УРОК 34. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

Регулярные выражения являются мощным инструментом для работы с текстовыми данными. Они позволяют искать и извлекать информацию из строк на основе заданных шаблонов. Мотивация использования регулярных выражений включает поиск, замену, валидацию и извлечение данных из текста. Такие задачи, как проверка правильности формата email-адреса, поиск всех ссылок в веб-странице или извлечение номеров телефонов из текстового документа, могут быть эффективно решены с помощью регулярных выражений.

СИНТАКСИС РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ

Синтаксис регулярных выражений опирается на специальные символы и конструкции. Например, символ . обозначает любой символ, символ * указывает на ноль или более повторений предыдущего символа или группы символов, а символ + означает одно или более повторений. Кроме того, существуют специальные конструкции, такие как [...], которые задают классы символов, и (...), которые обозначают группы символов.

Регулярные выражения предлагают различные специальные конструкции для обозначения групп символов. Например, \d соответствует любой цифре, \w соответствует любой букве или цифре, а \s соответствует любому пробельному символу. Также можно использовать квантификаторы, такие как ? (ноль или одно повторение), {n} (ровно n повторений) и {m, n} (от m до n повторений), чтобы указать количество повторений символа или группы символов.

ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

Попробуйте “расшифровать” регулярное выражение для поиска дат по учебнику истории:

\d{4}|[IVX]+ век

МОДУЛЬ RE

В Python для работы с регулярными выражениями используется модуль re. Он предоставляет различные методы для работы с регулярными выражениями. Некоторые из наиболее популярных методов включают re.match(), re.search(), re.findall(), re.split(), re.sub() и re.compile(). Метод re.match() ищет совпадение регулярного выражения только в начале строки, re.search() ищет совпадение в любом месте строки, re.findall() возвращает список всех совпадений, re.split() разделяет строку на список подстрок по заданному шаблону, re.sub() выполняет замену совпадений на заданную строку или функцию, а re.compile() компилирует регулярное выражение для повторного использования.

```python
import re

pattern = r"\b\w{3}\b"
text = "Hello, how are you?"

result = re.findall(pattern, text)
print(result) # ['how', 'are']
```

Регулярные выражения позволяют группировать результаты с помощью круглых скобок (...). Это позволяет извлекать отдельные части совпадений или использовать их для замены. Обратные ссылки позволяют ссылаться на группы символов, найденных ранее в выражении. Например, \1 ссылается на первую группу символов, \2 - на вторую и так далее.

```python
import re

pattern = r"(\b\w+)\s\1"
text = "hello hello world world"

result = re.findall(pattern, text)
```


```
print(result) # ['hello', 'world']

replaced_text = re.sub(pattern, r"\1", text)
print(replaced_text) # hello world
```

ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

Объясните, что происходит в фрагменте кода, приведенном ниже:

```python
import re

text = "Python is an amazing programming language."
pattern = "amazing"

match = re.search(pattern, text)
if match:
  print("Совпадение найдено:", match.group())
else:
  print("Совпадение не найдено")
```

ОПЕРАТОР “МОРЖ”

Оператор "морж" (:=) (Python >= 3.8) в сочетании с модулем re может использоваться для более удобного применения регулярных выражений. Он позволяет объединить поиск и присваивание результата поиска в одном выражении.

```python
import re

text = "Hello, how are you?"

if match := re.search(r"\b\w{3}\b", text):
  print("Match found:", match.group(0))
else:
  print("No match")
```

Оператор "морж" позволяет избежать повторного вызова метода re.search() и сохранить результат поиска в переменную match. Если совпадение найдено, выводится сообщение "Match found" вместе с совпавшим текстом, в противном случае выводится сообщение "No match".

Таким образом, регулярные выражения предоставляют мощный инструмент для работы с текстовыми данными в Python. Они позволяют искать, извлекать и изменять информацию на основе заданных шаблонов. С помощью модуля re и синтаксиса регулярных выражений можно решать широкий спектр задач, связанных с обработкой текста.

ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

Перепишите строку кода, используя оператор “морж”:

```python
result = [func(x), func(x)**2, func(x)**3]
```

ПРАКТИЧЕСКАЯ РАБОТА

1. Написать парсер для:
    - телефонных номеров (обработать как можно больше форматов)
    - для email
2. Напишите функцию validate_password(password), которая принимает строку с паролем и проверяет его на соответствие следующим условиям:
    - Длина пароля должна быть не менее 8 символов
    - Пароль должен содержать хотя бы одну заглавную букву, одну строчную букву и одну цифру
    - Пароль может содержать только следующие специальные символы: !@#$%^&*()

ПОЛЕЗНЫЕ МАТЕРИАЛЫ

1. Python RegEx: практическое применение регулярок
2. Регулярные выражения в Python от простого к сложному.