

**КАЛУЖСКИЙ ФИЛИАЛ
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ Н.Э. БАУМАНА
(национальный исследовательский университет)»**



Факультет «Информатика и управление»

Кафедра «Программное обеспечение ЭВМ, информационные технологии»

Высокоуровневое программирование

Лекция №13. «Обработка текстов. Регулярные выражения»

Калуга - 2020

Регулярные выражения

- **Регулярные выражения** (regular expressions) описывают множество строк, используя специальный язык. (Строка, в которой задано регулярное выражение, будет называться **шаблоном**.)
- Для работы с регулярными выражениями в Python используется модуль **re**.

```
1 import re
2 pattern = r"[0-9]+"
3 number_re = re.compile(pattern)
4 number_re.findall("122 234 65435")
```

```
['122', '234', '65435']
```

Регулярные выражения

- В этом примере шаблон *pattern* описывает множество строк, которые состоят из одного или более символов из набора "0", "1" , ..., "9" .
- Функция *re.compile()* компилирует шаблон в специальный Regex-объект, который имеет несколько методов, в том числе метод *findall()* для получения списка всех непересекающихся вхождений строк, удовлетворяющих шаблону, в заданную строку.
- То же самое можно было сделать и так:

```
1 import re
2 re.findall(r"[0-9]+", "122 234 65435")
```

```
['122', '234', '65435']
```

- Предварительная компиляция шаблона предпочтительнее при его частом использовании, особенно внутри цикла.

Синтаксис регулярных выражений

- Синтаксис регулярных выражений в **Python** почти такой же, как в **Perl**, **grep** и некоторых других инструментах.
- Часть символов (в основном буквы и цифры) обозначают сами себя.
- Строка удовлетворяет (соответствует) шаблону, если она входит во множество строк, которые этот шаблон описывает.
- Здесь стоит также отметить, что различные операции используют шаблон по-разному. Так, *search()* ищет первое вхождение строки, удовлетворяющей шаблону, в заданной строке, а *match()* требует, чтобы строка удовлетворяла шаблону с самого начала.

Синтаксис регулярных выражений

Символ	Что обозначает в регулярном выражении
"."	Любой символ
"^"	Начало строки
"\$"	Конец строки
"*"	Повторение фрагмента нуль или более раз (жадное)
"+"	Повторение фрагмента один или более раз (жадное)
"?"	Предыдущий фрагмент либо присутствует, либо отсутствует
"{m, n}"	Повторение предыдущего фрагмента от m до n раз включительно (жадное)
"[...]"	Любой символ из набора в скобках. Можно задавать диапазоны символов с идущими подряд кодами, например: $a-z$
"[^...]"	Любой символ не из набора в скобках
"\""	Обратная косая черта отменяет специальное значение следующего за ней символа
" "	Фрагмент справа или фрагмент слева
"*?"	Повторение фрагмента нуль или более раз (не жадное)
"+?"	Повторение фрагмента один или более раз (не жадное)
"{m, n}?"	Повторение предыдущего фрагмента от m до n раз включительно (не жадное)

Синтаксис регулярных выражений

- Если А и В – регулярные выражения, то их конкатенация АВ является новым регулярным выражением.
- Можно считать, что **конкатенация** – основной способ составления регулярных выражений.
- Скобки, описанные ниже, применяются для задания приоритетов и выделения групп (фрагментов текста, которые потом можно получить по номеру или из словаря, и даже сослаться в том же регулярном выражении).
- Алгоритм, который сопоставляет строки с регулярным выражением, проверяет соответствие того или иного фрагмента строки регулярному выражению. Например, строка "a" соответствует регулярному выражению "[a-z]", строка "fruit" соответствует "fruit|vegetable", а вот строка "apple" не соответствует шаблону "pineapple".

Синтаксис регулярных выражений

Обозначение	Описание
" (регвыр) "	Обособляет регулярное выражение в скобках и выделяет группу
" (? :регвыр) "	Обособляет регулярное выражение в скобках без выделения группы
" (?=регвыр) "	Взгляд вперед: строка должна соответствовать заданному регулярному выражению, но дальнейшее сопоставление с шаблоном начнется с того же места
" (?!регвыр) "	То же, но с отрицанием соответствия
" (?<=регвыр) "	Взгляд назад: строка должна соответствовать, если до этого момента соответствует регулярному выражению. Не занимает места в строке, к которой применяется шаблон. Параметр <code>регвыр</code> должен быть фиксированной длины (то есть, без "+" и "*")
" (?<!регвыр) "	То же, но с отрицанием соответствия
" (?P<имя>регвыр) "	Выделяет именованную группу с именем <code>имя</code>
" (?P=имя) "	Точно соответствует выделенной ранее именованной группе с именем <code>имя</code>
" (?#регвыр) "	Комментарий (игнорируется)
" (? (имя) рв1 рв2) "	Если группа с номером или именем <code>имя</code> оказалась определена, результатом будет сопоставление с <code>рв1</code> , иначе - с <code>рв2</code> . Часть <code> рв2</code> может отсутствовать
" (?флаг) "	Задаёт флаг для всего данного регулярного выражения. Флаги необходимо задавать в начале шаблона

Синтаксис регулярных выражений

Последовательность	Чему соответствует
"\1" – "\9"	Группа с указанным номером. Группы нумеруются, начиная с 1
"\A"	Промежуток перед началом всей строки (почти аналогично "^")
"\Z"	Промежуток перед концом всей строки (почти аналогично "\$")
"\b"	Промежуток между символами перед словом или после него
"\B"	Наоборот, не соответствует промежутку между символами на границе слова
"\d"	Цифра. Аналогично "[0-9]"
"\s"	Любой пробельный символ. Аналогично "[\t\n\r\f\v]"
"\S"	Любой непробельный символ. Аналогично "[^\t\n\r\f\v]"
"\w"	Любая цифра или буква (зависит от флага <code>LOCALE</code>)
"\W"	Любой символ, не являющийся цифрой или буквой (зависит от флага <code>LOCALE</code>)

Флаги, используемые с регулярными выражениями

- `"(?i)", re.I, re.IGNORECASE`

Сопоставление проводится без учета регистра букв.

- `"(?L)", re.L, re.LOCALE`

Влияет на определение буквы в `"\w"`, `"\W"`, `"\b"`, `"\B"` в зависимости от текущей культурной среды (locale).

- `"(?m)", re.M, re.MULTILINE`

Если этот флаг задан, `"^"` и `"$"` соответствуют началу и концу любой строки.

- `"(?s)", re.S, re.DOTALL`

Если задан, `"."` соответствует также и символу конца строки `"\n"`.

- `"(?x)", re.X, re.VERBOSE`

- Если задан, пробельные символы, не экранированные в шаблоне обратной косой чертой, являются незначащими, а все, что расположено после символа `"#"`, -- комментарии. Позволяет записывать регулярное выражение в несколько строк для улучшения его читаемости и записи комментариев.

- `"(?u)", re.U, re.UNICODE`

В шаблоне и в строке использован Unicode.

Методы объекта-шаблона

- В результате успешной компиляции шаблона функцией `re.compile()` получается шаблон-объект (он именуется `SRE_Pattern`).
- `match(s)`

Сопоставляет строку `s` с шаблоном, возвращая в случае удачного сопоставления объект с результатом сравнения (объект `SRE_Match`). В случае неудачи возвращает `None`. Сопоставление начинается от начала строки.

- `search(s)`

Аналогичен `match(s)`, но ищет подходящую подстроку по всей строке `s`.

Методы объекта-шаблона

- `split(s[, maxsplit=0])`

Разбивает строку на подстроки, разделенные подстроками, заданными шаблоном. Если в шаблоне выделены группы, они попадут в результирующий список, перемежаясь с подстроками между разделителями. Если указан `maxsplit`, будет произведено не более `maxsplit` разбиений.

- `findall(s)`

Ищет все неперекрывающиеся подстроки `s`, удовлетворяющие шаблону.

- `finditer(s)`

Возвращает итератор по объектам с результатами сравнения для всех неперекрывающихся подстрок, удовлетворяющих шаблону.

Методы объекта-шаблона

- `sub(repl, s)`

Заменяет в строке `s` все (или только `count`, если он задан) вхождения неперекрывающихся подстрок, удовлетворяющих шаблону, на строку, заданную с помощью `repl`. В качестве `repl` может выступать строка или функция. Возвращает строку с выполненными заменами. В первом случае строка `repl` подставляется не просто так, а интерпретируется с заменой вхождений "`\номер`" на группу с соответствующим номером и вхождений "`\g<имя>`" на группу с номером или именем имя. В случае, когда `repl` - функция, ей передается объект с результатом каждого успешного сопоставления, а из нее возвращается строка для замены.

- `subn(repl, s)`

Аналогичен `sub()`, но возвращает кортеж из строки с выполненными заменами и числа замен.

Примеры

```
1 match = re.search(r'\d\d\D\d\d', r'Телефон 123-12-12')
2 print(match)
3 print(match[0] if match else 'Not found')
```

<re.Match object; span=(9, 14), match='23-12'>
23-12

```
1 match = re.search(r'\d\d\D\d\d', r'Телефон 1231212')
2 print(match[0] if match else 'Not found')
```

Not found

```
1 match = re.fullmatch(r'\d\d\D\d\d', r'12-12')
2 print('YES' if match else 'NO')
```

YES

Примеры

```
1 match = re.fullmatch(r'\d\d\D\d\d', r'T. 12-12')
2 print('YES' if match else 'NO')
```

NO

```
1 print(re.split(r'\W+', 'Скажите, дядя, ведь не даром...'))
```

['Скажите', 'дядя', 'ведь', 'не', 'даром', '']

```
1 print(re.findall(r'\d\d\.\d\d\.\d{4}',
2                 r'Эта строка написана 24.11.2020, а могла бы и 18.11.2020'))
```

['24.11.2020', '18.11.2020']

Примеры

```
1 for m in re.finditer(r'\d\d\.\d\d\.\d{4}',  
2                       r'Эта строка написана 24.11.2020, а могла бы и 18.11.2020'):  
3     print('Дата', m[0], 'начинается с позиции', m.start())
```

Дата 24.11.2020 начинается с позиции 20

Дата 18.11.2020 начинается с позиции 45

```
1 print(re.sub(r'\d\d\.\d\d\.\d{4}',  
2             r'DD.MM.YYYY',  
3             r'Эта строка написана 24.11.2020, а могла бы и 18.11.2020'))
```

Эта строка написана DD.MM.YYYY, а могла бы и DD.MM.YYYY

Пример

```
1 import re
2 delim_re = re.compile(r"[;,]")
3 text = "This,is;example"
4 print(delim_re.split(text))
```

['This', 'is', 'example']

```
1 delim_re = re.compile(r"([;,])")
2 print(delim_re.split(text))
```

['This', ',', 'is', ';', 'example']

Примеры

- `r"\b\w+\b"`

Соответствует слову из букв и знаков подчеркивания.

- `r"[+-]?\d+"`

Соответствует целому числу. Возможно, со знаком.

- `r"\([+-]?\d+\)"`

Число, стоящее в скобках. Скобки используются в самих регулярных выражениях, поэтому они экранируются `"\"`.

Примеры

- `r" [a-zA-Z] { 2 } "`

Соответствует строке из двух букв "a", "b" или "c".
Например, "Ac", "CC", "bc".

- `r" aa | bb | cc | AA | BB | CC "`

Строка из двух одинаковых букв.

- `r" ([a-zA-Z]) \ 1 "`

Строка из двух одинаковых букв, но шаблон задан с использованием групп

- `r" aa | bb "`

Соответствует "aa" или "bb"

Примеры

- `r"a(a|b)b"`

Соответствует "aab" или "abb"

- `r"^(?:\d{8}|\d{4}):\s*(.*)$"`

Соответствует строке, которая начинается с набора из восьми или четырех цифр и двоеточия. Все, что идет после двоеточия и после следующих за ним пробелов, выделяется в группу с номером 1, тогда как набор цифр в группу не выделен.

- `r"(\w+)=.*\b\1\b"`

Слова слева и справа от знака равенства присутствуют. Операнд "\1" соответствует группе с номером 1, выделенной с помощью скобок.

Примеры

- `r" (?P<var>\w+) = . * \b (?P=var) \b "`

То же самое, но теперь используется именованная группа `var`.

- `r"\bregular (?=\s+expression) "`

Соответствует слову "regular" только в том случае, если за ним после пробелов следует "expression"

- `r"(?<=regular)expression"`

Соответствует слову "expression", перед которым стоит "regular" и один пробел.

Он-лайн отладка регулярных выражений

regular expressions ¹⁰¹

regex101.com

</> Python



REGULAR EXPRESSION v3 ▾

`r" \d{5}"`

TEST STRING

333

4444

55555

EXPLANATION

- ▼ `" \d{5} " g`
 - ▼ `\d{5}` matches a digit (equal to `[0-9]`)
`{5}` **Quantifier** — Matches exactly 5 times
 - ▼ **Global pattern flags**
`g` **modifier**: global. All matches (don't return after first match)

MATCH INFORMATION

Match 1

Full match 9-14 55555

Он-лайн отладка регулярных выражений

DebuggexBeta

www.debuggex.com

Untitled Regex No description



Result: **Does not match** starting at the black triangle slider

1	test
2	34234324

START

POSITION

Python

[View Cheatsheet](#)

1 `\d{5}`

Flags

- ☐ Ignore case (i)
- ☐ ^ and \$ match lines (m)
- ☐ . matches newline (s)
- ☐ Ignore spaces (x)

Задачи для самостоятельного решения

- Найдите все натуральные числа (возможно, окружённые буквами);
- Найдите все «слова», написанные капсом (то есть строго заглавными), возможно внутри настоящих слов (aaaБББvvv);
- Найдите слова, в которых есть русская буква, а когда-нибудь 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);

Задачи для самостоятельного решения

- Найдите все натуральные числа, не находящиеся внутри или 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.