Inatel

C005 – Linguagens de Programação e Compiladores

Prof. Me. Marcelo Vinícius Cysneiros Aragão

Capítulo 2 – Análise Léxica – Parte 1

Prática de REGEX com Python

Como vimos, uma REGEX (Expressão Regular) é uma sequência de metacaracteres e regras que nos permite criar reconhecedores para identificar infinitos tipos de padrões de Strings.

No Python, por exemplo, podemos utilizar de uma biblioteca chamada "re" para se trabalhar com REGEX.

1. Importando o módulo "re" no Python:

Ex:

```
# Importando a biblioteca de REGEX no Python
import re

# Texto para realizarmos o teste
txt = "Os melhores engenheiros são do Brasil"

# Este teste verifica se o texto comeca com "Os" e termina com
"Brasil"
x = re.search("^Os.*Brasil$", txt)

# Verificando o teste
if x:
  print("PERFEITO! Tivemos um match no Teste :)")
else:
  print("Não tivemos um match no Teste :(")
```

- O match significa que o padrão que você definiu no REGEX foi encontrado dentro do texto que passou como parâmetro;
- Observe que para montarmos REGEXes, usamos de toda fundamentação que já vimos anteriormente.

Capítulo 2 – Análise Léxica – Parte 1

2. Funções de REGEX em Python

O módulo "re" do Python nos oferece uma série de funções importantes para se trabalhar com REGEX, são elas:

- findall() -> retorna uma lista contendo todos os matches encontrados;
- search() -> retorna um **"Match Object"** se existe ao menos um padrão encontrado em qualquer lugar de um texto;
- split() -> retorna uma lista onde um texto foi quebrado em cada match encontrado;
- sub() -> altera o conteúdo de um ou vários padrões identificados dentro de um texto.

3. A função findall()

Esta função retorna uma lista contendo todos os padrões identificados dentro de um texto fornecido.

Ex:

```
import re


# Texto para realizarmos o teste
txt = "oi! teu pai tem boi? Foi o que pensei."

# Este teste retorna uma lista contendo cada ocorrência de
"oi" dentro de um texto.
x = re.findall("oi", txt)

# Matches Encontrados
print('Matches:', x)
# Quantidade de Matches
print('Quantidade:', len(x))
```

- Observe o uso da função len() para pegar o tamanho da lista retornada.

4. A função search()

Esta função retorna um ***"Match Object"*** se um match for encontrado. Ele é uma espécie de variável que possui informações sobre o resultado da busca.

- Se existe mais de um match, apenas o primeiro será retornado;
- Se nenhum match é encontrado, o valor **"None"** é retornado.

Ex:

```
import re

# Texto para realizarmos o teste
txt = "O rato roeu a roupa do rei de roma. Que rato danado!"

# Este teste retorna um "Match Object" se a String rato for
encontrada
x = re.search("rato", txt)

# Se o "Match Object" nao for nulo (None)
if x:
    print("PERFEITO! Tivemos um match no Teste :)")
    print("O primeiro rato foi encontrado na posição:",
x.start())
```

Um **"Match Object"** oferece funções que podem ser usadas para se buscar mais detalhes do resultado de uma busca. Duas muito importantes são:

- .span() -> retorna uma tupla contendo a posição inicial e final de um match;
- .group() -> retorna a String em que um match foi encontrado.

4.1. .span()

Ex:

```
import re

# Texto para realizarmos o teste
txt = "O rato roeu a roupa do rei de roma. Que rato danado!"
# Este teste retorna um "Match Object" se uma palavra começar
com r e terminar com o
x = re.search("r[a-z]+o", txt)

# Retornando a posicao inicial e final do match
print(x.span())
```

Ex2:

```
import re

# Texto para realizarmos o teste
txt = "O rato roeu a roupa do rei de roma. Que rato danado!"

# Retornando a posicao inicial e final de todos os matches
for match in re.finditer("r[a-z]+o", txt):
    print(match.span())
```

4.2. .group()

Ex:

```
import re

# Texto para realizarmos o teste
txt = "O rato roeu a roupa do rei de roma. Que rato danado!"

# Este teste retorna um "Match Object" se uma palavra possuir
a no meio
x = re.search("[a-z]*a[a-z]*", txt)

print(x.group())
```

Ex2:

```
import re

# Texto para realizarmos o teste
txt = "O rato roeu a roupa do rei de roma. Que rato danado!"

# Retornando as Strings de todos os matches
for match in re.finditer("[a-z]*a[a-z]*", txt):
    print(match.group())
```

5. A função split()

Retorna uma lista onde a String pode ser quebrada em cada match.

```
import re
# Texto para realizarmos o teste
txt = "Eu gosto de sorvete de chocolate; meu pai, de creme;
meu irmão, de morango"

# Quebra (splits) o texto a cada ; encontrado
x = re.split(";", txt)
print(x)

# Quebra (splits) o texto a cada espaço encontrado
y = re.split("\s", txt)
print(y)
```

6. A função sub()

Substitui os matches encontrados por um texto da sua escolha.

```
import re
# Texto para realizarmos o teste
txt = "Quando chover, busque pelo arco-íris. Quando escurecer,
busque pelas estrelas."

# Este teste substitui cada palavra "Quando" por "Sempre que"
x = re.sub("Quando", "Sempre que", txt)
print(x)
```

Inatel

C005 – Linguagens de Programação e Compiladores

Prof. Me. Marcelo Vinícius Cysneiros Aragão

Capítulo 2 – Análise Léxica – Parte 1

EXERCÍCIOS PROPOSTOS

Agora que já possui uma noção de como utilizar de funções REGEX diretamente no Python, utilize do Interpretador online programiz e crie códigos que respondam às perguntas abaixo:

https://www.programiz.com/python-programming/online-compiler/

Texto Base:

Viver é acalentar sonhos e esperanças, fazendo da fé a nossa inspiração maior. É buscar nas pequenas coisas, um grande motivo para ser feliz

1. Código que retorne se um match pode ser encontrado ao se procurar pela palavra “sonhos”;
2. Código que retorne uma lista apenas das palavras terminadas com “as”; (Atenção, a classe de caracteres [a-zA-Z] naturalmente não aceita caracteres da língua portuguesa, por isso, você deve adicioná-los manualmente dentro da classe. Ex: [a-zA-Zç]
3. Código que em um único REGEX, troque as palavras “maior” e “grande” pela palavra “surreal” e mostre a nova frase após as trocas;
4. Código que retorna a posição inicial e final da palavra “inspiração” dentro do texto;
5. Código que retorna uma lista apenas com as palavras que possuam 9 letras ou mais;
6. Código que retorna uma lista de Strings que deverão ser quebradas toda vez que for encontrado no texto o caracter . (ponto) ou , (vírgula);
7. Código que retorne todas as palavras que possuam a letra “é” (com acento) seja em maiúscula ou minúscula;
8. Código que responda quantas palavras o texto base possui.

Para mais dicas e oportunidades de prática, acesse:
https://www.w3schools.com/python/python_regex.asp