**Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики.**

Факультет Программной Инженерии и Компьютерной Техники



Лабораторная работа №4

предмет «Информатика»

ФИО: Фам Мань Туан

Группа: P3112

Преподаватель: Павел Балакшин

Малышева Татьяна Алексеевна

**Санкт-Петербург 2020 г.**

**Цель задания:**

Ознакомьтесь с обработкой текстовых данных с помощью регулярных выражений.

**Задание:**

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.

Пример тестов приведён в таблице.

**Номер в ИСУ: 308549**

|  |  |
| --- | --- |
| **№ варианта** | **Задание** |
| 4 | Анатолий выложил пост с расписанием доп. Занятий по информатике, но везде перепутал время. Поэтому нужно заменить все вхождения времени на строку (TBD). Время – это строка вида HH:MM:SS или HH:MM, в которой HH – число от 00 до 23, а MM и SS – число от 00 до 59.  Пример:   |  |  | | --- | --- | | Ввод | Вывод | | Уважаемые студенты! В эту субботу в 15:00 планируется доп. занятие на 2 часа. То есть в 17:00:01 оно уже точно кончится. | Уважаемые студенты! В эту субботу в (TBD) планируется доп. занятие на 2 часа. То есть в (TBD) оно уже точно кончится. | |

**Код программы:**

*- Без использования регулярных выражений*

#Not using regular expressions

def counting(input, n):

if input[n].isnumeric() and input[n+1].isnumeric() and input[n+2] == ':' and input[n+3].isnumeric() and input[n+4].isnumeric():

if int(input[n:n+2]) >= 0 and int(input[n:n+2]) <= 23 and int(input[n+3:n+5]) >=0 and int(input[n+3:n+5]) <= 59:

if n == 0:

if n+5 == length: return 5

elif input[n+5].isspace() or input[n+5] == '.' or input[n+5] == '?' or input[n+5] == '!' or input[n+5] == ',' or input[n+5] == ';' or input[n+5] == '\'' or input[n+5] == '\"': return 5

elif input[n+5] == ":" and input[n+6].isnumeric() and input[n+7].isnumeric():

if int(input[n+6:n+8]) >=0 and int(input[n+6:n+8]) <= 59:

if n+8 == length: return 8

elif input[n+8].isspace() or input[n+8] == '.' or input[n+8] == '?' or input[n+8] == '!' or input[n+8] == ',' or input[n+8] == ';' or input[n+8] == '\'' or input[n+8] == '\"': return 8

else: return 0

else: return 0

else: return 0

elif input[n-1].isspace():

if n + 5 == length: return 5

elif input[n+5].isspace() or input[n+5] == '.' or input[n+5] == '?' or input[n+5] == '!' or input[n+5] == ',' or input[n+5] == ';' or input[n+5] == '\'' or input[n+5] == '\"': return 5

elif input[n + 5] == ":" and input[n + 6].isnumeric() and input[n + 7].isnumeric():

if int(input[n + 6:n + 8]) >= 0 and int(input[n + 6:n + 8]) <= 59:

if n + 8 == length: return 8

elif input[n + 8].isspace() or input[n+8] == '.' or input[n+8] == '?' or input[n+8] == '!' or input[n+8] == ',' or input[n+8] == ';' or input[n+8] == '\'' or input[n+8] == '\"': return 8

else: return 0

else: return 0

else: return 0

else: return 0

else: return 0

else: return 0

def replace(string, material, positions):

result = ""

for n in range(0,len(positions)):

if n > 0:

result += string[positions[n - 1][0] + positions[n-1][1]:positions[n][0]] + material

else:

result = string[0:positions[0][0]] + material

result += string[positions[n][0] + positions[n][1]:]

return result

input = open(r"C:\Users\the7u\PycharmProjects\lab4\_info\text", "r+", encoding="utf-8").read()

positions\_not\_ariana\_grandes\_reference = []

n = 0

result = ""

length = len(input)

for n in range(0,length-4):

count = counting(input, n)

if count != 0:

#print(input[n:n+count])

positions\_not\_ariana\_grandes\_reference.append([n,count])

print(replace(input, "(TBH)", positions\_not\_ariana\_grandes\_reference))

- *Используя регулярные выражения*

import re

test\_str = open(r"C:\Users\the7u\PycharmProjects\lab4\_info\text", "r+", encoding="utf-8").read()

replaced\_string = re.sub(r"\b([01][0-9]|2[0-3]):([0-6][0-9])(:([0-6][0-9])\b|\b)", "(TBH)" , test\_str)

print(replaced\_string)

**Результат**

|  |  |  |
| --- | --- | --- |
| **Тестов** | **Используя регулярные выражения** | **Без использования регулярных выражений** |
| Я обедаю в 12:30 и возвращаюсь в школу в 13:00. Я пошел домой после 18:15. | Я обедаю в (TBH) и возвращаюсь в школу в (TBH). Я пошел домой после (TBH). | Я обедаю в (TBH) и возвращаюсь в школу в (TBH). Я пошел домой после (TBH). |
| Уважаемые студенты! В эту субботу в 15:00 планируется доп. занятие на 2 часа. То есть в 17:00:01 оно уже точно кончится. | Уважаемые студенты! В эту субботу в (TBH) планируется доп. занятие на 2 часа. То есть в (TBH) оно уже точно кончится. | Уважаемые студенты! В эту субботу в (TBH) планируется доп. занятие на 2 часа. То есть в (TBH) оно уже точно кончится. |
| Один из способов решить эту проблему - пожертвовать точностью ради ясности. Ваш друг может попросить вас прибыть в аэропорт в 12:01 13 апреля или, если имеется в виду следующая полночь, в 23:59 13 апреля. В качестве альтернативы можно использовать 24-часовой формат. Здесь 00:00 означает полночь в начале дня, а 24:00 - это полночь в конце дня. | Один из способов решить эту проблему - пожертвовать точностью ради ясности. Ваш друг может попросить вас прибыть в аэропорт в (TBH) 13 апреля или, если имеется в виду следующая полночь, в (TBH) 13 апреля. В качестве альтернативы можно использовать 24-часовой формат. Здесь (TBH) означает полночь в начале дня, а 24:00 - это полночь в конце дня. | Один из способов решить эту проблему - пожертвовать точностью ради ясности. Ваш друг может попросить вас прибыть в аэропорт в (TBH) 13 апреля или, если имеется в виду следующая полночь, в (TBH) 13 апреля. В качестве альтернативы можно использовать 24-часовой формат. Здесь (TBH) означает полночь в начале дня, а 24:00 - это полночь в конце дня. |
| Парковка открывается в 08:00:00, у ворот только один дежурный, время обработки - 15 секунд на машину. В 08:00:00 одного дня уже 6 машин ждут у входа прямо перед началом работы дежурного). С 08:00:00 скорость прибытия автотранспорта - 6 машин в минуту. Однако после 08:30:00 скорость прибытия снижается до 3 автомобилей в минуту. | Парковка открывается в (TBH), у ворот только один дежурный, время обработки - 15 секунд на машину. В (TBH) одного дня уже 6 машин ждут у входа прямо перед началом работы дежурного). С (TBH) скорость прибытия автотранспорта - 6 машин в минуту. Однако после (TBH) скорость прибытия снижается до 3 автомобилей в минуту. | Парковка открывается в (TBH), у ворот только один дежурный, время обработки - 15 секунд на машину. В (TBH) одного дня уже 6 машин ждут у входа прямо перед началом работы дежурного). С (TBH) скорость прибытия автотранспорта - 6 машин в минуту. Однако после (TBH) скорость прибытия снижается до 3 автомобилей в минуту. |
| Тип данных для времени прибытия - varchar (10). Проблема, с которой я сталкиваюсь, заключается в том, что когда текущее время составляет около 23:30, а добавленная стоимость для текущего времени составляет 60 минут, время становится 00:30. Поэтому мне нужно получить записи, которые находятся между 23:30 и 00:30. Используя запрос, который я поставил, извлекает записи до 23:59:59. | Тип данных для времени прибытия - varchar (10). Проблема, с которой я сталкиваюсь, заключается в том, что когда текущее время составляет около (TBH), а добавленная стоимость для текущего времени составляет 60 минут, время становится (TBH). Поэтому мне нужно получить записи, которые находятся между (TBH) и (TBH).  используя запрос, который я поставил, извлекает записи до (TBH). | Тип данных для времени прибытия - varchar (10). Проблема, с которой я сталкиваюсь, заключается в том, что когда текущее время составляет около (TBH), а добавленная стоимость для текущего времени составляет 60 минут, время становится (TBH). Поэтому мне нужно получить записи, которые находятся между (TBH) и (TBH).  используя запрос, который я поставил, извлекает записи до (TBH). |

**Доп. задание №1**

1) Для выполнения задания скачайте текстовый документ по ссылке, соответствующей вашему варианту.

|  |  |  |
| --- | --- | --- |
| **№ варианта** | **Задание** | **Описание** |
| 29 | *https://drive.google.com/file/d/1YoQjz0dXSz9XFrPIIBHM\_CHmJKHpYlNo/view* | Ромео и  Джульетта |

2) Реализуйте программный продукт на языке Python, используя регулярные выражения, выделите все предложения из вашего текста, соответствующие критерию по вашему варианту.

3) Придумайте способ, как проверить корректность работы вашего регулярного выражения и проверьте его

|  |  |
| --- | --- |
| **№ варианта** | **Задание** |
| 1 | Все восклицательные предложения, содержащие хотя бы одну запятую |

**Код программы:**

import re

input = open(r"C:\Users\the7u\PycharmProjects\lab4\_info\RomeoAndJuliet", "r+", encoding="utf-8").read()

input = re.sub(r"\n\s+", " ", input)

string\_regex = re.compile(r"(?:[\s][A-Z]+[A-Za-z0-9;'\"\s]+,[A-Za-z0-9;'\"\s]+[.?!])|(?:^[A-Z]+[A-Za-z0-9;'\"\s]+,[A-Za-z0-9;'\"\s]+[.?!])")

print(re.findall(string\_regex,input))

**Доп. 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.

|  |  |
| --- | --- |
| **№ варианта** | **Задание** |
| 5 | Вывесили списки стипендиатов текущего семестра, которые представляют из себя список людей ФИО и номер группы этого человека. Вы решили подшутить над некоторыми из своих одногруппников и удалить их из списка. С помощью регулярного выражения найдите всех студентов своей группы, у которых  инициалы начинаются на одну и туже букву и исключите их из списка.  Пример (группа P000):   |  |  | | --- | --- | | Ввод | Вывод | | Петров П.П. P000  Анищенко А.А. P33113  Примеров Е.В. P000  Иванов И.И. P000 | Анищенко А.А. P33113  Примеров Е.В. P000 | |

**Код программы:**

import re

def removeblankline(str):

result = ""

lines = str.split('\n')

for line in lines:

if line.strip():

result += line + "\n"

return result

with open(r"C:\Users\the7u\PycharmProjects\lab4\_info\text\_zhadaniye2", "r", encoding="utf-8") as f:

input = f.read()

regex = re.sub(r"^(.)+\s(.)[.]\2{1}[.]\sP000$", "", input, flags=re.MULTILINE)

f.close()

with open(r"C:\Users\the7u\PycharmProjects\lab4\_info\text\_zhadaniye2", "w", encoding="utf-8") as f:

f.write(removeblankline(regex))

f.close()

**Результат**

*- Ввод*

Петров П.П. P000

Анищенко А.А. P31112

Примеров Е.В. P000

Иваов G.G. P000

Ивав И.И. P6522

Пеов E.E. P000

*- Вывод*

Анищенко А.А. P31112

Примеров Е.В. P000

Ивав И.И. P6522

**Вывод**

В ходе лабораторной работы я познакомился с основными принципами обработки текстовых данных с использованием регулярных выражений.