

Programación para *Data Science*

Unidad 5: Adquisición de datos en Python - Ejercicios y preguntas

Pregunta 1

Queremos saber los nombres de las personas hay en el espacio en un momento dado. Identificad qué método de la [siguiente API \(http://open-notify.org/Open-Notify-API\)](http://open-notify.org/Open-Notify-API) podemos utilizar para obtener este dato y contestad a las siguientes preguntas. **(1 punto)**

1. ¿A qué URL realizaremos la petición?
2. ¿Qué tipo de petición HTTP (qué acción) tendremos que realizar a la API para obtener los datos deseados?
3. ¿En qué formato obtendremos la respuesta de la API seleccionada?
4. ¿En qué campo de la respuesta encontraremos la información que buscamos?

Respuesta

- 1.
- 2.
- 3.
- 4.

Pregunta 2

Buscad información sobre el estándar OAuth 2.0. Explicad una situación concreta donde sería útil usar OAuth en vez de un protocolo de autenticación tradicional basado en usuario y contraseña. **(1 punto)**

Respuesta

Ejercicio 1

Seleccionad y descargad cualquier fichero de datos de alguno de los portales comentados en el Notebook de esta unidad.

Cargad los datos en una variable Python (podéis usar [pandas \(http://pandas.pydata.org/\)](http://pandas.pydata.org/) si queréis). Mostrad un mensaje que informe del número de muestras que contiene el conjunto de datos y los atributos disponibles para cada muestra. **(1 punto)**

```
In [1]: # Respuesta
```

Ejercicio 2

Implementad una función que devuelva el primer precio del día de cambio de bitcoins (BTC) a dólares (USD) en Bitstamp. **(1.5 puntos)**

Pista: [Aquí \(https://www.bitstamp.net/api/\)](https://www.bitstamp.net/api/) encontraréis la documentación de la API de Bistamp.

```
In [2]: # Respuesta
```

Ejercicio 3

Modificad la función del ejercicio anterior para que devuelva el primer precio del día de cambio de bitcoins (BTC) a euros (EUR). **(0.5 puntos)**

```
In [ ]: # Respuesta
```

Ejercicio 4

Programad una función que devuelva la fecha y hora de los 15 próximos pases de la estación espacial internacional (ISS) por encima de una localización concreta especificada por su **dirección**. La función debe devolver una lista de 15 elementos, cada uno de los cuales debe ser una cadena de caracteres con la fecha y hora de los pases. **(2.5 puntos)**

Pista: ¡pensad que podéis combinar resultados de varias API!

```
In [4]: # Respuesta
```

Ejercicio 5

[KDNuggets \(http://www.kdnuggets.com/\)](http://www.kdnuggets.com/) es una web con contenido sobre *data science*. Entre otros, la web publica tutoriales relacionados con la ciencia de los datos.

Programad un crawler que extraiga los títulos de todos los tutoriales y *overviews* que publicaron el mes de Agosto de este año en KDNuggets. **(2.5 puntos)**

Para hacerlo, utilizad la estructura de crawler que hemos visto en el notebook de esta unidad, **modificando únicamente dos líneas de código:**

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

Pista: Quizás os puede ser de utilidad investigar sobre la *scrapy shell* y usarla para encontrar la expresión XPath que necesitáis para resolver el ejercicio.

Nota: Si la ejecución del crawler os devuelve un error `ReactorNotRestartable`, reiniciad el kernel del notebook (en el menú, `Kernel - Restart`).

```
In [ ]: # Importamos scrapy.
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 en primer lugar
    # Includ aquí la URL de inicio:
    #####
    start_urls = [
        ""
    ]
    #####

    # Definimos el analizador.
    def parse(self, response):
        # Extraemos el título del grado.
        # Includ aquí la expresión 'xpath' que nos devuelve las etiquetas de las ofertas de trabajo.
        #####
        for t in response.xpath(''):
            #####
            yield {
                'title': t.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': False
    })

    # Inicializamos el crawler con nuestra araña.
    process.crawl(uoc_spider)

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

Ejercicio opcional

Programad una función que calcule si la estación espacial internacional (ISS) se encuentra actualmente encima de una posición geográfica concreta. La función recibirá como parámetros la longitud y latitud de la posición geográfica, y un radio de tolerancia en kilómetros (que será un parámetro opcional con valor por defecto 10 kilómetros). La función devolverá un booleano, True o False, indicando si la ISS se encuentra actualmente encima de la posición que recibe como parámetro .

```
In [ ]: # Respuesta
```