Федеральное государственное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина «Информатика»

**Отчёт по лабораторной работе №5**

**Вариант №9**

**Работа с электронными таблицами**

**Выполнил:**Смирнов Вадим Константинович

Группа: P3119

**Проверил:**  
Авксентьева Елена Юрьевна, доцент

г. Санкт-Петербург

2024

Оглавление

[**Задание 2**](#_Toc183774611)

[**Решение 5**](#_Toc183774612)

[**Вывод 8**](#_Toc183774613)

[**Список литературы 9**](#_Toc183774614)

# 

# Задание

**Обязательное задание**

Написать программу на языке Python 3.x или любом другом, которая бы осуществляла парсинг и конвертацию исходного файла в новый путём простой замены метасимволов исходного формата на метасимволы результирующего формата

**Дополнительное задание №1**

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. Объяснение должно быть отражено в отчёте.

**Дополнительное задание №2**

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

b) Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

**Дополнительное задание №3**

а) Переписать исходный код таким образом, чтобы для решения задачи использовались формальные грамматики. То есть ваш код должен уметь осуществлять парсинг и конвертацию любых данных, представленных в исходном формате, в данные, представленные в результирующем формате: как с готовыми библиотеками из дополнительного задания №1.

b) Проверку осуществить как минимум для расписания с двумя учебными днями по два занятия в каждом.

с) Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

**Дополнительное задание №4**

a) Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле.

b) Проанализировать полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

**Дополнительное задание №5**

a) Переписать исходную программу, чтобы она осуществляла парсинг и конвертацию исходного файла в любой другой формат (кроме JSON, YAML, XML, HTML): PROTOBUF, TSV, CSV, WML и т.п. 2

­b) Проанализировать полученные результаты, объяснить особенности использования формата. Объяснение должно быть отражено в отчёте.

# Решение

**Основное задание**

import time

PUNCTUATION = r"""!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~"""

def open\_tag1(s):

    for i in range(len(s)):

        if s[i] == '>':

            return s[:i+1]

def close\_tag1(s):

    return s.replace('<', '</')

def ww\_t(text, o\_t, c\_t):

    result = text.replace(o\_t, '').replace(c\_t, '').strip()

    if result:

        if not result.isalnum() or not result.isascii():

            result = f'\'{result}\''

    return result

def open\_cont(lines):

    o\_t = []

    yml\_content = []

    for row in lines:

        tab\_lenght = '\t' \* len(o\_t)

        tmp\_row = row.strip()

        if tmp\_row[0] == '<' and tmp\_row[1] not in PUNCTUATION:

            o\_t = open\_tag1(tmp\_row)

            c\_t = close\_tag1(o\_t)

            text = ww\_t(tmp\_row, o\_t, c\_t)

            if tmp\_row[-len(c\_t):] != c\_t:

                o\_t.append(c\_t)

            yml\_content.append(f'{tab\_lenght}{o\_t[1:-1]}: {text}')

        elif tmp\_row[0] == '<' and tmp\_row[1] == '/':

            o\_t.pop()

    return '\n'.join(yml\_content)

def main():

    start\_time = time.time()

    with open('test.xml', mode='r', encoding='utf-8') as f:

        xml\_lines = f.readlines()

    yaml\_content = open\_cont(xml\_lines)

    with open('test.yml', mode='w') as f:

        f.write(yaml\_content)

    end\_time = time.time()

    print(end\_time - start\_time)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**1.)**

import xmltodict

import yaml

from collections import OrderedDict

import time

start\_time = time.time()

with open("test1.xml", "r", encoding='utf-8') as xml\_file:

  xml\_string = xml\_file.read()

python\_dict = xmltodict.parse(xml\_string)

#print(python\_dict)

#python\_dict['timetable']['lessons'] = OrderedDict(python\_dict['timetable']['lessons'])

with open("dop1.yaml", "w", encoding='utf-8') as yaml\_file:

  yaml.dump(python\_dict, yaml\_file, allow\_unicode=True, default\_flow\_style=False)

end\_time = time.time()

print(end\_time - start\_time)

**2.)**

import re

import time

def open\_tag1(s):

    match = re.match(r'<([^>]\*)>', s)

    if match:

        return match.group(0)

def close\_tag1(s):

    return s.replace('<', '</')

def ww\_t(text, open\_tag, close\_tag):

    result = re.sub(f'{open\_tag}|{close\_tag}', '', text).strip()

    if result:

        if not result.isalnum() or not result.isascii():

            result = f'\'{result}\''

    return result

def open\_count(lines):

    open\_tags = []

    yml\_content = []

    for row in lines:

        tmp\_row = row.strip()

        tab\_lenght = '\t' \* len(open\_tags)

        match = re.match(r'<([^/>]+)>', tmp\_row)

        if match:

            open\_tag = match.group(0)

            close\_tag = close\_tag1(open\_tag)

            text = ww\_t(tmp\_row, open\_tag, close\_tag)

            if tmp\_row[-len(close\_tag):] != close\_tag:

                open\_tags.append(close\_tag)

            yml\_content.append(f'{tab\_lenght}{open\_tag[1:-1]}: {text}')

        elif re.match(r'</[^>]+>', tmp\_row):

            open\_tags.pop()

    return '\n'.join(yml\_content)

def main():

    start\_time = time.time()

    with open('test.xml', mode='r', encoding='utf-8') as f:

        xml\_lines = f.readlines()

    yaml\_content = open\_count(xml\_lines)

    with open('dop2.yml', mode='w') as f:

        f.write(yaml\_content)

    end\_time = time.time()

    print(end\_time - start\_time)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**3.)**

import time

PUNCTUATION = r"""!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~"""

def open\_tag1(s):

    for i in range(len(s)):

        if s[i] == '>':

            return s[:i+1]

def close\_tag1(s):

    return s.replace('<', '</')

def ww\_t(text, o\_t, c\_t):

    result = text.replace(o\_t, '').replace(c\_t, '').strip()

    if result:

        if not result.isalnum() or not result.isascii():

            result = f'\'{result}\''

    return result

def open\_cont(lines):

    o\_t = []

    yml\_content = []

    for row in lines:

        tab\_lenght = '\t' \* len(o\_t)

        tmp\_row = row.strip()

        if tmp\_row[0] == '<' and tmp\_row[1] not in PUNCTUATION:

            o\_t = open\_tag1(tmp\_row)

            c\_t = close\_tag1(o\_t)

            text = ww\_t(tmp\_row, o\_t, c\_t)

            if tmp\_row[-len(c\_t):] != c\_t:

                o\_t.append(c\_t)

            yml\_content.append(f'{tab\_lenght}{o\_t[1:-1]}: {text}')

        elif tmp\_row[0] == '<' and tmp\_row[1] == '/':

            o\_t.pop()

    return '\n'.join(yml\_content)

def main():

    start\_time = time.time()

    with open('test.xml', mode='r', encoding='utf-8') as f:

        xml\_lines = f.readlines()

    yaml\_content = open\_cont(xml\_lines)

    with open('test.yml', mode='w') as f:

        f.write(yaml\_content)

    end\_time = time.time()

    print(end\_time - start\_time)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**4.)**

Программа обязательного задания работает быстрее всего, потому что там реализована работа только исходного файла. Второе задание работает дольше, потому что применено использование регулярных выражений. Третье расширяет функционал парсера, следовательно, увеличивает время работы программы. Первое работает дольше всех, потому что является универсальным парсером, в библиотеке реализованы все аспекты перевода XML в YAML.

**main - 0.0009624958038330078**

**1 - 0.011582374572753906**

**2 - 0.0011382102966308594**

**3 - 0.00738485838**

**5.)**

import xmltodict

import csv

def convert\_xml\_to\_csv(xml\_file, csv\_file):

    with open(xml\_file, 'r', encoding='utf-8') as file:

        xml\_content = file.read()

    data\_dict = xmltodict.parse(xml\_content)

    lessons = data\_dict['timetable']['lessons']

    A = []

    for key in lessons.keys():

        A.append(lessons[key])

    headers = A[0].keys()

    with open(csv\_file, 'w', newline='', encoding='utf-8') as csv\_file:

        writer = csv.DictWriter(csv\_file, fieldnames=headers)

        writer.writeheader()

        for lesson in A:

            writer.writerow(lesson)

open1 = 'pn.xml'

close1 = 'timetable\_output1.csv'

convert\_xml\_to\_csv(open1, close1)

# 

# Вывод

В ходе выполнения работы я улучшил свои навыки работы с электронными таблицами, получил практику в построении графиков и диаграмм с использованием ЯП Python, а также с помощью Excel.

# Список литературы

Регистр флагов — регистр процессора (FLAGS), отражающий текущее состояние процессора. Регистр флагов содержит группу флагов состояния (арифметические флаги) и флаги управления.

Арифметические флаги формируются арифметическими командами (ADD, SUB, MUL, DIV) и являются признаками их результата.

Флаги управления оказывают влияние на процесс выполнения программ.

К арифметическим флагам относятся:

• (CF) Carry Flag – флаг переноса, в нем фиксируется перенос из стар-шего разряда при сложении и заем в старший разряд при вычитании. При умножении CF показывает возможность (= 0) и невозможность (= 1) пред-ставления произведения в том же формате, что и операндов.

• (PF) Parity Flag – флаг паритета (четности). Устанавливается в единицу при наличии четного числа единиц в младшем байте результата, в противном случае - сбрасывается. PF используется в качестве аппаратной поддержки контроля по четности.

• (AF) Auxiliary Carry Flag – флаг вспомогательного переноса, в кото-ром фиксируется межтетрадный перенос при сложении и межтетрадный заём при вычитании. Этот флаг используется командами десятичной арифметики.

• (ZF) Zero Flag – флаг нуля, устанавливается при нулевом значении ре-зультата, в противном случае сбрасывается.

• (SF) Sign Flag – флаг знака, в котором копируется старший разряд ре-зультата.

• (OF) Overflow Flag флаг переполнения. Устанавливается в командах сложения и вычитания, если результат не помещается в формате, при этом и операнды и результат интерпретируются как знаковые числа. Аппаратно он формируется совпадением переносов из двух старших разрядов при сложении и заёмов в два старших разряда при вычитании (если они совпадают, то флаг равен нулю).

Переполнение при сложении чисел возникает только в том случае, если операнды имеют одинаковые знаки. Переполнение фиксируется тремя спосо-бами:

• сравнение знаков операндов и суммы: если знак суммы отличается от знаков операндов, то фиксируется переполнение;

• сравнение переносов из двух старших разрядов: если они не совпадают, то фиксируется переполнение;

• использование модифицированного знака (под знак отводится два раз-ряда, второй разряд дублирует знак).