Университет ИТМО

Факультет ФПИ и КТ

**Лабораторная работа №4**

По информатике

Вариант 12

Выполнил: Кочнев Р.Д.

Группа: Р3117

Преподаватель: Балашкин П.В.

Санкт-Петербург

2021

Оглавление

[Задание 2](#_Toc86241625)

[Основные этапы вычисления 2](#_Toc86241626)

[1 2](#_Toc86241627)

[2 4](#_Toc86241628)

[3 5](#_Toc86241629)

[4 8](#_Toc86241630)

[Вывод 8](#_Toc86241631)

## Задание



## Основные этапы вычисления

### 1

with open('Расписание.yaml', 'r', encoding="utf-8") as f:

s = f.read()

lens = (len(s.split("\n")))

strings = s.split("\n")

spc = []

name = []

word = []

xml = '<?xml version="1.0" encoding="UTF-8" ?>\n'

xml += "<root>\n"

def name\_class(strr):

v = ""

for j in range(len(strr)):

if strr[j] == " " or strr[j] == "-":

continue

elif strr[j] == ":":

name.append(v)

return v

else:

v += strr[j]

def spc\_count(strr):

strr = strr.replace("-", " ")

spc.append(len(strr) - len(strr.lstrip(' ')))

return (len(strr) - len(strr.lstrip(' ')))

def find\_words(strr):

b = 0

b1 = ""

for j in range(len(strr)):

if strr[j] == "'" and b != 0:

word.append(b1)

return b1

elif b == 1:

b1 += strr[j]

elif strr[j] == "'":

b = 1

for i in range(lens):

a = name\_class(strings[i])

b = spc\_count(strings[i])

c = find\_words(strings[i])

if c == None:

word.append("")

print(b, a, c)

maxspc = max(spc)

elem = set(spc)

v = len(elem)

prev = [-1] \* maxspc

print(prev)

print(("name", name))

print("spc", spc)

print(len(word), "word", word)

print("das'd", lens)

for i in range(lens):

if spc[i] == maxspc:

xml += " " \* maxspc

xml += f"<{name[i]}> {word[i]} </{name[i]}>\n"

else:

n = prev[spc[i]]

if n == -1:

prev[spc[i]] = i

xml += " " \* (spc[i] + 2)

xml += f"<{name[i]}> {word[i]}\n"

# print(name[i], word[i],i, "re")

else:

xml += " " \* (spc[i] + 2)

xml += f"</{name[n]}>\n"

xml += " " \* spc[i]

xml += f"<{name[i]}> {word[i]}\n"

prev[spc[i]] = i

for i in range(len(prev)):

if prev[3 - i] != -1:

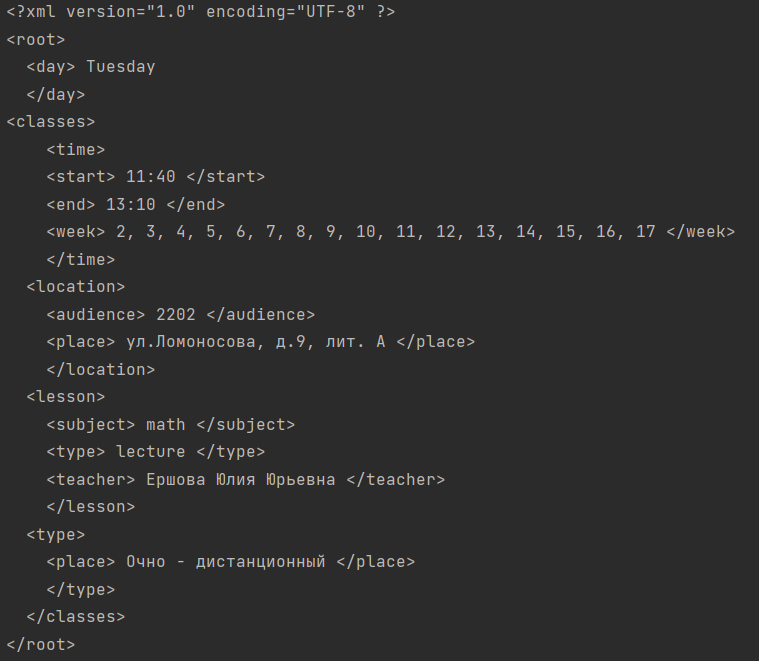
xml += " " \* (spc[prev[3 - i]] + 2)

xml += f"</{name[prev[3 - i]]}>\n"

print(i)

xml += "</root>"

print(xml)



Задание1

### 2

С использованием библиотек

import yaml

from dicttoxml import dicttoxml

import time

start\_time = time.time()

with open("table1.yaml", 'r', encoding="utf-8") as stream:

parsed\_yaml = yaml.safe\_load(stream)

print(parsed\_yaml)

xml = dicttoxml(parsed\_yaml)

print(xml)

print('%.5f' % ((time.time() - start\_time)\*10))

<?xml version="1.0" encoding="UTF-8" ?><root><day type="str">Tuesday</day><classes type="list"><item type="dict"><time type="dict"><start type="str">11:40</start><end type="str">13:10</end><week type="str">2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17</week></time><location type="dict"><audience type="str">2202</audience><place type="str">Lomonosov st., 9, lit. A</place></location><lesson type="dict"><subject type="str">math</subject><type type="str">lecture</type><teacher type="str">Ershova julia</teacher></lesson><type type="dict"><place type="str">in person - remotely</place></type></item></classes></root>

С доп. библиотеками код получается сильно короче

Доп задание 1

### 3

С регулярными выражениями

import re

import time

start\_time = time.time()

with open('table1.yaml', 'r', encoding="utf-8") as f:

s = f.read()

lens = (len(s.split("\n")))

strings = s.split("\n")

spc = []

name = []

word = []

xml = '<?xml version="1.0" encoding="UTF-8" ?>\n'

xml += "<root>\n"

def name\_class(strr):

patern = "\w+:"

v = (re.findall(patern, strr))

name.append(str(v[0])[:-1])

return str(v[0])[:-1]

def spc\_count(strr):

strr = re.sub(r"-", r" ", strr)

v = re.findall(r"^\s+", strr)

try:

spc.append(len(v[0]))

return len(v[0])

except:

spc.append(0)

return 0

def find\_words(strr):

patern = "'[^+]+'"

name = re.findall(patern, strr)

if len(name) > 0:

word.append(str(name[0])[1:-1])

return str(name[0])[1:-1]

else:

return None

for i in range(lens):

a = name\_class(strings[i])

b = spc\_count(strings[i])

c = find\_words(strings[i])

if c == None:

word.append("")

print(b, a, c)

maxspc = max(spc)

elem = set(spc)

v = len(elem)

prev = [-1] \* maxspc

for i in range(lens):

if spc[i] == maxspc:

xml += " " \* (maxspc + 2)

xml += f"<{name[i]}> {word[i]} </{name[i]}>\n"

else:

n = prev[spc[i]]

if n == -1:

prev[spc[i]] = i

xml += " " \* (spc[i] + 2)

xml += f"<{name[i]}> {word[i]}\n"

# print(name[i], word[i],i, "re")

else:

xml += " " \* (spc[i] + 2)

xml += f"</{name[n]}>\n"

xml += " " \* (spc[i] + 2)

xml += f"<{name[i]}> {word[i]}\n"

prev[spc[i]] = i

for i in range(len(prev)):

if prev[3 - i] != -1:

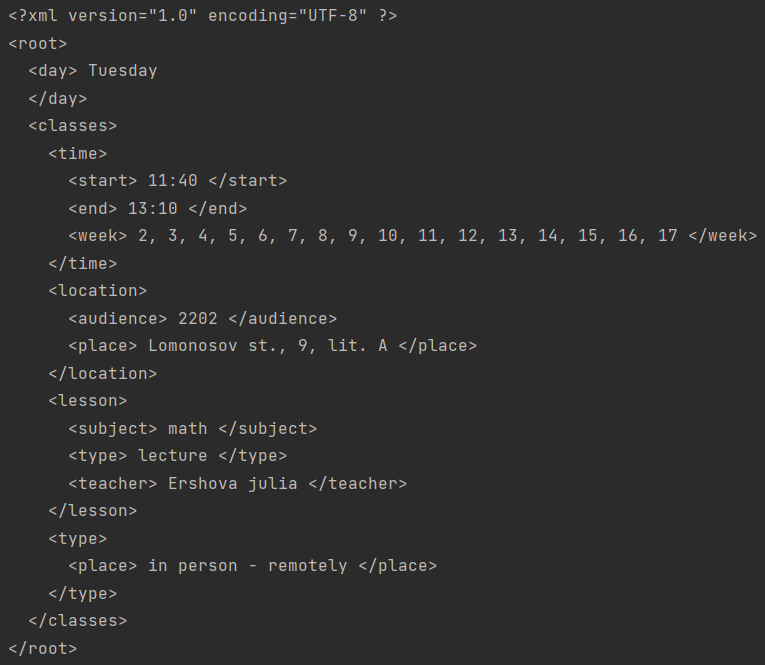
xml += " " \* (spc[prev[3 - i]] + 2)

xml += f"</{name[prev[3 - i]]}>\n"

print(i)

xml += "</root>"

print(xml)

print('%.5f' % ((time.time() - start\_time)\*10)) 

Доп задание 2

Регулярные выражения облегчают нахождение классов, с ними код получается короче

# 4

С доп. библиотеками программа выполняется 10 раз за 

С регулярными выражениями 10 раз за 

## Вывод

С готовыми библиотеками программа работает более чем в 10 раз дольше, чем без них.

В процессе работы над этой лабораторной работой я понял важность использования регулярных выражений, познакомился с YAML, XML json и HTML. Познакомился с языком разметки markdown, и написал парсер.