ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Программирование на языке Python. Уровень 1.

Регулярные выражения.

Понятие о регулярном выражении

- **Регулярные выражения (regular expressions)** описывают множество строк, используя специальный язык.
- Регулярные выражения предназначены для выполнения в строке сложного поиска или замены.
- Используются для задания формата и проверки ввода данных по шаблону.
- Шаблон – строка, в которой задано регулярное выражение.
- Для работы с регулярными выражениями в Python используется модуль **re**.

```
import re
```

- **Схема работы с регулярными выражениями:**
  - ✓ создать шаблон с регулярным выражением;
  - ✓ сопоставить шаблон с текстовой строкой;
  - ✓ проанализировать результат.

Работа с регулярными выражениями

- Функция `re.compile()` компилирует шаблон в специальный Regex-объект (объект-шаблон), который имеет несколько методов для работы с шаблонами.
- Формат функции `re.compile()`:

```
<Шаблон> = re.compile (<Регулярное выражение>[, <Модификатор>])
```

- В параметре <модификатор> могут быть указаны следующие флаги (или их комбинация через оператор |):
  - **I** или **IGNORECASE** — поиск без учета регистра:

```
>>> import re
>>> p = re.compile(r"^[а-яё]+$", re.I | re.U)
>>> print("Найдено" if p.search("АБВГДЕЁ") else "Нет")
Найдено
>>> p = re.compile(r"^[а-яё]+$", re.U)
>>> print("Найдено" if p.search("АБВГДЕЁ") else "Нет")
Нет
```

Работа с регулярными выражениями

- **M** или **MULTILINE** — поиск в строке, состоящей из нескольких подстрок, разделенных символом новой строки ("\n");
- **S** или **DOTALL** — метасимвол «точка» по умолчанию соответствует любому символу, кроме символа перевода строки (\n). Символу перевода строки метасимвол «точка» будет соответствовать в присутствии дополнительного модификатора.
- **X** или **VERBOSE** — если флаг указан, то пробелы и символы перевода строки будут проигнорированы.
- **A** или **ASCII** — классы \w, \W, \b, \B, \d, \D, \s и \S будут соответствовать символам в кодировке ASCII (по умолчанию указанные классы соответствуют Unicode-символам);
- **L** или **LOCALE** — учитываются настройки текущей локали. Начиная с Python 3.6, могут быть использованы только в том случае, когда регулярное выражение задается в виде значения типов bytes или bytearray.

Работа с регулярными выражениями

- Перед всеми строками, содержащими регулярные выражения, указывается модификатор **r**.
- Таким образом используются неформатированные строки.
- Если модификатор не указывать, то все слэши необходимо экранировать. Например, строку:

```
p = re.compile(r"A\w+$")
```

нужно было бы записать так:

```
p = re.compile("A\\w+$")
```

- Внутри регулярного выражения символы **.**, A, $,*,+,?, {, [,], \, | имеют специальное значение. Если эти символы должны трактоваться как есть, их следует экранировать с помощью слэша.
- Некоторые специальные символы теряют свое особое значение, если их разместить внутри квадратных скобок — в этом случае экранировать их не нужно.
- Например, метасимвол «точка» по умолчанию соответствует любому символу, кроме символа перевода строки. Если необходимо найти именно точку, то перед точкой нужно указать символ \ или разместить точку внутри квадратных скобок: [**.**].

Синтаксис регулярных выражений

- В квадратных скобках [ ] указываются символы, которые могут встречаться на данной позиции в строке.
- Разрешается записывать символы подряд или указать диапазон через дефис:
  - [09] — соответствует числу 0 или 9;
  - [0-9] — соответствует любому числу от 0 до 9;
  - [абв] — соответствует буквам «а», «б» и «в»;
  - [а-г] — соответствует буквам «а», «б», «в» и «г»;
  - [а-яё] — соответствует любой букве от «а» до «я»;
  - [АБВ] — соответствует буквам «А», «Б» и «В»;
  - [А-ЯЁ] — соответствует любой букве от «А» до «Я»;
  - [0-9а-яА-ЯёЁa-zA-Z] — любая цифра и любая буква.

Примечание.

Буква «ё» не входит в диапазон [а-я], а буква «Ё» — в диапазон [А-Я].

Синтаксис регулярных выражений

- Специальные символы в регулярных выражениях:

`.`	(точка) любой символ, кроме символа конца строки.
`[]`	любой символ из тех, что заключены в скобки; символы можно перечислить как напрямую `[abcdef]`, так и через дефис `[a-f]`
`^`	(крышечка) начало строки
`$`	конец строки
`*`	предыдущий символ или фрагмент повторяется 0 или более раз
`+`	предыдущий символ или фрагмент повторяется 1 или более раз
`?`	предыдущий символ или фрагмент может как присутствовать, так и отсутствовать
`{m}`	предыдущий символ или фрагмент повторяется ровно m раз
`{m,n}`	предыдущий символ или фрагмент повторяется от m до n раз
`\`	экранирует специальный символ (что позволяет использовать в регулярном выражении обычные точки, звездочки и проч.), или обозначает специальную последовательность
`()`	выделяет фрагмент (группу) в регулярном выражении
`(A|B)`	фрагмент (группа), соответствующий либо регулярному выражению `A`, либо регулярному выражению `B`
`\b`	пустая строка, обозначающая начало или конец слова
`\d`	любая цифра
`\s`	любой символ помежутка (пробел, знак табуляции, символ конца строки, символ возврата каретки)
`\S`	любой символ, *кроме* символа промежутка
`\w`	буква, цифра или знак подчеркивания

Синтаксис регулярных выражений

- Привязки к началу и концу строки осуществляются с помощью метасимволов:

  ^ — привязка к началу строки или подстроки. Зависит от флагов **M** (или **MULTILINE**) и (**S** или **DOTALL**);

  $ — привязка к концу строки или подстроки. Зависит от флагов **M** (или **MULTILINE**) и (**S** или **DOTALL**);

  \A — привязка к началу строки (не зависит от модификатора);

  \Z — привязка к концу строки (не зависит от модификатора).

  \b — привязка к началу слова (началом слова считается пробел или любой символ, не являющийся буквой, цифрой или знаком подчеркивания);

  \B — привязка к позиции, не являющейся началом слова.

- Если в параметре **<модификатор>** указан флаг **M** (или **MULTILINE**), то поиск производится в строке, состоящей из нескольких подстрок, разделенных символом новой строки.
- В этом случае символ **^** соответствует привязке к началу каждой подстроки, а символ **$** — позиции перед символом перевода строки.

Методы объекта-шаблона

- В результате успешной компиляции шаблона функцией `re.compile()` получается объект-шаблон, который имеет следующие методы:
- `match(s)`

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

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

ЦЕНТР
ДОПОЛНИТЕЛЬНОГО
ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

do.bmstu.ru