Programació per a *Data Science*

Unitat 5: Adquisició de dades en Python

Instruccions d'ús

A continuació es presentaran explicacions i exemples d'adquisició de dades en Python. Recordeu que podeu anar executant els exemples per obtenir-ne els resultats.

Introducció

Els processos d'adquisició de dades són molt diversos. En aquesta unitat, veurem exemples d'adquisició de dades d'internet amb tres mètodes diferents:

- descàrrega directa
- petició a APIs de tercers
- *web crawling*

Pel que respecta a la interacció amb APIs de tercers, repassarem dues alternatives, la construcció manual de les peticions HTTP i l'ús de llibreries Python.

En relació amb el *web crawling*, veurem com utilitzar la llibreria Scrapy (https://scrapy.org/) per construir un petit *web crawler* que capturi dades del nostre interès.

Primers passos

En aquesta unitat treballarem diverses vegades amb dades en format JSON (recordeu que ja hem introduït el format JSON a la xwiki).

La llibreria json de Python ens ofereix algunes funcions molt útils per a treballar en aquest format. Per exemple, podem obtenir la representació JSON d'objectes Python o crear objectes Python a partir de la seva representació en JSON.

In [1]:
```
# Construïm un diccionari d'exemple i mostrem el tipus de dades i el contingut de la variable.
diccionario_ejemplo = {"nombre": "Yann", "apellidos": {"apellido1": "LeCun", "apellido2": "-"}, "edad": 56}
print(type(diccionario_ejemplo))
print(diccionario_ejemplo)

# Construïm una llista d'exemple i mostrem el tipus de dades i el contingut de la variable.
lista_ejemplo = [1, 2, 3]
print(type(lista_ejemplo))
print(lista_ejemplo)
```

```
<type 'dict'>
{'apellidos': {'apellido2': '-', 'apellido1': 'LeCun'}, 'nombre': 'Yann', 'edad': 56}
<type 'list'>
[1, 2, 3]
```

In [2]:
```
# Importem la llibreria json.
import json

# Mostrem la representació JSON del diccionari.
json_dict = json.dumps(diccionario_ejemplo)
print type(json_dict)
print json_dict

# Mostrem la representació JSON de la llista.
json_list = json.dumps(lista_ejemplo)
print type(json_list)
print json_list
```

```
<type 'str'>
{"apellidos": {"apellido2": "-", "apellido1": "LeCun"}, "nombre": "Yann", "edad": 56}
<type 'str'>
[1, 2, 3]
```

Fixeu-vos que, en ambdós casos, obtenim una cadena de caràcters que ens representa, en format JSON, els objectes Python. Aquest procés es coneix com a **serialitzar** l'objecte.

També podem fer el procés invers (conegut com a **desserialitzar**), creant objectes Python (per exemple, llistes o diccionaris) a partir de cadenes de text en format JSON.

In [3]:
```
# Desserialitzem la cadena 'json_dict'.
diccionario_ejemplo2 = json.loads(json_dict)
print(type(diccionario_ejemplo2))
print(diccionario_ejemplo2)

# Desserialitzem la cadena 'json_list'.
lista_ejemplo2 = json.loads(json_list)
print(type(lista_ejemplo2))
print(lista_ejemplo2)
```

```
<type 'dict'>
{u'apellidos': {u'apellido2': u'-', u'apellido1': u'LeCun'}, u'nombre': u'Yann', u'edad': 56}
<type 'list'>
[1, 2, 3]
```

Per millorar la llegibilitat de les dades que obtindrem de les APIs, definirem una funció que mostrarà cadenes JSON per pantalla formatades per millorar-ne la lectura. La funció acceptarà tant cadenes de caràcters amb contingut JSON com objectes Python, i mostrarà el contingut per pantalla.

A més, la funció rebrà un paràmetre opcional que ens permetrà indicar el nombre màxim de línies que cal mostrar. Així, podrem fer servir la funció per a visualitzar les primeres línies d'un JSON llarg, sense haver de mostrar el JSON complet per pantalla.

In [4]:
```
# Defineix la funció 'json_print', que té un paràmetre obligatori 'json_data' i un paràmetre opcional 'limit'
# i no torna cap valor.
# La funció mostra per pantalla el contingut de la variable 'json_data' en format JSON, limitant el nombre
# de línies per mostrar si s'inclou el paràmetre 'limit'.
def json_print (json_data, limit = None):
    if isinstance(json_data, (str, unicode)):
        json_data = json.loads(json_data)
    nice = json.dumps(json_data, sort_keys=True, indent=3, separators=(',', ': '))
    print "\n".join(nice.split("\n")[0:limit])
    if limit is not None:
        print "[...]"
```

Vegem un exemple del resultat d'utilitzar la funció que acabem de definir.

In [5]:
```
# Mostra el valor de la variable 'json_exemple' amb la funció 'print'.
json_exemple = '{"nombre": "Yann", "apellidos": {"apellido1": "LeCun", "apellido2": "-"}, "edad": 56}'
print json_exemple
```

```
{"nombre": "Yann", "apellidos": {"apellido1": "LeCun", "apellido2": "-"}, "edad": 56}
```

In [6]:
```
# Mostra el valor de la variable 'json_exemple' amb la funció 'json_print' que acabem de definir.
json_print (json_exemple)
```

```
{
   "apellidos": {
      "apellido1": "LeCun",
      "apellido2": "-"
   },
   "edad": 56,
   "nombre": "Yann"
}
```

In [7]:
```
# Mostrem únicament les tres primeres línies.
json_print (json_exemple, 3)
```

```
{
   "apellidos": {
      "apellido1": "LeCun",
[...]
```

Descàrrega directa de dades

La descàrrega directa del conjunt de dades és potser el mètode més senzill d'adquisició de dades i consisteix a descarregar un fitxer amb les dades d'interès ja recopilades per algun altre analista. De fet, a la unitat anterior ja hem fet servir aquest mètode per adquirir el fitxer amb les dades sobre els personatges de còmic de Marvel. Un cop descarregat el fitxer, el procediment per carregar-lo en Python dependrà del format concret (ja hem vist un exemple de càrrega de dades des d'un fitxer .csv).

Alguns dels llocs web on podeu trobar conjunts de dades a analitzar són:

- Open Data gencat (http://dadesobertes.gencat.cat/en/), el portal de dades obertes de la Generalitat.
- datos.gov.es (http://datos.gob.es/es/catalogo), el catàleg de conjunts de dades del Govern d'Espanya.
- European Data Sources (https://data.europa.eu/), el portal de dades obertes de la Unió Europea.
- Mark Newman network datasets (http://www-personal.umich.edu/~mejn/netdata/), conjunts de dades en forma de xarxa recopilats per Mark Newman.
- Stanford Large Network Dataset Collection (http://snap.stanford.edu/data/), un altre recopilatori de conjunts de dades en forma de xarxa, en aquest cas creat per Jure Leskovec.
- SecRepo.com (http://www.secrepo.com/), dades relacionades amb la seguretat.
- AWS Public Datasets (https://aws.amazon.com/public-datasets/), conjunts de dades recopilades i hostatjades per Amazon.
- UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/), dades recopilades per un grup de recerca de la Universitat de Califòrnia, Irvine.
- El repositori de Five Thirty Eight (https://github.com/fivethirtyeight), que recull dades utilitzades a articles de la publicació i que ja hem vist a la unitat anterior.

Ús d'API de tercers

Accés a API manualment

Podem utilitzar la llibreria de Python Requests (http://docs.python-requests.org/) per a realitzar peticions als webs API de manera manual. Per fer-ho, haurem d'accedir a la documentació de l'API amb la qual vulguem actuar, construir manualment les peticions per obtenir la informació desitjada i processar també manualment la resposta rebuda.

Vegem un exemple de petició HTTP a una API pública. El lloc http://postcodes.io/ (http://postcodes.io/) ofereix una API de geolocalització sobre codis postals al Regne Unit. Llegint la documentació, podem veure que té un mètode GET amb la URL http://api.postcodes.io/postcodes/:código-postal (http://api.postcodes.io/postcodes/:código-postal), que ens retorna informació del codi postal especificat.

In [8]:
```
# Importem la llibreria.
import requests

# Fem una petició get a l'API, preguntant sobre el codi postal "E98 1TT".
# Fixeu-vos que el caràcter espai es codifica com a% 20 a la URL.
response = requests.get('http://api.postcodes.io/postcodes/E98%201TT')

# Mostrem la resposta rebuda.
print "Codi d'estat de la resposta: ", response.status_code, "\n"
print "Capçalera de la resposta: "
json_print(dict(response.headers))
print "\nCos de la resposta: "
json_print(response.content)
```

```
Codi d'estat de la resposta:  200

Capçalera de la resposta:
{
   "Access-Control-Allow-Origin": "*",
   "Connection": "keep-alive",
   "Content-Length": "793",
   "Content-Type": "application/json; charset=utf-8",
   "Date": "Mon, 16 Oct 2017 14:56:54 GMT",
   "ETag": "W/\"319-6bYi9c4AZnAn0WJknW59sYe5nfk\"",
   "Server": "nginx/1.13.5",
   "X-GNU": "Michael J Blanchard"
}

Cos de la resposta:
{
   "result": {
      "admin_county": null,
      "admin_district": "Tower Hamlets",
      "admin_ward": "St Katharine's & Wapping",
      "ccg": "NHS Tower Hamlets",
      "codes": {
         "admin_county": "E99999999",
         "admin_district": "E09000030",
         "admin_ward": "E05009330",
         "ccg": "E38000186",
         "nuts": "UKI42",
         "parish": "E43000220",
         "parliamentary_constituency": "E14000882"
      },
      "country": "England",
      "eastings": 534427,
      "european_electoral_region": "London",
      "incode": "1TT",
      "latitude": 51.5080245566444,
      "longitude": -0.0643935343625153,
      "lsoa": "Tower Hamlets 026B",
      "msoa": "Tower Hamlets 026",
      "nhs_ha": "London",
      "northings": 180564,
      "nuts": "Tower Hamlets",
      "outcode": "E98",
      "parish": "Tower Hamlets, unparished area",
      "parliamentary_constituency": "Poplar and Limehouse",
      "postcode": "E98 1TT",
      "primary_care_trust": "Tower Hamlets",
      "quality": 1,
      "region": "London"
   },
   "status": 200
}
```

Com podem veure, l'estat de la resposta és 200, la qual cosa ens indica (https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html) que la petició s'ha processat correctament. Entre d'altres camps, la capçalera de la resposta inclou el tipus de contingut que trobarem al cos, que serà un text en format JSON. Finalment, el cos de la resposta inclou dades sobre el codi postal consultat. Per exemple, podem veure que correspon a la nació d'Anglaterra (concretament, a la ciutat de Londres).

Fixeu-vos que podem visualitzar també la resposta accedint a la mateixa URL (http://api.postcodes.io/postcodes/E98%201TT) amb un navegador web. En aquest cas, es poden instal·lar extensions específiques que gestionin la visualització millorada del JSON retornat (per exemple, JSONView (https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc) per Chrome o Firefox).

Accés a API amb llibreries de Python

Encara que podríem fer servir aquest mètode per interactuar amb qualsevol API HTTP, la veritat és que quan la complexitat de les funcions disponibles incrementa (per exemple, en incloure autenticació) pot no resultar gaire pràctic. Quan vulguem accedir a APIs populars, normalment trobarem que ja existeixen llibreries de Python dissenyades per interactuar amb aquestes API, de manera que podrem obtenir dades sense necessitat de gestionar les peticions HTTP manualment.

Google maps disposa d'un conjunt d'API (https://developers.google.com/maps/) molt populars que permeten, entre d'altres, obtenir les coordenades geogràfiques d'una adreça, aconseguir indicacions per desplaçar-se d'un punt a un altre, o adquirir dades sobre l'elevació del terreny a qualsevol punt del món. La llibreria googlemaps (https://googlemaps.github.io/google-maps-services-python/docs/2.4.6/) integra peticions a l'API de Google en codi Python.

Per fer servir les APIs de Google Maps, cal registrar un usuari i obtenir una clau d'autenticació, que adjuntarem a les peticions que es facin contra l'API. A més, haurem d'especificar quines APIs concretes farem servir.

A l'exemple següent, farem aquests tres passos per obtenir la clau d'autenticació:

1. Crearem un projecte a la plataforma de Google Developers.
2. Activarem les APIs desitjades.
3. Sol·licitarem credencials d'accés.

En primer lloc crearem un nou projecte a l'entorn de desenvolupadors de google. Ens dirigirem a: https://console.developers.google.com/apis/library (https://console.developers.google.com/apis/library) i farem clic sobre «Project: New project». Assignarem un nom qualsevol al projecte i confirmarem la creació clicant sobre «Create».

Un cop creat el projecte, activarem les APIs que farem servir. Primer, seleccionarem l'API de geocodificació (*Google Maps Geocoding API* (https://console.developers.google.com/apis/api/geocoding_backend)), que es troba a la categoria *Google Maps APIs* (és possible que hagueu de prémer sobre el botó «more» per veure la llista completa d' APIs). Farem clic sobre «Enable» per activar-la.

Repetirem el procés per a l'API d'adreces (*Google Maps Directions API* (https://console.developers.google.com/apis/api/directions_backend)), que es troba també a la categoria *Google Maps APIs*.

Finalment, farem clic sobre el menú «Credentials», indicarem «Create credentials» i escollirem «API Key». Ens apareixerà una finestra amb una cadena de caràcters que representa la nostra clau. Perquè l'exemple següent funcioni, **cal que assigneu a la variable `api_key` el valor de la vostra clau**.

In [9]:
```
# Importem la llibreria googlemaps, que interactuarà amb l'API de Google Maps.
import googlemaps

# Importem la llibreria datetime, que ens ofereix funcions de maniobres de dates.
from datetime import datetime

################################################################################
# ATENCIÓ! Assigneu a la variable api_key la clau que hagueu obtingut de Google.
api_key = ""
################################################################################

# Inicialitzem el client, indicant la clau d'autenticació,
gmaps = googlemaps.Client(key=api_key)
```

En primer lloc, farem servir l'API de geocodificació (https://developers.google.com/maps/documentation/geocoding/start) per obtenir dades d'una adreça per mitjà del mètode Geocode (https://googlemaps.github.io/google-maps-services-python/docs/2.4.6/#googlemaps.Client.geocode) del client de Google Maps que ens ofereix la llibreria (emmagatzemat a la variable `gmaps`).

In [10]:
```
# Utilitzem l'API de geocodificació per obtenir dades d'una adreça.
geocode_result = gmaps.geocode('Rambla del Poblenou, 156, Barcelona')
print("------ Resultado de geocode ------")
json_print(geocode_result, 20)
```

```
------ Resultado de geocode ------
[
   {
      "address_components": [
         {
            "long_name": "156",
            "short_name": "156",
            "types": [
               "street_number"
            ]
         },
         {
            "long_name": "Rambla del Poblenou",
            "short_name": "Rambla del Poblenou",
            "types": [
               "route"
            ]
         },
         {
            "long_name": "Barcelona",
            "short_name": "Barcelona",
[...]
```

Un altre exemple de l'ús de l'API de geocodificació (https://developers.google.com/maps/documentation/geocoding/start) utilitza el mètode reverse_geocode (https://googlemaps.github.io/google-maps-services-python/docs/2.4.6/#googlemaps.Client.reverse_geocode) per obtenir informació sobre unes coordenades geogràfiques concretes:

In [11]:
```
# Obtenim dades sobre unes coordenades geogràfiques.
reverse_geocode_result = gmaps.reverse_geocode((41.2768089, 1.9884642))
print("------ Resultado de reverse geocode ------")
json_print(reverse_geocode_result, 20)
```

```
------ Resultado de reverse geocode ------
[
   {
      "address_components": [
         {
            "long_name": "17",
            "short_name": "17",
            "types": [
               "street_number"
            ]
         },
         {
            "long_name": "Avinguda del Canal Ol\u00edmpic",
            "short_name": "Av. del Canal Ol\u00edmpic",
            "types": [
               "route"
            ]
         },
         {
            "long_name": "Castelldefels",
            "short_name": "Castelldefels",
[...]
```

L'exemple següent interactua amb l'API d'adreces (https://developers.google.com/maps/documentation/directions/) fent servir el mètode *directions* (https://googlemaps.github.io/google-maps-services-python/docs/2.4.6/#googlemaps.Client.directions) de la llibreria googlemaps de Python, per obtenir indicacions de desplaçament entre dos punts.

In [12]:
```
# Obtenim indicacions sobre com anar d'una adreça a una altra, considerant el trànsit del moment actual.
now = datetime.now()
directions_result = gmaps.directions("Carrer Colom, 114, Terrassa",
                                     "Carrer Sant Antoni, 1, Salt",
                                     mode="transit",
                                     departure_time=now)
print("------ Resultado de directions ------")
json_print(directions_result, 15)
```

```
------ Resultado de directions ------
[
   {
      "bounds": {
         "northeast": {
            "lat": 41.98096779999999,
            "lng": 2.8171612
         },
         "southwest": {
            "lat": 41.3787125,
            "lng": 2.0075253
         }
      },
      "copyrights": "Map data \u00a92017 Google, Inst. Geogr. Nacional",
      "legs": [
         {
[...]
```

Fixeu-vos que, en aquest cas, no hem hagut de gestionar les peticions HTTP manualment: la llibreria ho ha fet per nosaltres de manera transparent.

A més, les funcions de la llibreria ens tornen directament objectes Python, que es poden fer servir com qualsevol altre. Per exemple, podem seleccionar només una part de les respostes de les API segons el nostre interès:

In [13]:
```
# Mostrem les claus del diccionari que retorna la crida a geocode.
geocode_result[0].keys()
```

Out[13]:
```
[u'geometry',
 u'address_components',
 u'place_id',
 u'formatted_address',
 u'types']
```

In [14]:
```
# Mostrem únicament les coordenades geogràfiques de la direcció d'interès.
geocode_result[0]["geometry"]["location"]
```

Out[14]:
```
{u'lat': 41.4063554, u'lng': 2.1947451}
```

In [15]:
```
# Mostrem les localitzacions properes a les coordenades geogràfiques que hem preguntat amb 'reverse_geocode',
# tot imprimint-ne les coordenades exactes i l'adreça.
for result in reverse_geocode_result:
    print result["geometry"]["location"], result["formatted_address"]
```

```
{u'lat': 41.2772149, u'lng': 1.9892062} Av. del Canal Olímpic, 17, 08860 Castelldefels, Barcelona, Spain
{u'lat': 41.2800161, u'lng': 1.9766294} Castelldefels, Barcelona, Spain
{u'lat': 41.2790599, u'lng': 1.9734743} Castelldefels, Barcelona, Spain
{u'lat': 41.2792267, u'lng': 1.9636914} 08860 Sitges, Barcelona, Spain
{u'lat': 41.3847492, u'lng': 1.949021} Baix Llobregat, Barcelona, Spain
{u'lat': 41.383401, u'lng': 2.027319} Barcelona Metropolitan Area, Barcelona, Spain
{u'lat': 41.3850477, u'lng': 2.1733131} Barcelona, Spain
{u'lat': 41.5911589, u'lng': 1.5208624} Catalonia, Spain
{u'lat': 40.46366700000001, u'lng': -3.74922} Spain
```

In [16]:
```
# Mostrem únicament la distància del trajecte entre els dos punts preguntats a l'API d'adreces.
print directions_result[0]["legs"][0]["distance"]
```

```
{u'text': u'125 km', u'value': 124706}
```

Capturant les dades manualment: *web crawling*

Scrapy (https://scrapy.org/) és una llibreria de Python que proveeix d'un *framework* per a l'extracció de dades de pàgines web. Scrapy és molt complet i disposa de múltiples funcionalitats, però en veurem un exemple senzill d'ús.

Suposeu que volem obtenir un llistat de les titulacions de grau que ofereix la UOC. La UOC no ofereix una API amb aquesta informació, però sí que podem trobar-la a la pàgina http://estudios.uoc.edu/es/grados (http://estudios.uoc.edu/es/grados). De totes maneres, no volem anar copiant manualment els noms de totes les titulacions per obtenir el llistat d'interès, per la qual cosa desenvoluparem un petit *crawler* que obtingui aquestes dades per nosaltres.

Ja tenim identificada la URL que volem explorar (http://estudios.uoc.edu/es/grados (http://estudios.uoc.edu/es/grados)), així que només caldrà identificar on es troben les dades d'interès dins de la pàgina. Per fer-ho, en primer lloc ens fixarem en algun títol de grau que aparegui a la pàgina, per exemple, "Diseño y Creación Digitales" o "Multimedia". Seguidament accedirem al codi font de la pàgina (podem fer servir la combinació de tecles `CTRL + u` als navegadors Firefox o Chrome) i buscarem els noms dels graus que hem vist anteriorment:

(/es/grados/diseño-creacion-digital/presentacion) (/es/grados/multimedia/presentacion)

Com es pot apreciar, les dades que volem recopilar (els noms de les titulacions de grau que ofereix la UOC) es troben a l'atribut títol (*title*) d'un enllaç (un element assenyalat amb l'etiqueta <a>) que té l'atribut classe fixat a «card-absolute-link».

Per a indicar que volem seleccionar aquestes dades, utilitzarem la sintaxi XPath. En concret, utilitzarem l'expressió

```
//a[@class="card-absolute-link"]/@title
```

que ens indica que volem seleccionar totes les etiquetes <a> que tinguin com a atribut classe el valor «card-absolute-link» i extreure'n el títol. Amb això ja podem programar la nostra aranya perquè n'extregui les dades d'interès.

L'estructura d'un *crawler* amb Scrapy ve prefixada. En el nostre cas, només serà necessari definir una aranya i incloure un *parser* que extregui les dades de les titulacions i que disposi de l'URL d'inici.

In [17]:
```
# Importem Scrapy.
import scrapy
from scrapy.crawler import CrawlerProcess

# Creem l'aranya.
class uoc_spider(scrapy.Spider):

    # Assignem un nom a l'aranya.
    name = "uoc_spider"

    # Indiquem l'URL que volem analitzar en primer lloc.
    start_urls = [
        "http://estudios.uoc.edu/es/grados"
    ]

    # Definim l'analitzador.
    def parse(self, response):
            # Extraiem el títol del grau.
        for grado in response.xpath('//a[@class="card-absolute-link"]/@title'):
            yield {
                'title': grado.extract()
            }
```

Un cop definida l'aranya, llançarem el *crawler* indicant que volem que usi l'aranya `uoc_spider` que acabem de definir:

In [18]:
```
if __name__ == "__main__":

    # Creem un crawler:
    process = CrawlerProcess({
        'USER_AGENT': 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)',
        'DOWNLOAD_HANDLERS': {'s3': None},
        'LOG_ENABLED': False
    })

    # Inicialitzem el crawler amb la nostra aranya:
    process.crawl(uoc_spider)

    # Llancem l'aranya:
    process.start()
```

```
INFO:scrapy.utils.log:Scrapy 1.0.3 started (bot: scrapybot)
INFO:scrapy.utils.log:Optional features available: ssl, http11, boto
INFO:scrapy.utils.log:Overridden settings: {'USER_AGENT': 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)', 'LOG_E
NABLED': False}
INFO:scrapy.middleware:Enabled extensions: CloseSpider, TelnetConsole, LogStats, CoreStats, SpiderState
INFO:scrapy.middleware:Enabled downloader middlewares: HttpAuthMiddleware, DownloadTimeoutMiddleware, UserAgentMiddlew
are, RetryMiddleware, DefaultHeadersMiddleware, MetaRefreshMiddleware, HttpCompressionMiddleware, RedirectMiddleware,
 CookiesMiddleware, ChunkedTransferMiddleware, DownloaderStats
INFO:scrapy.middleware:Enabled spider middlewares: HttpErrorMiddleware, OffsiteMiddleware, RefererMiddleware, UrlLengt
hMiddleware, DepthMiddleware
INFO:scrapy.middleware:Enabled item pipelines:
INFO:scrapy.core.engine:Spider opened
INFO:scrapy.extensions.logstats:Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
DEBUG:scrapy.telnet:Telnet console listening on 127.0.0.1:6023
DEBUG:scrapy.core.engine:Crawled (200) <GET http://estudios.uoc.edu/es/grados> (referer: None)
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Antropolog\xeda y Evoluci\xf3n Humana (interuniversitario: URV, UOC)'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Artes'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Ciencias Sociales'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Historia, Geograf\xeda e Historia del Arte (interuniversitario: UOC, UdL)'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Humanidades'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Lengua y Literatura Catalanas'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Traducci\xf3n, Interpretaci\xf3n y Lenguas Aplicadas (interuniversitario: UVic-UCC, UOC)'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Logopedia (interuniversitario: UVic-UCC, UOC)'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Comunicaci\xf3n'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Dise\xf1o y Creaci\xf3n Digitales'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Informaci\xf3n y Documentaci\xf3n'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Criminolog\xeda'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Derecho'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Gesti\xf3n y Administraci\xf3n P\xfablica (interuniversitario: UOC, UB)'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Relaciones Internacionales'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Artes'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Dise\xf1o y Creaci\xf3n Digitales'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Multimedia'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Administraci\xf3n y Direcci\xf3n de Empresas'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Doble titulaci\xf3n de Administraci\xf3n y Direcci\xf3n de Empresas y de Turismo'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Econom\xeda'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Marketing e Investigaci\xf3n de Mercados'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Relaciones Laborales y Ocupaci\xf3n'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Doble titulaci\xf3n de Ingenier\xeda Inform\xe1tica y de Administraci\xf3n y Direcci\xf3n de Empresas'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Ingenier\xeda Inform\xe1tica'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Multimedia'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Tecnolog\xedas de la Telecomunicaci\xf3n'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Educaci\xf3n Social'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Psicolog\xeda'}
DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados>
{'title': u'Turismo'}
INFO:scrapy.core.engine:Closing spider (finished)
INFO:scrapy.statscollectors:Dumping Scrapy stats:
{'downloader/request_bytes': 240,
 'downloader/request_count': 1,
 'downloader/request_method_count/GET': 1,
 'downloader/response_bytes': 220188,
 'downloader/response_count': 1,
 'downloader/response_status_count/200': 1,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2017, 10, 16, 14, 57, 0, 116758),
 'item_scraped_count': 30,
 'log_count/DEBUG': 32,
 'log_count/INFO': 7,
 'response_received_count': 1,
 'scheduler/dequeued': 1,
 'scheduler/dequeued/memory': 1,
 'scheduler/enqueued': 1,
 'scheduler/enqueued/memory': 1,
 'start_time': datetime.datetime(2017, 10, 16, 14, 56, 58, 784725)}
```

```
INFO:scrapy.core.engine:Spider closed (finished)
```

L'execució de Scrapy mostra un registre detallat amb tots els esdeveniments que han anat passant, fet que és molt útil per identificar problemes, sobretot en captures complexes. En el nostre cas, a més, podem veure com s'han extret els noms de les titulacions de grau:

DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados> {'title': u'Antropolog\xeda y Evoluci\xf3n Humana (interuniversitario: URV, UOC)'} DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados> {'title': u'Ciencias Sociales'} DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados> {'title': u'Historia, Geograf\xeda e Historia del Arte (interuniversitario: UOC, UdL)'} DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados> {'title': u'Humanidades'} DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados> {'title': u'Lengua y Literatura Catalanas'} DEBUG:scrapy.core.scraper:Scraped from <200 http://estudios.uoc.edu/es/grados> {'title': u'Traducci\xf3n, Interpretaci\xf3n y Lenguas Aplicadas (interuniversitario: UVic-UCC, UOC)'}