Санкт-Петербургский Национальный Исследовательский Университет
ИнформационныхТехнологий,Механики и ОптикиФакультетПрограммной Инженерии и
КомпьютернойТехники

Вариант № 1

Лабораторная работа № 4

по дисциплине

’Информатика’

Выполнил:
Веласкес Хуан
291005;
P3113;
Преподаватель:
Балакшин Павел Валерьевич

Санкт-Петербург 2020 г.

Текст задания:

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.

Вариант 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.

Пример:

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

Вывод:

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

Программа :

```
import re
import string

Test1= "Довольно распространённая ошибка ошибка - это лишний повтор повтор слова слова.
Смешно, не не правда ли? Не нужно портить хор хоровод."
Test2="Объктно-ориентиированное программиирование     программиирование это методология
программирования, основанная основанная на представлении программы в виде совокупности
совокупности объектов."
Test3="Регулярные выражения выражения это формальный язык поиска и осуществления
осуществления манипуляций с подстроками в тексте, основанный основанный на использовании
метасимволов."
Test4="Python это высокоуровневый язык язык программирования общего назначения,
ориентированный на повышение повышение производительности разработчика и читаемости
читаемости кода."
Test5="Информатика это наука наука о методах и процессах сбора, хранения, обработки обработки,
передачи, анализа и оценки информации информации с применением компьютерных технологий,
обеспечивающих обеспечивающих возможность её использования для принятия решений."


def replaceReg(string):
    re_output=re.sub(r'\b(\w+)\s*( \1\b)+', r'\1', string)
    return re_output
```

```
def withoutPunct(stringWithPunct):
    for c in string.punctuation:
        stringWithPunct= stringWithPunct.replace(c,"")
    return stringWithPunct

def replaceNormal(string):
    count=0
    new=[]
    split=string.split()
    actual=""
    next=""
    nextWithoutPunt=""
    temp=0
    for word in split:
        if(count+1<len(split)):
            actual=word
            nextWord=split[count+1]
            temp=len(nextWord)
        else:
            actual=word
            nextWord=""

        if(nextWord[temp - 1:]=='.' or nextWord[temp - 1:]==','):
            nextWithoutPunt=nextWord[:temp - 1]
        if(actual==nextWord):
            new.append("")
        elif(actual==nextWithoutPunt):
            new.append('')
        else:
            new.append(actual)

        count+=1
    return ' '.join(new)

re_output1 = replaceReg(Test1)
re_output2 = replaceReg(Test2)
re_output3 = replaceReg(Test3)
re_output4 = replaceReg(Test4)
re_output5 = replaceReg(Test5)

output1=replaceNormal(Test1)
output2=replaceNormal(Test2)
output3=replaceNormal(Test3)
output4=replaceNormal(Test4)
output5=replaceNormal(Test5)


print("REgex1:"+ re_output1)
print("Normal1: "+ output1+'\n')

print("REgex2:"+ re_output2)
print("Normal2: "+ output2+'\n')

print("REgex3:"+ re_output3)
print("Normal3: "+ output3+'\n')

print("REgex4:"+ re_output4)
print("Normal4: "+ output4+'\n')

print("REgex5:"+ re_output5)
```

```
print("Normal5: "+ output5+'\n')
```

Результат:

REgex1:Довольно распространённая ошибка - это лишний повтор слова. Смешно, не правда ли? Не нужно портить хор хоровод.
Normal1: Довольно распространённая ошибка - это лишний повтор слова. Смешно, не правда ли? Не нужно портить хор хоровод.

REgex2:Объктно-ориентиированное программиирование это методология программирования, основанная на представлении программы в виде совокупности объектов.
Normal2: Объктно-ориентиированное программиирование это методология программирования, основанная на представлении программы в виде совокупности объектов.

REgex3:Регулярные выражения это формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов.
Normal3: Регулярные выражения это формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов.

REgex4:Python это высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода.
Normal4: Python это высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода.

REgex5:Информатика это наука о методах и процессах сбора, хранения, обработки, передачи, анализа и оценки информации с применением компьютерных технологий, обеспечивающих возможность её использования для принятия решений.
Normal5: Информатика это наука о методах и процессах сбора, хранения, обработки, передачи, анализа и оценки информации с применением компьютерных технологий, обеспечивающих возможность её использования для принятия решений.

Дополнительное задание 1 - вариант 2 / Все предложения, в которых используется две и более запятых

Текст : Гамлет - https://drive.google.com/file/d/1j2p-EAwainLQ7dbOf2CFAxDTjQ9O7tNl/view

Программа :

```
import re

with open('Hamlet.txt', 'r') as rf:
        with open('outpoud.txt', 'w') as wf:
                rf_content=rf.readlines()

                regex = '^[^,\n]*((,[^,\n]*){2,}$)'

                for line in rf_content:
                        m=re.search(regex,line)

                        if(m):
                                print(line)
                                wf.write(line)
```

Результаты:

https://drive.google.com/file/d/1Gnr2dt4c3HYF8QxlLGBrkmxAtidpT97L/view?usp=sharing

Дополнительное задание 2 - вариант 0

Написать регулярное выражение, которое проверяет корректность email и в качестве ответа выдаёт почтовый сервер (почтовый сервер – часть email идущая после «@»). Для простоты будем считать, что почтовый адрес может содержать в себе буквы, цифры, «.» и «_», а почтовый сервер только буквы и «.». При этом почтовый сервер, обязательно должен содержать верхний уровень домена («.ru», «.com», etc.)

Программа :

```
import re

email1="example@example"
email2="students.spam@yandex.ru"
email3="example@example.com"
email4="jsebastian.va@gmail.com"

email=input('Enter your email:')

regex = '^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}$' #correct?

regexDomine='[@]\w+[.]\w{2,3}$'

m=re.search(regex,email)

if(m):
        domine=re.search(regexDomine,email)
        print("Domine:" + domine[0])
else:
        print("Fail!")
```

Результаты:

1. Enter your email:example@example
   Fail!
2. Enter your email:students.spam@yandex.ru
   Domine:@yandex.ru
3. Enter your email:example@example.com
   Domine:@example.com
4. Enter your email:jsebastian.va@gmail.com
   Domine:@gmail.com
5. Enter your email:jsebastian.va@gmail.c
   Fail!

Выводы:

Благодаря регулярным выражениям мы можем упрощать процессы для работы с символами и строкой, таким образом, процесс поиска данных становится более эффективным.

Сравнивая две функции для преобразования текста, вы можете заметить, что с несколькими строками кода, через регулярных выражений, вы делаете то же самое, что и со многими строками кода, не используя регулярные выражения

В первом дополнительном задании мы можем понять, что с несколькими строками кода и регулярным выражением мы можем эффективно фильтровать длинные тексты без каких-либо проблеми.

Во втором дополнительном задании мы можем понять, что регулярные выражения отлично работают для проверки данных, в этом случае электронные почти.