NBA_Data_Scraping

December 12, 2022

```
[ ]: import requests
     import pandas as pd
     import time
```

```
[ ]: pd.set_option('display.max_columns', None)
```

```
[ ]: def make_request(endpoint, params=None, record_path=None, verbose=False):
         root = "https://www.balldontlie.io/api/v1/"
         response = requests.get(root + endpoint, params=params)
         if response.status_code != 200:
             print(response.status_code)
             return response
         if verbose: print("Success!")
         res = response.json()
         res = pd.json_normalize(res, record_path=record_path)
         return res
```

```
[ ]: game_data = make_request("games", params={"page":1, "per_page":100, "seasons":
       ↪[2017,2018,2019,2020,2021,2022]}, record_path="data", verbose=True)
```

```
Success!
```

```
[ ]: %%time
     for i in range(2,501):
         if i%10 == 0: print(i)
         time.sleep(1.2)
         new_data = make_request("games", params={"page":i, "per_page":100,␣
       ↪"seasons":[2017,2018,2019,2020,2021,2022]}, record_path="data")
         game_data = game_data.append(new_data)
```

```
10
20
30
40
50
60
70
80
```

```
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/urllib3/connectionpool.py in␣
 ↪_make_request(self, conn, method, url, timeout, chunked, **httplib_request_kw
    376             try:  # Python 2.7, use buffering of HTTP responses
--> 377                 httplib_response = conn.getresponse(buffering=True)
    378             except TypeError:  # Python 3

TypeError: getresponse() got an unexpected keyword argument 'buffering'

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
<timed exec> in <module>

<ipython-input-2-0a05eb41497a> in make_request(endpoint, params, record_path,␣
 ↪verbose)
      1 def make_request(endpoint, params=None, record_path=None, verbose=False):
      2     root = "https://www.balldontlie.io/api/v1/"
----> 3     response = requests.get(root + endpoint, params=params)
      4     if response.status_code != 200:
      5         print(response.status_code)

/usr/local/lib/python3.7/dist-packages/requests/api.py in get(url, params,␣
 ↪**kwargs)
     74
     75     kwargs.setdefault('allow_redirects', True)
---> 76     return request('get', url, params=params, **kwargs)
```

```
     77
     78

/usr/local/lib/python3.7/dist-packages/requests/api.py in request(method, url,␣
 ↪**kwargs)
     59     # cases, and look like a memory leak in others.
     60     with sessions.Session() as session:
---> 61         return session.request(method=method, url=url, **kwargs)
     62
     63

/usr/local/lib/python3.7/dist-packages/requests/sessions.py in request(self,␣
 ↪method, url, params, data, headers, cookies, files, auth, timeout,␣
 ↪allow_redirects, proxies, hooks, stream, verify, cert, json)
    528         }
    529         send_kwargs.update(settings)
--> 530         resp = self.send(prep, **send_kwargs)
    531
    532         return resp

/usr/local/lib/python3.7/dist-packages/requests/sessions.py in send(self,␣
 ↪request, **kwargs)
    641
    642         # Send the request
--> 643         r = adapter.send(request, **kwargs)
    644
    645         # Total elapsed time of the request (approximately)

/usr/local/lib/python3.7/dist-packages/requests/adapters.py in send(self,␣
 ↪request, stream, timeout, verify, cert, proxies)
    447                     decode_content=False,
    448                     retries=self.max_retries,
--> 449                     timeout=timeout
    450                 )
    451

/usr/local/lib/python3.7/dist-packages/urllib3/connectionpool.py in␣
 ↪urlopen(self, method, url, body, headers, retries, redirect, assert_same_host ␣
 ↪timeout, pool_timeout, release_conn, chunked, body_pos, **response_kw)
    598                                                   timeout=timeout_obj,
    599                                                   body=body,␣
 ↪headers=headers,
--> 600                                                   chunked=chunked)

    601
    602             # If we're going to release the connection in ``finally:``,␣
 ↪then
```

```
/usr/local/lib/python3.7/dist-packages/urllib3/connectionpool.py in␣
 ↪_make_request(self, conn, method, url, timeout, chunked, **httplib_request_kw
    378             except TypeError:  # Python 3
    379                 try:
--> 380                     httplib_response = conn.getresponse()
    381                 except Exception as e:
    382                     # Remove the TypeError from the exception chain in␣
 ↪Python 3;

/usr/lib/python3.7/http/client.py in getresponse(self)
   1371         try:
   1372             try:
-> 1373                 response.begin()
   1374             except ConnectionError:
   1375                 self.close()

/usr/lib/python3.7/http/client.py in begin(self)
    317         # read until we get a non-100 response
    318         while True:
--> 319             version, status, reason = self._read_status()
    320             if status != CONTINUE:
    321                 break

/usr/lib/python3.7/http/client.py in _read_status(self)
    278
    279     def _read_status(self):
--> 280         line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
    281         if len(line) > _MAXLINE:
    282             raise LineTooLong("status line")

/usr/lib/python3.7/socket.py in readinto(self, b)
    587         while True:
    588             try:
--> 589                 return self._sock.recv_into(b)
    590             except timeout:
    591                 self._timeout_occurred = True

/usr/lib/python3.7/ssl.py in recv_into(self, buffer, nbytes, flags)
   1069                   "non-zero flags not allowed in calls to recv_into() o␣
 ↪%s" %
   1070                   self.__class__)
-> 1071             return self.read(nbytes, buffer)
   1072         else:
   1073             return super().recv_into(buffer, nbytes, flags)

/usr/lib/python3.7/ssl.py in read(self, len, buffer)
    927         try:
    928             if buffer is not None:
```

```
--> 929                 return self._sslobj.read(len, buffer)
    930             else:
    931                 return self._sslobj.read(len)

KeyboardInterrupt:
```

[ ]:
```
game_data.set_index("id", inplace=True)
```

[ ]:
```
game_data.to_csv("/content/drive/MyDrive/Machine Learning NBA /Final Data/games.
 ↪csv")
```

[ ]:
```
# there are over 11000 pages!
all_stats_data_meta = make_request("stats", params={"page":1, "per_page":100},␣
 ↪record_path=None)

all_stats_data_meta
```

[ ]:
```
                                                 data  meta.total_pages  \
0  [{'id': 1069008, 'ast': 0, 'blk': 1, 'dreb': 2…             11931

   meta.current_page  meta.next_page  meta.per_page  meta.total_count
0                  1               2            100           1193044
```

[ ]:
```
stats_data = pd.DataFrame()

for i in range(1, 11931):
    # Print what page we're on every 10 pages to keep track of progress
    if i % 1000 == 0:
        print(i)

    # Make sure not to exceed 60 API requests per minute (balldontlie API is␣
 ↪free but limits request per minite)
    time.sleep(1.1)

    # Make the request and append to the dataframe
    new_data = make_request("stats", params={"page":i, "per_page":100},␣
 ↪record_path="data")
    stats_data = stats_data.append(new_data)
print("Done!")
```

```
1000
2000
3000
4000
5000
6000
7000
```

```
8000
9000
10000
11000
Done!
```

[ ]:
```
stats_data.set_index("id", inplace=True)

stats_data.to_csv("/content/drive/MyDrive/Machine Learning NBA /Final Data/
 ↪stats_raw.csv")
```