P.03 - Requests + NBA

El objetivo de la práctica es aprender a parsear respuestas de peticiones HTTP con Python. En este caso particular, se trata de parsear un diccionario.

Instalación

Para esta práctica solo necesitas tener instalado Requests y Python.

Descargar datos

- Abre un navegador - el que más rabia te dé -, copia en él la siguiente URL y dale al INTRO para cargar la página:

  https://stats.nba.com/stats/playercareerstats?LeagueID=00&PerMode=PerGame&PlayerID=977

- ¿Pasa algo raro? ¿A qué crees que se debe?

- Abre Postman, copia la misma URL y haz una petición GET

  > 💡 **¡OJO AL DATO!**
  > Fíjate como Postman automáticamente detecta los `query strings` y los mete como parámetros de la petición.

- ¿Y? ¿Pasa algo? ¿Te reafirmas en lo que crees que pasaba cuando no funcionaba la petición en navegador o crees que ocurre algo distinto?

- Abre consola (o tu IDE) y copia y ejecuta el siguiente código.

  Si no puedes ver el “User-Agent” completo en el PDF es este: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36”

```
import requests

url = "https://stats.nba.com/stats/playercareerstats"
params = {
    "LeagueID": "00",
    "PerMode": "PerGame",
    "PlayerID": 977
}
headers = {
    "Accept": "application/json, text/plain, */*",
    "Accept-Encoding": "gzip, deflate, br",
    "Accept-Language": "en-US,en;q=0.5",
    "Connection": "keep-alive",
    "DNT": "1",
    "Host": "stats.nba.com",
    "Origin": "https://www.nba.com",
    "Referer": "https://www.nba.com/",
    "TE": "Trailers",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/53
}
response = requests.get(url, headers=headers, params=params)
print(response.status_code)
print(response.json())
```

Deberías recibir una respuesta (en consola) similar a esta:

```
>>> response = requests.get(url, headers=headers, params=params)
>>> print(response.status_code)
200
>>> print(response.json())
{'resource': 'playercareerstats', 'parameters': {'PerMode': 'PerGame', 'PlayerID': 977, 'LeagueID': '00'}, 'resultSets': [{'name': 'SeasonTotalsRegularS
eason', 'headers': ['PLAYER_ID', 'SEASON_ID', 'LEAGUE_ID', 'TEAM_ID', 'TEAM_ABBREVIATION', 'PLAYER_AGE', 'GP', 'GS', 'MIN', 'FGM', 'FGA', 'FG_PCT', 'FG3
M', 'FG3A', 'FG3_PCT', 'FTM', 'FTA', 'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS'], 'rowSet': [[977, '1996-97', '00', 161061
2747, 'LAL', 18.0, 71, 6, 15.5, 2.5, 5.9, 0.417, 0.7, 1.9, 0.375, 1.9, 2.3, 0.819, 0.7, 1.2, 1.9, 1.3, 0.7, 0.3, 1.6, 1.4, 7.6], [977, '1997-98', '00',
1610612747, 'LAL', 19.0, 79, 1, 26.0, 4.9, 11.6, 0.428, 0.9, 2.8, 0.341, 4.6, 5.8, 0.794, 1.0, 2.1, 3.1, 2.5, 0.9, 0.5, 2.0, 2.3, 15.4], [977, '1998-99'
, '00', 1610612747, 'LAL', 20.0, 50, 50, 37.9, 7.2, 15.6, 0.465, 0.5, 2.0, 0.267, 4.9, 5.8, 0.839, 1.1, 4.2, 5.3, 3.8, 1.4, 1.0, 3.1, 3.1, 19.9], [977,
'1999-00', '00', 1610612747, 'LAL', 21.0, 66, 62, 38.2, 8.4, 17.9, 0.468, 0.7, 2.2, 0.319, 5.0, 6.1, 0.821, 1.6, 4.7, 6.3, 4.9, 1.6, 0.9, 2.8, 3.3, 22.5
], [977, '2000-01', '00', 1610612747, 'LAL', 22.0, 68, 68, 41.0, 10.3, 22.2, 0.464, 0.9, 2.9, 0.305, 7.0, 8.2, 0.853, 1.5, 4.3, 5.9, 5.0, 1.7, 0.6, 3.2,
 3.3, 28.5], [977, '2001-02', '00', 1610612747, 'LAL', 23.0, 80, 80, 38.3, 9.4, 20.0, 0.469, 0.4, 1.7, 0.25, 6.1, 7.4, 0.829, 1.4, 4.1, 5.5, 5.5, 1.5, 0
.4, 2.8, 2.9, 25.2], [977, '2002-03', '00', 1610612747, 'LAL', 24.0, 82, 82, 41.5, 10.6, 23.5, 0.451, 1.5, 4.0, 0.383, 7.3, 8.7, 0.843, 1.3, 5.6, 6.9, 5
.9, 2.2, 0.8, 3.5, 2.7, 30.0], [977, '2003-04', '00', 1610612747, 'LAL', 25.0, 65, 64, 37.7, 7.9, 18.1, 0.438, 1.1, 3.3, 0.327, 7.0, 8.2, 0.852, 1.6, 3.
9, 5.5, 5.1, 1.7, 0.4, 2.6, 2.7, 24.0], [977, '2004-05', '00', 1610612747, 'LAL', 26.0, 66, 66, 40.7, 8.7, 20.1, 0.433, 2.0, 5.9, 0.339, 8.2, 10.1, 0.81
6, 1.4, 4.5, 5.9, 6.0, 1.3, 0.8, 4.1, 2.6, 27.6], [977, '2005-06', '00', 1610612747, 'LAL', 27.0, 80, 80, 41.0, 12.2, 27.2, 0.45, 2.3, 6.5, 0.347, 8.7,
10.2, 0.85, 0.9, 4.4, 5.3, 4.5, 1.8, 0.4, 3.1, 2.9, 35.4], [977, '2006-07', '00', 1610612747, 'LAL', 28.0, 77, 77, 40.8, 10.6, 22.8, 0.463, 1.8, 5.2, 0.
344, 8.7, 10.0, 0.868, 1.0, 4.7, 5.7, 5.4, 1.4, 0.5, 3.3, 2.7, 31.6], [977, '2007-08', '00', 1610612747, 'LAL', 29.0, 82, 82, 38.9, 9.5, 20.6, 0.459, 1.
8, 5.1, 0.361, 7.6, 9.0, 0.84, 1.1, 5.2, 6.3, 5.4, 1.8, 0.5, 3.1, 2.8, 28.3], [977, '2008-09', '00', 1610612747, 'LAL', 30.0, 82, 82, 36.1, 9.8, 20.9, 0
.467, 1.4, 4.1, 0.351, 5.9, 6.9, 0.856, 1.1, 4.1, 5.2, 4.9, 1.5, 0.5, 2.6, 2.3, 26.8], [977, '2009-10', '00', 1610612747, 'LAL', 31.0, 73, 73, 38.8, 9.8
, 21.5, 0.456, 1.4, 4.1, 0.329, 6.0, 7.4, 0.811, 1.1, 4.3, 5.4, 5.0, 1.5, 0.3, 3.2, 2.6, 27.0], [977, '2010-11', '00', 1610612747, 'LAL', 32.0, 82, 82,
33.9, 9.0, 20.0, 0.451, 1.4, 4.3, 0.323, 5.9, 7.1, 0.828, 1.0, 4.1, 5.1, 4.7, 1.2, 0.1, 3.0, 2.1, 25.3], [977, '2011-12', '00', 1610612747, 'LAL', 33.0,
 58, 58, 38.5, 9.9, 23.0, 0.43, 1.5, 4.9, 0.303, 6.6, 7.8, 0.845, 1.1, 4.3, 5.4, 4.6, 1.2, 0.3, 3.5, 1.8, 27.9], [977, '2012-13', '00', 1610612747, 'LAL
', 34.0, 78, 78, 38.6, 9.5, 20.4, 0.463, 1.7, 5.2, 0.324, 6.7, 8.0, 0.839, 0.8, 4.7, 5.6, 6.0, 1.4, 0.3, 3.7, 2.2, 27.3], [977, '2013-14', '00', 1610612
747, 'LAL', 35.0, 6, 6, 29.5, 5.2, 12.2, 0.425, 0.5, 2.7, 0.188, 3.0, 3.5, 0.857, 0.3, 4.0, 4.3, 6.3, 1.2, 0.2, 5.7, 1.5, 13.8], [977, '2014-15', '00',
1610612747, 'LAL', 36.0, 35, 35, 34.5, 7.6, 20.4, 0.373, 1.5, 5.3, 0.293, 5.6, 6.9, 0.813, 0.7, 4.9, 5.7, 5.6, 1.3, 0.2, 3.7, 1.9, 22.3], [977, '2015-16
', '00', 1610612747, 'LAL', 37.0, 66, 66, 28.2, 6.0, 16.9, 0.358, 2.0, 7.1, 0.285, 3.5, 4.3, 0.826, 0.6, 3.1, 3.7, 2.8, 0.9, 0.2, 2.0, 1.7, 17.6]]}, {'n
ame': 'CareerTotalsRegularSeason', 'headers': ['PLAYER_ID', 'LEAGUE_ID', 'Team_ID', 'GP', 'GS', 'MIN', 'FGM', 'FGA', 'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT'
, 'FTM', 'FTA', 'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS'], 'rowSet': [[977, '00', 0, 1346, 1198, 36.1, 8.7, 19.5, 0.447,
 1.4, 4.1, 0.329, 6.2, 7.4, 0.837, 1.1, 4.1, 5.2, 4.7, 1.4, 0.5, 3.0, 2.5, 25.0]]}, {'name': 'SeasonTotalsPostSeason', 'headers': ['PLAYER_ID', 'SEASON_
ID', 'LEAGUE_ID', 'TEAM_ID', 'TEAM_ABBREVIATION', 'PLAYER_AGE', 'GP', 'GS', 'MIN', 'FGM', 'FGA', 'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT', 'FTM', 'FTA', 'FT_
PCT', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS'], 'rowSet': [[977, '1996-97', '00', 1610612747, 'LAL', 18.0, 9, 0, 14.8, 2.3, 6.1,
0.382, 0.7, 2.6, 0.261, 2.9, 3.3, 0.867, 0.1, 1.1, 1.2, 1.2, 0.3, 0.2, 1.6, 2.6, 8.2], [977, '1997-98', '00', 1610612747, 'LAL', 19.0, 11, 0, 20.0, 2.8,
 6.9, 0.408, 0.3, 1.3, 0.214, 2.8, 4.1, 0.689, 0.6, 1.3, 1.9, 1.5, 0.3, 0.7, 1.0, 2.5, 8.7], [977, '1998-99', '00', 1610612747, 'LAL', 20.0, 8, 8, 39.4,
 7.6, 17.8, 0.43, 1.0, 2.9, 0.348, 3.5, 4.4, 0.8, 1.6, 5.3, 6.9, 4.6, 1.9, 1.3, 3.9, 3.0, 19.8], [977, '1999-00', '00', 1610612747, 'LAL', 21.0, 22, 22,
 39.0, 7.9, 17.9, 0.442, 1.0, 2.9, 0.344, 4.3, 5.7, 0.754, 1.2, 3.3, 4.5, 4.4, 1.5, 1.5, 2.5, 4.0, 21.1], [977, '2000-01', '00', 1610612747, 'LAL', 22.0
, 16, 16, 43.4, 10.5, 22.4, 0.469, 0.7, 2.1, 0.324, 7.8, 9.4, 0.821, 1.8, 5.4, 7.3, 6.1, 1.6, 0.8, 3.2, 3.3, 29.4], [977, '2001-02', '00', 1610612747, '
LAL', 23.0, 19, 19, 43.8, 9.8, 22.7, 0.434, 1.2, 3.1, 0.379, 5.8, 7.6, 0.759, 1.5, 4.4, 5.8, 4.6, 1.4, 0.9, 2.8, 3.4, 26.6], [977, '2002-03', '00', 1610
612747, 'LAL', 24.0, 12, 12, 44.3, 11.4, 26.4, 0.432, 2.1, 5.2, 0.403, 7.2, 8.7, 0.827, 1.3, 3.8, 5.1, 5.2, 1.2, 0.1, 3.5, 2.9, 32.1], [977, '2003-04',
'00', 1610612747, 'LAL', 25.0, 22, 22, 44.2, 8.6, 20.9, 0.413, 1.1, 4.4, 0.247, 6.1, 7.5, 0.813, 0.8, 3.9, 4.7, 5.5, 1.9, 0.3, 2.8, 2.7, 24.5], [977, '2
005-06', '00', 1610612747, 'LAL', 27.0, 7, 7, 44.9, 10.3, 20.7, 0.497, 2.0, 5.0, 0.4, 5.3, 6.9, 0.771, 0.6, 5.7, 6.3, 5.1, 1.1, 0.4, 4.7, 3.6, 27.9], [9
```

- ¿Qué ocurre ahora? ¿Por qué ahora sí funciona y antes no? Cuando la URL es la misma en los 3 casos…

- Copia la respuesta a la petición (el diccionario ese enooorme) y pégalo en tu IDE. Guarda el archivo en tu ordenador local y ponle un nombre, por ejemplo: kobe.json.

  Ten en cuenta que el formato .json sólo admite dobles comillas. Si copias y pegas desde consola recuerda sustituirlas.

  Acabas de descargarte las estadísticas de toda la carrera baloncestística de Kobe Bryant. Incluyendo estadísticas desde el instituto hasta los Lakers, de la temporada regular, play-offs y all-stars.

  > 💡 **PARA AQUÉLLOS A LOS QUE NO OS GUSTA EL BASKET...**
  > Primero de todo, hacéroslo mirar.
  > Ahora en serio, te paso el significado de una serie de términos que no tienen por qué sonarte para que entiendas el contenido del archivo que te acabas de descargar:
  > - Season: una temporada, normalmente empieza en septiembre y acaba en junio con un ganador.
  > - Regular Season: la temporada regular, tiempo durante el cual todos los equipos juegan contra todos. Es decir, todos los equipos y jugadores (por malos que sean) juegan la "Regular Season" a menos que estén lesionados, claro está.
  > - Post Season: llegado mayo, los peores equipos quedan eliminados y los mejores juegan una ronda eliminatoria (play-off). Conforme van pasando las rondas se van eliminando equipos y terminan quedando solo 2, que luchan por el título en la final. Es decir, no todos los equipos juegan "Post Season" todos los años.

Práctica

- Crea un archivo de Python.

- Importa en él el archivo JSON anterior y guárdalo, como diccionario, en una variable con el siguiente código:

```
import json

def load_json(path):
    """
    Loads JSON file in given path into a dictionary that will
    be returned.

    Params
    -------
    path : str
        The path to the target JSON file.

    Return
    -------
    d : dic
        The dictionary with the JSON content loaded.
    """
    with open(path, "r") as input_file:
        d = json.load(input_file)
    return d

d = load_json("{{METE EL PATH A TU ARCHIVO AQUÍ}}")
```

- A continuación - usando Python, obviamente - explora el archivo para extraer las siguientes variables:
  - Crea una matriz Nx6, donde "N" es el número de temporadas para las que hay datos en Temporada Regular, y cuyas cabeceras sean las siguientes (te dejo la primera fila hecha por si sirve de ayuda): "AÑO DE LA TEMPORADA", "EDAD DEL JUGADOR", "PARTIDOS DISPUTADOS", "MEDIA DE PUNTOS", "MEDIA DE ASISTENCIAS", "MEDIA DE REBOTES".

  - Ayúdate de los glosarios de términos en la web de la nba para identificar la cabecera en el archivo descargado. Llama a la variable `regular_season_stats`:

    ```
    regular_season_stats = [
      ["AÑO DE LA TEMPORADA", "EDAD DEL JUGADOR", "PARTIDOS DISPUTADOS", "MEDIA DE PUNTOS", "MEDIA DE ASISTENCIAS", "MEDIA DE REBO
      ["1996-97", 18, 71, 7.6, 1.3, 1.9],
      ...
    ]
    ```

  - Usando la matriz recién calculada, obtén la temporada con mayor cantidad total de puntos anotados (¡ojo! no mayor media de puntos, sino mayor cantidad total de puntos) y la temporada con menor cantidad de puntos anotados. Llama a las variables `season_max_points` y `season_min_points`.

  - Usando la matriz recién calculada, obtén la media de puntos absoluta durante toda la carrera de Kobe Bryant en Temporada Regular y guárdala en la variable `avg_career_points`.

    > 💡 **¡OJO A LA ESTADÍSTICA!**
    > Ten en cuenta que no sirve con hacer una media de la columna "Media de Puntos Anotados", sino que tienes que multiplicar temporada a temporada ésta por el número de partidos disputados y después dividir por el total de Partidos Disputados a lo largo de la carrera.

  - Haz el mismo cálculo para obtener también la media absoluta - durante toda la carrera de Kobe Bryant en Temporada Regular - de rebotes y asistencias y guárdalas en las variables `avg_career_rebounds` y `avg_career_assists`.

- Bonus-track voluntario para nota.
  Usando Python, obviamente, crea un array al que vas a llamar `better_in_post_season` que tendrá tantos elementos como temporadas jugó Kobe en la NBA. Cada elemento podrá tomar 3 valores:

- `True`, si durante dicha temporada la media de puntos anotados *por partido* en Post Season fue superior a la media de puntos anotados *por partido* en Regular Season.
- `False`, si fue inferior.
- `"N/A"`, si en dicha temporada no disputó Post Season.

Si te sirve de ayuda para resolver la práctica pudes usar la funcionalidad `zip()` de Python, que os va a solucionar la vida y es interesante que aprendáis a usarla si no la conocéis (en Stack Overflow os enseñan cómo si estáis muy perdidos con la documentación oficial de Python).