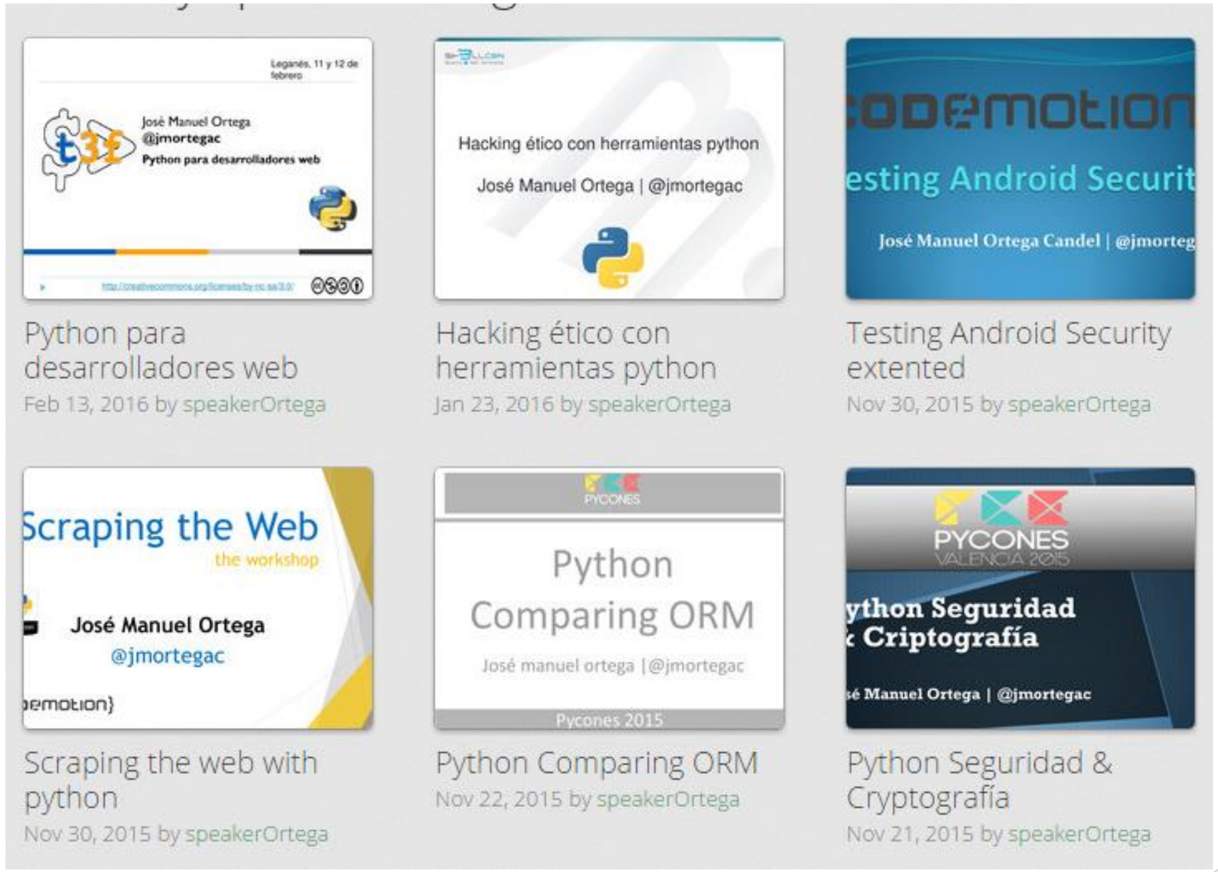
Python tools for webscraping

José Manuel Ortega
@jmortegac

PyData Madrid 2016

SpeakerDeck space

https://speakerdeck.com/jmortega



Python para desarrolladores web
Feb 13, 2016 by speakerOrtega

Hacking ético con herramientas python
Jan 23, 2016 by speakerOrtega

Testing Android Security extented
Nov 30, 2015 by speakerOrtega

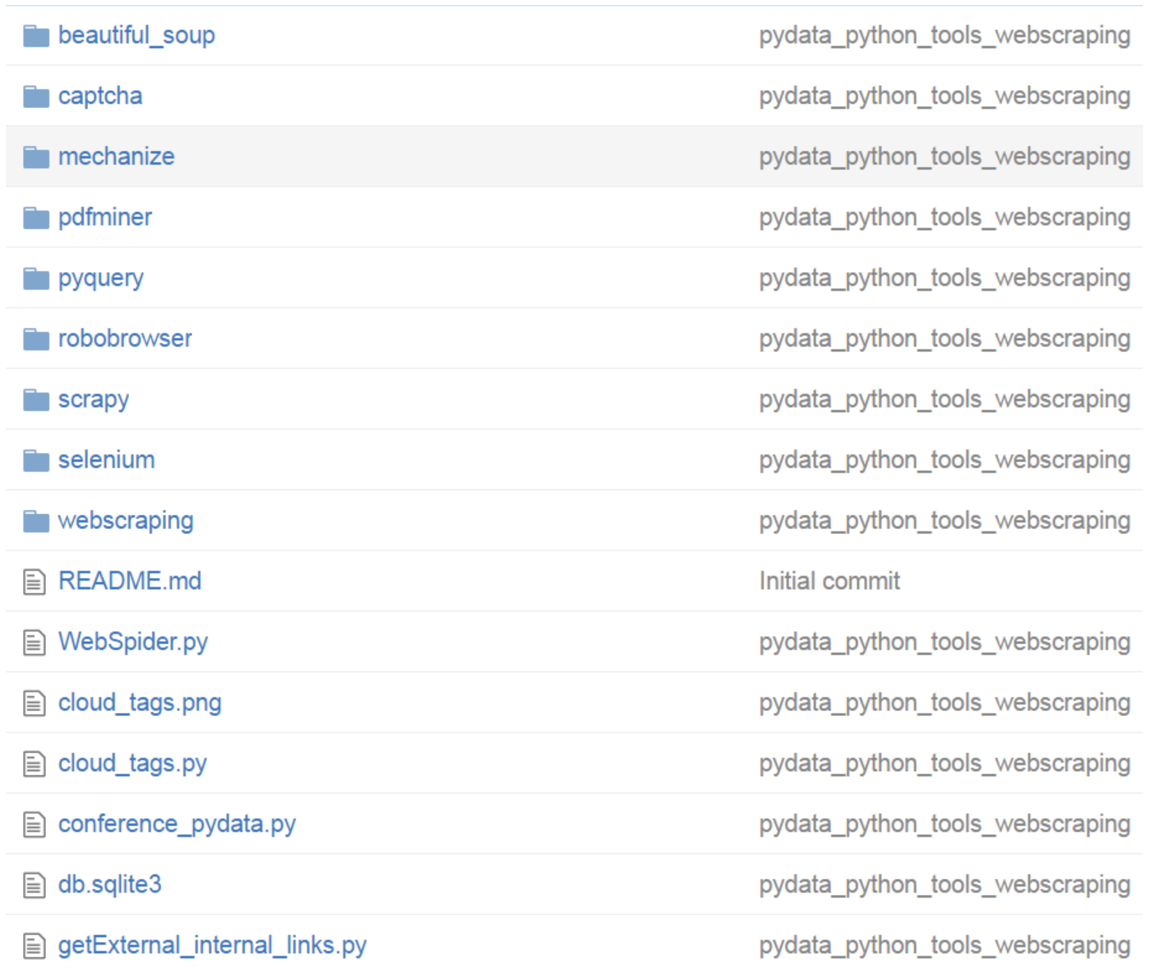
Scraping the web with python
Nov 30, 2015 by speakerOrtega

Python Comparing ORM
Nov 22, 2015 by speakerOrtega

Python Seguridad & Cryptografía
Nov 21, 2015 by speakerOrtega

Github repository

https://github.com/jmortega/pydata_webscraping



beautiful_soup	pydata_python_tools_webscraping
captcha	pydata_python_tools_webscraping
mechanize	pydata_python_tools_webscraping
pdfminer	pydata_python_tools_webscraping
pyquery	pydata_python_tools_webscraping
robobrowser	pydata_python_tools_webscraping
scrapy	pydata_python_tools_webscraping
selenium	pydata_python_tools_webscraping
webscraping	pydata_python_tools_webscraping
README.md	Initial commit
WebSpider.py	pydata_python_tools_webscraping
cloud_tags.png	pydata_python_tools_webscraping
cloud_tags.py	pydata_python_tools_webscraping
conference_pydata.py	pydata_python_tools_webscraping
db.sqlite3	pydata_python_tools_webscraping
getExternal_internal_links.py	pydata_python_tools_webscraping

Agenda

- Scraping techniques
- Introduction to webscraping
- Python tools for webscraping
- Scrapy project

Scraping techniques

- Screen scraping
- Report mining
- Web scraping
- Spiders /Crawlers

Screen scraping

- **Selenium**
- Mechanize
- Robobrowser

PhantomJS

Selenium

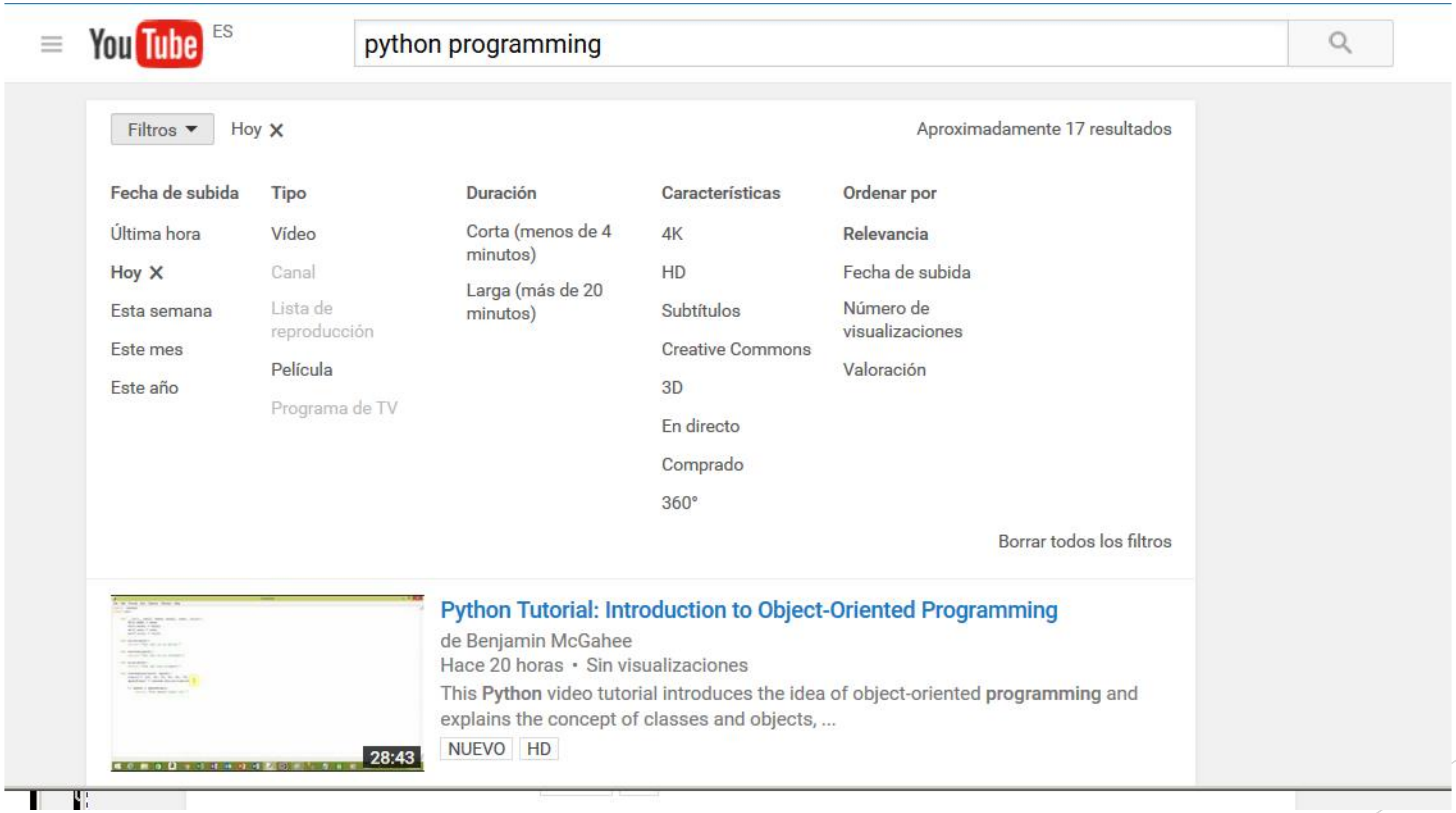
- Open Source framework for **automating browsers**
- Python-Module
  - http://pypi.python.org/pypi/selenium
- **pip install selenium**
- Firefox-Driver

```
>>> from selenium import webdriver
>>> browser = webdriver.Firefox()
>>> browser.get('http://google.com')
>>> browser.find_element_by_tag_name('title')
<selenium.webdriver.remote.webelement.WebElement ...>
```

Selenium

- find_element_
  by_link_text('text'): find the link by text
  by_css_selector: just like with lxml css
  by_tag_name: 'a' for the first link or all links
  by_xpath: practice xpath regex
  by_class_name: CSS related, but this finds all different types that have the same class

Selenium youtube



python programming

Filtros Hoy ✕ Aproximadamente 17 resultados

Fecha de subida	Tipo	Duración	Características	Ordenar por
Última hora	Vídeo	Corta (menos de 4 minutos)	4K	**Relevancia**
Hoy ✕	Canal	Larga (más de 20 minutos)	HD	Fecha de subida
Esta semana	Lista de reproducción		Subtítulos	Número de visualizaciones
Este mes	Película		Creative Commons	Valoración
Este año	Programa de TV		3D	
			En directo	
			Comprado	
			360°	

Borrar todos los filtros

Python Tutorial: Introduction to Object-Oriented Programming
de Benjamin McGahee
Hace 20 horas • Sin visualizaciones
This **Python** video tutorial introduces the idea of object-oriented **programming** and explains the concept of classes and objects, ...
NUEVO HD

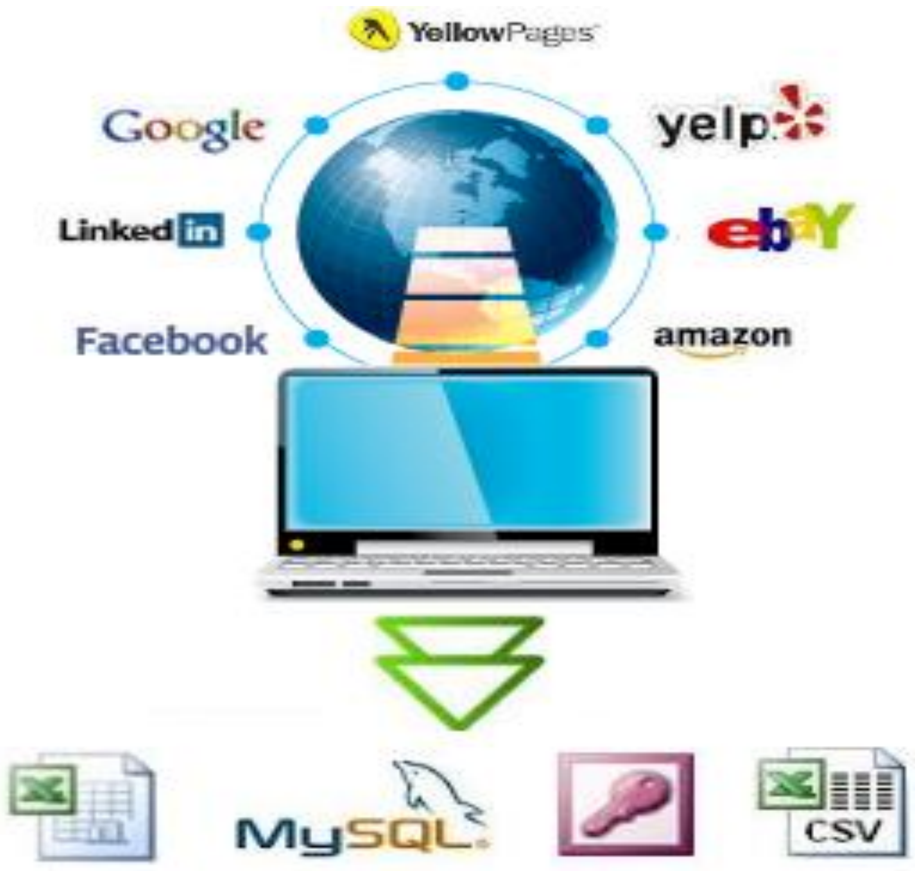
Selenium youtube search

```
import random
import time
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.keys import Keys

browser = webdriver.Firefox()

browser.get("http://www.youtube.com")
search_bar=browser.find_element_by_id('masthead-search-term')
search_bar.send_keys("python programming")
search_bar.submit()

filter_button = browser.find_element_by_class_name("filter-button-container").find_element_by_tag_name("button")
filter_button.click()
time.sleep(1)
browser.find_element_by_link_text("Hoy").click()
time.sleep(1)

videos = browser.find_elements_by_class_name("yt-uix-tile-link")
videoIndex = random.randint(2,len(videos))
print videos[videoIndex]
videos[videoIndex].click()
```

Report mining

PDF Miner

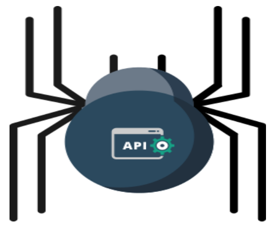
Webscraping



Python tools

- **Requests**
- **Beautiful Soup 4**
- **Pyquery**
- **Webscraping**
- **Scrapy**

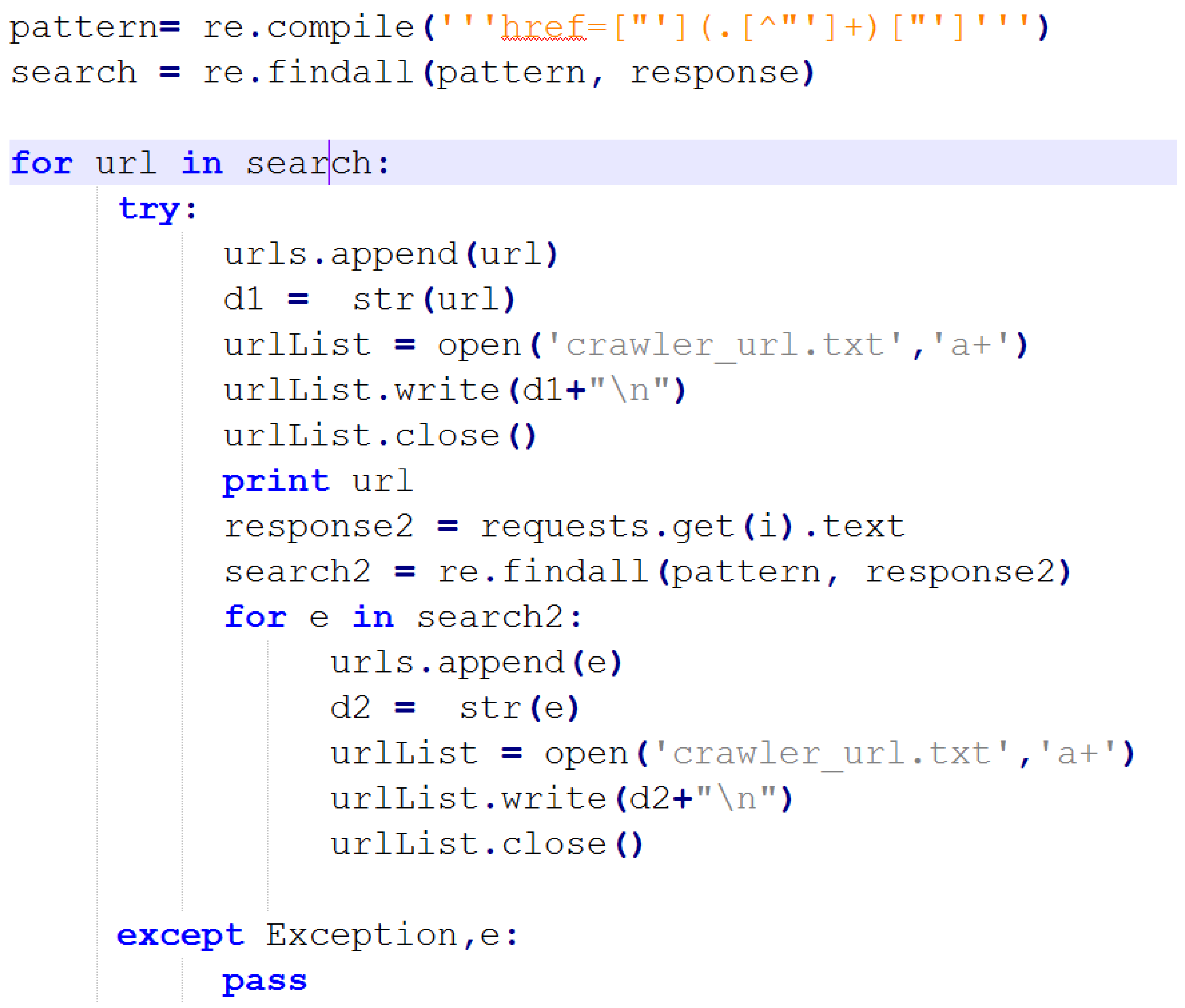
Spiders /crawlers



- A Web crawler is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing. A Web crawler may also be called a Web spider.

https://en.wikipedia.org/wiki/Web_crawler

Spiders /crawlers



```python
pattern= re.compile('''href=["'](.[^"']+)["']''')
search = re.findall(pattern, response)

for url in search:
    try:
        urls.append(url)
        d1 =  str(url)
        urlList = open('crawler_url.txt','a+')
        urlList.write(d1+"\n")
        urlList.close()
        print url
        response2 = requests.get(i).text
        search2 = re.findall(pattern, response2)
        for e in search2:
            urls.append(e)
            d2 =  str(e)
            urlList = open('crawler_url.txt','a+')
            urlList.write(d2+"\n")
            urlList.close()

    except Exception,e:
        pass
```

Spiders /crawlers

Scrapy

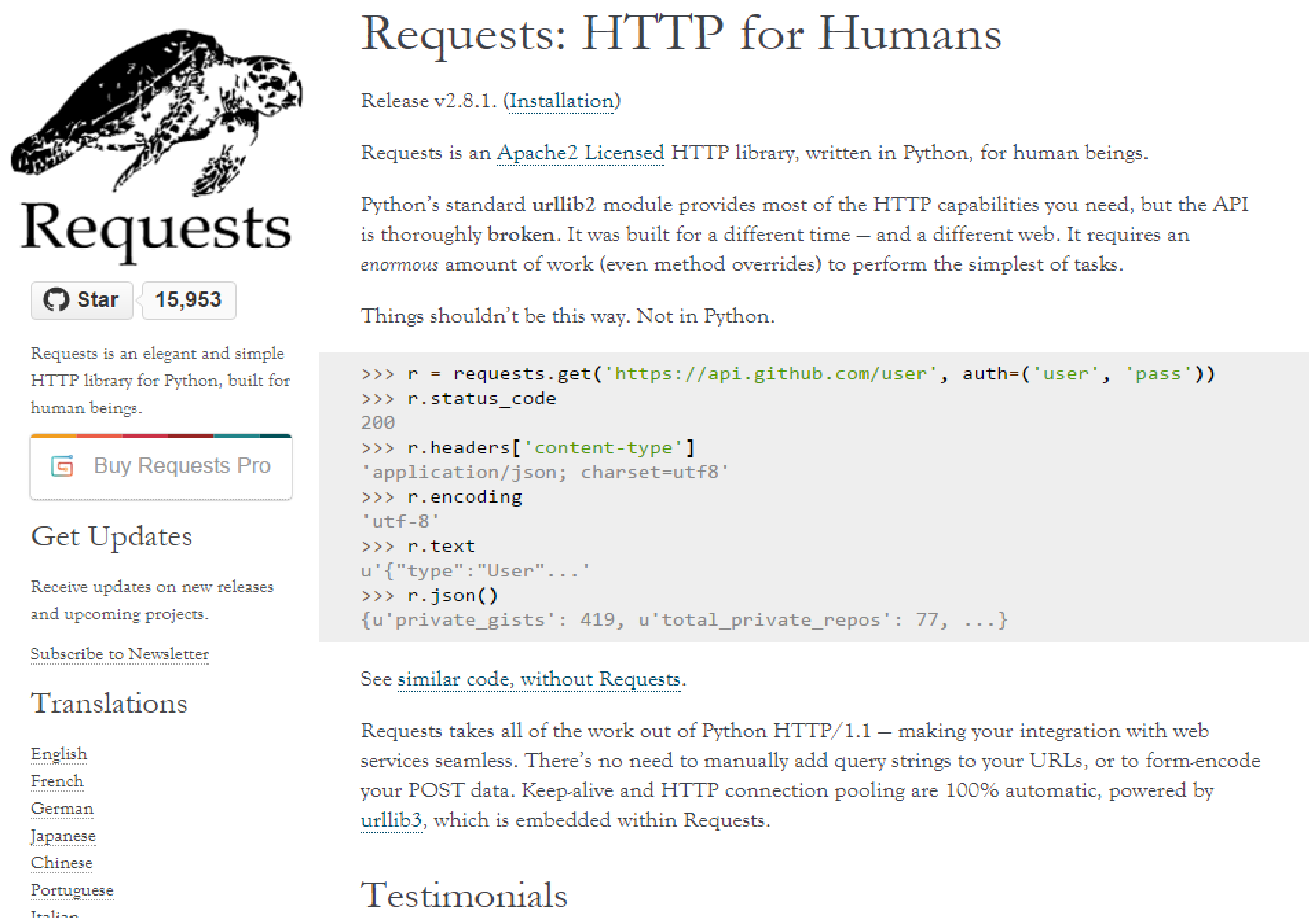
scrapinghub.com

Request libraries

- **Urllib2**
- Python ***requests***: HTTP for Humans
  - $ pip install requests

Requests

Requests http://docs.python-requests.org/en/latest



Requests is an elegant and simple HTTP library for Python, built for human beings.

Get Updates

Receive updates on new releases and upcoming projects.

Subscribe to Newsletter

Translations

English
French
German
Japanese
Chinese
Portuguese

Requests: HTTP for Humans

Release v2.8.1. (Installation)

Requests is an Apache2 Licensed HTTP library, written in Python, for human beings.

Python's standard **urllib2** module provides most of the HTTP capabilities you need, but the API is thoroughly **broken**. It was built for a different time — and a different web. It requires an *enormous* amount of work (even method overrides) to perform the simplest of tasks.

Things shouldn't be this way. Not in Python.

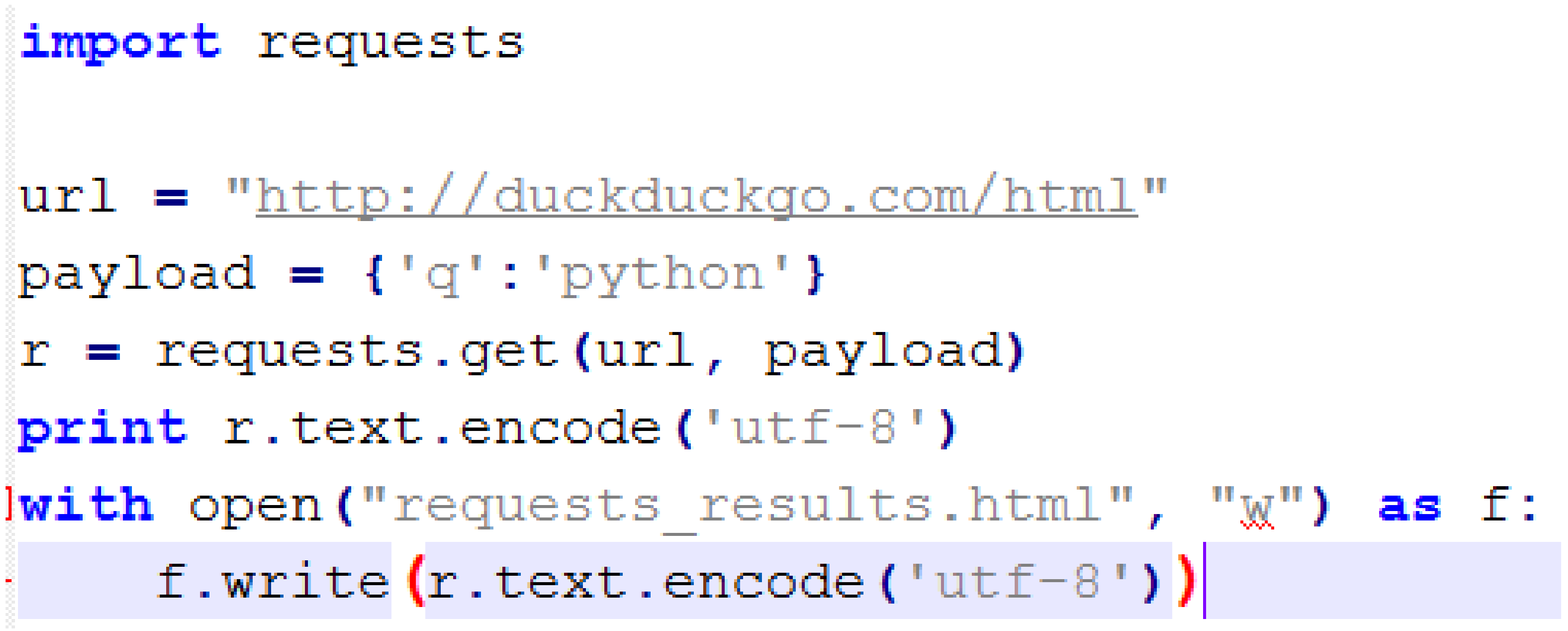
```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type":"User"...'
>>> r.json()
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

See similar code, without Requests.

Requests takes all of the work out of Python HTTP/1.1 — making your integration with web services seamless. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, powered by urllib3, which is embedded within Requests.

Testimonials

Requests



```
import requests

url = "http://duckduckgo.com/html"
payload = {'q':'python'}
r = requests.get(url, payload)
print r.text.encode('utf-8')
with open("requests_results.html", "w") as f:
    f.write(r.text.encode('utf-8'))
```

Web scraping with Python

1. **Download webpage with requests**

2. **Parse the page with BeautifulSoup/lxml**

3. **Select elements with Regular expressions,XPath or css selectors**

Xpath selectors

Expression	Meaning
name	matches all nodes on the current level with the specified name
name[n]	matches the nth element on the current level with the specified name
/	Do selection from the root
//	Do selection from current node
*****	matches all nodes on the current level
. Or ..	Select current / parent node
@name	the attribute with the specified name
[@key='value']	all elements with an attribute that matches the specified key/value pair
name[@key='value']	all elements with the specified name and an attribute that matches the specified key/value pair
[text()='value']	all elements with the specified text
name[text()='value']	all elements with the specified name and text

BeautifulSoup

- **Parsers support→ lxml,html5lib**
- **Installation**
  - **pip install lxml**
  - **pip install html5lib**
  - **pip install beautifulsoup4**
  - http://www.crummy.com/software/BeautifulSoup

BeautifulSoup

- soup = BeautifulSoup(html_doc,’lxml’)
- Print all: print(soup.prettify())
- Print text: print(soup.get_text())

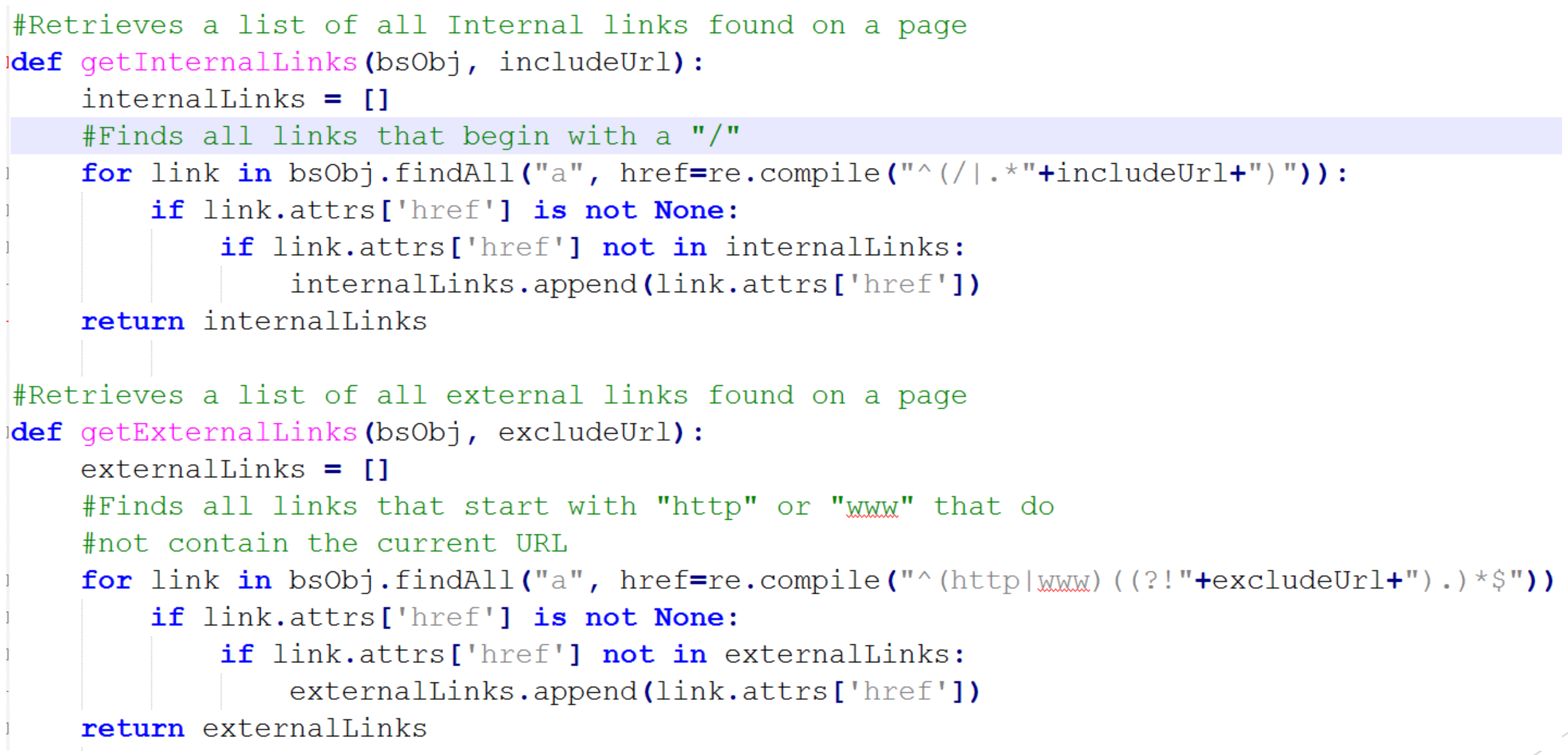
```
from bs4 import BeautifulSoup
```

BeautifulSoup functions

- **find_all('a')**→Returns all links
- **find('title')**→Returns the first element <title>
- **get('href')**→Returns the attribute href value
- **(element).text** → Returns the text inside an element

```
for link in soup.find_all('a'):
    print(link.get('href'))
```

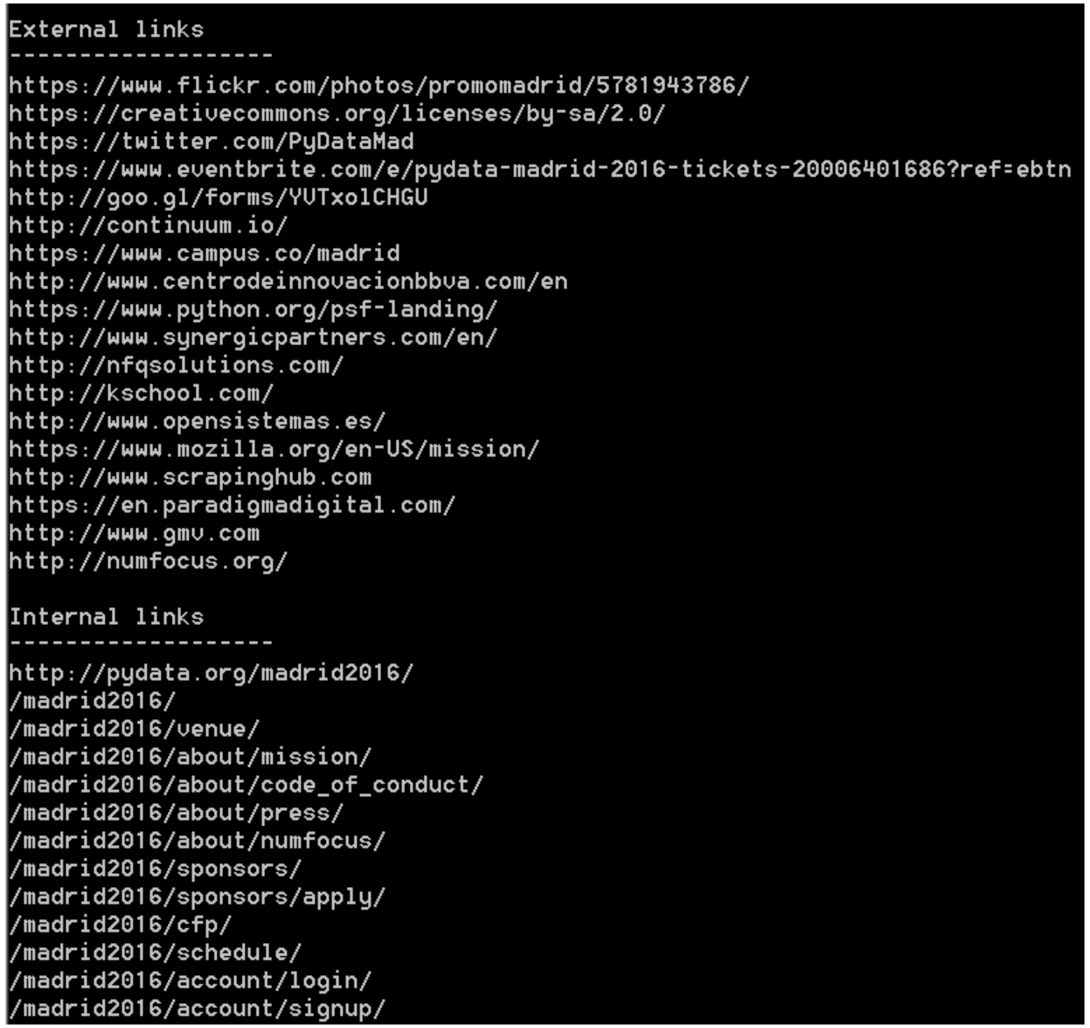
External/internal links



```
#Retrieves a list of all Internal links found on a page
def getInternalLinks(bsObj, includeUrl):
    internalLinks = []
    #Finds all links that begin with a "/"
    for link in bsObj.findAll("a", href=re.compile("^(/|.*"+includeUrl+")")):
        if link.attrs['href'] is not None:
            if link.attrs['href'] not in internalLinks:
                internalLinks.append(link.attrs['href'])
    return internalLinks

#Retrieves a list of all external links found on a page
def getExternalLinks(bsObj, excludeUrl):
    externalLinks = []
    #Finds all links that start with "http" or "www" that do
    #not contain the current URL
    for link in bsObj.findAll("a", href=re.compile("^(http|www)((?!"+excludeUrl+").)*$")):
        if link.attrs['href'] is not None:
            if link.attrs['href'] not in externalLinks:
                externalLinks.append(link.attrs['href'])
    return externalLinks
```

External/internal links

http://pydata.org/madrid2016



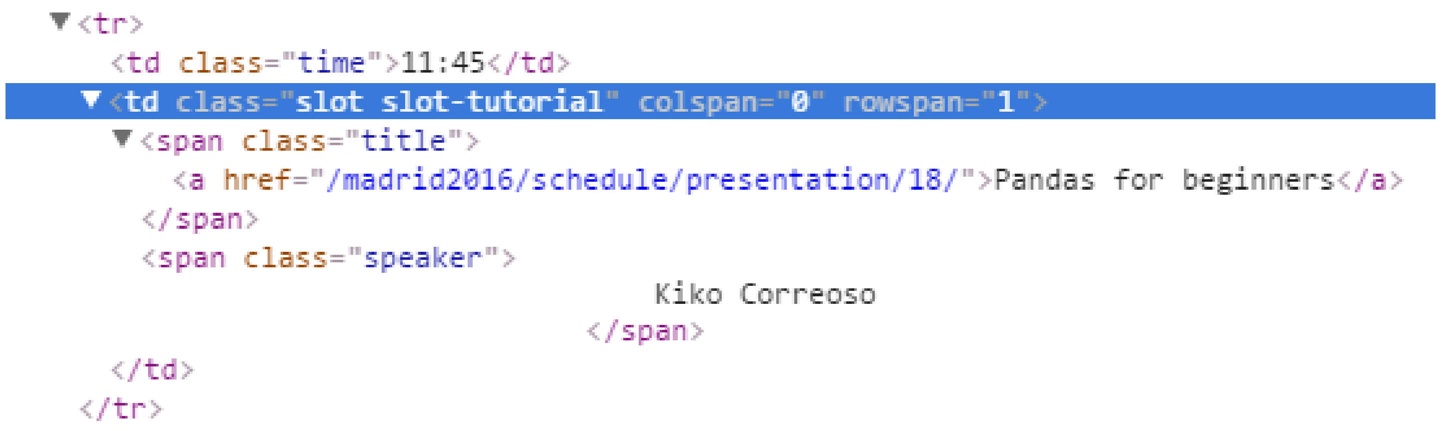
```
External links
--------------------
https://www.flickr.com/photos/promomadrid/5781943786/
https://creativecommons.org/licenses/by-sa/2.0/
https://twitter.com/PyDataMad
https://www.eventbrite.com/e/pydata-madrid-2016-tickets-20006401686?ref=ebtn
http://goo.gl/forms/YVTxolCHGU
http://continuum.io/
https://www.campus.co/madrid
http://www.centrodeinnovacionbbva.com/en
https://www.python.org/psf-landing/
http://www.synergicpartners.com/en/
http://nfqsolutions.com/
http://kschool.com/
http://www.opensistemas.es/
https://www.mozilla.org/en-US/mission/
http://www.scrapinghub.com
https://en.paradigmadigital.com/
http://www.gmv.com
http://numfocus.org/

Internal links
--------------------
http://pydata.org/madrid2016/
/madrid2016/
/madrid2016/venue/
/madrid2016/about/mission/
/madrid2016/about/code_of_conduct/
/madrid2016/about/press/
/madrid2016/about/numfocus/
/madrid2016/sponsors/
/madrid2016/sponsors/apply/
/madrid2016/cfp/
/madrid2016/schedule/
/madrid2016/account/login/
/madrid2016/account/signup/
```

Webscraping

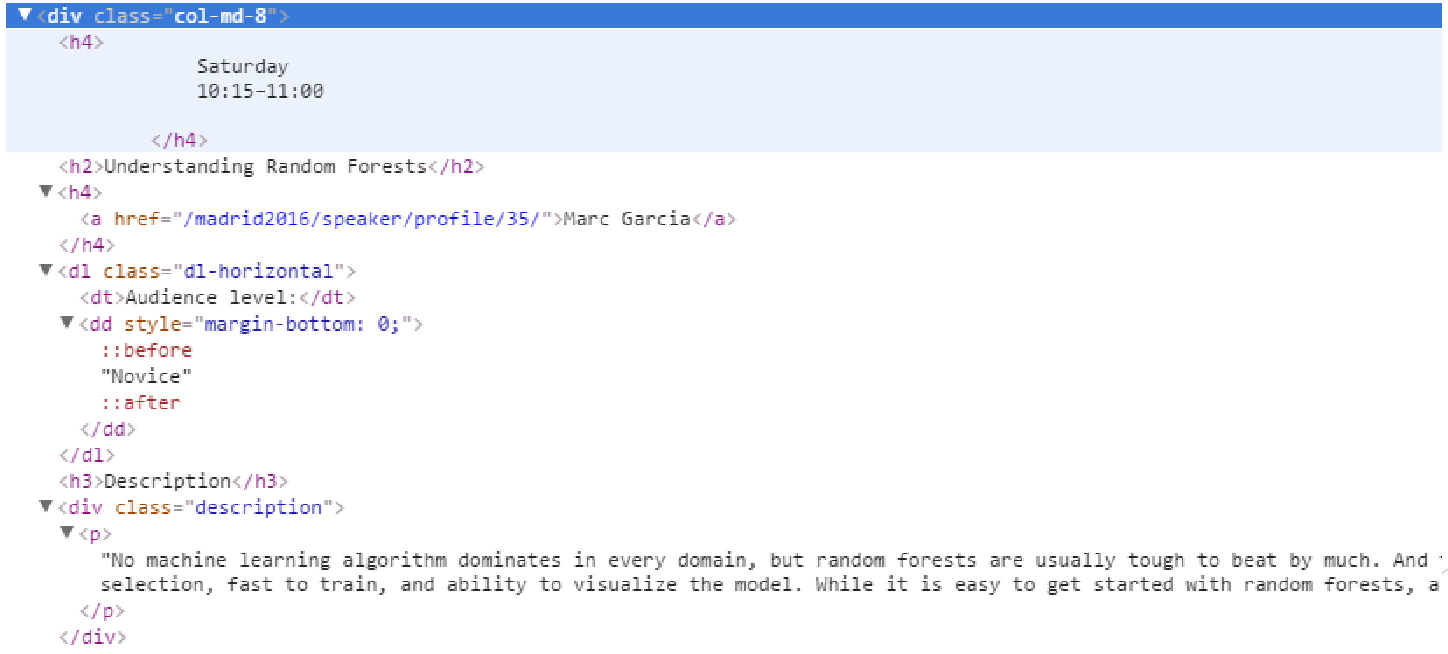
- ***pip install webscraping***

```
#Download instance
D = download.Download()

#get page
html =
D.get('http://pydata.org/madrid2016/schedule/')

#get element where is located information
xpath.search(html, '//td[@class="slot slot-talk"]')
```

Pydata agenda code structure

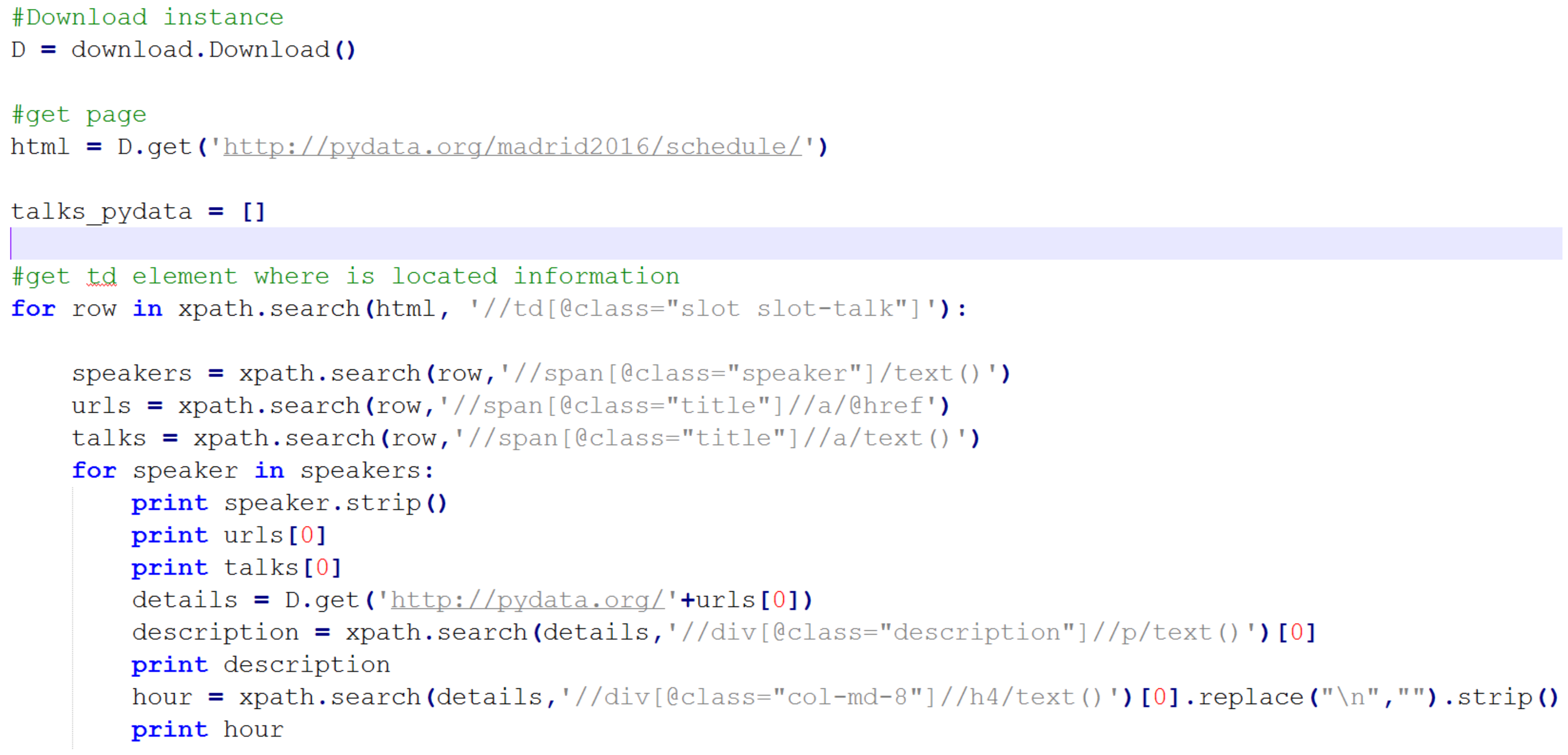


```
▼<tr>
   <td class="time">11:45</td>
  ▼<td class="slot slot-tutorial" colspan="0" rowspan="1">
    ▼<span class="title">
       <a href="/madrid2016/schedule/presentation/18/">Pandas for beginners</a>
     </span>
     <span class="speaker">
                              Kiko Correoso
                          </span>
   </td>
 </tr>
```



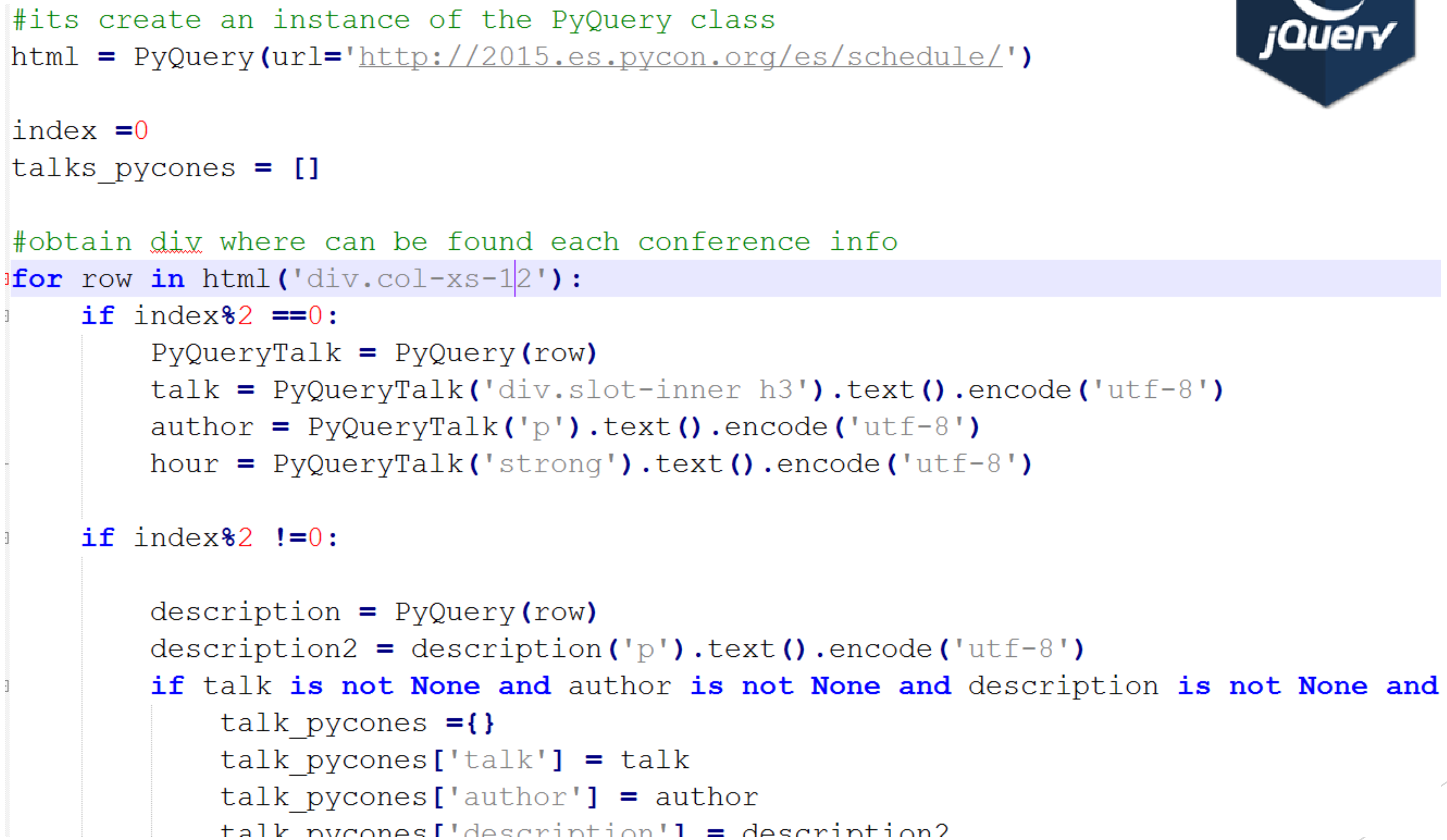
```
▼<div class="col-md-8">
   <h4>
          Saturday
          10:15-11:00

       </h4>
   <h2>Understanding Random Forests</h2>
  ▼<h4>
     <a href="/madrid2016/speaker/profile/35/">Marc Garcia</a>
   </h4>
  ▼<dl class="dl-horizontal">
     <dt>Audience level:</dt>
    ▼<dd style="margin-bottom: 0;">
       ::before
       "Novice"
       ::after
     </dd>
   </dl>
   <h3>Description</h3>
  ▼<div class="description">
    ▼<p>
       "No machine learning algorithm dominates in every domain, but random forests are usually tough to beat by much. And 
       selection, fast to train, and ability to visualize the model. While it is easy to get started with random forests, a
     </p>
   </div>
```

Extract data from pycones agenda

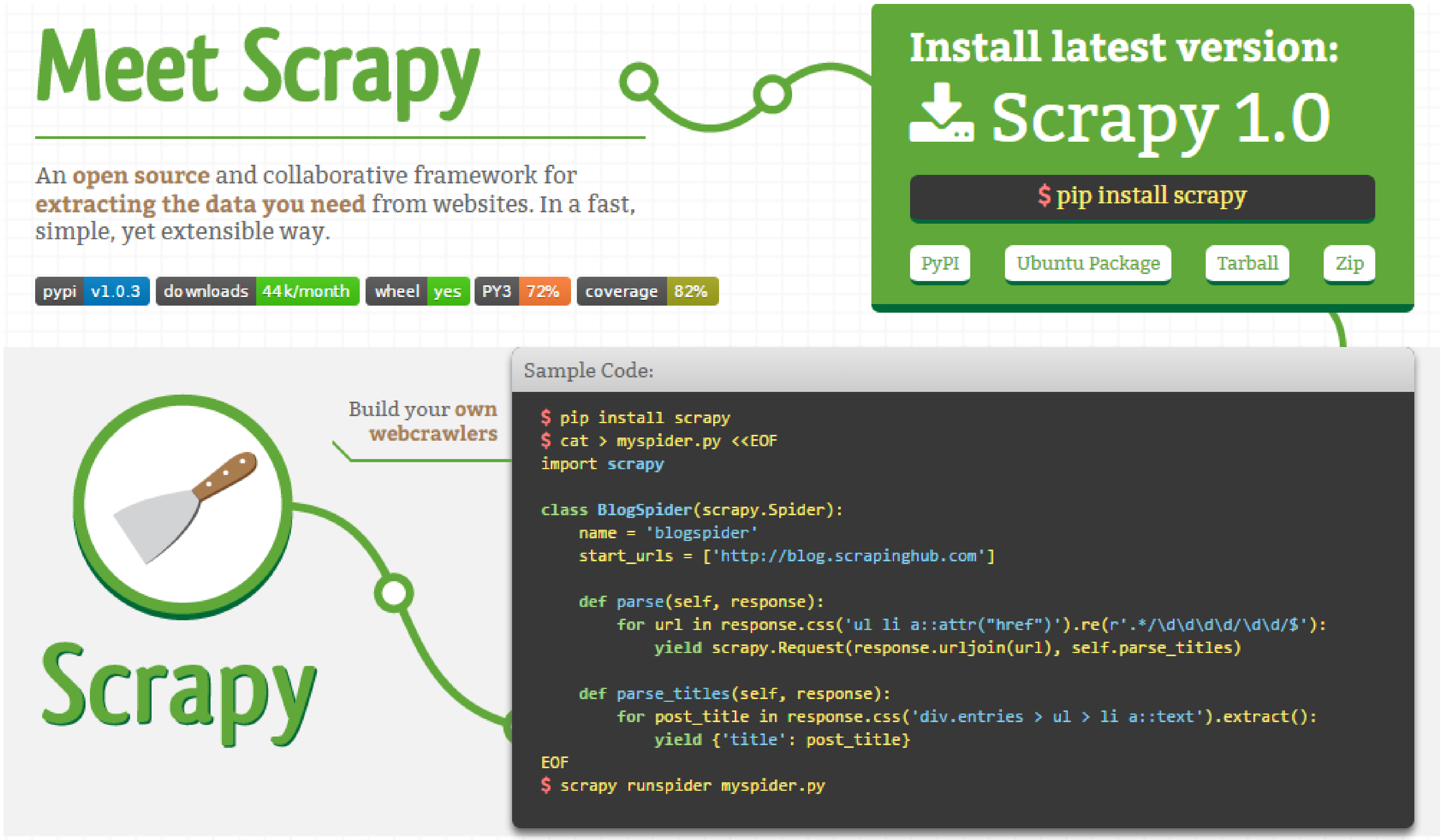


```
#Download instance
D = download.Download()

#get page
html = D.get('http://pydata.org/madrid2016/schedule/')

talks_pydata = []

#get td element where is located information
for row in xpath.search(html, '//td[@class="slot slot-talk"]'):

    speakers = xpath.search(row,'//span[@class="speaker"]/text()')
    urls = xpath.search(row,'//span[@class="title"]//a/@href')
    talks = xpath.search(row,'//span[@class="title"]//a/text()')
    for speaker in speakers:
        print speaker.strip()
        print urls[0]
        print talks[0]
        details = D.get('http://pydata.org/'+urls[0])
        description = xpath.search(details,'//div[@class="description"]//p/text()')[0]
        print description
        hour = xpath.search(details,'//div[@class="col-md-8"]//h4/text()')[0].replace("\n","").strip()
        print hour
```

PyQuery



```
#its create an instance of the PyQuery class
html = PyQuery(url='http://2015.es.pycon.org/es/schedule/')

index =0
talks_pycones = []

#obtain div where can be found each conference info
for row in html('div.col-xs-12'):
    if index%2 ==0:
        PyQueryTalk = PyQuery(row)
        talk = PyQueryTalk('div.slot-inner h3').text().encode('utf-8')
        author = PyQueryTalk('p').text().encode('utf-8')
        hour = PyQueryTalk('strong').text().encode('utf-8')

    if index%2 !=0:

        description = PyQuery(row)
        description2 = description('p').text().encode('utf-8')
        if talk is not None and author is not None and description is not None and
            talk_pycones ={}
            talk_pycones['talk'] = talk
            talk_pycones['author'] = author
            talk_pycones['description'] = description2
```



Meet Scrapy

An **open source** and collaborative framework for **extracting the data you need** from websites. In a fast, simple, yet extensible way.

Install latest version:

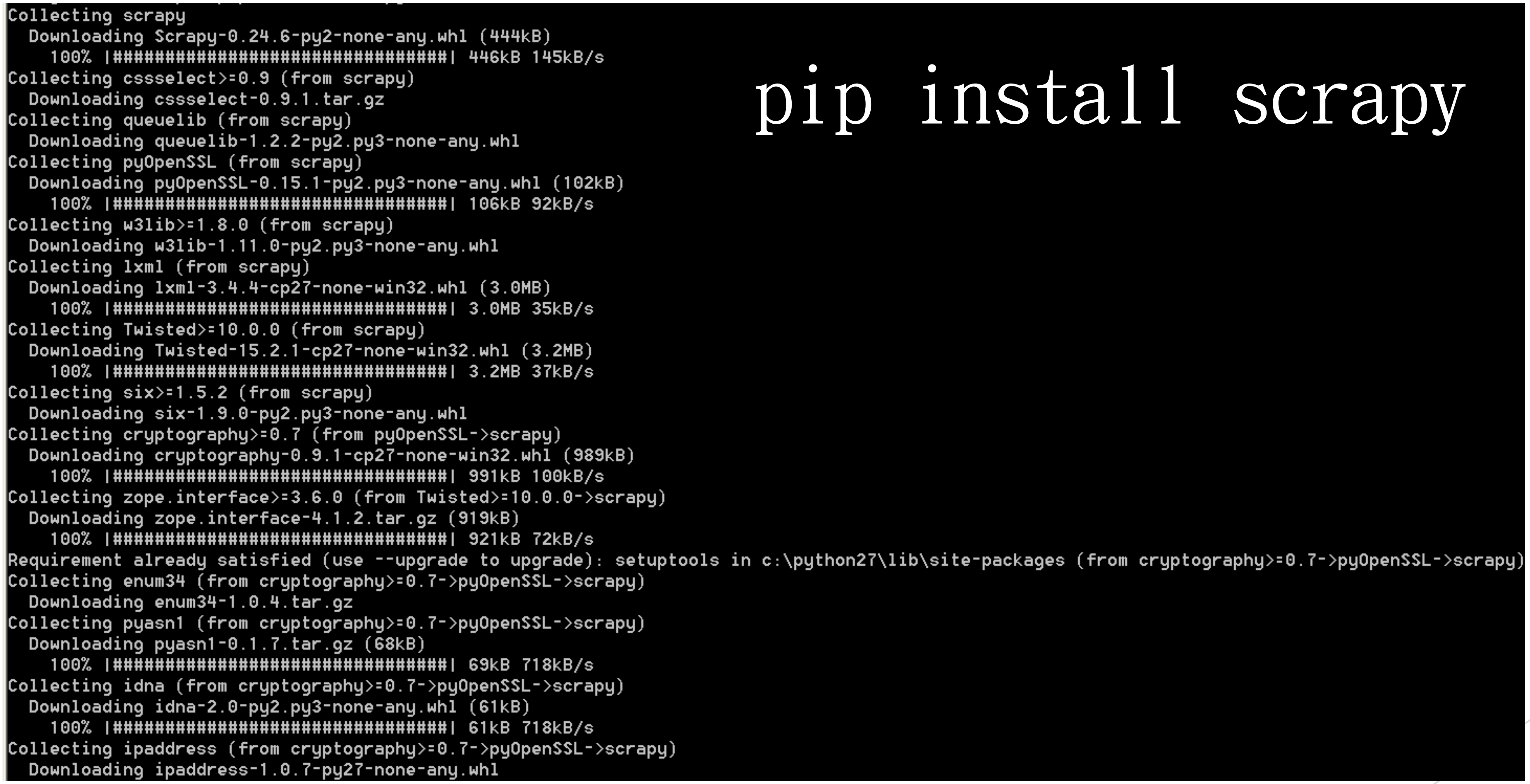
Scrapy 1.0

```
$ pip install scrapy
```

Sample Code:

```
$ pip install scrapy
$ cat > myspider.py <<EOF
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['http://blog.scrapinghub.com']

    def parse(self, response):
        for url in response.css('ul li a::attr("href")').re(r'.*/\d\d\d\d/\d\d/$'):
            yield scrapy.Request(response.urljoin(url), self.parse_titles)

    def parse_titles(self, response):
        for post_title in response.css('div.entries > ul > li a::text').extract():
            yield {'title': post_title}
EOF
$ scrapy runspider myspider.py
```

Scrapy installation



pip install scrapy

```
Collecting scrapy
  Downloading Scrapy-0.24.6-py2-none-any.whl (444kB)
    100% |################################| 446kB 145kB/s
Collecting cssselect>=0.9 (from scrapy)
  Downloading cssselect-0.9.1.tar.gz
Collecting queuelib (from scrapy)
  Downloading queuelib-1.2.2-py2.py3-none-any.whl
Collecting pyOpenSSL (from scrapy)
  Downloading pyOpenSSL-0.15.1-py2.py3-none-any.whl (102kB)
    100% |################################| 106kB 92kB/s
Collecting w3lib>=1.8.0 (from scrapy)
  Downloading w3lib-1.11.0-py2.py3-none-any.whl
Collecting lxml (from scrapy)
  Downloading lxml-3.4.4-cp27-none-win32.whl (3.0MB)
    100% |################################| 3.0MB 35kB/s
Collecting Twisted>=10.0.0 (from scrapy)
  Downloading Twisted-15.2.1-cp27-none-win32.whl (3.2MB)
    100% |################################| 3.2MB 37kB/s
Collecting six>=1.5.2 (from scrapy)
  Downloading six-1.9.0-py2.py3-none-any.whl
Collecting cryptography>=0.7 (from pyOpenSSL->scrapy)
  Downloading cryptography-0.9.1-cp27-none-win32.whl (989kB)
    100% |################################| 991kB 100kB/s
Collecting zope.interface>=3.6.0 (from Twisted>=10.0.0->scrapy)
  Downloading zope.interface-4.1.2.tar.gz (919kB)
    100% |################################| 921kB 72kB/s
Requirement already satisfied (use --upgrade to upgrade): setuptools in c:\python27\lib\site-packages (from cryptography>=0.7->pyOpenSSL->scrapy)
Collecting enum34 (from cryptography>=0.7->pyOpenSSL->scrapy)
  Downloading enum34-1.0.4.tar.gz
Collecting pyasn1 (from cryptography>=0.7->pyOpenSSL->scrapy)
  Downloading pyasn1-0.1.7.tar.gz (68kB)
    100% |################################| 69kB 718kB/s
Collecting idna (from cryptography>=0.7->pyOpenSSL->scrapy)
  Downloading idna-2.0-py2.py3-none-any.whl (61kB)
    100% |################################| 61kB 718kB/s
Collecting ipaddress (from cryptography>=0.7->pyOpenSSL->scrapy)
  Downloading ipaddress-1.0.7-py27-none-any.whl
```

Scrapy

- Define your own data structures
- Built-in XPath selectors to extracting data
- Write spiders to extract data
- Built-in JSON, CSV, XML output
- Interactive shell console

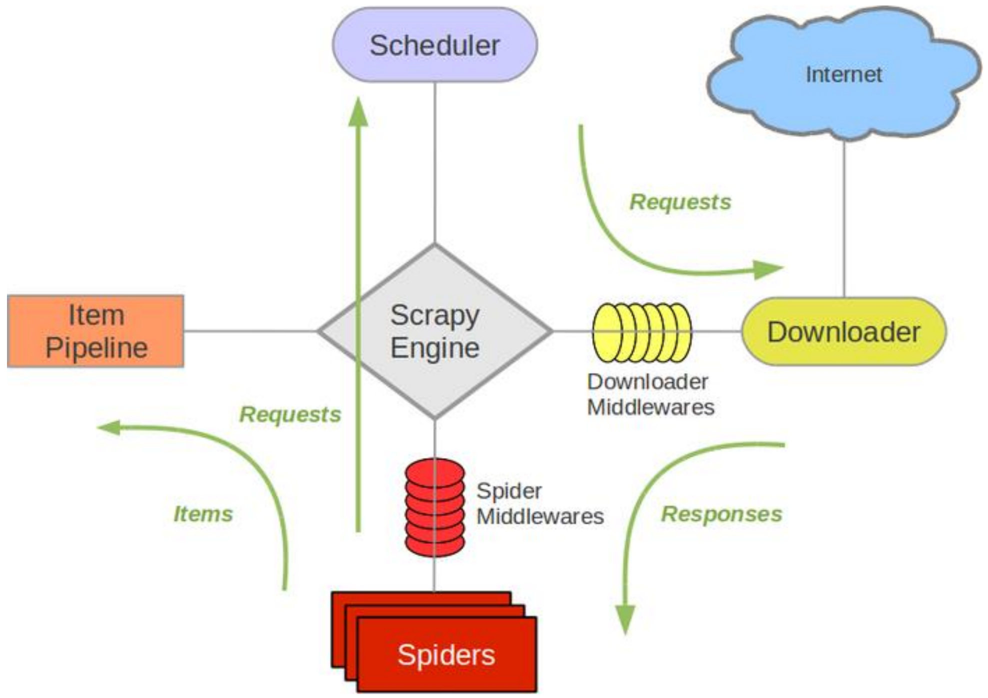
Scrapy

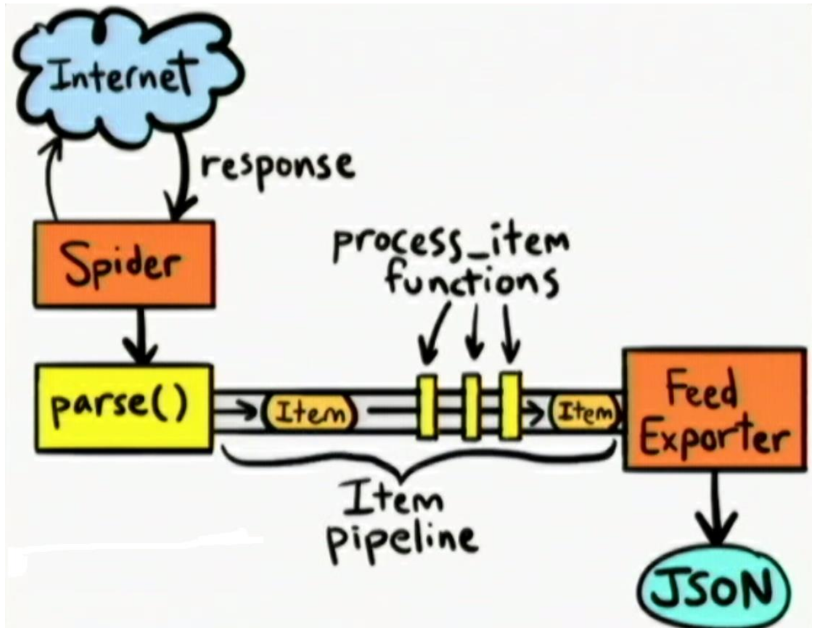
- Uses a mechanism based on XPath expressions called **Xpath Selectors**.
- Uses Parser **LXML** to find elements
- **Twisted** for asyncronous operations

Scrapy advantages

- Faster than mechanize because it uses asynchronous operations (Twisted).
- Scrapy has better support for html parsing.
- Scrapy handles better unicode characters, redirections, gzipped responses, encodings.
- HTTP cache integrated.
- You can export the extracted data directly to csv to JSON.

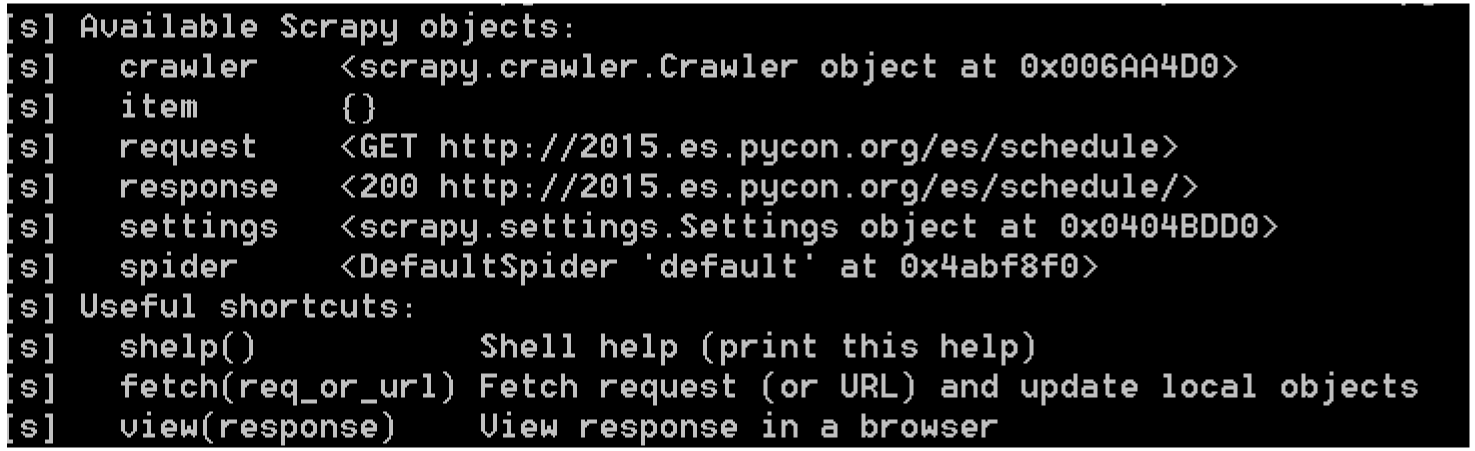
Architecture





Scrapy Shell

```
scrapy shell <url>
```

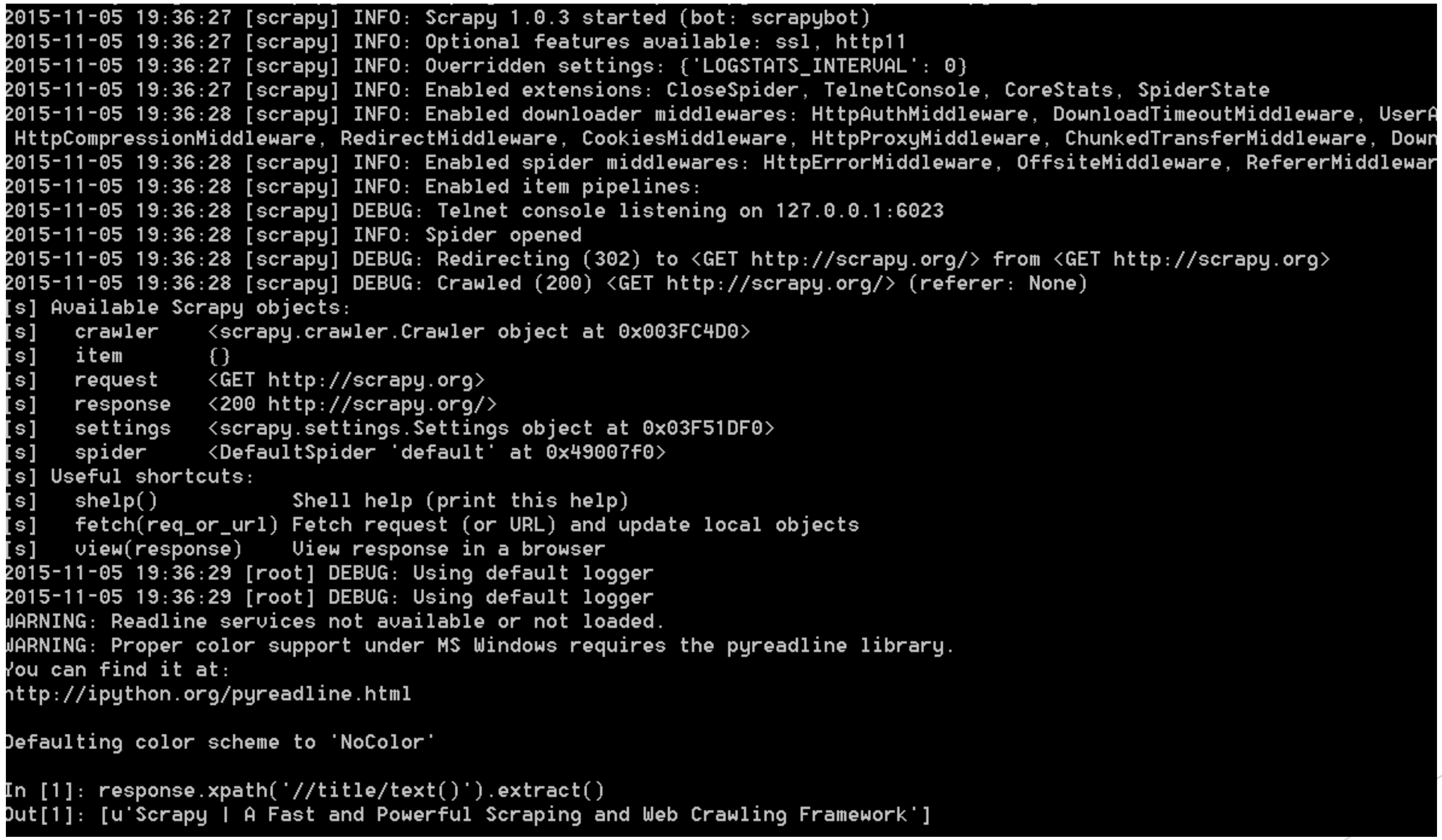


```
[s] Available Scrapy objects:
[s]   crawler    <scrapy.crawler.Crawler object at 0x006AA4D0>
[s]   item       {}
[s]   request    <GET http://2015.es.pycon.org/es/schedule>
[s]   response   <200 http://2015.es.pycon.org/es/schedule/>
[s]   settings   <scrapy.settings.Settings object at 0x0404BDD0>
[s]   spider     <DefaultSpider 'default' at 0x4abf8f0>
[s] Useful shortcuts:
[s]   shelp()           Shell help (print this help)
[s]   fetch(req_or_url) Fetch request (or URL) and update local objects
[s]   view(response)    View response in a browser
```

```
from scrapy.select import Selector
hxs = Selector(response)
Info = hxs.select(‘//div[@class=“slot-inner”]’)
```

Scrapy Shell

scrapy shell http://scrapy.org



```
2015-11-05 19:36:27 [scrapy] INFO: Scrapy 1.0.3 started (bot: scrapybot)
2015-11-05 19:36:27 [scrapy] INFO: Optional features available: ssl, http11
2015-11-05 19:36:27 [scrapy] INFO: Overridden settings: {'LOGSTATS_INTERVAL': 0}
2015-11-05 19:36:27 [scrapy] INFO: Enabled extensions: CloseSpider, TelnetConsole, CoreStats, SpiderState
2015-11-05 19:36:28 [scrapy] INFO: Enabled downloader middlewares: HttpAuthMiddleware, DownloadTimeoutMiddleware, UserA
 HttpCompressionMiddleware, RedirectMiddleware, CookiesMiddleware, HttpProxyMiddleware, ChunkedTransferMiddleware, Down
2015-11-05 19:36:28 [scrapy] INFO: Enabled spider middlewares: HttpErrorMiddleware, OffsiteMiddleware, RefererMiddlewar
2015-11-05 19:36:28 [scrapy] INFO: Enabled item pipelines:
2015-11-05 19:36:28 [scrapy] DEBUG: Telnet console listening on 127.0.0.1:6023
2015-11-05 19:36:28 [scrapy] INFO: Spider opened
2015-11-05 19:36:28 [scrapy] DEBUG: Redirecting (302) to <GET http://scrapy.org/> from <GET http://scrapy.org>
2015-11-05 19:36:28 [scrapy] DEBUG: Crawled (200) <GET http://scrapy.org/> (referer: None)
[s] Available Scrapy objects:
[s]   crawler    <scrapy.crawler.Crawler object at 0x003FC4D0>
[s]   item       {}
[s]   request    <GET http://scrapy.org>
[s]   response   <200 http://scrapy.org/>
[s]   settings   <scrapy.settings.Settings object at 0x03F51DF0>
[s]   spider     <DefaultSpider 'default' at 0x49007f0>
[s] Useful shortcuts:
[s]   shelp()           Shell help (print this help)
[s]   fetch(req_or_url) Fetch request (or URL) and update local objects
[s]   view(response)    View response in a browser
2015-11-05 19:36:29 [root] DEBUG: Using default logger
2015-11-05 19:36:29 [root] DEBUG: Using default logger
WARNING: Readline services not available or not loaded.
WARNING: Proper color support under MS Windows requires the pyreadline library.
You can find it at:
http://ipython.org/pyreadline.html

Defaulting color scheme to 'NoColor'

In [1]: response.xpath('//title/text()').extract()
Out[1]: [u'Scrapy | A Fast and Powerful Scraping and Web Crawling Framework']
```

Scrapy project

$ scrapy startproject <project_name>

scrapy.cfg: the project configuration file.

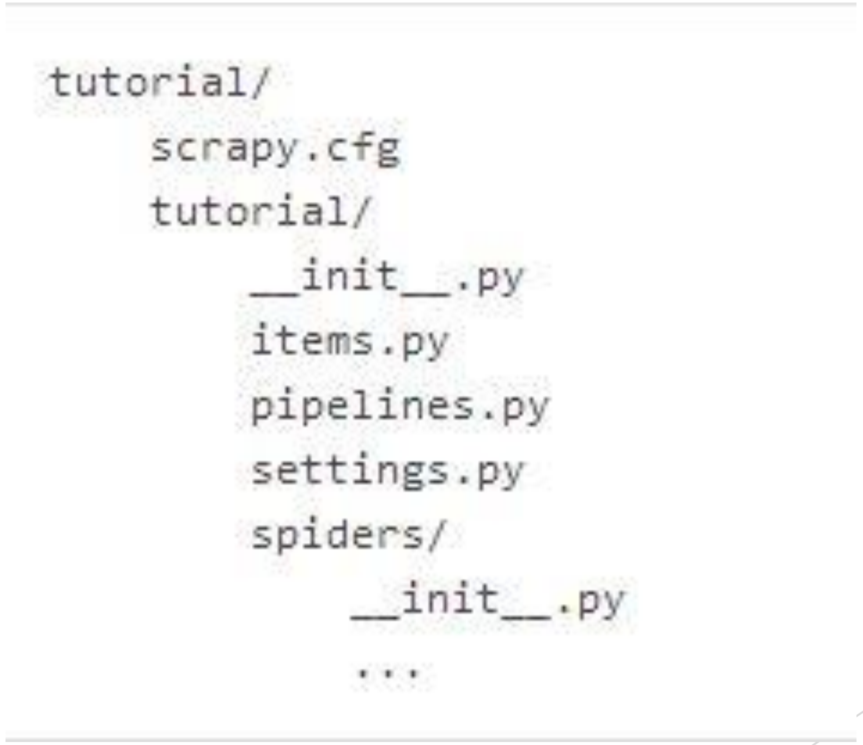
tutorial/:the project's python module.

items.py: the project's items file.

pipelines.py : the project's pipelines file.

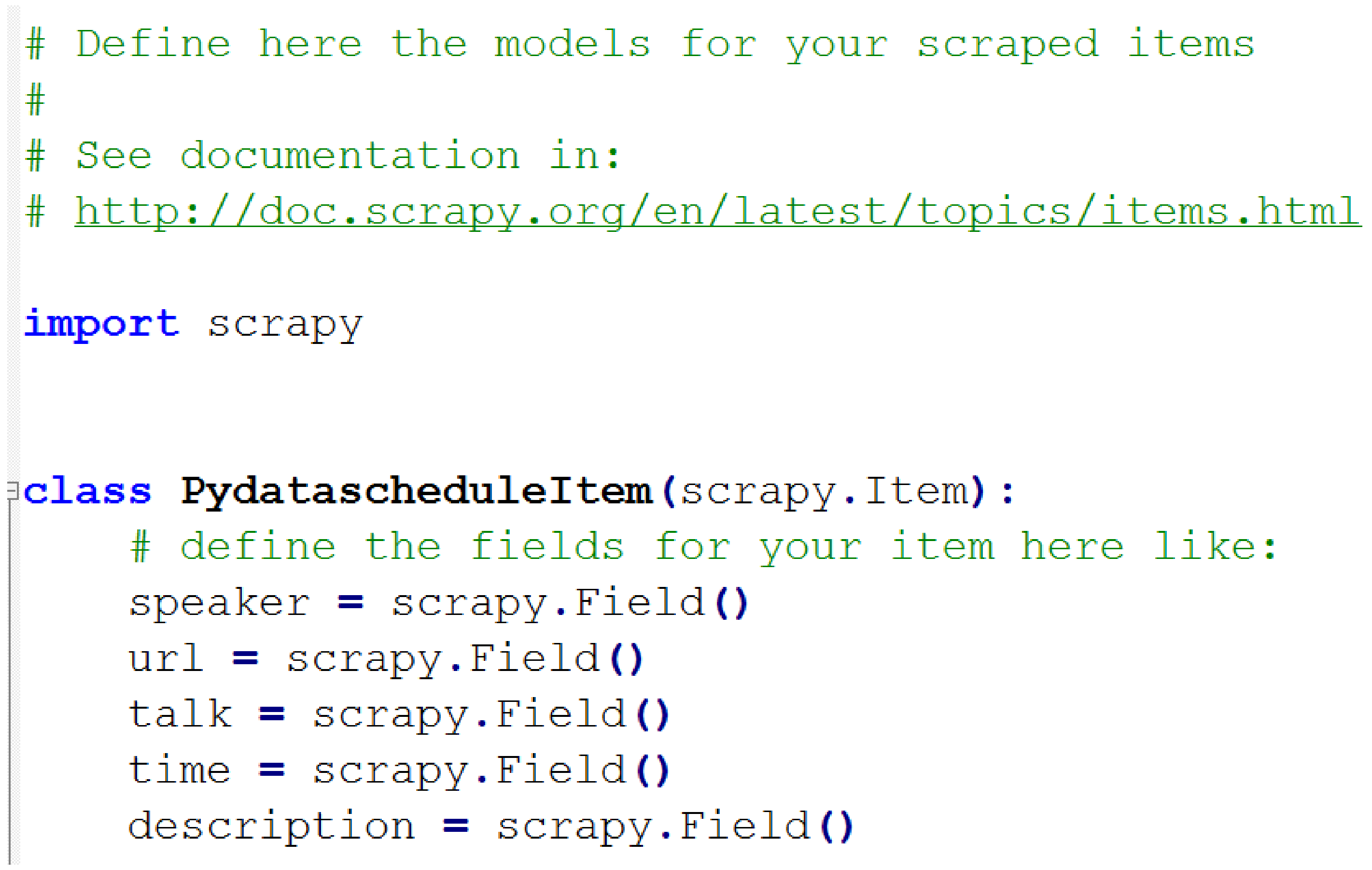
setting.py : the project's setting file.

spiders/ : spiders directory.



```
tutorial/
    scrapy.cfg
    tutorial/
        __init__.py
        items.py
        pipelines.py
        settings.py
        spiders/
            __init__.py
            ...
```

Pydata conferences



```python
# Define here the models for your scraped items
#
# See documentation in:
# http://doc.scrapy.org/en/latest/topics/items.html

import scrapy


class PydatascheduleItem(scrapy.Item):
    # define the fields for your item here like:
    speaker = scrapy.Field()
    url = scrapy.Field()
    talk = scrapy.Field()
    time = scrapy.Field()
    description = scrapy.Field()
```

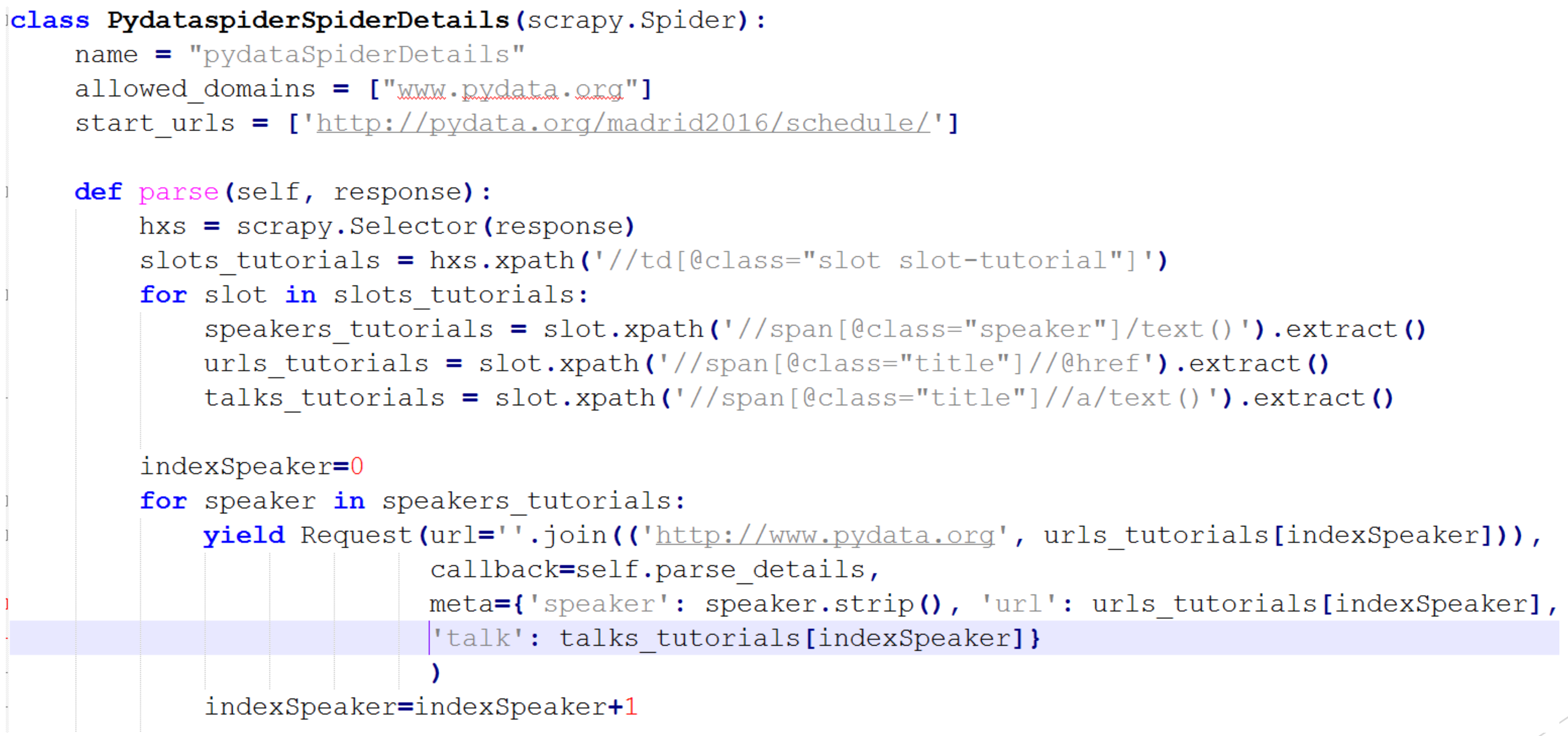
Spider generating

- **$** scrapy **genspider** -t basic <SPIDER_NAME> <DOMAIN>

Spiders list

- **$** scrapy list

Pydata spyder



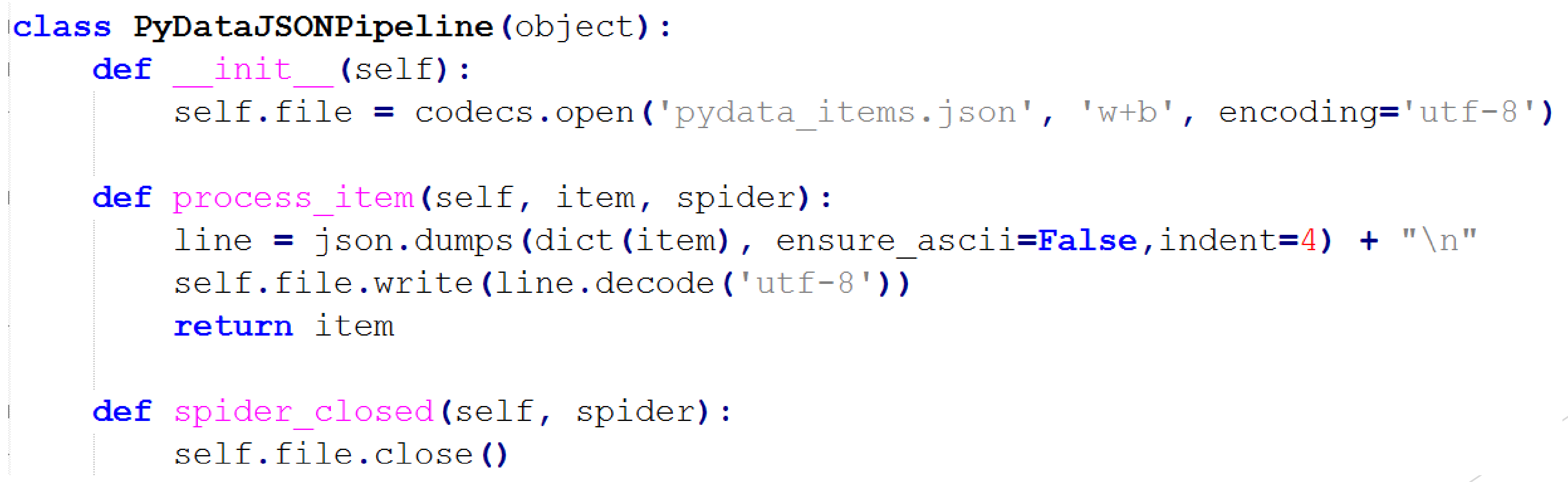
```
class PydataspiderSpiderDetails(scrapy.Spider):
    name = "pydataSpiderDetails"
    allowed_domains = ["www.pydata.org"]
    start_urls = ['http://pydata.org/madrid2016/schedule/']

    def parse(self, response):
        hxs = scrapy.Selector(response)
        slots_tutorials = hxs.xpath('//td[@class="slot slot-tutorial"]')
        for slot in slots_tutorials:
            speakers_tutorials = slot.xpath('//span[@class="speaker"]/text()').extract()
            urls_tutorials = slot.xpath('//span[@class="title"]//@href').extract()
            talks_tutorials = slot.xpath('//span[@class="title"]//a/text()').extract()

        indexSpeaker=0
        for speaker in speakers_tutorials:
            yield Request(url=''.join(('http://www.pydata.org', urls_tutorials[indexSpeaker])),
                          callback=self.parse_details,
                          meta={'speaker': speaker.strip(), 'url': urls_tutorials[indexSpeaker],
                          'talk': talks_tutorials[indexSpeaker]}
                          )
            indexSpeaker=indexSpeaker+1
```

Pydata sypder

```python
def parse_details(self, response):
    hxs = scrapy.Selector(response)
    item = PydatascheduleItem()
    item['speaker'] = response.meta['speaker'].encode('utf8')
    item['url'] = response.meta['url'].encode('utf8')
    item['talk'] = response.meta['talk'].encode('utf8')
    item['time'] = hxs.xpath('//div[@class="col-md-8"]/h4/text()').extract()[0].replace("\n","").strip()
    item['description'] = hxs.xpath('//div[@class="description"]/p/text()').extract()[0].encode('utf-8')
    return item
```

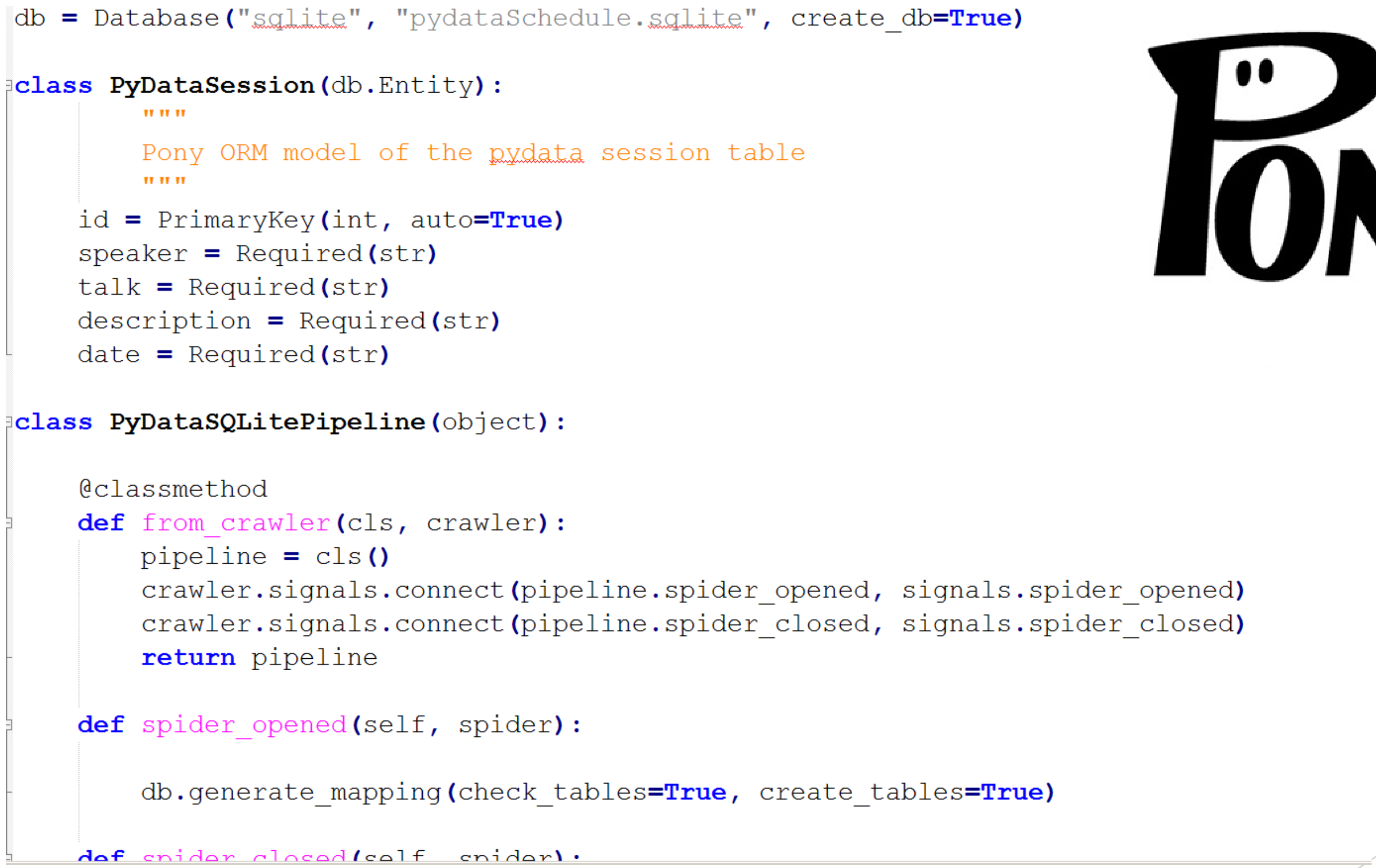
Pipelines

- ITEM_PIPELINES = {'pydataSchedule.pipelines.PyDataSQLitePipeline': 100, 'pydataSchedule.pipelines.PyDataJSONPipeline':200,}
- pipelines.py



```python
class PyDataJSONPipeline(object):
    def __init__(self):
        self.file = codecs.open('pydata_items.json', 'w+b', encoding='utf-8')

    def process_item(self, item, spider):
        line = json.dumps(dict(item), ensure_ascii=False,indent=4) + "\n"
        self.file.write(line.decode('utf-8'))
        return item

    def spider_closed(self, spider):
        self.file.close()
```

Pydata SQLitePipeline

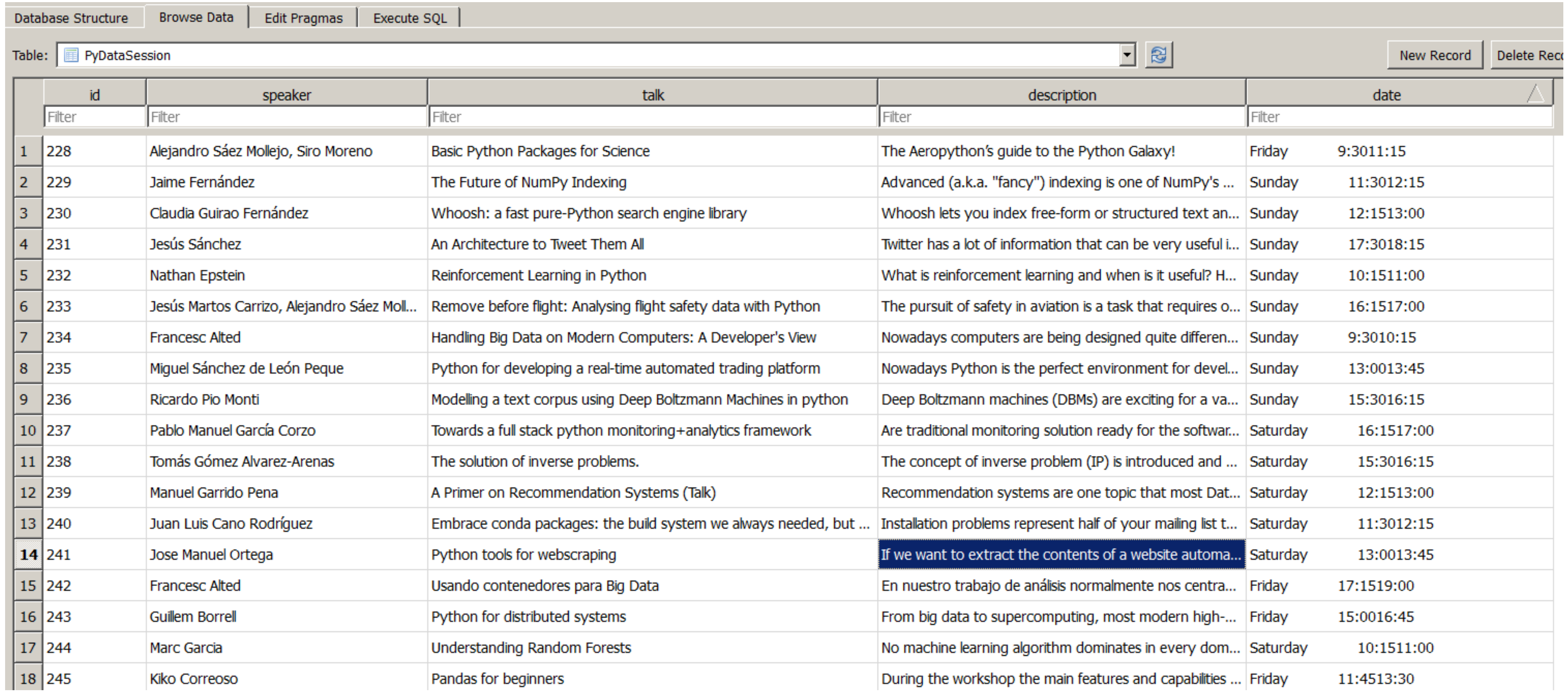


```
db = Database("sqlite", "pydataSchedule.sqlite", create_db=True)

class PyDataSession(db.Entity):
    """
    Pony ORM model of the pydata session table
    """
    id = PrimaryKey(int, auto=True)
    speaker = Required(str)
    talk = Required(str)
    description = Required(str)
    date = Required(str)

class PyDataSQLitePipeline(object):

    @classmethod
    def from_crawler(cls, crawler):
        pipeline = cls()
        crawler.signals.connect(pipeline.spider_opened, signals.spider_opened)
        crawler.signals.connect(pipeline.spider_closed, signals.spider_closed)
        return pipeline

    def spider_opened(self, spider):

        db.generate_mapping(check_tables=True, create_tables=True)

    def spider_closed(self, spider):
```

Execution

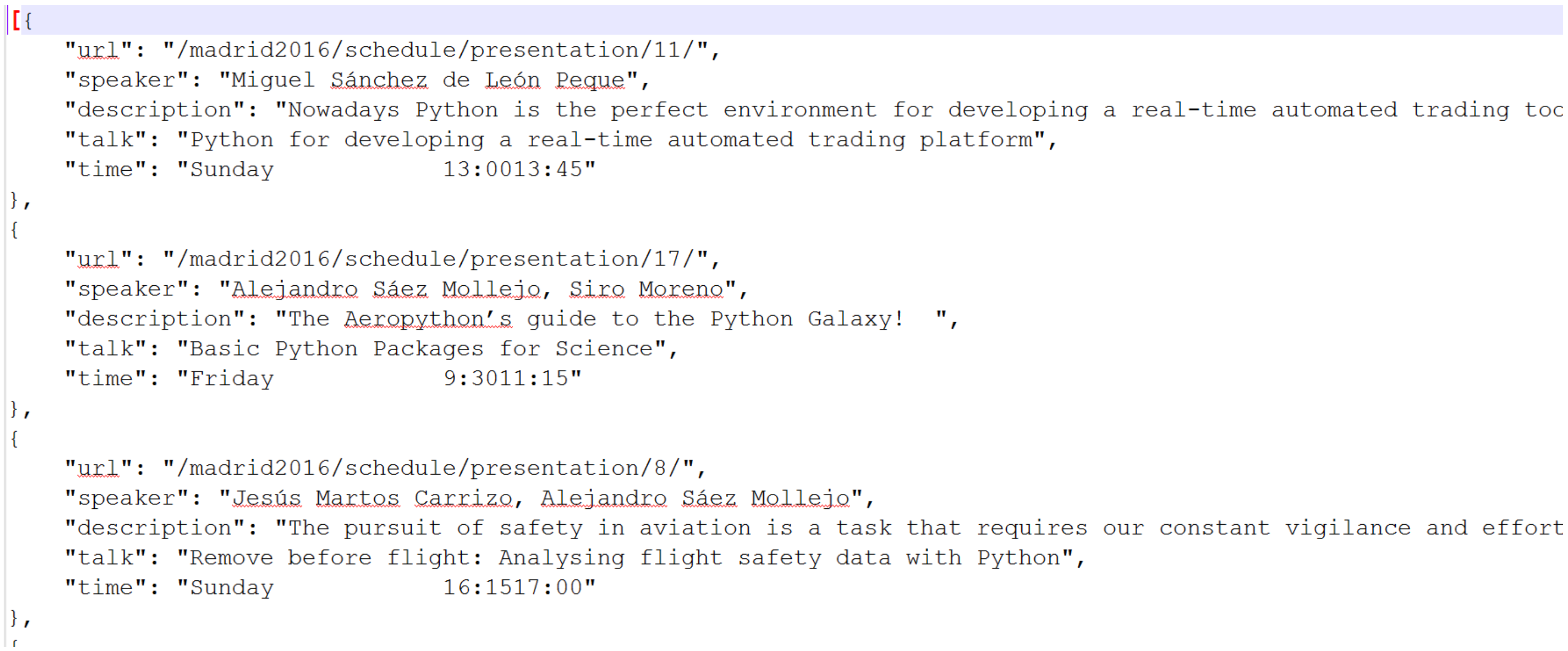
```
$   scrapy crawl <spider_name>
$   scrapy crawl <spider_name> -o items.json  -t json
$   scrapy crawl <spider_name> -o items.csv  -t csv
$   scrapy crawl <spider_name> -o items.xml  -t xml
```

Pydata conferences



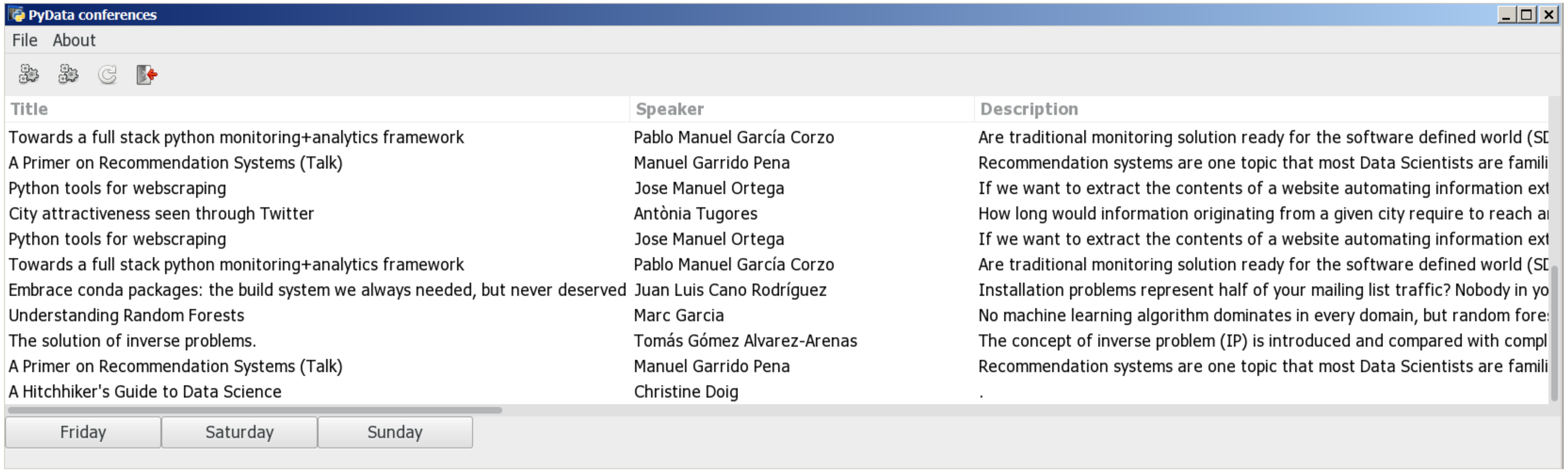
	id	speaker	talk	description	date
	Filter	Filter	Filter	Filter	Filter
1	228	Alejandro Sáez Mollejo, Siro Moreno	Basic Python Packages for Science	The Aeropython's guide to the Python Galaxy!	Friday 9:3011:15
2	229	Jaime Fernández	The Future of NumPy Indexing	Advanced (a.k.a. "fancy") indexing is one of NumPy's ...	Sunday 11:3012:15
3	230	Claudia Guirao Fernández	Whoosh: a fast pure-Python search engine library	Whoosh lets you index free-form or structured text an...	Sunday 12:1513:00
4	231	Jesús Sánchez	An Architecture to Tweet Them All	Twitter has a lot of information that can be very useful i...	Sunday 17:3018:15
5	232	Nathan Epstein	Reinforcement Learning in Python	What is reinforcement learning and when is it useful? H...	Sunday 10:1511:00
6	233	Jesús Martos Carrizo, Alejandro Sáez Moll...	Remove before flight: Analysing flight safety data with Python	The pursuit of safety in aviation is a task that requires o...	Sunday 16:1517:00
7	234	Francesc Alted	Handling Big Data on Modern Computers: A Developer's View	Nowadays computers are being designed quite differen...	Sunday 9:3010:15
8	235	Miguel Sánchez de León Peque	Python for developing a real-time automated trading platform	Nowadays Python is the perfect environment for devel...	Sunday 13:0013:45
9	236	Ricardo Pio Monti	Modelling a text corpus using Deep Boltzmann Machines in python	Deep Boltzmann machines (DBMs) are exciting for a va...	Sunday 15:3016:15
10	237	Pablo Manuel García Corzo	Towards a full stack python monitoring+analytics framework	Are traditional monitoring solution ready for the softwar...	Saturday 16:1517:00
11	238	Tomás Gómez Alvarez-Arenas	The solution of inverse problems.	The concept of inverse problem (IP) is introduced and ...	Saturday 15:3016:15
12	239	Manuel Garrido Pena	A Primer on Recommendation Systems (Talk)	Recommendation systems are one topic that most Dat...	Saturday 12:1513:00
13	240	Juan Luis Cano Rodríguez	Embrace conda packages: the build system we always needed, but ...	Installation problems represent half of your mailing list t...	Saturday 11:3012:15
14	241	Jose Manuel Ortega	Python tools for webscraping	If we want to extract the contents of a website automa...	Saturday 13:0013:45
15	242	Francesc Alted	Usando contenedores para Big Data	En nuestro trabajo de análisis normalmente nos centra...	Friday 17:1519:00
16	243	Guillem Borrell	Python for distributed systems	From big data to supercomputing, most modern high-...	Friday 15:0016:45
17	244	Marc Garcia	Understanding Random Forests	No machine learning algorithm dominates in every dom...	Saturday 10:1511:00
18	245	Kiko Correoso	Pandas for beginners	During the workshop the main features and capabilities ...	Friday 11:4513:30

Pydata conferences {JSON}



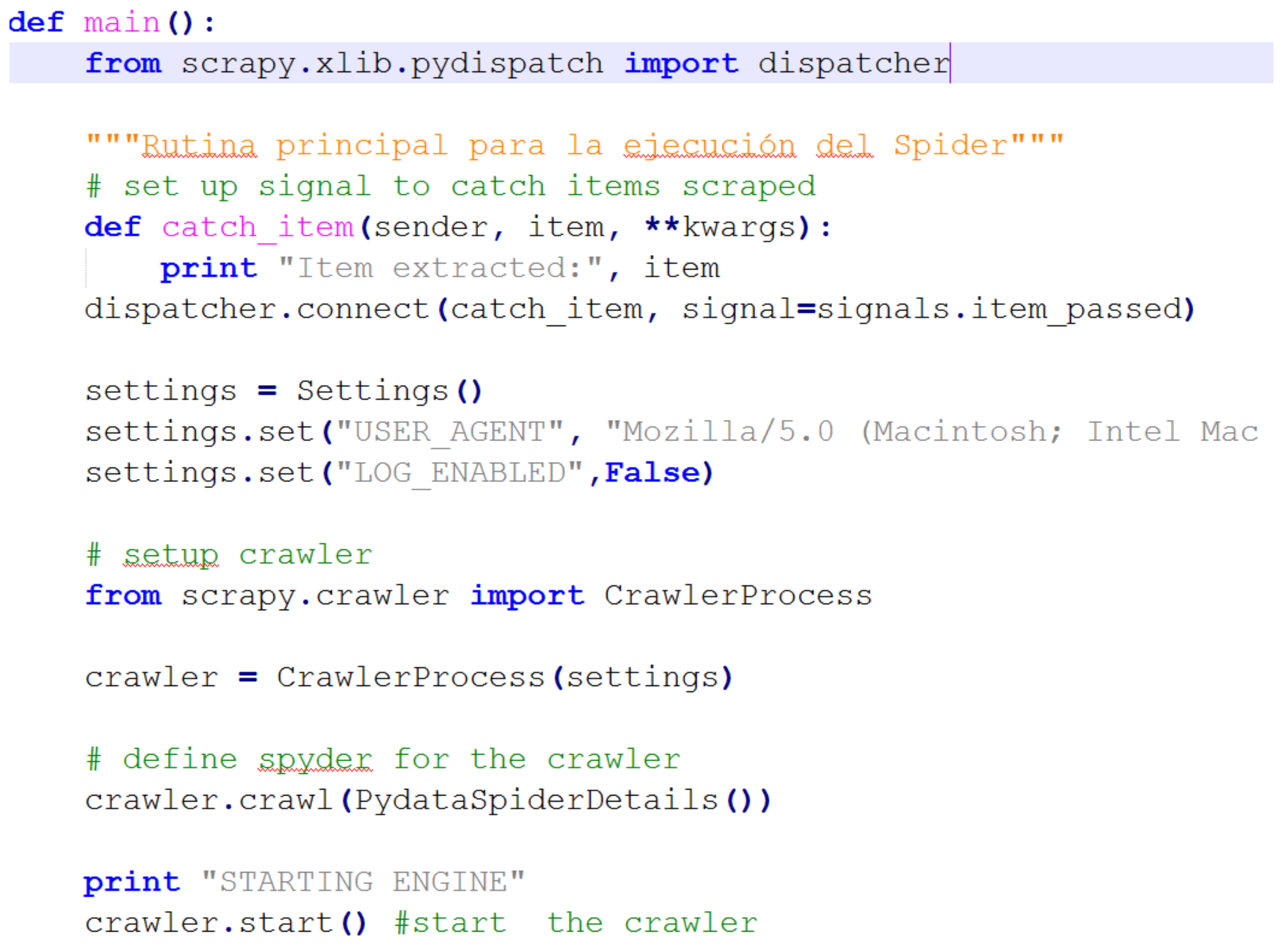
```
[{
    "url": "/madrid2016/schedule/presentation/11/",
    "speaker": "Miguel Sánchez de León Peque",
    "description": "Nowadays Python is the perfect environment for developing a real-time automated trading toc
    "talk": "Python for developing a real-time automated trading platform",
    "time": "Sunday            13:0013:45"
},
{
    "url": "/madrid2016/schedule/presentation/17/",
    "speaker": "Alejandro Sáez Mollejo, Siro Moreno",
    "description": "The Aeropython’s guide to the Python Galaxy!  ",
    "talk": "Basic Python Packages for Science",
    "time": "Friday            9:3011:15"
},
{
    "url": "/madrid2016/schedule/presentation/8/",
    "speaker": "Jesús Martos Carrizo, Alejandro Sáez Mollejo",
    "description": "The pursuit of safety in aviation is a task that requires our constant vigilance and effort
    "talk": "Remove before flight: Analysing flight safety data with Python",
    "time": "Sunday            16:1517:00"
},
```

Pydata conferences GTK



Title	Speaker	Description
Towards a full stack python monitoring+analytics framework	Pablo Manuel García Corzo	Are traditional monitoring solution ready for the software defined world (SD
A Primer on Recommendation Systems (Talk)	Manuel Garrido Pena	Recommendation systems are one topic that most Data Scientists are famili
Python tools for webscraping	Jose Manuel Ortega	If we want to extract the contents of a website automating information ext
City attractiveness seen through Twitter	Antònia Tugores	How long would information originating from a given city require to reach a
Python tools for webscraping	Jose Manuel Ortega	If we want to extract the contents of a website automating information ext
Towards a full stack python monitoring+analytics framework	Pablo Manuel García Corzo	Are traditional monitoring solution ready for the software defined world (SD
Embrace conda packages: the build system we always needed, but never deserved	Juan Luis Cano Rodríguez	Installation problems represent half of your mailing list traffic? Nobody in yo
Understanding Random Forests	Marc Garcia	No machine learning algorithm dominates in every domain, but random fore
The solution of inverse problems.	Tomás Gómez Alvarez-Arenas	The concept of inverse problem (IP) is introduced and compared with compl
A Primer on Recommendation Systems (Talk)	Manuel Garrido Pena	Recommendation systems are one topic that most Data Scientists are famili
A Hitchhiker's Guide to Data Science	Christine Doig	.

Launch spiders without scrapy command



```
def main():
    from scrapy.xlib.pydispatch import dispatcher

    """Rutina principal para la ejecución del Spider"""
    # set up signal to catch items scraped
    def catch_item(sender, item, **kwargs):
        print "Item extracted:", item
    dispatcher.connect(catch_item, signal=signals.item_passed)

    settings = Settings()
    settings.set("USER_AGENT", "Mozilla/5.0 (Macintosh; Intel Mac
    settings.set("LOG_ENABLED",False)

    # setup crawler
    from scrapy.crawler import CrawlerProcess

    crawler = CrawlerProcess(settings)

    # define spyder for the crawler
    crawler.crawl(PydataSpiderDetails())

    print "STARTING ENGINE"
    crawler.start() #start  the crawler
```

Scrapy Cloud

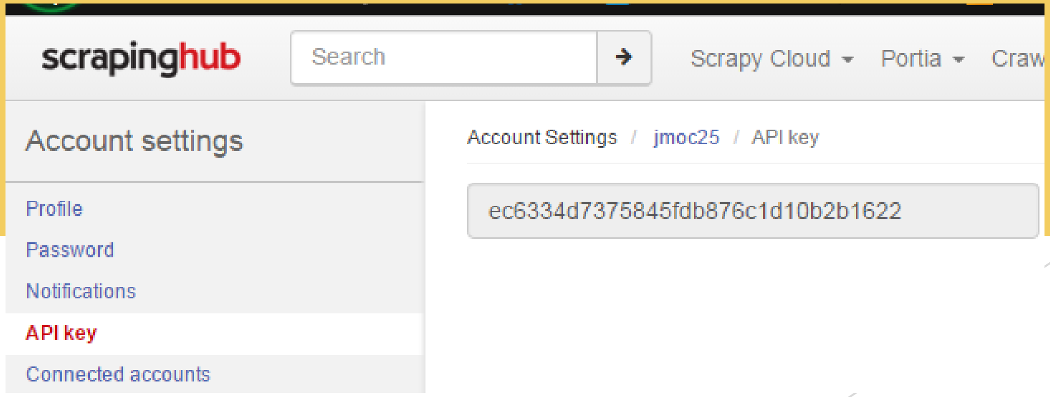
http://doc.scrapinghub.com/scrapy-cloud.html

https://dash.scrapinghub.com

```
>>pip install shub
>>shub login
        >>Insert your ScrapingHub API Key:
```

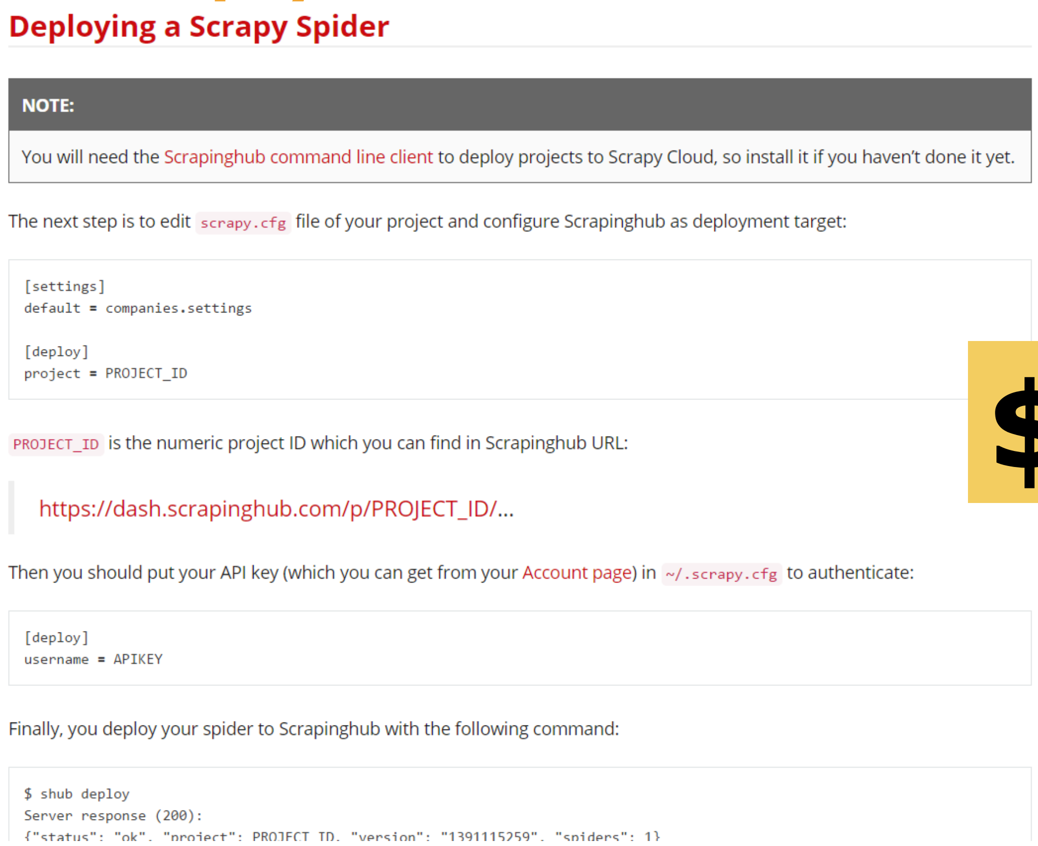
Scrapy Cloud /scrapy.cfg

```
# Project: demo
[deploy]
url =https://dash.scrapinghub.com/api/scrapyd/
#API_KEY
username = ec6334d7375845fdb876c1d10b2b1622
password =
#project identifier
project = 25767
```



ec6334d7375845fdb876c1d10b2b1622

Scrapy Cloud



Deploying a Scrapy Spider

NOTE:

You will need the Scrapinghub command line client to deploy projects to Scrapy Cloud, so install it if you haven't done it yet.

The next step is to edit `scrapy.cfg` file of your project and configure Scrapinghub as deployment target:

```
[settings]
default = companies.settings

[deploy]
project = PROJECT_ID
```

`PROJECT_ID` is the numeric project ID which you can find in Scrapinghub URL:

> https://dash.scrapinghub.com/p/PROJECT_ID/...

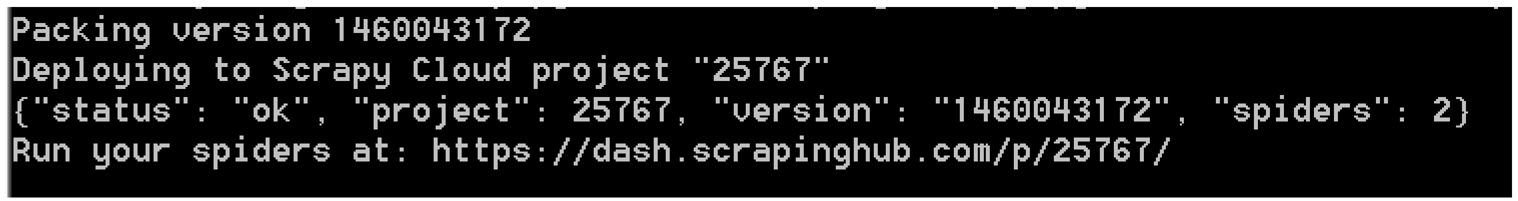
Then you should put your API key (which you can get from your Account page) in `~/.scrapy.cfg` to authenticate:

```
[deploy]
username = APIKEY
```

Finally, you deploy your spider to Scrapinghub with the following command:

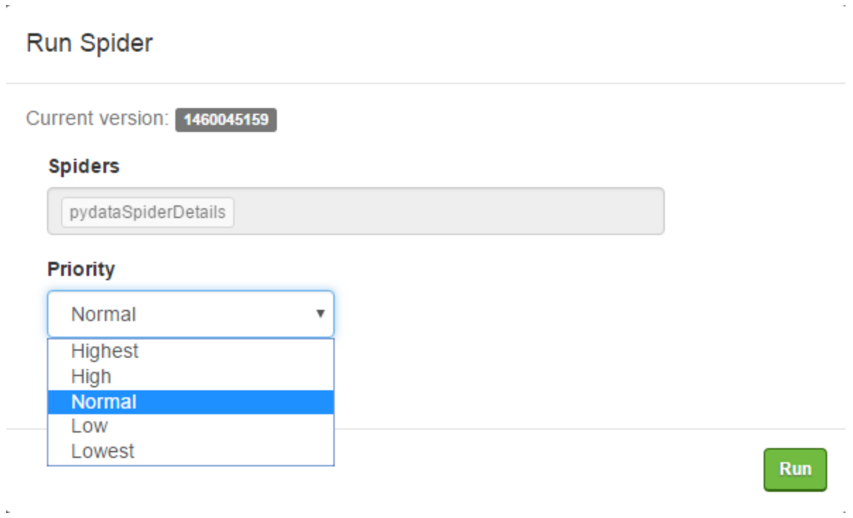
```
$ shub deploy
Server response (200):
{"status": "ok", "project": PROJECT_ID, "version": "1391115259", "spiders": 1}
```

$ shub deploy



```
Packing version 1460043172
Deploying to Scrapy Cloud project "25767"
{"status": "ok", "project": 25767, "version": "1460043172", "spiders": 2}
Run your spiders at: https://dash.scrapinghub.com/p/25767/
```

Scrapy Cloud



Run Spider

Current version: 1460045159

Spiders

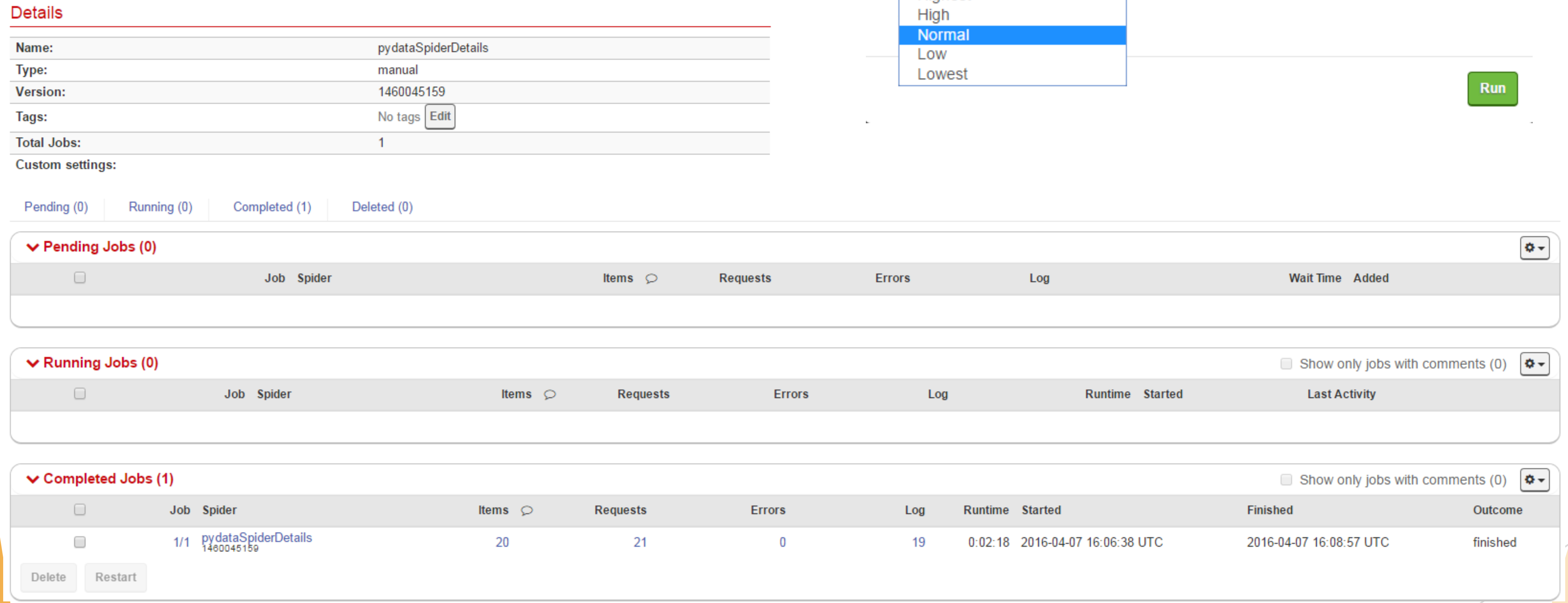
pydataSpiderDetails

Priority

Normal

- Highest
- High
- Normal
- Low
- Lowest

Run



Details

Name:	pydataSpiderDetails
Type:	manual
Version:	1460045159
Tags:	No tags Edit
Total Jobs:	1
Custom settings:	

Pending (0) Running (0) Completed (1) Deleted (0)

Pending Jobs (0)

Job	Spider	Items	Requests	Errors	Log	Wait Time	Added

Running Jobs (0) Show only jobs with comments (0)

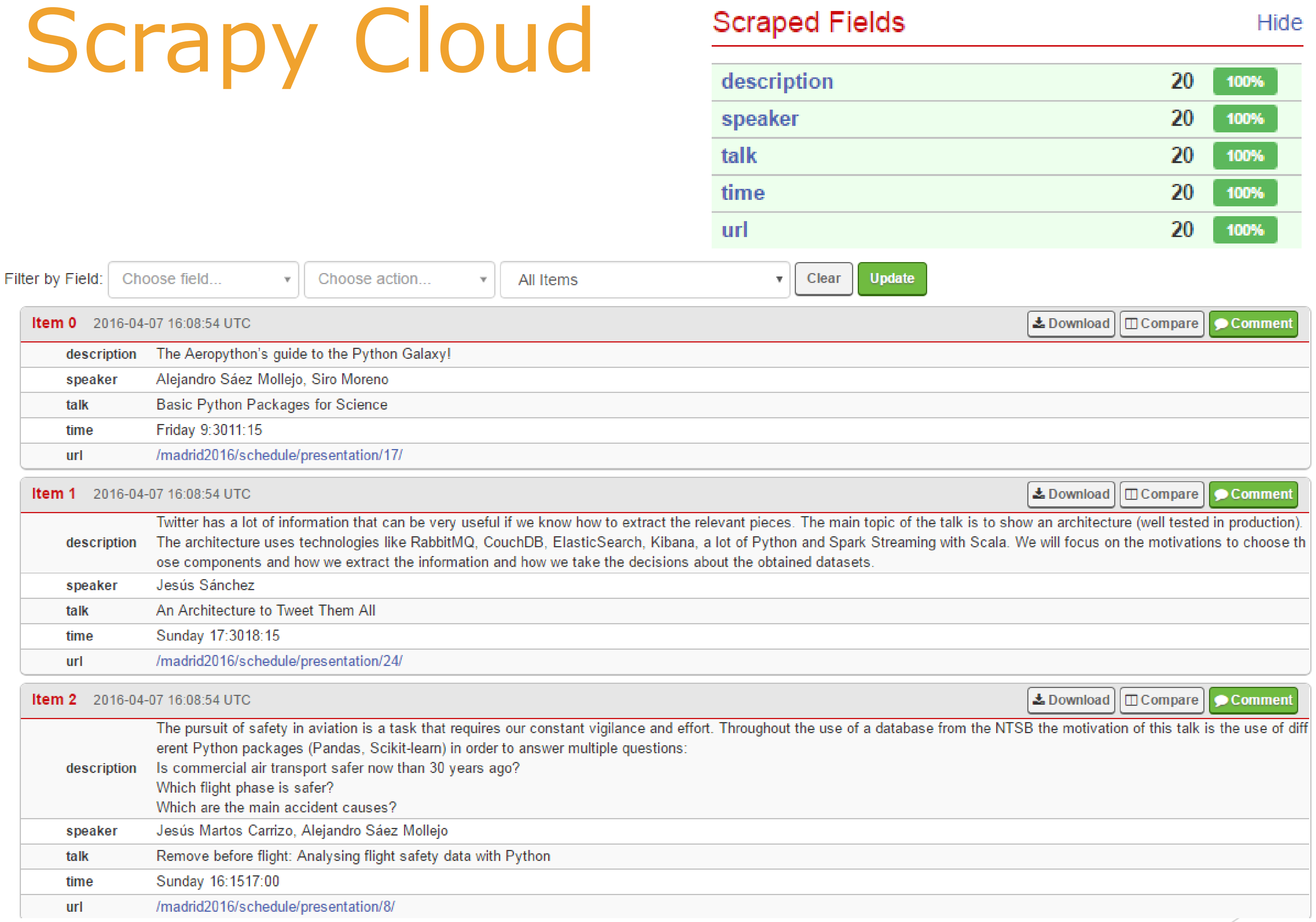
Job	Spider	Items	Requests	Errors	Log	Runtime	Started	Last Activity

Completed Jobs (1) Show only jobs with comments (0)

Job	Spider	Items	Requests	Errors	Log	Runtime	Started	Finished	Outcome
1/1	pydataSpiderDetails 1460045159	20	21	0	19	0:02:18	2016-04-07 16:06:38 UTC	2016-04-07 16:08:57 UTC	finished

Delete Restart

Scrapy Cloud



Scraped Fields   Hide

description	20	100%
speaker	20	100%
talk	20	100%
time	20	100%
url	20	100%

Filter by Field: Choose field... | Choose action... | All Items | Clear | Update

Item 0 2016-04-07 16:08:54 UTC Download Compare Comment

description	The Aeropython's guide to the Python Galaxy!
speaker	Alejandro Sáez Mollejo, Siro Moreno
talk	Basic Python Packages for Science
time	Friday 9:3011:15
url	/madrid2016/schedule/presentation/17/

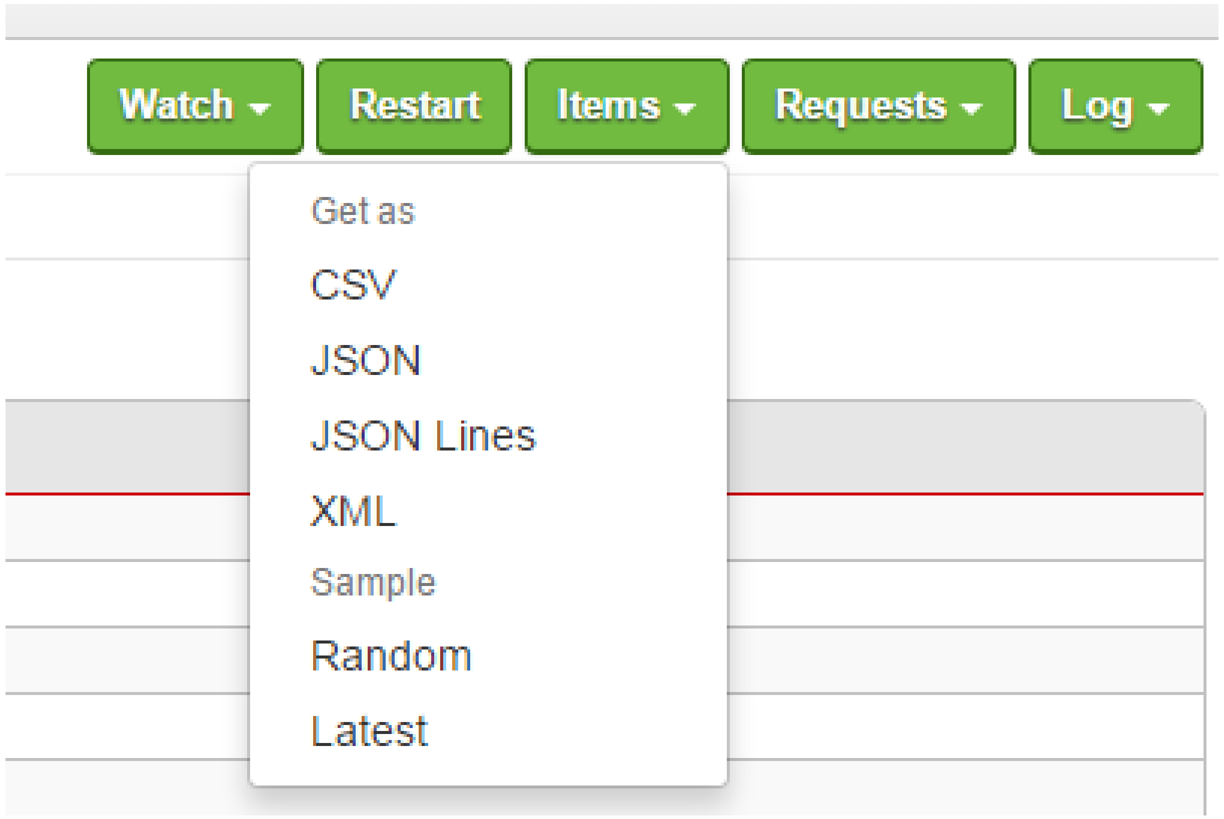
Item 1 2016-04-07 16:08:54 UTC Download Compare Comment

description	Twitter has a lot of information that can be very useful if we know how to extract the relevant pieces. The main topic of the talk is to show an architecture (well tested in production). The architecture uses technologies like RabbitMQ, CouchDB, ElasticSearch, Kibana, a lot of Python and Spark Streaming with Scala. We will focus on the motivations to choose th ose components and how we extract the information and how we take the decisions about the obtained datasets.
speaker	Jesús Sánchez
talk	An Architecture to Tweet Them All
time	Sunday 17:3018:15
url	/madrid2016/schedule/presentation/24/

Item 2 2016-04-07 16:08:54 UTC Download Compare Comment

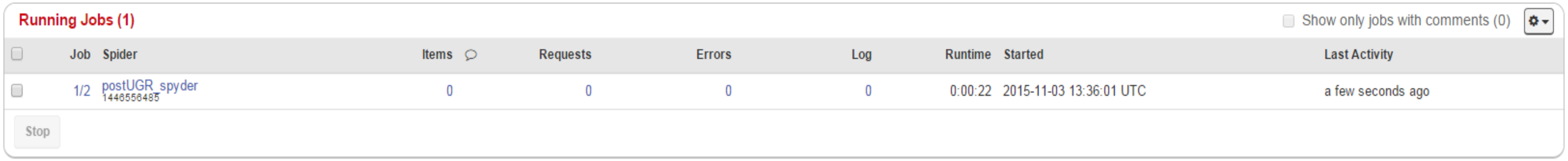
description	The pursuit of safety in aviation is a task that requires our constant vigilance and effort. Throughout the use of a database from the NTSB the motivation of this talk is the use of diff erent Python packages (Pandas, Scikit-learn) in order to answer multiple questions: Is commercial air transport safer now than 30 years ago? Which flight phase is safer? Which are the main accident causes?
speaker	Jesús Martos Carrizo, Alejandro Sáez Mollejo
talk	Remove before flight: Analysing flight safety data with Python
time	Sunday 16:1517:00
url	/madrid2016/schedule/presentation/8/

Scrapy Cloud



Scrapy Cloud Scheduling

```
curl -u APIKEY:
https://dash.scrapinghub.com/api/schedule.json -d
project=PROJECT -d spider=SPIDER
```



Running Jobs (1)

Show only jobs with comments (0)

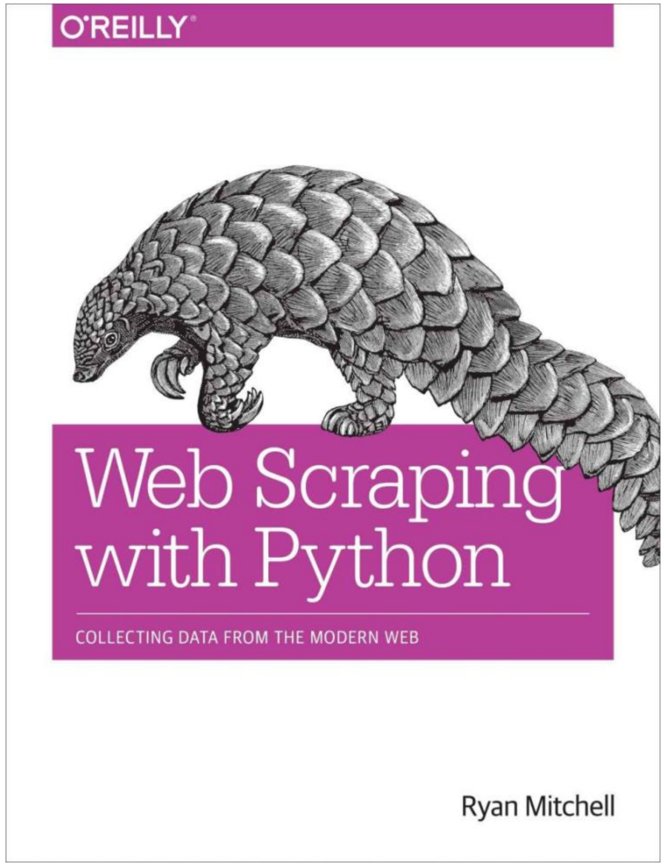
	Job	Spider	Items	Requests	Errors	Log	Runtime	Started	Last Activity
	1/2	postUGR_spyder 1446556485	0	0	0	0	0:00:22	2015-11-03 13:36:01 UTC	a few seconds ago

Stop

References

- http://www.crummy.com/software/BeautifulSoup
- http://scrapy.org
- https://pypi.python.org/pypi/mechanize
- http://docs.webscraping.com
- http://docs.python-requests.org/en/latest
- http://selenium-python.readthedocs.org/index.html
- https://github.com/REMitchell/python-scraping

Books

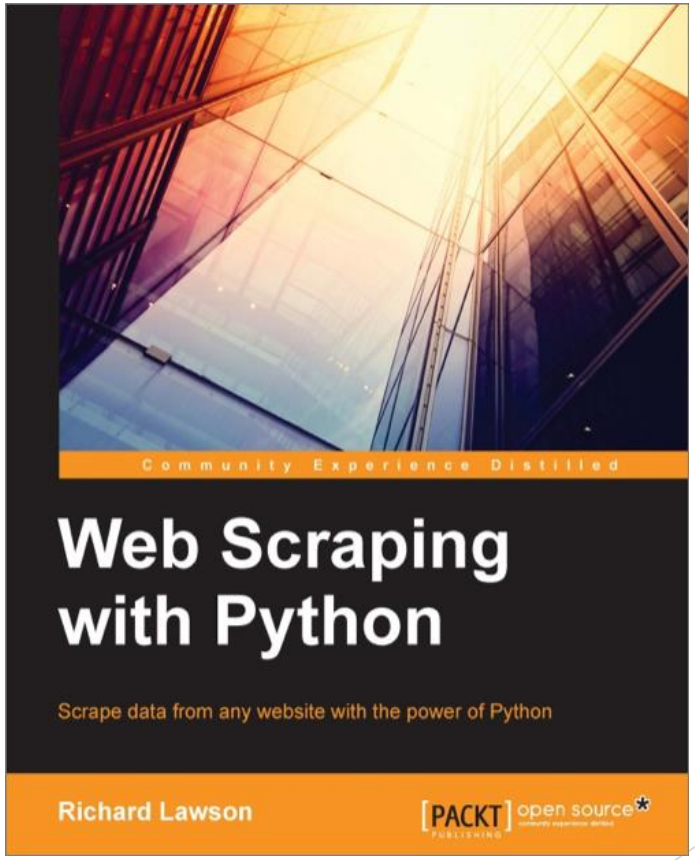


O'REILLY

Web Scraping with Python

COLLECTING DATA FROM THE MODERN WEB

Ryan Mitchell



Community Experience Distilled

Web Scraping with Python

Scrape data from any website with the power of Python

Richard Lawson

[PACKT] open source*

Thank you!