RASPADOR

Mini-biblioteca para extração de dados em documentos semi-estruturados

SOBRE MIM

- Desenvolvedor desde 2003
- Conheci Python em 2009
- Trabalho na NCR Corporation
- Na NCR, Python não é a linguagem primária

```
from raspador import history
```

- Foi utilizado para extração de dados de Espelhos MFD
- Virou código de base do projeto

OUTRO PARSER?

- lxml (XPath, cssselectors)
- html5lib (html parser)
- BeautifulSoup (tree parser api)
- PyQuery (cssselectors)
- Scrapely (magia negra)
- Scrapy (crawler: request, responsing)
- pyparsing (grammar)
- NLTK (grammar)
- Plain Python + regex

O QUE?

- Extrair dados de arquivos texto que não foram projetados para isso.

```
CNPJ: 40.100.280/0001-25       IE:   600020060001

IM:   36/3372
 18/01/2013 11:07:04        CCF:002902 COO:007490
       CUPOM FISCAL
ITEM CÓDIGO DESCRIÇÃO QTD.UN.VL UNIT R$ ST VL ITEM R$
 001 1 prd1                            1UN I1 1,00€
 002 2 prd2      Nincid                1UN N1 2,00€
 003 9999999999991 PIZZAS              1UN I1 14,33€
Subtotal  R$                             17,33
ACRÉSCIMO                                +0,30€
  TOTAL   R$        17,63
Dinheiro                                 17,63
--------------------------------------------
MD5: A3BBE73BD09B18ECE607A50F92868A4E
02B 131B4 35A4E F59000 B6 59504C 72A1E 0669F 027
ECF-IF    VERSÃO:01.01.00     ECF:001    Lj:
BBBBBBBBBBAABFCDEI           18/01/2013 11:07:06
FAB:XX00000000000207053                   BR
```

```
{
  'COO': 7490,
  'CCF': 2902,
  'Total': 17.63,
  'Acrescimo': 0.30,
  'Cancelado': False,
  'Cancelamento': False,
  'DataDeEmissao': datetime(2013, 01, 18, 11, 7, 4),
  'NumeroDeSerie': 'DR0510BR000000207153',
  'NumeroDoEcf': 1,
  'Itens': [
    {
```

PROBLEMA

- Extrair dados em documentos de texto
  - Texto sem marcação
  - Arquivos grandes
- Pequenas variações entre arquivos
- Precisão na extração dos dados

OPÇÕES?

- ~~lxml~~ (XPath, cssselectors)
- ~~html5lib~~ (html parser)
- ~~BeautifulSoup~~ (tree parser api)
- ~~PyQuery~~ (cssselectors)
- ~~Scrapely~~ (magia negra)
- ~~Scrapy~~ (crawler: request, responsing)
- pyparsing (grammar)
- NLTK (grammar)
- **Plain Python + regex**

PLAIN PYTHON + REGEX

- Fácil de escrever
- Difícil de manter

> *Write only code*

O que faz?

```
res = []
for linha in entrada.splitlines():
    if not linha:
        continue
    item = {}
    for parte in linha.split():
        k, v = parte.split(':')
        item[k] = v
    res.append(item)
```

Você entende o código, mas não tem significado.

REGULAR EXPRESSIONS

> *Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems. (Jamie Zawinski, 1997)*

```
In []:

# O que isso faz?
regex = "^((([!#$%&'*+\-/=?^_`{|}~\w])|([!#$%&'*+\-/=?^_
`{|}~\w][!#$%&'*+\-/=?^_`{|}~\.\w]{0,}[!#$%&'*+\-/=?^_`{
|}~\w]))[@]\w+([-.]\w+)*\.\w+([-.]\w+)*)$"
```

Email validation - RFC 2821, 2822 compliant

Não exagere

> ***I love regular expressions*** *(Jeff Atwood)*

OBJETIVOS

- Reduzir complexidade
  - Incluir semântica
  - Favorecer composição
- Código testável

pessoa_parser.py

```
from raspador import Parser
from raspador import StringField, IntegerField

class ParserDeInformacoesPessoais(Parser):
    Nome = StringField(r'Nome: (.*)')
    Idade = IntegerField(r'(\d+) anos')
```

A definição de um atributo e o tipo de dado agregam semântica

pessoa.txt

Nome: Guido van Rossum

Guido van Rossum é um programador de computadores dos Países Baixos que é mais conhecido por ser o autor da linguagem de programação Python. Wikipédia

Nascimento: 31 de janeiro de 1956 (57 anos), Países Baixos

Cônjuge: Kim Knapp (desde 2000)

Educação: Universidade de Amsterdã (1982)

Filho: Orlijn Michiel Knapp-van Rossum

Irmão: Just van Rossum

pessoa_utilizacao.py

```
from pessoa_parser import ParserDeInformacoesPessoais

parser = ParserDeInformacoesPessoais()

with open('pessoa.txt') as f:
    for pessoa in parser.parse(f):
        print(pessoa.Nome)
        print(pessoa.Idade)
```

Guido van Rossum

57

```
# parser.parse retorna um generator
with open('pessoa.txt') as f:
    g = parser.parse(f)
    print(type(g))
    print(next(g))
```

```
<type 'generator'>
```

```
Dictionary([('Nome', 'Guido van Rossum'), ('Idade', 57)])
```

RASPADOR.ITEM

```
class Dictionary(OrderedDict):
    """
    Dictionary that exposes keys as properties for
    easy read access.
    """
    def __getattr__(self, name):
        if name in self:
            return self[name]
        raise AttributeError(
            "%s without attr '%s'" %
                (type(self).__name__, name))
```

CAMPOS BUILT-IN

```
from raspador import (
    BaseField, IntegerField,
    StringField, BooleanField,
    FloatField, BRFloatField,
    DateField, DateTimeField)
```

TODO: `BRFloatField`, definir sistema de localização.

BASEFIELD

search

```
>>> s = "02/01/2013 10:21:51           COO:022734"
>>> field = BaseField(search=r'COO:(\d+)')
>>> field.parse_block(s)
'022734'
```

BASEFIELD

input_processor

```
>>> s = "02/01/2013 10:21:51           COO:022734"
>>> def double(value):
...     return int(value) * 2
...
>>> field = BaseField(r'COO:(\d+)',
...                   input_processor=double)
>>> field.parse_block(s)  # 45468 = 2 x 22734
45468
```

BASEFIELD

is_list

```
>>> s = "02/01/2013 10:21:51           COO:022734"
>>> field = BaseField(r'COO:(\d+)', is_list=True)
>>> field.parse_block(s)
['022734']
```

Por convenção, quando o campo retorna uma lista, os valores serão acumulados.

DATEFIELD

format_string

```
>>> s = "2013-01-02T10:21:51           COO:022734"
>>> field = DateField(r'^(\d+-\d+-\d+)',
...                   format_string='%Y-%m-%d')
>>> field.parse_block(s)
datetime.date(2013, 1, 2)
```

PARSER

- Responsável por conduzir a iteração
- Podem ser alinhados

NEM TUDO QUE É TEXTO

... está em texto

pdftotext

Dica:

```
pdftotext -layout <arquivo.pdf>
```

Mantém a estrutura do arquivo gerado próxima com o original.

REGULAR EXPRESSIONS

- Debuggex: visualize suas REs
  https://www.debuggex.com/
- Aurélio
  Expressões regulares, uma abordagem divertida

COMPATIBILIDADE

- CPython 2.6+
  - 2.6: pip install ordereddict
- CPython 3.2+
- PyPy

TESTES

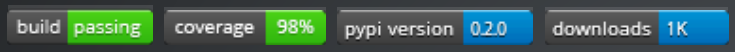
Testes automatizados com **tox**.

```
$ tox
```

Bibliotecas de terceiros para os testes são instaladas automaticamente no ambiente virtual da versão do Python:

```
nose==1.3.0
coverage==3.6
flake8==2.0
```

É NOSSO



https://github.com/fgmacedo/raspador

https://pypi.python.org/pypi/raspador

https://raspador.readthedocs.org/

OBRIGADO!

Fernando Macedo

@fgmacedo

fgmacedo.com

fgmacedo@gmail.com

http://code.fgmacedo.com/talks (Slides)