Студентка группы ИС-23 Васильева

Практическое занятие № 14

Тема: составление программ с использованием регулярных выражений в IDE PyCharm Community.

Цель: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ с использованием с использованием регулярных выражений вIDE PyCharm Community.

Постановка дополнительного упражнения:

```
# • Найдите все натуральные числа (возможно, окружённые буквами);
# • Найдите все «слова», написанные капсом (то есть строго заглавными), возможно внутри настоящих слов (аааБББввв);
# • Найдите слова, в которых есть русская буква, а за ней цифра;
# • Найдите все слова, начинающиеся с русской или латинской большой буквы (\b — граница слова);
# • Найдите слова, которые начинаются на гласную (\b — граница слова);
# • Найдите все натуральные числа, не находящиеся на границе слова;
# • Найдите строчки, в которых есть символ * (. — это точно не конец строки!);
# • Найдите строчки, в которых есть открывающая и когда-нибудь потом закрывающая скобки;
# • Выделите одним махом весь кусок оглавления (в конце примера, вместе с тегами);
# • Выделите одним махом только текстовую часть оглавления, без тегов;
# • Найдите пустые строчки;
# • Найти все теги, не включая их содержимое.
```

Тип алгоритма: циклический.

Текст программы:

```
import re
regex_list = [
    r'\b\d+\b',
    r'\b[A-ЯA-Z]+\b',
    r'\b\w*[A-Яa-я]\d\w*\b',
    r'\b[A-ЯA-Z]\w*\b',
    r'\b[АЕЁИОУЫЭЮЯAEIOUY]\w*\b',
    r'\B\d+\B',
    r'.*\*.*',
    r'.*\(.*\).*',
    r'<a\s.*?<\/a>',
    r'(?<=\>).*?(?=\<)',
    r'^\s*$',
    r'<[^>]+>'
]

test_str = ("Регулярные выражения представляют собой похожий, но гораздо более сильный инструмент \n"
 "для поиска строк, проверки их на соответствие какому-либо шаблону и другой подобной \n"
 "работы. Англоязычное название этого инструмента — Regular Expressions или просто RegExp. \n"
```

```
 "Строго говоря, регулярные выражения — специальный язык для описания
шаблонов строк.\n\n"
 "AAAA aaaa AaAaAaAa 123 123 12345 11223344\n"
 "A1Б2В3 АА11 ББ22ВВ 33ГГ44\n\n"
 "Тест!    Ещё!   Даёшь!   ЁЁЁёёё\n\n"
 "QwertyЙцукен\n\n"
 "+-,/[](), * (*), a*(b+[c+d])*e/f+g-h\n\n"
 "!!\"\"\"####$$$$$%%%%%&&&'''(((()))***++++,,,,,-----
..//:::;;;;<<<<<===>>>????\n"
 "@@@@@[[[[\\\\\\]]]]]^^^__`````{{{{|||||}}}}}~~~~~\n\n"
 "<a href=\"#10\">10: CamelCase -> under_score</a>;\n"
 "<a href=\"#11\">11: Удаление повторов</a>;\n"
 "<a href=\"#12\">12: Близкие слова</a>;\n"
 "<a href=\"#13\">13: Форматирование больших чисел</a>;\n"
 "<a href=\"#14\">14: Разделить текст на предложения</a>;\n"
 "<a href=\"#15\">15: Форматирование номера телефона</a>;\n"
 "<a href=\"#16\">16: Поиск e-mail'ов — 2</a>;\n")

for regex in regex_list:
    matches = re.finditer(regex, test_str)

    print("Results for regex: {}".format(regex))
    for matchNum, match in enumerate(matches, start=1):
        print ("Match {matchNum} was found at {start}-{end}:
{match}".format(matchNum=matchNum, start=match.start(), end=match.end(),
match=match.group()))
```

Протокол работы программы:

Results for regex: \b\d+\b

Match 1 was found at 365-368: 123

Match 2 was found at 369-372: 123

Match 3 was found at 373-378: 12345

Match 4 was found at 379-387: 11223344

Match 5 was found at 633-635: 10

Match 6 was found at 637-639: 10

Match 7 was found at 681-683: 11

Match 8 was found at 685-687: 11

Match 9 was found at 722-724: 12

Match 10 was found at 726-728: 12

Match 11 was found at 759-761: 13

Match 12 was found at 763-765: 13

Match 13 was found at 811-813: 14

Match 14 was found at 815-817: 14

Match 15 was found at 865-867: 15

Match 16 was found at 869-871: 15

Match 17 was found at 919-921: 16

Match 18 was found at 923-925: 16

Match 19 was found at 945-946: 2

Results for regex: \b[A-ЯA-Z]+\b

Match 1 was found at 346-350: AAAA

Results for regex: \b\w*[A-Яa-я]\d\w*\b

Match 1 was found at 388-394: А1Б2В3

Match 2 was found at 395-399: AA11

Match 3 was found at 400-406: ББ22ВВ

Match 4 was found at 407-413: 33ГГ44

Results for regex: \b[A-ЯA-Z]\w*\b

Match 1 was found at 0-10: Регулярные

Match 2 was found at 179-191: Англоязычное

Match 3 was found at 221-228: Regular

Match 4 was found at 229-240: Expressions

Match 5 was found at 252-258: RegExp

Match 6 was found at 261-267: Строго

Match 7 was found at 346-350: AAAA

Match 8 was found at 356-364: AaAaAaAa

Match 9 was found at 388-394: А1Б2В3

Match 10 was found at 395-399: AA11

Match 11 was found at 400-406: ББ22ВВ

Match 12 was found at 415-419: Тест

Match 13 was found at 424-427: Ещё

Match 14 was found at 431-436: Даёшь

Match 15 was found at 448-460: QwertyЙцукен

Match 16 was found at 641-650: CamelCase

Match 17 was found at 689-697: Удаление

Match 18 was found at 730-737: Близкие

Match 19 was found at 767-781: Форматирование

Match 20 was found at 819-828: Разделить

Match 21 was found at 873-887: Форматирование

Match 22 was found at 927-932: Поиск

Results for regex: \b[АЕЁИОУЫЭЮЯAEIOUY]\w*\b

Match 1 was found at 179-191: Англоязычное

Match 2 was found at 229-240: Expressions

Match 3 was found at 346-350: AAAA

Match 4 was found at 356-364: AaAaAaAa

Match 5 was found at 388-394: А1Б2В3

Match 6 was found at 395-399: AA11

Match 7 was found at 424-427: Ещё

Match 8 was found at 440-446: ЁЁЁёёё

Match 9 was found at 689-697: Удаление

Results for regex: \B\d+\B

Match 1 was found at 366-367: 2

Match 2 was found at 370-371: 2

Match 3 was found at 374-377: 234

Match 4 was found at 380-386: 122334

Match 5 was found at 389-390: 1

Match 6 was found at 391-392: 2

Match 7 was found at 397-398: 1

Match 8 was found at 402-404: 22

Match 9 was found at 408-409: 3

Match 10 was found at 411-412: 4

Results for regex: .**.*

Match 1 was found at 462-498: +-,/[](), * (*), a*(b+[c+d])*e/f+g-h

Match 2 was found at 500-574: !!"""####$$$$$%%%%&&&'''(((())***++++,,,,,-----..//:::;;;;<<<<<===>>>????

Results for regex: .*\(.*\).*

Match 1 was found at 462-498: +-,/[](), * (*), a*(b+[c+d])*e/f+g-h

Match 2 was found at 500-574: !!"""####$$$$$%%%%&&&'''(((())***++++,,,,,-----..//:::;;;;<<<<<===>>>????

Results for regex: <a\s.*?<\/a>

Match 1 was found at 623-669: 10: CamelCase -> under_score

Match 2 was found at 671-710: 11: Удаление повторов

Match 3 was found at 712-747: 12: Близкие слова

Match 4 was found at 749-799: 13: Форматирование больших чисел

Match 5 was found at 801-853: 14: Разделить текст на предложения

Match 6 was found at 855-907: 15: Форматирование номера телефона

Match 7 was found at 909-950: 16: Поиск e-mail'ов — 2

Results for regex: (?<=\>).*?(?=\<)

Match 1 was found at 637-665: 10: CamelCase -> under_score

Match 2 was found at 685-706: 11: Удаление повторов

Match 3 was found at 726-743: 12: Близкие слова

Match 4 was found at 763-795: 13: Форматирование больших чисел

Match 5 was found at 815-849: 14: Разделить текст на предложения

Match 6 was found at 869-903: 15: Форматирование номера телефона

Match 7 was found at 923-946: 16: Поиск e-mail'ов — 2

Results for regex: ^\s*$

Results for regex: <[^>]+>

Match 1 was found at 559-568: <<<<<===>

Match 2 was found at 623-637:

Match 3 was found at 665-669:

Match 4 was found at 671-685:

Match 5 was found at 706-710:

Match 6 was found at 712-726:

Match 7 was found at 743-747:

Match 8 was found at 749-763:

Match 9 was found at 795-799:

Match 10 was found at 801-815:

Match 11 was found at 849-853:

Match 12 was found at 855-869:

Match 13 was found at 903-907:

Match 14 was found at 909-923:

Match 15 was found at 946-950:

Process finished with exit code 0

Постановка задачи № 1:

Из исходного текстового файла (ip_address.txt) из раздела «Частоупотребимые маски» перенести в первый файл строки с нулевым четвертым октетом, а во второй – все остальные. Посчитать количество полученных строк в каждом файле.

Тип алгоритма: циклический.

Текст программы:

```
import re

with open("ip_address.txt", "r") as file:
    data = file.read()

zero_octet_lines = [line for line in re.findall(r"\b255\.255\.255\.0+\b",
data)]
other_lines = [line for line in re.findall(r"\b255\.255\.255\.\d+\b", data)]

with open("zero_octet.txt", "w") as file:
    for line in zero_octet_lines:
        file.write(line + "\n")

other_lines = [line for line in re.findall(r"\b255\.255\.255\.\d+\b", data)
if not re.match(r"\b255\.255\.255\.0+\b", line)]

with open("other_octets.txt", "w") as file:
    for line in other_lines:
        file.write(line + "\n")

zero_octet_count = len(zero_octet_lines)
other_count = len(other_lines)

print(f"Количество строк с нулевым четвертым октетом: {zero_octet_count}")
print(f"Количество строк с другими четвертыми октетами: {other_count}")
```

Протокол работы программы:

Количество строк с нулевым четвертым октетом: 1
Количество строк с другими четвертыми октетами: 7

Process finished with exit code 0

Вывод: Я закрепила усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрела навыки составления программ с использованием регулярных выражений в IDE PyCharm Community.

Выполнены разработка кода, отладка, тестирование, оптимизация программного кода.

Готовые программные коды выложены на GitHub.