Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Информатика»

По лабораторной работе №3

Регулярные выражения

Вариант 202, 2, 1

Студент

*Александров А. А.*

*Р3106*

Преподаватель

*Балакшин П. В.*

*Кандидат технических наук*

*Доцент факультета ПИиКТ*

Санкт-Петербург 2023

Содержание

[Задание 4](#_Toc148737990)

[Основные этапы вычисления 5](#_Toc148737991)

[Var.py 6](#_Toc148737992)

[MainTask.py 6](#_Toc148737993)

[TestsMainTask.py 6](#_Toc148737994)

[FirstExtraTask.py 6](#_Toc148737995)

[TestsFirstExtraTask.py 7](#_Toc148737996)

[SecondExtraTask.py 7](#_Toc148737997)

[TestsSecondExtraTask.py 8](#_Toc148737998)

[Заключение 9](#_Toc148737999)

[Список используемой литературы 9](#_Toc148738000)

# Задание

***Основное задание:***

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

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.

Программа должна считать число смайликов определённого вида (вид смайлика описан в таблице вариантов) в предложенном тексте. Все смайлики имеют такую структуру: [*глаза*][*нос*][*рот*].

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описаниеВариантом является различные наборы глаз, носов и ртов.

***Первое доп. задание:***

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

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.

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

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

***Второе доп. задание:***

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

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

Протестируйте свою программу на этих тестах.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеМожно использовать циклы и условия, но основной частью решения должны быть регулярные выражения.

# Основные этапы вычисления

Исходя из ТЗ я выяснил, что нужно написать программы сортировок на языке Python используя Regax и написать для них тесты. Найдя всю необходимую информацию на просторах Google, я преступил к выполнению Лабораторной работы №3.

## Var.py

#основной таск вариант для глаз - 2  
print(408116 % 6)  
  
#основной таск вариант для носа - 0  
print(408116 % 4)  
  
#основной таск вариант для рта - 2  
print(408116 % 7)  
  
#первый дополнительный таск - 2  
print(408116 % 6)  
  
#второй дополнительный таск - 1  
print(408116 % 5)

## MainTask.py

import re  
  
def smile(string):  
 a = (re.findall('X-O', string))  
 return len(a)

## TestsMainTask.py

import unittest  
from MainTask import \*  
  
class MainTaskTest(unittest.TestCase):  
 def test\_only\_letters(self):  
 res = smile('AfnklAKsmdkksnfdnfBJNDS')  
 self.assertEqual(res, 0)  
 def test\_only\_numbers(self):  
 res = smile('17653926634939347858686')  
 self.assertEqual(res, 0)  
 def test\_only\_sings(self):  
 res = smile('!$@IU%@I$@O#PO%U@\_---$@')  
 self.assertEqual(res, 0)  
 def test\_empty\_string(self):  
 res = smile('')  
 self.assertEqual(res, 0)  
 def test\_random\_string(self):  
 res = smile('X-OsdlfjXPXD:OX-OX-oX-O')  
 self.assertEqual(res, 3)

## FirstExtraTask.py

import re  
def VTITMO (string):  
 a = re.findall(r'ВТ(?:\W+\w+|\W+){1,4}\s+ИТМО', string)  
 b = []  
 for i in a:  
 b.append(re.sub(r'[^\w\s]+', '', i))  
  
 return b

## TestsFirstExtraTask.py

import unittest  
from FirstExtraTask import \*  
  
class FirstExtraTask(unittest.TestCase):  
 def test\_VT\_Empty\_ITMO(self):  
 res = VTITMO('ВТ ИТМО мама овыарвжоарп валфоы')  
 self.assertEqual(res, [])  
 def test\_VT\_One\_Word\_ITMO(self):  
 res = VTITMO('ВТ хахахаха ИТМО мама овыарвжоарп валфоы')  
 self.assertEqual(res,['ВТ хахахаха ИТМО'] )  
 def test\_VT\_More\_Than\_Four\_ITMO(self):  
 res = VTITMO('ВТ z в в о в ИТМО мама овыарвжоарп валфоы')  
 self.assertEqual(res, [])  
 def test\_VT\_Other\_Sings\_ITMO(self):  
 res = VTITMO('ВТ 12ава 323 45324 (.;, ИТМО мама овыарвжоарп валфоы')  
 self.assertEqual(res, ['ВТ 12ава 323 45324 ИТМО'])  
 def test\_VT\_Random\_ITMO(self):  
 res = VTITMO('ВТ \_ 1 в фыатлыsdjfsdijosd1243\_\_ ИТМО мама овыарвжоарп валфоы ВТ ИТМО ВТ фыв ваы ыва ффффф ИМТО')  
 self.assertEqual(res, ['ВТ \_ 1 в фыатлыsdjfsdijosd1243\_\_ ИТМО'])  
 def test\_VT\_Special\_ITMO(self):  
 res = VTITMO('ВТ ИТМО ИТМО ИТМО ИТМО ИТМО ИТМО ИТМО ВТ ВТ ВТ ВТ ВТ ИТМО ИТМО ИТМО ИТМО ИТМО ИТМО ИТМО')  
 self.assertEqual(res, ['ВТ ИТМО ИТМО ИТМО ИТМО ИТМО', 'ВТ ВТ ВТ ВТ ВТ ИТМО'])

## SecondExtraTask.py

import re  
def Vowels (string):  
 string = string.lower()  
 a = re.findall(r'[а-яА-ЯёЁ]+', string)  
 c = []  
 helper = []  
 for i in a:  
 b = []  
 help\_iter = []  
 b += re.findall(r'[уеыаоэёяиюУЕЫАОЭЁЯИЮ]', i)  
 help\_iter += re.findall(r'\w+', i)  
 c.append(b)  
 helper.append(help\_iter)  
  
 final = []  
 counter = 0  
 for i in c:  
 i\_iter = i[0]  
 flag = True  
 for j in i:  
 if i\_iter == j:  
 flag = True  
 else:  
 flag = False  
 if flag:  
 final.append(helper[counter])  
 counter += 1  
 final.sort(key = lambda x: (len(x[0]), x[0][0]))  
  
 return final

## TestsSecondExtraTask.py

import unittest  
from SecondExtraTask import \*  
  
class SecondExtraTask(unittest.TestCase):  
 def test\_Vowels\_Basic(self):  
 res = Vowels('Классное слово – обороноспособность, которое должно идти после слов: трава и молоко.')  
 self.assertEqual(res, [['и'], ['идти'], ['слов'], ['слово'], ['трава'], ['должно'], ['молоко'], ['обороноспособность']])  
 def test\_Vowels\_Change\_N\_D(self):  
 res = Vowels('Классное слово – обороноспособность, которое нолжно идти после слов: трава и молоко.')  
 self.assertEqual(res, [['и'], ['идти'], ['слов'], ['слово'], ['трава'], ['молоко'], ['нолжно'], ['обороноспособность']])  
 def test\_Vowels\_Basic\_Second\_Var(self):  
 res = Vowels('Мама мыла раму, папа ел горох')  
 self.assertEqual(res, [['ел'], ['мама'], ['папа'], ['горох']])  
 def test\_Vowels\_Empty(self):  
 res = Vowels('Яма была глубокая. Таня, упалав туда, сломала ногу')  
 self.assertEqual(res, [])  
 def test\_Vowels\_Without\_Words(self):  
 res = Vowels('!!! %$# &&& <3, I love Informatics')  
 self.assertEqual(res, [])

# Заключение

Во время выполнения лабораторной работы узнал, что такое Regax, в общем и целом, а также научился применять регулярные выражения в языке программирования Python, также по ТЗ было необходимо написать тесты для программ сортировок, что было реализовано с помощью встроенного модуля тестирования Unittest.

# Список используемой литературы

Статья пользователя [**ShashkovS**](https://habr.com/ru/users/ShashkovS/) на тему "Регулярные выражения в Python от простого к сложному. Подробности, примеры, картинки, упражнения" была взята с сайта. URL https://habr.com/ru/articles/349860/ (дата обращения: 18.10.23)

Статья пользователя Андрей Смирнов на тему " Тестируем на Python: unittest и pytest. Инструкция для начинающих" была взята с сайта. URL https://tproger.ru/articles/testiruem-na-python-unittest-i-pytest-instrukcija-dlja-nachinajushhih (дата обращения: 18.10.23)

Видео пользователя Андрей Викторович на тему " Python с нуля. Урок 12 | Регулярные выражения. Часть 1 и 2" была взята с YouTube. URL https://www.youtube.com/watch?v=\_PSyCOuueFs, https://www.youtube.com/watch?v=kbeC4djs0mo&t=1s (дата обращения: 18.10.23)