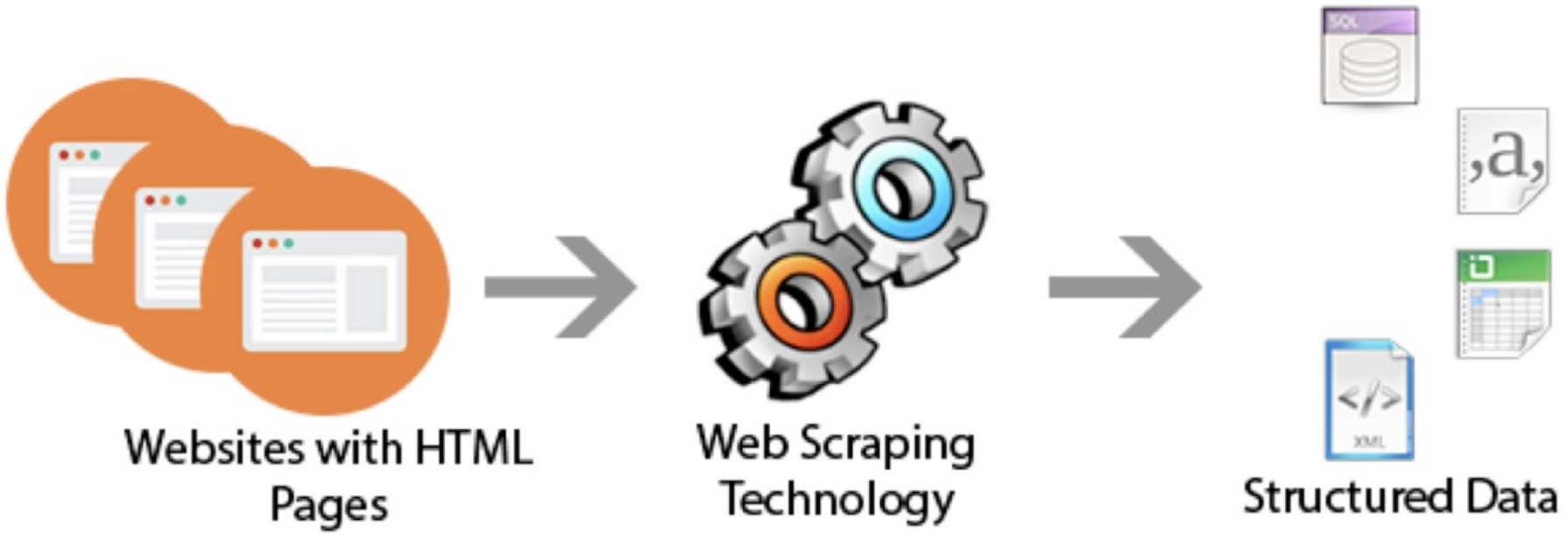
Luiz Gustavo Lourenço Moura
luiz.gustavo@gsuite.iff.edu.br


Websites with HTML Pages
Web Scraping Technology
Structured Data

O Que Web Scraping?

- Técnica para extrair dados de páginas Web
- Concentra-se na transformação de dados não estruturados da Web(Html) em dados estruturados(banco de Dados)

WEB Scraping x Web Crawling

WEB Scraping

- Extrair e reunir conjuntos de dados da web (o que pode ser considerado Big Data em alguns casos), dados esses que são a pedra angular do Big Data Analytics, Machine Learning e Inteligência Artificial

Web Crawling

- o ato de baixar automaticamente os dados de uma página web, extrair os hiperlinks contidos nela e segui-los.
- Os dados baixados são geralmente armazenados em um índice ou banco de dados para facilitar sua busca.
- Indexação, é usado para indexar as informações em uma página web usando bots, também chamados de crawlers. Utilizados pelos principais motores de busca como o Google, Bing e Yahoo.

Robots.txt


```
# Directions for robots.  See this URL:
# http://www.robotstxt.org/wc/norobots.html
# for a description of the file format.

User-agent: HTTrack
User-agent: puf
User-agent: MSIECrawler
Disallow: /

# The Krugle web crawler (though based on Nutch) is OK.
User-agent: Krugle
Allow: /
Disallow: /~guido/orlijn/
Disallow: /webstats/

# No one should be crawling us with Nutch.
User-agent: Nutch
Disallow: /

# Hide old versions of the documentation and various large sets of files.
User-agent: *
Disallow: /~guido/orlijn/
Disallow: /webstats/
```


Python Scrapy: Capture Dados Web de forma rápida e escalável

Professor Luiz Gustavo

O que é Scrapy?

Scrapy é um framework para crawlear web sites e extrair dados estruturados que podem ser usados para uma gama de aplicações úteis (data mining, arquivamento, etc).

- *Scraping*
  - extrair dados do conteúdo da página
- *Crawling*
  - seguir links de uma página a outra

Requests e Beautiful Soup

Beautiful Soup 4.4.0 documentation »

index

Table Of Contents

- Beautiful Soup Documentation
  - Getting help
- Quick Start
- Installing Beautiful Soup
  - Problems after installation
  - Installing a parser
- Making the soup
- Kinds of objects
  - Tag
    - Name
    - Attributes
      - Multi-valued attributes
  - NavigableString
  - BeautifulSoup
  - Comments and other special strings
- Navigating the tree
  - Going down
    - Navigating using tag names
    - .contents and .children
    - .descendants
    - .string
    - .strings and stripped_strings
  - Going up
    - .parent
    - .parents

Beautiful Soup Documentation ¶

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

The examples in this documentation should work the same way in Python 2.7 and Python 3.2.

You might be looking for the documentation for Beautiful Soup 3. If so, you should know that Beautiful Soup 3 is no longer being developed, and that Beautiful Soup 4 is recommended for all new projects. If you want to learn about the differences between Beautiful Soup 3 and Beautiful Soup 4, see Porting code to BS4.

This documentation has been translated into other languages by Beautiful Soup users:

- 这篇文档当然还有中文版.
- このページは日本語で利用できます(外部リンク)
- 이 문서는 한국어 번역도 가능합니다. (외부 링크)

Getting help

If you have questions about Beautiful Soup, or run into problems, send mail to the discussion group. If your problem involves parsing an HTML document, be sure to mention what the diagnose() function says about that document.

Quick Start

Here's an HTML document I'll be using as an example throughout this document. It's part of a story from *Alice in Wonderland*:

```
html_doc = """
```

Instalando o Scrapy

Com o Anaconda Instalado use:

```
conda install -c conda-forge scrapy
```

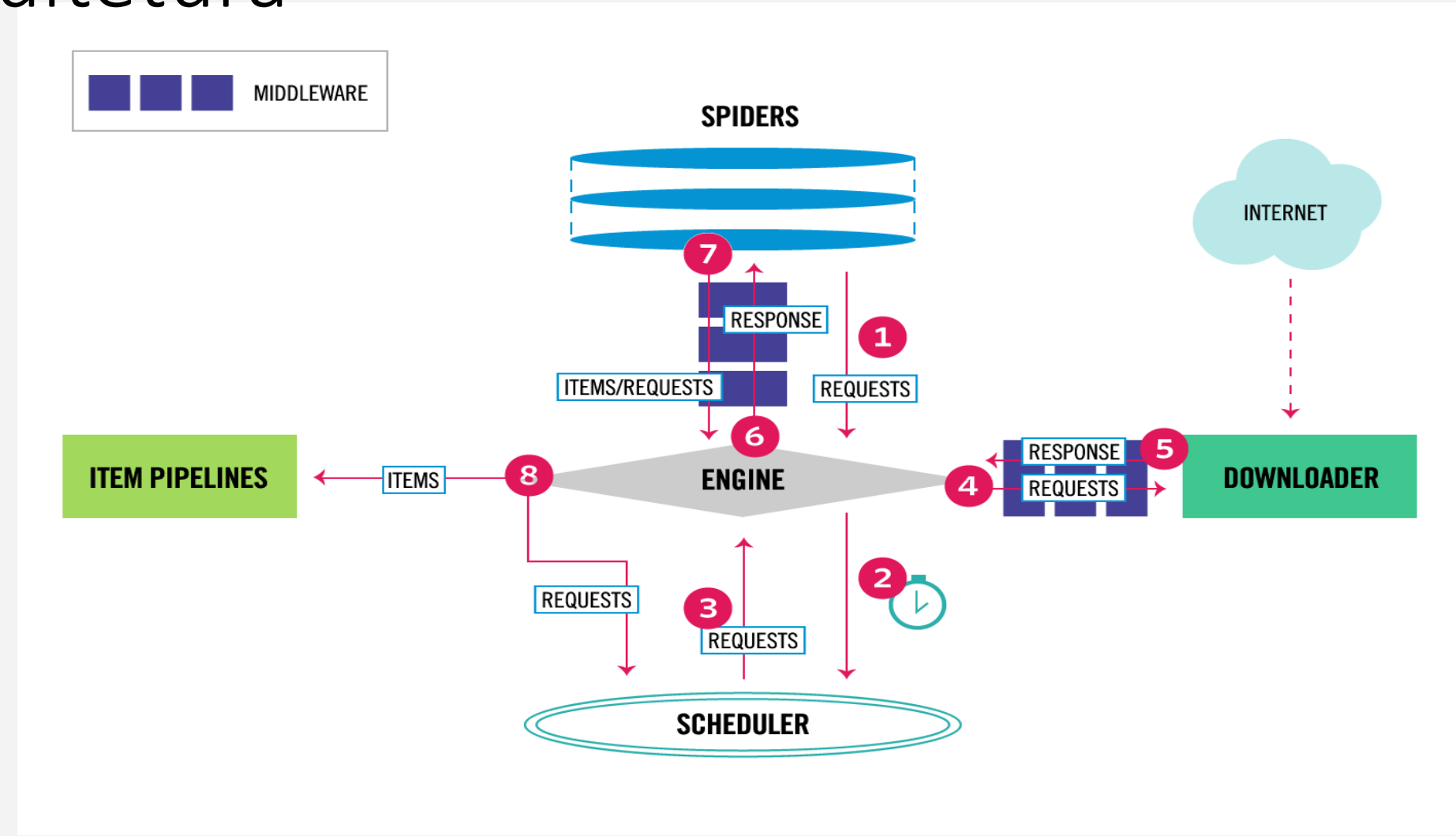
Padrão Python:

```
pip install scrapy
```

Para verificar se o Scrapy está instalado corretamente, rode o comando:

```
scrapy version
```


Arquitetura



Componentes

- **Scrapy Engine**

  É responsavel por controlar o fluxo de dados entre os componentes dos sistema
- **Scheduler**
  - recebe solicitações do engine e as enfileira para alimentá-las mais tarde (também para o mecanismo) quando o mecanismo as solicita.
- **Downloader**
  - Responsável por buscar páginas da web e alimentá-las ao mecanismo que, por sua vez, as alimenta às spiders.
- **Spiders**
  - São classes personalizadas escritas por usuários do Scrapy para analisar respostas e extrair itens (também conhecidos como itens raspados) ou solicitações adicionais a serem seguidas.
- **Item Pipeline**
  - Responsável pelo processamento dos itens depois de terem sido extraídos (ou raspados) pelas aranhas. Tarefas típicas incluem limpeza, validação e persistência (como armazenar o item em um banco de dados). Para mais informações, consulte Item Pipeline.

Spiders

- Conceito central no Scrapy,
- São classes que herdam de scrapy.Spider
  - definindo de alguma maneira as requisições iniciais do crawl e como proceder para tratar os resultados dessas requisições.

```python
import scrapy

class SpiderSimples(scrapy.Spider):
    name = 'meuspider'
    start_urls = ['http://example.com']

    def parse(self, response):
        self.log('Visitei o site: %s' % response.url)
```


Callbacks e próximas requisições

- o método parse()
  - recebe um objeto response que representa uma resposta HTTP;
  - é o que chamamos de um callback.
- Os métodos callbacks no Scrapy são generators (ou retornam uma lista ou iterável) de objetos que podem ser:
  - dados extraídos (dicionários Python ou objetos que herdam de scrapy.Item)
  - requisições a serem feitas a seguir (objetos scrapy.Request)
- O motor do Scrapy itera sobre os objetos resultantes dos callbacks e os encaminha para o pipeline de dados ou para a fila de próximas requisições a serem feitas

Settings

- Oferecem uma maneira de configurar componentes do Scrapy, podendo ser setadas de várias maneiras, tanto via linha de comando, variáveis de ambiente em um arquivo settings.py no caso de você estar usando um projeto Scrapy ou ainda diretamente no spider definindo um atributo de classe custom_settings.

```
class MeuSpider(scrapy.Spider):
    name = 'meuspider'

    custom_settings = {
        'DOWNLOAD_DELAY': 1.5,
    }
```


Criando Projeto

scrapy startproject myproject [project_dir]

```
tutorial/
    scrapy.cfg            # deploy configuration file

    tutorial/             # project's Python module, you'll import your code from here
        __init__.py

        items.py          # project items definition file

        middlewares.py    # project middlewares file

        pipelines.py      # project pipelines file

        settings.py       # project settings file

        spiders/          # a directory where you'll later put your spiders
            __init__.py
```


Criando uma nova Spider

scrapy genspider mydomain mydomain.com

Listando as Spiders

scrapy list

courses ▸ spiders ▸ coursera.py ▸ ...

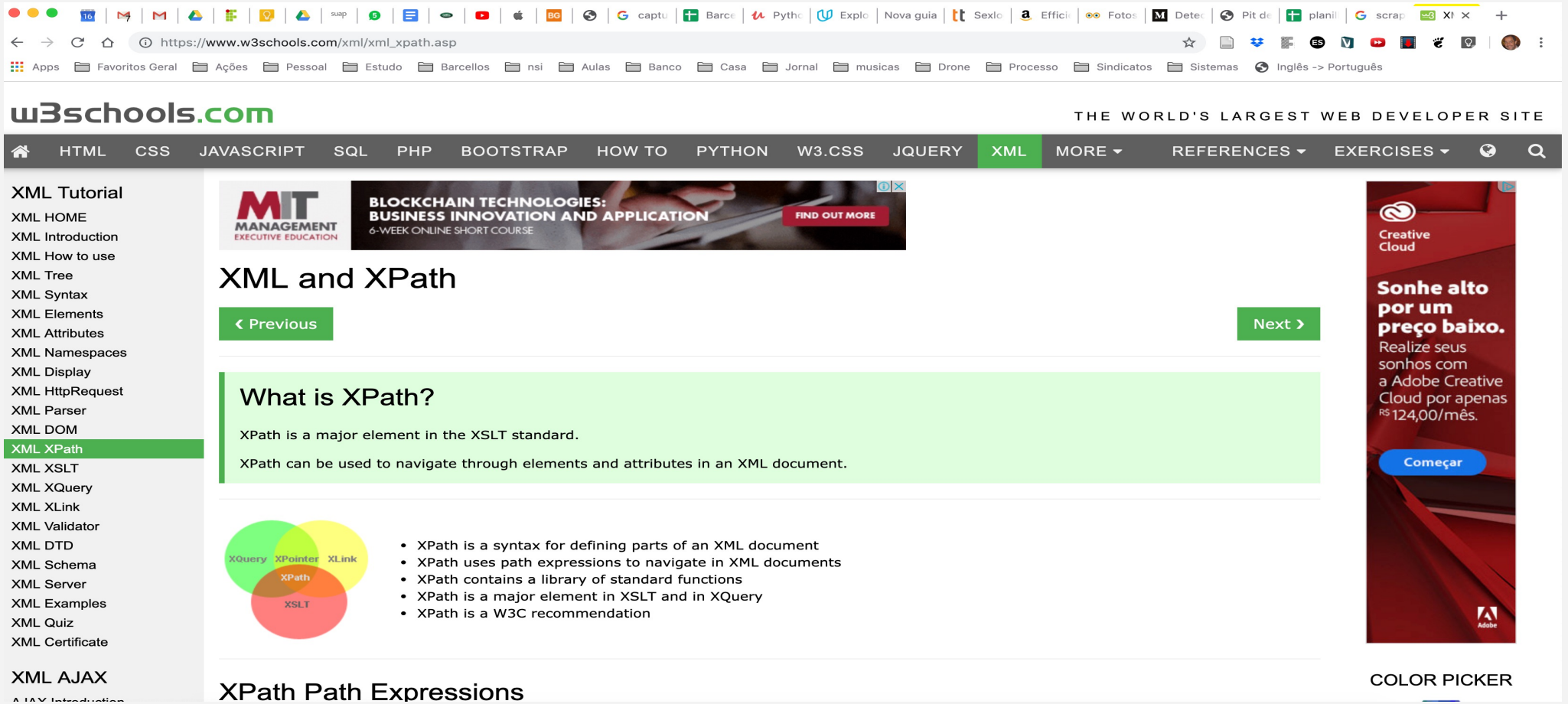
```
1   # -*- coding: utf-8 -*-
2   import scrapy
3
4
5   class CourseraSpider(scrapy.Spider):
6       name = 'coursera'
7       allowed_domains = ['https://www.coursera.org/browse']
8       start_urls = ['https://www.coursera.org/browse/']
9
10      def parse(self, response):
11          self.log('Estou no coursera')
12          #self.log(response.body)
13
```


Comandos de Linha

Crawl

scrapy crawl myspider

XPATH



Professor Luiz Gustavo

Scrapy Shell

- Cria um ambiente interativo com Scrapy
- Metodo parse

```
scrapy shell dominio
```



рахмат
danke
謝謝
spas
teşekkür ederim
Баярлалаа
mersi
kia ora
barka
welalin
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
blagodaram
dank je
misaotra
matondo
paldies
grazzi
спасибі
thank you
gracias
kiitos
dankie
nanni
nandri
dhanyavad
mauruuru
köszönöm
хвала
asante
manana
obrigada
murakoze
tenki
bedankt
bayarlalaa
gràçie
hvala
akun dankon aciū
djiere dieuf
tau
mochchakkeram
enkosi
дякую
mamnun
chokrane
dziękuję
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
sobodi
dēkuji
obrigado
sagolun
sukriya
kop khun krap
taiku
arigatô
takk
dakujem
trugarez
ありがとう
mèsi
najis tuke
terima kasih
tanemirt
grazie
rahmet
diolch
dhanyavadagalu
shukriya
мерси