

Федеральное государственное автономное образовательное

учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет программной инженерии и компьютерной техники*

**Отчёт по лабораторной работе 3**

Тема: «Регулярные выражения»

Вариант 466662

Выполнил**:** студент группы Р3115 Матвеева П.П.

Проверил**:** Белокон Юлия Алексеевна

Дата сдачи: 21.10.2024

г. Санкт-Петербург 2024

**Вариант: ису 466662 (466662%6 = 0, 466662%4= 2, 466662%7= 0)**

**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. Все 5 тестов необходимо показать при защите.

3) Программа должна считать число 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.

Смайлик: :-{(

**Листинг №1:**

import re

def count\_smile(text):

return len(re.findall(r':-{\(', text))

#Тест 1

print(count\_smile('$%:-):>{)X-O:-{(XX;<=-{('))

#Ответ: 1

#Тест 2

print(count\_smile('Здравствуйте ;-) , введите смайл(Пример: :-{( ) :'))

#Ответ: 1

#Тест 3

print(count\_smile('\*:-);{(:-{{(X-/$%:-):>{)X-O:-)X<O'))

#Ответ: 0

#Тест 4

print(count\_smile(':-{(\*:-{(:-);{(:-{{(X-/$%:-):-{(:>{)X-O:-)X<O:-{('))

#Ответ: 4

#Тест 5

print(count\_smile(': -{(:- {(:-{ (:-{ ('))

#Ответ: 0

Вывод:

Тест 1: 1

Тест 2: 1

Тест 3: 0

Тест 4: 4

Тест 5: 0

**Задание №2(дополнительное):**

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. Все 5 тестов необходимо показать при защите. Пример тестов приведён в таблице.

3) Можно использовать циклы и условия, но основной частью решения должны быть регулярные выражения

Вариант:

Хайку – жанр традиционной японской лирической поэзии века, известный с XIV века. Оригинальное японское хайку состоит из 17 слогов, составляющих один столбец иероглифов. Особыми разделительными словами – кирэдзи – текст хайку делится на части из 5, 7 и снова 5 слогов. При переводе хайку на западные языки традиционно вместо разделительного слова использую разрыв строки и, таким образом, хайку записываются как трёхстишия. Перед вами трёхстишия, которые претендуют на то, чтобы быть хайку. В качестве разделителя строк используются символы «/». Если разделители делят текст на строки, в которых 5/7/5 слогов, то выведите «Хайку!». Если число строк не равно 3, то выведите строку «Не хайку. Должно быть 3 строки.». Иначе выведите строку вида «Не хайку.» Для простоты будем считать, что слогов ровно столько же, сколько гласных, не задумываясь о тонкостях.

**Листинг №2:**

import re

def Is\_haiku(text):

pattern1 = r'/'

pattern2 = r'[уеыаоэяиюёУЕЫАОЭЯИЮЁeyuioaEYUIOA]'

k = len(re.findall(pattern1, text))

if k!=2:

return "Не хайку. Должно быть три строки."

lines = text.split('/')

if len(re.findall(pattern2, lines[0]))==5 and len(re.findall(pattern2, lines[1]))==7\

and len(re.findall(pattern2, lines[2]))==5:

return "Хайку!"

return "Не хайку."

#Тест 1

#Вывод: Хайку!

print(Is\_haiku("На мертвой ветке/ Ворон сидит, одинок./Осенний вечер."))

#Тест 2

#Вывод: Не хайку. Должно быть три строки.

print(Is\_haiku("Я Вас любил...."))

#Тест 3

#Вывод: Не хайку.

print(Is\_haiku("В минуту жизни трудную/Теснится ль в сердце грусть,/Одну молитву чудную твержу я наизусть."))

#Тест 4

#Вывод: Не хайку.

print(Is\_haiku("Morning cloudiness/ cotton picking my boyhood/ ahead to the past"))

#Тест 5

#Вывод: Хайку!

print(Is\_haiku("slow moving river/ flash of iridescent black/ kingfisher strikes"))

Вывод:

Хайку!

Не хайку. Должно быть три строки.

Не хайку.

Не хайку.

Хайку!

**Задание №3(дополнительное):**

1)Реализуйте программный продукт на языке Python, используя регулярные выражения по варианту, представленному в таблице.

2) Для своей программы придумайте минимум 5 тестов. Все 5 тестов необходимо показать при защите.

3) Протестируйте свою программу на этих тестах.

4) Можно использовать циклы и условия, но основной частью решения должны быть регулярные выражения.

Вариант: Студент Вася очень любит курс «Компьютерная безопасность». Однажды Васе задали домашнее задание зашифровать данные, переданные в сообщении. Недолго думая, Вася решил заменить все целые числа на функцию от этого числа. Функцию он придумал не сложную 4𝑥 2 − 7, где 𝑥 − исходное число. Помогите Васе с его домашним заданием. Предусмотреть ситуацию, когда в тексте будут не только целые числа, но и любые символы.

**Листинг №3:**

import re

def encrypt\_numbers(text):

#Вычислим значения (a= 4\*x^2 - 7)

def encrypt(match):

x = int(match.group())

return str(4 \* (x \*\* 2) - 7)

#Замена всех целых чисел в тексте

encrypt\_text = re.sub(r'-?\d+', encrypt, text)

return encrypt\_text

#Тест 1

#Ввод: Корни данного уравнения: 28, 726 и -52.

#Вывод: Корни данного уравнения: 3129, 2108297 и 10809.

print(encrypt\_numbers("Корни данного уравнения: 28, 726 и -52."))

#Тест 2

#Ввод: У нас сегодня 3 опоздавших и 5 болеющих!

#Вывод: У нас сегодня 29 опоздавших и 93 болеющих!

print(encrypt\_numbers("У нас сегодня 3 опоздавших и 5 болеющих!"))

#Тест 3

#Ввод: История Санкт-Петербурга началась 27 (16 по старому стилю) мая 1703 года.

#Вывод: История Санкт-Петербурга началась 2909 (1017 по старому стилю) мая 11600829 года.

print(encrypt\_numbers("История Санкт-Петербурга началась 27 (16 по старому стилю) мая 1703 года"))

#Тест 4

#Ввод: Множество содержит числа: 1, 3, 5, 7, 9, 11, 13, 15.

#Вывод: Множество содержит числа: -3, 29, 93, 189, 317, 477, 669, 893.

print(encrypt\_numbers("Множество содержит числа: 1, 3, 5, 7, 9, 11, 13, 15."))

#Тест 5

#Ввод: Пётр I правил с 1682 по 1725 год.

#Вывод: Пётр I правил с 11316489 по 11902493 год.

print(encrypt\_numbers("Пётр I правил с 1682 по 1725 год."))

Вывод:

Корни данного уравнения: 3129, 2108297 и 10809.

У нас сегодня 29 опоздавших и 93 болеющих!

История Санкт-Петербурга началась 2909 (1017 по старому стилю) мая 11600829 года

Множество содержит числа: -3, 29, 93, 189, 317, 477, 669, 893.

Пётр I правил с 11316489 по 11902493 год.

**Вывод:**

В ходе выполнения лабораторной работы я узнала что такое регулярные выражения, научилась с ними работать, а также написала три программы на языке программирования python и тесты к ним.

**Список литературы:**

1. Регулярные выражения. 2021 г. Режим доступа:

<https://habr.com/ru/articles/545150/>