prog_datasci_5_api_entrega

April 20, 2021

0.1 Unidad 5: Adquisición de datos en Python

1 Fundamentos de Programación

En este Notebook se encontraréis el conjunto de actividades evaluables como PEC de la asignatura. Veréis que cada una de ellas tiene asociada una puntuación, que indica el peso que tiene la actividad sobre la nota final de la PEC. Adicionalmente, hay un ejercicio opcional, que no tiene puntuación dentro de la PEC, pero que se valora al final del semestre de cara a conceder las matrículas de honor y redondear las notas finales. Podréis sacar la máxima nota de la PEC sin necesidad de hacer este ejercicio. El objetivo de este ejercicio es que sirva como pequeño reto para los estudiantes que quieran profundizar en el contenido de la asignatura.

Veréis que todas las actividades de la PEC 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 independiente**: 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.

1.0.1 Ejercicio 1

Hemos visto el uso de la librería Requests para realizar peticiones a la Web API de manera manual.

Mediante esta librería podemos realizar solicitudes como en el ejemplo que hemos visto de post-codes.io.

Hemos visto que, al hacer una petición a una web o API, recuperamos un objeto que contiene, entre otros, los siguientes atributos: `status.code`,`content`, `headers`.

Implementa una función de forma que acepte una *url* y devuelva el atributo `content` en caso de que el código de error sea correcto, en caso de que no sea correcto la función debe devolver el valor `None`. En todos los casos la función debe imprimir por pantalla el resultado de la llamada a función con un texto de descripción por los códigos 200, 301, 400, 401, 403, 404, 500 y 501. Deberás buscar la información estos códigos de error http de `status.code`.

Por ejemplo, por el código de retorno 200, el texto que ha de imprimir por pantalla la función es `Código 200: Correcto`.

Prueba la función con las siguientes URLs:

- http://google.com
- http://wikipedia.org
- https://mikemai.net/
- http://google.com/idontexist

Almacena los resultados (no los códigos) de las llamadas en una lista de cuatro elementos.

(1 punto) EI

[ ]:
```
# Respuesta
```

1.0.2 Ejercicio 2

Para cada una de las siguientes direcciones:

1. http://www.ismycomputeron.com
2. https://crouton.net
3. http://hasthelargehadroncolliderdestroyedtheworldyet.com
4. https://api.fbi.gov/wanted/v1/list

LLama a cada dirección con `query.get()`, y muestra:

- El código de `status.code`.
- El contenido de la dirección web, limitando la salida hasta 2500 caracteres

Accede a cada una de las direcciones con tu ordenador y observa el contenido en el navegador con el texto de la web. ¿Cuál es la diferencia en el contenido de la última dirección respecto a las tres primeras?

(1.5 puntos) NM

[ ]:
```
# Respuesta
```

1.0.3 Ejercicio 3

En este ejercicio haremos un poco de *Fun with numbers*. Hay una API para *number-facts* (curiosidades sobre números) en la base de datos http://numbersapi.com. Esta API tiene unos puntos de acceso en los que podemos interrogar la API sobre diferentes curiosidades sobre números enteros y fechas:

Dado un número N, o una fecha con día D y más M, la llamada a las siguientes direcciones proporcionan la siguiente información:

Call | Description |

|: ———: |: ——————————— ———————————————— ———— | | Http: //numbersapi.com/N | Trivia about N | | Http: //numbersapi.com/N/math | Mathematical facts about N | | Http: //numbersapi.com/M/D/date | Facts about a particular date | | Http: //numbersapi.com/N/year | Facts about a particular date |

Así, para obtener el *fact* número 1, debemos construir una solicitud a la *url*:

- *http: //numbersapi.com/1*

El objeto que se nos devolverá, contendrá la información indicada en la tabla en formato de texto directo.

a) Construye una función que tenga de argumento dos año AI y AF. La función debe recoger los *facts* desde el año AI hasta el año AF ambos incluidos, y devolver un diccionari donde las claves sean el año y los valores sean el *fact* sobre ese año.

(1 punto) NM

```
[ ]: # Respuesta
```

b) Usa esta función para mostrar por pantalla los *facts* entre 1900 y 1930, ambos incluidos.

(0.5 puntos) NM

```
[ ]: # Respuesta
```

1.0.4 Ejercicio 4

Una de tacos.

El proyecto TacoFancy API está construyendo una base de datos de acceso público para ordenar las recetas de Tacos. Estas recetas son accesibles mediante una API a http://taco-randomizer.herokuapp.com/ (si accede a la dirección, tendrá una receta de taco aleatoria, gratis).

Esta API construye un taco a partir de cinco elementos: *base_layer*, *mixin*, *condiment*, *seasoning* i *shell*.

La API del proyecto acepta diferentes entradas, a partir de la raíz anterior, para acceder a la información de cada elemento y su receta, y también a las recetas de los tacos a partir de sus diferentes ingredientes. Estas son:

- *base_layers/* por las bases del Taco
- *mixins/* para rellenar el Taco

- *condiments/* para las recetas de diferentes condimentos
- *seasonings/* por las diferentes salsas
- *shells/* por la *tortilla*

Así, por ejemplo, http://taco-randomizer.herokuapp.com/shells/ devolverá el listado de las posibles recetas de *tortillas*. Verá, por ejemplo, que el *slug* de la *French Corn Tortillas* es `fresh_corn_tortillas`. Esto le dice que la receta de esta Tortilla en particular está a http://taco-randomizer.herokuapp.com/shells/fresh_corn_tortillas. Cada componente tiene su *slug*, y por tanto, para construir la receta de una taco a partir de sus ingredientes, podemos construir una *url* como la siguiente:

- http://taco-randomizer.herokuapp.com/<mixin>/<condiment>/<seasoning>/<shell>/

Así, si encuentran los *slugs* correspondientes, podemos encontrar la receta final de un taco construyendo una *url* a partir de los *slugs*, como esta:

- http://taco-randomizer.herokuapp.com/bulgar_black_bean_filling/sweet_potato_and_apple_hash/mang

a) Construye una función que devuelva un diccionario que contenga en las llaves (keys) los nombres de cada *seasoning* y los valores (values) la *slug* para cada condimento. Muestra el diccionario por pantalla.

Nota: Se denomina usualmente un slug como una parte de una url, o un nombre corto de una url. El término proviene del mundo editorial y se usa extensivamente en internet. Puedes buscar más información en https://en.wikipedia.org/wiki/Slug_(publishing)

(1 punto) EG

[ ]:
```
# Respuesta
```

b) Construye un diccionario, de forma que contenga por cada uno de los 5 elementos como*keys*: *base_layer*, *mixin*, *condiment*, *seasoning* i *shell*, un subdiccionario que contenga en las *keys* los nombres de cada ingrediente y en los *values* la *slug* de cada ingrediente. Imprimir el diccionario por pantalla. El resultado debe ser del tipo:

```
{'base_layers': {'Soyrizo': 'soyrizo',   'Roasted Butternut Squash':
'roasted_butternut_squash',   ...   'Pepper Tempeh': 'pepper_tempeh'},  'mixins':
{'Sweet Potato and Apple Hash': 'sweet_potato_and_apple_hash',   'Potato
Hash': 'potato_hash',   ...   ...   'Homemade Sriracha': 'homemade_sriracha'},
'shells': {'Fresh Corn Tortillas': 'fresh_corn_tortillas',  ...   'bad-ass
tortillas': 'bad_ass_tortillas'}}
```

(1 punto) EG

Nota: Recuerda que has creado una función de apoyo en el ejercicio anterior que te será útil en este apartado.

[ ]:
```
# Respuesta
```

c) Construye un objeto dataframe de pandas, con todas las posibles combinaciones de tacos que sean posibles con los ingredientes de la base de datos. Este objeto debe tener 6 columnas, *base_layer*, *mixin*, *condiment*, *seasoning*, *shell*, que contendrán el nombre de cada ingrediente, y una última con la *url* de la receta del taco. Imprimir las últimas 10 entradas.

(1.5 puntos) NM

```
[ ]: # Respuesta
```

d) Elige una url al azar del DataFrame, imprime su *url*, puedes comprobar la receta del taco abriéndola en un navegador. ¿Cuántas recetas únicas son posibles?

Nota: Puedes generar un número al azar con el módulo random https://docs.python.org/3/library/random.html

(0.5 puntos) NM

```
[ ]: # Respuesta
```

1.0.5 Ejercicio 5

En los ejercicios anteriores, hemos utilizado directamente una API para hacer solicitudes a servicios en línea, y nos encargamos directamente de la gestión de los datos de salida.

Sin embargo, también hemos visto en los materiales del curso el uso de librerías que facilitan el acceso a una API, como *tweepy*. La mayoría de estas librerías (y la mayoría de APIs de proyectos populares) requieren un registro en una web de desarrolladores o API Key.

En este ejercicio os proponemos el uso de geopy https://geopy.readthedocs.io,

Usa la librería **geopy** para programar dos funciones.

- La primera función debe tener de entrada un string que contenga una dirección, y de salida devolver una tupla con la geolocalización de esta dirección (Latitud, Longitud)
- La segunda función, aceptará de entrada una tupla con latitud y longitud, y aplicará un método de geolocalización reverso para devolver una string con la dirección postal inferida.

Ejecuta la primera función con la dirección postal "Av. Del Tibidabo, 39, 08035 Barcelona" para encontrar su latitud y longitud, luego ejecuta la segunda función para intentar inferir la dirección postal original. Imprime los dos resultados.

(1 punto) EG

```
[ ]: # Resultado
```

1.0.6 Ejercicio 6

Https://meteo.cat/observacions/llistat-xema contiene información de una Red de Estaciones Meteorológicas Automáticas (XEMA) de área catalana. En esta página se puede encontrar el listado de estaciones meteorológicas en área de Cataluña incluyendo información de municipio, comarca, estación, localización, altitud, fechas de alta y baja, y estado actual.

Use `scrappy` para mostrar la información del listado por pantalla de las comarcas en la que está cada estación listada, tanto si está activa o no.

Como por ejemplo:

{ 'Num': 117, 'comarca': 'Solsonès'}

Por ello:

- Utilice el tutorial de Scrappy para encontrar un `xpath` que contenga la información requerida
- Muestre la información requerida en forma de diccionario.

(1 punto) EI

1.0.7 Ejercicio opcional

Consulte la pagína web de Open Notify, que contiene la información tanto sobre los humanos residentes fuera de la tierra (es decir, en el espacio) como de la posición de la estación internacional Espacial. [Open Notify] (http://api.open-notify.org).

Programe una función que imprima por pantalla cada 10 segundos, la longitud y latitud, y la dirección postal de la posición sobre la que está volando la ISS. Ejecute esta función 100 veces con un periodo entre llamadas de 10 segundos.

Puede comprobar su solución con http://www.isstracker.com.

EI

[ ]:
```
# Respuesta
```