Строки. Регулярные выражения. Часть №2

Регулярные выражения — это мощный инструмент для поиска, извлечения и работы с текстовыми данными на основе заданных шаблонов.

Модуль re в Python

Модуль re — стандартный модуль Python для работы с регулярными выражениями.

Основные функции:

```
re.findall(pattern, string)
```

- ищет все совпадения шаблона в строке и возвращает список

```
re.fullmatch(pattern, string)
```

- проверяет, совпадает ли шаблон с строкой

```
re.search(pattern, string)
```

- ищет первое совпадение шаблона в строке

```
re.sub(pattern, replacement, string)
```

- заменяет все совпадения шаблона на указанную подстроку

Регулярные выражения позволяют находить и извлекать числа, операторы, последовательности чисел и другие элементы арифметических выражений.

Используя шаблоны, можно анализировать структуры выражений, проверять их корректность и разбивать на токены.

Пример №1. Поиск чисел в строке

```
import re
s = "123 + 456 - 789"
matches = re.findall(r'\d+', s)
print(matches)
```

Вывод >>> ['123', '456', '789']

Объяснение:

- \d+ — соответствует одному или более символам (цифрам):
  - \d — это обозначение для цифры (от 0 до 9).
  - + — означает "один или более" (то есть выражение будет находить последовательности из одной или более цифр).

Пример №2. Поиск целых чисел с учетом знака

```
import re
s = "-123 + 456 - -789"
matches = re.findall(r'-?\d+', s)
print(matches)
```

Вывод >>> ['-123', '456', '-789']

Объяснение:

- -? — означает, что знак минус (-) может быть необязательным:
  - - — соответствует символу минус.
  - ? — означает, что предыдущий символ (в данном случае минус) может встречаться 0 или 1 раз. Это позволяет учитывать как положительные, так и отрицательные числа.
- \d+ — соответствует одному или более цифровым символам (число).

Пример №3. Поиск десятичных дробных чисел

```
import re
s = "3.14 + 2.71 - 42.0"
matches = re.findall(r'-?\d+(?:\.\d+)?', s)
print(matches)
```

Вывод >>> ['3.14', '2.71', '42.0']

Объяснение:

- -? — необязательный минус (для отрицательных чисел).
- \d+ — одно или более чисел (целая часть).
- (?:\.\d+)? — необязательная дробная часть (с точкой и цифрами).

Пример №4. Токенизация арифметического выражения

Токенизация — это разбиение выражения на числа, операторы и скобки.

```
import re
s = "3 + (5 * (2 - 8)) / 4"
tokens = re.findall(r'\d+|[+\-*/()]', s)
print(tokens)
```

Вывод >>> ['3', '+', '(', '5', '*', '(', '2', '-', '8', ')', ')', '/', '4']

Объяснение:

- **\d+:** Найдет числа.
- **[+\-*/()]:** Найдет операторы и скобки.
- **|:** Логическое ИЛИ, объединяет два шаблона.

Пример №5. Проверка корректности арифметических выражений

```
import re
s = "3 + (5 * (2 - 8)) / 4"
pattern = r'^\s*[\d+\-*/().\s]+\s*$'
is_valid = re.match(pattern, s) is not None
print(is_valid)
```

Вывод >>> True

Объяснение:

- **\s*:** Игнорирует пробелы.
- **[\d+\-*/().]+:** Проверяет, что строка содержит только допустимые символы.