AT05.02- JSON en Python

En esta actividad veremos cómo emplear la librería "json" para trabajar en Python con objetos de tipo JSON.
Lo primero que tendremos que hacer será importar la librería json, que contiene las funciones necesarias para trabajar con documentos JSON:

```
CMD>pip install json
```

Tal como hemos visto, en Python los ficheros se leen como strings, mientras que la mejor forma de trabajar con objetos json es empleando diccionarios y listas. La librería json básicamente nos provee de herramientas que nos permiten convertir el contenido de un fichero a un diccionario o una lista, y convertir listas y diccionarios a strings para poder guardarlos en ficheros.
Para ello usaremos las siguientes funciones:

- **json.loads**(string): permite convertir un string en un diccionario
- **json.load**('fichero.json'): permite guardar directamente los datos de un fichero en un diccionario
- **json.dump**(diccionario): convierte un diccionario en un string
- **json.dumps**(diccinario,indent,sort_keys): permite volcar un diccionario en un fichero de forma que se vea estructurado (de forma similar a cómo hacíamos en XML con "pretty_print"). En indent le decimos cuantos espacios debe usar para indentar cada elemento, y sort_keys nos permitiría mostrar los datos ordenados alfabéticamente por clave (por defecto es false, se mostrarán tal cual están en el diccionario)

1. Probamos la lectura desde un fichero. Cuando vamos a leer datos de un fichero json podemos encontrarnos 2 posibilidades: que el fichero contenga una lista de objetos JSON, o que el fichero contenga un solo objeto JSON. La forma de leerla será la misma, pero tendremos que tener cuidado con el tipo de fichero de que se trate porque la variable donde volquemos su contenido podrá ser una lista (si son varios objetos) o un diccionario (si es uno solo), y por tanto la tendremos que tratar de forma diferente en función de qué tipo sea.
   a. Creamos un fichero "lista.json" con una lista de personas:

```
[{"nombre": "Roberto", "idiomas": ["inglés", "francés"]},
{"nombre": "Margarita", "idiomas": ["rumano", "chino"]},
{"nombre": "Berta","edad": 12,"hijos": "None"}]
```

   b. Creamos un programa que cargue el fichero anterior y muestre por pantalla determinada información del fichero. En este caso la información se carga en una variable llamada "datos" que será una lista de python, por lo que para ver su contenido lo mostraríamos con un bucle for:

```
import json

# cargamos el fichero en una variable llamada "datos"
with open('lista.json') as f:
  datos = json.load(f)

for persona in datos:
  print(persona) # mostramos su contenido
```

   c. Creamos otro fichero "diccionario.json" con un solo objeto "contactos":

```
{"contactos":
  [ {"nombre": "Roberto", "idiomas": ["inglés", "francés"]},
    {"nombre": "Margarita", "idiomas": ["rumano", "chino"]},
    {"nombre": "Berta","edad": 12,"hijos": "None"}
  ]
```

```
}
```

d. En este caso lo que cargamos es un solo objeto, por lo que podemos usarlo directamente:

```
import json

# cargamos el fichero en una variable llamada "datos"
with open('diccionario.json') as f:
  datos = json.load(f)

print(datos) # mostramos su contenido
```

e. Si en lugar de usar información de un fichero tuviésemos el objeto json en un string, podríamos cargarlo usando la función "loads".

```
import json

doc = '{"nombre": "Margarita", "idiomas": ["rumano", "chino"]}'#está entre
comillas, por lo que es un string no un diccionario
datos = json.loads(doc)
print (datos) # muestra todo el documento
print (datos["nombre"])# muestra el nombre
print (datos["idiomas"]) # muestra solo los idiomas
```

2. La información de los diccionarios y las listas la podemos modificar usando las funciones de listas y diccionarios vistas anteriormente. Vamos a crear una una lista y un diccionario desde cero y a hacer varias modificaciones sobre ellos:

```
import json

lista=[]
objeto1 = {}
objeto1['nombre']='Berta'
objeto1['edad']=12
objeto1['hijos']= None
objeto2 ={}
objeto2['nombre']='Alberto'
objeto2['edad']=35
objeto2['hijos']= 2
lista.append(objeto1)
lista.append(objeto2)
print(lista)
objeto1['nombre']=objeto2.get('nombre')
lista.remove(objeto2)
print(lista)
objeto3={"nombre": "Bartolomé", "familia":{"padre":"Ramón", "madre":"Jimena"}}
print(objeto3.get("familia").get("padre"))
lista.append(objeto3)
print(lista)
lista.pop(0)
print(lista)
```

3. Y por último, podemos guardar el objeto o la lista en un nuevo fichero json. Para ello, los tendremos que convertir previamente en un string, usando json.dump:

```
import json

diccionario = {}
diccionario['nombre']='Berta'
diccionario['edad']=12
diccionario['hijos']= None

with open('persona2.json', 'w') as fichero: #lo abrimos con 'w' para escritura
  json.dump(diccionario, fichero)
```

```
#para que se vea bien el contenido, le aplicamos primero json.dumps, que lo
convierte en un string con formato:
with open('persona3.json', 'w') as fichero:
  json.dump(json.dumps(diccionario,indent=4,sort_keys=True),fichero)
```

EJERCICIOS

1. Crea un documento "libreria.json" con el siguiente contenido:

```
{"libreria": {
  "libro": [
    {"titulo": {"idioma": "en","texto": "Everyday Italian"},"autor": "Giada De
Laurentiis","anho": "2005","precio": "30.00","categoria": "COCINA"},
    {"titulo": {"idioma": "es","texto": "Harry Potter"},"autor": "J K. Rowling","anho":
"2005","precio": "29.99","categoria": "INFANTIL"},
    {"titulo": {"idioma": "en","texto": "XQuery Kick Start"},"autor": ["James
McGovern","Per Bothner","Kurt Cagle","James Linn","Vaidyanathan Nagarajan"],"anho":
"2003","precio": "49.99","categoria": "WEB"},
    {"titulo": {"idioma": "es","texto": "Aprendiendo XML"},"autor": "Erik T.
Ray","anho": "2003","precio": "39.95","categoria": "WEB"}
]}}
```

   Escribe un programa en python que lea el fichero "libreria.json" y realice las siguientes operaciones:
   a. Carga el contenido del fichero en una variable. ¿Qué tipo de variable será necesaria?
   b. Imprime el número de libros que contiene la librería
   c. Muestra los idiomas en los que está escrito cada libro.
   d. Solicita que el usuario indique por teclado un límite inferior y superior para el precio y muestra todos los libros cuyo precio esté en ese intervalo. Para solicitar un valor por pantalla:

```
limite_inf = input("Indica el límite inferior: ")
```

   e. Pide una cadena por teclado, y muestra el título y el año de publicación de los libros cuyo título empiece por la cadena introducida.
   f. Muestra todos los títulos de los libros con la lista de sus autores.

2. Escribe un programa python que lea el contenido del fichero "AT05.02.02-pruebas.json" y lo guarde en una variable. ¿Qué tipo de variable necesitamos en este caso?.

   Una vez cargado el fichero, muestra la siguiente información:
   a. ¿Cuantas pruebas de idiomas están descritas en el documento?
   b. Muestra el título de las pruebas de nivel que van a durar más de dos horas.
   c. De las pruebas de tipo "No Presencial" muestra su URL de información. Ten en cuenta que puede hacer pruebas que no tengan indicada una URL.
   d. Pide por teclado el código de una prueba (su ID) y muestra su título y profesores.
   e. Para cada una de las pruebas, muestra su título y sus profesores.

3. Utilizando el fichero "AT05.02.03-poblaciones.json" de provincias y municipios de España, muestra la siguiente información:
   a. Listado de todas las provincias.
   b. Listado de todos los municipios.
   c. Muestra para cada provincia, el número de municipios que tiene
   d. Pide por teclado el nombre de una provincia y muestra sus municipios.
   e. Pide por teclado el nombre de un municipio y muestra la provincia donde se encuentra.

4. A partir del siguiente programa Python que lee de internet datos de unos alumnos que tienen que realizar una tarea:

```
import requests
```

```
import json

# guardamos en una variable los copiados datos directamente de una página web
res = requests.get("https://jsonplaceholder.typicode.com/todos")
dic = json.loads(res.text) # y la volcamos a un diccionario

completados=[]

for usuario in dic:
  if usuario["completed"]:
    completados.append(usuario)
```

a. Muestra el número de alumnos y el número de alumnos que han completado la tarea
b. Muestra el id de los alumnos que han completado la tarea
c. Guarda en un fichero llamado “alumnos.json” la información de todos los alumnos, y en “completados.json” los ids de los alumnos que han completado la tarea.

PROCESAMIENTO DE TWEETS EN PYTHON

En esta actividad vamos a comprobar la utilización en un caso real de Python para procesar documentos JSON. Lo que haremos será monitorizar en tiempo real los tweets que se estén publicando en twitter, y veremos qué podemos hacer con ellos.
Para ello, tendremos que crear en primer lugar una cuenta de twitter, que usaremos para poder capturar la información (en formato JSON), y el lenguaje de programación Python para procesar el fichero JSON capturado.

1. Creamos la cuenta de twitter
   a. Acceder a la página https://apps.twitter.com/ y pulsar el botón de "Create New App"
   b. Llenar los campos y aceptar las condiciones de uso.
   c. Una vez creada la APP, acceder a la pestaña “Permissions” y cambiar los permisos a "Read, Write and Access direct messages"
   d. Accedemos a la pestaña "Keys and Access Tokens" y presionamos sobre "Create My Access Token". Esto generará unas claves que emplearemos posteriormente para loguearnos desde nuestros programas.
2. Instalamos la librería “tweepy” de Python, que contiene las funcionalidades necesarias para trabajar con twitter:

```
CMD>pip install tweepy
```

3. Capturamos los tweets. Para capturar los tweets[1] que se publiquen en tiempo real, usaremos el siguiente código en Python:

```
import tweepy

#Sustituye los asteriscos por tus claves...
CONSUMER_KEY = '********************'
CONSUMER_SECRET = '********************'
ACCESS_KEY = '********************'
ACCESS_SECRET = '********************'

#Creamos una clase que se encargará de escribir los tweets en el fichero
class MyListener(tweepy.StreamListener):
  def on_data(self,data):
    try:
      with open('Tweets.json','a') as f:
        f.write(data)
        return True
    except BaseException as e:
```

[1] Documentación de la API de twitter: https://developer.twitter.com/en/docs/api-reference-index

```
      print("Error en el dato: %s" % str(e))
    return True

  def on_error(self,status):
    print(status)
    return(True)

#Nos logueamos con las claves definidas anteriormente
auth=tweepy.OAuthHandler(CONSUMER_KEY,CONSUMER_SECRET)
auth.set_access_token(ACCESS_KEY,ACCESS_SECRET)
api=tweepy.API(auth)

#Y capturamos los tweets que se compartan en tiempo real
twitter_stream=tweepy.Stream(auth,MyListener())
twitter_stream.filter(track=['twitter'])
```

Este programa utiliza la API de twitter "tweepy" para loguearse en el sistema y capturar todos los tweets que se estén publicando en tiempo real que contengan la palabra "twitter", guardándolos en el fichero "tweets.json". Si comentásemos la última línea, capturaría todos los tweets que se publiquen.

En las claves consumer_key, consumer_secret, access_token y access_secret, cada uno tendrá que indicar los valores de su cuenta de twitter generados en el apartado anterior.

Copiamos el código del programa y los guardamos en un fichero al que llamaremos "captura_tweets.python", y lo ejecutamos desde la línea de comandos con la sentencia:

```
python captura_tweets.python
```

4. Procesamos los datos. Ya tenemos los tweets guardados en un fichero JSON. Para trabajar con los datos, usaremos de nuevo python: cargaremos la información del fichero en una lista de tweets, y luego recorreremos dicha lista para mostrar la información que queramos de cada tweet. Los principales campos de cada tweet son los siguientes:
   - text: texto del tweet
   - created_at: fecha de creación
   - favorite_count, retweet_count: número de likes y retweets
   - favorited, retweeted: Boolean que indica si tu usuario lo ha marcado como like o retweeteado
   - lang: lenguaje
   - id: identificador del tweet
   - place, coordinates, geo: geo-localización, si está disponible
   - user: usuario autor del tweet. El campo user es a su vez un objeto que contiene otros campos: id, name, screen_name, followers_count, friends_count, etc.
   - entities: lista de entidades como URLs, hashtags, menciones (@) o símbolos
   - in_reply_to_user_id: indica si es una contestación a un tweet de otro usuario
   - in_reply_to_status_id: identificador del tweet al que se contesta

   a. Crea un fichero "procesa_tweets.python" y copia en él el siguiente contenido:

```
import json

#creamos una lista vacia, donde guardaremos los tweets
tweets=list()
#abrimos el fichero
with open ('Tweets.json', 'r') as fichero:
  #volcamos el fichero linea a línea volcando su contenido en un string
  for linea in fichero.readlines():
    if not linea.strip (): # omitimos las lineas vacias
      continue
```

```
    #convertimos la linea en un objeto JSON
    objeto_json = json.loads (linea)
    #y lo cargamos en la lista
    tweets.append(objeto_json)

#recorremos los 5 primeros elementos de la lista (del 0 al 5) y mostramos su
contenido
for tweet in tweets[0:5]:
  print('Creado en:', tweet['created_at'])
  print('Texto:', tweet['text'])
  print('Usuario:', tweet['user']['name'])
  print('')
#mostramos el lenguaje del elemento numero 20 de la lista
print('Lenguaje: ',tweets[20]['lang']
#mostramos el identificador de los ultimos 10 tweets de la lista y si fue o no
retweeteado
for tweet in tweets[:5]:
  print('Id:', tweet['id'],' retweeteado:',tweet['retweet_count'])
```

b. Muestra la siguiente información de los tweets capturados:

- Número de tweets capturados
- id y si ha sido retweeteado o no de los 10 primeros elementos.
- id y hashtags de los elementos entre el 5 y el 20.
- Número de seguidores y número de amigos del elemento número 50 de la lista.
- Crea un nuevo diccionario vacío(dic={}). Guarda en dicho diccionario el número de tweets que ha publicado cada usuario (es decir, la clave serán los distintos nombres de los usuarios y en el valor tendrás que contar los tweets publicados por el usuario en cuestión). Para ello tendrás que recorrer toda la lista de tweets, y para cada tweet incrementar en 1 el número de tweets de ese usuario (si no tuviese ningún tweet, le tendrías que asignar el valor 1).