УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Информатика»

Отчет

По лабораторной работе №4

«Исследование протоколов, форматов обмена информацией и языков разметки документов»

Вариант 14

Студент

Рахимов И.И.

Р3119

Преподаватель

Рыбаков С. Д.

Санкт-Петербург, 2023 г.

**Содержание**

[**Описание заданий**](#_az45b5rqg8cc) **3**

[Описание выполнения заданий](#_vzmtyrbypnlm) 4

[Заключение 1](#_5l6ldv7i6b00)4

[Список используемой литературы 1](#_76879t4gakfc)5

# Описание заданий

**Обязательное задание**

Написать программу на языке Python 3.x, которая бы осуществляла парсинг и конвертацию исходного файла в формате JSON в новый в формате YAML 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 отчете.

**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 отчете.

**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 отчете.

## Описание выполнения заданий

**Выполнение обязательного занятия**

Парсер:

| **import** re  ind = 0   **def** change\_file(file):  text = **''**  flag = **False**  **for** line **in** file.readlines(): *# Убираем ненужную табуляцию и тд*  **for** i **in** range(1, len(line) - 1):  **if** line[i] == **'\"'**:  flag = **not** flag  text += line[i]  **elif** ((line[i] == **' '** **and** flag) **or** (line[i] != **' '**) **or** (line[i] == **' '** **and** line[i - 1] == **':'**)) **and** (  line[i] != **'\t'**):  text += line[i]  **return** text   **def** value\_checker(value):  **if** **not** (isinstance(value, str)):  **return** value  **if** value == **'true'**:  value = **True**  **elif** value == **'False'**:  value = **False**  **elif** value == **'null'**:  value = **None**  **elif** re.fullmatch(**r'(?:(?:\b| )(-?\d+[.]?(?:\d+)?\b))'**, value):  **if** **'.'** **in** value:  value = float(value)  **else**:  value = int(value)  **return** value   **def** add\_seg(text):  **global** ind  **if** text[ind] == **'['**:  doc = []  key = 0  ind += 1  **else**:  doc = {}  key = **None**  ind += 1  value = **None**  el = **''**  **while** ind < len(text):  **if** text[ind] == **':'**:  key = el.replace(**'\"'**, **''**)  el = **''**  **elif** text[ind] == **'{'**:  value = add\_seg(text)  **elif** text[ind] == **'}'**:  **if** value **is** **None**:  value = el  doc[key] = value\_checker(value)  **return** doc  **elif** text[ind] == **'['**:  value = add\_seg(text)  **elif** text[ind] == **']'**:  **if** value **is** **None**:  value = el  doc.append(value\_checker(value))  **return** doc  **elif** text[ind] == **','**:  **if** value **is** **None**:  value = el  **if** isinstance(doc, list):  key += 1  doc.append(value\_checker(value))  **else**:  doc[key] = value\_checker(value)  value = **None**  el = **''**  **elif** text[ind] != **'"'** **or** (text[ind] == **'"'** **and** el != **''** **and** el[-1] == **'\\'**):  **if** **not** (el == **''** **and** text[ind] == **' '**):  el += text[ind]  ind += 1  doc[key] = value\_checker(value)  **return** doc   **def** parser(path\_to\_json):  **global** ind  ind = 0  json\_file = open(path\_to\_json, **'r'**, encoding=**'utf-8'**)  text = change\_file(json\_file)  data = add\_seg(text)  **return** data |
| --- |

Конвертер:

| **import** my\_parser  cbrace\_counter = 0 nbrace\_counter = 0 sbrace\_counter = 0 stack = [**''**, **''**]   **def** change\_file(file):  text = **''**  flag = **False**  **for** line **in** file.readlines(): *# Убираем ненужную табуляцию и тд*  **for** i **in** range(1, len(line) - 1):  **if** line[i] == **'\"'**:  flag = **not** flag  text += line[i]  **elif** ((line[i] == **' '** **and** flag) **or** (line[i] != **' '**) **or** (line[i] == **' '** **and** line[i - 1] == **':'**)) **and** (  line[i] != **'\t'**):  text += line[i]  **return** text   **def** linefeed\_checker(line, flag, yaml\_file):  **global** cbrace\_counter, nbrace\_counter, sbrace\_counter, stack  **if** line != (**' '** \* len(line)) **and** line[-2:] != **'- '**:  **if** **'\":'** **in** line:  line = line.replace(**'\"'**, **''**, 2)  yaml\_file.write(line + **'\n'**)  **if** stack[-1] == **'['** **or** (stack[-2] == **'['** **and** stack[-1] == **'{'** **and** flag):  line = **' '** \* (cbrace\_counter + sbrace\_counter - 1) + **'- '**  **else**:  line = **' '** \* (cbrace\_counter + sbrace\_counter)  **return** line   **def** converter(path\_to\_json, path\_to\_yaml):  **try**:  my\_parser.parser(path\_to\_json)  **except**:  print(**"Ваш json файл очень кривой, сделайте его нормальным!!!!!!"**)  json\_file = open(path\_to\_json, **'r'**, encoding=**'utf-8'**)  yaml\_file = open(path\_to\_yaml, **'w'**, encoding=**'utf-8'**)  text = change\_file(json\_file)  **global** cbrace\_counter, nbrace\_counter, sbrace\_counter, stack  line = **''**  **for** ch **in** text:  **if** ch == **'{'**:  stack.append(**'{'**)  **if** stack[-2] != **'['**:  cbrace\_counter += 1  line = linefeed\_checker(line, **False**, yaml\_file)  **else**:  line = linefeed\_checker(line, **True**, yaml\_file)  **elif** ch == **'}'**:  **if** stack[-2] != **'['**:  cbrace\_counter -= 1  stack.pop()  line = linefeed\_checker(line, **False**, yaml\_file)  **elif** ch == **'['**:  sbrace\_counter += 1  stack.append(**'['**)  line = linefeed\_checker(line, **False**, yaml\_file)  **elif** ch == **']'**:  sbrace\_counter -= 1  stack.pop()  line = linefeed\_checker(line, **False**, yaml\_file)  **elif** ch == **','**:  **if** **'\\n'** **in** line:  d = **'|'**  **if** line[-2:] != **'n\"'**:  d += **'-'**  **else**:  line = line[:len(line) - 3] + **'\"'**  line = line.replace(**':'**, **f': {d}\n '**, 1)  line = line.replace(**'\\n'**, **f'\n {" " \* (cbrace\_counter + sbrace\_counter)}'**)  line = line.replace(**'\"'**, **''**)  line = linefeed\_checker(line, **False**, yaml\_file)  **else**:  line += ch  linefeed\_checker(line, **False**, yaml\_file)  json\_file.close()  yaml\_file.close()  cbrace\_counter = 0  nbrace\_counter = 0  sbrace\_counter = 0  stack = [**''**, **''**] |
| --- |

**Дополнительное задание №1**

парсер:

| **import** json   **def** parser(path\_to\_json):  **with** open(path\_to\_json, **'r'**, encoding=**'utf-8'**) **as** json\_file:  data = json.load(json\_file)  **return** data |
| --- |

конвертер:

| **import** json **import** yaml   **def** converter(path\_to\_json, path\_to\_yaml):  **with** (open(path\_to\_json, **'r'**, encoding=**'utf-8'**) **as** json\_file,  open(path\_to\_yaml, **"w"**, encoding=**'utf-8'**) **as** yaml\_file):  json\_payload = json.load(json\_file)  yaml.dump(json\_payload, yaml\_file, sort\_keys=**False**, allow\_unicode=**True**) |
| --- |

JSON файл c расписанием:

| {  "day": **"13.12.2023"**, "schedule": [  {  "typeOfActivity": **"Лекция"**,  "name": **"Программирование"**,  "teacher": **"Письмак Алексей Евгеньевич"**,   "place": **"Ауд. Актовый зал (1216/0 (усл)), ул.Ломоносова, д.9, лит. М"**,   "time": **"10:00-11:30"**,  "comment": **"Актовый зал"**  },  {  "typeOfActivity": **"Лабораторная"**,  "name": **"Информатика"**,  "teacher": **"Рыбаков Степан Дмитриевич"**,  "place": **"Ауд. 2112, Кронверкский пр., д.49, лит.А"**,  "time": **"13:30-15:00"**,  "comment": **""**  },  {  "typeOfActivity": **"Лабораторная"**,  "name": **"Информатика"**,  "teacher": **"Рыбаков Степан Дмитриевич"**,  "place": **"Ауд. 2112, Кронверкский пр., д.49, лит.А"**,  "time": **"15:20-16:50"**,  "comment": **""**  }  ] } |
| --- |

YAML файл полученный с помощью моего конвертора:

| day: **"13.12.2023"** schedule:  - typeOfActivity: **"Лекция"**  name: **"Программирование"**  teacher: **"Письмак Алексей Евгеньевич"**  place: **"Ауд. Актовый зал (1216/0 (усл)) ул.Ломоносова д.9 лит. М"**  time: **"10:00-11:30"**  comment: **"Актовый зал"** - typeOfActivity: **"Лабораторная"**  name: **"Информатика"**  teacher: **"Рыбаков Степан Дмитриевич"**  place: **"Ауд. 2112 Кронверкский пр. д.49 лит.А"**  time: **"13:30-15:00"**  comment: **""** - typeOfActivity: **"Лабораторная"**  name: **"Информатика"**  teacher: **"Рыбаков Степан Дмитриевич"**  place: **"Ауд. 2112 Кронверкский пр. д.49 лит.А"**  time: **"15:20-16:50"**  comment: **""** |
| --- |

YAML файл полученный с помощью конвертора из библиотек:

| day: 13.12.2023 schedule: - typeOfActivity: **Лекция**  name: **Программирование**  teacher: **Письмак** **Алексей** **Евгеньевич**  place: **Ауд.** **Актовый** **зал** **(1216/0** **(усл)),** **ул.Ломоносова,** **д.9,** **лит.** **М**  time: 10**:00-11:30**  comment: **Актовый** **зал** - typeOfActivity: **Лабораторная**  name: **Информатика**  teacher: **Рыбаков** **Степан** **Дмитриевич**  place: **Ауд.** 2112**,** **Кронверкский** **пр.,** **д.49,** **лит.А**  time: 13**:30-15:00**  comment: **''** - typeOfActivity: **Лабораторная**  name: **Информатика**  teacher: **Рыбаков** **Степан** **Дмитриевич**  place: **Ауд.** 2112**,** **Кронверкский** **пр.,** **д.49,** **лит.А**  time: 15**:20-16:50**  comment: **''** |
| --- |

Результаты парсинга оказались одинаковыми и выглядят вот так:

| {**'day'**: **'13.12.2023'**, **'schedule'**: [{**'typeOfActivity'**: **'Лекция'**, **'name'**: **'Программирование'**, **'teacher'**: **'Письмак Алексей Евгеньевич'**, **'place'**: **'лит. М'**, **'00-11'**: 30, **'comment'**: **'Актовый зал'**}, {**'typeOfActivity'**: **'Лабораторная'**, **'name'**: **'Информатика'**, **'teacher'**: **'Рыбаков Степан Дмитриевич'**, **'place'**: **'лит.А'**, **'30-15'**: 0, **'comment'**: **''**}, {**'typeOfActivity'**: **'Лабораторная'**, **'name'**: **'Информатика'**, **'teacher'**: **'Рыбаков Степан Дмитриевич'**, **'place'**: **'лит.А'**, **'20-16'**: 50, **'comment'**: **''**}]} |
| --- |

Сходства парсинга можно объяснить тем, что парсинг в структуры данных Python строгий и нельзя записать данные двумя разными способами.

Различия конвертаций в том, что библиотечный конвертор сканирует строки на наличие экранируемых элементов и применяет кавычки для экранизация только если это необходимо. Мой же конвертор всегда экранирует строки.

**Дополнительное задание № 2**

Парсер с добавлением регулярных выражений

| **import** re  ind = 0   **def** change\_file(file):  text = **''**  flag = **False**  **for** line **in** file.readlines(): *# Убираем ненужную табуляцию и тд*  **for** i **in** range(1, len(line) - 1):  **if** line[i] == **'\"'**:  flag = **not** flag  text += line[i]  **elif** ((line[i] == **' '** **and** flag) **or** (line[i] != **' '**) **or** (line[i] == **' '** **and** line[i - 1] == **':'**)) **and** (  line[i] != **'\t'**):  text += line[i]  **return** text   **def** value\_checker(value):  **if** **not** (isinstance(value, str)):  **return** value  **if** value == **'true'**:  value = **True**  **elif** value == **'False'**:  value = **False**  **elif** value == **'null'**:  value = **None**  **elif** re.fullmatch(**r'(?:(?:\b| )(-?\d+[.]?(?:\d+)?\b))'**, value):  **if** **'.'** **in** value:  value = float(value)  **else**:  value = int(value)  **return** value   **def** add\_seg(text):  **global** ind  **if** text[ind] == **'['**:  doc = []  key = 0  ind += 1  **else**:  doc = {}  key = **None**  ind += 1  value = **None**  el = **''**  **while** ind < len(text):  **if** text[ind] == **':'**:  key = el.replace(**'\"'**, **''**)  el = **''**  **elif** text[ind] == **'{'**:  value = add\_seg(text)  **elif** text[ind] == **'}'**:  **if** value **is** **None**:  value = el  doc[key] = value\_checker(value)  **return** doc  **elif** text[ind] == **'['**:  value = add\_seg(text)  **elif** text[ind] == **']'**:  **if** value **is** **None**:  value = el  doc.append(value\_checker(value))  **return** doc  **elif** text[ind] == **','**:  **if** value **is** **None**:  value = el  **if** isinstance(doc, list):  key += 1  doc.append(value\_checker(value))  **else**:  doc[key] = value\_checker(value)  value = **None**  el = **''**  **elif** text[ind] != **'"'** **or** (text[ind] == **'"'** **and** el != **''** **and** el[-1] == **'\\'**):  **if** **not** (el == **''** **and** text[ind] == **' '**):  el += text[ind]  ind += 1  doc[key] = value\_checker(value)  **return** doc   **def** parser(path\_to\_json):  **global** ind  ind = 0  json\_file = open(path\_to\_json, **'r'**, encoding=**'utf-8'**)  text = change\_file(json\_file)  data = add\_seg(text)  **return** data |
| --- |

Результаты парсинга оказались одинаковыми и выглядят вот так:

| {**'day'**: **'13.12.2023'**, **'schedule'**: [{**'typeOfActivity'**: **'Лекция'**, **'name'**: **'Программирование'**, **'teacher'**: **'Письмак Алексей Евгеньевич'**, **'place'**: **'Ауд. Актовый зал (1216/0 (усл)), ул.Ломоносова, д.9, лит. М'**, **'time'**: **'10:00-11:30'**, **'comment'**: **'Актовый зал'**}, {**'typeOfActivity'**: **'Лабораторная'**, **'name'**: **'Информатика'**, **'teacher'**: **'Рыбаков Степан Дмитриевич'**, **'place'**: **'Ауд. 2112, Кронверкский пр., д.49, лит.А'**, **'time'**: **'13:30-15:00'**, **'comment'**: **''**}, {**'typeOfActivity'**: **'Лабораторная'**, **'name'**: **'Информатика'**, **'teacher'**: **'Рыбаков Степан Дмитриевич'**, **'place'**: **'Ауд. 2112, Кронверкский пр., д.49, лит.А'**, **'time'**: **'15:20-16:50'**, **'comment'**: **''**}]} |
| --- |

Сходства парсинга можно объяснить тем, что парсинг в структуры данных Python строгий и нельзя записать данные двумя разными способами.

**Дополнительное задание №4**

Код для проверки скорости работы парсеров и конверторов:

| **import** lib\_parser **import** lib\_converter **import** my\_parser **import** my\_converter **import** time   **def** measurement(path\_to\_json, path\_to\_yaml1, path\_to\_yaml2):  t\_start = time.time()  **for** i **in** range(100):  my\_parser.parser(path\_to\_json)  t\_finish = time.time()  print(**f"\tСкорость работы моего парсера: {t\_finish - t\_start}"**)  t\_start = time.time()  **for** i **in** range(100):  my\_converter.converter(path\_to\_json, path\_to\_yaml1)  t\_finish = time.time()  print(**f"\tСкорость работы моего конвертора: {t\_finish - t\_start}"**)  t\_start = time.time()  **for** i **in** range(100):  lib\_parser.parser(path\_to\_json)  t\_finish = time.time()  print(**f"\tСкорость работы библиотечного парсера: {t\_finish - t\_start}"**)  t\_start = time.time()  **for** i **in** range(100):  lib\_converter.converter(path\_to\_json, path\_to\_yaml2)  t\_finish = time.time()  print(**f"\tСкорость библиотечного конвертора: {t\_finish - t\_start}"**) |
| --- |

Данные:

Скорость работы моего парсера: 0.05360245704650879

Скорость работы моего конвертора: 0.139817476272583

Скорость работы библиотечного парсера: 0.008169174194335938

Скорость библиотечного конвертора: 0.16286063194274902

Как видим скорость работы библиотечного парсера примерно в 6,5 раз меньше. Это может быть вызвано использованием различных оптимизаций в коде в библиотеке

Скорость работы же моего конвертора примерно на 20% меньше. Вызвано это может быть наличием распознавания ошибок в библиотечном конверторе.

### Заключение

В процессе решения задач из лабораторной работы, я многое узнал о протоколах, форматах обмена информацией и языках разметки документов, в том числе научился работать с форматами обмена информацией JSON и YAML. Также я немного узнал про способы конвертации и парсинга различной информации. Приобрел некоторый опыт в написании подобных программ. Также спустя века страданий и мучений входе выполнения работы, я полностью овладел Microsoft Word и научился создавать все отчеты, соответствующие всем ГОСТам и требованиям. Очень надеюсь, что эти знания мне хоть раз понадобятся в жизни.

#### 

#### 

#### 

#### Список используемой литературы

1. Лямин А.В., Череповская Е.Н. Объектно-ориентированное программирование. Компьютерный практикум. – СПб: Университет 5 ИТМО, 2017. – 143 с. – Режим доступа: https://books.ifmo.ru/file/pdf/2256.pdf.
2. «Пишем изящный парсер на Питоне» Режим доступа: https://habr.com/ru/post/309242/.