Регулярные выражения для задачи №24

Важно: не каждую задачу можно решить с помощью регулярных выражений. Это не универсальный способ решения.

Задачи взяты с kompege, вместо файликов взяты 100 символов из файла для наглядности. Но если захочется запустить код на файликах, то можно скачать здесь.

Полезные источники:

- Лекция Алексея Кабанова
- Лекция Юрия Ипатова в Сириусе

re — regular expressions — регулярные выражения.

Это стандартная библиотека, устанавливать её не нужно. На ЕГЭ ей можно будет воспользоваться.

Примеры решений

import re

r перед строкой делает текст "сырым" и не использует специальные символы

рассмотрим примеры. \n — символ переноса на новую строку.

text = '1\n2'

print(text)

# если добавить r, то \ будет восприниматься как обычный символ

text = r'1\n2'

print(text)

1

2

1\n2

возьмём за привычку писать регулярные выражения в r"", иногда это будет важно

Задача №1

№ 3792 (Уровень: Базовый)

В текстовом файле находится цепочка из символов латинского алфавита A, B, C, D, E. Найдите длину самой длинной подцепочки, состоящей из символов A, B или C (в произвольном порядке).

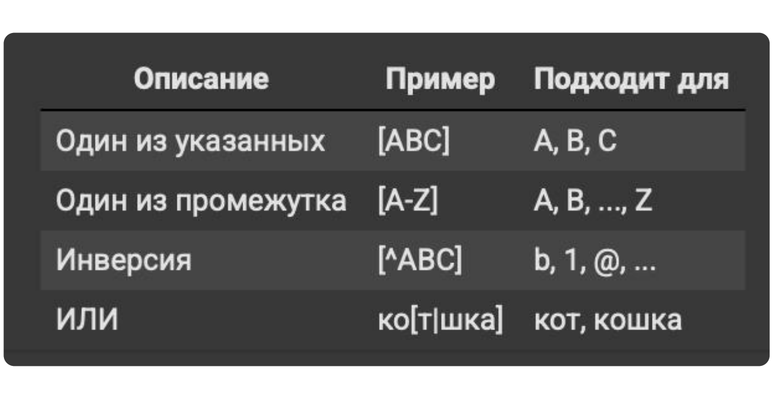
В комментариях оставлю использование файла, если захочется поиграться с данными целиком.

```
# f_name = '24_3792.txt'

# with open(f_name) as f:

#    text = f.readline()

text = 'AEECBACEDDEEACECBADAAAEECECEEBAEEACBDEDBEEABDBBBCBAABBABBDBABEEACBEEBDAEAEDDBDECBABBCADBBEBDCABCEDCC'

pattern = r'[ABC]+'

lines = re.findall(pattern, text)

print(lines)

line_max = max(lines, key=len)

print(len(line_max), line_max)
```

```
['A', 'CBAC', 'AC', 'CBA', 'AAA', 'C', 'C', 'BA', 'ACB', 'B', 'AB', 'BBBCBAABBABB', 'BAB', 'ACB', 'B', 'A', 'A', 'B', 'CBABBCA', 'BB', 'B', 'CABC', 'CC']
```

12 BBBCBAABBABB

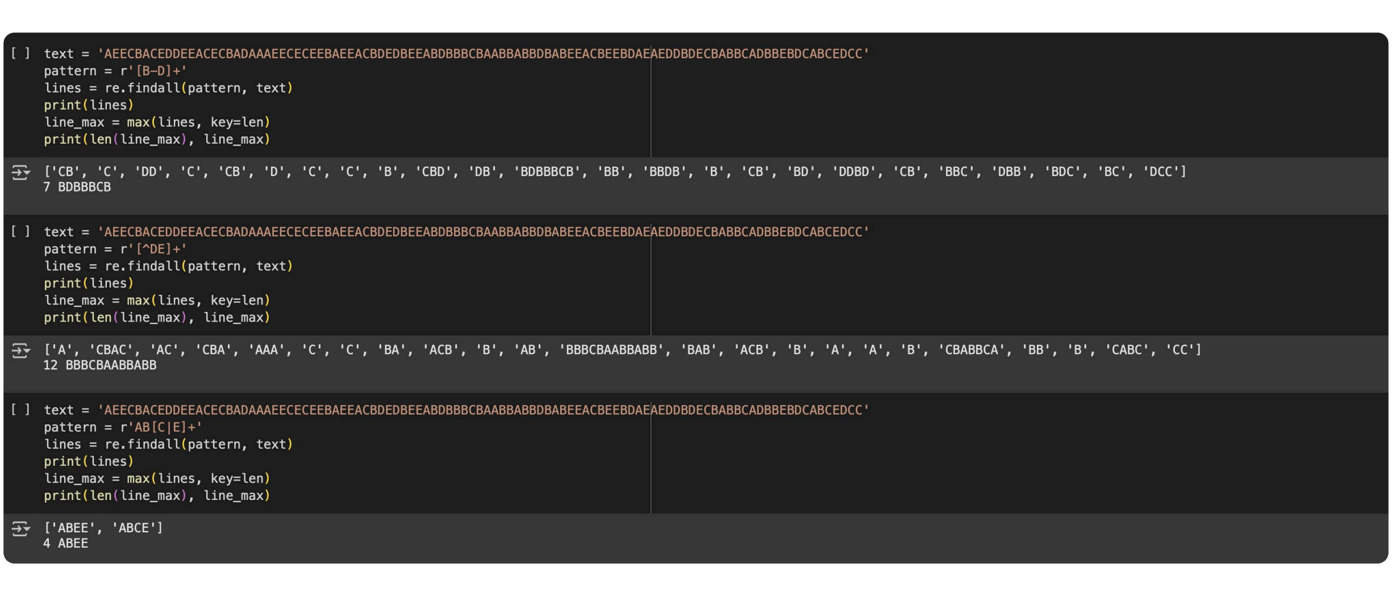
Если метасимволов не хватает, можно использовать символьные классы.



Описание	Пример	Подходит для
Один из указанных	[ABC]	A, B, C
Один из промежутка	[A-Z]	A, B, ..., Z
Инверсия	[^ABC]	b, 1, @, ...
ИЛИ	ко[т|шка]	кот, кошка

Символьные классы

Попробуем на примере



```
[ ] text = 'AEECBACEDDEEACECBADAAAEECECEEBAEEACBDEDBEEABDBBBCBAABBABBDBABEEACBEEBDAEAEDDBDECBABBCADBBEBDCABCEDCC'
    pattern = r'[B-D]+'
    lines = re.findall(pattern, text)
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max), line_max)
```

```
['CB', 'C', 'DD', 'C', 'CB', 'D', 'C', 'C', 'B', 'CBD', 'DB', 'BDBBBCB', 'BB', 'BBDB', 'B', 'CB', 'BD', 'DDBD', 'CB', 'BBC', 'DBB', 'BDC', 'BC', 'DCC']
7 BDBBBCB
```

```
[ ] text = 'AEECBACEDDEEACECBADAAAEECECEEBAEEACBDEDBEEABDBBBCBAABBABBDBABEEACBEEBDAEAEDDBDECBABBCADBBEBDCABCEDCC'
    pattern = r'[^DE]+'
    lines = re.findall(pattern, text)
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max), line_max)
```

```
['A', 'CBAC', 'AC', 'CBA', 'AAA', 'C', 'C', 'BA', 'ACB', 'B', 'AB', 'BBBCBAABBABB', 'BAB', 'ACB', 'B', 'A', 'A', 'B', 'CBABBCA', 'BB', 'B', 'CABC', 'CC']
12 BBBCBAABBABB
```

```
[ ] text = 'AEECBACEDDEEACECBADAAAEECECEEBAEEACBDEDBEEABDBBBCBAABBABBDBABEEACBEEBDAEAEDDBDECBABBCADBBEBDCABCEDCC'
    pattern = r'AB[C|E]+'
    lines = re.findall(pattern, text)
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max), line_max)
```

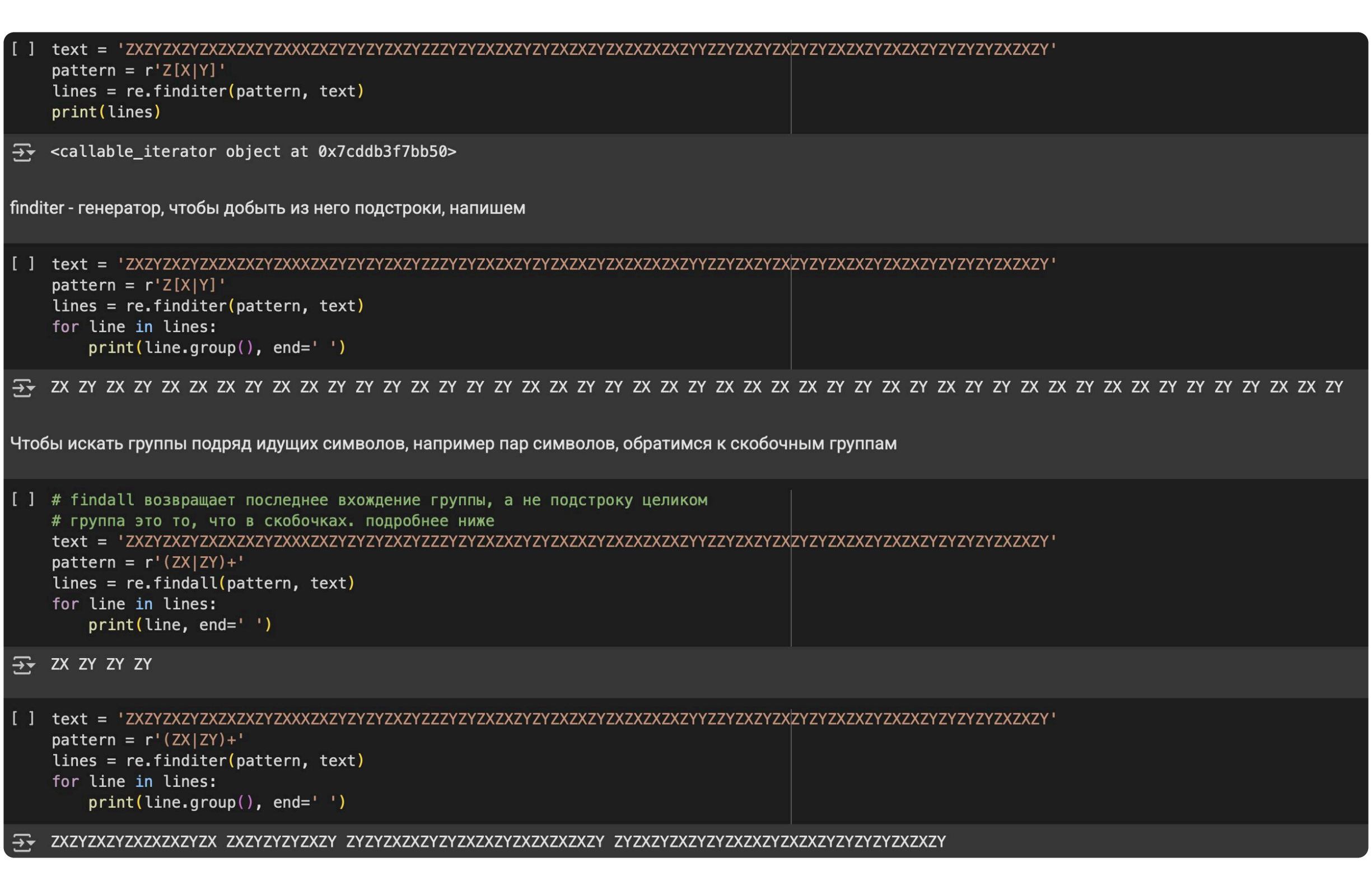
```
['ABEE', 'ABCE']
4 ABEE
```

Задача №2

№ 3020 (Уровень: Базовый)

Текстовый файл содержит строку из заглавных латинских букв X, Y и Z, всего не более чем из 10^6 символов. Определите максимальное количество идущих подряд пар символов ZX или ZY.

Чтобы искать группы символов, например подряд идущие пары, findall не подойдёт. Воспользуемся finditer



```
[ ] text = 'ZXZYZXZYZXZXZXZYZXXXZXZYZYZYZXZYZZZYZYZXZXZYZYZXZXZYZXZXZXZXZYYZZYZXZYZXZYZYZXZXZYZXZXZYZYZYZYZXZXZY'
    pattern = r'Z[X|Y]'
    lines = re.finditer(pattern, text)
    print(lines)
```

```
<callable_iterator object at 0x7cddb3f7bb50>
```

finditer - генератор, чтобы добыть из него подстроки, напишем

```
[ ] text = 'ZXZYZXZYZXZXZXZYZXXXZXZYZYZYZXZYZZZYZYZXZXZYZYZXZXZYZXZXZXZXZYYZZYZXZYZXZYZYZXZXZYZXZXZYZYZYZYZXZXZY'
    pattern = r'Z[X|Y]'
    lines = re.finditer(pattern, text)
    for line in lines:
        print(line.group(), end=' ')
```

```
ZX ZY ZX ZY ZX ZX ZX ZY ZX ZX ZY ZY ZY ZX ZY ZY ZY ZX ZX ZY ZY ZX ZX ZY ZX ZX ZX ZX ZY ZY ZX ZY ZX ZY ZY ZX ZX ZY ZX ZX ZY ZY ZY ZY ZX ZX ZY
```

Чтобы искать группы подряд идущих символов, например пар символов, обратимся к скобочным группам

```
[ ] # findall возвращает последнее вхождение группы, а не подстроку целиком
    # группа это то, что в скобочках. подробнее ниже
    text = 'ZXZYZXZYZXZXZXZYZXXXZXZYZYZYZXZYZZZYZYZXZXZYZYZXZXZYZXZXZXZXZYYZZYZXZYZXZYZYZXZXZYZXZXZYZYZYZYZXZXZY'
    pattern = r'(ZX|ZY)+'
    lines = re.findall(pattern, text)
    for line in lines:
        print(line, end=' ')
```

```
ZX ZY ZY ZY
```

```
[ ] text = 'ZXZYZXZYZXZXZXZYZXXXZXZYZYZYZXZYZZZYZYZXZXZYZYZXZXZYZXZXZXZXZYYZZYZXZYZXZYZYZXZXZYZXZXZYZYZYZYZXZXZY'
    pattern = r'(ZX|ZY)+'
    lines = re.finditer(pattern, text)
    for line in lines:
        print(line.group(), end=' ')
```

```
ZXZYZXZYZXZXZXZYZX ZXZYZYZYZXZY ZYZYZXZXZYZYZXZXZYZXZXZXZXZY ZYZXZYZXZYZYZXZXZYZXZXZYZYZYZYZXZXZY
```

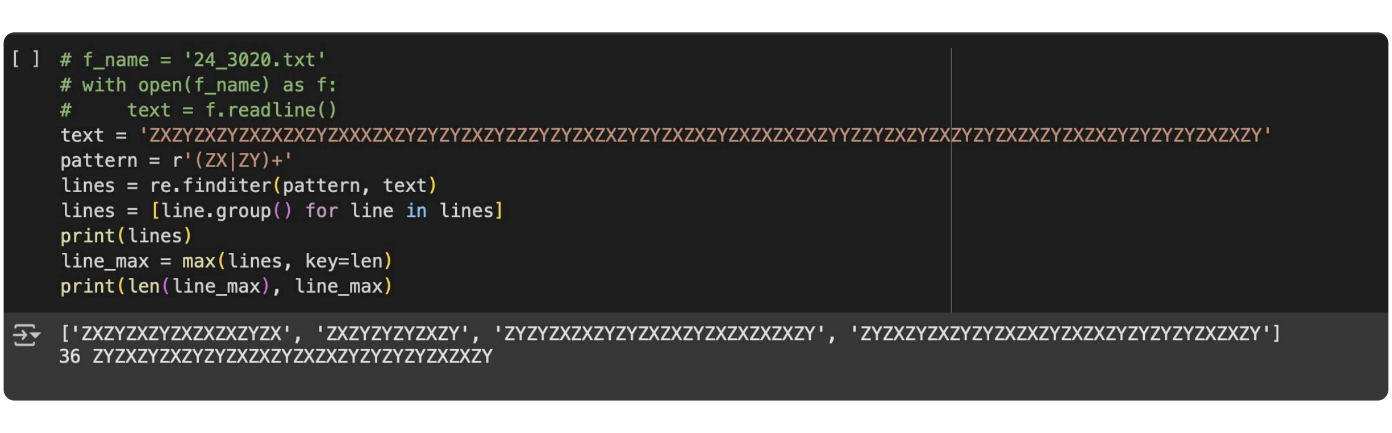
Скобочные группы находят непересекающиеся подстроки.

Например, в тексте ABABA ABA встречается дважды, но код ниже найдёт подстроку только один раз, так как подстроки пересекаются. Как с этим справиться, разберёмся чуть ниже в разделе «Скобочные группы».

```
text = 'ABABA'

pattern = r'(ABA)'

lines = re.finditer(pattern, text)

for line in lines:

  print(line.group(), end=' ')
```

ABA

Решим задачу полностью



```
[ ] # f_name = '24_3020.txt'
    # with open(f_name) as f:
    #     text = f.readline()
    text = 'ZXZYZXZYZXZXZXZYZXXXZXZYZYZYZXZYZZZYZYZXZXZYZYZXZXZYZXZXZXZXZYYZZYZXZYZXZYZYZXZXZYZXZXZYZYZYZYZXZXZY'
    pattern = r'(ZX|ZY)+'
    lines = re.finditer(pattern, text)
    lines = [line.group() for line in lines]
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max), line_max)
```

```
['ZXZYZXZYZXZXZXZYZX', 'ZXZYZYZYZXZY', 'ZYZYZXZXZYZYZXZXZYZXZXZXZXZY', 'ZYZXZYZXZYZYZXZXZYZXZXZYZYZYZYZXZXZY']
36 ZYZXZYZXZYZYZXZXZYZXZXZYZYZYZYZXZXZY
```

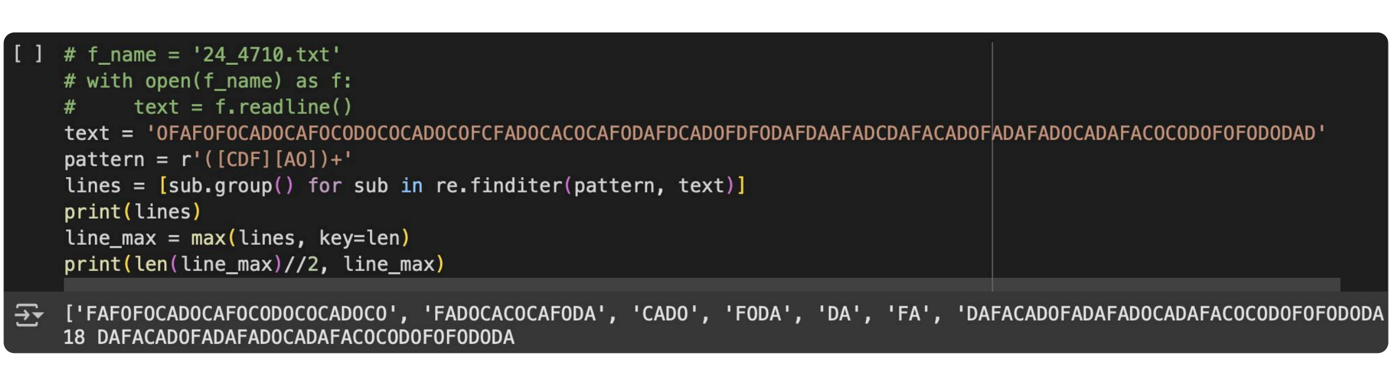
Задача №3

№ 4710 Демоверсия 2023 (Уровень: Базовый)

Текстовый файл состоит из символов A, C, D, F и O.

Определите максимальное количество идущих подряд пар символов вида
согласная + гласная
в прилагаемом файле.

Для выполнения этого задания следует написать программу.

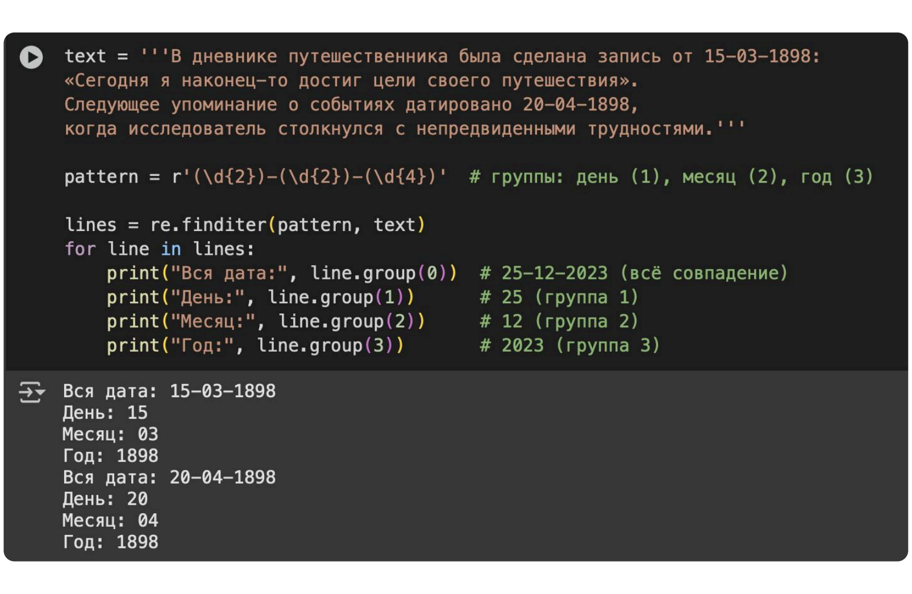


```
[ ] # f_name = '24_4710.txt'
    # with open(f_name) as f:
    #     text = f.readline()
    text = 'OFAFOFOCADOCAFOCODOCOCADOCOFCFADOCACOCAFODAFDCADOFDFODAFDAAFADCDAFACADOFADAFADOCADAFACOCODOFOFODODAD'
    pattern = r'([CDF][AO])+'
    lines = [sub.group() for sub in re.finditer(pattern, text)]
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max)//2, line_max)
```

```
['FAFOFOCADOCAFOCODOCOCADOCO', 'FADOCACOCAFODA', 'CADO', 'FODA', 'DA', 'FA', 'DAFACADOFADAFADOCADAFACOCODOFOFODODA
18 DAFACADOFADAFADOCADAFACOCODOFOFODODA
```

Скобочные группы

С помощью круглых скобок можно создать группы в регулярном выражении. Они позволяют выделить части шаблона, к которым можно обратиться отдельно после нахождения совпадения.



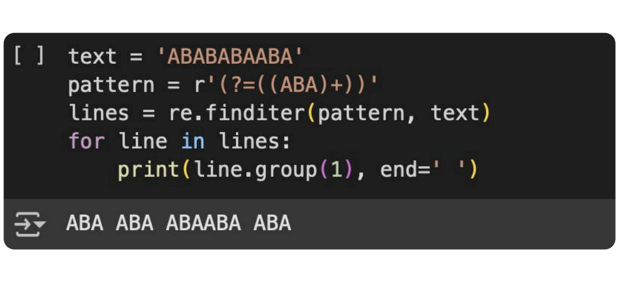
```
text = '''В дневнике путешественника была сделана запись от 15-03-1898:
«Сегодня я наконец-то достиг цели своего путешествия».
Следующее упоминание о событиях датировано 20-04-1898,
когда исследователь столкнулся с непредвиденными трудностями.'''

pattern = r'(\d{2})-(\d{2})-(\d{4})'  # группы: день (1), месяц (2), год (3)

lines = re.finditer(pattern, text)
for line in lines:
    print("Вся дата:", line.group(0))  # 25-12-2023 (всё совпадение)
    print("День:", line.group(1))      # 25 (группа 1)
    print("Месяц:", line.group(2))     # 12 (группа 2)
    print("Год:", line.group(3))       # 2023 (группа 3)
```

```
Вся дата: 15-03-1898
День: 15
Месяц: 03
Год: 1898
Вся дата: 20-04-1898
День: 20
Месяц: 04
Год: 1898
```

Чтобы искать подстроки с пересечением, воспользуемся просмотром вперёд (?=). Он перебором ищет совпадение с шаблоном с каждого символа, тем самым находя даже пересекающиеся подстроки.

Сам просмотр вперёд ничего не возвращает, поэтому выбираем первую группу при выводе.

Важно весь шаблон после знака = брать в скобки.



```
[ ] text = 'ABABABAABA'
    pattern = r'(?=((ABA)+))'
    lines = re.finditer(pattern, text)
    for line in lines:
        print(line.group(1), end=' ')
```

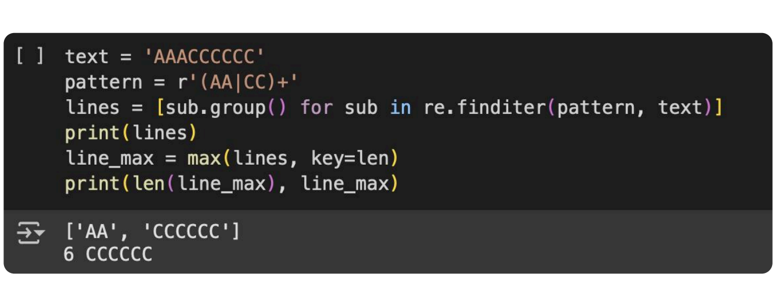
```
ABA ABA ABAABA ABA
```

Задача №4

Задача с сайта К.Ю. Полякова

204 (Е. Джобс). Текстовый файл 24-204.txt содержит строку из заглавных латинских букв A, B и C, всего не более чем из 10^6 символов. Найдите максимальное количество подряд идущих пар символов AA или CC. Искомая подстрока может включать только пары AA, только пары CC или содержать одновременно как пары AA, так и пары CC.

Подстроки могут пересекаться, поэтому обычный поиск не сможет найти все совпадения.



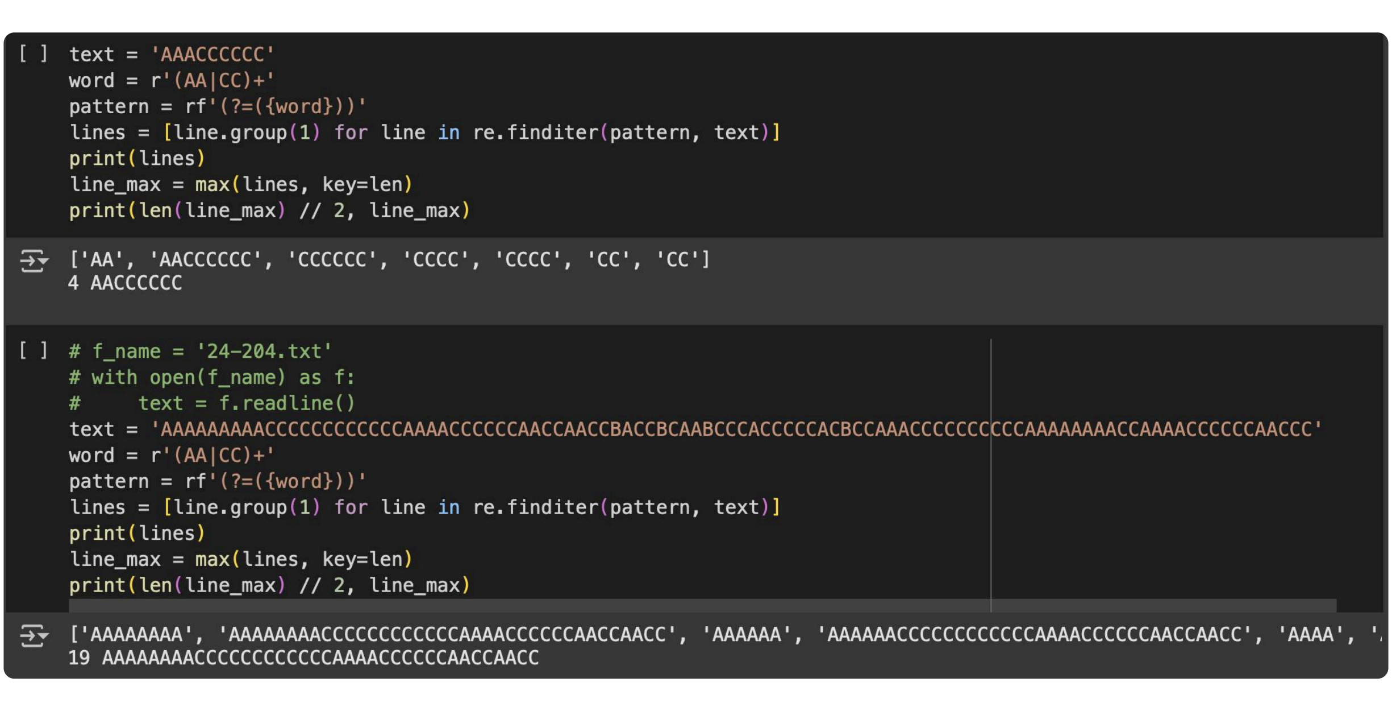
```
[ ] text = 'AAACCCCCC'
    pattern = r'(AA|CC)+'
    lines = [sub.group() for sub in re.finditer(pattern, text)]
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max), line_max)
```

```
['AA', 'CCCCCC']
6 CCCCCC
```

Воспользуемся просмотром вперёд (?=)

Просмотр вперёд проверяет вхождения начиная с каждого символа. Так мы сможем найти даже пересекающиеся подстроки.

Проверьте использование команды на файле целиком. Даже на миллионе символов код выполняется довольно быстро.



```
[ ] text = 'AAACCCCCC'
    word = r'(AA|CC)+'
    pattern = rf'(?=({word}))'
    lines = [line.group(1) for line in re.finditer(pattern, text)]
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max) // 2, line_max)
```

```
['AA', 'AACCCCCC', 'CCCCCC', 'CCCC', 'CCCC', 'CC', 'CC']
4 AACCCCCC
```

```
[ ] # f_name = '24-204.txt'
    # with open(f_name) as f:
    #     text = f.readline()
    text = 'AAAAAAAAACCCCCCCCCCCAAAACCCCCCAACCAACCBACCBCAABCCCACCCCCACBCCAAACCCCCCCCCCAAAAAAAAACCAAAACCCCCCAACCC'
    word = r'(AA|CC)+'
    pattern = rf'(?=({word}))'
    lines = [line.group(1) for line in re.finditer(pattern, text)]
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max) // 2, line_max)
```

```
['AAAAAAAA', 'AAAAAAAACCCCCCCCCCCAAAACCCCCCAACCAACC', 'AAAAAA', 'AAAAAACCCCCCCCCCCAAAACCCCCCAACCAACC', 'AAAA', '
19 AAAAAAAACCCCCCCCCCCAAAACCCCCCAACCAACC
```

Практика

Дальше новых конструкций не будет. Попробуйте решить задачки сами, затем посмотрите их решения.

Задача №1

Задача с сайта К.Ю. Полякова

219. Текстовый файл 24-215.txt содержит строку из символов A, B, C и цифр 1, 2, 3, всего не более чем 10^6 символов. Определите максимальное количество идущих подряд троек символов вида «цифра+ буква+ цифра».

Задача №2

Задача с сайта К.Ю. Полякова

239 (Е. Джобс). Текстовый файл 24-239.txt состоит не более чем из 10^6 символов и содержит только латинские буквы X, У, Z. Найдите максимальную длину подстроки, которая состоит из сочетаний XV, YZ, YZZ, записанных в произвольном порядке. В ответе укажите наибольшую длину подходящей подстроки. Например, в строке ZZXZXZZXYYZYZZYYY самая длинная подходящая подстрока — XYYZYZZ имеет длину 7.

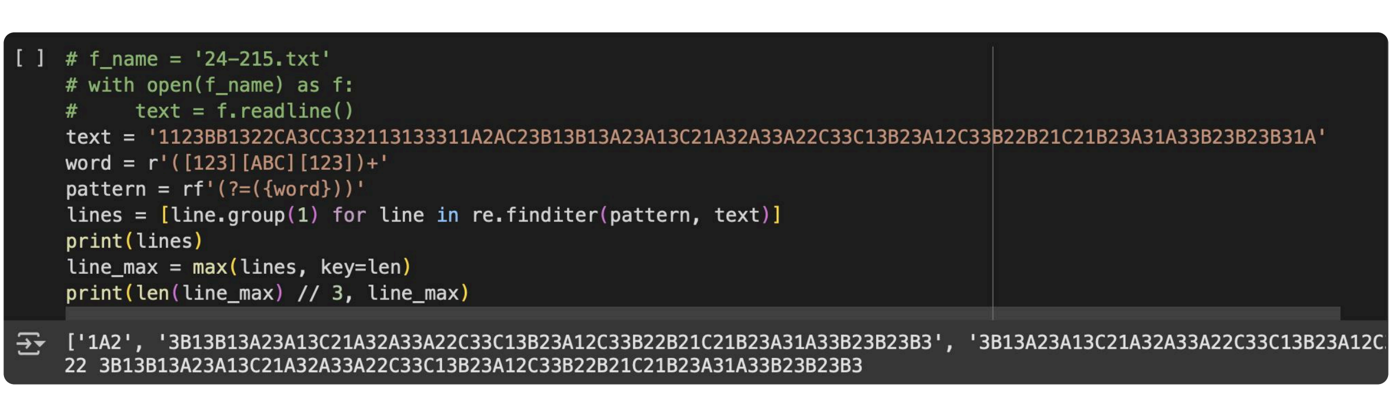
Задача №3

Задача с сайта К.Ю. Полякова

313. Текстовый файл 2-310.txt состоит не более чем из 10^6 символов и содержит только цифры шестнадцатеричной системы счисления и знаки арифметических операций «+» и «*». Определите максимальное количество символов в непрерывной последовательности, которая является корректным арифметическим выражением с целыми неотрицательными числами, записанными в троичной системе счисления. В этом выражении никакие два знака арифметических операций не стоят рядом, в записи чисел отсутствуют незначащие (ведущие) нули и число 0 не имеет знака.

Ответы

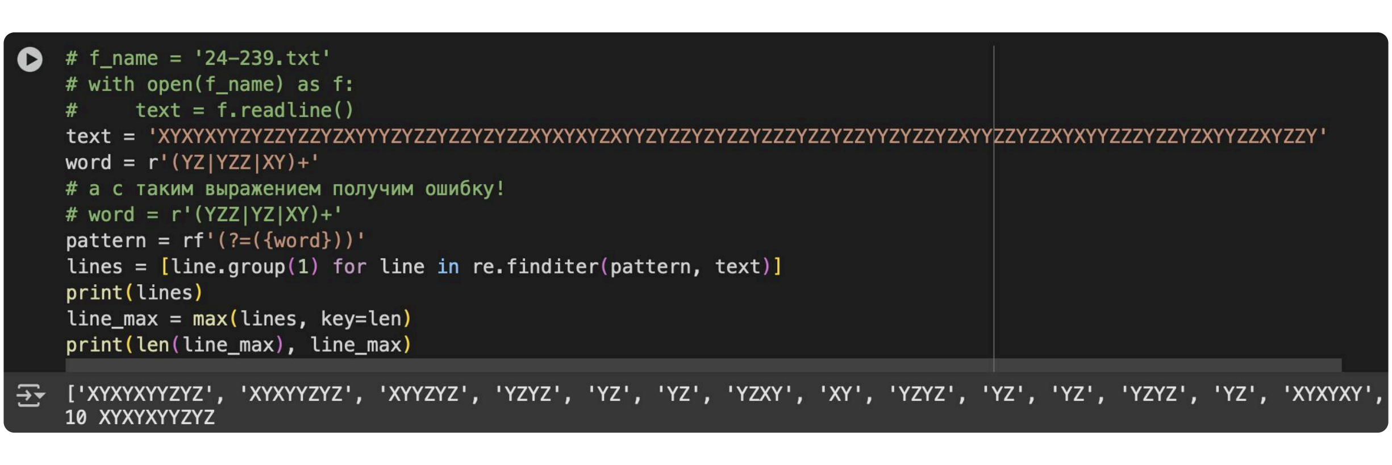
Задача №1



```
[ ] # f_name = '24-215.txt'
    # with open(f_name) as f:
    #     text = f.readline()
    text = '1123BB1322CA3CC332113133311A2AC23B13B13A23A13C21A32A33A22C33C13B23A12C33B22B21C21B23A31A33B23B23B31A'
    word = r'([123][ABC][123])+'
    pattern = rf'(?=({word}))'
    lines = [line.group(1) for line in re.finditer(pattern, text)]
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max) // 3, line_max)
```

```
['1A2', '3B13B13A23A13C21A32A33A22C33C13B23A12C33B22B21C21B23A31A33B23B23B3', '3B13A23A13C21A32A33A22C33C13B23A12C
22 3B13B13A23A13C21A32A33A22C33C13B23A12C33B22B21C21B23A31A33B23B23B3
```

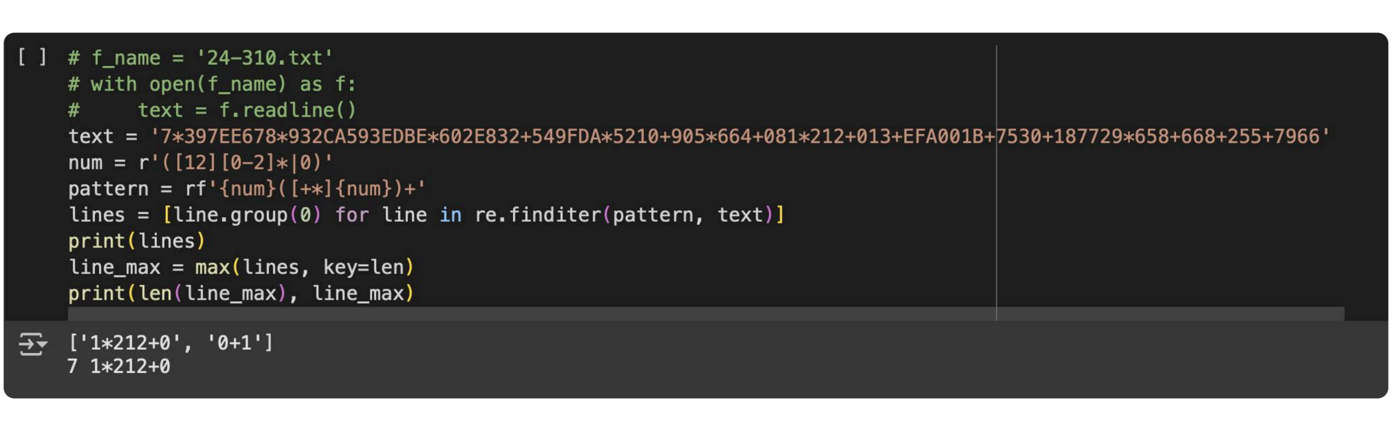
Задача №2



```
# f_name = '24-239.txt'
# with open(f_name) as f:
#     text = f.readline()
text = 'XYXYXYYZYZZYZZYZXYYYZYZZYZZYZYZZXYXYXYZXYYZYZZYZYZZYZZZYZZYZZYYZYZZYZXYYZZYZZXYXYYZZZYZZYZXYYZZXYZZY'
word = r'(YZ|YZZ|XY)+'
# а с таким выражением получим ошибку!
# word = r'(YZZ|YZ|XY)+'
pattern = rf'(?=({word}))'
lines = [line.group(1) for line in re.finditer(pattern, text)]
print(lines)
line_max = max(lines, key=len)
print(len(line_max), line_max)
```

```
['XYXYXYYZYZ', 'XYXYYZYZ', 'XYYZYZ', 'YZYZ', 'YZ', 'YZ', 'YZXY', 'XY', 'YZYZ', 'YZ', 'YZ', 'YZYZ', 'YZ', 'XYXYXY',
10 XYXYXYYZYZ
```

Задача №3



```
[ ] # f_name = '24-310.txt'
    # with open(f_name) as f:
    #     text = f.readline()
    text = '7*397EE678*932CA593EDBE*602E832+549FDA*5210+905*664+081*212+013+EFA001B+7530+187729*658+668+255+7966'
    num = r'([12][0-2]*|0)'
    pattern = rf'{num}([+*]{num})+'
    lines = [line.group(0) for line in re.finditer(pattern, text)]
    print(lines)
    line_max = max(lines, key=len)
    print(len(line_max), line_max)
```

```
['1*212+0', '0+1']
7 1*212+0
```

Попробуйте по аналогии решить несколько задачек на нашей платформе (ссылка ведёт на базу задач).

по информатике
с Яндекс Учебником
LVL UP
ЕГЭ
ПО ИНФОРМАТИКЕ
С ЯНДЕКС УЧЕБНИКОМ