prog_datasci_5_api_entrega

November 21, 2019

1 Programación para *Data Science*

1.1 Unidad 5: Adquisición de datos en Python

En este Notebook encontraréis dos conjuntos de ejercicios: un primer conjunto de **ejercicios para practicar** y un segundo conjunto de **actividades evaluables** como PEC de la asignatura.

En cuanto al conjunto de ejercicios para practicar, éstos no puntúan para la PEC, pero os recomendamos que los intentéis resolver como parte del proceso de aprendizaje. Encontraréis ejemplos de posibles soluciones a los ejercicios al propio notebook, pero es importante que intentéis resolverlos vosotros antes de consultar las soluciones. Las soluciones os permitirán validar vuestras respuestas, así como ver alternativas de resolución de las actividades. También os animamos a preguntar cualquier duda que surja sobre la resolución de los **ejercicios para practicar** en el foro del aula.

Además, veréis que todas las actividades tienen una etiqueta que indica los recursos necesarios para llevarla a cabo. Hay tres posibles etiquetas:

- NM **Sólo materiales**: las herramientas necesarias para realizar la actividad se pueden encontrar en los materiales de la asignatura.
- EG **Consulta externa guiada**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, pero el enunciado contiene indicaciones de dónde o cómo encontrar la información adicional necesaria para resolver la actividad.
- EI **Consulta externa independente**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, y el enunciado puede no incluir la descripción de dónde o cómo encontrar esta información adicional. Será necesario que el estudiante busque esta información utilizando los recursos que se han explicado en la asignatura.

Es importante notar que estas etiquetas no indican el nivel de dificultad del ejercicio, sino únicamente la necesidad de consulta de documentación externa para su resolución. Además, recordad que las **etiquetas son informativas**, pero podréis consultar referencias externas en cualquier momento (aunque no se indique explícitamente) o puede ser que podáis hacer una actividad sin consultar ningún tipo de documentación. Por ejemplo, para resolver una actividad que sólo requiera los materiales de la asignatura, puedéis consultar referencias externas si queréis, ya sea tanto para ayudaros en la resolución como para ampliar el conocimiento!

En cuanto a la consulta de documentación externa en la resolución de los ejercicios, recordad **citar siempre la bibliografía utilizada** para resolver cada actividad.

1.2 Ejercicios para practicar

Los siguientes 3 ejercicios no puntúan para la PEC, pero os recomendamos que los intentéis resolver antes de pasar a los ejercicios propios de la PEC. También encontraréis las soluciones a estos ejercicios al final del Notebook.

1.2.1 Ejercicio 1

Queremos saber los crímenes que se han producido en Reino Unido en una localización (latitud, longitud) y fecha concretas. Identificad qué métodos de la API siguiente podemos utilizar para obtener la información y contestad a las siguientes preguntas.NM

1. £A qué URL haremos la petición?
2. £Qué tipo de petición HTTP (qué acción) deberemos realizar contra la API para obtener los datos deseados?
3. £En qué formato obtendremos la respuesta de la API?
4. £Qué parámetros deberemos proporcionar en la petición a la API?

Respuesta

[ ]:

1.2.2 Ejercicio 2

Programad una función que retorne el estado meteorológico actual en una cierta localización, definida por su código postal (**zip code**) y código de país (e.g: us, uk, es, fr, etc). La función debe devolver una lista de tuplas de dos elementos, correspondientes al resumen del estado actual del tiempo **(weather.main)** y a la descripción extendida **(weather.description)**. Utilizad la API de openweathermap para obtener las predicciones.

Para utilizar la API necesitareis registraros y obtener una API key. Podéis registraros aquí y obtener vuestra API key aquí una vez registrados. Tened en cuenta que la API key puede tardar un rato en funcionar después de registraros, y la API os devolverá un error 401 conforme la clave no es valida:

```
{"cod":401, "message": "Invalid API key. Please see http://openweathermap.org/faq#error401 for more info."}
```

Simplemente esperad un rato antes de utilizar la clave.NM

Hints:

- Veréis que en general la API esta documentada sin incluir la API key, aun que esta es necesaria. Deberéis incluir la API key en la llamada como uno de los parámetros de la URL (&appid=your_api_key):

  http://example_url.com?param1=value1&param2=value2&appid=your_api_key

- Os animamos a que paséis por el proceso de registro para que veáis de que trata y cómo se generan las API keys. Aún así, os proporcionamos una API key en caso de que tengáis problemas con el proceso.

  owm_api_key = 'd54f26dbcf6d4136bc0ef8ba5f07825b'

Respuesta

[ ]:

1.2.3 Ejercicio 3

CoinMarketCap es una web con contenido acerca de las 100 criptomonedas con más capitalización de mercado. Programad un *crawler* que extraiga los nombres y la capitalización de todas les monedas que se muestran en CoinMarketCap. Utilizad la estructura de *crawler* que hemos visto en el Notebook de esta unidad **modificando únicamente dos lineas de código**:EG - URL de inicio. - La expresión XPath que selecciona el contenido a capturar.

Pista: tal vez os puede ser de utilidad investigar sobre la scrapy shell y utilizarla para encontrar la expresión XPath que necesitas para resolver el ejercicio.

Nota: si la ejecución del *crawler* os devuelve un error `ReactorNotRestartable`, reiniciad el núcleo del Notebook (en el meú: `Kernel` - `Restart`).

Respuesta

[ ]:

1.3 Ejercicios y preguntas teóricas para la PEC

A continuación, encontraréis los **ejercicios y preguntas teóricas que debéis completar en esta PEC** y que forman parte de la evaluación de esta unidad.

1.3.1 Ejercicio 1

La librería Tweepy nos permite interactuar con la API de Twitter de una forma sencilla. Utilizando la librería Tweepy, recuperad la descripción, la fecha de creación y localización de vuestra cuenta de Twitter. Si lo preferís, podéis obtener dicha información de la cuenta del usuario de Twitter de la Python Software Foundation, `ThePSF`(en vez de utilizar vuestra cuenta). **(1 punto)** NM

Nota: Necesitáis las claves ***Consumer API keys y Access token & acces token secret***. Para obtener las claves, seguid las indicaciones que encontraréis en el Notebook de esta unidad. Podéis utilizar el código presente en el Notebook, adaptándolo para resolver el ejercicio.

Respuesta

[ ]:

1.3.2 Ejercicio 2

Implementad un conjunto de funciones para obtener la **secuencia de ADN** del organismo *Homo sapiens* del cromosoma 1 (**chr1**) desde la posición 100000 hasta 101000 para la referencia **hg19**. Para realizar el ejercicio utilizad la API de UCSC. **(1.5 puntos)** NM

Nota: El genoma de referencia de una célula es un repositorio de secuencias de ADN (ácido desoxirribonucleico) empaquetado en forma de cromosoma. El ADN es un ácido nucleico que contiene la información genética que dirige el desarrollo y el funcionamiento de todos los seres vivos. El ADN se puede entender como una secuencia de nucleótidos (A, C, T y G) de una determinada longitud. Este material hereditario codifica los genes que, una vez descifrados, son indispensables para la síntesis de las proteínas.

Un genoma de referencia es la representación de la secuencia de ADN del genoma de una especie. En el caso del organismo *Homo sapiens*, existen diferentes versiones del genoma de referencia. La última versión, hg38, se publicó en el 2014 y es la más detallada y precisa.

UCSC es un navegador de la Universidad de Santa Cruz de California que ofrece acceso a secuencias genómicas y su correspondiente anotación (genes, mRNAs, CpG,...) de una gran variedad de organismos, vertebrados e invertebrados.

Referencia: Genómica Computacional. Enrique Blanco. Barcelona, Universitat Oberta de Catalunya, 2011.

Importante: No es necesario entender toda la información que podéis obtener a través de la API de UCSC. Fijaros bien con lo que os pide el enunciado (prestad atención a la palabras clave en negrita), y revisad los ejemplos de acceso a los datos que hay en la web de UCSC.

Respuesta

[ ]:

1.3.3 Ejercicio 3

Dada la API de UCSC del ejercicio anterior, obtened la longitud del chr1 del organismo *Homo sapiens* según la versión del genoma de referencia hg19. Calculad la diferencia entre la longitud del cromosoma chr1 entre las versiones hg19 y hg18. **(1.5 puntos)** NM

Respuesta

[ ]:

1.3.4 Ejercicio 4

La NASA mediante su API publica cada día una imagen de astronomía. Implementad una función para descargar y visualizar la imagen dentro del notebook. **(2 puntos)** NM

Respuesta

[ ]:

1.3.5 Ejercicio 5

Scimago Journal es una web para consultar la información de las principales revistas de la comunidad científica. Programad un crawler que devuelva una tupla con el código y la área de todas las revistas que se muestran en la web. Utilizad la estructura de crawler que hemos visto en el Notebook de esta unidad modificando únicamente dos líneas de código:

- URL de inicio.
- La expresión XPath que selecciona el contenido a capturar.

Nota: si la ejecución del *crawler* os devuelve un error `ReactorNotRestartable`, reiniciad el núcleo del Notebook (en el menú: `Kernel` - `Restart`). **(2 puntos)** EG

Respuesta

[ ]:

1.3.6 Ejercicio 6

Queremos conocer la Agenda de actos de la Anella Olímpica de la ciudad de Barcelona. Imprimid por pantalla el nombre del grupo o cantante que celebrará un concierto en la Anella Olímpica durante el año 2020. Para realizar el ejercicio, consultad el portal de datos abiertos del Ayuntamiento de Barcelona mediante la siguiente url. Primero tenéis que identificar qué método utilizar para descargar los datos. Seguidamente, descargad los datos y procesarlos para responder la pregunta. **(2 puntos)** EG

Respuesta

[ ]:

2 Ejercicio Opcional

Modificad la función del ejercicio 5 para obtener las 10 primeras revistas asociadas en la área Computer Science del 2017.EI

Respuesta

[ ]:

2.1 Soluciones ejercicios para practicar

2.1.1 Ejercicio 1

Queremos saber los crímenes que se han producido en Reino Unido en una localización (latitud, longitud) y fecha concretas. Identificad qué métodos de la API siguiente podemos utilizar para obtener la información y contestad a las siguientes preguntas.NM

1. £A qué URL haremos la petición?
2. £Qué tipo de petición HTTP (qué acción) deberemos realizar contra la API para obtener los datos deseados?
3. £En qué formato obtendremos la respuesta de la API?
4. £Qué parámetros deberemos proporcionar en la petición a la API?

Respuesta 1. https://data.police.uk/docs/method/crimes-at-location/ 2. Tenemos que realizar una petición tipo GET 3. La respuesta la obtendremos en formato JSON 4. Tenemos que proporcionar la fecha (date), latitud (lat) y longitud (lng)

2.1.2 Ejercicio 2

Programad una función que retorne el estado meteorológico actual en una cierta localización, definida por su código postal (**zip code**) y código de país (e.g: us, uk, es, fr, etc). La función debe devolver una lista de tuplas de dos elementos, correspondientes al resumen del estado actual del tiempo **(weather.main)** y a la descripción extendida **(weather.description)**. Utilizad la API de openweathermap para obtener las predicciones.NM

Para utilizar la API necesitareis registraros y obtener una API key. Podéis registraros aquí y obtener vuestra API key aquí una vez registrados. Tened en cuenta que la API key puede tardar

un rato en funcionar después de registraros, y la API os devolverá un error 401 conforme la clave no es valida:

`{"cod":401, "message": "Invalid API key. Please see http://openweathermap.org/faq#error401 for more info."}`

Simplemente esperad un rato antes de utilizar la clave.

Hints:

- Veréis que en general la API esta documentada sin incluir la API key, aun que esta es necesaria. Deberéis incluir la API key en la llamada como uno de los parámetros de la URL (&appid=your_api_key):

  http://example_url.com?param1=value1&param2=value2&appid=your_api_key

- Os animamos a que paséis por el proceso de registro para que veáis de que trata y cómo se generan las API keys. Aún así, os proporcionamos una API key en caso de que tengáis problemas con el proceso.

  owm_api_key = 'd54f26dbcf6d4136bc0ef8ba5f07825b'

Respuesta

Lo primero que haremos será revisar la API de openweathermap para identificar qué endpoints nos pueden ser útiles. El enunciado nos pide devolver el estado meteorológico actual dado un código postal, podemos utilizar https://openweathermap.org/current.

Existe un método que nos devuelve el estado meteorológico a partir del código postal y el código del país separado por coma:

api.openweathermap.org/data/2.5/weather?zip=zip_code,country_code

[3]:
```
import json
import requests
def parse_response(response):
    data = None
    if response.status_code == 200:
        # Data is formatted as JSON but received as string. Load it as JSON␣
 ↪object
        data = json.loads(response.content)

    # Raise an error otherwise
    else:
        raise Exception("Unexpected response (%s: %s)." %(response.status_code,␣
 ↪response.reason))

    return data


def get_weather_zip(zip_code, country, api_key):
    # Query the data from the API
    base_url = 'http://api.openweathermap.org/data/2.5/weather?
 ↪zip=%s,%s&appid=%s'

    # We also add the API KEY to the request
    response = requests.get(base_url % (zip_code, country, api_key))
```

```
    # Check the response code and act accordingly
    data = parse_response(response)

    # If the data was properly processed
    if data:
        weather = data.get('weather')
        r = [(w.get('main'), w.get('description')) for w in weather]
    else:
        raise Exception("Couldn't get weather data.")

    return r

api_key = '169af185292dd6119b14bc20d23400fb'
zip_code = '08018'
country_code = 'es'
weather_data = get_weather_zip(zip_code, country_code, api_key)

print (weather_data)
```

```
[('Clouds', 'scattered clouds')]
```

2.1.3 Ejercicio 3

CoinMarketCap es una web con contenido acerca de las 100 criptomonedas con más capitalización de mercado.Programad un *crawler* que extraiga los nombres y la capitalización de todas les monedas que se muestran en CoinMarketCap. Para hacerlo, utilizad la estructura de *crawler* que hemos visto en el Notebook de esta unidad **modificando únicamente dos lineas de código**:

- URL de inicio.
- La expresión XPath que selecciona el contenido a capturar.

Pista: tal vez os puede ser de utilidad investigar sobre la scrapy shell y utilizarla para encontrar la expresión XPath que necesitas para resolver el ejercicio.

Nota: si la ejecución del *crawler* os devuelve un error `ReactorNotRestartable`, reiniciad el núcleo del Notebook (en el menú: `Kernel` - `Restart`).NM

Respuesta

[1]:
```
import scrapy
from scrapy.crawler import CrawlerProcess

# Creamos la araña.
class uoc_spider(scrapy.Spider):

    # asignamos un nombre a la araña
    name = "uoc_spider"

    # Indicamos la URL que queremos analizar.
    # Incluimos aquí la URL de inicio:
```

```
    ################################################
    start_urls = [
        "https://coinmarketcap.com/"
    ]
    ################################################

    # Definimos el analizador.
    def parse(self, response):
        # Extraer el nombre de la moneda
        # Incluir la expresión 'xpath' que nos devuelve los nombres de las␣
 ↪monedas.
    ################################################
        for currency in response.xpath('//td[@class="no-wrap currency-name"]/
 ↪@data-sort'):
        ################################################
            yield {
                'currency': currency.extract()
            }
if __name__ == "__main__":

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

    # Inicializamos el crawler con la muestra de la araña.
    process.crawl(uoc_spider)

    # Lanzamos la aranya.
    process.start()
```

```
2019-11-16 13:49:08 [scrapy.utils.log] INFO: Scrapy 1.7.4 started (bot:
scrapybot)
2019-11-16 13:49:08 [scrapy.utils.log] INFO: Versions: lxml 4.4.1.0, libxml2
2.9.9, cssselect 1.1.0, parsel 1.5.2, w3lib 1.21.0, Twisted 19.7.0, Python 3.7.3
(default, Mar 27 2019, 09:23:15) - [Clang 10.0.1 (clang-1001.0.46.3)], pyOpenSSL
19.0.0 (OpenSSL 1.1.1d  10 Sep 2019), cryptography 2.8, Platform
Darwin-19.0.0-x86_64-i386-64bit
2019-11-16 13:49:08 [scrapy.crawler] INFO: Overridden settings: {'USER_AGENT':
'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)'}
2019-11-16 13:49:08 [scrapy.extensions.telnet] INFO: Telnet Password:
3f21dc90b528ae78
2019-11-16 13:49:08 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
```

```
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.memusage.MemoryUsage',
 'scrapy.extensions.logstats.LogStats']
2019-11-16 13:49:08 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2019-11-16 13:49:08 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2019-11-16 13:49:08 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2019-11-16 13:49:08 [scrapy.core.engine] INFO: Spider opened
2019-11-16 13:49:08 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0
pages/min), scraped 0 items (at 0 items/min)
2019-11-16 13:49:08 [scrapy.extensions.telnet] INFO: Telnet console listening on
127.0.0.1:6023
2019-11-16 13:49:08 [scrapy.core.engine] DEBUG: Crawled (200) <GET
https://coinmarketcap.com/> (referer: None)
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Bitcoin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Ethereum'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'XRP'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Bitcoin Cash'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Tether'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Litecoin'}
```

```
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'EOS'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Binance Coin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Bitcoin SV'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Stellar'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'TRON'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Cardano'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Monero'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Chainlink'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'UNUS SED LEO'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Huobi Token'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'NEO'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Tezos'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Cosmos'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'IOTA'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Dash'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Maker'}
```

```
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Ethereum Classic'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Ontology'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'USD Coin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Crypto.com Coin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'VeChain'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'NEM'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Basic Attention Token'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Dogecoin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Zcash'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Decred'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Paxos Standard'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Qtum'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'HedgeTrade'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'TrueUSD'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': '0x'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Centrality'}
```

```
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Holo'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'OmiseGO'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Bitcoin Gold'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'V Systems'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Ravencoin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Synthetix Network Token'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'ZB'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'LUNA'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Nano'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Augur'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'ABBC Coin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Algorand'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Komodo'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Bytom'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Dai'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'EDUCare'}
```

```
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'KuCoin Shares'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Lisk'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Silverway'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Bitcoin Diamond'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'BitTorrent'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'DigiByte'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Siacoin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Seele'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Quant'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'ICON'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'THETA'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'FTX Token'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'IOST'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Swipe'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Verge'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Waves'}
```

```
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'HyperCash'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Bytecoin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'DxChain Token'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'BitShares'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'MonaCoin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'MCO'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Aeternity'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Nexo'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'MaidSafeCoin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'iExec RLC'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Zilliqa'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'BitMax Token'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Ardor'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Steem'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Chiliz'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Aurora'}
```

```
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Enjin Coin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Energi'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'aelf'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'RIF Token'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Horizen'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Ren'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Golem'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Status'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'UNI COIN'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Newton'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'ILCoin'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Crypterium'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'Pundi X'}
2019-11-16 13:49:09 [scrapy.core.scraper] DEBUG: Scraped from <200
https://coinmarketcap.com/>
{'currency': 'GXChain'}
2019-11-16 13:49:09 [scrapy.core.engine] INFO: Closing spider (finished)
2019-11-16 13:49:09 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 232,
 'downloader/request_count': 1,
 'downloader/request_method_count/GET': 1,
 'downloader/response_bytes': 54284,
```

```
 'downloader/response_count': 1,
 'downloader/response_status_count/200': 1,
 'elapsed_time_seconds': 0.543706,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2019, 11, 16, 12, 49, 9, 341609),
 'item_scraped_count': 100,
 'log_count/DEBUG': 101,
 'log_count/INFO': 10,
 'memusage/max': 66461696,
 'memusage/startup': 66461696,
 'response_received_count': 1,
 'scheduler/dequeued': 1,
 'scheduler/dequeued/memory': 1,
 'scheduler/enqueued': 1,
 'scheduler/enqueued/memory': 1,
 'start_time': datetime.datetime(2019, 11, 16, 12, 49, 8, 797903)}
2019-11-16 13:49:09 [scrapy.core.engine] INFO: Spider closed (finished)
```