baseball$_{id}$

Jun 14, 2022

Contents:

1 Introduction 1

2 baseball_id 3

3 Indices and tables 7

Python Module Index 9

Index 11

CHAPTER 1

Introduction

Do you analyze baseball data from several sources and need to find a way to correlate it for the same player? This takes advantage of the data published at https://github.com/spilchen/baseball_id_db. With the data you can take a player from Yahoo! and cross reference with data from sites such as http://baseball-reference.com, http://fangraphs.com, http://www.seanlahman.com. This package gives you a way to access this map within Python. A set of APIs allow you to do lookup of players by a variety of different IDs.

CHAPTER 2

baseball_id

APIs to lookup into the baseball ID at https://github.com/spilchen/baseball_id_db

All of the from_* APIs return a DataFrame that has baseball player particulars for the IDs in the lookup. For each player that it finds it will a return row in a panda's DataFrame that has:

```
>>> Index(['mlb_id', 'mlb_name', 'mlb_pos', 'bats', 'throws', 'birth_year',
>>>        'bp_id', 'bref_id', 'bref_name', 'cbs_id', 'cbs_name', 'cbs_pos',
>>>        'espn_id', 'espn_name', 'espn_pos', 'fg_id', 'fg_name', 'fg_pos',
>>>        'lahman_id', 'nfbc_id', 'nfbc_name', 'nfbc_pos', 'retro_id',
>>>        'retro_name', 'debut', 'yahoo_id', 'yahoo_name', 'ottoneu_id',
>>>        'ottoneu_name', 'ottoneu_pos', 'rotowire_id', 'rotowire_name', 'rotowire_
↪pos'],
>>>       dtype='object')
```

class baseball_id.lookup.**Cache**(*source*)

Class that caches results of baseball ID lookups.

You can use this to cut down on network traffic if you intend to do many calls to the APIs. The first time we use an API, we'll cache the results for subsequent callers.

from_mlb_ids(*ids*)

Lookup given a list of mlb_ID's

Accepts a list of MLB IDs. It will return a DataFrame containing Series for players that match the MLB IDs passed in. If nothing was found an empty DataFrame is returned.

Parameters **ids** (*int list*) – mlb_ID's to lookup

Returns Players that match the given IDs. If none are found an empty DataFrame is returned.

Return type DataFrame

```
>>> In [1]: lookup.from_mlb_ids([430945, 607680, 669456])
>>> Out[1]:
>>>       mlb_id      mlb_name mlb_pos ... ottoneu_pos rotowire_id rotowire_
↪name  rotowire_pos
```

(continues on next page)

(continued from previous page)

```
>>> 39    430945    Adam Jones      CF ...          OF      8165.0    Adam␣
↪Jones            OF
>>> 1746  607680  Kevin Pillar      CF ...          OF     12678.0  Kevin␣
↪Pillar             OF
>>> 2545  669456  Shane Bieber       P ...          SP     14383.0  Shane␣
↪Bieber             P
>>>
>>> [3 rows x 33 columns]
```

from_yahoo_ids(*ids*)

Lookup given a list of Yahoo! IDs

Accepts a list of IDs from Yahoo! fantasy baseball. It will return a DataFrame containing Series for players that match the Yahoo! IDs. If nothing was found an empty DataFrame is returned.

Parameters **ids** (*int list*) – Yahoo! IDs to lookup

Returns Players that match the given IDs. If none are found an empty DataFrame is returned.

Return type DataFrame

```
>>> In [1]: c.from_yahoo_ids([10794, 9542, 7578])
>>> Out[1]:
>>>       mlb_id        mlb_name mlb_pos ... ottoneu_pos rotowire_id  ␣
↪rotowire_name  rotowire_pos
>>> 5     621345     A.J. Minter       P ...          RP     13889.0     A.J.␣
↪Minter             P
>>> 204   605151  Archie Bradley       P ...          RP     12131.0  Archie␣
↪Bradley             P
>>> 2340  448179       Rich Hill       P ...          SP      7965.0       ␣
↪Rich Hill             P
>>>
>>> [3 rows x 33 columns]
```

from_cbs_ids(*ids*)

Lookup given a list of CBS IDs

Accepts a list of IDs from CBS fantasy baseball. It will return a DataFrame containing Series for players that match the IDs. If nothing was found an empty DataFrame is returned.

Parameters **ids** (*int list*) – CBS IDs to lookup

Returns Players that match the given IDs. If none are found an empty DataFrame is returned.

Return type DataFrame

```
>>> In [1]: lookup.from_cbs_ids([1660162, 2507367])
>>> Out[1]:
>>>       mlb_id      mlb_name mlb_pos ... ottoneu_pos rotowire_id rotowire_
↪name  rotowire_pos
>>> 423   457763  Buster Posey       C ...        C/1B     10426.0  Buster␣
↪Posey              C
>>> 1657  665742     Juan Soto      LF ...          OF     13960.0     Juan␣
↪Soto            OF
>>>
>>> [2 rows x 33 columns]
```

from_espn_ids(*ids*)

Lookup given a list of ESPN IDs

Accepts a list of IDs from ESPN fantasy baseball. It will return a DataFrame containing Series for players that match the IDs. If nothing was found an empty DataFrame is returned.

Parameters **ids** (*int list*) – ESPN IDs to lookup

Returns Players that match the given IDs. If no Series are found an empty DataFrame is returned.

Return type DataFrame

```
>>> In [1]: c.from_espn_ids([29252])
>>> Out[1]:
>>>      mlb_id       mlb_name mlb_pos ... ottoneu_pos rotowire_id  rotowire_
↪name  rotowire_pos
>>> 836  451594  Dexter Fowler      RF ...          OF      8271.0  Dexter␣
↪Fowler            OF
>>>
>>> [1 rows x 33 columns]
```

from_fangraphs_ids(*ids*)

Lookup given a list of FanGraph IDs

Accepts a list of IDs as maintained at fangraphs.com. It will return a DataFrame containing Series for players that match the IDs. If nothing was found an empty DataFrame is returned.

Parameters **ids** (*int list*) – FanGraph IDs to lookup

Returns Players that match the given IDs. If no Series are found an empty DataFrame is returned.

Return type DataFrame

```
>>> In [35]: c.from_fangraphs_ids([19753, 19566])
>>> Out[35]:
>>>       mlb_id           mlb_name mlb_pos  ... rotowire_id      rotowire_
↪name rotowire_pos
>>> 1510  642528  Jonathan Loaisiga       P  ...     15256.0  Jonathan␣
↪Loaisiga            P
>>> 2133  663993          Nate Lowe      1B  ...         NaN               ␣
↪NaN          NaN
>>>
>>> [2 rows x 33 columns]
```

from_names(*names*, *filter_missing=None*)

Lookup given a list of player names

Accepts a list of player names. The names must match exactly. It will return a DataFrame containing tuples for each player name that matches.

An optional argument exists to filter based on names and those that have a missing field. See explanation of the filter_missing parameter.

Parameters

- **names** – Player names to lookup
- **filter_missing** (*str*) – Optional parameter that allows for additional filtering. Only players that have this field name missing (Nan) will be returned. This is useful for use with one of the other ID based lookup where the ID may not be in the database. For example, set this to 'yahoo_id' to lookup of a rookie who doesn't yet have a yahoo_id in the database.

Returns Players that match the given names. If no Series are found an empty DataFrame is returned.

Return type DataFrame

```
>>> In [28]: lk.from_names(['Khris Davis', 'Enrique Hernandez'])
>>> Out[28]:
>>>       mlb_id           mlb_name            mlb_pos ...  ottoneu_name ␣
↪ottoneu_pos  rotowire_id      rotowire_name rotowire_pos
>>>       966   571771  Enrique Hernandez      CF       ...  Enrique Hernandez␣
↪ 1B/2B/SS/OF      11139.0  Enrique Hernandez           SS
>>>       1753  501981        Khris Davis      LF       ...  Khristopher Davis␣
↪           OF      11664.0        Khris Davis           OF
>>>
>>> [2 rows x 33 columns]
```

class baseball_id.factory.**Factory**

Factory methods to construct the ID lookup cache object.

classmethod create()

Factory method to create a cache object from github/spilchen/baseball_id_db

This is called as part of package initialization and so can be refered to via the Lookup variable.

```
>>> from baseball_id import Lookup
>>> Lookup.from_yahoo_ids([10794, 9542, 7578])
```

classmethod create_fake()

Factory method to create a fake data source

This refers to a static data file that is in the current package. This function exists for testing purposes as it avoids network traffic to get the actual up-to-date ID mapping.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

baseball_id.lookup, 3

Index

B

baseball_id.lookup (*module*), 3

C

Cache (*class in baseball_id.lookup*), 3
create() (*baseball_id.factory.Factory class method*), 6
create_fake() (*baseball_id.factory.Factory class method*), 6

F

Factory (*class in baseball_id.factory*), 6
from_cbs_ids() (*baseball_id.lookup.Cache method*), 4
from_espn_ids() (*baseball_id.lookup.Cache method*), 4
from_fangraphs_ids() (*baseball_id.lookup.Cache method*), 5
from_mlb_ids() (*baseball_id.lookup.Cache method*), 3
from_names() (*baseball_id.lookup.Cache method*), 5
from_yahoo_ids() (*baseball_id.lookup.Cache method*), 4