МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

**ЛАБОРАТОРНАЯ РАБОТА №2**

по дисциплине

«ИНФОРМАТИКА»

Вариант № 21

***Выполнил:***

Студент группы P3117

Хороших Дмитрий

Максимович

***Преподаватель:***

Машина Екатерина

А.

# Содержание

[Содержание 2](#_Toc115941141)

[Задание 3](#_Toc115941142)

[Схемы декодирования 4](#_Toc115941143)

[Основные этапы вычисления 6](#_Toc115941144)

[Список литературы 9](#_Toc115941145)

[Вывод 10](#_Toc115941146)

Выбор варианта

Порядковый номер в группе: 17, занятия проводятся по: вторникам и четвергам.

**Итоговый вариант: 21**

Задание

1. Исходя из структуры расписания конкретного дня, сформировать файл с расписанием в формате, указанном в задании в качестве исходного.
2. *Обязательное задание*. Написать программу на языке Python 3.x, которая бы осуществляла парсинг и конвертацию исходного файла в новый.
3. *Дополнительное задание №1*. Выполнить следующие пункты:
   1. Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.
   2. Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать.
   3. Сравнить полученные результаты и объяснить их сходство/различие.
4. *Дополнительное задание №2*. Выполнить следующие пункты:
   1. Переписать исходный код, добавив в него использование регулярных выражений.
   2. Сравнить полученные результаты и объяснить их сходство/различие.
5. *Дополнительное задание №3*. Выполнить следующие пункты:
   1. Используя свою исходную программу из обязательного задания, программу из дополнительного задания №1 и программу из дополнительного задания №2, сравнить стократное время выполнения парсинга + конвертации в цикле.
   2. Проанализировать полученные результаты и объяснить их сходство/различие.
6. *Дополнительное задание №4*. Выполнить следующие пункты:
   1. 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. Проанализировать полученные результаты, объяснить особенности использования формата.

|  |  |  |
| --- | --- | --- |
| Исходный формат | Результирующий формат | День недели |
| YAML | JSON | Четверг |

Основные этапы выполнения

1. *Обязательное задание*.

**yamlParser.py**

'''

From Mitya with love

and bad readability

'''

def getLevel(str):

     return len(str) - len(str.lstrip())

def isChild(str1, str2):

    return getLevel(str1)< getLevel(str2)

def isItem(line):

    words = line.strip().split()

    return words[0] == '-'

def parseInt(text):

    if text.isnumeric():

        if text.count('.') != 0:

            return float(text)

        else:

            return int(text)

    return None

def parseSpecs(text):

    if text == 'true':

            return True

    elif text == 'false':

        return False

    return None

def parseText(text):

    if text[0] + text[-1] in ('\"\"', '\'\''):

        return text[1:-1]

    return text

def parseObj(text):

    a = parseSpecs(text)

    if a != None:

        return a

    a = parseInt(text)

    if a != None:

        return a

    return parseText(text)

def isSameLevel(line1, line2):

    return getLevel(line1.replace('-', ' ',1)) == getLevel(line2.replace('-', ' ',1))

def parse(root, level, lines):

    if len(lines) == 0:

        return

    line = lines.pop()

    words = line.strip().split()

    # - smth

    if words[0] == '-':

        # - smth:

        if words[1][-1] == ':':

            words = words[1:]

            # - smth: smth

            if len(words) != 1:

                newRoot = {(words[0][:-1]) : parseObj(' '.join(words[1:]))}

                if isSameLevel(line, lines[-1]) and lines[-1].rstrip()[0] != '-':

                    while len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                        parse(newRoot, getLevel(line), lines)

                    root.append(newRoot)

                else:

                    root.append(newRoot)

            # - smth:

            elif isChild(line, lines[-1]):

                nextL = lines[-1]

                if isItem(nextL):

                    if len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                        tempRoot = list()

                        parse(tempRoot, getLevel(line), lines)

                        newRoot = {(words[0][:-1]) : tempRoot}

                        while len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                            parse(newRoot, getLevel(line), lines)

                        root.append(newRoot)

                    else:

                        newRoot = list()

                        parse(newRoot, getLevel(line), lines)

                        root.append({words[0][:-1] : tuple(newRoot)})

                else:

                    if len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                        tempRoot = dict()

                        parse(tempRoot, getLevel(line), lines)

                        newRoot = {(words[0][:-1]) : tempRoot}

                        while len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                            parse(newRoot, getLevel(line), lines)

                        root.append(newRoot)

                    else:

                        newRoot = dict()

                        parse(newRoot, getLevel(line), lines)

                        root.append({words[0][:-1] : newRoot})

        # - smth

        else:

            root.append(parseObj(' '.join(words[1:])))

    # smth:

    elif words[0][-1] == ':':

            if len(words) != 1:

                root[words[0][:-1]] = parseObj(' '.join(words[1:]))

            elif isChild(line, lines[-1]):

                nextL = lines[-1]

                if isItem(nextL):

                    newRoot = list()

                    parse(newRoot, getLevel(line), lines)

                    root[words[0][:-1]] = tuple(newRoot)

                else:

                    newRoot = dict()

                    parse(newRoot, getLevel(line), lines)

                    root[words[0][:-1]] = newRoot

    if len(lines) != 0 and level >= getLevel(lines[-1]):

        return

    parse(root, level, lines)

def ParseText(text):

    # yamlLines = open("itmo.yaml", encoding="utf8").read().split('\n')

    yamlLines = text.split('\n')

    yamlLines = [i for i in yamlLines if len(i.lstrip()) != 0 and i.lstrip()[0] != '#' ]

    yamlLines = yamlLines[1:]

    yamlLines.reverse()

    fileRoot = dict()

    parse(fileRoot, 0 ,yamlLines)

    return fileRoot

**yamlTojson.py**

from yamlParser import ParseText

level = 0

def parseValue(val):

    if type(val) in [int, float]:

        return str(val)

    if val in (True, False, None):

        if val is True:

            return "true"

        elif val is False:

            return "false"

        else:

            return "null"

    else:

        return f'\"{val}\"'

def getJSON(root):

    global level

    json = ""

    # json += '{\n'

    # level+=1

    for i in  range(len(root.keys())):

        val = root[list(root.keys())[i]]

        json += '  ' \* level + "\"" + str(list(root.keys())[i]) + "\"" + ": "

        if type(val) is dict:

            json += '{\n'

            level+=1

            json+=getJSON(val)

            json += '\n' +'  '\*level + '}'

            level-=1

        elif type(val) is tuple:

            json += '[\n'

            level+=1

            for j in range(len(val)):

                item = val[j]

                if type(item) is dict:

                    json+= '  '\*level+'{\n'

                    level+=1

                    json+=getJSON(item)

                    json+= '\n' + '  '\*level+'}'

                    level-=1

                else:

                    json += '  '\*level + parseValue(item)

                if j == len(val) -1:

                    json+='\n'

                else:

                    json+=',\n'

            json += '\n' +'  '\*level + ']'

            level-=1

        else:

            json+=parseValue(val)

        if i != len(root.keys())-1:

            json+=',\n'

    return json

def main():

    level = 0

    file = open('itmo.yaml', encoding='utf8')

    rootDict = ParseText(file.read())

    json = '{\n'

    level+=1

    json+= getJSON(rootDict)

    json+= '\n}'

    f = open('itmoDefault.json', 'w', encoding='utf8')

    f.write(json)

**itmo.yaml**

---

time-table:

#Расписание на четверик! Хороший день

#вообще-то!

  - thursday:

    pares:

      - pare1:

         time: "08:20-09:50"

         weeks:

           - 2

           - 4

           - 6

           - 8

           - 10

           - 12

           - 14

           - 16

         where:

           adress: "Кронверкский пр., д.49, лит.А"

           room: "2435/1(БЫВШ. 431А) АУД."

         lesson:

           name: "ИНФОРМАТИКА(ЛАБ)"

           even-week: true

           professor: "Машина Екатерина Алексеевна"

         format: "Очно-дистанционный"

      - pare2:

         time: "10:00-11:30"

         weeks:

           - 2

           - 4

           - 6

           - 8

           - 10

           - 12

           - 14

           - 16

         where:

           adress: "Кронверкский пр., д.49, лит.А"

           room: "2435/1(БЫВШ. 431А) АУД."

         lesson:

           name: "ИНФОРМАТИКА(ЛАБ)"

           even-week: true

           professor: "Машина Екатерина Алексеевна"

         format: "Очно-дистанционный"

      - pare3:

         time: "11:40-13:10"

         weeks:

           - 2

           - 4

           - 6

           - 8

           - 10

           - 12

           - 14

           - 16

         where:

           adress: "Кронверкский пр., д.49, лит.А"

           room: "1328(БЫВШ. 371) АУД."

         lesson:

           name: "ПРОГРАММИРОВАНИЕ(ЛАБ)"

           even-week: true

           professor: "Письмак Алексей Евгеньевич"

         format: "Очно-дистанционный"

      - pare4:

         time: "13:30-15:00"

         weeks:

           - 2

           - 4

           - 6

           - 8

           - 10

           - 12

           - 14

           - 16

         where:

           adress: "Кронверкский пр., д.49, лит.А"

           room: "1328(БЫВШ. 371) АУД."

         lesson:

           name: "ПРОГРАММИРОВАНИЕ(ЛАБ)"

           even-week: true

           professor: "Письмак Алексей Евгеньевич"

         format: "Очно-дистанционный"

  - friday:

    pares:

      - pare1:

         time: "08:20-09:50"

         weeks:

           - 2

           - 4

           - 6

           - 8

           - 10

           - 12

           - 14

           - 16

         where:

           adress: "Кронверкский пр., д.49, лит.А"

           room: "2435/1(БЫВШ. 431А) АУД."

         lesson:

           name: "ИНФОРМАТИКА(ЛАБ)"

           even-week: true

           professor: "Машина Екатерина Алексеевна"

         format: "Очно-дистанционный"

**itmo.json**

{

"time-table": [

  {

    "thursday": {

      "pares": [

        {

          "pare1": {

            "time": "08:20-09:50",

            "weeks": [

              2,

              4,

              6,

              8,

              10,

              12,

              14,

              16

              ],

            "where": {

              "adress": "Кронверкский пр., д.49, лит.А",

              "room": "2435/1(БЫВШ. 431А) АУД."

              },

            "lesson": {

              "name": "ИНФОРМАТИКА(ЛАБ)",

              "even-week": true,

              "professor": "Машина Екатерина Алексеевна"

              },

            "format": "Очно-дистанционный"

            }

          },

        {

          "pare2": {

            "time": "10:00-11:30",

            "weeks": [

              2,

              4,

              6,

              8,

              10,

              12,

              14,

              16

              ],

            "where": {

              "adress": "Кронверкский пр., д.49, лит.А",

              "room": "2435/1(БЫВШ. 431А) АУД."

              },

            "lesson": {

              "name": "ИНФОРМАТИКА(ЛАБ)",

              "even-week": true,

              "professor": "Машина Екатерина Алексеевна"

              },

            "format": "Очно-дистанционный"

            }

          },

        {

          "pare3": {

            "time": "11:40-13:10",

            "weeks": [

              2,

              4,

              6,

              8,

              10,

              12,

              14,

              16

              ],

            "where": {

              "adress": "Кронверкский пр., д.49, лит.А",

              "room": "1328(БЫВШ. 371) АУД."

              },

            "lesson": {

              "name": "ПРОГРАММИРОВАНИЕ(ЛАБ)",

              "even-week": true,

              "professor": "Письмак Алексей Евгеньевич"

              },

            "format": "Очно-дистанционный"

            }

          },

        {

          "pare4": {

            "time": "13:30-15:00",

            "weeks": [

              2,

              4,

              6,

              8,

              10,

              12,

              14,

              16

              ],

            "where": {

              "adress": "Кронверкский пр., д.49, лит.А",

              "room": "1328(БЫВШ. 371) АУД."

              },

            "lesson": {

              "name": "ПРОГРАММИРОВАНИЕ(ЛАБ)",

              "even-week": true,

              "professor": "Письмак Алексей Евгеньевич"

              },

            "format": "Очно-дистанционный"

            }

          }

        ]

      }

    },

  {

    "friday": {

      "pares": [

        {

          "pare1": {

            "time": "08:20-09:50",

            "weeks": [

              2,

              4,

              6,

              8,

              10,

              12,

              14,

              16

              ],

            "where": {

              "adress": "Кронверкский пр., д.49, лит.А",

              "room": "2435/1(БЫВШ. 431А) АУД."

              },

            "lesson": {

              "name": "ИНФОРМАТИКА(ЛАБ)",

              "even-week": true,

              "professor": "Машина Екатерина Алексеевна"

              },

            "format": "Очно-дистанционный"

            }

          }

        ]

      }

    }

  ]

}

1. *Дополнительное задание №1.*

**yamlTojsonLIBS.py**

import yaml

import json

def main():

    dataDict = {}

    f = open('itmo.yaml', encoding='utf8')

    dataDict = yaml.load(f, Loader=yaml.FullLoader)

    n = open('itmoByLib.json', 'w', encoding='utf8')

    (json.dump(dataDict, n, indent=4,  ensure\_ascii=False))

**itmoByLib.json**

{

    "time-table": [

        {

            "thursday": null,

            "pares": [

                {

                    "pare1": {

                        "time": "08:20-09:50",

                        "weeks": [

                            2,

                            4,

                            6,

                            8,

                            10,

                            12,

                            14,

                            16

                        ],

                        "where": {

                            "adress": "Кронверкский пр., д.49, лит.А",

                            "room": "2435/1(БЫВШ. 431А) АУД."

                        },

                        "lesson": {

                            "name": "ИНФОРМАТИКА(ЛАБ)",

                            "even-week": true,

                            "professor": "Машина Екатерина Алексеевна"

                        },

                        "format": "Очно-дистанционный"

                    }

                },

                {

                    "pare2": {

                        "time": "10:00-11:30",

                        "weeks": [

                            2,

                            4,

                            6,

                            8,

                            10,

                            12,

                            14,

                            16

                        ],

                        "where": {

                            "adress": "Кронверкский пр., д.49, лит.А",

                            "room": "2435/1(БЫВШ. 431А) АУД."

                        },

                        "lesson": {

                            "name": "ИНФОРМАТИКА(ЛАБ)",

                            "even-week": true,

                            "professor": "Машина Екатерина Алексеевна"

                        },

                        "format": "Очно-дистанционный"

                    }

                },

                {

                    "pare3": {

                        "time": "11:40-13:10",

                        "weeks": [

                            2,

                            4,

                            6,

                            8,

                            10,

                            12,

                            14,

                            16

                        ],

                        "where": {

                            "adress": "Кронверкский пр., д.49, лит.А",

                            "room": "1328(БЫВШ. 371) АУД."

                        },

                        "lesson": {

                            "name": "ПРОГРАММИРОВАНИЕ(ЛАБ)",

                            "even-week": true,

                            "professor": "Письмак Алексей Евгеньевич"

                        },

                        "format": "Очно-дистанционный"

                    }

                },

                {

                    "pare4": {

                        "time": "13:30-15:00",

                        "weeks": [

                            2,

                            4,

                            6,

                            8,

                            10,

                            12,

                            14,

                            16

                        ],

                        "where": {

                            "adress": "Кронверкский пр., д.49, лит.А",

                            "room": "1328(БЫВШ. 371) АУД."

                        },

                        "lesson": {

                            "name": "ПРОГРАММИРОВАНИЕ(ЛАБ)",

                            "even-week": true,

                            "professor": "Письмак Алексей Евгеньевич"

                        },

                        "format": "Очно-дистанционный"

                    }

                }

            ]

        },

        {

            "friday": null,

            "pares": [

                {

                    "pare1": {

                        "time": "08:20-09:50",

                        "weeks": [

                            2,

                            4,

                            6,

                            8,

                            10,

                            12,

                            14,

                            16

                        ],

                        "where": {

                            "adress": "Кронверкский пр., д.49, лит.А",

                            "room": "2435/1(БЫВШ. 431А) АУД."

                        },

                        "lesson": {

                            "name": "ИНФОРМАТИКА(ЛАБ)",

                            "even-week": true,

                            "professor": "Машина Екатерина Алексеевна"

                        },

                        "format": "Очно-дистанционный"

                    }

                }

            ]

        }

    ]

}

1. *Дополнительное задание №2.*

**yamParserREGEX.py**

'''

From Mitya with love

and bad readability

'''

import re

def getLevel(str):

     return len(str) - len(str.lstrip())

def isChild(str1, str2):

    return getLevel(str1)< getLevel(str2)

itemPattern = re.compile(r'^\s\*-\s.\*$')

def isItem(line):

    return re.fullmatch(itemPattern, line) is not None

numPattern = re.compile(r'\b[\-\+]?\d+(\.\d+)?(E[\-\+]\d+)?\b')

def parseInt(text):

    if re.fullmatch(numPattern, text) is not None:

            if '.' in text:

                return float(text)

            else:

                return int(text)

    return None

def parseSpecs(text):

    if text == 'true':

            return True

    elif text == 'false':

        return False

    return None

literalPattern = re.compile(r'[\"\'][^\'\"]\*[\"\']')

def parseText(text):

    if re.fullmatch(literalPattern, text) is not None:

        return text[1:-1]

    return text

def parseObj(text):

    a = parseSpecs(text)

    if a != None:

        return a

    a = parseInt(text)

    if a != None:

        return a

    return parseText(text)

def isSameLevel(line1, line2):

    return getLevel(line1.replace('-', ' ',1)) == getLevel(line2.replace('-', ' ',1))

def parse(root, level, lines):

    if len(lines) == 0:

        return

    line = lines.pop()

    words = line.strip().split()

    # - smth

    if re.fullmatch(itemPattern, line) is not None:

        # - smth:

        if words[1][-1] == ':':

            words = words[1:]

            # - smth: smth

            if len(words) != 1:

                newRoot = {(words[0][:-1]) : parseObj(' '.join(words[1:]))}

                if isSameLevel(line, lines[-1]) and lines[-1].rstrip()[0] != '-':

                    while len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                        parse(newRoot, getLevel(line), lines)

                    root.append(newRoot)

                else:

                    root.append(newRoot)

            # - smth:

            elif isChild(line, lines[-1]):

                nextL = lines[-1]

                if isItem(nextL):

                    if len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                        tempRoot = list()

                        parse(tempRoot, getLevel(line), lines)

                        newRoot = {(words[0][:-1]) : tempRoot}

                        while len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                            parse(newRoot, getLevel(line), lines)

                        root.append(newRoot)

                    else:

                        newRoot = list()

                        parse(newRoot, getLevel(line), lines)

                        root.append({words[0][:-1] : tuple(newRoot)})

                else:

                    if len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                        tempRoot = dict()

                        parse(tempRoot, getLevel(line), lines)

                        newRoot = {(words[0][:-1]) : tempRoot}

                        while len(lines) != 0 and isSameLevel(line, lines[-1]) and lines[-1].strip()[0] != '-':

                            parse(newRoot, getLevel(line), lines)

                        root.append(newRoot)

                    else:

                        newRoot = dict()

                        parse(newRoot, getLevel(line), lines)

                        root.append({words[0][:-1] : newRoot})

        # - smth

        else:

            root.append(parseObj(' '.join(words[1:])))

    # smth:

    elif words[0][-1] == ':':

            if len(words) != 1:

                root[words[0][:-1]] = parseObj(' '.join(words[1:]))

            elif isChild(line, lines[-1]):

                nextL = lines[-1]

                if isItem(nextL):

                    newRoot = list()

                    parse(newRoot, getLevel(line), lines)

                    root[words[0][:-1]] = tuple(newRoot)

                else:

                    newRoot = dict()

                    parse(newRoot, getLevel(line), lines)

                    root[words[0][:-1]] = newRoot

    if len(lines) != 0 and level >= getLevel(lines[-1]):

        return

    parse(root, level, lines)

def ParseText(text):

    yamlLines = text.split('\n')

    yamlLines = [i for i in yamlLines if len(i.lstrip()) != 0 and i.lstrip()[0] != '#' ]

    yamlLines = yamlLines[1:]

    yamlLines.reverse()

    fileRoot = dict()

    parse(fileRoot, 0 ,yamlLines)

    return fileRoot

# jsonText = str(fileRoot).\

#     replace('(', '[').\

#     replace(')',']').\

#     replace('{', '{\n').\

#     replace('}', '\n}').\

#     replace('[', '[\n').\

#     replace(']', '\n]').\

#     replace('\'', '\"').\

#     replace('True', 'true').\

#     replace('False', 'false').\

#     replace('None', 'null')

# print(jsonText)

# f = open('test.json', 'x', encoding='utf8')

# f.write(jsonText)

1. *Дополнительное задание №3.*

**timetiter.py**

import timeit

print(timeit.timeit('import yamlTojson; yamlTojson.main()', number=100))

print(timeit.timeit('import yamlTojsonLIBS; yamlTojsonLIBS.main()', number=100))

print(timeit.timeit('import yamlTojsonREGEX; yamlTojsonREGEX.main()', number=100))

**Вывод timeiter.py**

*0.08615659992210567*

*1.0928360000252724*

*0.11317580007016659*

Следовательно время выполнения без использования библиотек и регулярных выражение меньше всего (из-за отсутствия необходимости генерации объектов регулярных выражений и проверок на корректность ввода). Время выполнения с использованием сторонних библиотек самое большое из-за двойного прохода по документу, совершаемого функциями с целью обеспечения корректности ввода.

1. *Дополнительное задание №4.*

**yamlToCSV.py**

import yamlParser

cols = dict()

file = open('table.yaml', encoding='utf8')

root = yamlParser.ParseText(file.read())

def findList(root):

    maxL = 0

    maxTuple = None

    if type(root) is tuple:

        return root

    else:

        for i in root.values():

            temp = findList(i)

            if temp is not None:

                if len(temp) > maxL:

                    maxTuple = temp

        return maxTuple

ls = findList(root)

print(findList(root))

for n in range(len(ls)):

    item = ls[n]

    if type(item) != dict:

        continue

    for key in item.keys():

        if key in cols.keys():

            cols[key].append(item[key])

        else:

            cols[key] = [None for i in range(n)]

            cols[key].append(item[key])

    for key in cols.keys():

        if key not in item.keys():

            cols[key].append(None)

file = open('table.csv', 'w', encoding='utf8')

file.write(', '.join(list(cols.keys())))

for i in range(len(cols[list(cols.keys())[0]])):

    file.write('\n')

    for j in range(len(list(cols.keys()))):

        if j  ==  len(list(cols.keys()))-1:

            file.write(str(cols[list(cols.keys())[j]][i]))

        else:

            file.write(str(cols[list(cols.keys())[j]][i]) + ', ')

**table.yaml**

---

race:

  entries:

    - id: 11

      name: Johnson Smith and Jones Co.

    - id: 22

    - id: 33

      name: Barney & Company

    - id: 44

      name: Johnson's Automotive

      food: Pasta

**table.cs**

id, name, food

11, Johnson Smith and Jones Co., None

22, None, None

33, Barney & Company, None

44, Johnson's Automotive, Pasta

Вывод

При выполнении лабораторной работы я отработал навыки работы с популярными языками разметки документов, написал несколько программ на языке Python, осуществляющие парсинг и конвертер для различных языков разметки. Всё это поможет в будущей работе с подобными документами.

# Список литературы

|  |  |
| --- | --- |
| [1] | А. В. Лямин и Е. Н. Череповская, Объектно-ориентированное программирование. Компьютерный практикум., Санкт-Петербург: Университет ИТМО, 2017. |
| [2] | Wikipedia, «Форма Бэкуса-Наура,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Форма\_Бэкуса\_—\_Наура. |