Python XV:
Регулярные выражения

Регулярные выражения (Regular Expressions или Regex) - это инструмент для работы с текстом, который позволяет искать и извлекать информацию из строк, основываясь на шаблонах.

В Python для работы с регулярными выражениями используется встроенный модуль re. Он предоставляет несколько функций, которые позволяют выполнить различные операции с регулярными выражениями.

В регулярных выражениях есть несколько специальных символов, которые используются для определенных задач. Некоторые из них:

- **.** - соответствует любому символу, кроме символа новой строки (\n)
- **^** - соответствует началу строки
- **$** - соответствует концу строки
- ***** - соответствует 0 или более повторениям предыдущего символа
- **+** - соответствует 1 или более повторениям предыдущего символа
- **?** - соответствует 0 или 1 повторению предыдущего символа
- **{m}** - соответствует ровно m повторениям предыдущего символа
- **{m,}** - соответствует m или более повторениям предыдущего символа
- **{m,n}** - соответствует от m до n повторений предыдущего символа
- **[]** - соответствует любому из перечисленных символов в квадратных скобках
- **()** - группирует символы в подвыражение, которое может быть использовано с другими специальными символами
- **|** - соответствует любому из двух подвыражений, разделенных вертикальной чертой

Кроме того, есть также множество предопределенных классов символов, которые могут использоваться в регулярных выражениях:

- **\d** - соответствует любой цифре (то же, что и [0-9])
- **\D** - соответствует любому символу, не являющемуся цифрой (то же, что и [^0-9])
- **\w** - соответствует любой букве или цифре (то же, что и [a-zA-Z0-9_])
- **\W** - соответствует любому символу, не являющемуся буквой или цифрой (то же, что и [^a-zA-Z0-9_])
- **\s** - соответствует любому символу пробела (то же, что и [\t\n\r\f\v])
- **\S** - соответствует любому символу, не являющемуся символом пробела (то же, что и [^\t\n\r\f\v])

Эти специальные символы позволяют задавать более сложные шаблоны для поиска и замены в строках.

Допустим, у нас есть строка с именем и фамилией, разделенными пробелом: "John Smith". Мы хотим извлечь имя и фамилию, используя регулярные выражения.

Для начала мы должны импортировать модуль регулярных выражений:

```
import re
```

Затем мы можем создать шаблон, используя функцию compile:

```
pattern = re.compile(r'(\w+) (\w+)')
```

Этот шаблон ищет два слова, разделенных пробелом, и сохраняет их в группы с помощью скобок.

Мы можем применить шаблон к строке с помощью функции search:

```
result = pattern.search('John Smith')
```

Эта функция возвращает объект Match с информацией о найденном совпадении. Мы можем использовать этот объект, чтобы получить имя и фамилию:

```
first_name = result.group(1)
last_name = result.group(2)
```

Здесь мы используем метод group, чтобы получить значения, соответствующие группам в нашем шаблоне.

Таким образом, полный код будет выглядеть следующим образом:

```
import re

pattern = re.compile(r'(\w+) (\w+)')
result = pattern.search('John Smith')
first_name = result.group(1)
last_name = result.group(2)

print(first_name) # выводит "John"
print(last_name) # выводит "Smith"
```

Некоторые из наиболее часто используемых методов этого модуля:

- **re.search(pattern, string)**: ищет первое совпадение регулярного выражения **pattern** в строке **string** и возвращает объект **match** (если совпадение найдено) или **None** (если совпадение не найдено).

- **re.match(pattern, string)**: ищет совпадение регулярного выражения **pattern** с началом строки **string** и возвращает объект **match** (если совпадение найдено) или **None** (если совпадение не найдено).

- **re.findall(pattern, string)**: ищет все совпадения регулярного выражения **pattern** в строке **string** и возвращает их в виде списка строк.

- **re.sub(pattern, repl, string)**: заменяет все совпадения регулярного выражения **pattern** в строке **string** на строку **repl** и возвращает результат.