МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПИиКТ

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине

«ИНФОРМАТИКА»

Вариант №9

Выполнил:

Студент группы P3119

Билобрам Денис Андреевич

Преподаватель:

Рыбаков Степан Дмитриевич

Санкт-Петербург, 2022

**Обязательное задание main.py**

def main():

    def val\_key\_to\_types(st):

        if st:

            if st.isnumeric():

                st = int(st)

            elif st == 'false':

                st = False

            elif st == 'Flase':

                st = False

            elif st == 'null':

                st = None

            else:

                if (st[0] == "'" and st[-1] == "'") or (st[0] == '"' and st[-1] == '"'):

                    st = st[1:-1]

                st = st.replace("'", "\&comma321654897321265468&").replace('"', "\&doublecomma&321654897321265468")

        return st

    with open('yaml.yaml', 'r') as yamlfile:

        yamlstrs = yamlfile.readlines()

        pars\_file = []

        for i in range(len(yamlstrs)):

            yamlstr = yamlstrs[i].strip().replace('\n', '')

            j = 0

            for ch in yamlstrs[i]:

                if ch == ' ':

                    j += 1

                    continue

                break

            if yamlstr[0] == '#':

                continue

            pars\_file.append([j, yamlstr])

        objs = {}

        last\_i = False

        buff = objs

        buffers = []

        last = dict

        for i in range(len(pars\_file)):

            if i == len(pars\_file)-1:

                last\_i = True

            if not last\_i:

                diff = (pars\_file[i][0] - pars\_file[i+1][0]) // 2

            else:

                diff = 0

            vals = pars\_file[i][1].split(': ')

            for j in range(len(pars\_file[i][1])-2):

                if pars\_file[i][1][j:j+2] == ': ':

                    vals = (pars\_file[i][1][:j], pars\_file[i][1][j+2:])

                    break

            key = value = False

            if len(vals) == 2:

                key, value = vals[0], vals[1]

            else:

                if vals[0][-1] == ':':

                    key = vals[0][:-1]

                else:

                    value = vals[0]

            value = val\_key\_to\_types(value)

            key = val\_key\_to\_types(key)

            if diff == 0:

                if last == list:

                    if value and key:

                        key = key[2:]

                        key = val\_key\_to\_types(key)

                        buff.append({key: value})

                    else:

                        value = value[2:]

                        value = val\_key\_to\_types(value)

                        buff.append(value)

                elif last == dict:

                    if key and value:

                        buff[key] = value

                    elif value:

                        if None not in buff.keys():

                            buff[None] = []

                        buff[None].append(value)

            elif diff < 0:

                if pars\_file[i+1][1][:2] == '- ':

                    if last == list:

                        key = key[2:]

                        key = val\_key\_to\_types(key)

                        buff.append({key: []})

                        buff = buff[-1][key]

                        last = list

                    elif last == dict:

                        buff[key] = []

                        buff = buff[key]

                        last = list

                else:

                    if last == list:

                        key = key[2:]

                        key = val\_key\_to\_types(key)

                        buff.append({key: {}})

                        buff = buff[-1][key]

                        last = dict

                    elif last == dict:

                        buff[key] = {}

                        buff = buff[key]

                        last = dict

                buffers.append(buff)

            elif diff > 0:

                if last == list:

                    value = value[2:]

                    value = val\_key\_to\_types(value)

                    buff.append(value)

                elif last == dict:

                    if value and key:

                        buff[key] = value

                    elif value:

                        if None not in buff.keys():

                            buff[None] = []

                        buff[None].append(value)

                buff = buffers[-diff-1]

                last = type(buff)

                buffers = buffers[:-diff]

        with open('json.json', 'w') as jsonfile:

            content = list(str(objs))

            for i in range(len(content)-1):

                if content[i] == "'":

                    content[i] = '"'

            content = "".join(content)

            content = content.replace(r"\\&comma321654897321265468&", "'").replace(r'\&doublecomma321654897321265468&', '"')

            jsonfile.write(content)

main()

**Дополнительное задание №1 dop1.py**

Для задачи конвертирования yaml в json существует библиотека PyYAML.

Установить её можно следующим образом:

$ **pip install pyyaml**

Код с использованием pyyaml

import json

import yaml

with open('yaml.yaml', 'r') as yamlfile, open('json.json', 'w') as jsonfile:

yaml\_object = yaml.safe\_load(yamlfile)

json.dump(yaml\_object, jsonfile)

Несложно заметить, что использование готовых библиотек сильно экономит время и силы, особенно при разработке программ для таких тривиальных задач, как конвертирование файла из одного формата в другой.

**Дополнительное задание №2 dop2.py**

Используя библиотеку с регулярными выражениями (re) можно без особых усилий исправить недоработки исходного парсера, например, изменив строки 52-61:

val\_i = re.match(r'^[^\'\"]\*: ', pars\_file[i][1])

key = value = False

if val\_i:

key, value = pars\_file[i][1][:val\_i.end()-1], pars\_file[i][1][val\_i.end():]

else:

if pars\_file[i][1][-1] == ':':

key = pars\_file[i][1][:-1]

else:

value = pars\_file[i][1]

Дополнительное задание №3 dop3.py

Используя библиотеки datetime и time замерим время стократного исполнения каждой программы:

from datetime import datetime

import time

from main import main

from dop1 import dop1\_main

from dop2 import dop2\_main

# main time

start\_time = datetime.now()

for i in range(100):

main()

print(f"Main time: {datetime.now() - start\_time}")

# dop1 time

start\_time = datetime.now()

for i in range(100):

dop1\_main()

print(f"Dop1 time: {datetime.now() - start\_time}")

# dop2 time

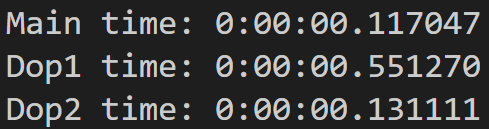
start\_time = datetime.now()

for i in range(100):

dop2\_main()

print(f"Dop2 time: {datetime.now() - start\_time}")

В результате получим следующие значения:



Из них можно сделать вывод, что программа из обязательного задания работает быстрее всех, однако без использования регулярных выражений надежность её парсинга под сомнением. Программа из дополнительного задания №2 отработала чуть медленнее из-за использования регулярных выражений, однако она гораздо надежнее.

Программа из дополнительного задания №3 работала дольше всего, однако на ее надежность можно положиться, так как она использует проверенные библиотеки.

**Выводы:**

Во время выполнения данной лабораторной работы я написал интересный алгоритм парсера Yaml to Json на языке программировании Python. Данный алгоритм без ошибочно конвертирует Yaml файлы в Json. Также познакомился с библиотеками pyyaml и json для той же конвертации файлов. Я повторил синтаксис регулярных выражений и их использование в Python библиотекой re. Использовал библиотеки datetime и time для замера времени исполнения программ.

**Источники:**

1. Лекция №4 «Языки разметки», автор П.В Балакшин.
2. Лекция №3 «Регулярные выражения», автор П.В Балакшин.