Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет программной инженерии и компьютерной техники (ФПИ и КТ)

**Лабораторная работа №3**

## По информатике

Вариант 465716

Выполнил: Дарабзадех Захра Студент группы P3130 Преподаватель: Stepen



## Санкт-Петербург 2024

Содержание

[Основные этапы вычисления](#a1) 3

[Task 1](#a2) …………………………………………………………………………4

[Extra Task](#a4) \*…………………………………………………………………………………6

[Task 2](#a3) ………………………………………………………………………………………7

[Task 3](#a5) ………………………………………………………………………………………9

[Вывод](#a6) …………………………………………………………………………………..…10

# Основные этапы вычисления

## Задание на 60 баллов (Смайлики)

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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер в  ИСУ % 6 | Глаза | Номер в  ИСУ % 4 | Нос | Номер в  ИСУ % 8 | Рот |
| 0 | 8 | 0 | - | 0 | ( |
| 1 | ; | 1 | < | 1 | ) |
| 2 | X | 2 | -{ | 2 | P |
| 3 | : | 3 | <{ | 3 | | |
| 4 | = |  |  | 4 | \ |
| 5 | [ |  |  | 5 | / |
|  |  |  |  | 6 | O |
|  |  |  |  | 7 | = |

**# Номер ИСУ - 465716 --> Смайлик (X-\)**

Task 1 : Подсчет смайликов в тексте

import re

def solve(string):

    """

    Возвращает кол-во смайликов вида X-P в строке string

    409856 % 6 = 2 => Глаза: X

    409856 % 4 = 0 => Нос: -

    409856 % 7 = 6 => Рот: \

    :param string

    :return: int: number of emotions found in the text

    """

    pattern = r'X-\\'

    return len(re.findall(pattern, string))

# Пример использования

import unittest

import task1

class TestTask1(unittest.TestCase):

    def test\_no\_emotions(self):

        data = 'nobody hi'

        result = 0

        print(task1.solve(data))

        self.assertEqual(result, task1.solve(data))

    def test\_one\_emotions(self):

        data = 'X-\\'

        result = 1

        print(task1.solve(data))

        self.assertEqual(result, task1.solve(data))

    def test\_many\_emotions(self):

        data = 'X-\\X-\\X-\\X-\\X-\\X-\\'

        result = 6

        print(task1.solve(data))

        self.assertEqual(result, task1.solve(data))

    def test\_emotions\_with\_splits(self):

        data = '[]X-\\ hi XXX-\\\\\\     X-\\X-\\X-\\--\\'

        result = 5

        print(task1.solve(data))

        self.assertEqual(result, task1.solve(data))

    def test\_broken\_emotions(self):

        data = 'Hello X- \\ X-0 X -\\'

        result = 0

        print(task1.solve(data))

        self.assertEqual(result, task1.solve(data))

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

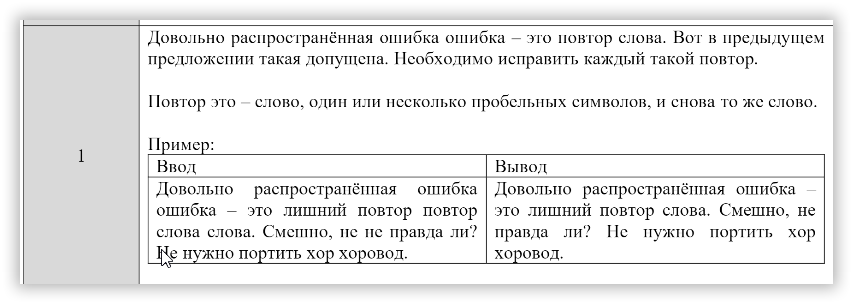
### Необязательное задания для получения оценки «4» или «5» (позволяет набрать

**+18 баллов от максимального числа баллов БаРС за данную лабораторную)**

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.

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



## Task 2 : Извлечение фраз с «ВТ» и «ИТМО»

import re

def solve(string):

    """

    1)

     Довольно распространённая ошибка ошибка – это повтор слова. Вот в предыдущем

    предложении такая допущена. Необходимо исправить каждый такой повтор.

    Повтор это – слово, один или несколько пробельных символов, и снова то же слово.

    """

    pattern = r'\b(\w+)\s+\1\b'

    return re.sub(pattern, '\\1', string)

# Пример использования

import unittest

import task2

class TestTask2(unittest.TestCase):

    def test\_no\_duplicates(self):

        data = 'nothing interesting'

        self.assertEqual(data, task2.solve(data))

    def test\_one\_duplicate(self):

        data = 'nothing nothing interesting'

        result = 'nothing interesting'

        self.assertEqual(result, task2.solve(data))

    def test\_one\_duplicate\_with\_comma(self):

        data = 'nothing, nothing interesting'

        self.assertEqual(data, task2.solve(data))

    def test\_many\_spaces(self):

        data = 'nothing          nothing interesting'

        result = 'nothing interesting'

        self.assertEqual(result, task2.solve(data))

    def test\_suffix\_fake\_duplicate(self):

        data = 'no nothing'

        self.assertEqual(data, task2.solve(data))

    def test\_example(self):

        data = 'Довольно распространённая ошибка ошибка – это лишний повтор повтор слова слова. Смешно, не не правда ли? Не нужно портить хор хоровод.'

        result = 'Довольно распространённая ошибка – это лишний повтор слова. Смешно, не правда ли? Не нужно портить хор хоровод.'

        self.assertEqual(result, task2.solve(data))

    def test\_numbers(self):

        data = '11 11 22 22'

        result = '11 22'

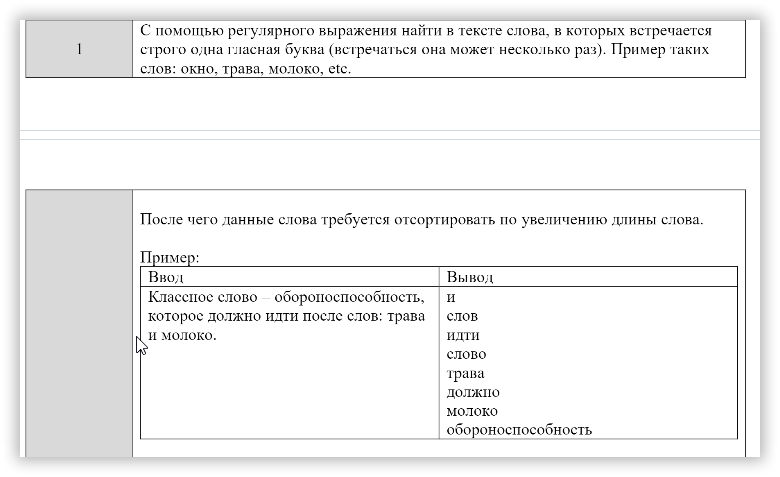
        self.assertEqual(result, task2.solve(data))

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

### Необязательное задания для получения оценки «4» или «5» (позволяет набрать +22 балла от максимального числа баллов БаРС за данную лабораторную)

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.



## Task 3

import re

def solve(string):

    """

    С помощью регулярного выражения найти в тексте слова, в которых встречается

    строго одна гласная буква (встречаться она может несколько раз). Пример таких

    слов: окно, трава, молоко, etc.

    После чего данные слова требуется отсортировать сначала по увеличению длины

    слова, а затем леWSYHJ64ксикографически.

    :param string:

    :return: list

    """

    words = re.findall(r'\b\w+\b', string)

    words\_with\_one\_vowel = []

    for word in words:

        for vowel in ['Аа', 'Ее', 'Ёё', 'Ии', 'Оо', 'Уу', 'Ыы', 'Ээ', 'Юю', 'Яя']:

            x = re.sub(f'[{vowel}]', '', 'АЕЁИОУЫЭЮЯаеёиоуыэюя')

            if re.search(rf"^[^{x}]\*[{vowel}]+[^{x}]\*$", word):

                words\_with\_one\_vowel.append(word)

    return sorted(set(words\_with\_one\_vowel), key=lambda x: (len(x), x))

# Пример использования

import unittest

import task3

class TestTask3(unittest.TestCase):

    def test\_no\_vowels(self):

        data = 'пппп ннн цц прпрпрпр звпркнмь'

        result = []

        self.assertEqual(result, task3.solve(data))

    def test\_one\_vowels(self):

        data = 'пап пуп уп у пп'

        result = ['у', 'уп', 'пап', 'пуп']

        self.assertEqual(result, task3.solve(data))

    def test\_example(self):

        data = 'Классное слово – обороноспособность, которое должно идти после слов: трава и молоко.'

        result = ['и', 'идти', 'слов', 'слово', 'трава', 'должно', 'молоко', 'обороноспособность']

        self.assertEqual(result, task3.solve(data))

    def test\_duplicates(self):

        data = 'пап пап пап пап пап'

        result = ['пап']

        self.assertEqual(result, task3.solve(data))

    def test\_only\_vowels(self):

        data = 'а у о уу оу уоу'

        result = ['а', 'о', 'у', 'уу']

        self.assertEqual(result, task3.solve(data))

    def test\_some\_words(self):

        data = 'окно трава молоко'

        result = ['окно', 'трава', 'молоко']

        self.assertEqual(result, task3.solve(data))

    def test\_sentence(self):

        data = 'съешь? этих!!! мягких... французских<> булок</title>, да. выпей чаю'

        result = ['да', 'съешь']

        self.assertEqual(result, task3.solve(data))

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

# Вывод

Выполняя работу, я использовалa Sublime Text Notebook для работы с языком программирования Python, а также изучилa некоторые конструкции регулярных выражений.