
Министерство образования и науки Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

“ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет программной инженерии и компьютерной техники (ПИКТ)

Лабораторная работа № 3

Выполнил: Юксель Хамза

Группа P3132

Выполнил: Белокон Юлия Алексеевна

г. Санкт-Петербург

2024 г.

Оглавление

İçindekiler

Вариант:

Таблица 1. Варианты

Задание	Вариант
1	**8-{0**
2	**0**
3	**6**

Задания:

Задание 1. (60 баллов)

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 таблицевариантов) 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	=

Задание 2. (задания для получения оценки «4» или «5» позволяет набрать +18 баллов)

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.

<table>
<tr><th>Номер в ИСУ % 6</th><th>Задание</th></tr>
<tr><td>0</td><td>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.

Пример:
<table>
<tr><th>Ввод</th><th>Вывод</th></tr>
<tr><td>Вечер за окном. / Еще один день прожит. / Жизнь скоротечна...</td><td>Хайку!</td></tr>
<tr><td>Просто текст</td><td>Не хайку. Должно быть 3 строки.</td></tr>
<tr><td>Как вишня расцвела! / Она с коня согнала / И князя-гордеца.</td><td>Не хайку.</td></tr>
</table>
</td></tr>
</table>

Задание 3. (задания для получения оценки «4» или «5» позволяет набрать +22 балла)

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.

6	Вам необходимо поменять падежи в тексте у прилагательных, которые встречаются несколько раз. На вход подаётся текст и порядковый номер слова, падежная форма которого будет использована для замены. Пример ввода: 1)Номер 2 2)Текст: Футбольный клуб «Реал Мадрид» является 15-кратным обладателем главного футбольного европейского трофея – Лиги Чемпионов. Данный турнир организован

	Союзом европейских футбольных ассоциаций (УЕФА). Идея о континентальном футбольном турнире пришла к журналисту Габриэлю Ано в 1955 году. Пример вывода: Футбольного клуб «Реал Мадрид» является 15-кратным обладателем главного футбольного европейского трофея – Лиги Чемпионов. Данный турнир организован Союзом европейских футбольного ассоциаций (УЕФА). Идея о континентальном футбольного турнире пришла к журналисту Габриэлю Ано в 1955 году.

Отчет:

- **Задание 1**

В этой задаче требуется подсчитать, сколько раз шаблон (эмодзи) встречается во всём тексте. Для этого используется функция findall из библиотеки re, которая принимает в качестве аргументов регулярное выражение для поиска и строку текста, в которой необходимо выполнить этот поиск.

Функция вернёт список со всеми найденными совпадениями в тексте. Если совпадений нет, она вернёт пустой список.

Наконец, используется функция len() для подсчёта количества элементов в списке, и результат выводится на экран.

Тест эмодзи с помощью библиотеки Regex:


```python
import re

def search_emotions(string):
    '''
    my isu number 408078
    408078 % 6 = 0 ==> 8
    408078 % 4 = 2 ==> -{
    408078 % 8 = 6 ==> O
    pattern = 8-{O
    '''
    pattern = r"8-\{O"
    return len(re.findall(pattern, string))
```

Сравнение ручной проверки (без использования Regex) и проверки с помощью Regex с использованием модуля unittest:

```python
import task1_re
import unittest
# pattern = 8-{O


class TestTask1(unittest.TestCase):
    def test_no_emotions(self):
        text = "Hello my name is Hamza:)"
        result = 0
        return (self.assertEqual(result, task1_re.search_emotions(text)))

    def test_emotion_with_blank(self):
        text = "Today, wather 8-{ Ois so good"
        result = 0
        return (self.assertEqual(result, task1_re.search_emotions(text)))

    def test_multiple_emotions(self):
        text = "Studying 8-{Oat ITMO is not -{O8-{O8-{O8-{O stressful(!)"
        result = 4
        return (self.assertEqual(result, task1_re.search_emotions(text)))

    def test_one_emotion(self):
        text = "8-{O"
        result = 1
        return (self.assertEqual(result, task1_re.search_emotions(text)))

    def test_splited_emotions(self):
        text = "8-8-{O8-8-{O{O{8-{O8-{O8-{OO8-{O"
        result = 6
```


```
        return (self.assertEqual(result, task1_re.search_emotions(text)))


if __name__ == '__main__':
    unittest.main()
```

Output:

```
PS C:\Users\hamza\Desktop\LABLAR\info_lab3> & C:/Users/hamza/AppData/Local/Programs/Python/Python312/pyt
hon.exe c:/Users/hamza/Desktop/LABLAR/info_lab3/task1_test.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.000s

OK
PS C:\Users\hamza\Desktop\LABLAR\info_lab3>
```

- **Задание 2**

Разделение строк во введенной строке знаком «/». После определения строк проверяется с помощью regex-шаблона, соответствуют ли полученные на вход стихи правилу 5-7-5 слогов, то есть являются ли они стихами хайки (Хайку).

```
import re

def find_pattern(string):
    # 408078 % 6 = 0 ==> Variant 0

    lines = string.split("/")

    if len(lines) != 3:
        return "The number of lines should be 3"

    syllable_patterns = [r"^([^aeiou]*[aeiou]){5}[^aeiou]*$",
r"^([^aeiou]*[aeiou]){7}[^aeiou]*$", r"^([^aeiou]*[aeiou]){5}[^aeiou]*$"]

    for line, pattern in zip(lines, syllable_patterns):
        if not re.match(pattern, line, re.IGNORECASE):
            return "Not Haiku!"
    return "Haiku!"
```


Сравнение ручной проверки (без использования Regex) и проверки с помощью Regex с использованием модуля unittest:

```
import task2_re
import unittest


class TaskTest2(unittest.TestCase):
    def test_less_syllables(self):
        text = "Evening at the window./Another day lived./ Without you"
        result = "Not Haiku!"
        return (self.assertEqual(result, task2_re.find_pattern(text)))

    def test_less_lines(self):
        text = "This is/not a haiku poem."
        result = "The number of lines should be 3"
        return (self.assertEqual(result, task2_re.find_pattern(text)))

    def test_traditional_haiku(self):
        text = "An old silent pond/A frog jumps into the pond/Splash! Silence
again."
        result = "Not Haiku!"
        return (self.assertEqual(result, task2_re.find_pattern(text)))

    def test_modern_haiku(self):
        text = "In the twilight rain/These brilliant-hued hibiscus/A lovely
sunset."
        result = "Not Haiku!"
        return (self.assertEqual(result, task2_re.find_pattern(text)))

    def test_personal_reflection(self):
        text = "First autumn morning/The mirror I stare into/Shows my father's
face"
        result = "Haiku"
        return (self.assertEqual(result, task2_re.find_pattern(text)))


if __name__ == "__main__":
    unittest.main()
```

Output:


```
PS C:\Users\hamza\Desktop\LABLAR\info_lab3> & C:/Users/hamza/AppData/Local/Programs/Python/Python312/pyt
hon.exe c:/Users/hamza/Desktop/LABLAR/info_lab3/task2_test.py
...F.
======================================================================
FAIL: test_personal_reflection (__main__.TaskTest2.test_personal_reflection)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\hamza\Desktop\LABLAR\info_lab3\task2_test.py", line 29, in test_personal_reflection
    return (self.assertEqual(result, task2_re.find_pattern(text)))
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: 'Haiku' != 'Not Haiku!'
- Haiku
+ Not Haiku!


----------------------------------------------------------------------
Ran 5 tests in 0.001s

FAILED (failures=1)
PS C:\Users\hamza\Desktop\LABLAR\info_lab3>
```

Стихотворение, которое я называю хайку с ручной проверкой, не является хайку с проверкой regex.

- **Задание 3**

Я изменил падежи прилагательных в заданном тексте на заданный номер (новый падеж). Для этого я нашел суффиксы прилагательных с помощью regex. Затем, используя re.sub, я изменил суффиксы.

```python
import re

# 408078 % 8 = 6 ==> Variant 6

def replace_suffix(text, case_number):
    pattern = r"\b(\w*?)(ый|ий|ая|ое|ую|ем|их|ого|ому|ым|ыми|ом|ой|ые|ых)\b"

    replacements = {
        1: 'ый', # for masculine case 1
        2: 'ый', # for masculine case 2
        3: 'ого',# for masculine case 3
        4: 'ому',# for masculine case 4
        5: 'ым', # for masculine case 5
        6: 'ом'  # for masculine case 6
    }
    if case_number not in replacements:
        raise ValueError("Invalid case number. Choose a number between 1 and
6.")
```



```
    new_suffix = replacements[case_number]
    return re.sub(pattern, rf'\1{new_suffix}', text, flags=re.UNICODE)
```

Сравнение ручной проверки (без использования Regex) и проверки с помощью Regex с использованием модуля unittest:

```
import task3_re
import unittest


class TaskTest3(unittest.TestCase):
    def test_case1(self):
        text = "В солнечный день я гулял по парку и наслаждался природой.
Вокруг были цветы, птицы пели, и всё казалось идеальным."
        result = "В солнечный день я гулял по парку и наслаждался природый.
Вокруг были цветы, птицы пели, и всё казалось идеальный."
        return (self.assertEqual(result, task3_re.replace_suffix(text,1)))

    def test_case2(self):
        text = "В солнечный день я гулял по парку и наслаждался природой.
Вокруг были цветы, птицы пели, и всё казалось идеальным."
        result = "В солнечный день я гулял по парку и наслаждался природый.
Вокруг были цветы, птицы пели, и всё казалось идеальный."
        return (self.assertEqual(result, task3_re.replace_suffix(text,2)))

    def test_case3(self):
        text = "В солнечный день я гулял по парку и наслаждался природой.
Вокруг были цветы, птицы пели, и всё казалось идеальным."
        result = "В солнечного день я гулял по парку и наслаждался природого.
Вокруг были цветы, птицы пели, и всё казалось идеального."
        return (self.assertEqual(result, task3_re.replace_suffix(text,3)))

    def test_case4(self):
        text = "В солнечный день я гулял по парку и наслаждался природой.
Вокруг были цветы, птицы пели, и всё казалось идеальным."
        result = "В солнечному день я гулял по парку и наслаждался природому.
Вокруг были цветы, птицы пели, и всё казалось идеальному."
        return (self.assertEqual(result, task3_re.replace_suffix(text,4)))

    def test_case5(self):
        text = "В солнечный день я гулял по парку и наслаждался природой.
Вокруг были цветы, птицы пели, и всё казалось идеальным."
```


```
        result = "В солнечным день я гулял по парку и наслаждался природым.
Вокруг были цветы, птицы пели, и всё казалось идеальным."
        return (self.assertEqual(result, task3_re.replace_suffix(text,5)))

    def test_case6(self):
        text = "В солнечный день я гулял по парку и наслаждался природой.
Вокруг были цветы, птицы пели, и всё казалось идеальным."
        result = "В солнечном день я гулял по парку и наслаждался природом.
Вокруг были цветы, птицы пели, и всё казалось идеальном."
        return (self.assertEqual(result, task3_re.replace_suffix(text,6)))


if __name__ == "__main__":
    unittest.main()
```

Output:

```
PS C:\Users\hamza\Desktop\LABLAR\info_lab3> & C:/Users/hamza/AppData/Local/Programs/Python/Python312/pyt
hon.exe c:/Users/hamza/Desktop/LABLAR/info_lab3/task3_test.py
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK
PS C:\Users\hamza\Desktop\LABLAR\info_lab3>
```

- **Main.py**

```
import task1_re, task2_re, task3_re

task = input("Enter task number (1-3) you want to run: ")

if task == '1':
    print(task1_re.search_emotions(input("Enter your string to find the
number of emotions: ")))

elif task == '2':
    print(task2_re.find_pattern(input("Enter a poem whose lines are
seperated with '/': ")))

elif task == '3':
    print(task3_re.replace_suffix(text=input("Enter your text to replace
adjective cases with desired case number(text): "),case_number =
int(input("Case number(1-6): "))))

else:
    print("Can't resolve task number")
```

```
PS C:\Users\hamza\Desktop\LABLAR\info_lab3> & C:/Users/hamza/AppData/Local/Programs/Python/Pyt
hon312/python.exe c:/Users/hamza/Desktop/LABLAR/info_lab3/main.py
Enter task number (1-3) you want to run: 1
Enter your string to find the number of emotions: Studying 8-{Oat ITMO is not -{O8-{O8-{O8-{O
stressful(!)
4
```

вывод для task1, поиск эмоций.

```
PS C:\Users\hamza\Desktop\LABLAR\info_lab3> & C:/Users/hamza/AppData/Local/Programs/Python/Pyt
hon312/python.exe c:/Users/hamza/Desktop/LABLAR/info_lab3/main.py
Enter task number (1-3) you want to run: 2
Enter a poem whose lines are seperated with '/': First autumn morning/The mirror I stare into/
Shows my father's face.
Not Haiku!
```

вывод для task2, поиск стихов хайку.

```
PS C:\Users\hamza\Desktop\LABLAR\info_lab3> & C:/Users/hamza/AppData/Local/Programs/Python/Pyt
hon312/python.exe c:/Users/hamza/Desktop/LABLAR/info_lab3/main.py
Enter task number (1-3) you want to run: 3
Enter your text to replace adjective cases with desired case number(text): В солнечный день я
гулял по парку и наслаждался природой. Вокруг были цветы, птицы пели, и всё казалось идеальным
.
Case number(1-6): 6
В солнечном день я гулял по парку и наслаждался природом. Вокруг были цветы, птицы пели, и всё
 казалось идеальном.
PS C:\Users\hamza\Desktop\LABLAR\info_lab3>
```

вывод для taks3, изменяющего падежи прилагательных.

Вывод:

Использование регулярных выражений предоставляет мощный инструмент для работы с текстовыми данными в различных приложениях. Они позволяют эффективно выполнять сложные задачи, такие как анализ, преобразование и конвертация форматов. Эти техники помогают упростить обработку информации, делая ее более структурированной и доступной. Автоматизируя многие рутинные задачи, регулярные выражения экономят время

Список литературы:

- Habr. (2019). *Регулярные выражения: знакомство с основами и применение в повседневной практике*. [Электронный ресурс]. URL: https://habr.com/ru/articles/442964/
- GeeksforGeeks, Regex tutorial - *How to Write Regular Expressions?* [Электронный ресурс]. URL: https://www.geeksforgeeks.org/write-regular-expressions/
- Russian Adjectives [Электронный ресурс]. URL: https://www.russianlessons.net/grammar/adjectives.php

