем основное, так как необходимо время на загрузку библиотек).

Вывод:

В ходе выполнения лабораторной работы я изучил регулярные выражения на языке Python, ознакомился с тем, что представляет из себя парсинг данных и языки разметки, попробовал решить задачу парсинга разными способами.

Список литературы:

1. Регулярные выражения в Python от простого к сложному. // Habr URL: https://habr.com/ru/post/349860/
2. Регулярные выражения в Python: синтаксис, полезные функции и задачи // Skillbox URL: https://skillbox.ru/media/code/regulyarnye-vyrazheniya-v-python-sintaksis-poleznyefunktsii-i-zadachi/
3. YAML vs JSON. // Habr URL: https://habr.com/ru/company/rambler_and_co/blog/525498/