**Министерство образования и науки Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО”**

Факультет программной инженерии и компьютерной техники (ПИКТ)

Направление подготовки (специальность) – 09.03.04 (Нейротехнологии и программная инженерия)

Информатика

Регулярные выражения и языки разметки документов. Python

Лабораторная работа № 3

Выполнил студент

Слудная Виктория Евгеньевна

Группа №P3122

Преподаватель: Болдырева Елена Александровна

г. Санкт-Петербург

2024 г.

Оглавление

[**Задание:** 3](#_Toc180366150)

[Задание 1. 3](#_Toc180366151)

[Задание 2. 3](#_Toc180366152)

[Задание 3. 4](#_Toc180366153)

[**Отчет:** 5](#_Toc180366154)

[***Задание 1.*** 5](#_Toc180366155)

[***Задание 2*** 5](#_Toc180366156)

[***Задание 3*** 6](#_Toc180366157)

[***Задание 3.1*** 8](#_Toc180366158)

[***Задание 3.2*** 9](#_Toc180366159)

[**Вывод:** 10](#_Toc180366160)

[**Список литературы:** 10](#_Toc180366161)

**Вариант: 476011 = 61**

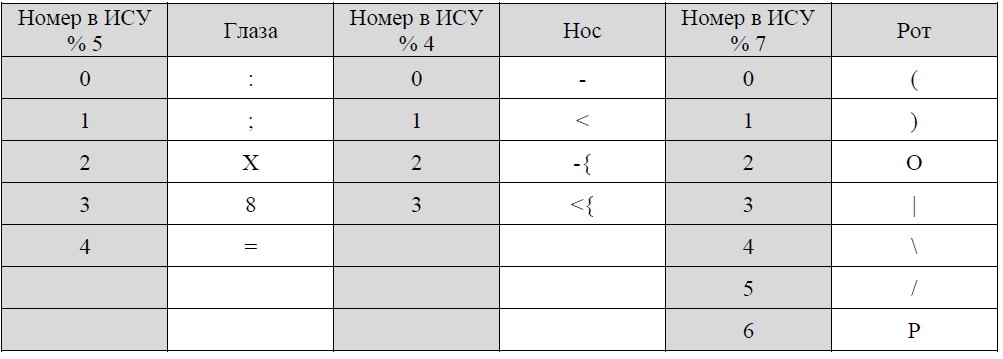
# **Задание:**

# Задание 1.

1. Реализуйте программный продукт на языке Python, используя регулярные выражения по варианту, представленному в таблице.
2. Для своей программы придумайте минимум 5 тестов. Каждый тест является отдельной сущностью, передаваемой регулярному выражению для обработки. Для каждого теста необходимо самостоятельно (без использования регулярных выражений) найти правильный ответ. После чего сравнить ответ, выданный программой, и полученный самостоятельно.
3. Программа должна считать количество смайликов определённого вида (вид смайлика описан в таблице вариантов) в предложенном тексте. Все смайлики имеют такую структуру:

[глаза][нос][рот].

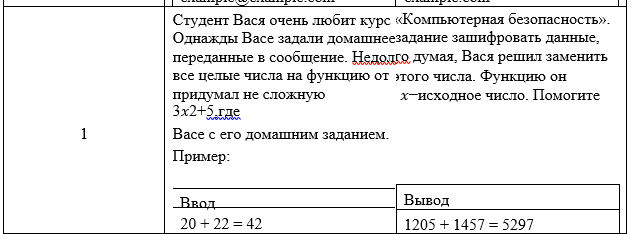
Вариантом является различные наборы глаз, носов и ртов.



*Рис. 1. Смайлики.*

# Задание 2.

1. Реализуйте программный продукт на языке Python, используя регулярные выражения по варианту, представленному в таблице.
2. Для своей программы придумайте минимум 5 тестов.
3. Протестируйте свою программу на этих тестах.



*Рис. 2. Задание 2.*

# Задание 3.

1. Определить номер варианта как остаток деления номера в ИСУ на 36. В случае, если в данный день недели нет занятий, то увеличить номер варианта на восемь.
2. Изучить форму Бэкуса-Наура.
3. Изучить особенности протоколов и форматов обмена информацией между системами: JSON, YAML, XML.
4. Понять устройство страницы с расписанием для своей группы:

https://itmo.ru/ru/schedule/0/P3110/schedule.htm

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. Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML-файлов.

1. **Дополнительное задание №1** (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
2. Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.
3. Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать.
4. Сравнить полученные результаты и объяснить их сходство/различие.

9. **Дополнительное задание №2** (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).

1. Переписать исходный код, добавив в него использование регулярных выражений.
2. Сравнить полученные результаты и объяснить их сходство/различие.



*Рис. 3. Вариант задания 3.*

# **Отчет:**

## ***Задание 1.***

*Исходный код:*

import sys

import re

sys.stdin = open('t1.txt')

eyes = [':', ';', 'X', ';', '8', '=']

nose = ['-', '<', '-{', '<{']

mouth = ['(', ')', 'O', '|', '\\', '/', 'P']

def makeSmileFace(isu):

return eyes[isu%5] + nose[isu%4] + mouth[isu%7]

def countAmmount (patt, str):

return len(re.findall(patt, str))

patt = r";<{\\"

print('Your smile face: ' + patt)

for \_ in range(5):

string = input()

print('This line: ',

string ,

' has ' ,

countAmmount(patt, string) ,

' smile face(s)')

*Пример ввода:*

:<O|\=8=-{;<{\\):<{P()|\/;<{\\X-):<{P:<{;<{\\P:<{P

8X:<{P{P-;<{\\<O:<|8-);<{\\:<{:<{PP:-{\:<{P

X<{/;<{\\P=-//:<{P:<{P:;<{\\<{PO\;<{\\;<{\\|:-O:<{P

::=--{(=-|:;<{\\<{PX:</=<{PX<):<{P8-{)

=<O:-{X<{|:-PX<O8-{\:-/;-P:;-{|8-/

*Пример вывода:*

Your smile face: ;<{\\

This line: :<O|\=8=-{;<{\\):<{P()|\/;<{\\X-):<{P:<{;<{\\P:<{P has 3 smile face(s)

This line: 8X:<{P{P-;<{\\<O:<|8-);<{\\:<{:<{PP:-{\:<{P has 2 smile face(s)

This line: X<{/;<{\\P=-//:<{P:<{P:;<{\\<{PO\;<{\\;<{\\|:-O:<{P has 4 smile face(s)

This line: ::=--{(=-|:;<{\\<{PX:</=<{PX<):<{P8-{) has 1 smile face(s)

This line: =<O:-{X<{|:-PX<O8-{\:-/;-P:;-{|8-/ has 0 smile face(s)

При ручном подсчете получились следующие результаты: 3, 2, 4, 1, 0.

## ***Задание 2***

*Исходный код:*

import re

import sys

sys.stdin = open('t2.txt')

#Функция для вычисления 3x^2 + 5

def transform(match):

#re возвращает объект-группу когда появляется соотвествие, работаем с ним

num = int(match.group(0))

return str(3 \* num \*\* 2 + 5)

def functionTransformer(message):

#Используем регулярное выражение для поиска всех целых чисел

#-? - для отрицательных и положительных чисел, d+ - все цифры числа(как миниум 1)

return re.sub(r'-?\d+', transform, message)

for \_ in range(5):

text = input()

print(f"Your text: " + text + " now is " + functionTransformer(text))

*Пример ввода:*

20 + 22 = 42

3 \* 5 - 10

The value is -4 for x

1 + 2 + 3

1000000 / 333333 + 1

*Пример вывода:*

Your text: 20 + 22 = 42 now is 1205 + 1457 = 5297

Your text: 3 \* 5 - 10 now is 32 \* 80 - 305

Your text: The value is -4 for x now is The value is 53 for x

Your text: 1 + 2 + 3 now is 8 + 17 + 32

Your text: 1000000 / 333333 + 1 now is 3000000000005 / 333332666672 + 8

Предполагаемый вывод для первого (подсчитан на калькуляторе): 1205 + 1457 = 5297, последующие ответы также совпадают.

## ***Задание 3***

*Исходный код:*

import sys

yaml\_file = open('t3.txt', 'r', encoding='utf-8')

print("{")

def wordOarser(str):

words = []

word = ""

#флаг-проверка на наличие двоеточия

has\_two\_points = False

for i in range(len(str)):

#если еще не было двоеточия

if(has\_two\_points == False):

if(str[i] == ':') :

if(word != ""):

words.append("\"" + word + "\"")

word = ""

words.append(":")

has\_two\_points = True

if(str[i] != ':' and str[i] != " " and str[i] != "\n"):

word += str[i]

if str[i] == " " or str[i] == "\n":

if(word != ""):

words.append(word)

word = ""

#было двоеточие. дальше нет смыслап делить строку на слова

else :

if(str[i] != ':' and str[i] != "\n"):

word += str[i]

if str[i] == "\n":

if(word != ""):

while(word[0] == ' ') :

word = word[1:]

words.append("\"" + word + "\"")

word = ""

return words

#пробелы в оригинале

flag = 0

#наличие { в конце строки

a\_flag = 0

first\_ = 0

def toJson(str):

global flag

global a\_flag

global first\_

count\_of\_space = 0;

while str[0] == ' ':

count\_of\_space +=1

str = str[1:]

# уменьшается количество пробелов = закрываются скобки

if(count\_of\_space < flag):

print("\n", (count\_of\_space\*4)\*' ', " ", "},", end ='')

else:

if(a\_flag == 0 and first\_ == 1):

print(",", end='', sep='')

flag = count\_of\_space

json\_ = ""

words\_ = wordOarser(str)

a\_flag=0

#если после : нет слов - это начало нового блока

if(words\_[len(words\_)-1] == ":"):

words\_.append("{")

a\_flag = 1

if first\_ == 1:

print()

if(first\_ == 0):

first\_ = 1

string\_ = ' ' \* (count\_of\_space \* 4) + " "

print(string\_, end='')

for i in range(len(words\_)) :

print(words\_[i] + " ", end='')

for line in yaml\_file:

toJson(line)

while(flag != 0):

print("\n", (flag\*4)\*' ', "}", end='')

flag -=2

print("\n}")

*Пример ввода:*

timetable:

subject1:

day: Четверг

time: 10:00-11:30

name: История русской культуры(Практика)

teacher: Осипов Святослав Владимирович

week: четная

location: Кронверкский пр. д.49 лит.А

room: 2417 ауд.

format: Очный

subject2:

day: Четверг

time: 11:40-13:10

name: История русской культуры(Лекция)

teacher: Осипов Святослав Владимирович

week: четная

location: Кронверкский пр. д.49 лит.А

room: 2308 (бывш. 306) ауд.

format: Очно - дистанционный

*Пример вывода:*

{

"timetable" : {

"subject1" : {

"day" : "Четверг" ,

"time" : "1000-1130" ,

"name" : "История русской культуры(Практика)" ,

"teacher" : "Осипов Святослав Владимирович" ,

"week" : "четная" ,

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

"room" : "2417 ауд." ,

"format" : "Очный"

},

"subject2" : {

"day" : "Четверг" ,

"time" : "1140-1310" ,

"name" : "История русской культуры(Лекция)" ,

"teacher" : "Осипов Святослав Владимирович" ,

"week" : "четная" ,

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

"room" : "2308 (бывш. 306) ауд." ,

"format" : "Очно - дистанционный"

}

}

}

## ***Задание 3.1***

*Исходный код:*

import sys

import yaml

import json

# Открываем файл

with open('t3.txt', 'r', encoding='utf-8') as yaml\_file:

# Загружаем содержимое yaml файла

content = yaml.safe\_load(yaml\_file)

# Конвертируем данные в формат JSON

# ensure\_ascii - экранирует не ASCII-символы

json\_output = json.dumps(content, ensure\_ascii=False, indent=2)

# Печатаем результат

print(json\_output)

Вывод не изменился. Благодаря использованию специальной библиотеки, код стал лаконичным и коротким.

## ***Задание 3.2***

*Исходный код:*

import sys

import re

yaml\_file = open('t3.txt', 'r', encoding='utf-8')

print("{")

def word\_parser(s):

words = []

# Разбиваем строку на части: слова, двоеточия, и игнорируем пробелы и переносы строк

parts = re.split(r'(s+|:)', s)

word = ""

has\_two\_points = False

for part in parts:

part = part.strip()

if part == ':':

words.append("\"" + word + "\"")

word = ""

words.append(":")

has\_two\_points = True

elif part:

word += part

# Если встречен перенос строки, обрабатываем накопленное слово

if word != "":

words.append("\"" + word + "\"")

return words

#пробелы в оригинале

flag = 0

#наличие { в конце строки

a\_flag = 0

first\_ = 0

def toJson(str):

global flag

global a\_flag

global first\_

c = re.findall(r"\s\s+", str);

c.append('')

count\_of\_space = len(c[0])

str = str.lstrip()

# уменьшается количество пробелов = закрываются скобки

if(count\_of\_space < flag):

print("\n", (count\_of\_space\*4 + 4)\*' ', "},", end ='')

else:

if(a\_flag == 0 and first\_ == 1):

print(",", end='', sep='')

flag = count\_of\_space

json\_ = ""

words\_ = word\_parser(str)

a\_flag=0

#если после : нет слов - это начало нового блока

if(words\_[len(words\_)-1] == ":"):

words\_.append("{")

a\_flag = 1

if first\_ == 1:

print()

if(first\_ == 0):

first\_ = 1

string\_ = ' ' \* (count\_of\_space \* 4 + 4)

print(string\_, end='')

for i in range(len(words\_)) :

print(words\_[i] + " ", end='')

for line in yaml\_file:

toJson(line)

while(flag != 0):

print("\n", (flag\*4)\*' ', "}", end='')

flag -=2

print("\n}")

Ввод и вывод никак не изменились. Благодаря добавлению регулярных выражений, исчезли некоторые циклы и if-else: код уменьшился в объеме, стал понятнее и стал быстрее работать.

# **Вывод:**

В ходе выполнения данной лабораторной работы я изучила регулярные выражения, научилась работать с ними в Python, изучила основы JSON и YAML и научилась делать парсинг.

# **Список литературы:**

1. Регулярные выражения в Python от простого к сложному. Подробности, примеры, картинки, упражнения / [Электронный ресурс] // Habr : [сайт]. — URL: https://habr.com/ru/articles/349860/ (дата обращения: 20.10.2024).
2. Yaml vs. Json — что круче? / [Электронный ресурс] // Habr : [сайт]. — URL: https://habr.com/ru/companies/rambler\_and\_co/articles/525498/ (дата обращения: 20.10.2024).
3. Регулярные выражения (regexp) — основы / [Электронный ресурс] // Habr : [сайт]. — URL: https://habr.com/ru/articles/545150/ (дата обращения: 20.10.2024).